LECTURE NOTES

ON

COMPUTER GRAPHICS

IV B. Tech I semester (JNTUH-R09)

Ms. S J SOWJANYA
Associate Professor

Ms. V DIVYAVANI
Assistant Professor

COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF AERONAUTICAL ENGINEERING

DUNDIGAL, HYDERABAD - 500 043

<u>**UNIT- 1**</u>

Overview of Computer Graphics

1.1 Application of Computer Graphics

Computer-Aided Design for engineering and architectural systems etc.
Objects maybe displayed in a wireframe outline form. Multi-window environment is also favored for producing various zooming scales and views.
Animations are useful for testing performance.

Presentation Graphics
To produce illustrations which summarize various kinds of data. Except 2D, 3D graphics are good tools for reporting more complex data.

Computer Art
Painting packages are available. With cordless, pressure-sensitive stylus, artists can produce electronic paintings which simulate different brush strokes, brush widths, and colors. Photorealistic techniques, morphing and animations are very useful in commercial art. For films, 24 frames per second are required. For video monitor, 30 frames per second are required.

Entertainment
Motion pictures, Music videos, and TV shows, Computer games

Education and Training
Training with computer-generated models of specialized systems such as the training of ship captains and aircraft pilots.

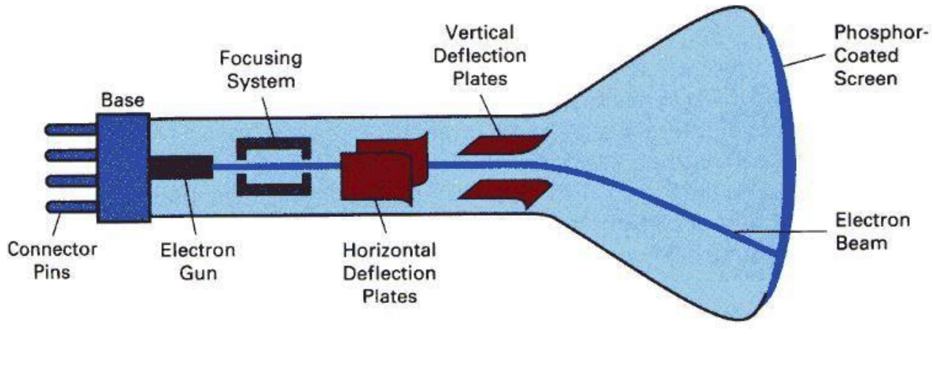
Visualization
For analyzing scientific, engineering, medical and business data or behavior. Converting data to visual form can help to understand mass volume of data very efficiently.

Image Processing
Image processing is to apply techniques to modify or interpret existing pictures. It is widely used in medical applications.

Graphical User Interface
Multiple window, icons, menus allow a computer setup to be utilized more efficiently.

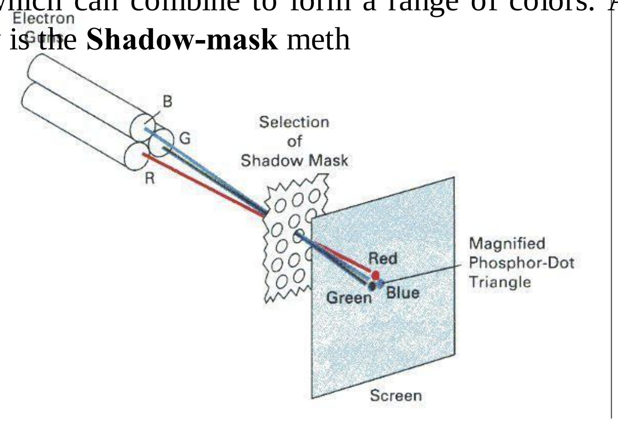
1.2 Video Display devices

1.2.1Cathode-Ray Tubes (CRT) - still the most common video display device presently



Electrostatic deflection of the electron beam in a CRT

An electron gun emits a beam of electrons, which passes through focusing and deflection systems and hits on the phosphor-coated screen. The number of points displayed on a CRT is referred to as r**esolutions** (eg. 1024x768). Different phosphors emit small light spots of different colors, which can combine to form a range of colors. A common methodology for color CRT display is the **Shadow-mask** meth



1.2.2 Raster-Scan

The electron beam is swept across the screen one row at a time from top to bottom. As it moves across each row, the beam intensity is turned on and off to create a pattern of illuminated spots. This scanning process is called refreshing.

The refreshing rate, called the **frame rate**, is normally 60 to 80 frames per second, or described as 60 Hz to 80 Hz.

Picture definition is stored in a memory area called the **frame buffer**. This frame buffer stores the intensity values for all the screen points. Each screen point is called a **pixel** (picture element).

On black and white systems, the frame buffer storing the values of the pixels is called a **bitmap**. Each entry in the bitmap is a 1-bit data which determine the on (1) and off (0) of the intensity of the pixel.

On color systems, the frame buffer storing the values of the pixels is called a **pixmap**

1.2.3 Random-Scan (Vector Display)

The CRT's electron beam is directed only to the parts of the screen where a picture is to be drawn. The picture definition is stored as a set of line-drawing commands in a refresh display file or a refresh buffer in memory.

Random-scan generally have higher resolution than raster systems and can produce smooth line drawings, however it cannot display realistic shaded scenes.

1.2.4 COLOR CRT MONITORS

CRT monitor displays color pictures by using a combination of phosphors that emit different colored light. By combining the emitted light from the different phosphors , a range of colors can be generated. Color CRTs have 3 phosphor color dots at each pixel position for red , green and blue color Three electron guns one for each color dot A metal *shadow mask* to differentiate the beams

The 2 basic techniques for producing color CRT displays are

1. Beam penetration method 2. Shadow mask method
Commonly used in raster scan systems because they produce wide range of colours than beam penetration method.

Beam Penetration method

Commonly used in raster scan systems
Two layers of phosphor, usually red and green, are coated onto the inside of CRT screen, and the displayed color depends on how far the electron beam penetrates into the phosphor layers.

A beam of slow electrons excites only the outer red layer.
A beam of very fast electrons penetrates through the red layer and excites the inner green layer.At intermediate beam speeds, combinations of other colors are produced.

SHADOW MASK

The **shadow mask** is one of two major technologies used to manufacture cathode ray tube (CRT) televisions and computer displays that produce color images (the other is aperture grille and its improved variant Cromaclear). Tiny holes in a metal plate separate the colored phosphors in the layer behind the front glass of the screen. The holes are placed in a manner ensuring that electrons from each of the tube's three cathode guns reach only the appropriately-colored phosphors on the display. All three beams pass through the same holes in the mask, but the angle of approach is different for each gun. The spacing of the holes, the spacing of the phosphors, and the placement of the guns is arranged so that for example the blue gun only has an unobstructed path to blue phosphors. The red, green, and blue phosphors for each pixel are generally arranged in a triangular shape (sometimes called a "triad")

1.2.5 Flat Panel Displays

A flat CRT is obtained by initially projecting the electron beam parallel to the screen and then reflecting it throught 90. Reflecting the electron beam significantly reduces the depth of the CRT bottle and, consequently, of the display.
Types of Flat panel displays:
I. Plasma Panels.
II. Thin-film electro luminescent display
III. Light-emitted diode

Plasma Panels

Constructed by filling the region between two glass plates with a mixture of gases that usually includes neon. A series of vertical conducting ribbons is placed on one glass panel, and a set of horizontal conducting ribbons is built into the other glass panel. Firing voltages applied to an intersecting pair of horizontal and vertical conductors cause the gas at the intersection of the two conductors to break down into a glowing plasma of electrons and ions. Picture definition is stored in a refresh buffer, and the firing voltages are applied to refresh the pixel positions (at the intersection of the conductors) 60 times per second.

Thin-film electroluminescent display

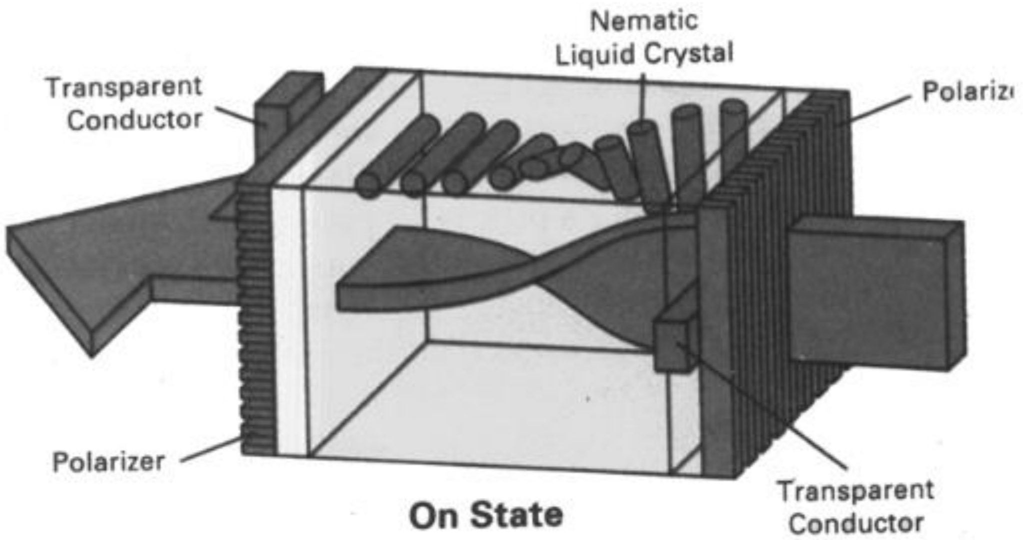
The xenon, neon, and helium gas in a plasma television is contained in hundreds of thousands of tiny cells positioned between two plates of glass. Long electrodes are also put together between the glass plates, in front of and behind the cells. The address electrodes sit behind the cells, along the rear glass plate. The transparent display electrodes, which are surrounded by an insulating dielectric material and covered by a magnesium oxide protective layer, are mounted in front of the cell, along the front glass plate. Control circuitry charges the electrodes that cross paths at a cell, creating a voltage difference between front and back and causing the gas to ionize and form a plasma. As the gas ions rush to the electrodes and collide, photons are emitted.

Thin-Film Electroluminescent

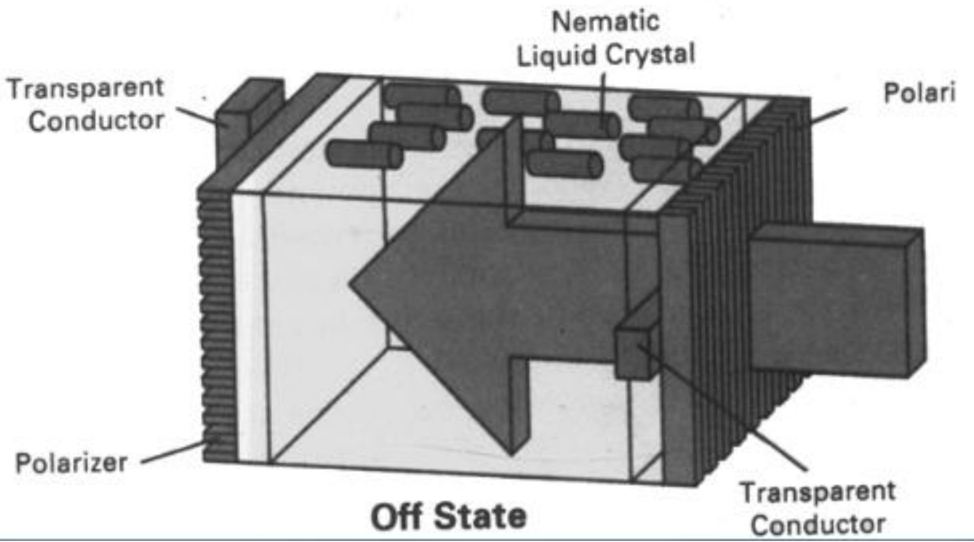
These are similar in construction to a plasma panel. The only difference is that the enfilement of the region between the glass plates is with a phosphor, such as zinc sulphide doped with manganese, instead of a gas.

Light Emitting Diode (LED)

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



On State



Off State

Active-matrix LCD

This type of LCD is constructing by placing a transistor at each pixel location, using thin-film transistor technology. • The transistors are used to control the voltage at pixel locations and to prevent charge from gradually leaking out of the liquid-crystal cells.

Passive-matrix LCD

Two glass plates, each containing a light polarizer that is aligned at a right angle to the other plate, sandwich the liquid-crystal material. • Rows of horizontal, transparent conductors are built into one glass plate, and columns of vertical conductors are put into the other plate. • The intersection of the two defines a pixel position. Polarized light passing through the material is twisted so that it will pass through the opposite polarizer. The light is then reflected back to the viewer. • To turn off the pixel, we apply a voltage to the two intersecting conductors to align the molecules so that the light is not twisted.

Input Devices:

1. Keyboard

The computer **keyboard** is used to enter text information into the computer, as when you type the contents of a report. The keyboard can also be used to type commands directing the computer to perform certain actions. Commands are typically chosen from an on-screen menu using a mouse, but there are often keyboard shortcuts for giving these same commands. In addition to the keys of the main keyboard (used for typing text), keyboards usually also have a numeric keypad (for entering numerical data efficiently), a bank of editing keys (used in text editing operations), and a row of function keys along the top (to easily invoke certain program functions). Laptop computers, which don't have room for large keyboards, often include a —fn key so that other keys can perform double duty (such as having a numeric keypad function embedded within the main keyboard keys). Improper use or positioning of a keyboard can lead to repetitive-stress injuries. Some **ergonomic** keyboards are designed with angled arrangements of keys and with built-in wrist rests that can minimize your risk of RSIs. Most keyboards attach to the PC via a PS/2 connector or USB port (newer). Older Macintosh computers used an ABD connector, but for several years now all Mac keyboards have connected using USB.

Pointing Devices The graphical user interfaces (GUIs) in use today require some kind of device for positioning the on-screen cursor. Typical pointing devices are: mouse, trackball, touch pad, trackpoint, graphics tablet, joystick, and touch screen.

Pointing devices, such as a mouse, connected to the PC via aserial ports (old), PS/2 mouse port (newer), or USB port (newest). Older Macs used ADB to connect their mice, but all recent Macs use USB (usually to a USB port right on the USB keyboard).

2. Mouse

The **mouse** pointing device sits on your work surface and is moved with your hand. In older mice, a ball in the bottom of the mouse rolls on the surface as you move the mouse, and internal rollers sense the ball movement and transmit the information to the computer via the cord of the mouse. The newer **optical mouse** does not use a rolling ball, but instead uses a light and a small optical sensor to detect the motion of the mouse by tracking a tiny image of the desk surface. Optical mice avoid the problem of a dirty mouse ball, which causes regular mice to roll unsmoothly if the mouse ball and internal rollers are not cleaned frequently. A **cordless** or **wireless mouse** communicates with the computer via radio waves (often using ***BlueTooth*** hardware and protocol) so that a cord is not needed (but such mice need internal batteries). A mouse also includes one or more buttons (and possibly a scroll wheel) to allow users to interact.

3. Light Pen

It is a pen-like device, which is connected to the machine by a cable. A light pen is a hand-held electro-optical pointing device which when touched to or aimed closely at a connected computer monitor, will allow the computer to determine where on that screen he pen is aimed. It actually does not emit light; its light sensitive –diode would sense the light coming from the screen. They are sensitive to the short burst of light emitted from the phosphor coating at the instant the electron beam strikes a particular point. Other light sources, such as the background light in the room, are usually not detected by a light pen. An activated light pen, pointed at a spot on the screen as the electron beam lights up that spot, causes the photocell to respond by generating an electrical pulse. This electric pulse response is transmitted to the processor that identifies the position to which the light pen is pointing. As with cursor-positioning devices, recorded light-pen coordinates can be used to position an object or to select a processing option**.** It facilitates drawing images and selects objects on the display screen by directly pointing the objects with the pen. Although light pens are still with us, they are not as popular as they once were since they have severage disadvantages compared to other input devices that have been developed.

4.Touch screen

It is an easisest way to enter data with the tough of a finger. Touch screens enable the user to select an option by pressing a specific part of the screen. Touch input can be recorded using optical, electrical or acoustical methods.

Infrared (optical Touch sensitive Screen)

An infrared touch screen uses an array of X-Y infrared LED and photo detector pairs around the edges of the screen to detect a disruption in the pattern of LED beams. A major benefit of such a system is that is can detect essentially any input including a finger, gloved finger, stylus or pen. It is generally used in outdoor applications and point-of-sale systems which can't rely on a conductor (such as a bare finger) to activate the touch screen. Unlike capacitive touch screens, infrared touch screens do not require any patterning on the glass which increases durability and optical clarity of the overall system.

Resistive (Electrical Touch Sensitive Screen) A resistive touch screen panel is composed of several layers, the most important of which are two thin, metallic, electrically conductive layers separated by a narrow gap. When an object, such as a finger, presses down on a point on the panel's outer surface the two metallic layers become connected at that point: the panel then behaves as a pair of voltage dividers with connected outputs. This causes a change in the electrical current, which is registered as a touch event and sent to the controller for processing.

5.Graphics tablet

A graphics tablet consists of an electronic writing area and a special —pen that works with it. graphics tablets allows artists to create graphical images with motions and actions similar to using more traditional drawing tools. The pen of the graphics tablet is pressure sensitive, so pressing harder or softer can result in brush strokes of different width (in an appropriate graphics program). A graphics tablet is an input device used by artists which allows one to draw a picture onto a computer screen without having to utilize a mouse or keyboard. A graphics tablet consists of a flat tablet and some sort of drawing device, usually either a pen or stylus. A graphics tablet may also be referred to as a drawing tablet or drawing pad. While the graphics tablet is most suited for artists and those who want the natural feel of a pen-like object to manipulate the cursor on their screen, non-artists may find them useful as well. The smooth flow of a graphics tablet can be refreshing for those who find the mouse to be a jerky input device, and repetitive stress injuries such as carpal tunnel syndrome are less likely when using a graphics tablet. These devices are more accurate than light pens. Based on the mechanism used to find two-dimensional coordianes on a flat surface, there are two types of tablets: Electromagnetic Field and Acoustic tablet.

6.Scanners

A scanner is a device that images a printed page or graphic by digitizing it, producing an image made of tiny pixels of different brightness and color values which are represented numerically and sent to the computer. Scanners scan graphics, but they can also scan pages of text which are then run through OCR (Optical Character Recognition) software that identifies the individual letter shapes and creates a text file of the page's contents.

7.Microphone

A microphone can be attached to a computer to record sound (usually through a sound card input or circuitry built into the motherboard). The sound is digitized—turned into numbers that represent the original analog sound waves—and stored in the computer to later processing and playback.

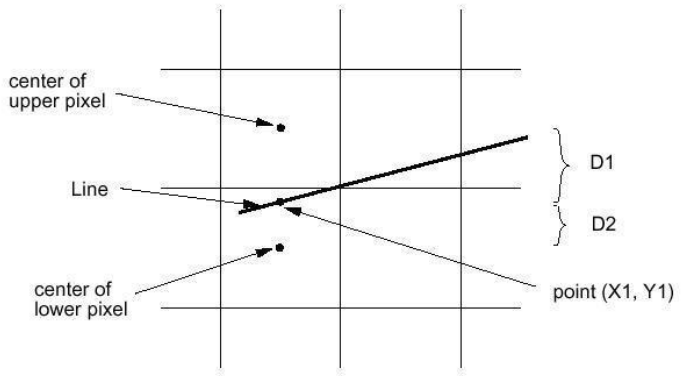
UNIT -2

1.Line drawing algorithms

1.1 DDA Algorithm

```
Procedure DDA(X1,Y1,X2,Y2 :Integer);
Var Length, I :Integer;
 X,Y,Xinc,Yinc :Real;

Begin
 Length := ABS(X2 - X1);
If ABS(Y2 - Y1)  > Length Then
 Length := ABS(Y2-Y1);
 Xinc := (X2 - X1)/Length;
 Yinc := (Y2 - Y1)/Length;
 X := X1;
 Y := Y1;
For I := 0 To Length Do
           Begin
                   Plot(Round(X), Round(Y));
                   X := X + Xinc;
                   Y := Y + Yinc
          End  {For}
End;  {DDA}
```



1.2. Bresenham's Line Algorithm

1. Input the two line end-points, storing the left end-point in (x_0, y_0)
2. Plot the point (x_0, y_0)
3. Calculate the constants Δx, Δy, $2\Delta y$, and $(2\Delta y - 2\Delta x)$ and get the first value for the decision parameter as:
   $P0=2\Delta y- \Delta x$
4. At each x_k along the line, starting at $k = 0$, perform the following test. If $p_k < 0$, the next point to plot is (x_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:

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

1.3.Mid-point Circle Algorithm

f_{circle} <0 if $(x, y)$is outside thecircleboundary

=0if $(x, y)$is on thecircleboundary

>0 if(x,y)is inside thecircleboundary

1.Input radius r and circle center (xc,yc) and obtain the first point on the circumference of the circle centered on the origin as(x0,y0) = (0,r)

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

3. At each xk position, starting at k=0, perform the following test.

If Pk<0 the next point along the circle centered on (0,0) is (xk+1,yk) and Pk+1=Pk+2xk+1+1

Otherwise the next point along the circle is (xk+1,yk-1) and Pk+1=Pk+2xk+1+1-2 yk+1

Where 2xk+1=2xk+2 and 2yk+1=2yk-2

4. Determine symmetry points in the other seven octants.

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

x=x+xc

y=y+yc

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

1.4.Midpoint Ellipse Algorithm

1.Input rx,ryand ellipse center (xc,yc) and obtain the first point on an ellipse centered on the origin as(x0,y0) = (0,r)

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

P10=ry2-rx2ry+(1/4)rx2

3.At each xkposition in region1 starting at k=0 perform the following test.

If P1k<0, the next point along the ellipse centered on (0,0) is (xk+1, yk) and

p1k+1 = p1k+2ry2xk +1+ ry2

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

p1k+1 = p1k+2ry2xk +1-2rx2yk+1 + ry2

with 2ry2xk +1 = 2ry2xk + 2ry2

2rx2yk +1 = 2rx2yk + 2rx2And continue until 2ry2x>=2rx2y

4.Calculate the initial value of the decision parameter in region 2

using the last point (x0,y0) is the last position calculated in region 1.

P20 = ry2(x0+1/2)2+rx2(yo-1)2–rx2ry2

5.At each position yk in region 2, starting at k=0 perform the following test, If p2k>0 the next point along the ellipse centered on (0,0) is (xk,yk-1) and p2k+1 = p2k–2rx2yk+1+rx2

Otherwise the next point along the ellipse is (xk+1,yk-1) andp2k+1= p2k+ 2ry2xk+1–2rxx2yk+1+ rx2

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

6.Determine symmetry points in the other three quadrants

C

2. Filled area primitives

Two methods

1. **Inside-Outside test**
2. **Winding number method**

2.1. Inside-Outside Tests:

The above algorithm only works for standard polygon shapes. However, for the cases which the edge of the polygon intersects, we need to identify whether a point is an interior or exterior point. Students may find interesting descriptions of 2 methods to solve this problem in many text books: odd-even rule and nonzero winding number rule.

2.2 Winding number method

Like even-odd method, in winding number method we have to picturise a line segment running from outside the polygon to the point in question and consider the polygon sides which it crosses

3.Polygon Filling

Filling the polygon means highlighting all the pixels which lie inside the polygon with any colour other than background colour. Polygons are easier to fill since they have linear boundaries.
There are two basic approaches used to fill the polygon.
1. Seed-fill
2.Scan line algorithm

Seed Fill

The seed fill algorithm is further classified as flood fill algorithm and boundary fill algorithm.Algorithms that fiIl interior-defined regions are called flood-fill algorithms.Those that fill boundary-defined regions are called boundary-fill algorithms or edge-fill algorithms.

Boundary Fill Algorithm or Edge Fill Algorithm

In this method, edges of the polygons are drawn.Then starting with some any point inside the polygon we examine the neighbouring pixels to check whether the toundary pixel is reached Boundary defined regions may be either 4-cormected or 8-connected.

```
void BoundaryFill(int x, int y, COLOR fill, COLOR boundary)

 {

    COLOR current;current=GetPixel(x,y);
     if   (current<>boundary)  and   (current<>fill)   then   {
     SetPixel(x,y,fill);
          BoundaryFill(x+1,y,fill,boundary);
          BoundaryFill(x-1,y,fill,boundary);
          BoundaryFill(x,y+1,fill,boundary);
          BoundaryFill(x,y-1,fill,boundary);
     }
}
```

Flood Fill Algorithm

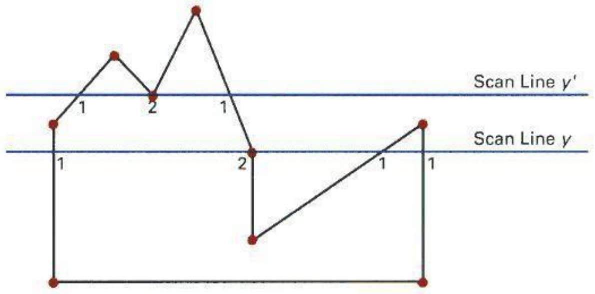
Sometimes it is required to fill in an area that is not defined within a single colour boundary. In such cases we can fill areas by replacing a specified interior colour instead of searching for a boundary colour

```
flood_fill (x, y, old-colour, new-colour).
{
       if ( getpixel (x, y) = old-colour)
       {
               putpixel (x, y, new-colour);
               flood-fitl (x + 1, y, old-colour, new-colour);
               flood-fill (x - 1, y, old-colour, new-colour);
               flood-fill (x, y + 1, old-colour, new-colour);
               flood-fill (x, y - 1, old-colour, new-colour);
               flood-fill (x + 1, y + \, old-colour, new-colour);
               flood-fill (x - 1, y - 1, old-colour, new-colour);
               flood-fill (x + 1, y - 7, old-colour, new-colour);
               flood-fill (x - 1, y + 1, old-colour, new-colour);
        }
}
```

Scan-Line Polygon Fill Algorithm

Basic idea: For each scan line crossing a polygon, this algorithm locates the intersection points of the scan line with the polygon edges. These intersection points are shorted from left to right. Then, we fill the pixels between each intersection pair.

-



Some scan-line intersection at polygon vertices require special handling. A scan line passing through a vertex as intersecting the polygon twice. In this case we may or may not add 2 points to the list of intersections, instead of adding 1 points. This decision depends on whether the 2 edges at both sides of the vertex are both above, both below, or one is above and one is below the scan line. Only for the case if both are above or both are below the scan line, then we will add 2 points.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

UNIT 3

Two Dimensional Transformations

In many applications, changes in orientations, size, and shape are accomplished with geometric transformations that alter the coordinate descriptions of objects.

Basic geometric transformations are:
- Translation
- Rotation
- Scaling

Other transformations:
- Reflection
- Shear

3.1 Basic Transformations

Translation

We translate a 2D point by adding translation distances, tx and ty, to the original coordinate position (x,y):

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

3.2 Rotation About the Origin

To rotate an object about the origin (0,0), we specify the rotation angle ?. Positive and negative values for the rotation angle define counterclockwise and clockwise rotations respectively. The followings is the computation of this rotation for a point:

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

3 Scaling With Respect to the Origin

We scale a 2D object with respect to the origin by setting the scaling factors s_x and s_y, which are multiplied to the original vertex coordinate positions (x,y):

$$x' = x * s_x,\ y' = y * s_y$$

Homogeneous co-ordinates

The previous section showed how we could mathematically represent the transformations of translation, scaling and rotation. Both scaling and rotation can be represented very conveniently using matrices, but unfortunately translation cannot. It would be nice to be able to perform all transformations using the same operation, i.e. by multiplying a position vector by a transformation matrix. This is where homogeneous co-ordinates come in useful. Homogeneous co-ordinates add an extra scaling co-ordinate (typically *w*) to a normal 2D or 3D vector. In order to convert between homogeneous co-ordinates and normal co-ordinates we must take this scaling factor into consideration.

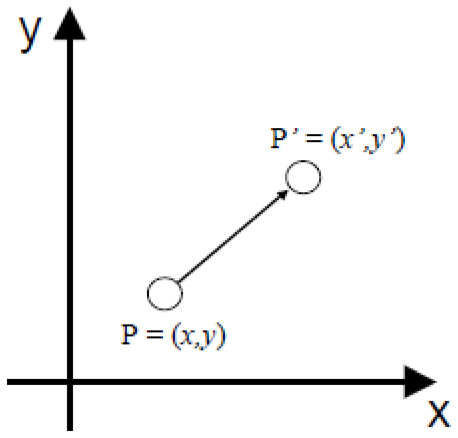
Translation

$$x' = x + t_x$$

$$y' = y + t_y$$

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

$$\mathbf{P'} = \mathbf{T}(t_x, t_y) \bullet \mathbf{P}$$



Rotation

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

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

$$[x' \;\; y' \;\; 1] = \begin{vmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{vmatrix} \quad [x \;\; y \;\; 1]$$

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

Scaling

$$x' = Sx \bullet x$$

$$y' = Sy \bullet y$$

$$[x' \;\; y' \;\; 1] = \begin{vmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{vmatrix} \quad [x \;\; y \;\; 1]$$

2D Translation

2D Translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \mathbf{P}' = \mathbf{T}\left(t_x, t_y\right) \cdot \mathbf{P}$$

2D Rotation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \mathbf{P}' = \mathbf{R}(\theta) \cdot \mathbf{P}$$

2D Scaling

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \mathbf{P}' = \mathbf{S}\left(S_x, S_y\right) \cdot \mathbf{P}$$

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

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

UNIT -4
2-Dimensional viewing

4.1Images on the Screen

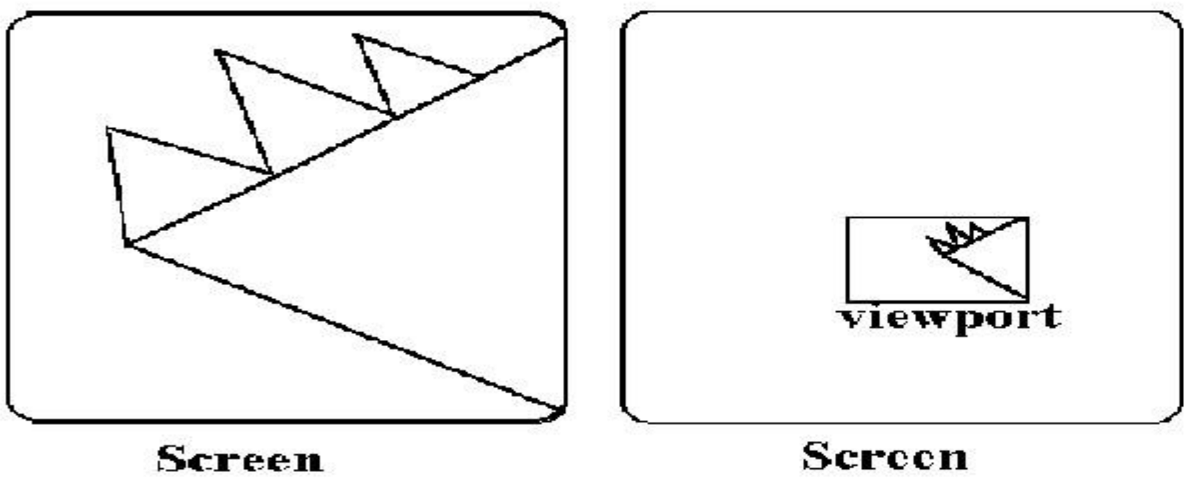
All objects in the real world have size. We use a unit of measure to describe both the size of an object as well as the location of the object in the real world. For example, meters can be used to specify both size and distance. When showing an image of an object on the screen, we use a screen coordinate system that defines the location of the object in the same relative position as in the real world. After we select the screen coordinate system, we change the picture to display interior screen that means change it into screen coordinate system.

4.1.1 Windows and Clipping

The world coordinate system is used to define the position of objects in the natural world. This
system does not depend on the screen coordinate system , so the interval of number can be
anything(positive, negative or decimal). Sometimes the complete picture of object in the world
coordinate system is too large and complicate to clearly show on the screen, and we need to
show only some part of the object. The capability that show some part of object internal a specify
window is called windowing and a rectangular region in a world coordinate system is called
window. Before going into clipping, you should understand the differences between w**indow** and
a **viewport.**

A **Window** is a rectangular region in the **world coordinate system**. This is the coordinate
system used to locate an object in the natural world. The world coordinate system does not
depend on a display device, so the units of measure can be positive, negative or decimal
numbers.

A **Viewport** is the section of the screen where the images encompassed by the window on the
world coordinate system will be drawn. A coordinate transformation is required to display the
image, encompassed by the window, in the viewport. The viewport uses the **screen coordiante
system** so this transformation is from the world coordinate system to the screen coordinate
system.



When a window is "placed" on the world, only certain objects and
parts of objects can be seen.
Points and lines which are outside the window are "cut off" from view.
This process of "cutting
off" parts of the image of the world is called **Clipping.** In clipping, we
examine each line to
determine whether or not it is completely inside the window,
completely outside the window, or
crosses a window boundary. If inside the window, the line is displayed.
If outside the
window,the lines and points are not displayed. If a line crosses the
boundary, we must determine
the point of intersection and display only the part which lies inside the
window.

CLIPPING
Clipping may be described as the procedure that identifies the portions of a picture lie inside the region, and therefore, should be drawn or, outside the specified region, and hence, not to be drawn. The algorithms that perform the job of clipping are called clipping algorithms there are various types, such as:
• Point Clipping
• Line Clipping
• Polygon Clipping
• Text Clipping
• Curve Clipping

Further, there are a wide variety of algorithms that are designed to perform certain types of clipping operations, some of them which will be discussed in unit. Line Clipping Algorithms: • Cohen Sutherland Line Clippings • Cyrus-Beck Line Clipping Algorithm Polygon or Area Clipping Algorithm • Sutherland-Hodgman Algorithm

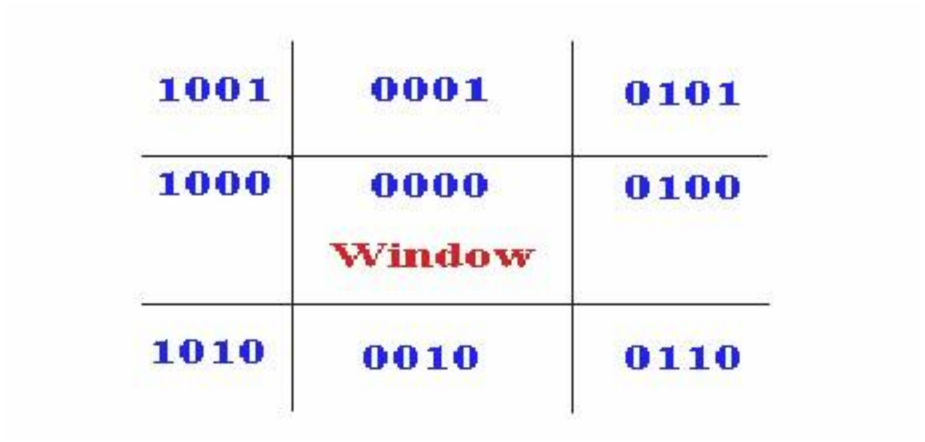
4.2 Cohen-Sutherland Line Clipping

The Cohen-Sutherland line clipping algorithm quickly detects and dispenses with two common
and trivial cases. To clip a line, we need to consider only its endpoints. If both endpoints of a line
lie inside the window, the entire line lies inside the window. It is trivially accepted and needs no
clipping. On the other hand, if both endpoints of a line lie entirely to one side of the window, the
line must lie entirely outside of the window. It is trivially and needs to be neither clipped nor
displayed.

Inside-Outside Window Codes

To determine whether endpoints are inside or outside a window, the algorithm sets up a **half-space code** for each endpoint. Each edge of the window defines an infinite line that divides the whole space into two half-spaces, the **inside half-space** and the **outside half-space**, as shown below.

As you proceed around the window, extending each edge and defining an inside half-space and an outside half-space, nine regions are created - the eight "outside" regions and the one "inside" region. Each of the nine regions associated with the window is assigned a 4-bit code to identify the region. Each bit in the code is set to either a **1**(true) or a **0**(false). If the region is to the **left** of the window, the **first** bit of the code is set to 1. If the region is to the **top** of the window, the **second** bit of the code is set to 1. If to the **right**, the **third** bit is set, and if to the **bottom**, the **fourth** bit is set. The 4 bits in the code then identify each of the nine regions as shown below.



For any endpoint **(x , y)** of a line, the code can be determined that identifies which region the endpoint lies. The code's bits are set according to the following conditions:

- First bit set **1** : Point lies to **left** of window $\mathbf{x < x_{min}}$
- Second bit set **1** : Point lies to **right** of window $\mathbf{x > x_{max}}$
- Third bit set **1** : Point lies below(**bottom**) window $\mathbf{y < y_{min}}$
- fourth bit set **1** : Point lies above(**top**) window $\mathbf{y > y_{max}}$

The sequence for reading the codes' bits is **LRBT** (Left, Right, Bottom, Top).

Once the codes for each endpoint of a line are determined, the logical **AND** operation of the codes determines if the line is completely outside of the window. If the logical AND of the endpoint codes is **not zero**, the line can be trivially rejected. For example, if an endpoint had a code of 1001 while the other endpoint had a code of 1010, the logical AND would be 1000 which indicates the line segment lies outside of the window. On the other hand, if the endpoints had codes of 1001 and 0110, the logical AND would be 0000, and the line could not be trivially rejected.

The logical **OR** of the endpoint codes determines if the line is completely inside the window. If the logical OR is **zero**, the line can be trivially accepted. For example, if the endpoint codes are 0000 and 0000, the logical OR is 0000 - the line can be trivially accepted. If the endpoint codes are 0000 and 0110, the logical OR is 0110 and the line cannot be trivially accepted.

Algorithm

The Cohen-Sutherland algorithm uses a divide-and-conquer strategy. The line segment's endpoints are tested to see if the line can be trivially accepted or rejected. If the line cannot be trivally accepted or rejected, an intersection of the line with a window edge is determined and the trivial reject/accept test is repeated. This process is continued until the line is accepted.

To perform the trivial acceptance and rejection tests, we extend the edges of the window to divide the plane of the window into the nine regions. Each end point of the line segment is then assigned the code of the region in which it lies.

1. Given a line segment with endpoint $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$
2. Compute the 4-bit codes for each endpoint.

   If both codes are **0000**,(bitwise OR of the codes yields 0000) line lies completely **inside** the window: pass the endpoints to the draw routine.

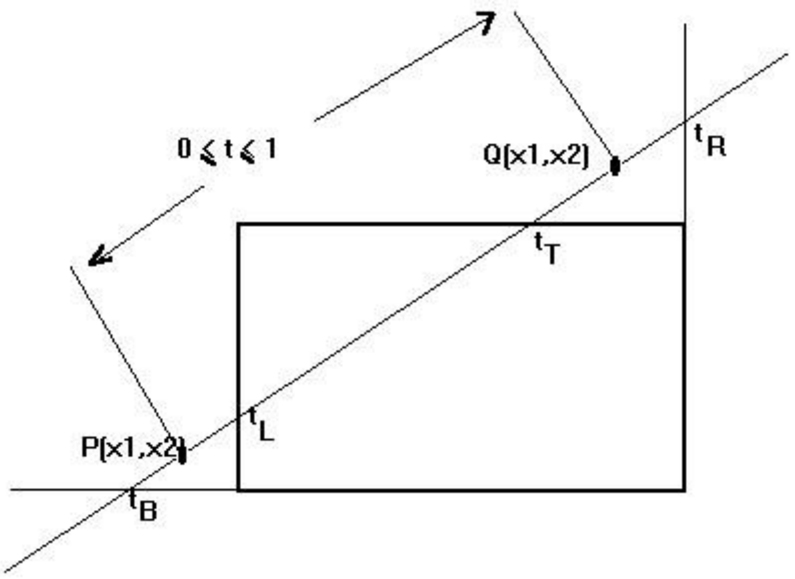
   If both codes have a 1 in the same bit position (bitwise AND of the codes is **not** 0000), the line lies **outside** the window. It can be trivially rejected.

3. If a line cannot be trivially accepted or rejected, at least one of the two endpoints must lie outside the window and the line segment crosses a window edge. This line must be **clipped** at the window edge before being passed to the drawing routine.
4. Examine one of the endpoints, say $P_1 = (x_1, y_1)$. Read P_1 's 4-bit code in order: **Left**-to-**Right**, **Bottom**-to-**Top**.
5. When a set bit (1) is found, compute the intersection **I** of the corresponding window edge with the line from P_1 to P_2. Replace P_1 with **I** and repeat the algorithm.

4.3 Liang-Barsky Line Clipping

The ideas for clipping line of Liang-Barsky and Cyrus-Beck are the same. The only difference is Liang-Barsky algorithm has been optimized for an upright rectangular clip window. So we will study only the idea of **Liang-Barsky.**

Liang and Barsky have created an algorithm that uses floating-point arithmetic but finds the appropriate end points with at most four computations. This algorithm uses the parametric equations for a line and solves four inequalities to find the range of the parameter for which the line is in the viewport.



Let P(x1,y1) , Q(x2,y2)be the line which we want to study. The **parametric equation of the line segment** from gives x-values and y-values for every point in terms of a **parameter t**hat ranges from 0 to 1. The equations are

$x = x_1 + (x_2 - x_1)*t = x_1 + dx*t$ and $y = y_1 + (y_2 - y_1)*t = y_1 + dy*t$

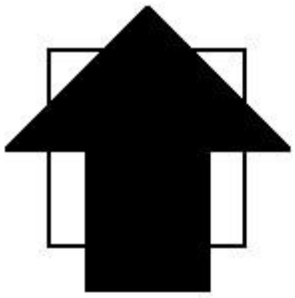
We can see that when t = 0, the point computed is P(x1,y1); and when t = 1, the point computed is Q(x2,y2).

Algorithm

1. Set $t_{min} = 0$ and $t_{max} = 1$
2. Calculate the values of tL, tR, tT, and tB (tvalues).
   - if $t < t_{min}$ or $t > t_{max}$ ignore it and go to the next edge
   - otherwise classify the **t**value as entering or exiting value (using inner product to classify)
   - if **t** is entering value set $t_{min} = t$; if **t** is exiting value set $t_{max} = t$
3. If $t_{min} < t_{max}$ then **draw a line** from (x1 + dx*tmin, y1 + dy*tmin) to (x1 + dx*tmax, y1 + dy*tmax)
4. If the line crosses over the window, you will see (x1 + dx*tmin, y1 + dy*tmin) and (x1 + dx*tmax, y1 + dy*tmax) are intersection between line and edge.

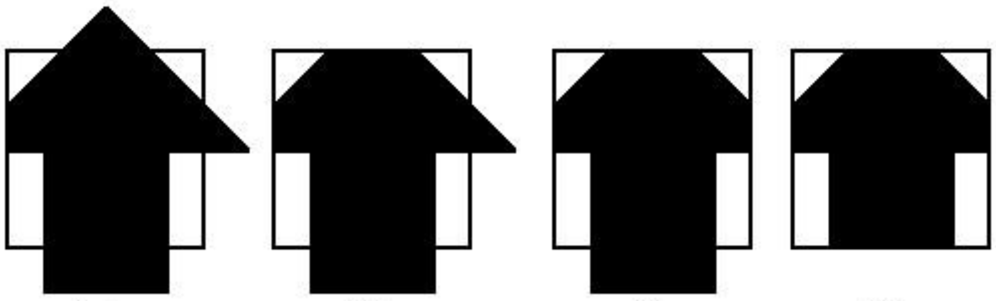
4.4 Sutherland - Hodgman Polygon Clipping

The Sutherland - Hodgman algorithm performs a clipping of a polygon against each window edge in turn. It accepts an ordered sequence of verices v1, v2, v3, ..., vn and puts out a set of vertices defining the clipped polygon.


Before clipping

This figure represents a polygon (the large, solid, upward pointing arrow) before clipping has occurred.

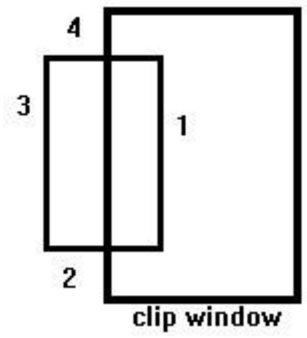
The following figures show how this algorithm works at each edge, clipping the polygon.



a. Clipping against the left side of the clip window.
b. Clipping against the top side of the clip window.
c. Clipping against the right side of the clip window.
d. Clipping against the bottom side of the clip window.

Four Types of Edges

As the algorithm goes around the edges of the window, clipping the polygon, it encounters four types of edges. All four edge types are illustrated by the polygon in the following figure. For each edge type, zero, one, or two vertices are added to the output list of vertices that define the clipped polygon.



The four types of edges are:

1. Edges that are totally inside the clip window. - add the second inside vertex point
2. Edges that are leaving the clip window. - add the intersection point as a vertex
3. Edges that are entirely outside the clip window. - add nothing to the vertex output list
4. Edges that are entering the clip window. - save the intersection and inside points as vertices

UNIT-5

3D Object Representations

Methods:

- Polygon and Quadric surfaces: For simple Euclidean objects
- Spline surfaces and construction: For curved surfaces
- Procedural methods: Eg. Fractals, Particle systems
- Physically based modeling methods
- Octree Encoding
- Isosurface displays, Volume rendering, etc.

Classification:

Boundary Representations (B-reps) eg. Polygon facets and spline patches
Space-partitioning representations eg. Octree Representation

Objects may also associate with other properties such as mass, volume, so as to determine their response to stress and temperature etc.

5.1 Polygon Surfaces

This method simplifies and speeds up the surface rendering and display of objects.

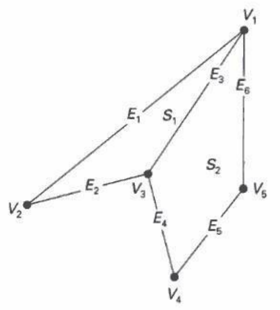
For other 3D objection representations, they are often converted into polygon surfaces before rendering.

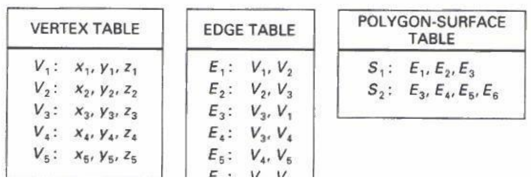
Polygon Mesh
- Using a set of connected polygonally bounded planar surfaces to represent an object, which may have curved surfaces or curved edges.
- The wireframe display of such object can be displayed quickly to give general indication of the surface structure.
- Realistic renderings can be produced by interpolating shading patterns across the polygon surfaces to eliminate or reduce the presence of polygon edge boundaries.

5.2 Polygon Tables

This is the specification of polygon surfaces using vertex coordinates and other attributes:

1. Geometric data table: vertices, edges, and polygon surfaces.
2. Attribute table: eg. Degree of transparency and surface reflectivity etc.





VERTEX TABLE	
V_1:	x_1, y_1, z_1
V_2:	x_2, y_2, z_2
V_3:	x_3, y_3, z_3
V_4:	x_4, y_4, z_4
V_5:	x_5, y_5, z_5

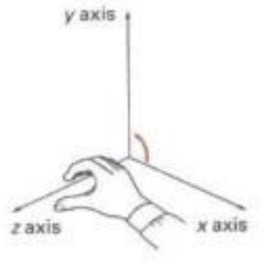
EDGE TABLE	
E_1:	V_1, V_2
E_2:	V_2, V_3
E_3:	V_3, V_1
E_4:	V_3, V_4
E_5:	V_4, V_5
[illegible]	[illegible]

POLYGON-SURFACE TABLE	
S_1:	E_1, E_2, E_3
S_2:	E_3, E_4, E_5, E_6

Some consistency checks of the geometric data table:

- Every vertex is listed as an endpoint for at least 2 edges
- Every edge is part of at least one polygon
- Every polygon is closed

5.3 Plane equation and visible points



Consider a cube, each of the 6 planes has 2 sides: inside face and outside face.

For each plane (in a right-handed coordinate system), if we look at its surface and take 3 points in counter-clockwise direction: (x_1,y_1), (x_2,y_2), and (x_3,y_3), we can compute 4 values: A,B,C,D as

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

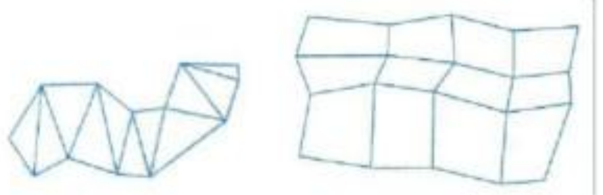
Then, the plane equation at the form: Ax+By+Cz+D=0 has the property that:

If we substitute any arbitrary point (x,y) into this equation, then,

$Ax + By + Cz + D < 0$ implies that the point (x,y) is inside the surface, and

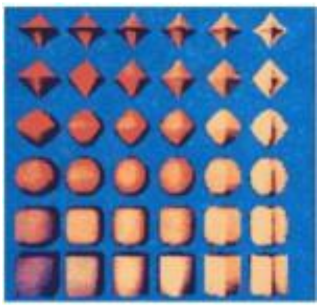
$Ax + By + Cz + D < 1$ implies that the point (x,y) is outside the surface.

Polygon Meshes
Common types of polygon meshes are triangle strip and quadrilateral mesh.



Fast hardware-implemented polygon renderers are capable of displaying up to 1,000,000 or more shaded triangles per second, including the application of surface texture and special lighting effects.

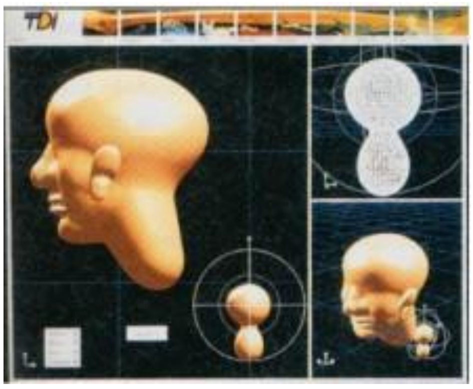
Curved Surfaces



1. Regular curved surfaces can be generated as
- Quadric Surfaces, eg. Sphere, Ellipsoid, or
- Superquadrics, eg. Superellipsoids

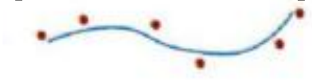
Where s1, r_x, r_y, and r_x are constants. By varying the values of and , points on the surface can be computed.

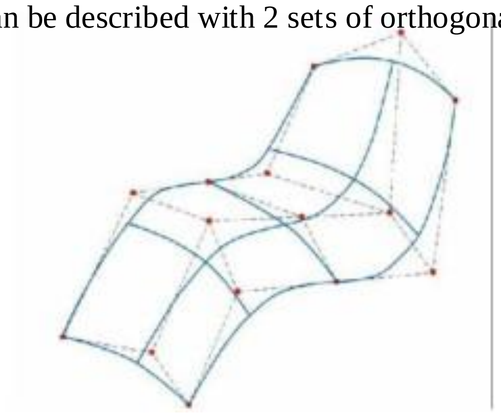
2. Irregular surfaces can also be generated using some special formulating approach, to form a kind of **blobby objects** -- The shapes showing a certain degree of fluidity.



5.4 Spline Representations

Spline means a flexible strip used to produce a smooth curve through a designated set of points. Several small weights are distributed along the length of the strip to hold it in position on the drafting table as the curve is drawn.
We can mathematically describe such a curve with a piecewise cubic polynomial function => spline curves. Then a spline surface can be described with 2 sets of orthogonal spline curves.

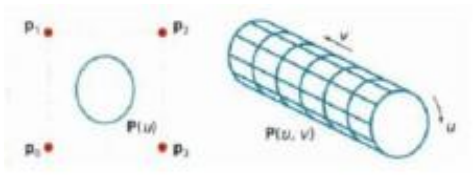




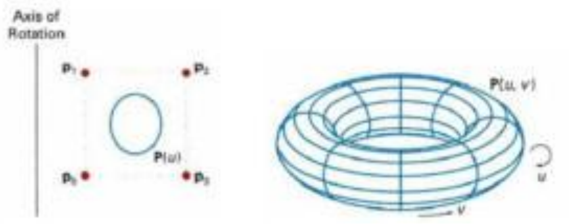
Sweep Representations

Sweep representations mean sweeping a 2D surface in 3D space to create an object. However, the objects created by this method are usually converted into polygon meshes and/or parametric surfaces before storing.

A Translational Sweep:



A Rotational Sweep:



Other variations:
- We can specify special path for the sweep as some curve function.
- We can vary the shape or size of the cross section along the sweep path.

- We can also vary the orientation of the cross section relative to the sweep path.

Unit-6

Three Dimensional Transformations:

Methods for geometric transforamtions and object modelling in 3D are extended from 2D methods by including
the considerations for the z coordinate.
Basic geometric transformations are: Translation, Rotation, Scaling

6.1 Basic Transformations

Translation

We translate a 3D point by adding translation distances, tx, ty, and tz, to the original coordinate position (x,y,z):

x' = x + tx, y' = y + ty, z' = z + tz

Alternatively, translation can also be specified by the transformation matrix in the following formula:

$$\begin{array}{cccccc} x' & 1 & 0 & 0 & t_x & x \\ y' & 0 & 1 & 0 & t & y \\ z' & 0 & 0 & 1 & tz & z \\ 1 & 0 & 0 & 0 & 1 & 1 \end{array}$$

Scaling With Respect to the Origin

We scale a 3D object with respect to the origin by setting the scaling factors sx, sy and sz, which are
multiplied to the original vertex coordinate positions (x,y,z):

x' = x * sx, y' = y * sy, z' = z * sz

Alternatively, this scaling can also be specified by the transformation matrix in the following formula:

$$\begin{array}{cccccc} x' & s_x & 0 & 0 & 0 & x \\ y' & 0 & s_y & 0 & 0 & y \\ z' & 0 & 0 & sz & 0 & z \\ 1 & 0 & 0 & 0 & 1 & 1 \end{array}$$

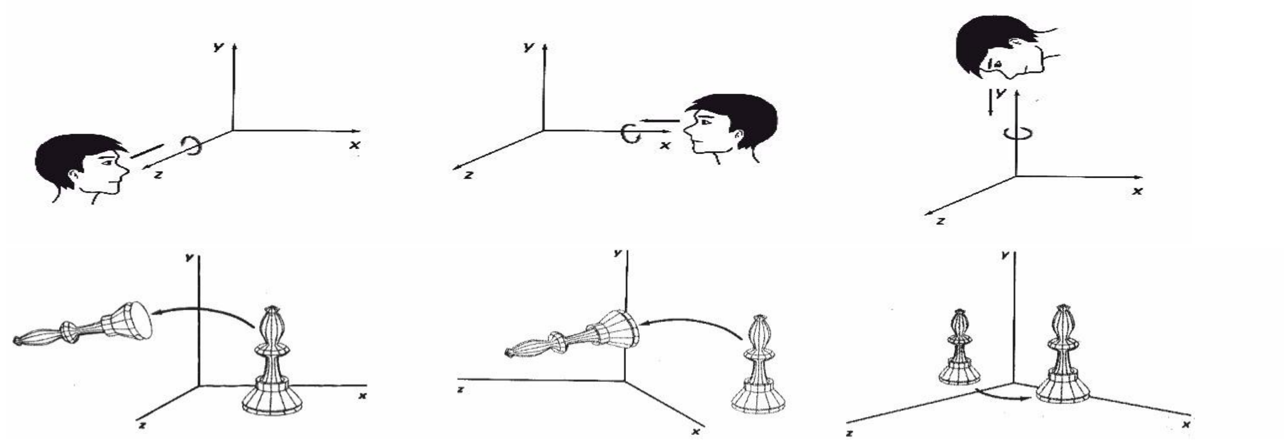
6.2 Scaling with respect to a Selected Fixed Position

Exercise: What are the steps to perform scaling with respect to a selected fixed position? Check your answer with the text book.

Exercise: Scale a triangle with vertices at original coordinates (10,25,5), (5,10,5), (20,10,10) by sx=1.5, sy=2, and sz=0.5 with respect to the centre of the triangle. For verification, roughly plot the x and y values of the original and resultant triangles, and imagine the locations of z values.

Coordinate-Axes Rotations

A 3D rotation can be specified around any line in space. The easiest rotation axes to handle are the
coordinate axes.



Z-axis rotation: $x' = x \cos\theta - y \sin\theta$

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

$z' = z$

write matrix for z- axis rotation

X-axis rotation:

$y' = y \cos\theta - z \sin\theta$,

$z' = y \sin\theta + z \cos\theta$, and

$x' = x$

write matrix for x- axis rotation

Y-axis rotation:

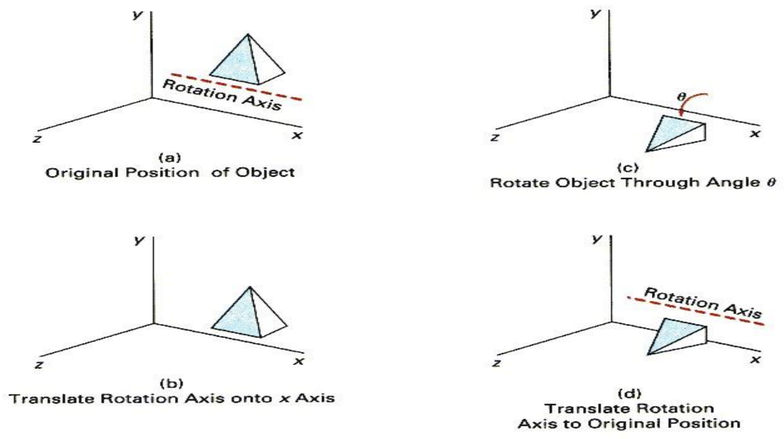
$z' = z \cos\theta - x \sin\theta$

$x' = z \sin\theta + x \cos\theta$, and

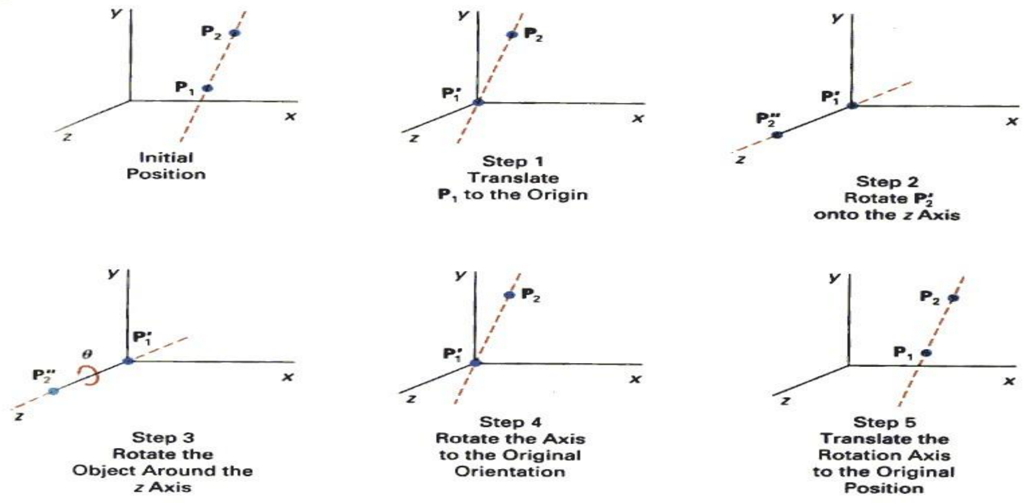
$y' = y$

write matrix for y- axis rotation

3D Rotations About an Axis Which is Parallel to an Axis


y
Rotation Axis
z
x
(a)
Original Position of Object
y
θ
z
x
(c)
Rotate Object Through Angle θ
y
z
x
(b)
Translate Rotation Axis onto x Axis
y
Rotation Axis
z
x
(d)
Translate Rotation
Axis to Original Position

Step 1. Translate the object so that the rotation axis coincides with the parallel coordinate axis.
Step 2. Perform the specified rotation about that axis.
Step 3. Translate the object so that the rotation axis is moved back to its original position.
General 3D Rotations



Step 1. Translate the object so that the rotation axis passes through the coordinate origin.
Step 2. Rotate the object so that the axis of rotation coincides with one of the coordinate axes.
Step 3. Perform the specified rotation about that coordinate axis.
Step 4. Rotate the object so that the rotation axis is brought back to its original orientation.
Step 5. Translate the object so that the rotation axis is brought back to its original position.

6.3 Three-Dimensional Viewing

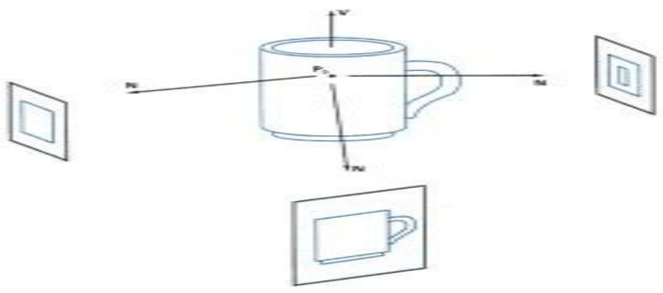
Viewing in 3D involves the following considerations:

- We can view an object from any spatial position, eg. In front of an object, Behind the object, In the middle of a group of objects, Inside an object, etc.
- 3D descriptions of objects must be projected onto the flat viewing surface of the output device.
- The clipping boundaries enclose a volume of space

6.4 Viewing Pipeline

$$\begin{tikzcd}
\text{Modelling Coordinates} \arrow[d] \\
\text{Modelling Transformations} \arrow[d] \\
\text{World Coordinates} \arrow[d] \\
\text{Viewing Transformation} \arrow[d] \\
\text{Viewing Coordinates} \arrow[d] \\
\text{Projection Transformation} \arrow[d] \\
\text{Projection Coordinates} \arrow[d] \\
\text{Workstation Transformation} \arrow[d] \\
\text{Device Coordinates}
\end{tikzcd}$$

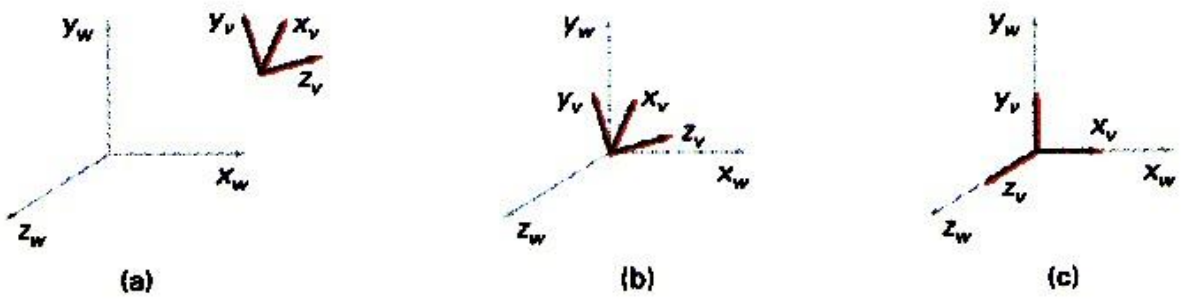
Modelling Transformation and Viewing Transformation can be done by 3D transformations. The viewing-coordinate system is used in graphics packages as a reference for specifying the observer viewing position and the position of the projection plane. Projection operations convert the viewing-coordinate description (3D) to coordinate positions on the projection plane (2D). (Usually combined with clipping, visual-surface identification, and surface-rendering)Workstation transformation maps the coordinate positions on the projection plane to the output device.



6.5 Viewing Transformation

Conversion of objection descriptions from world to viewing coordinates is equivalent to a transformation that superimposes the viewing reference frame onto the world frame using the basic
geometric translate-rotate operations:

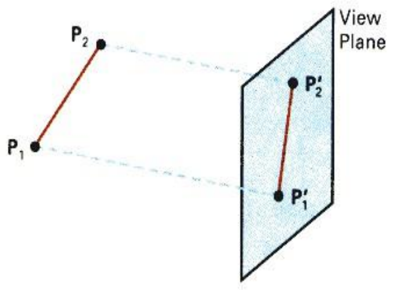
1. Translate the view reference point to the origin of the world-coordinate system.
2. Apply rotations to align the xv, yv, and zv axes (viewing coordinate system) with the world xw, yw,zw axes, respectively.

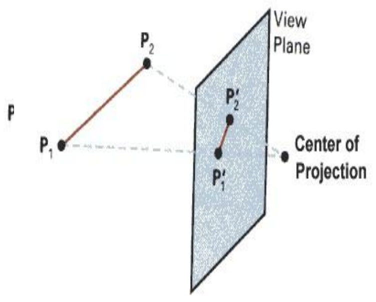


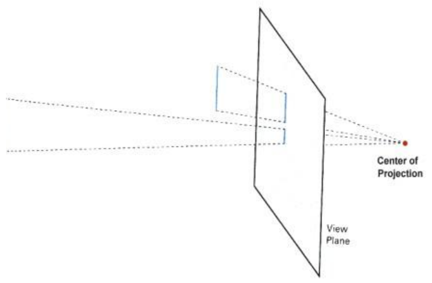
6.6 Projections

Projection operations convert the viewing-coordinate description (3D) to coordinate positions on the projection plane (2D). There are 2 basic projection methods:

1. Parallel Projection transforms object positions to the view plane along parallel lines.
A parallel projection preserves relative proportions of objects. Accurate views of the various sides ofan object are obtained with a parallel projection. But not a realistic representation
2. Perspective Projection transforms object positions to the view plane while converging to a center
point of projection. Perspective projection produces realistic views but does not preserve relative proportions. Projections of distant objects are smaller than the projections of objects of the same size that are closer to the
projection plane.



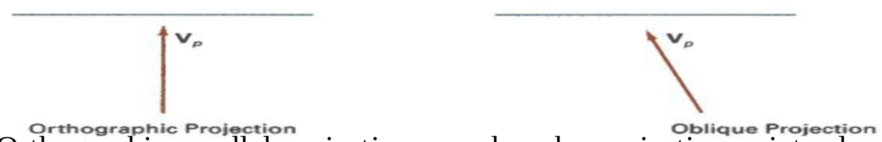




6.6.1 Parallel Projection

Classification:

Orthographic Parallel Projection and Oblique Projection:



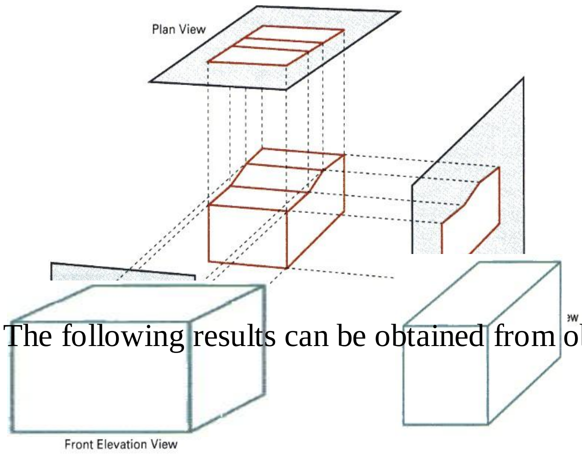
Orthographic parallel projections are done by projecting points along parallel lines that are perpendicular to the projection plane.

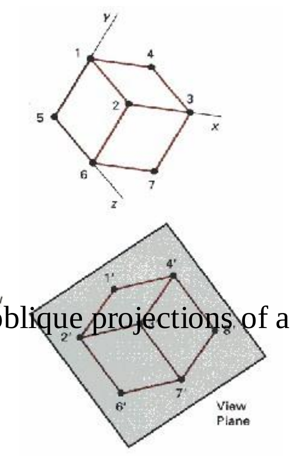
Oblique projections are obtained by projecting along parallel lines that are NOT perpendicular

to

the projection plane.Some special Orthographic Parallel Projections involve Plan View

(Top projection), Side Elevations, and Isometric Projection:



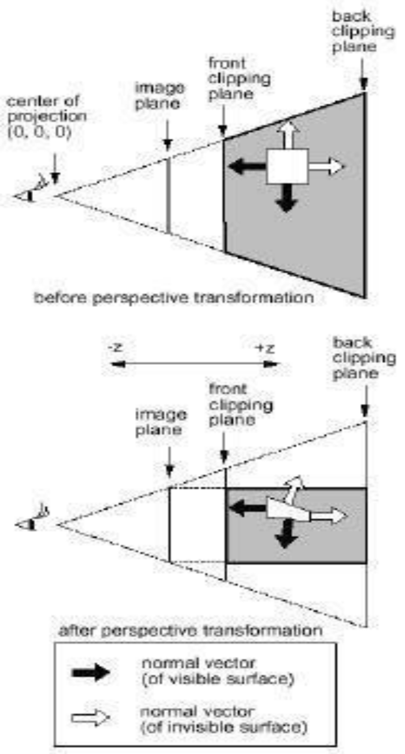


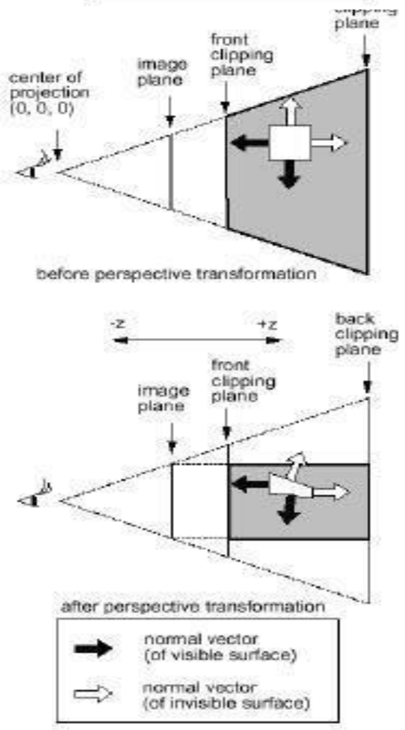
Isometric Projection

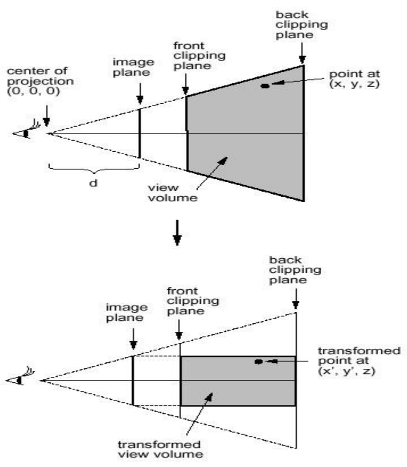
The following results can be obtained from oblique projections of a cube:

6.6.2 Perspective Projection

In Perspective projection all lines converge to a point called projection reference point or center of projection

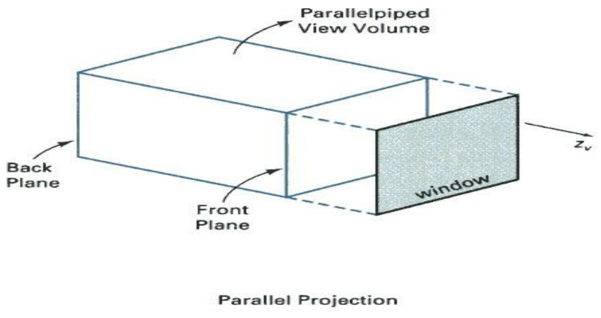

back clipping plane
front clipping plane
image plane
center of projection (0, 0, 0)
before perspective transformation
-z
+z
back clipping plane
front clipping plane
image plane
after perspective transformation
normal vector (of visible surface)
normal vector (of invisible surface)


plane
front clipping plane
image plane
center of projection (0, 0, 0)
before perspective transformation
-z
+z
back clipping plane
front clipping plane
image plane
after perspective transformation
normal vector (of visible surface)
normal vector (of invisible surface)

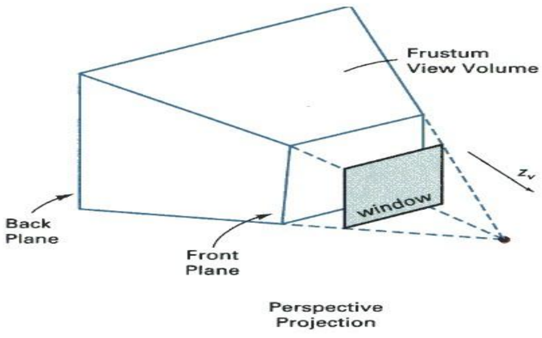


6.7 View Volumes

View window - A rectangular area in the view plane which controls how much of the scene is viewed.The edges of the view window are parallel to the xv and yv viewing axes. View volume - formed by the view window and the type of projection to be used. Only those objects within the view volume will appear in the generated display. Hence a view volume is bounded by 6 planes => rectangular parallelepiped or a frustum, for parallel projection and perspective projection respectively.



Parallel Projection



Perspective
Projection

6.8 Clipping

The purpose of 3D clipping is to identify and save all surface segments within the view volume for display on the output device. All parts of objects that are outside the view volume are discarded. Thus the computing time is saved. 3D clipping is based on 2D clipping.

Unit-7

Visible-Surface Detection Methods

More information about Modelling and Perspective Viewing:

Before going to visible surface detection, we first review and discuss the followings:

Problem definition of Visible-Surface Detection Methods:

To identify those parts of a scene that are visible from a chosen viewing position.

Surfaces which are obscured by other opaque surfaces along the line of sight (projection) are invisible to the viewer.

Characteristics of approaches:

- Require large memory size?
- Require long processing time?
- Applicable to which types of objects?

Considerations:

- Complexity of the scene
- Type of objects in the scene
- Available equipment
- Static or animated?

Classification of Visible-Surface Detection Algorithms:

Object-space Methods

Compare objects and parts of objects to each other within the scene definition to determine which surfaces, as a whole, we should label as visible: For each object in the scene do

```
Begin
1. Determine those part of the object whose view is unobstructed by other parts of it or
any other object with respect to the viewing specification.
2. Draw those parts in the object color.
End
```

- Compare each object with all other objects to determine the visibility of the object parts.
- If there are n objects in the scene, complexity = O(n2)
- Calculations are performed at the resolution in which the objects are defined (only limited by the computation hardware).

- Process is unrelated to display resolution or the individual pixel in the image and the result of the process is applicable to different display resolutions.

- Display is more accurate but computationally more expensive as compared to image space methods because step 1 is typically more complex, eg. Due to the possibility of intersection between surfaces.

- Suitable for scene with small number of objects and objects with simple relationship with each other.

Image-space Methods (Mostly used)

Visibility is determined point by point at each pixel position on the projection plane.

```
For each pixel in the image do
Begin
1. Determine the object closest to the viewer that is pierced by the projector through the
pixel
2. Draw the pixel in the object colour.
End
```

- For each pixel, examine all n objects to determine the one closest to the viewer.
- If there are p pixels in the image, complexity depends on n and p (O(np)).
- Accuarcy of the calculation is bounded by the display resolution.
- A change of display resolution requires re-calculation

Application of Coherence in Visible Surface Detection Methods:

- Making use of the results calculated for one part of the scene or image for other nearby parts.
- Coherence is the result of local similarity
- 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.

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.

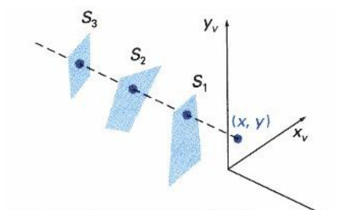
7.1 Back-Face Detection

In a solid object, there are surfaces which are facing the viewer (front faces) and there are surfaces
which are opposite to the viewer (back faces). These back faces contribute to approximately half of the total number of surfaces. Since we cannot see these surfaces anyway, to save processing time, we can remove them before the clipping process with a simple test. Each surface has a normal vector. If this vector is pointing in the direction of the center of projection, it is a front face and can be seen by the viewer. If it is pointing away from the center of projection, it is a back face and cannot be seen by the viewer. The test is very simple, if the z component of the normal vector is positive, then, it is a back face. If the z component of the vector is negative, it is a front face. Note that this technique only caters well for non overlapping convex polyhedral.
For other cases where there are concave polyhedra or

overlapping objects, we still need to apply other methods to further determine where the obscured faces are partially or completely hidden by other objects (eg.Using Depth-Buffer Method or Depth-sort Method).

7.7 Depth-Buffer Method (Z-Buffer Method)

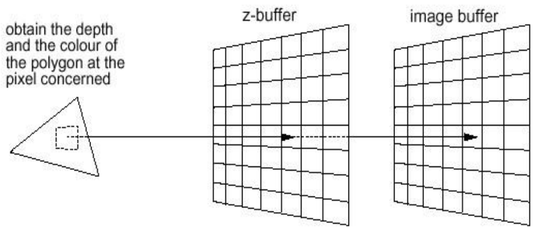
This approach compare surface depths at each pixel



At view-plane position (x, y), surface S_1 has the smallest depth from the view plane and so is visible at that position.

position on the projection plane.

Object depth is usually measured from the view plane along the z axis of a viewing system. This method requires 2 buffers: one is the image buffer and the other is called the z-buffer (or the depth buffer). Each of these buffers has the same resolution as the image to be


obtain the depth and the colour of the polygon at the pixel concerned
z-buffer
image buffer

captured. As surfaces are processed, the image buffer is used to store the color values of each pixel position and the z-buffer is used to store the depth values for each (x,y) position.

Algorithm:

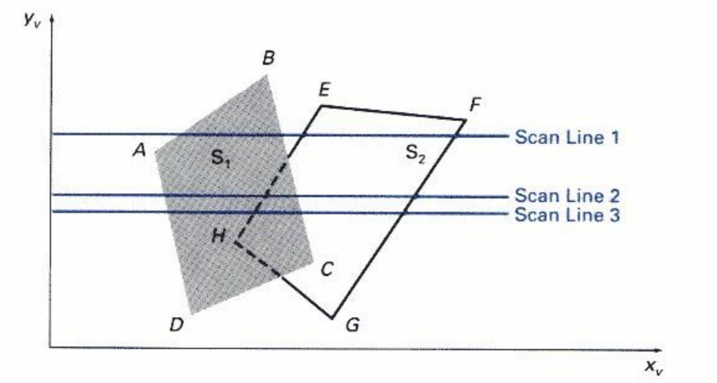
1. Initially each pixel of the z-buffer is set to the maximum depth value (the depth of the back clipping plane).
2. The image buffer is set to the background color.
3. Surfaces are rendered one at a time.
4. For the first surface, the depth value of each pixel is calculated.
5. If this depth value is smaller than the corresponding depth value in the z-buffer (ie. it is closer to the view point), both the depth value in the z-buffer and the color value in the image buffer are replaced by the depth value and the color value of this surface calculated at the pixel position.
6. Repeat step 4 and 5 for the remaining surfaces.
7. After all the surfaces have been processed, each pixel of the image buffer represents the color of a visible surface at that pixel. This method requires an additional buffer (if compared with the Depth-Sort Method) and the overheads involved in updating the buffer. So this method is less attractive in the cases where only a few objects in the scene are to be rendered.

- Simple and does not require additional data structures.
- The z-value of a polygon can be calculated incrementally.
- No pre-sorting of polygons is needed.

- No object-object comparison is required.
- Can be applied to non-polygonal objects.
- Hardware implementations of the algorithm are available in some graphics workstation.
- For large images, the algorithm could be applied to, eg., the 4 quadrants of the image separately, so as to reduce the requirement of a large additional buffer

7.2Scan-Line Method

In this method, as each scan line is processed, all polygon surfaces intersecting that line are examined to determine which are visible. Across each scan line, depth calculations are made for each overlapping surface to determine which is nearest to the view plane. When the visible surface has been determined, the intensity value for that position is entered into the image buffer.



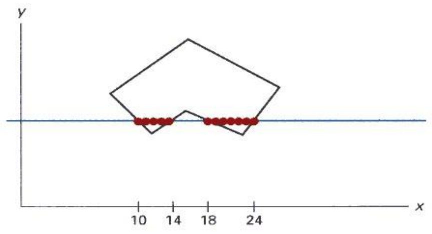
Scan lines crossing the projection of two surfaces, S_1 and S_2, in the view plane. Dashed lines indicate the boundaries of hidden surfaces.

```
For each scan line do
Begin
   For each pixel (x,y) along the scan line do ------------ Step 1
Begin
  z_buffer(x,y) = 0
   Image_buffer(x,y)  =  background_color
End
  For each polygon in the scene do ----------- Step 2
Begin
   For each pixel (x,y) along the scan line that is covered by the polygon do
Begin
   2a. Compute the depth or z of the polygon at pixel location (x,y).
   2b. If z < z_buffer(x,y) then
   Set z_buffer(x,y) = z
    Set Image_buffer(x,y) = polygon's colour
```

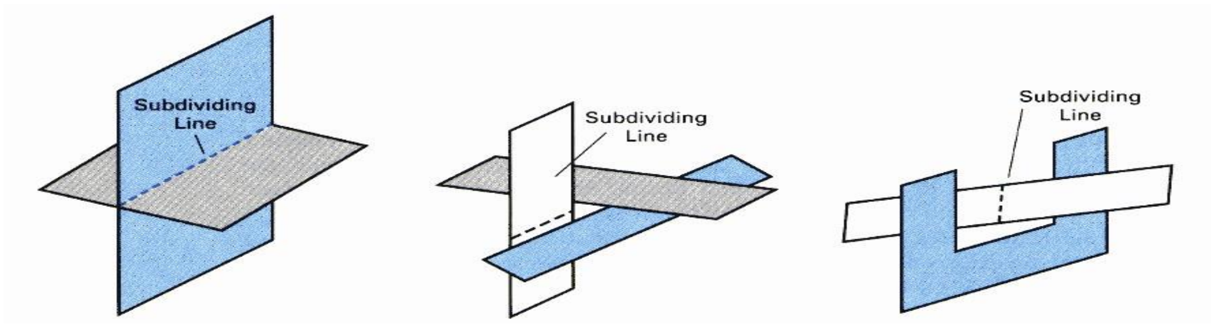
```
End
End
End
```

- Step 2 is not efficient because not all polygons necessarily intersect with the scan line.
- Depth calculation in 2a is not needed if only 1 polygon in the scene is mapped onto a segment of
the scan line.
- To speed up the process:

Recall the basic idea of polygon filling: For each scan line crossing a polygon, this algorithm locates the intersection points of the scan line with the polygon edges. These intersection points are sorted from left to right. Then, we fill the pixels between each intersection pair.



With similar idea, we fill every scan line span by span. When polygon overlaps on a scan line, we perform depth calculations at their edges to determine which polygon should be visible at which span. Any number of overlapping polygon surfaces can be processed with this method. Depth calculations are performed only when there are polygons overlapping. We can take advantage of coherence along the scan lines as we pass from one scan line to the next. If no changes in the pattern of the intersection of polygon edges with the successive scan lines, it is not necessary to do depth calculations. This works only if surfaces do not cut through or otherwise cyclically overlap each other. If cyclic overlap happens, we can divide the surfaces to eliminate the overlaps.

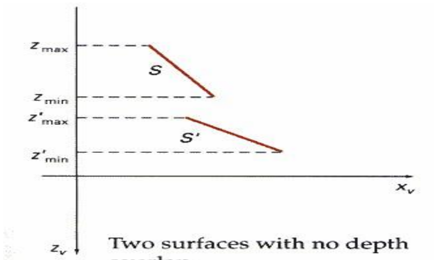


- The algorithm is applicable to non-polygonal surfaces (use of surface and active surface table, zvalue
is computed from surface representation).
- Memory requirement is less than that for depth-buffer method.
- Lot of sortings are done on x-y coordinates and on depths.

7.3Depth-Sort Method

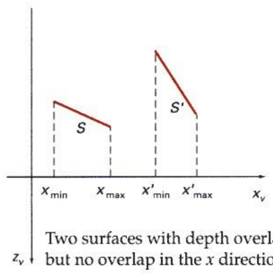
1. Sort all surfaces according to their distances from the view point.
2. Render the surfaces to the image buffer one at a time starting from the farthest surface.
3. Surfaces close to the view point will replace those which are far away.
4. After all surfaces have been processed, the image buffer stores the final image.

The basic idea of this method is simple. When there are only a few objects in the scene, this method can be very fast. However, as the number of objects increases, the sorting process can become very complex and time consuming.
Example: Assuming we are viewing along the z axis. Surface S with the greatest depth is then compared to other surfaces in the list to determine whether there are any overlaps in depth. If no depth

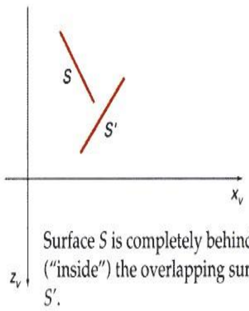


Two surfaces with no depth overlap.

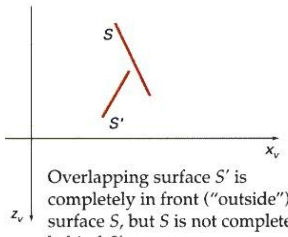
overlaps occur, S can be scan converted. This process is repeated for the next surface in the list. However, if depth overlap is detected, we need to make some additional comparisons to determine whether any of the surfaces should be reordered.



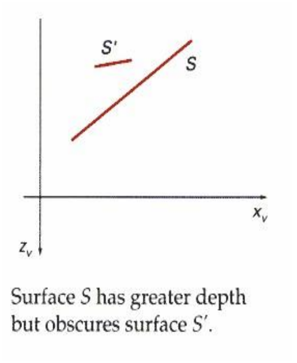
Two surfaces with depth overlap but no overlap in the x direction.



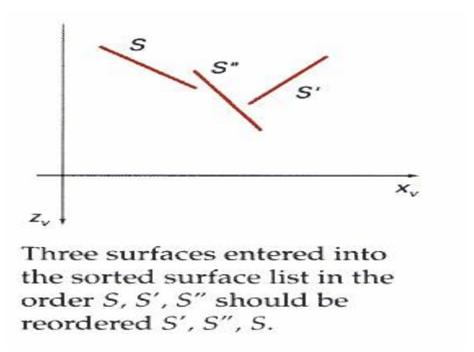
Surface S is completely behind ("inside") the overlapping surface S'.



Overlapping surface S' is completely in front ("outside") of surface S, but S is not completely behind S'.



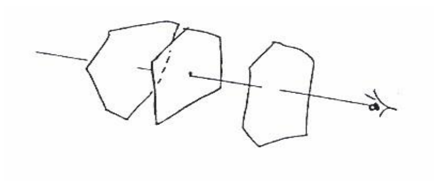
Surface S has greater depth but obscures surface S'.

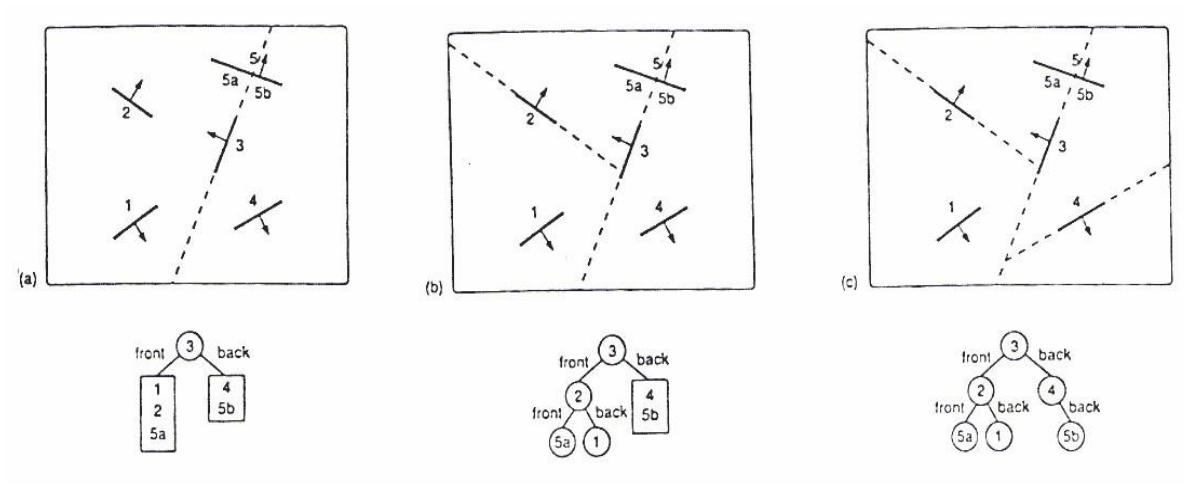


Three surfaces entered into the sorted surface list in the order S, S', S'' should be reordered S', S'', S.

7.4Binary Space Partitioning

- suitable for a static group of 3D polygon to be viewed from a number of view points
- based on the observation that hidden surface elimination of a polygon is guaranteed if all polygons on the other side of it as the viewer is painted first, then itself, then all polygons on the same side of it as the viewer





1.The algorithm first build the BSP tree:

- a root polygon is chosen (arbitrarily) which divides the region into 2 half-spaces (2 nodes => front and back)
- a polygon in the front half-space is chosen which divides the half-space into another 2 halfspaces

- the subdivision is repeated until the half-space contains a single polygon (leaf node of the tree)

- the same is done for the back space of the polygon.

2.To display a BSP tree:

- see whether the viewer is in the ront or the back half-space of the root polygon.

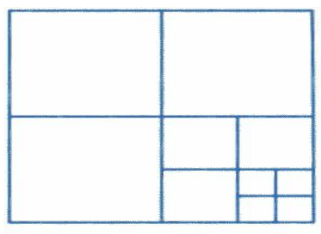
- if front half-space then first display back child (subtree) then itself, followed by its front child / subtree

- the algorithm is applied recursively to the BSP tree.

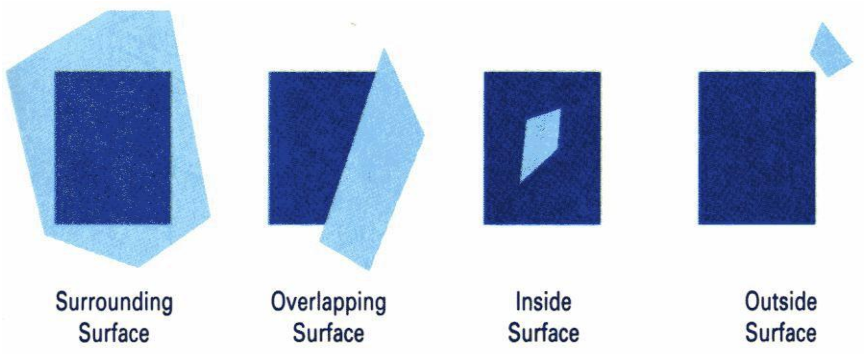
BSP Algorithm

```
Procedure
DisplayBSP(tree:
BSP_tree)
Begin
If tree is not empty then
If viewer is in
front of the root
then
Begin
DisplayBSP
(tree.back_c
hild)
displayPoly
gon(tree.roo
t)
DisplayBSP
(tree.front_c
hild)
End
Else
Begin
DisplayBSP
(tree.front_c
hild)
```

```
displayPoly
gon(tree.roo
t)
DisplayBSP
(tree.back_c
hild)
End
End
```



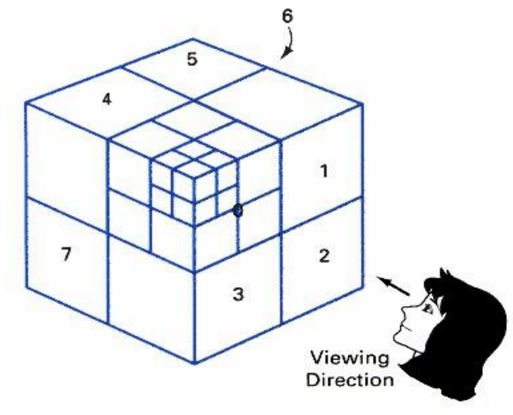
Dividing a square area into equal-sized quadrants at each step.



Possible relationships between polygon surfaces and a rectangular area.

7.4Area Subdivision Algorithms

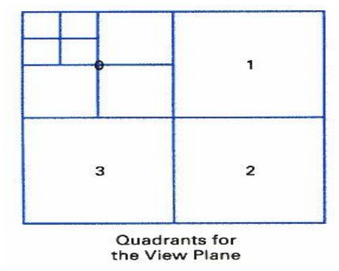
The area-subdivision method takes advantage of area coherence in a scene by locating those
view areas that represent part of a single surface. The total viewing area is
successively divided into smaller and smaller rectangles until each small area
is simple, ie. it is a single pixel, or is covered wholly by a part of a single
visible surface or no surface at all.



Octants in Space

The procedure to determine whether we should subdivide an area into smaller rectangle is:

1. We first classify each of the surfaces, according to their relations with the area:



Surrounding surface - a single surface completely encloses the area Overlapping surface - a
single surface that is partly inside and partly outside the area Inside surface - a single surface that
is completely inside the area Outside surface - a single surface that is completely outside the
area. To improve the speed of classification, we can make use of the bounding rectangles of
surfaces for early confirmation or rejection that the surfaces should be belong to that type.
2. Check the result from 1., that, if any of the following condition is true, then, no subdivision of this area is needed.
a. All surfaces are outside the area.
b. Only one surface is inside, overlapping or surrounding surface is in the area.
c. A surrounding surface obscures all other surfaces within the area boundaries.
For cases b and c, the color of the area can be determined from that single surface.

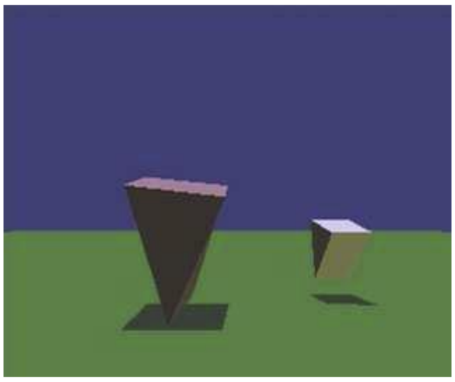
7. 5 Octree Methods

In these methods, octree nodes are projected onto the viewing surface in a front-to-back order. Any surfaces toward the rear of the front octants (0,1,2,3) or in the back octants (4,5,6,7) may be hidden by the front surfaces.With the numbering method (0,1,2,3,4,5,6,7), nodes
representing octants 0,1,2,3 for the entire region are visited before the nodes representing octants
4,5,6,7. Similarly the nodes for the front four suboctants of octant 0 are visited before the nodes

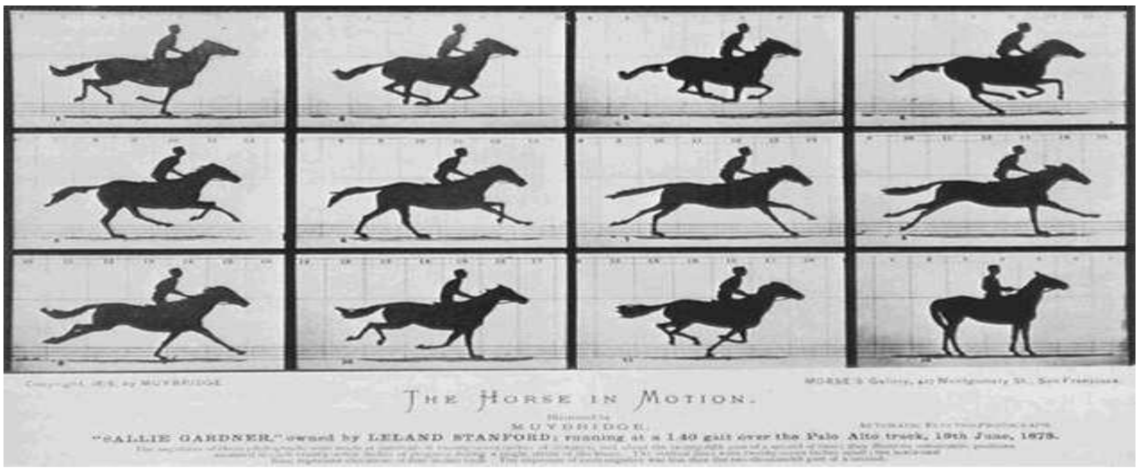
Unit-8

Computer Animation

8.1 Overview



Motion can bring the simplest of characters to life. Even simple polygonal shapes can convey a number of human qualities when animated: identity, character, gender, mood, intention, emotion, and so on. Very simple


THE HORSE IN MOTION.
MUYBRIDGE.
"SALLIE GARDNER," owned by LELAND STANFORD; running at a 1.40 gait over the Palo Alto track, 19th June, 1878.

Very simple characters (image by Ken Perlin)

A movie is a sequence of frames of still images. For video, the frame rate is typically 24 frames per second. For film, this is 30 frames per second.

In general, animation may be achieved by specifying a model with n parameters that identify degrees of freedom that an animator may be interested in such as

• polygon vertices,

• spline control,

• joint angles,

• muscle contraction,

• camera parameters, or

• color.

With n parameters, this results in a vector $\tilde{q}$ in n-dimensional state space. Parameters may be varied to generate animation. A model's motion is a trajectory through its state space or a set of motion curves for each parameter over time, i.e. $\tilde{q}(t)$, where t is the time of the current frame. Every animation technique reduces to specifying the state space trajectory.
The basic animation algorithm is then: `for t=t1 to tend: render(~q(t)).`
Modeling and animation are loosely coupled. Modeling describes control values and their actions.
Animation describes how to vary the control values. There are a number of animation techniques,
including the following:

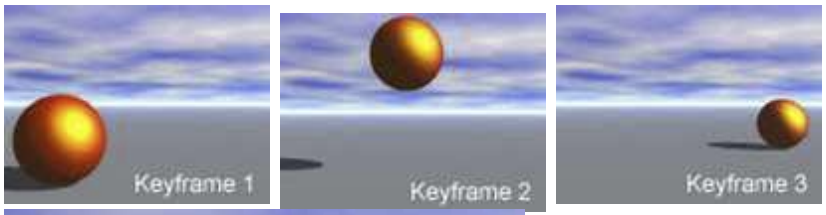
• User driven animation

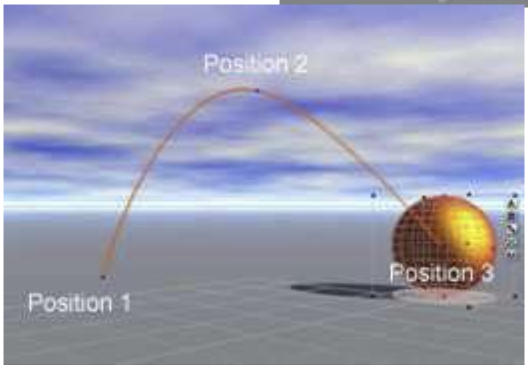
- Keyframing
- Motion capture

• Procedural animation

- Physical simulation
- Particle systems
- Crowd behaviors

• Data-driven animation





8.2 Keyframing

Keyframing is an animation technique where motion curves are interpolated through states at times, ($\tilde{q}_1, \ldots, \tilde{q}_T$), called keyframes, specified by a user

Catmull-Rom splines are well suited for keyframe animation because they pass through their control points.

• Pros:

- Very expressive

- Animator has complete control over all motion parameters

• Cons:

Very labor intensive

- Difficult to create convincing physical realism

• Uses:

- Potentially everything except complex physical phenomena such as smoke, water, or fire

8.3 Kinematics

Kinematics describe the properties of shape and motion independent of physical forces that cause motion. Kinematic techniques are used often in keyframing, with an animator either setting joint parameters explicitly with **forward kinematics** or specifying a few key joint orientations and having the rest computed automatically with **inverse kinematics**.

16.3.1 Forward Kinematics

With forward kinematics, a point $\bar{p}$ is positioned by $\bar{p}$ = (_) where_is a state vector (θ_1, θ_2, … θ_n)
specifying the position, orientation, and rotation of all joints.

For the above example, $\bar{p} = (l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2),\ l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2))$.

Inverse Kinematics

With inverse kinematics, a user specifies the position of the end effector, $\bar{p}$, and the algorithm has to evaluate the required _ give $\bar{p}$. That is, _ = $f^{-1}(\bar{p})$.
Usually, numerical methods are used to solve this problem, as it is often nonlinear and either underdetermined or overdetermined. A system is underdetermined when there is not a unique solution, such as when there are more equations than unknowns. A system is overdetermined when it is inconsistent and has no solutions.
Extra constraints are necessary to obtain unique and stable solutions. For example, constraints may be placed on the range of joint motion and the solution may be required to minimize the kinetic energy of the system.

8.3.1 Motion Capture

In motion capture, an actor has a number of small, round markers attached to his or her body that reflect light in frequency ranges that motion capture cameras are specifically designed to pick up

image from movement.nyu.edu)

With enough cameras, it is possible to reconstruct the position of the markers accurately in 3D. In practice, this is a laborious process. Markers tend to be hidden from cameras and 3D reconstructions fail, requiring a user to manually fix such drop outs. The resulting motion curves are often noisy, requiring yet more effort to clean up the motion data to more accurately match what an animator wants. Despite the labor involved, motion capture has become a popular technique in the movie and game industries, as it allows fairly accurate animations to be created from the motion of actors. However, this is limited by the density of markers that can be placed on a single actor. Faces, for example, are still very difficult to convincingly reconstruct.

Pros:
- Captures specific style of real actors

• Cons:
- Often not expressive enough
- Time consuming and expensive
- Difficult to edit

• Uses:
- Character animation
- Medicine, such as kinesiology and biomechanics