SCHOOL OF COMPUTING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT-1- SCSA1503- Computer Graphics and Multimedia Applications

Output Primitives: Survey of computer graphics – Overview of graphics systems – Line drawing algorithm –Circle drawing algorithm – Curve drawing algorithm - Attributes of output primitives – Anti-aliasing

Basics of Computer Graphics

Computer Graphics involves creation, display, manipulation and storage of pictures and experimental data/models or images for proper visualization using a computer. Such models come from diverse and expanding set of fields including physical, biological, mathematical, artistic, and conceptual/abstract structures.Typical graphics system comprises of a host computer with support of fast processor, large memory, frame buffer and

•Display devices (color monitors),

• Input devices (mouse, keyboard, joystick, touch screen, trackball)

• Output devices (LCD panels, laser printers, color printers. Plotters etc.)

•Interfacing devices such as, video I/O, TV interface etc.

Application of Computer Graphics

1. **Computer Aided Design (CAD) :**

• It is used in the design of Buildings, Automobiles, Aircraft, Watercraft, Spacecraft, Computers, Textiles and many more products

- For some design applications, objects are first displayed in a wireframe outline which shows the overall shape and internal features of objects

- Circuits and Networks for communications, water supply or other utilities are constructed

- The shapes used in a design represent the different network or circuit components

- Real time animations using wireframe displays on a video monitor are useful for testing performance of a vehicle or system

- Realistic displays are generated for advertising of automobiles and other vehicles using special lighting effects and background scenes

- Architects use interactive graphics methods to layout floor plans such as positioning of rooms, windows, stairs, shelves and other building features
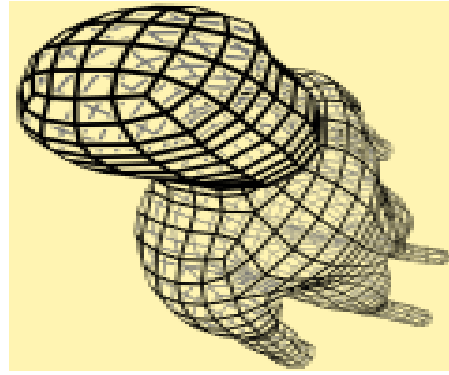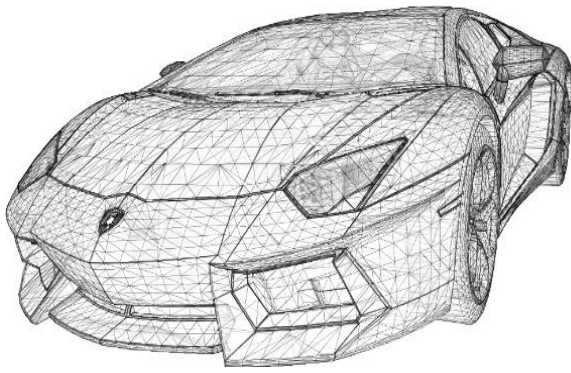
Fig 1.1 Example for Wireframe drawing
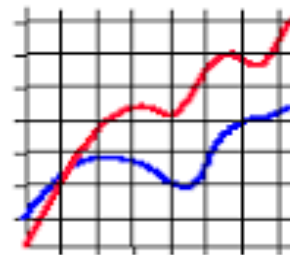
2. Presentation Graphics

Presentation Graphics is used to produce illustrations for reports, Slides or transparencies for use with projectors. It is used to summarize Financial, Statistical, Mathematical, Scientific as well as Economic Data.
For example,Barcharts, Line Graphs, Surface Graphs, Pie charts and other displays shows relationships between multiple parameters.
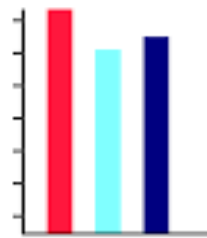
Pie Chart   Bar Chart   Graph

Fig 1.2 Example for Presentation Graphics

2.   Computer Art

Computer Graphics are used in both fine art and commercial art applications. Artists use a variety of computer methods including special purpose hardware, artists paint brush programs, specially developed software,symbolic mathematical packages, CAD software, Desktop publishing software and Animation packages that provide facilities for designing object shapes and specifying object motions

Eg) LOGOs , TV advertising, Combining Text and Graphics, Morphing, etc.

Fig 1.3 Example for Computer Art

4. Entertainment

Graphics objects are combined with actors and live scenes.Eg) Motion pictures, Music Videos, Television shows.

Fig 1.4   Example for animation

5. Education and Training
Computer generated models of physical, financial and economic systems are often used as educational aids.
Eg) Simulators for practice sessions, Training of ship captions, Training of pilots, Air traffic control personnel, etc.

Fig 1.5 Example for aircraft simulator

6. Visualization
Scientists, Engineers, Medical Personnel, Business Analysts need to analyze large amount of data. Numerical simulations carried out on super computers frequently produce data files containing thousands and millions of data values. Satellite cameras and other sources can process amassing large data files faster than they can be interpreted .
Visualization can be of two types :
1.   Scientific : Graphical representation for scientific, engineering and Medical datasets

2.   Business : Datasets related to commerce,industry and other non-scientific areas

Fig 1.5 Example for Data Analysis

7. Image Processing
Image processing is applied to modify / interpret existing pictures such as photographs and TV scans.
Applications of Image Processing:
1.   Improving picture quality

2.   M/C perception of visual information as used in robotics

In order to apply image-processing methods:
 Digitize a photograph or picture into an image file
 Apply techniques to enhance color separation or improve the quality of shading
e.g., Analyze satellite photos, Photos of earth, Galaxy, etc.
Medical Image processing uses Ultrasonics which converts High frequency sound waves to process data
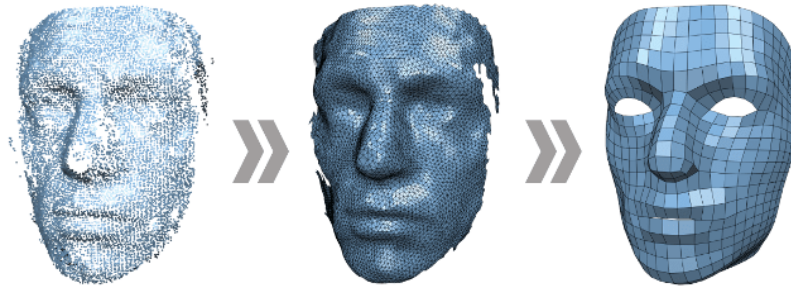   and Medical scanners

Fig 1.6 Example for Image Processing

8. Graphical User Interface
Uses Window manager which allows a user to display multiple window areas. It can be activated by an
icon.

Fig 1.6 Example for Graphical User Interface

<h1 style="text-align:center">Video Display devices</h1>

Cathode Ray Tube (CRT)

The primary output device in a graphical system is the video monitor. The main element of a video monitor is the **Cathode Ray Tube,** shown in the following illustration (Fig. 1.1).

The operation of CRT is very simple:

- The electron gun emits a beam of electrons (cathode rays), when the filament is heated.

- Intensity of the electron beam is controlled by setting voltage levels on the control grid.
- The electron beam passes through focusing and deflection systems that direct it towards specified positions on the phosphor-coated screen. The focusing system is needed to force he electron beam to converge into a small spot as it strikes the phosphor screen, else the bam would spread out as it approaches the screen.

- When the beam hits the screen, the phosphor emits a small spot of light at each position contacted by the electron beam.

- It redraws the picture by directing the electron beam back over the same screen points quickly. This is called refreshing, hence the CRT is called as Refresh CRT.

- The difference between the kinds of phosphors is their persistence- how long they continue to emit light after the CRT beam is removed.
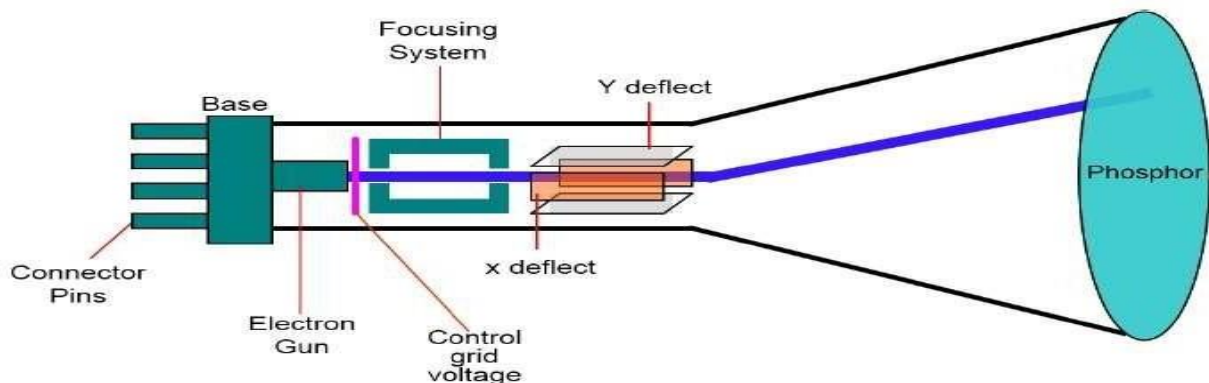
Fig 1.1 Cathode ray tube

There are two ways (Random scan and Raster scan) by which we can display an object on the screen.

Raster Scan

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

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

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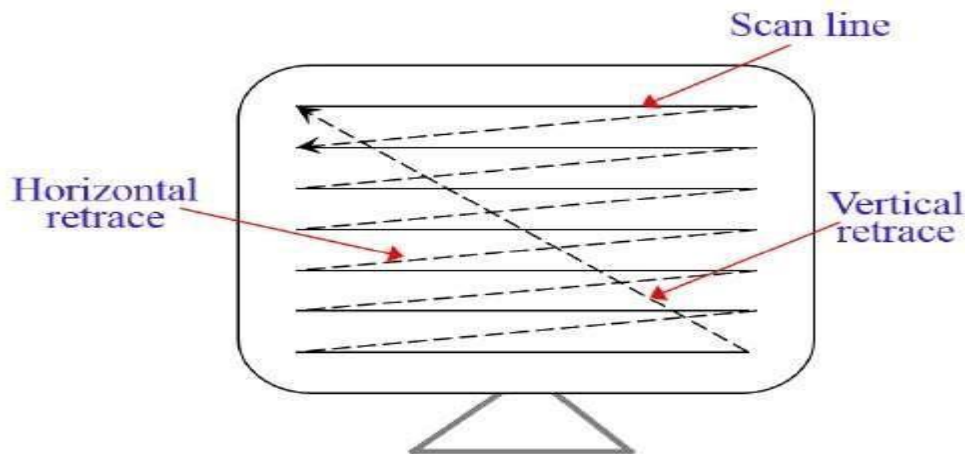

Fig 1.2 Raster scan display

Random Scan (Vector Scan)

In this technique, the electron beam is directed only to the part of the screen where the picture is to be drawn rather than scanning from left to right and top to bottom as in raster scan. It is also

called **vector display, stroke-writing display,** or **calligraphic display**.

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

Random-scan displays are designed to draw all the component lines of a picture 30 to 60 times each second. Fig 1.3 shows random scan method.
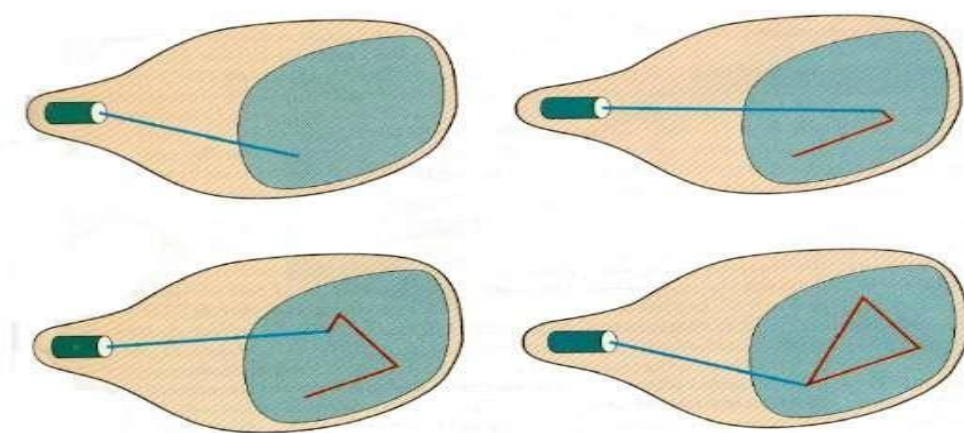

Fig 1.3: Random scan display

Color CRT Monitor

A CRT monitor displays color picture by using a combination of phosphor that emit different-colored light. By combining the emitted light from the different phosphor, a range of colors can be generated. The two basic techniques for producing color displays with a CRT are the beam-penetration method and the shadow-mask method.

Beam-penetration method

The beam-penetration method for displaying color pictures has been used with random-scan monitors. Two layers of phosphor, usually red and green, are coated onto the inside of the 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 electron penetrates through the red layer and excites the inner green layer. At intermediate beam speeds, combinations of red and green light are emitted to show two additional colors, orange and yellow.

The speed of the electrons, and hence the screen color at any point, is controlled by the beam-acceleration voltage. Beam penetration has been aninexpensive way to produce color in random-scan monitor, but only four colors are possible, and the quality of picture is not as good as with other methods.

Shadow Mask Method

Shadow-mask,(Fig 1.4) methods are commonly used in raster-scan system (including color TV) because they produce a much wider range of colors than the beam penetration method. A shadow-mask CRT has three phosphor color dots at each pixel position. One phosphor dot emits a red light, another emits a green light, and the third emits a blue light. This type ofCRT has three electron guns, one for each color dot, and a shadow-mask grid just behind the phosphor-coated screen. Figure illustrates the delta-delta shadow-mask method, commonly used in color CRT system. The three beams are deflected and focused as a group onto the shadow mask, which contains a series of holes aligned with the phosphor-dot patterns. When the three beams pass through a hole in the shadow mask, they activate a dot triangle, which appears as a small color spot on the screen. The phosphor dots in the triangles are arranged so that each electron beam can activate only its corresponding color dot when it passes through the shadow mask. Another configuration for the three electron guns is an in-line arrangement in which the three electron guns, and the corresponding redgreen-blue color dots on the screen, are aligned along one scan line instead of in a triangular pattern. This in-line arrangement of electron guns is easier to keep in alignment and is commonly used in high-resolution color CRTs.
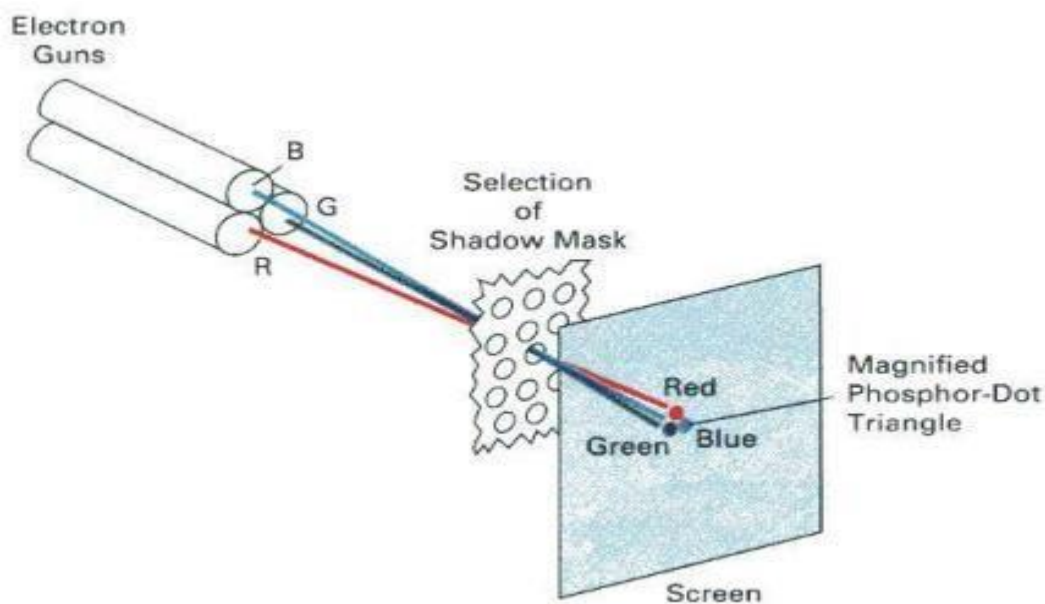
Fig 1.4 : Shadow mask method

We obtain color variations in a shadow-mask CRT by varying the intensity levels of the three electron beams. By turning off the red and green guns, we get only the color coming from the blue phosphor. Other combinations of beam intensities produce a small light spot for each pixel position, since our eyes tend to merge the three colors into one composite. The color we see depends on the amount of excitation of the red, green, and blue phosphors. A white (or gray) area is the result of activating all three dots with equal intensity. Yellow is produced with the green and red dots only, magenta is produced with the blue and red dots, any cyan shows up when blue and green are activated equally. In some low-cost systems, the electron beam can only be set to on or off, limiting displays to eight colors. More sophisticated systems can set intermediate level for the electron beam, allowing several million different colors to be generated.

Direct – View Storage Tubes (DVST)

In raster scan display, we do refreshing of the screen to maintain a screen image. The DVST gives the alternative method of maintaining the screen image.A DVST uses the storage grid which stores the picture information as a charge distribution just behind the phosphor-coated screen.The figure below (Fig 1.5) shows the general arrangement of the DVST. It consists of two electron guns: a primary gun and a flood gun.
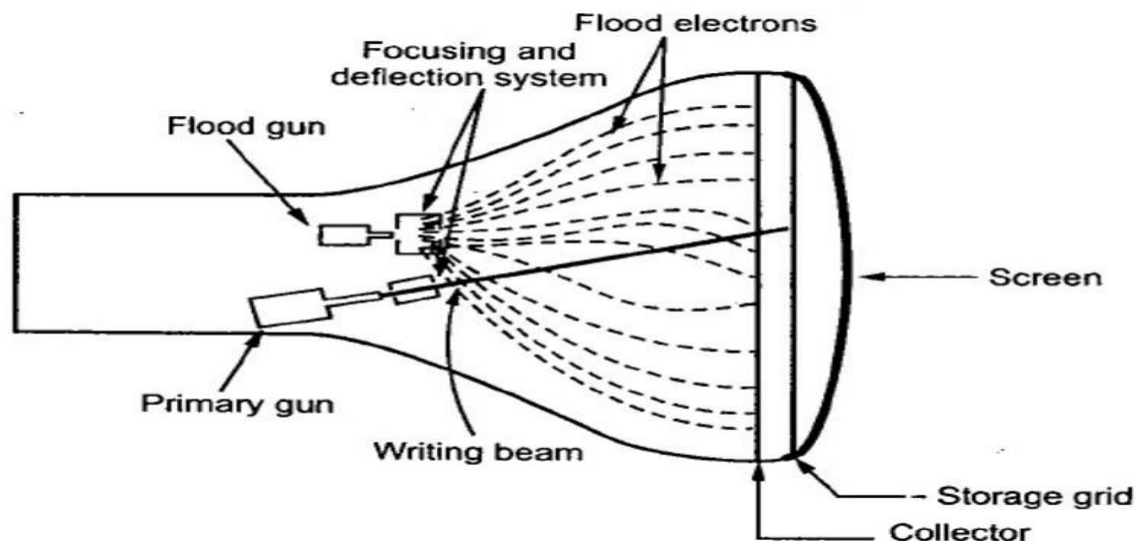

Fig 1.5 DVST

The primary gun stores the picture pattern and the flood gun maintains the picture display.

The primary gun produces high speed electrons which strike on the storage grid to draw the picture pattern. As electron beam strikes on the storage grid with high speed, it knocks out electrons from the storage grid keeping the net positive charge. The knocked out electrons are attracted towards the collector. The net positive charge on the storage grid is nothing but the picture pattern. The continuous low speed electrons from flood gun pass through the control grid and are attracted to the positive charged areas of the storage grid. The low speed electrons then penetrate the storage grid and strike the phosphor coating without affecting the positive charge pattern on the storage grid. During this process the collector just behind the storage grid smooth's out the flow of flood electrons.

Advantages:

- Refreshing of CRT is not required.
- Because no refreshing is required, very complex pictures can be displayed at very high resolution without flicker.
- It has flat screen.

Disadvantages:

- They do not display colors and are available with single level of line intensity.
- Erasing requires removal of charge on the storage grid Thus erasing and redrawing process takes several seconds.
- Selective or part erasing of screen is not possible.
- Erasing of screen produces unpleasant flash over the entire screen surface which prevents its use of dynamics graphics applications.
- It has poor contrast as a result of the comparatively low accelerating potential applied to the flood electrons.
- The Performance of DVST is somewhat inferior to the refresh CRT.

Flat Panel Displays

Although most graphics monitors are still constructed with CRTs, other technologies are emerging that may soon replace CRT monitors. The term Flat panel displays refers to aclass of video devices that have reduced volume, weight and power requirements compared to a CRT. The

important feature of flat panel display is that they are thinner than the CRTs. Since we can even write on some flat-panel displays, they will soon be available as pocket notepads. Current uses for flat-panel displays include small TV monitors, calculators, pocket video games, laptop computers, armrest viewing of movies on airlines, as advertisement boards in elevators, and as graphics displays in applications requiring rugged, portable monitors. There are 2 types: Emissive and Non Emissive displays.

Emissive Displays: They covert electrical energy into light energy. E.g. Plasma panels, Thin Film Electroluminescent displays, LEDs.

Non Emissive Displays: They use optical effects to convert sunlight or light from some other source into graphics patterns. E.g. LCD (Liquid Crystal Display)

Plasma panels, also called gas-discharge displays, are 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 ribbons is built into the other glass panel (Fig 1.6 below). Firing voltages applied to a 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 intersections of the conductors) 60 times per second. Alternating –current methods are used to provide faster application of the firing voltages, and thus brighter displays. Separation between pixels is provided by the electric field of the conductors. Figure below shows a high definition plasma panel. One disadvantage of plasma panels has been that they were strictly monochromatic devices, but systems have been developed that are now capable of displaying color and grayscale.
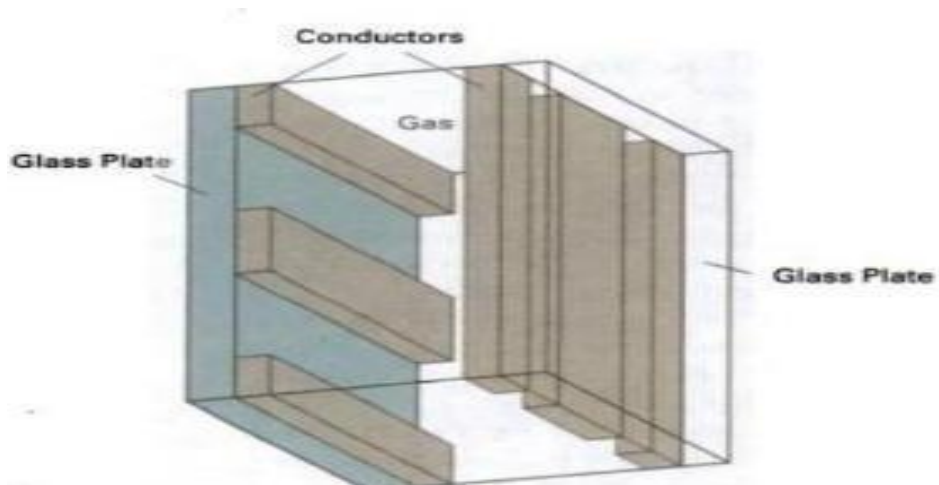
Fig 1.6 Basic design of a Plasma Panel display device

Thin-film electroluminescent displays are similar in construction to a plasma panel. The difference is that the region between the glass plates is filled with a phosphor, such as zinc sulfide doped with manganese,instead of a gas (see Fig 1.7 below). When a sufficiently high voltage is applied to a pair of crossing electrodes, the phosphor becomes a conductor in the area of the intersection of the two electrodes. Electrical energy is then absorbed by the manganese atoms, which then release the energy as a spot of light similar to the glowing plasma effect in a plasma panel. Electroluminescent displays require more power than plasma panels, and good color and gray scale displays are hard to achieve.
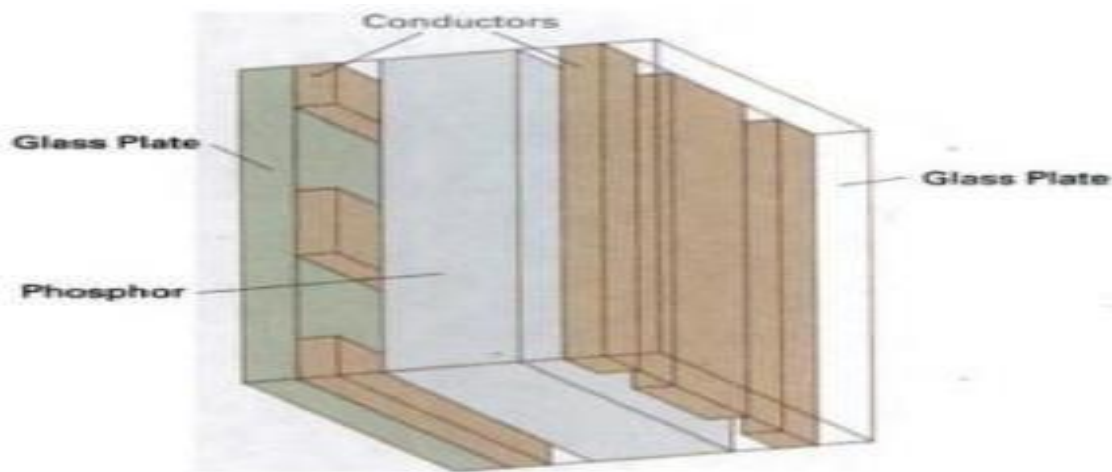

Fig 1.7 Basic design of a thin-film electroluminescent display device

A third type of emissive device is the **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. As in scan-line refreshing of a CRT, 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.

Liquid Crystal Display (LCD) are commonly used in small systems, such as calculators (see figure below) and portable, laptop computers. These nonemissive devices produce a picture by passing polarized light from the surroundings or from an internal light source through a liquid-Crystal material that can be aligned to either block or transmit the light. The term liquid crystal refers to the fact that these compounds have a crystalline arrangement of molecules, yet they flow like a liquid. Flat-panel displays commonly use nematic (threadlike) liquid-crystal compounds that tend to keep the long axes of the rodshaped molecules aligned. A flat-panel display can then be constructed with a nematic liquid crystal. Two glass plates, each containing a light polarizer at

right angles to the-other plate, sandwich the liquid-crystal material. Rows of horizontal transparent conductors are built into oneglass plate, and columns of vertical conductors are put into the other plate. The intersection of two conductors defines a pixel position.

Normally, the molecules are aligned as shown in the "on state". 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. This type of flat-panel device is referred to as a passive-matrix LCD. Picture definitions are stored in a refresh buffer, and the screen is refreshed at the rate of 60 frames per second, as in the emissive devices. Back lighting is also commonly applied using solidstate electronic devices, so that the system is not completely dependent on outside light sources. Colors can be displayed by using different materials or dyes and by placing a triad of color pixels at each screen location. Another method for constructing LCDs is to place 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. These devices are called active-matrix displays. Fig 1.8 shows a hand calculator with LCD display

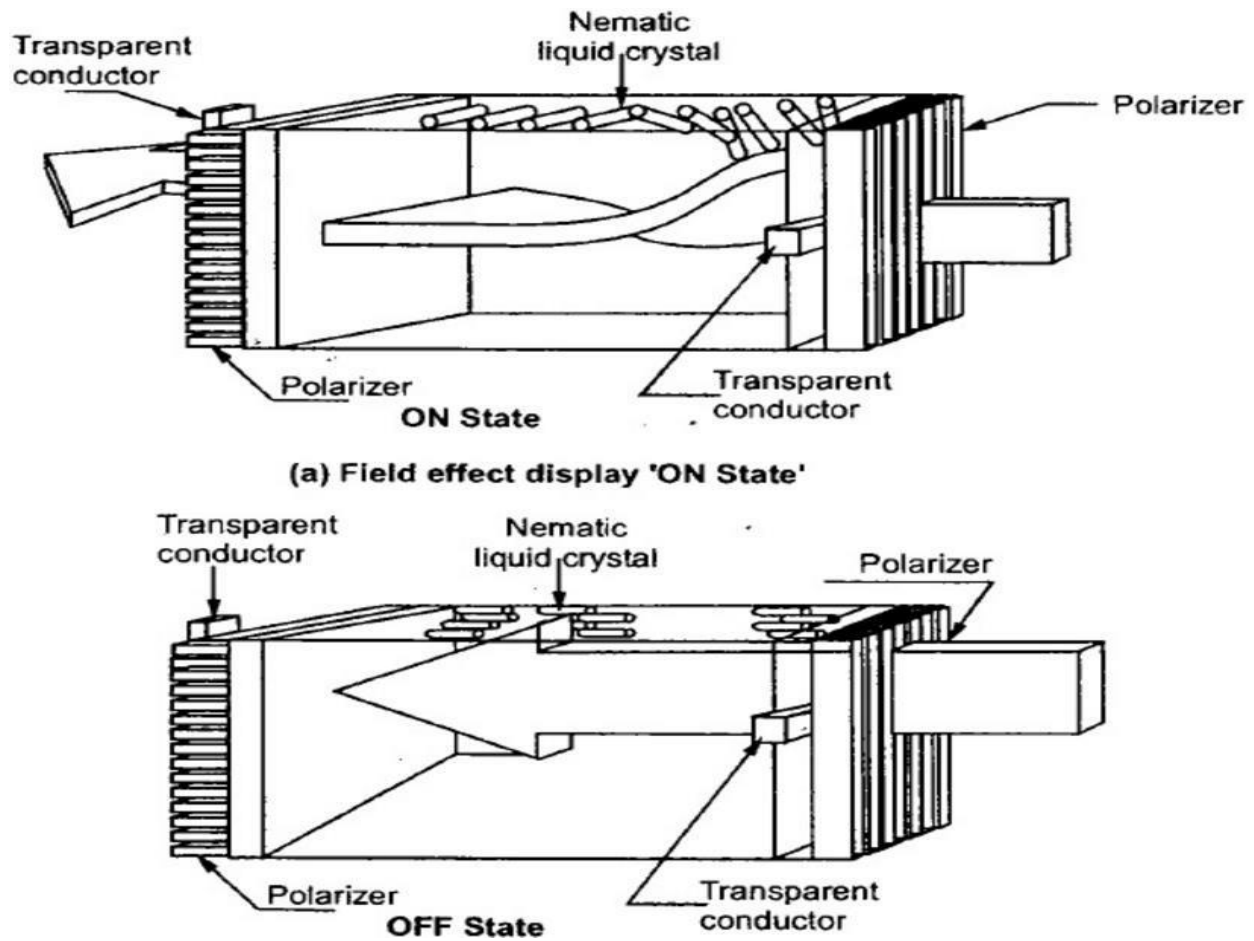**Fig 1.8 Hand calculator with a LCD screen**

Fig 1.9 The light-twisting, shutter effect used in the design of most liquid crystal display devices.

INPUT DEVICES

- Keyboards, Button Boxes, and Dials
- Mouse Devices
- Trackball, Spaceball
- Joystick
- Data Gloves
- Digitizers
- Image Scanners
- Touch Panels
- Light Pens

* Voice Systems

HARD-COPY DEVICES

* Printers
* Plotters

Line Drawing Algorithms

A line connects two points. It is a basic element in graphics. To draw a line, you need two points between which you can draw a line.

The DDA is a scan conversion line algorithm based on calculating either dy or dx. A line issampled at unit intervals in one coordinate and corresponding integer values nearest the line path are determined for other coordinates.

Considering a line with positive slope, if the slope is less than or equal to 1, we sample at unit x intervals (dx=1) and compute successive y values as

$$Y_{k+1} = Y_k + m$$

Subscript k takes integer values starting from 0, for the 1st point and increases by until end point is reached. Y value is rounded off to nearest integer to correspond to a screen pixel. For lines with slope greater than 1, we reverse the role of x and y i.e. we sample at dy=1 and calculate consecutive x values as

$$X_{k+1} = X_k + 1/m$$

Similar calculations are carried out to determine pixel positions along a line with negative slope. Thus, if the absolute value of the slope is less than 1, we set dx=1 if $x_{start} < x_{end}$ i.e. the starting extreme point is at the left.

Digital Differential Analyzer (DDA) line drawing algorithm

Digital Differential Analyzer (DDA) algorithm is the simple line generation algorithm which is explained step by step here.

- Get the input of two end points (X0, Y0) and (X1, Y1).

- Calculate the difference between two end points.

- Based on the calculated difference, you need to identify the number of steps to put pixel. If dx >dy, then you need more steps in x coordinate; otherwise in y coordinate.

- Calculate the increment in x coordinate and y coordinate.

- Put the pixel by successfully incrementing x and y coordinates accordingly and complete the drawing of the line.

Algorithm:

```
void lineDDA (int xa, int ya, int xb, int yb)
{
    int dx = xb - xa, dy = yb - ya, steps, k;
    float xIncrement, yIncrement, x = xa, y = ya;

    if (abs (dx) > abs (dy)) steps = abs (dx);
    else steps = abs  dy);
    xIncrement = dx / (float) steps;
    yIncrement = dy / (float) steps;

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

Advantages:

- It is the simplest algorithm and it does not require special skills for implementation.
- It is a faster method for calculating pixel positions than the direct use of equation y=mx + b. It eliminates the multiplication in the equation by making use of raster characteristics, so that appropriate increments are applied in the x or y direction to find the pixel positions along the line path.

Disadvantages:

- Floating point arithmetic in DDA algorithm is still time-consuming.
- The algorithm is orientation dependent. Hence end point accuracy is poor.

Bresenham's line drawing algorithm

The Bresenham's algorithm is another incremental scan conversion algorithm. The big advantage of this algorithm is that, it uses only integer calculations. Moving across the x axis in unit intervals and at each step choose between two different y coordinates.

For example, as shown in the following illustration, from position (2, 3) you need to choose between (3, 3) and (3, 4). You would like the point that is closer to the original line.
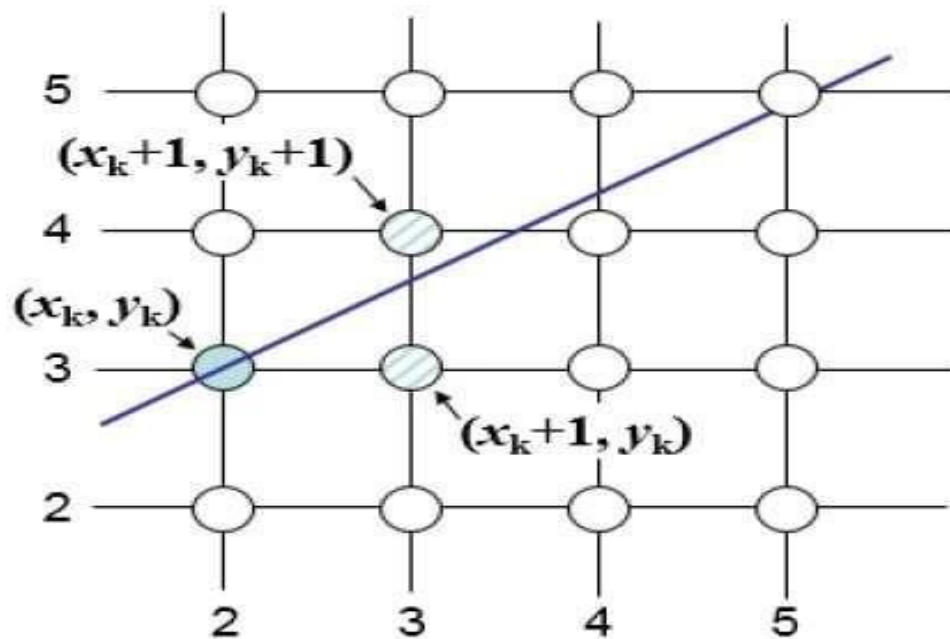

Fig 1.10 Bresenham's line drawing algorithm

At sample position Xk+1,the vertical separations from the mathematical line are labelled as d_{upper} and d_{lower}.
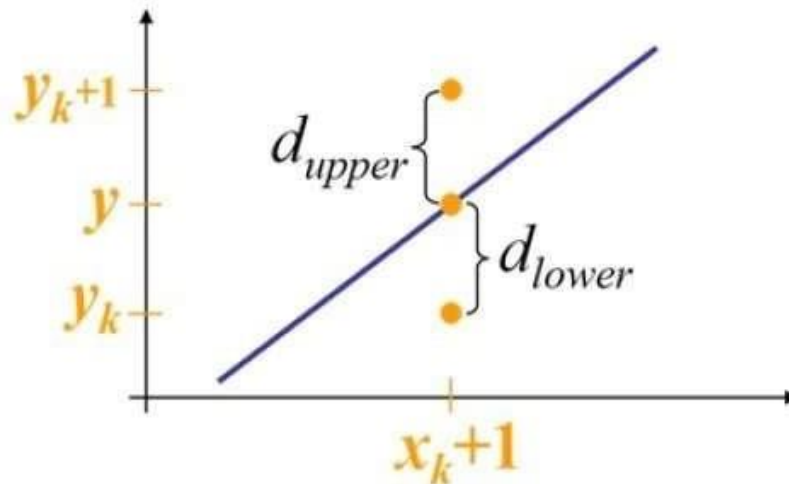
Fig 1.11 Bresenham's line drawing algorithm

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

1. Input the two line endpoints and store the left endpoint in (x_0, y_0).
2. Load (x_0, y_0) into the frame buffer; that is, plot the first point.
3. Calculate constants Δx, Δy, $2\Delta y$, and $2\Delta y - 2\Delta x$, and obtain the starting value for the decision parameter as

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

4. At each x_k along the line, starting at $k = 0$, perform the following test: If $p_k < 0$, the next point to plot is $(x_k + 1, y_k)$ and

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

Otherwise, the next point to plot is $(x_k + 1, y_k + 1)$ and

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

5. Repeat step 4 Δx times.

Algorithm

procedurelineBresenham(xa, ya, xb, yb : integer)
var
dx,dy,x, y, xend, p : integer;
begin
dx=abs(xa-xb);
dy=abs(ya-yb);
  p=2*dy-dx;
 if(xa>xb) then
begin
x=xb;y=yb;xend=xa;
end
else
begin
    x=xa; y=ya;xend=xb;
end;
putpixel(x,y,4);
while(x<=xend) do
begin
    x=x+1;
if (p<0) then
p=p+2*dy;
else
begin
  y=y+1; p=p+2*(dy-dx);
end;
putpixel(x,y,4);
end end

Difference between DDA Line Drawing Algorithm and Bresenhams Line Drawing Algorithm

Table 1 .Difference between DDA and Bresenhams Algorithm

	Digital Differential Analyzer Line Drawing Algorithm	**Bresenham's Line Drawing Algorithm**
Arithmetic	DDA algorithm uses floating points i.e. Real Arithmetic.	Bresenham's algorithm uses fixed points i.e. Integer Arithmetic
Operations	DDA algorithm uses multiplication and division in its operations.	Bresenham's algorithm uses only subtraction and addition in its operations.

Speed	DDA algorithm is rather slowly than Bresenham's algorithm in line drawing because it uses real arithmetic (floating point operations)	Bresenham's algorithm is faster than DDA algorithm in line drawing because it performs only addition and subtraction in its calculation and uses only integer arithmetic so it runs significantly faster.
Accuracy & Efficiency	DDA algorithm is not as accurate and efficient as Bresenham's algorithm	Bresenham's algorithm is more efficient and much accurate than DDA algorithm.
Drawing	DDA algorithm can draw circles and curves but that are not as accurate as Bresenham'salgorithm	Bresenham's algorithm can draw circles and curves with much more accuracy than DDA algorithm.
Round off	DDA algorithm round off the coordinates to integer that is nearest to the line	Bresenham's algorithm does not round but takes the incremental value in its operation.
Expensive	DDA algorithm uses an enormous number of floating-point multiplications so it is expensive	Bresenham's algorithm is less expensive than DDA algorithm as it uses only addition and subtraction.

CIRCLE DRAWING ALGORITHM

(Midpoint Circle drawing algorithm)

A Circle is defined as the set of points that are all at a given distance r from a center position (x_c, y_c).

The distance relationship is expressed by Pythagorean theorem in Cartesian coordinates as ,

$$(x-x_c)^2 - (y-y_c)^2 = r^2,$$

Therefore,

$$y = y_c \pm \sqrt{r^2 - (x_c - x)^2}$$

In parametric polar form,

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

The Circle function is given by,

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

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

The Midpoint Circle algorithm is as follows:

```
void circleMidpoint (int xCenter, int yCenter, int radius)
{
  int x = 0;
  int y = radius;
  int p = 1 - radius;
  void circlePlotPoints (int, int, int, int);

  /* Plot first set of points */
  circlePlotPoints (xCenter, yCenter, x, y);

  while (x < y) {
    x++;
    if (p < 0)
      p += 2 * x + 1;
    else {
      y--;
      p += 2 * (x - y) + 1;
    }
    circlePlotPoints (xCenter, yCenter, x, y);
  }
}

void circlePlotPoints (int xCenter, int yCenter, int x, int y)
{
  setPixel (xCenter + x, yCenter + y);
  setPixel (xCenter - x, yCenter + y);
  setPixel (xCenter + x, yCenter - y);
  setPixel (xCenter - x, yCenter - y);
  setPixel (xCenter + y, yCenter + x);
  setPixel (xCenter - y, yCenter + x);
  setPixel (xCenter + y, yCenter - x);
  setPixel (xCenter - y, yCenter - x);
}
```

Attributes of Output Primitives

Any parameter that affects the way a primitive is to be displayed is referred to as an attribute parameter. Example attribute parameters are color, size etc. A line drawing function for example could contain parameter to set color, width and other properties.

1. Line Attributes

2. Curve Attributes

3. Color and Grayscale Levels

4. Area Fill Attributes

5. Character Attributes

6. Bundled Attributes

Line Attributes

Basic attributes of a straight line segment are its type, its width, and its color. In some graphics packages, lines can also be displayed using selected pen or brush options

* Line Type

* Line Width

* Pen and Brush Options

* Line Color

Line type

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

setLinetype (lt)

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

Line width

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

setLinewidthScaleFactor (lw)

Line width parameter lw is assigned a positive number to indicate the relative width of line to be displayed. A value of 1 specifies a standard width line. A user could set lw to a value of 0.5 to plot a line whose width is half that of the standard line. Values greater than 1 produce lines thicker than the standard.

Line Cap

We can adjust the shape of the **line** ends to give them a better appearance by adding line caps.

There are three types of line cap. They are

* Butt cap

* Round cap

* Projecting square cap

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

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

Projecting square cap extend the line and add butt caps that are positioned one-half of the line width beyond the specified endpoints.
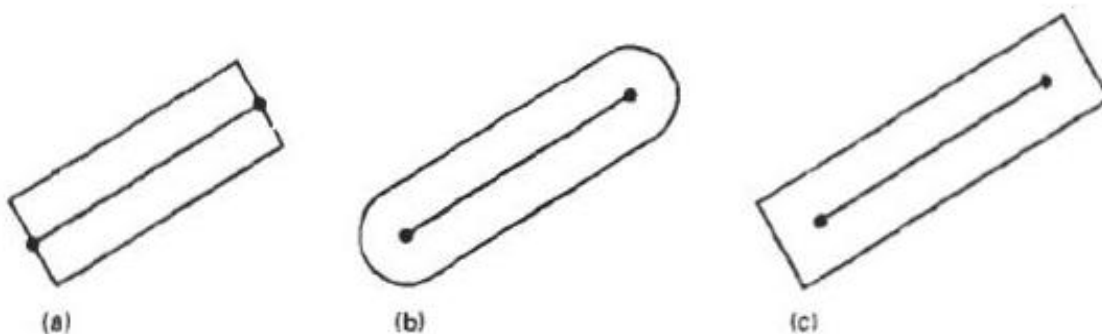
Thick lines drawn with (a) butt caps, (b) round caps, and (c) projecting square caps.

Three possible methods for smoothly joining two line segments
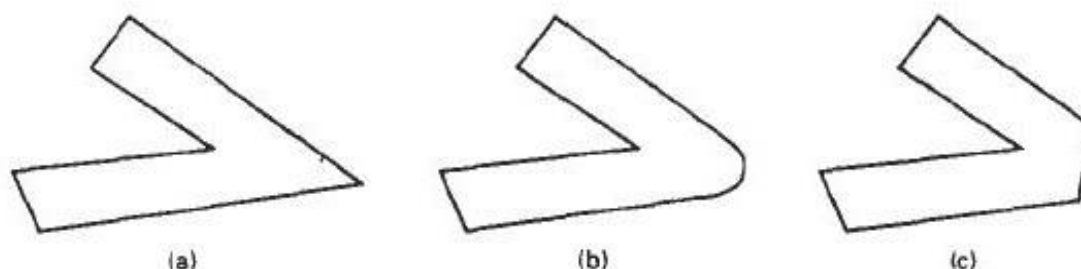
* Mitter Join
* Round Join
* Bevel Join

A miter join accomplished by extending the outer boundaries of each of the two lines

until they meet.

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

A bevel join is generated by displaying the line segment with but caps and filling in tri

angular gap where the segments meet.

(a)   (b)   (c)

Thick line segments connected with (a) miter join, (b) round join, and (c) bevel join.

Pen and Brush Options

With some packages, lines can be displayed with pen or brush selections. Options in this category include shape, size, and pattern. Some possible pen or brush shapes are given in Figure

Custom Document Brushes

Line color

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

We set the line color value in PHIGS with the function

setPolylineColourIndex (lc)

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

Example: Various line attribute commands in an applications program is given by the following sequence of statements

setLinetype(2);

setLinewidthScaleFactor(2);

setPolylineColourIndex (5);

polyline(n1, wc points1);

setPolylineColorIindex(6);

poly line (n2, wc points2);

This program segment would display two figures, drawn with double-wide dashed lines. The first is displayed in a color corresponding to code 5, and the second in color 6.

Curve attributes

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

Color and Grayscale Levels

Various color and intensity-level options can be made available to a user, depending on the capabilities and design objectives of a particular system

In a color raster system, the number of color choices available depends on the amount of storage provided per pixel in the frame buffer

Color-information can be stored in the frame buffer in two ways:

* We can store color codes directly in the frame buffer

* We can put the color codes in a separate table and use pixel values as an index into this table

With the direct storage scheme, whenever a particular color code is specified in an application program, the corresponding binary value is placed in the frame buffer for each-component pixel in the output primitives to be displayed in that color.

A minimum number of colors can be provided in this scheme with 3 bits of storage per pixel, as shown in Table

3 bits - 8 choice of color

6 bits – 64 choice of color

8 bits – 256 choice of color

A user can set color-table entries in a PHIGS applications program with the function

setColourRepresentation (ws, ci, colorptr)

Parameter **ws** identifies the workstation output device; parameter ci specifies the color index, which is the color-table position number (**0** to **255**) and parameter colorptr points to a trio of RGB color values (**r, g, b**) each specified in the range from **0** to 1

Grayscale

With monitors that have no color capability, color functions can be used in an application program to set the shades of gray, or grayscale, for displayed primitives. Numeric values over the range from 0 to 1 can be used to specify grayscale levels, which are then converted to appropriate binary codes for storage in the raster.

INTENSITY CODES FOR A FOUR-LEVEL GRAYSCALE SYSTEM		
Intensity Codes	Stored Intensity Values In The Frame Buffer (Binary Code)	Displayed Grayscale
0.0	0 (00)	Black
0.33	1 (01)	Dark gray
0.67	2 (10)	Light gray
1.0	3 (11)	White

Intensity = 0.5[min(r, g, b) + max(r, g, b)]

Area fill Attributes

Options for filling a defined region include a choice between a solid color or a pattern fill and choices for particular colors and patterns

Fill Styles

Areas are displayed with three basic fill styles: hollow with a color border, filled with a solid color, or filled with a specified pattern or design. A basic fill style is selected in a PHIGS program with the function

<div align="center">setInteriorStyle (fs)</div>

Values for the fill-style parameter fs include hollow, solid, and pattern. Another value for fill style is hatch, which is used to fill an area with selected hatching patterns-parallel lines or crossed lines

| Hollow | Solid | Patterned |

Diagonal Hatch Fill

Diagonal Cross-Hatch Fill
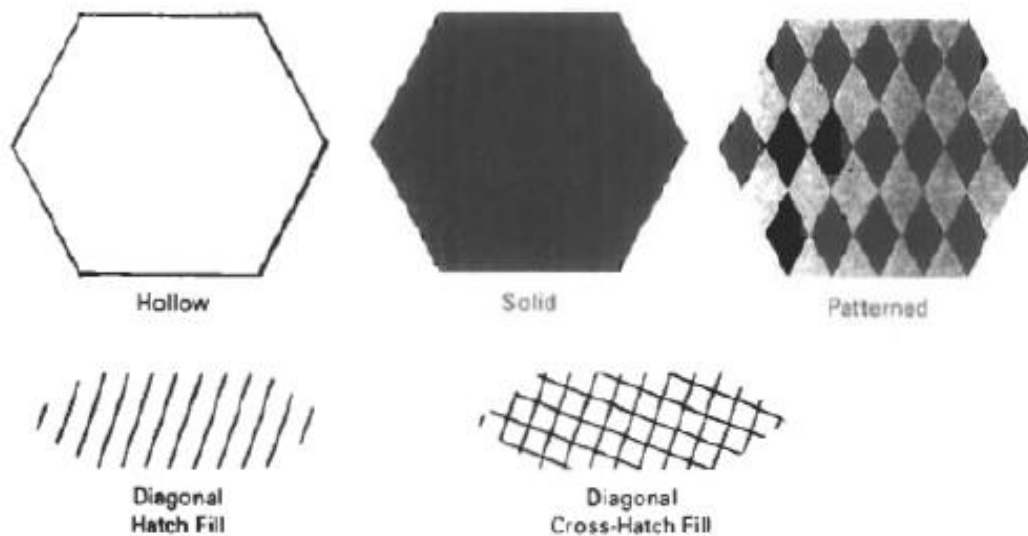
The color for a solid interior or for a hollow area outline is chosen with where fill color parameter fc is set to the desired color code

<div align="center">setInteriorColourIndex (fc)</div>

Pattern Fill

We select fill patterns with setInteriorStyleIndex (pi) where pattern index parameter pi specifies a table position

For example, the following set of statements would fill the area defined in the fillArea command with the second pattern type stored in the pattern table:

SetInteriorStyle(pattern)

SetInteriorStyleIndex(2);

Fill area (n, points)

Index (pi)	Pattern (cp)
1	$\begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}$
2	$\begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 1 \\ 2 & 1 & 2 \end{bmatrix}$

Character Attributes

The appearance of displayed character is controlled by attributes such as font, size, color and orientation. Attributes can be set both for entire character strings (text) and for individual characters defined as marker symbols

Text Attributes

The choice of font or type face is set of characters with a particular design style as courier, Helvetica, times roman, and various symbol groups.

The characters in a selected font also be displayed with styles. (solid, dotted, double) in **bold face** in **italics**, and in **outline** or shadow styles.

A particular font and associated stvle is selected in a PHIGS program by setting an

integer code for the text font parameter tf in the function

setTextFont (tf)

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

setTextColourIndex (tc)

Where text color parameter tc specifies an allowable color code.

Text size can be adjusted without changing the width to height ratio of characters with

setCharacterHeight (ch)

Height 1

Height 2

Height 3

Parameter ch is assigned a real value greater than 0 to set the coordinate height of capital letters

The width only of text can be set with function.

setCharacterExpansionFactor (cw)

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

width 0.5

width 1.0

width 2.0

Spacing between characters is controlled separately with

setCharacterSpacing (cs)

Where the character-spacing parameter cs can he assigned any real value

Spacing 0.0

Spacing 0.5

Spacing 1.0

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

setCharacterUpVector (upvect)

Parameter upvect in this function is assigned two values that specify the x and y

vector components. For example, with upvect = (1, 1), the direction of the up vector is 45o and text would be displayed as shown in Figure.

Up Vector
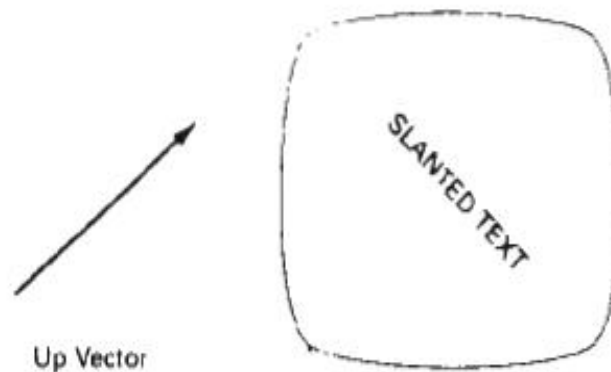
To arrange character strings vertically or horizontally

setTextPath (tp)

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

gnirts   string

Another handy attribute for character strings is alignment. This attribute specifies how text is to be positioned with respect to the $tart coordinates. Alignment attributes are set with

setTextAlignment (h,v)

where parameters h and v control horizontal and vertical alignment. Horizontal alignment is set by assigning h a value of left, center, or right. Vertical alignment is set by assigning v a value of top, cap, half, base or bottom.

A precision specification for text display is given with

<div align="center">

setTextPrecision (tpr)

</div>

tpr is assigned one of values string, char or stroke.

Marker Attributes

A marker symbol is a single character that can he displayed in different colors and in different sizes. Marker attributes are implemented by procedures that load the chosen character into the raster at the defined positions with the specified color and size. We select a particular character to be the marker symbol with

<div align="center">

setMarkerType (mt)

</div>

where marker type parameter mt is set to an integer code. Typical codes for marker type are the integers 1 through 5, specifying, respectively, a dot (.) a vertical cross (+), an asterisk (*), a circle (o), and a diagonal cross (X).

We set the marker size with

<div align="center">

setMarkerSizeScaleFactor (ms)

</div>

with parameter marker size ms assigned a positive number. This scaling parameter is applied to the nominal size for the particular marker symbol chosen. Values greater than 1 produce character enlargement; values less than 1 reduce the marker size.

Marker color is specified with

<div align="center">

setPolymarkerColourIndex (mc)

</div>

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

Bundled Attributes

The procedures considered so far each function reference a single attribute that specifies exactly how a primitive is to be displayed these specifications are called individual attributes.

A particular set of attributes values for a primitive on each output device is chosen by specifying appropriate table index. Attributes specified in this manner are called bundled attributes. The choice between a bundled or an unbundled specification is made by setting a switch called the aspect source flag for each of

these attributes

<div align="center">**setIndividualASF(attributeptr, flagptr)**</div>

where parameter attributer ptr points to a list of attributes and parameter flagptr points to the corresponding list of aspect source flags. Each aspect source flag can be assigned a value of individual or bundled.

Bundled line Attributes

Entries in the bundle table for line attributes on a specified workstation are set with the function

<div align="center">**setPolylineRepresentation (ws, li, lt, lw, lc)**</div>

Parameter ws is the workstation identifier and line index parameter li defines the bundle table position. Parameter lt, lw, tc are then bundled and assigned values to set the line type, line width, and line color specifications for designated table index.

Example

<div align="center">**setPolylineRepresentation (1, 3, 2, 0.5, 1)**</div>

<div align="center">**setPolylineRepresentation (4, 3, 1, 1, 7)**</div>

A poly line that is assigned a table index value of 3 would be displayed using dashed lines at half thickness in a blue color on work station 1; while on workstation 4, this same index generates solid, standard-sized white lines

Bundle area fill Attributes

Table entries for bundled area-fill attributes are set with

<div align="center">**setInteriorRepresentation (ws, fi, fs, pi, fc)**</div>

Which defines the attributes list corresponding to fill index fi on workstation ws. Parameter fs, pi and fc are assigned values for the fill style pattern index and fill color.

Bundled Text Attributes

<div align="center">**setTextRepresentation (ws, ti, tf, tp, te, ts, tc)**</div>

Bundles values for text font, precision expansion factor size an color in a table position for work station ws that is specified by value assigned to text index parameter ti.

Bundled marker Attributes

setPolymarkerRepresentation (ws, mi, mt, ms, mc)

That defines marker type marker scale factor marker color for index mi on workstation ws.

Inquiry functions

Current settings for attributes and other parameters as workstations types and status in the system lists can be retrieved with inquiry functions.

inquirePolylineIndex (lastli)

and

inquireInteriorcColourIndex (lastfc)

Copy the current values for line index and fill color into parameter lastli and lastfc.

Output Primitive Type	Associated Attributes	Attribute-Setting Functions	Bundled-Attribute Functions
		SUMMARY OF ATTRIBUTES	
Line	Type Width Color	setLinetype setLineWidthScaleFactor setPolylineColourIndex	setPolylineIndex setPolylineRepresentation
Fill Area	Fill Style Fill Color Pattern	setInteriorStyle setInteriorColorIndex setInteriorStyleIndex setPatternRepresentation setPatternSize setPatternReferencePoint	setInteriorIndex setInteriorRepresentation
Text	Font Color Size Orientation	setTextFont setTextColourIndex setCharacterHeight setCharacterExpansionFactor setCharacterUpVector setTextPath setTextAlignment	setTextIndex setTextRepresentation
Marker	Type Size Color	setMarkerType setMarkerSizeScaleFactor setPolymarkerColourIndex	setPolymarkerIndex setPolymarkerRepresentation

Antialiasing

Antialiasing is the smoothing of the image or sound roughness caused by aliasing . With images, approaches include adjusting pixel positions or setting pixel intensities so that there is a more gradual transition between the color of a line and the background color. With sound, aliases are removed by eliminating frequencies above half the sampling frequencies (Fig. 1.11).

Side effects of Scan Conversion

The most common side effects when working with raster devices are:

- Unequal Intensity
- Overstrike
- Aliasing

Unequal Intensity

- Human perception of light is dependent on density and intensity of light source.Thus a raster display with perfect squareness, a diagonal line of pixels will appear dimmer that a horizontal or vertical line.

Solution: By increasing the number of pixels on diagonal lines.

Overstrike:

- The same pixel is written more than once.
- This results in intensified pixels in case of photographic media, such as slide or transparency.

Solution: Check each pixel to see whether it has already been written to prior to writing a new point.

Aliasing

- The effect created when rasterization is performed over a discrete series of pixels.
- In particular, when lines or edges do not necessarily align directly with arrow or column of pixels, that line may appear unsmooth and have a stair-step edge appearance.
- Jagged appearance of curves or diagonal lines on a display screen, which is caused by low screen resolution.
- Refers to the plotting of a point in a location other than its true location in order to fit the point into the raster.
- Consider equation $Y=mX+b$, For $m=0.5$, $b=1$, $X=3$ -------- Y would be 2.5. So the point (3, 2.5) is plotted at alias location (3,3).

Anti-Aliasing

Fig 1.11. The image on the right shows the result of anti-aliasing through the use of higher resolution.

Anti-Aliasing

- Antialiasing utilizes blending techniques to blur the edges of the lines and provide the viewer with the illusion of a smoother line.

- Two general approaches:

- **Super-sampling**
  - samples at higher resolution, then filters down the resulting image
  - Sometimes called post-filtering
  - The prevalent form of anti-aliasing in hardware

- **Area sampling**
  - sample primitives with a box (or Gaussian, or whatever) rather than spikes
  - Requires primitives that have area (lines with width)
  - Sometimes referred to as pre-filtering

Super-sampling

- Sample at a higher resolution than required for display, and filter image down

- 4 to 16 samples per pixel is typical

- Samples might be on a uniform grid, or randomly positioned, or other variants

- Divide each pixel into sub-pixels.

- The number of intensities are the max number of sub-pixels selected on the line segment within a pixel.

- The intensity level for each pixel is proportional to the number of sub-pixels inside the polygon representing the line area.

- Line intensity is distributed over more pixels.
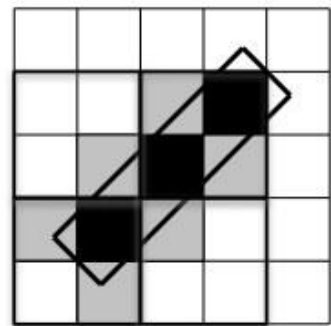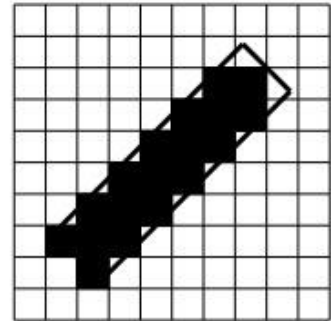
Area Sampling

- Determine the percentage of area coverage for a screen pixel, then set the pixel intensity proportional to this percentage.
- Consider a line as having thickness.
- Consider pixels as little squares.
- Unweighted area sampling: Fill pixels according to the proportion of their square covered by the line.
- Weighted area sampling: Weight the contribution according to where in the square the primitives falls. Fig 1.12 shows an illustration of area sampling
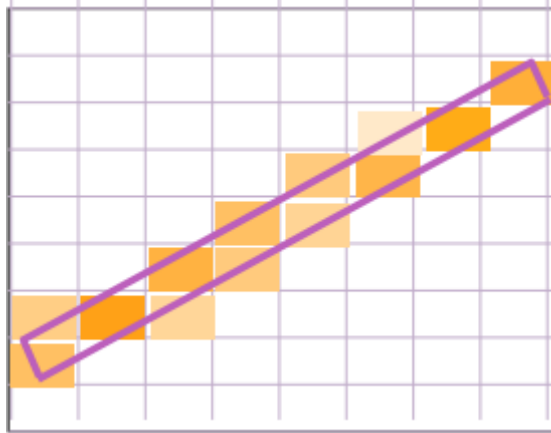
Fig 1.12 Area Sampling

Unweighted Area Sampling

- Primitive cannot affect intensity of pixel if it does not intersect the pixel.
- Equal areas cause equal intensity, regardless of distance from pixel center to area.
- Unweighted sampling colors two pixels identically when the primitive cuts the same area through the two pixels.
- Intuitively, pixel cut through the center should be more heavily weighted than one cut along corner. Fig 1.13 shows unweighted sampling

Fig 1.13 Unweighted sampling

Weighted Area Sampling

- weight the subpixel contributions according to position, giving higher weights to the central subpixels.

- weighting function, W(x,y)
    - specifies the contribution of primitive passing through the point (x, y) from pixel center
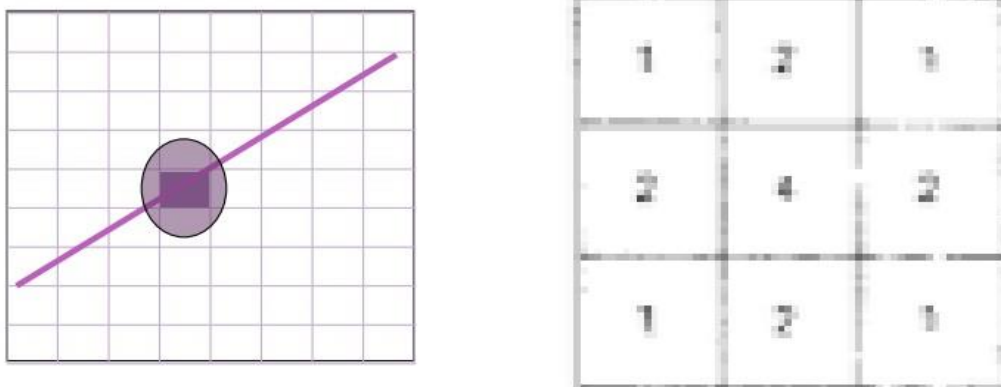
Fig 1.14 Weighting function

Filtering Techniques

- A continuous weighting surface (or filter function) covering the pixel.
- Applying the filter function by integrating over the pixel surface to obtain the weighted average intensity
- Weighted (Filter) Function

- Determines the influence on the intensity of a pixel of a given small area dA of a primitive.

- This function is constant for unweighted and decreases with increasing distance for weighted.

- Total intensity is the integral of the weighting (filter) function over the area of overlap.

- Ws is the volume (always between 0 and 1)

- I=Imax·Ws

- Box, Cone and Gaussiean
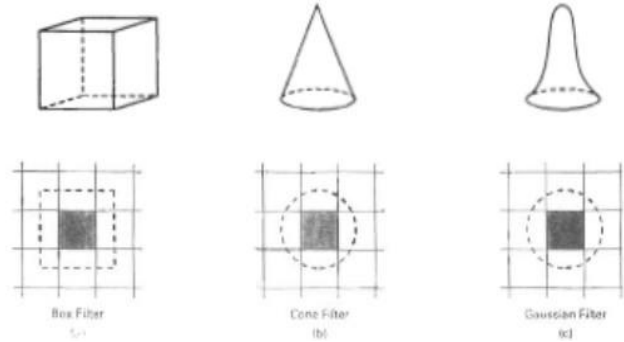
Fig 1.15 Filtering Techniques

http://www.graphics.cornell.edu/online/tutorial/

https://www.cs.utexas.edu/~fussell/courses/cs324e2003/hear50265_ch03.pdf

http://www.slideshare.net/rajeshkamboj/computer-graphics-notes-btech-kuk-mdu

http://www.slideshare.net/rhspcte/computer-graphics-ebookhearn-baker

UNIT-II - SCSA1503- Computer Graphics and Multimedia Applications

Basic two dimensional transformations – Other transformations – 2D and 3D viewing – Line clipping – Polygon clipping – Logical classification – Input functions – Interactive picture construction techniques.

2D Transformations and Viewing

2-Dimensional Transformations

The Basic Transformations:

1. Translation

2. Scaling

3. Rotation

Other Transformations:

4. Reflection

5. Shearing

1. Translations

Displacement of an object in a given distance and direction from its original position.

- Rigid body transformation that moves object without deformation

- Initial Position point P (x, y)

- The new point P' (x', y')

where

x' = x + tx , y' = y + ty , tx and ty is the displacement in x and y respectively.(Fig.2.1)
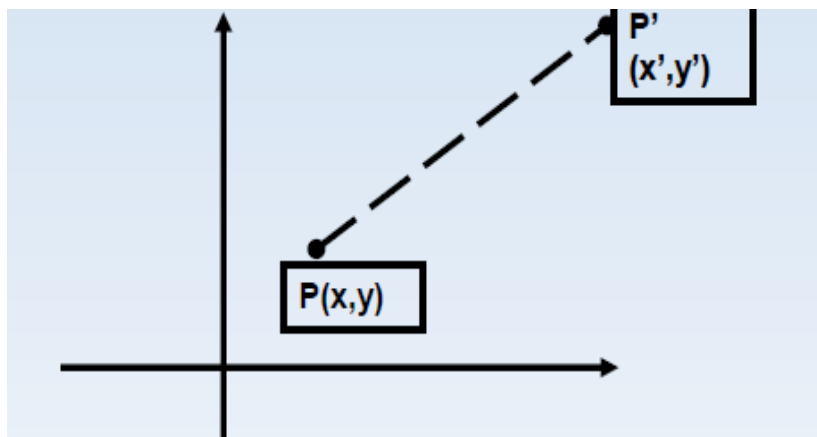
Fig. 2.1 Translation

The translation pair (tx, ty) is called a translation vector or shift vector.

2

Problem:

- Assume you are given a point at (x,y)=(2,1). Where will the point be if you move it 3 units to the right and 1 unit up? Ans: (x',y') = (5,2). How was this obtained? - (x',y') = (x+3,y+1). That is, to move a point by some amount *dx* to the right and *dy up*, you must add *dx* to the x-coordinate and add *dy* to the y-coordinate.(Fig.2.2)
- What was the required transformation to move the green triangle to the red triangle? Here the green triangle is represented by 3 points

triangle = { p1=(1,0), p2=(2,0), p3=(1.5,2) }
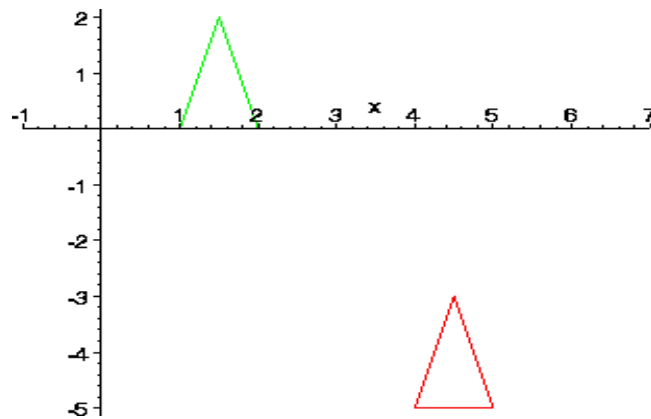
Fig. 2.2 Translation example

Matrix/Vector Representation of Translations

A translation can also be represented by a pair of numbers, $t=(t_x,t_y)$ where t_x is the change in the x-coordinate and t_y is the change in y coordinate. To translate the point p by t, we simply add to obtain the new (translated) point

$$p' = p + t.$$

$$\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} tx \\ ty \end{bmatrix} = \begin{bmatrix} x + tx \\ y + ty \end{bmatrix}$$

3

2. Rotation

Rotation is applied to an object by repositioning it along a circular path in the xy plane.

To generate a rotation, we specify

- Rotation angle θ
- Pivot point (xr , yr)

Positive values of θ for counterclockwise rotation

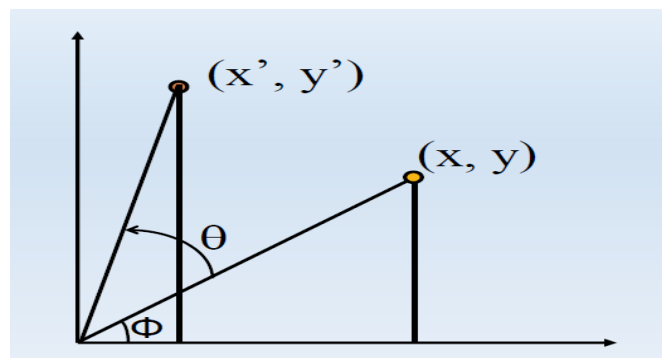Negative values of θ for clockwise rotation.(Fig.2.3)

Fig. 2.3 Rotation

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

Trig Identity...
$$x' = r \cos(\phi) \cos(\theta) - r \sin(\phi) \sin(\theta)$$
$$y' = r \sin(\phi) \sin(\theta) + r \cos(\phi) \cos(\theta)$$

Substitute...
$$x' = x \cos(\theta) - y \sin(\theta)$$
$$y' = x \sin(\theta) + y \cos(\theta)$$

Matrix Representation: P'=R.P

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

3. Scaling

- Scaling alters the size of an object.
- Operation can be carried out by multiplying each of its components by a scalar
- Uniform scaling means this scalar is the same for all components
- Non-uniform scaling: different scalars per component

x' = x* sx
y' = y * sy

In matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} sx & 0 \\ 0 & sy \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}$$

4. Reflection

A reflection is a transformation that produces a mirror image of an object
Generated relative to an axis of reflection
1. Reflection along x axis(Fig.2.4)
2. Reflection along y axis(Fig.2.5)
3. Reflection relative to an axis perpendicular to the xy plane and passing through the coordinate origin(Fig.2.6)
4. Reflection of an object relative to an axis perpendicular to the xy plane and passing through point P
5. Reflection of an object with respect to the line y=x.(Fig.2.7)

Reflection about x-axis:
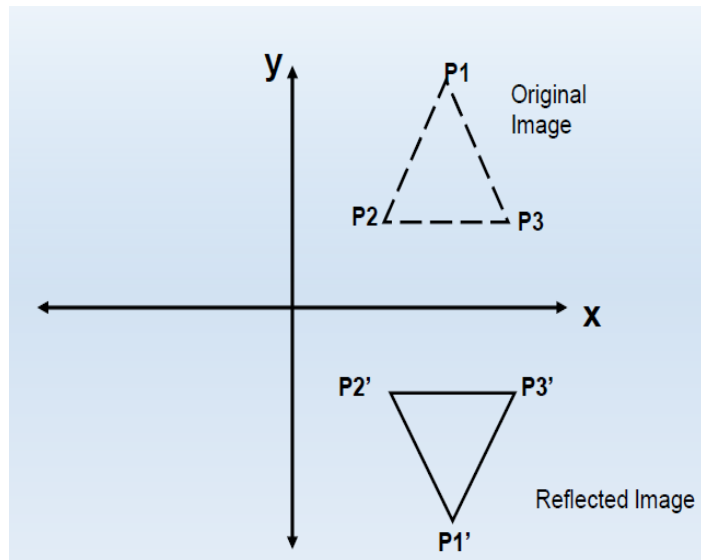
Fig. 2.4 Reflection about x-axis

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
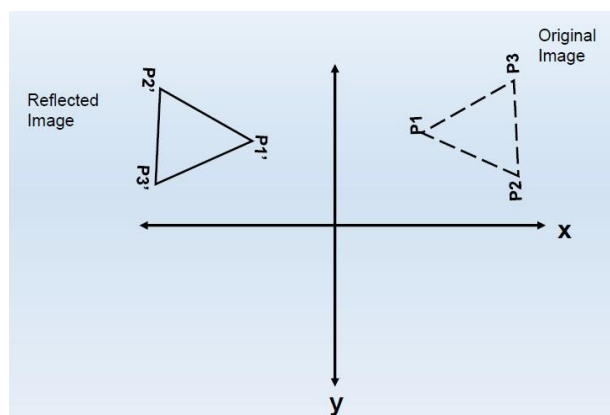
Reflection about y-axis:

Fig. 2.5 Reflection about y-axis

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

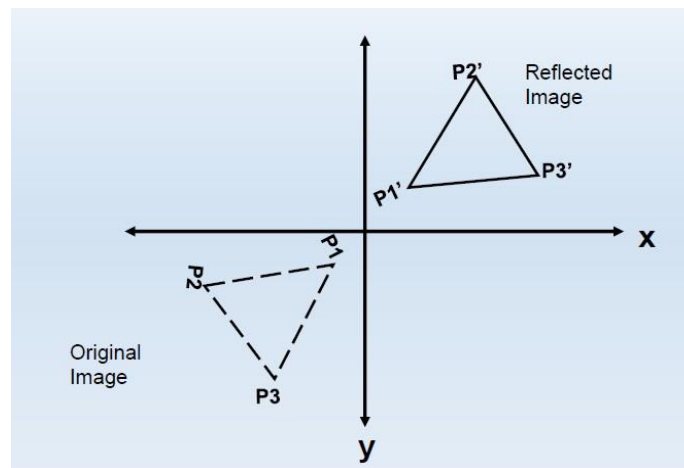**Reflection relative to an axis perpendicular to the xy plane and passing through the coordinate origin:**

Fig. 2.6 Reflection about xy plane

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

Reflection of an object with respect to the line y=x

$$M = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
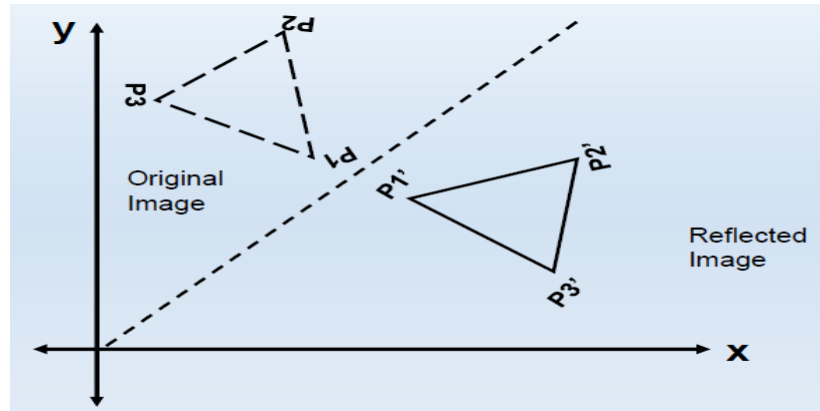
Fig. 2.7 Reflection about line y=x

5. Shearing

A transformation that distorts the shape of an object such that the transformed object appears as if the object were composed of internal layers that had been caused to slide over each other.

Shear relative to the x-axis

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

Shear relative to the y-axis

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

2D VIEWING

The mapping of a 2D world coordinate system to device coordinates is called a two-dimensional viewing transformation.

The **clipping window** is the section of the 2D scene that is selected for viewing.

The **display window** is where the scene will be viewed.

The **viewport** controls the placement of the scene within the display window

A window-viewport transformation describes the mapping of a (rectangular) window in one coordinate system into another (rectangular) window in another coordinate system. This transformation is defined by the section of the original image that is transformed (clipping window), the location of the resulting window (viewport), and how the window is translated, scaled or rotated. (Fig.8)
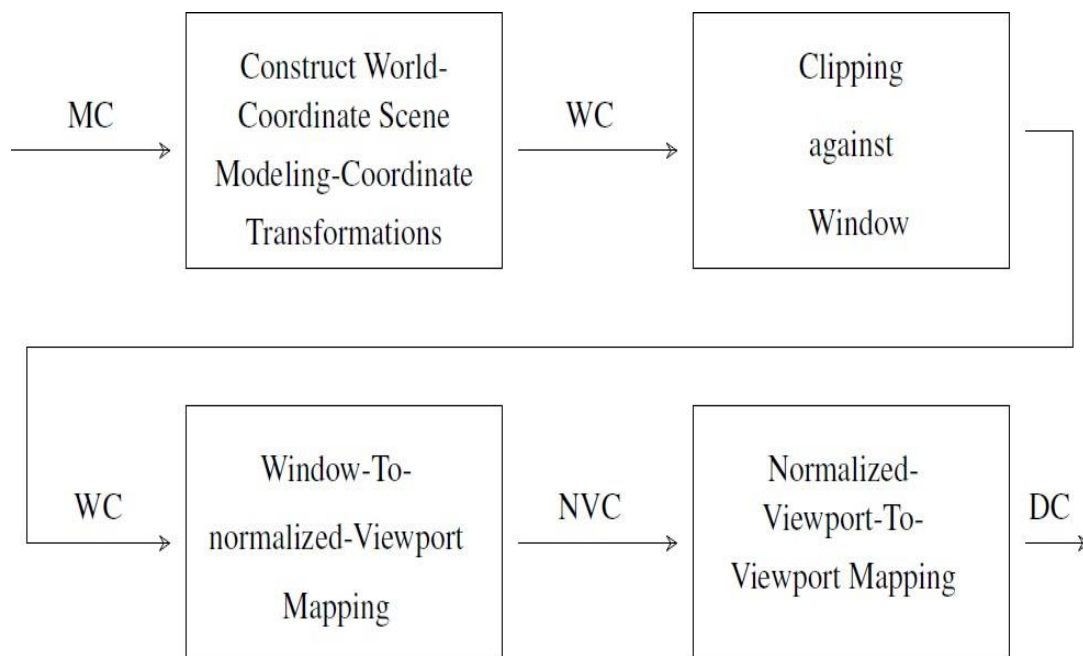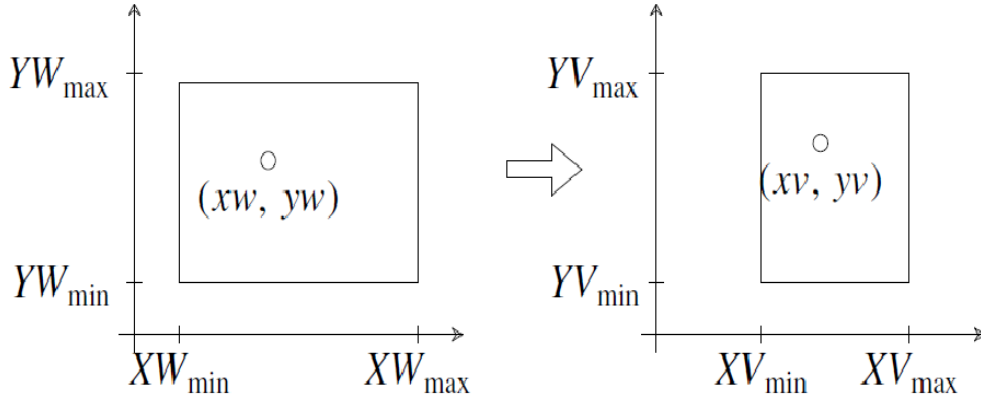

Fig. 2.8 Window Co-ordinate to Viewport Co-ordinate Transformation

$$xv = xv_{min} + (xw - xw_{min})sx$$

$$yv = yv_{min} + (yw - yw_{min})sy$$

sx and sy are scaling factors,

$$sx = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}}, \qquad sy = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}}$$

Cohen-Sutherland Line Clipping Algorithm

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 rejected and needs to be neither clipped nor displayed.(Fig.9)
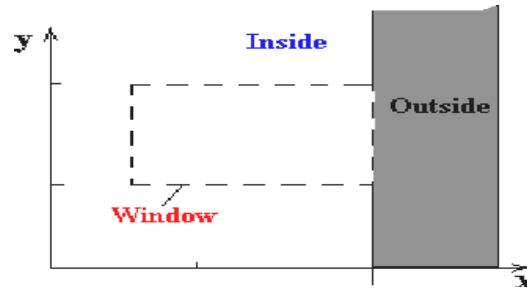
Fig. 2.9 Cohen Sutherland window Clipping

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.

Fig. 2.10 4 Bit Code Representation(LRBT)

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

The sequence for reading the codes' bits is LRBT (Left, Right, Bottom, Top).(Fig.10)

11

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 trivally 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 trivally 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 trivally accepted. For example, if the endpoint codes are 0000 and 0000, the logical OR is 0000 - the line can be trivally accepted. If the endpoint codes are 0000 and 0110, the logical OR is 0110 and the line cannot be trivally 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 trivally 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 p1=(x0,y0) and p2=(x1,y1)
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 p1=(x0,y0) . 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 <u>intersection</u> I of the corresponding window edge with the line from P_1 to P_2. Replace P_1 with I and repeat the algorithm.

Example illustration
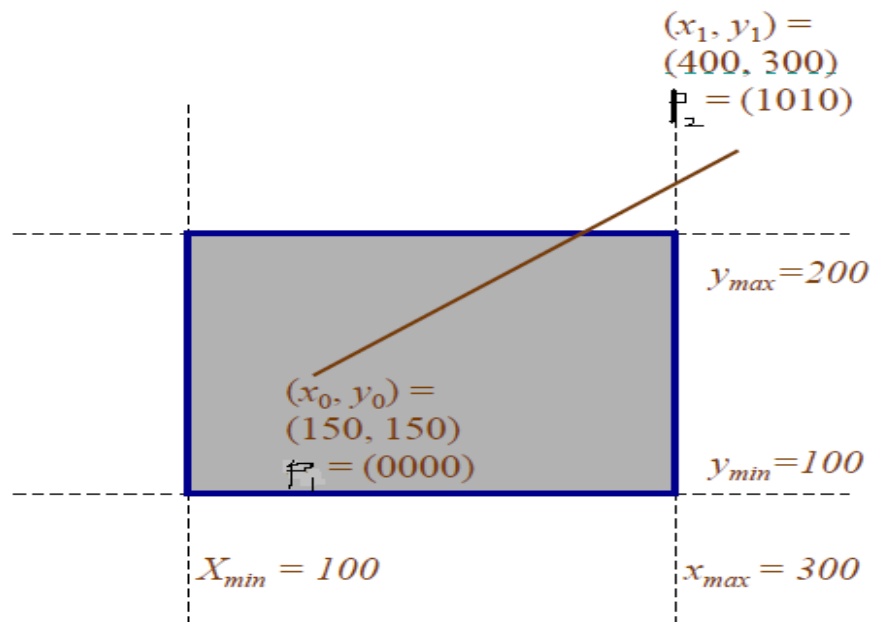
Fig. 2.11 Cohen Sutherland Window Clipping Example

For p1(150,150), (From Right to left)
First bit: x – xmin = 150 – 100 = 50 = set the bit to 0
Second bit: xmax – x = 300 – 150 = 150 =set the bit to 0
Third bit: y – ymin = 150 – 100 =50 = set the bit to 0
Fourth bit: ymax – y = 200 – 150 = 50 = set the bit to 0
So the region code for P1 is 0000
For p2(400,300),   (From Right to left)
First bit: x – xmin = 400 -100 = 300 = set the bit to 0

Second bit: xmax – x = 300 – 400 = - 100 = set the it to 1
Third bit: y – ymin = 300 – 300=0= set the bit to 0
Fourth bit: ymax – y = 200 – 300 = -100 = set the bit to 1.
So the region code for P2 is 1010
P1 is inside and p2 is in the top and above region code(Fig.11)
To find the intersection point:

$$y = y_1 + m \left(x_{boundary} - x_1 \right)$$

Where $x_{boundary}$ is set either to xmin or xmax

$$x = x_1 + \left(y_{boundary} - y_1 \right) / m$$

Where $y_{boundary}$ is set either to ymin or ymax

Polygon Clipping

An algorithm that clips a polygon is rather complex. Each edge of the polygon must be tested against each edge of the clipping window, usually a rectangle. As a result, new edges may be added, and existing edges may be discarded, retained, or divided. Multiple polygons may result from clipping a single polygon. We need an organized way to deal with all of these cases.

SUTHERLAND HODGEMAN POLYGON CLIPPING

The Sutherland and Hodgman's polygon clipping algorithm is used in this tool. This algorithm is based on a divide-and-conquer strategy that solves a series of simple and identical problems that, when combined, solve the overall problem. The simple problem is to clip a polygon against a single infinite clipping edge. This process outputs the series of vertices that define the clipped polygon. Four clipping edges, each defining one boundary of the clipping window, are used to successively to fully clip the polygon. (Fig. 2.12)
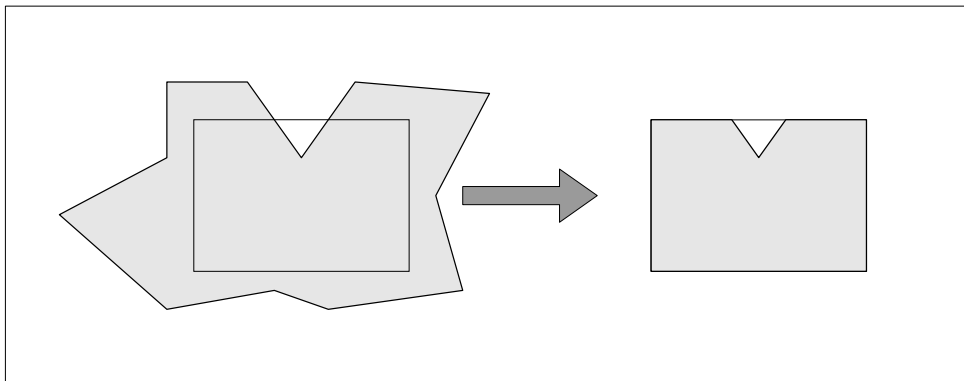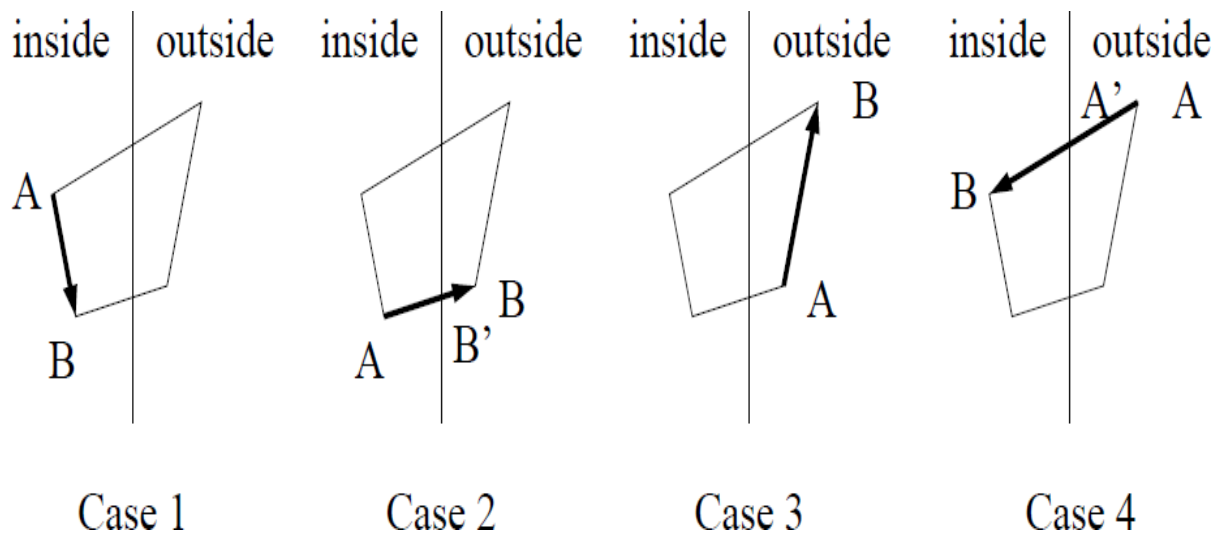

Fig. 2.12 Sutherland Hodgeman Clip Window

inside | outside   inside | outside   inside | outside   inside | outside

Case 1   Case 2   Case 3   Case 4

Assuming vertex A has already been processed,

Case 1 — vertex B is added to the output list

Case 2 — vertex B' is added to the output (edge AB is clipped to AB')

Case 3 — no vertex added (segment AB clipped out)

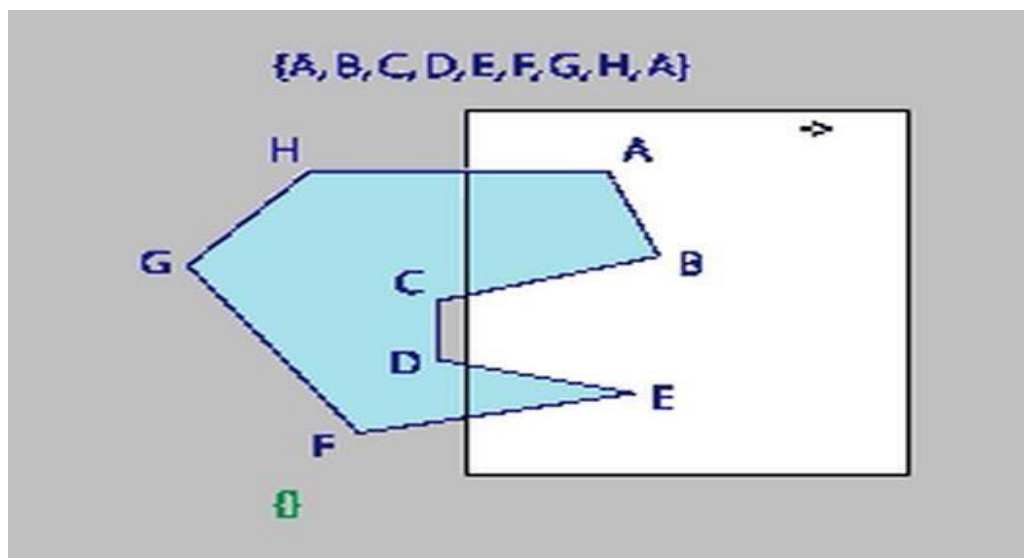Case 4 — vertices A' and B are added to the output

Sample Polygon:

{A, B, C, D, E, F, G, H, A}

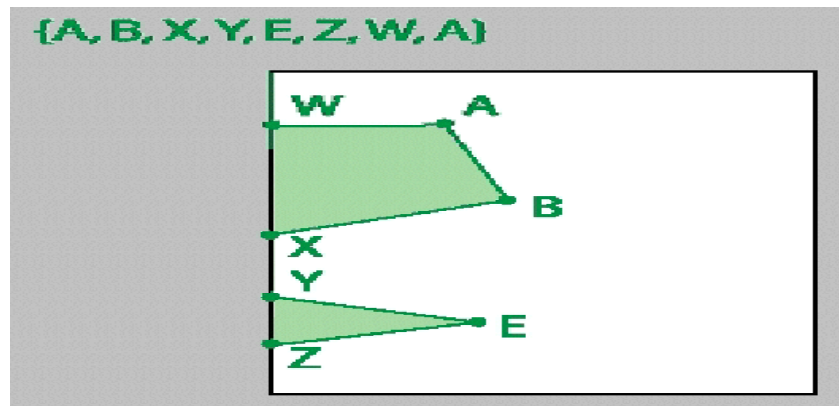**Fig. 2.13 Before Clipping Polygon**

After Clipping

Fig. 2.14 After Clipping Polygon

Drawback of Sutherland Hodgeman Algorithm:

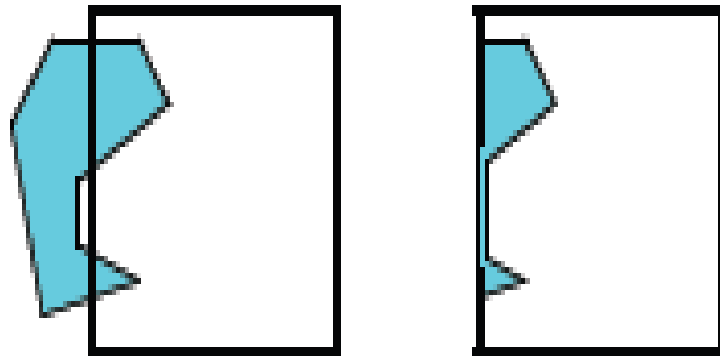Clipping of the concave polygon → can produce two CONNECTED areas

Fig. 2.15 Drawback of Sutherland hodgeman

Logical Classification of Input Devices

To make the graphical package independent of the hardware devices.

Input functions are designed based on the type of input data.

- Locator
- Stroke
- Valuator
- choice
- pick

Locator Devices:

- To select a co-ordinate on the screen.
- When the cursor is at the desired position on the screen, a button is clicked to select that co-ordinate point.
- Mouse, joystick, trackballs, space balls, digitizers.. Can be used as locator devices.
- Keyboard- along with the four , additional four buttons are provided to move horizontally.
- Keyboard along with the above can also be used

Stroke Device:

- A sequence of co-ordinate points can be selected.
- Locator devices in continuous mode.
- Graphical tablet or digitizer can also be used.

Valuator Devices:

- To give scalar input values like temperature and voltage levels.
- Floating point numbers within any range can be given as input
- E.g.: Set of Control dials.
- Rotation in one direction increases the values..
- Slide-potentiometers, keyboards with a set of numeric keys can also be used.
- Display sliders, buttons, rotating scales and menus on screen.

Choice Devices:

- To construct a picture.
- Selection from a list or menu of alternatives.
- Either a keyboard or stand-alone "button box" can be used for selecting item from the menu.
- The selected screen position(x,y) is compared with the menu option.
- Touch panels are also used.

Pick Devices:

- To select parts of the screen that are to be edited.
- Choice devices can be used here.

- If the selected point lies in the bounding rectangle of a particular object, then that'll be selected.
- If it lies in the area of 2 objects, then the squared distance from the point to the line segment both the objects are compared.

When triggered, input devices return information (their measure) to the system

Mouse - returns position information

Keyboard - returns ASCII code

Input Modes

The different modes are:

Request mode:

- The measure of the device is not returned to the program until the device is triggered

Sample mode

- Measure is returned immediately after the function is called in the user program (device sample).
- No trigger needed
- Useful in apps where the program guides the user

Event mode

- Can handle multiple inputs
- When device triggered an event is generated
- Identifier for device placed in the event queue
- Event queue process is independent of the application, asynchronous

The input class function can be given by:

SetMode(ws,deviceCode,inputMode,echoFlag)

ASSIGNMENT OF INPUT-DEVICE CODES

Device Code	Physical Device Type
1	Keyboard
2	Graphics Tablet
3	Mouse
4	Joystick
5	Trackball
6	Button

setLocatorMode(1,2,sample,noecho)

setTextMode(2,1,request,echo)

setPickMode(4,3,event,echo)

The Request mode can be given by:

request…(ws,deviceCode,status, …)

Locator and Stroke Input in Request Mode:

The request functions for these two logical input classes are:

requestLocator (ws, devCode, status, viewIndex, pt)

requestLocator (ws, devCode, nMax, status, viewIndex, n, pts)

Where nmax is the maximum number of points that can go in the input list

viewIndex is assigned a two dimensional view index number. Lower this value higher is the priority.

<u>Interactive Picture Construction Techniques</u>

Interactive picture- construction methods are commonly used in variety of applications, including design and painting packages. These methods provide user with the capability to position objects, to constrain fig. to predefined orientations or alignments, to sketch fig., and to drag objects around the screen. Grids, gravity fields, and rubber band methods are used to aid in positioning and other picture construction operations. The several techniques used for interactive picture construction that are incorporated into graphics packages are:

(1) Basic positioning methods:- coordinate values supplied by locator input are often used with positioning methods to specify a location for displaying an object or a character string. Coordinate positions are selected interactively with a pointing device, usually by positioning the screen cursor.

(2) constraints:- A constraint is a rule for altering input coordinates values to produce a specified orientation or alignment of the displayed coordinates. the most common constraint is a horizontal or vertical alignment of straight lines.

(3) Grids:- Another kind of constraint is a grid of rectangular lines displayed in some part of the screen area. When a grid is used, any input coordinate position is rounded to the nearest intersection of two grid lines.

(4) Gravity field:- When it is needed to connect lines at positions between endpoints, the graphics packages convert any input position near a line to a position on the line. The conversion is accomplished by creating a gravity area around the line. Any related position within the gravity field of line is moved to the nearest position on the line. It illustrated with a shaded boundary around the line.

(5) Rubber Band Methods:- Straight lines can be constructed and positioned using rubber band methods which stretch out a line from a starting position as the screen cursor.

(6) Dragging:- This methods move object into position by dragging them with the screen cursor.

(7) Painting and Drawing:- Cursor drawing options can be provided using standard curve shapes such as circular arcs and splices, or with freehand sketching procedures. Line widths, line styles and other attribute options are also commonly found in painting and drawing packages.

UNIT-III - SCSA1503- Computer Graphics and Multimedia Applications

3D object representation methods - B-REP, sweep representations, Three dimensional transformations. Curve generation - cubic splines, Beziers, blending of curves- other interpolation techniques, Displaying Curves and Surfaces, Shape description requirement, parametric function. Three dimensional concepts – Introduction - Fractals and self similarity- Successive refinement of curves, Koch curve and peano curves.

3D Concepts and Curves

3D object representation methods - B-REP, sweep representations

Polygon Surfaces

Objects are represented as a collection of surfaces. 3D object representation is divided into two categories.

- **Boundary Representations (B-reps)** − It describes a 3D object as a set of surfaces that separates the object interior from the environment.
- **Space–partitioning representations** − It is used to describe interior properties, by partitioning the spatial region containing an object into a set of small, non-overlapping, contiguous solids (usually cubes).

The most commonly used boundary representation for a 3D graphics object is a set of surface polygons that enclose the object interior. Many graphics system use this method. Set of polygons are stored for object description. This simplifies and speeds up the surface rendering and display of object since all surfaces can be described with linear equations.

The polygon surfaces are common in design and solid-modeling applications, since their **wireframe display** can be done quickly to give general indication of surface structure. Then realistic scenes are produced by interpolating shading patterns across polygon surface to illuminate (Fig 3.1).
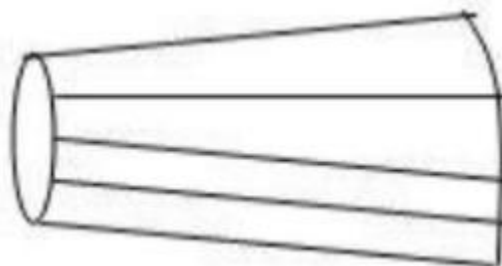

Fig 3.1 A 3D object represented by Polygons

Polygon Tables

In this method, the surface is specified by the set of vertex coordinates and associated attributes. As shown in the following figure, there are five vertices, from v_1 to v_5 (Fig. 3.2).

- Each vertex stores x, y, and z coordinate information which is represented in the table as v_1: x_1, y_1, z_1.
- The Edge table is used to store the edge information of polygon. In the following figure, edge E_1 lies between vertex v_1 and v_2 which is represented in the table as E_1: v_1, v_2.
- Polygon surface table stores the number of surfaces present in the polygon. From the following figure, surface S_1 is covered by edges E_1, E_2 and E_3 which can be represented in the polygon surface table as S_1: E_1, E_2, and E_3.

VERTEX TABLE	EDGE TABLE	POLYGON-SURFACE TABLE
V_1: x_1, y_1, z_1	E_1: V_1, V_2	S_1: E_1, E_2, E_3
V_2: x_2, y_2, z_2	E_2: V_2, V_3	S_2: E_3, E_4, E_5, E_6
V_3: x_3, y_3, z_3	E_3: V_3, V_1	
V_4: x_4, y_4, z_4	E_4: V_3, V_4	
V_5: x_5, y_5, z_5	E_5: V_4, V_5	
	E_6: V_5, V_1	
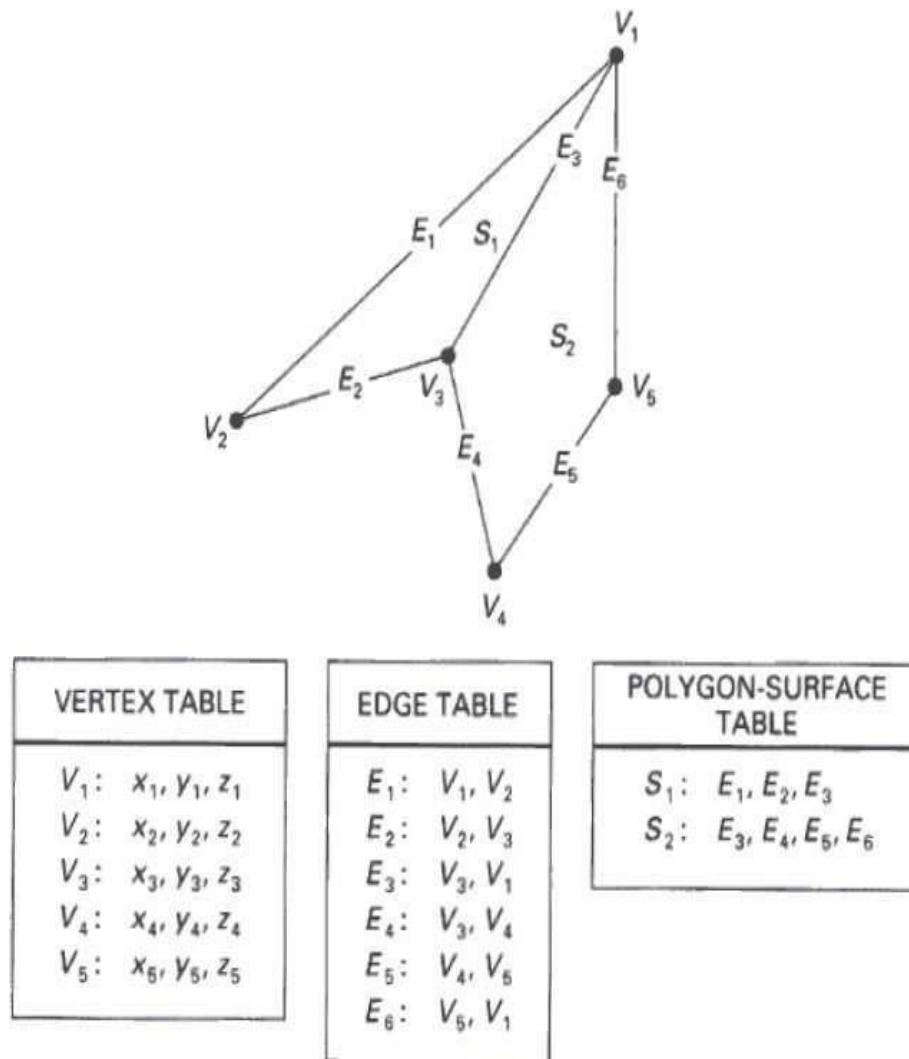
Fig 3.2 Vertex, Edge and Polygon Surface Table

Plane Equations

The equation for plane surface can be expressed as:

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

Where (x, y, z) is any point on the plane, and the coefficients A, B, C, and D are constants describing the spatial properties of the plane. We can obtain the values of A, B, C, and D by solving a set of three plane equations using the coordinate values for three non collinear points in the plane. Let us assume that three vertices of the plane are (x_1, y_1, z_1), (x_2, y_2, z_2) and (x_3, y_3, z_3).

Let us solve the following simultaneous equations for ratios A/D, B/D, and C/D. You get the values of A, B, C, and D.

$$(A/D) \; x_1 + (B/D) \; y_1 + (C/D) \; z_1 = -1$$

$$(A/D) \; x_2 + (B/D) \; y_2 + (C/D) \; z_2 = -1$$

$$(A/D) \; x_3 + (B/D) \; y_3 + (C/D) \; z_3 = -1$$

To obtain the above equations in determinant form, apply Cramer's rule to the above equations.

For any point (x, y, z) with parameters A, B, C, and D, we can say that −
- $Ax + By + Cz + D \neq 0$ means the point is not on the plane.
- $Ax + By + Cz + D < 0$ means the point is inside the surface.
- $Ax + By + Cz + D > 0$ means the point is outside the surface.

Polygon Meshes

3D surfaces and solids can be approximated by a set of polygonal and line elements. Such surfaces are called **polygonal meshes**. In polygon mesh, each edge is shared by at most two polygons. The set of polygons or faces, together form the "skin" of the object (Fig 3.3).

This method can be used to represent a broad class of solids/surfaces in graphics. A polygonal mesh can be rendered using hidden surface removal algorithms. The polygon mesh can be represented by three ways −
- Explicit representation
- Pointers to a vertex list
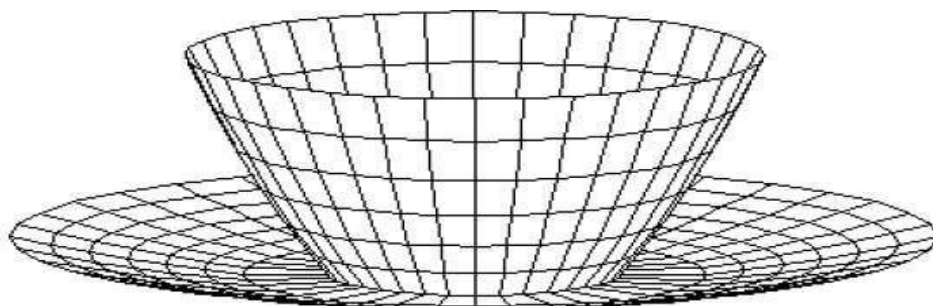- Pointers to an edge list

Fig 3.3 Polygon Mesh

Advantages
- It can be used to model almost any object.
- They are easy to represent as a collection of vertices.
- They are easy to transform.
- They are easy to draw on computer screen.

Disadvantages
- Curved surfaces can only be approximately described.
- It is difficult to simulate some type of objects like hair or liquid.

B-Rep:

1. B-Rep stands for Boundary Representation.
2. It is an extension to the wireframe model.
3. B-Rep describes the solid in terms of its surface boundaries: Vertices, edges and faces as shown below (Fig. 3.4).
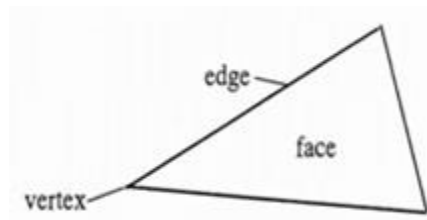
Fig 3.4 B-Rep

1. It is a method for representing shapes using the limits.
2. A solid is represented as a collection of connected surface elements, the boundary between solid and non-solid.
3. There are 2 types of information in a B – rep topological and geometric.
4. Topological information provides the relationships among vertices, edges and faces similar to that used in a wireframe model.
5. In addition to connectivity, topological information also includes orientation of edges and faces.
6. Geometric information is usually equations of the edges and faces.
7. The B-rep of 2 manifolds that have faces with holes (Fig. 3.5) satisfies the generalized Euler's formula:

$$V-E+F-H=2(C-G)$$

Where, V = Number of vertices.
E = Number of edges.
F = Number of faces.
H = Number of holes in the faces.
C is the number of separate components (parts).
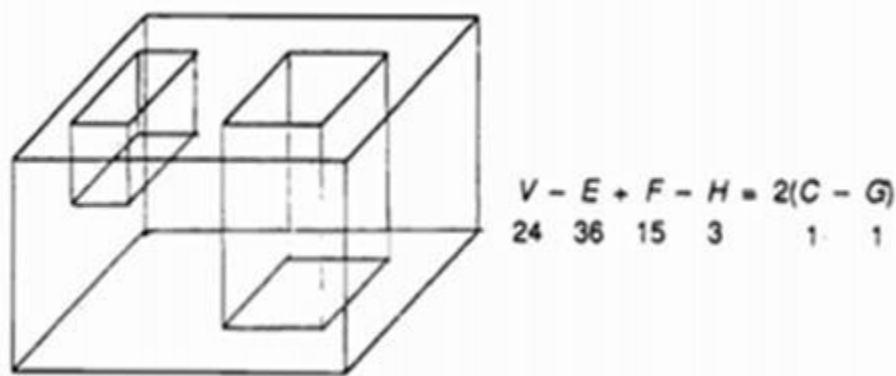G is the genus (for a torus G = 1)

$$V - E + F - H = 2(C - G)$$
$$24 \quad 36 \quad 15 \quad 3 \quad\quad 1 \quad 1$$

Fig 3.5 Example

<u>**Sweep Representations:**</u>

Sweep representations are used to construct 3D object from 2D shape that have some kind of symmetry.

For example, a prism can be generated using a translational sweep and rotational sweeps can be used to create curved surfaces like an ellipsoid or a torus.

In both cases, you start with a cross-section and generate more vertices by applying the appropriate transformation.

More complex objects can be formed by using more complex transformations. Also, we can define a path for a sweep that allows you to create more interesting shapes.

CSG:

1. CSG stands for Constructive Solid Geometry.
2. It is based on set of 3D solid primitives and regularized set theoretic operations.
3. Traditional primitives are: Block, cones, sphere, cylinder and torus.
4. Operations: union, intersection, difference + translation and rotation.
5. A complex solid is represented using with a binary tree usually called as CSG tree.

   CSG tree is shown below (Fig 3.6)
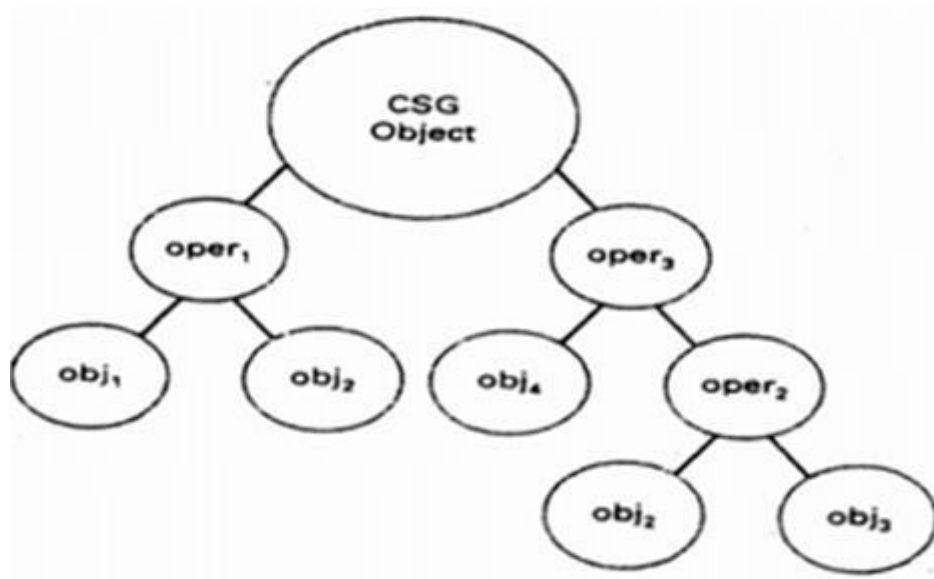
Fig 3.6 CSG Tree

1. Ray casting is a method used for determining boundaries of the resulting object if you start with a boundary representation.
2. Octree representations are designed to make this process easier.

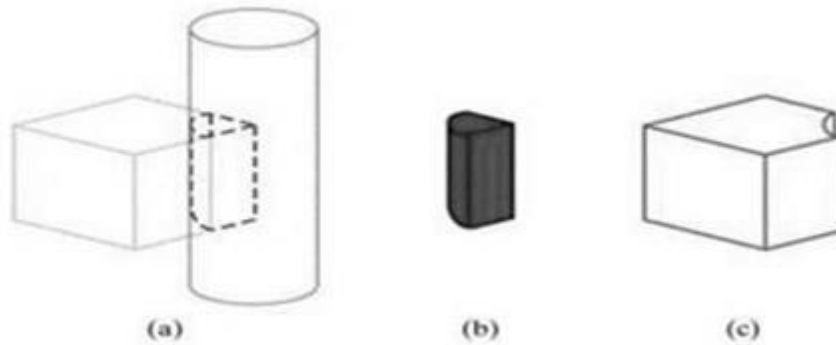   Example: Consider the below figure, say Fig. 3.7.

Fig. 3.7 CSG Construction

Fig. 3.7 (b) is the intersection of original solid.
Fig. 3.7 (c) is the difference between 2 original solid.

Sweep representations are used to construct 3D object from 2D shape that have some kind of symmetry.

For example, a prism can be generated using a translational sweep and rotational sweeps can be used to create curved surfaces like an ellipsoid or a torus.

In both cases, you start with a cross-section and generate more vertices by applying the appropriate transformation.

More complex objects can be formed by using more complex transformations. Also, we can define a path for a sweep that allows you to create more interesting shapes.

Translational sweep:

i. Define a shape as a polygon vertex table as shown in Fig. 3.8 (a).

ii. Define a sweep path as a sequence of translation vectors Fig. 3.8 (b).

iii. Translate the shape; continue building a vertex table Fig. 3.8 (c).
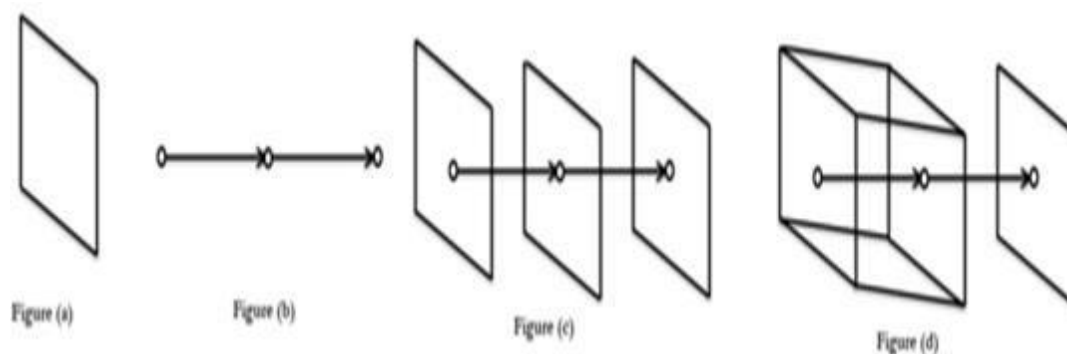
iv. Define a surface table Fig. 3.8 (d).

Fig. 3.8 Translational Sweep

Rotational sweep:

i. Define a shape as a polygon vertex table as shown in Fig. 3.9 (a).

ii. Define a sweep path as a sequence of rotations.

iii. Rotate the shape; continue building a vertex table as shown in Fig. 3.9 (b).

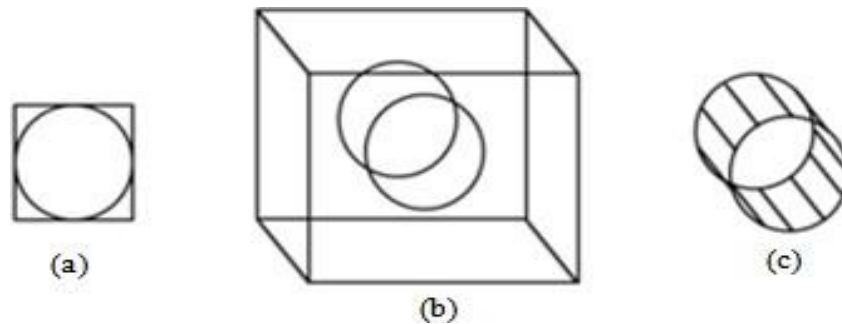iv. Define a surface table as shown in Fig. 3.9 (c).

Fig. 3.9 Rotational Sweep

In computer graphics, we often need to draw different types of objects onto the screen. Objects are not flat all the time and we need to draw curves many times to draw an object.

Three dimensional transformations

Basic Transformations:
1.Translation
2.Rotation
3.Scaling

Other Transformations:
1.Reflection
2.Shearing

Translation

A translation in space is described by tx, ty and tz. It is easy to see that this matrix realizes the equations:

$$x2=x1+tx$$
$$y2=y1+ty$$
$$z2=z1+tz$$

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

Translation Matrix

Rotation

3D rotation is not same as 2D rotation. In 3D rotation, we have to specify the angle of rotation along with the axis of rotation. We can perform 3D rotation about X, Y, and Z axes. They are represented in the matrix form as below (Fig. 3.10):

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

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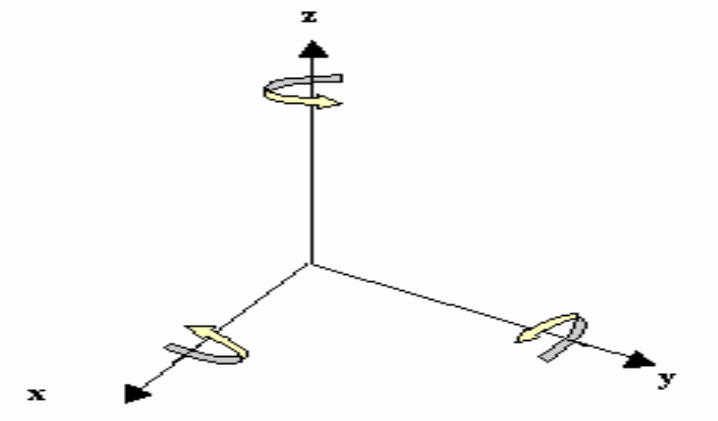

Fig. 3.10 Rotation

Scaling

We can change the size of an object using scaling transformation. In the scaling process, you either expand or compress the dimensions of the object. Scaling can be achieved by multiplying the original coordinates of the object with the scaling factor to get the desired result.

In 3D scaling operation, three coordinates are used. Let us assume that the original coordinates are (X, Y, Z), scaling factors are (S_x,S_y,S_z) respectively, and the produced coordinates are (X', Y', Z'). This can be mathematically represented as shown below :

$$S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

P' = P·S

$$[X'\ \ Y'\ \ Z'\ \ 1] = [X\ \ Y\ \ Z\ \ 1] \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= [X.\,S_x\ \ Y.\,S_y\ \ Z.\,S_z\ \ 1]$$

Reflection

A transformation that gives the mirror image of the object.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ We can mirror the different planes by using scaling factor -1 on the axis that is placed normally on the plane. Notice the matrix to the left. It mirrors around the xy-plane, and changes the coordinates from a right hand system to a left hand system.

Shear

A transformation that slants the shape of an object is called the shear transformation. Like in 2D shear, we can shear an object along the X-axis, Y-axis, or Z-axis in 3D (Fig. 3.11).
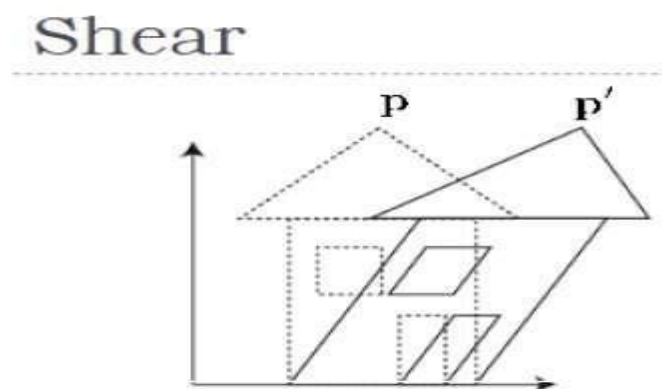

Fig. 3.11 Shearing

As shown in the above figure, there is a coordinate P. You can shear it to get a new coordinate P', which can be represented in 3D matrix form as below

$$Sh = \begin{bmatrix} 1 & sh_x^y & sh_x^z & 0 \\ sh_y^x & 1 & sh_y^z & 0 \\ sh_z^x & sh_z^y & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P' = P \cdot Sh$$

$$X' = X + Sh_x^y Y + Sh_x^z Z$$

$$Y' = Sh_y^x X + Y + sh_y^z Z$$

$$Z' = Sh_z^x X + Sh_z^y Y + Z$$

Curve generation - cubic splines, Beziers, blending of curves- other interpolation techniques

Types of Curves

A curve is an infinitely large set of points. Each point has two neighbors except endpoints. Curves can be broadly classified into three categories − **explicit, implicit,** and **parametric curves**.

Implicit Curves

Implicit curve representations define the set of points on a curve by employing a procedure that can test to see if a point in on the curve. Usually, an implicit curve is defined by an implicit function of the form −

f(x, y) = 0

It can represent multivalued curves (multiple y values for an x value). A common example is the circle, whose implicit representation is

x² + y² - R² = 0

Explicit Curves

A mathematical function y = f(x) can be plotted as a curve. Such a function is the explicit representation of the curve. The explicit representation is not general, since it cannot represent

vertical lines and is also single-valued. For each value of x, only a single value of y is normally computed by the function.

Parametric Curves

Curves having parametric form are called parametric curves. The explicit and implicit curve representations can be used only when the function is known. In practice the parametric curves are used. A two-dimensional parametric curve has the following form −

P(t) = f(t), g(t) or P(t) = x(t), y(t)

The functions f and g become the (x, y) coordinates of any point on the curve, and the points are obtained when the parameter t is varied over a certain interval [a, b], normally [0, 1].

Bezier Curves

Bezier curve is discovered by the French engineer **Pierre Bézier**. These curves can be generated under the control of other points. Approximate tangents by using control points are used to generate curve. The Bezier curve can be represented mathematically as –

$$\sum_{k=0}^{n} P_i B_i^n(t)$$

Where p_i is the set of points and

$$B_i^n(t)$$

represents the Bernstein polynomials which are given by –

$$B_i^n(t) = \binom{n}{i}(1-t)^{n-i}t^i$$

Where **n** is the polynomial degree, **i** is the index, and **t** is the variable.

The simplest Bézier curve is the straight line from the point P_0 to P_1. A quadratic Bezier curve is determined by three control points. A cubic Bezier curve is determined by four control points (Fig. 3.12).
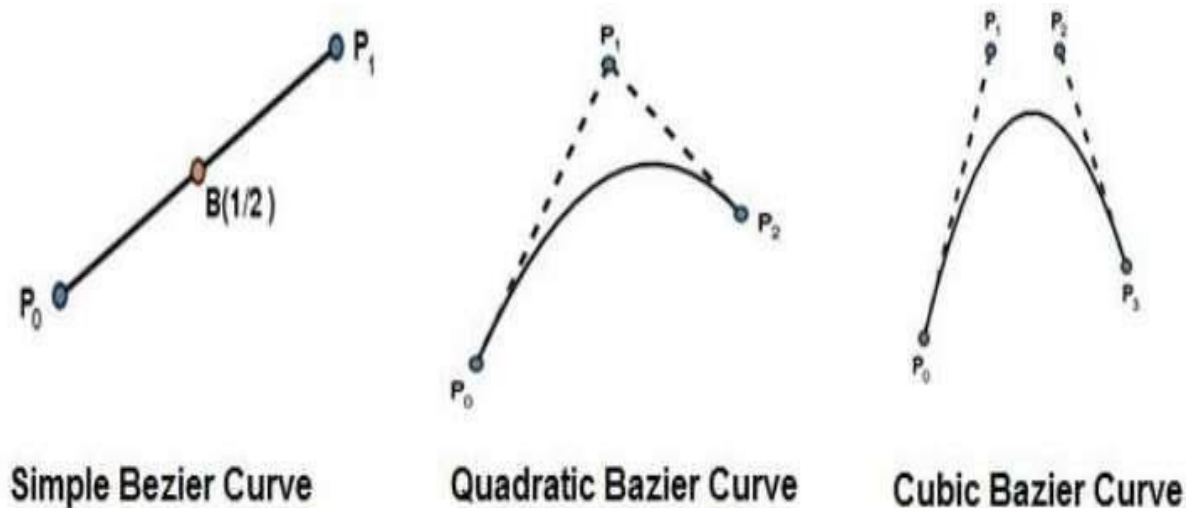
Fig. 3.12 Bezier Curve

Properties of Bezier Curves

Bezier curves have the following properties −

- They generally follow the shape of the control polygon, which consists of the segments joining the control points.
- They always pass through the first and last control points.
- They are contained in the convex hull of their defining control points.
- The degree of the polynomial defining the curve segment is one less that the number of defining polygon point. Therefore, for 4 control points, the degree of the polynomial is 3, i.e. cubic polynomial.
- A Bezier curve generally follows the shape of the defining polygon.
- The direction of the tangent vector at the end points is same as that of the vector determined by first and last segments.
- The convex hull property for a Bezier curve ensures that the polynomial smoothly follows the control points.
- No straight line intersects a Bezier curve more times than it intersects its control polygon.
- They are invariant under an affine transformation.
- Bezier curves exhibit global control means moving a control point alters the shape of the whole curve.
- A given Bezier curve can be subdivided at a point t=t0 into two Bezier segments which join together at the point corresponding to the parameter value t=t0.

B-Spline Curves

The Bezier-curve produced by the Bernstein basis function has limited flexibility.

- First, the number of specified polygon vertices fixes the order of the resulting polynomial which defines the curve.

- The second limiting characteristic is that the value of the blending function is nonzero for all parameter values over the entire curve.

The B-spline basis contains the Bernstein basis as the special case. The B-spline basis is non-global. A B-spline curve is defined as a linear combination of control points Pi and B-spline basis function Ni, k (t) given by

$$C(t) = \sum_{i=0}^{n} P_i N_{i,k}(t), \quad n \geq k-1, \quad t \in [tk-1, tn+1]$$

Where,

- $\{p_i: \text{i=0, 1, 2....n}\}$ are the control points
- k is the order of the polynomial segments of the B-spline curve. Order k means that the curve is made up of piecewise polynomial segments of degree k - 1,
- the $Ni, k(t)$ are the "normalized B-spline blending functions". They are described by the order k and by a non-decreasing sequence of real numbers normally called the "knot sequence".

$$ti: i=0,...n+K$$

The N_i, k functions are described as follows −

$$N_{i,1}(t) = \begin{cases} 1, & if \ u \in [t_i, t_{i+1}) \\ 0, & Otherwise \end{cases}$$

and if k > 1,

$$N_{i,k}(t) = \frac{t - t_i}{t_{i+k-1}} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t)$$

and

$$t \in [t_{k-1}, t_{n+1})$$

Properties of B-spline Curve

B-spline curves have the following properties −

- The sum of the B-spline basis functions for any parameter value is 1.
- Each basis function is positive or zero for all parameter values.
- Each basis function has precisely one maximum value, except for k=1.
- The maximum order of the curve is equal to the number of vertices of defining polygon.
- The degree of B-spline polynomial is independent on the number of vertices of defining polygon.

- B-spline allows the local control over the curve surface because each vertex affects the shape of a curve only over a range of parameter values where its associated basis function is nonzero.
- The curve exhibits the variation diminishing property.
- The curve generally follows the shape of defining polygon.
- Any affine transformation can be applied to the curve by applying it to the vertices of defining polygon.
- The curve line within the convex hull of its defining polygon.

Shape description requirement, parametric function

Shape description requirements

Generally shape representations have two different uses, an analytic use and a synthetic use. Representations are used analytically to describe shapes that can be measured; just as a curve can be fitted to a set of data points, a surface can be fitted to the measured properties of some real objects. The requirements of the user and the computer combine to suggest a number of properties that our representations must have. The following are some of the important properties that are used to design the curves. The similar properties can also be used for designing the surfaces as surfaces are made up of curves.

(a) Control Points: The shape of the curve can be controlled easily with the help of a set of control points, means the points will be marked first and curve will be drawn that intersects each of these points one by one in a particular sequence. The more number of control points makes the curve smoother. The following figure shows the curve with control points (Fig. 3.13).

Fig. 3.13   Control Points (indicated by dots) govern the shape of the Curve

(b) Multiple values: In general any curve is not a graph of single valued function of a coordinate, irrespective the choice of coordinate system. Generally single valued functions of a coordinate make the curves or graphs that are dependent on axis. The following figure shows multivalued curve with respect to all coordinate systems (Fig 3.14).
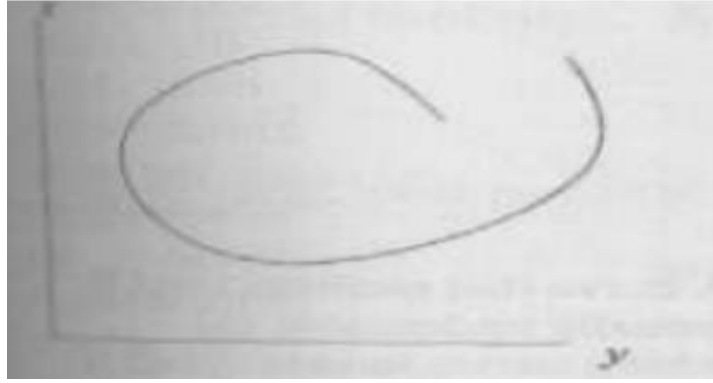
Fig. 3.14 A curve can be multivalued with respect to all coordinate systems

(c)   Axis independence: The shape of an object should not change when the control points are measured in different coordinate systems, that means when an object is rotated to certain angle in any direction (clockwise or anti-clockwise) the shape of the curve should not be affected.

(d)   Global or local control: The control points of a curve must be controlled globally from any function of the same program or it can also be controlled locally by the particular function used to design that curve by calculating the desired control points.

(e)   Variation-diminishing property: Some of the mathematical functions may diminish the curve at particular points and in some other points it may amplify the points. This leads to certain problems for curves appearance at the time of animations, (just as a vehicle looks curved when it is taking turn). This effect must be avoided with the selection of proper mathematical equations with multiple valued functions.

(f)   Versatility: The functions that define the shape of the curve should not be limited to only few varieties of shapes, instead they must provide vide varieties for the designers to make the curves according to their interest.

(g)   Order of continuity: For any complex shapes or curves or surfaces it is essential to maintain continuity in calculating control points. When we are not maintaining the proper continuity of control points it makes a mesh while marking the curve and the complex object.

Parametric functions

The dominant form used to model curves and surfaces is the parametric or vector valued function. A point on a curve is represented as a vector: $P(u)=[x(u)\ y(u)\ z(u)]$.

For surfaces, two parametric are required: $P(u, v)=[(x(u, v)\ y(u, v)\ z(u, v)]$.

As the parametric u and v take on values in a specified range, usually 0 to 1, the parametric functions x, y and z trace out the location of the curve or surface. The parametric functions can

themselves take many forms. A single curve be approximated in sever different ways as given below:

$$P(u) = [\cos u \sin u]$$

$$P(u) = [(1 - u^2)/(1 + u^2)\ 2u/(1 + u^2)]$$

$$P(u) = [u\ (1 - u^2)^{1/2}]$$

By using simple parametric functions, we cannot expect the designer to achieve a desirable curve by changing coefficients of parametric polynomial functions or of any other functional form. Instead we must find ways to determine the parametric function from the location of control points that are manipulated by the designer.

Three dimensional concepts

- Parallel Projection
- Perspective Projection
- Depth Cueing
- Visible Line and Surface Identification
- Surface Rendering
- Exploded and Cutaway Views
- Three-Dimensional and Stereoscopic Views

Parallel Projection

In this method a view plane is used. z co-ordinate is discarded. The 3 d view is constructed by extending lines from each vertex on the object until they intersect the view plane. Then connect the projected vertices by line segments which correspond to connections on the original object (Fig. 3.15).
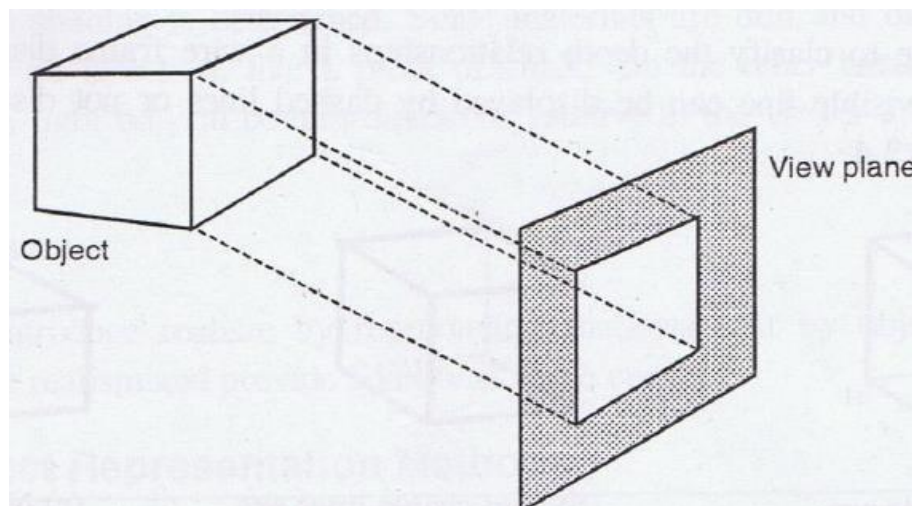
Fig. 3.15 Parallel Projection

Perspective Projection

Here the lines of projection are not parallel. Instead, they all converge at a single point called the `center of projection' or `projection reference point'. The object positions are transformed

to the view plane along these converged projection lines. In this method, Objects farther from the viewing position appear smaller (Fig. 3.16).
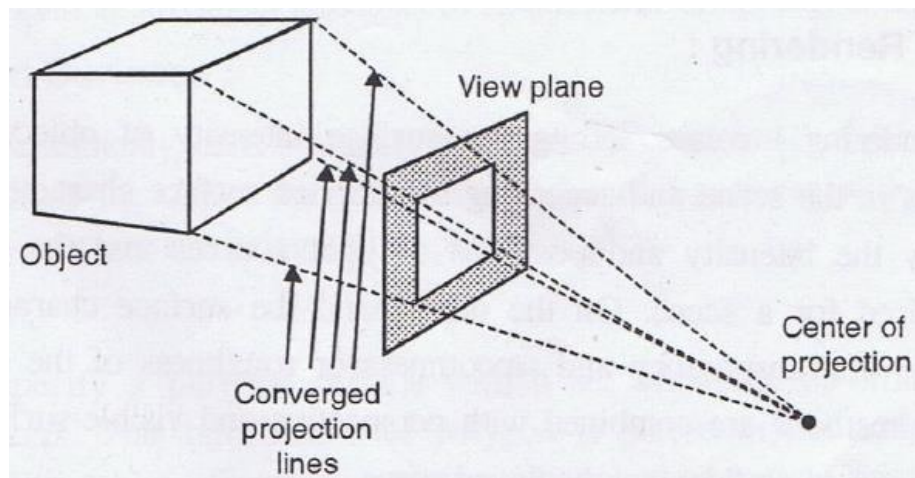

Fig. 3.16 Perspective Projection

Depth Cueing

Depth information is added. The depth of an object can be represented by the intensity of the image. The parts of the objects closest to the viewing position are displayed with the highest intensities. Objects farther away are displayed with decreasing intensities (Fig. 3.17).
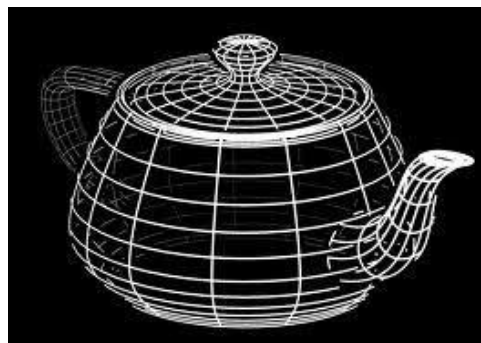

Fig. 3.17 Depth Cueing

Visible Line and Surface Identification

Visible lines are displayed in different color. Invisible lines either displayed in dashed lines or not at all displayed. Removing invisible lines also removes the info about the backside of the object. Surface rendering can be applied for the visible surfaces, so that hidden surfaces will become obscured (Fig. 3.18).
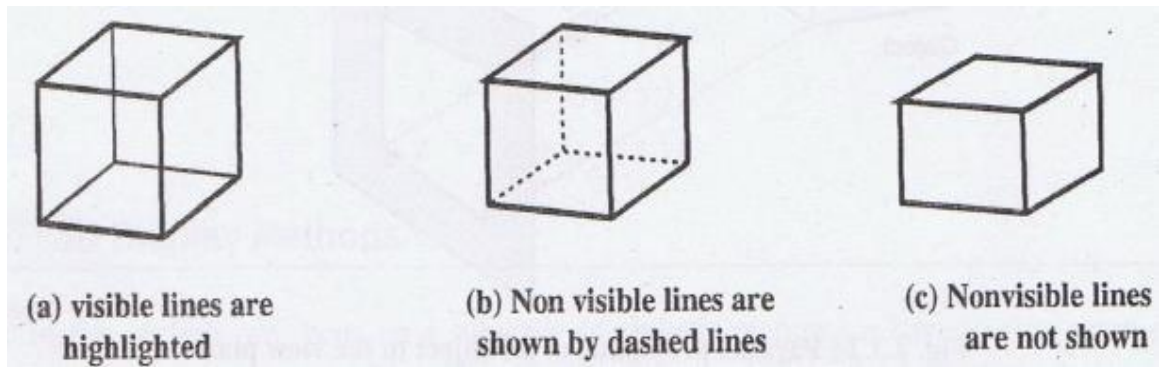
Fig. 3.18 Visible Line and Surface Identification

Surface Rendering

Surface intensity of objects will be according to the lighting conditions in the scene and according to assigned surface characteristics. This method is usually combined with the previous method to attain a degree of realism (Fig. 3.19).
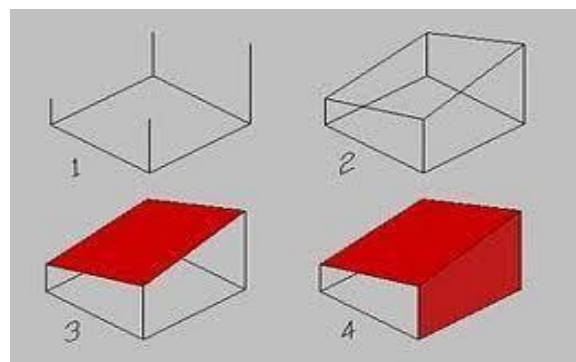

Fig. 3.19 Surface Rendering

Exploded and Cutaway Views

To show the internal details, we can define the object in hierarchical structures (Fig. 3.20).

Exploded view: Cutaway View:
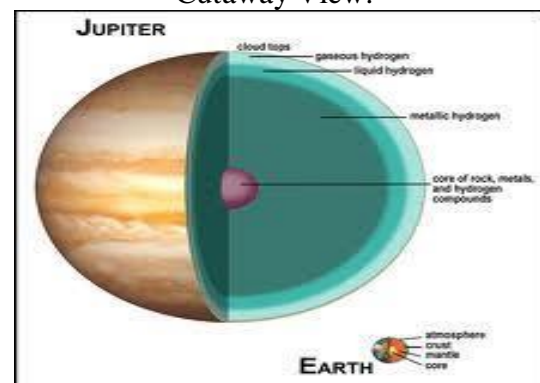
Fig. 3.20 Exploded and Cutaway Views

Fractals and self similarity- Successive refinement of curves, Koch curve and peano curves

Fractals

Fractals are very complex pictures generated by a computer from a single formula. They are created using iterations. This means one formula is repeated with slightly different values over and over again, taking into account the results from the previous iteration (Fig. 3.21).

Fractals are used in many areas such as −

- Astronomy − For analyzing galaxies, rings of Saturn, etc.
- Biology/Chemistry − For depicting bacteria cultures, Chemical reactions, human anatomy, molecules, plants,
- Others − For depicting clouds, coastline and borderlines, data compression, diffusion, economy, fractal art, fractal music, landscapes, special effect, etc.
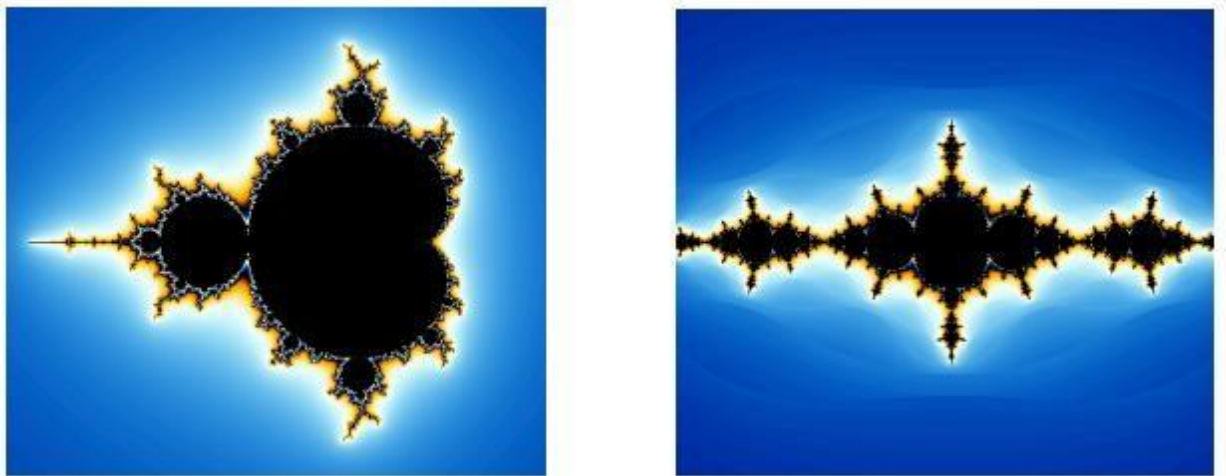

Fig. 3.21 Fractals

Three common techniques for generating fractals are:

- Iterated function systems — These have a fixed geometric replacement rule. Cantor set, Sierpinski carpet, Sierpinski gasket, Peano curve, Koch snowflake, Harter-Heighway dragon curve, T-Square, Menger sponge, are some examples of such fractals.
- Escape-time fractals — Fractals defined by a recurrence relation at each point in a space (such as the complex plane). Examples of this type are the Mandelbrot set, the Burning Ship fractal and the Lyapunov fractal.
- Random fractals, generated by stochastic rather than deterministic processes, for example, fractal landscapes, Lévy flight and the Brownian tree. The latter yields so-called mass- or dendritic fractals, for example, Diffusion Limited Aggregation or Reaction Limited Aggregation clusters.

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.

- 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.
- 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.

Generation of Fractals

Fractals can be generated by repeating the same shape over and over again as shown in the following Fig. 3.22. In fig. 3.22 (a) shows an equilateral triangle. In fig. 3.22 (b), we can see that the triangle is repeated to create a star-like shape. In fig. 3.22 (c), we can see that the star shape in fig. 3.22 (b) is repeated again and again to create a new shape.

We can do unlimited number of iteration to create a desired shape. In programming terms, recursion is used to create such shapes.
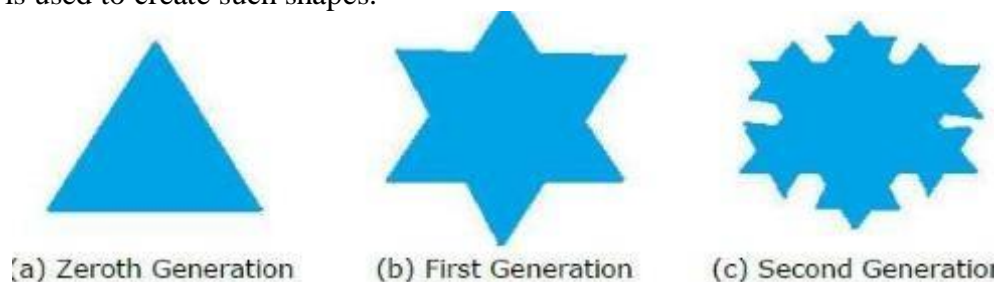

(a) Zeroth Generation   (b) First Generation   (c) Second Generation

Fig. 3.22 Generation of Fractals

Geometric Fractals

Geometric fractals deal with shapes found in nature that have non-integer or fractal dimensions. To geometrically construct a deterministic (nonrandom) self-similar fractal, we start with a given geometric shape, called the initiator. Subparts of the initiator are then replaced with a pattern, called the generator (Fig. 3.23).

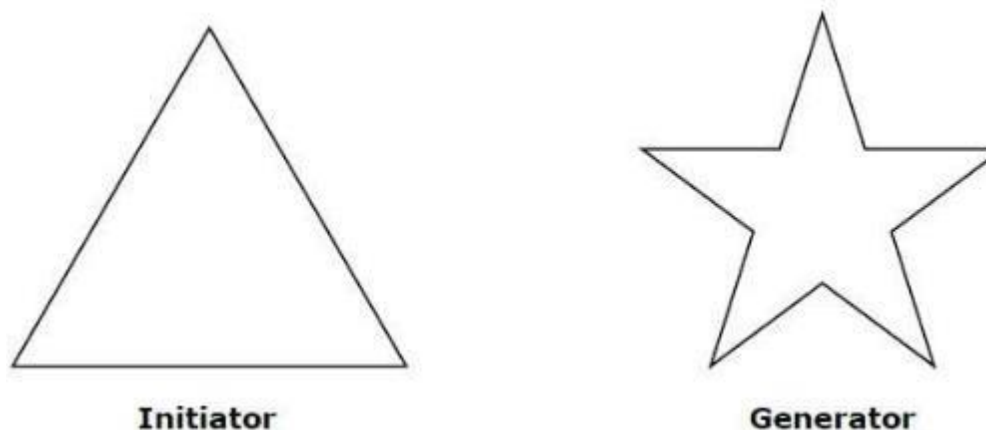**Initiator**   **Generator**

Fig. 3.23 Geometric Fractals

As an example, if we use the initiator and generator shown in the above figure, we can construct good pattern by repeating it. Each straight-line segment in the initiator is replaced with four equal-length line segments at each step. The scaling factor is 1/3, so the fractal dimension is $D = \ln 4/\ln 3 \approx 1.2619$.

Also, the length of each line segment in the initiator increases by a factor of 4/3 at each step, so that the length of the fractal curve tends to infinity as more detail is added to the curve as shown in the following Fig. 3.24.
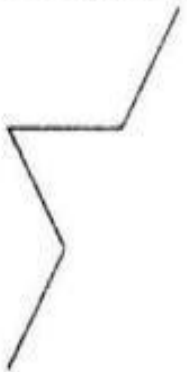

Segment Length = 1	Segment Length = 1/3	Segment Length = 1/9
Length = 1	Length = 4/3	Length = 16/9

Fig. 3.24 Length of the Segment

Many fractals also have a property of self-similarity – within the fractal lies another copy of the same fractal, smaller but complete. In this project we will study the simplest, and best known, fractals with this property, the strictly self-similar fractals. We will show how to describe and create these fractals, and how to measure their fractal dimension using the similarity dimension.

Strictly Self-Similar Fractals

A geometric figure is self-similar if there is a point where every neighborhood of the point contains a copy of the entire figure. For example, imagine the figure formed by inscribing a square within another square, rotated by 45°. Then inside the inner square, inscribe another square in the same manner, and so on ad infinitum. Of course, we can't really draw this figure, since it contains infinitely many nested squares, but an approximation of the result is shown below Fig. 3.25:
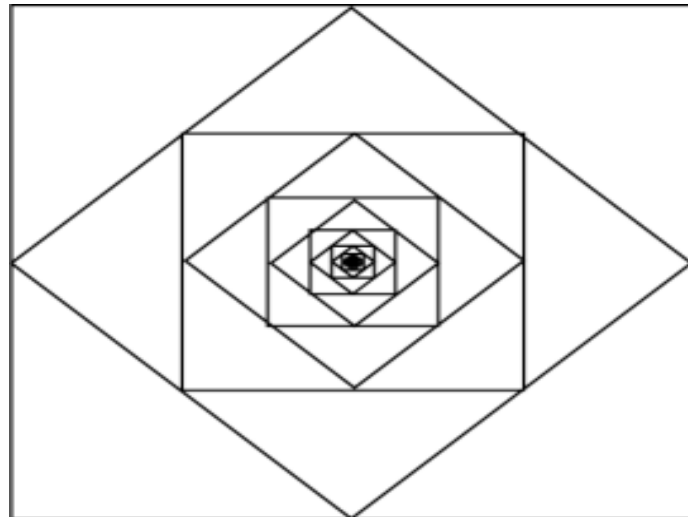
Fig. 3.25   Strictly Self-Similar Fractals

This figure is self-similar at the center of the squares – every ball around the center, no matter how small, will contain a complete copy of the entire figure. However, this is the only point where the figure is self-similar – at most points, the figure looks, at some magnification, like either a straight line or a corner where four lines meet, so this is not complicated enough to be a fractal. A figure is strictly self-similar if it is self-similar at every point. Equivalently, this means that the figure can be decomposed into some number of disjoint pieces, each of which is an exact copy of the entire figure. What does such a figure look like? Let's look at several classical examples. As with the inscribed squares above, these examples are really the limits of some repeated, or iterative, process, which we imagine repeating infinitely often.

Cantor Set

The Cantor set was first described by German mathematician Georg Cantor in 1883, and is the foundation for many important fractals. The description of the set is deceptively easy. Begin with the unit interval [0,1], which is represented below as a line segment (stage 0). Delete the middle third, which is the open interval (1/3, 2/3), leaving the two closed subintervals [0, 1/3] and [2/3, 1]. Now repeat this process on each of these subintervals, leaving 4 subintervals, and continue repeating this process on each new set of smaller subintervals forever. The Fig. 3.26 below shows the first few stages of this construction:

Fig. 3.26 Cantor Set

The Cantor set is the set of points remaining when all of these deletions of subintervals have taken place. This is more than you might expect – for example, all the endpoints of all the closed intervals at every stage of the process are in the Cantor set. In fact, there are just as many points in the Cantor set as there were in the original unit interval. Despite this, any point of the interval [0, 1] that is not in the Cantor set is the center of an open interval which contains no points of the Cantor set – we say that the Cantor set is nowhere dense. So the Cantor set is simultaneously as large as the entire unit interval, and so small that any point not in the set can be separated from it with an open interval. It is truly an amazing mathematical object! However, we are most interested in the fact that the Cantor set is strictly self-similar: the points in the Cantor set are all either in [0, 1/3] or [2/3, 1], and the subset of the Cantor set in either of these intervals is an exact copy of the entire set. So the Cantor set can be decomposed into two disjoint pieces, each of which is itself a Cantor set. In fact, at any stage n, the Cantor set can be decomposed into 2n disjoint pieces, each an exact copy of the entire set.

Koch Curve

Swedish mathematician Helge von Koch introduced the Koch curve in 1904, as an example of a curve that is continuous but nowhere differentiable. It is the result of a simple geometric construction that is easily generalized to give a wide variety of examples. Begin with a line segment in the plane – for example, the interval [0, 1] along the x-axis (stage 0). On the middle third of this interval, construct an equilateral triangle upward (so each side of the triangle has length 1/3, and one side is the interval[1/3, 2/3]), and remove the base (the open interval (1/3, 2/3)). The result is four line segments of length 1/3, arranged as shown below (stage 1). For stage 2, perform the same construction on each of these four line segments so that the new segments protrude outward from those of stage 1 (see below) and continue to repeat the process indefinitely on all the smaller line segments formed at each stage. A few stages of this process are shown below (Fig. 3.27):
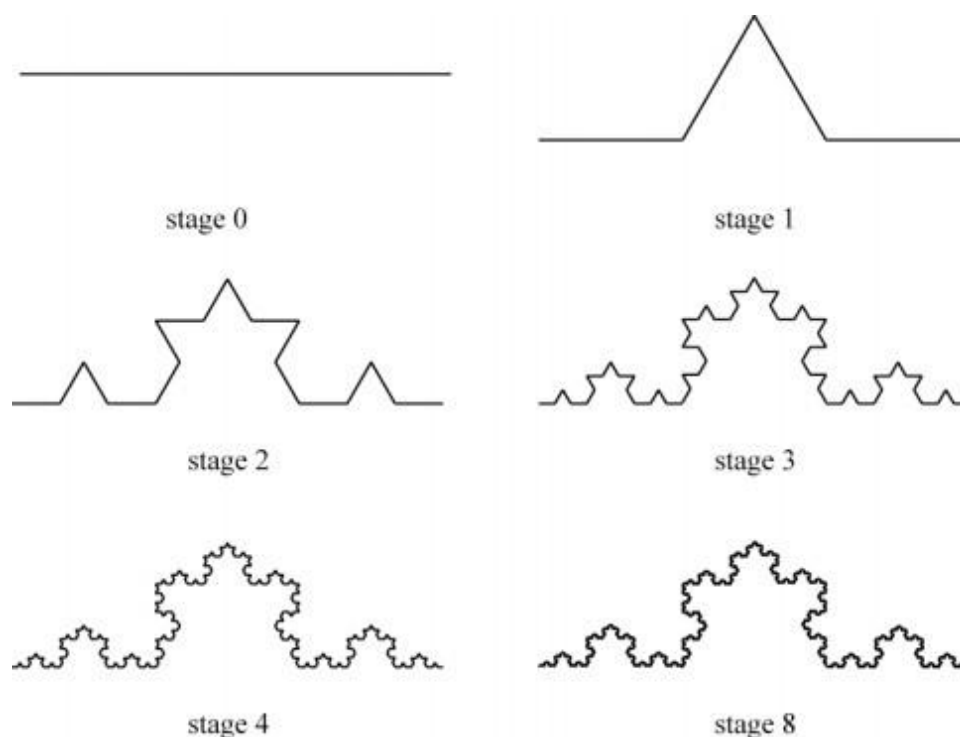

stage 0   stage 1

stage 2   stage 3

stage 4   stage 8

Fig. 3.27 Koch Curve

The Peano Curve and Fractal Curves

There are examples of curves (in the sense of continuous maps from the real line to the plane) that completely cover a two-dimensional region of the plane. We give a construction of such a Peano curve, adapted from David Hilbert's example. The construction is inductive, and is based on replacement rules. We consider building blocks of six shapes,
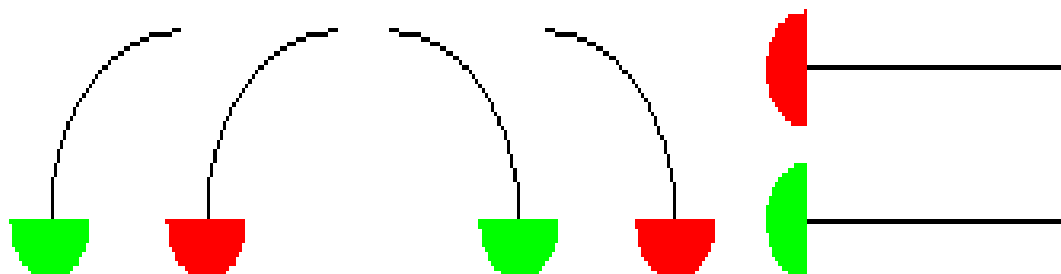
the length of the straight pieces being twice the radius of the curved ones. A sequence of these patterns end-to-end represents a curve, if we disregard the red and green half-disks. The replacement rules are the following (Fig. 3.28):
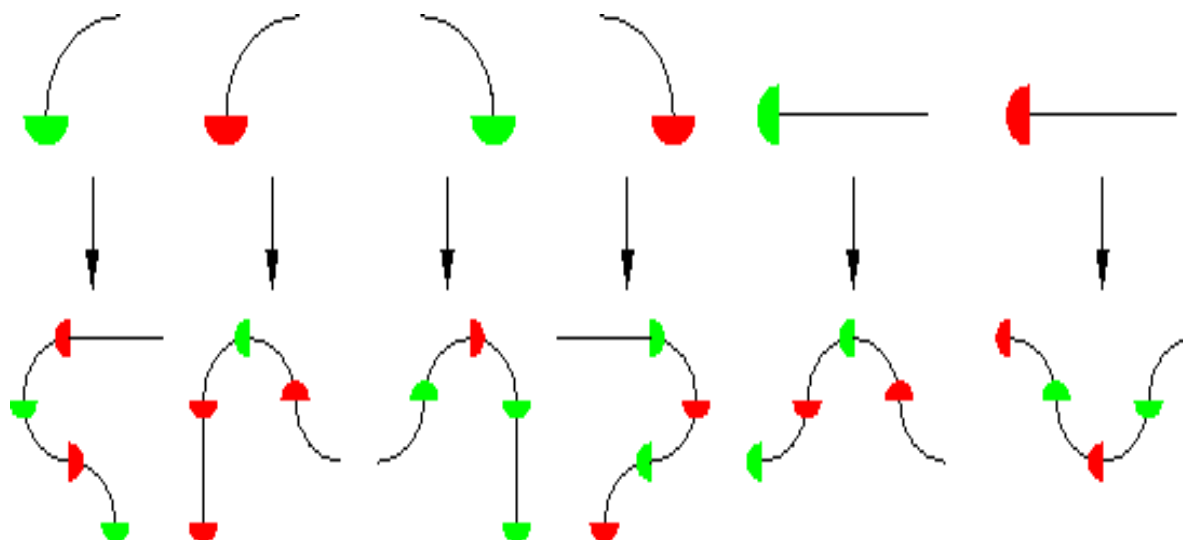

Fig. 3.28 Example 1: Peano Curve

The rules are applied taking into account the way each piece is turned. Here we apply the replacement rules to a particular initial pattern (Fig. 3.29):

Fig. 3.29 Example 2: Peano Curve

(We scale the result so it has the same size as the original.) Applying the process repeatedly gives, in the limit, the Peano curve. Here are the first five steps (Fig. 3.30):
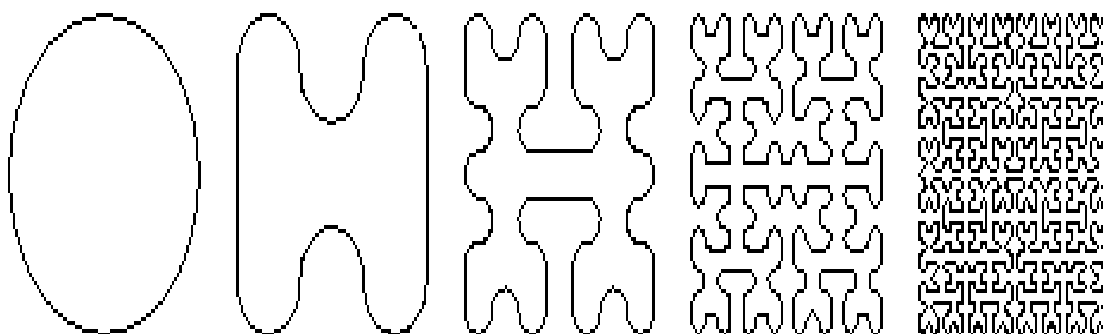

Fig. 3.30 Example 3: Peano Curve

The same idea of replacement rules leads to many interesting fractal, and often self-similar, curves. For example, the substitution ——— ➤ ‿︿‿ leads to the Koch snowflake (Fig. 3.31) when applied to an initial equilateral triangle, like this (the first three stages and the sixth are shown):
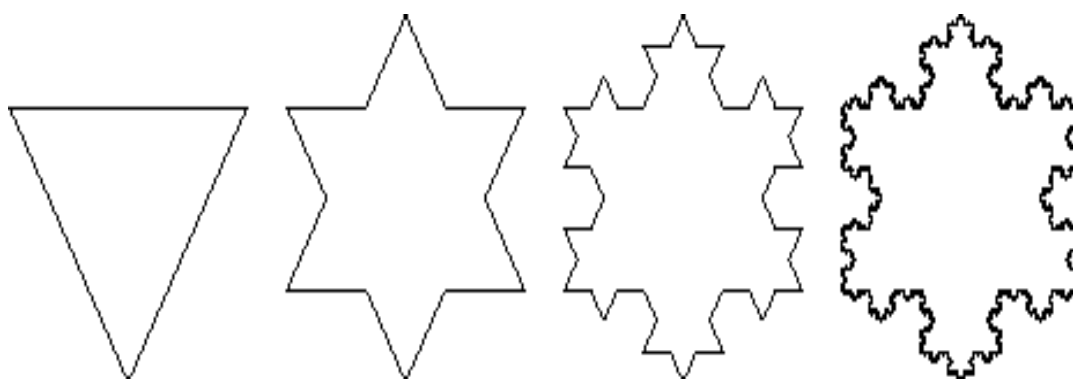

Fig. 3.31 Example 4: Peano to Koch Curve

UNIT-IV - SCSA1503- Computer Graphics and Multimedia Applications

Visible surface detection methods – Illumination models – Halftone patterns – Dithering techniques – Polygon rendering methods – Ray tracing methods – Color models and color application

Methods and Models

Visible Surface Detection Methods

Introduction

- To generate realistic graphics displays.
- To determine what is visible within a scene from a chosen viewing position.
- For 3D worlds, this is known as visible surface detection or hidden surface elimination.
- So many methods and algorithms are available.
- Some require more memory space, some require more processing time.
- Which method? For which application? Depends on which object to be displayed, available equipment, complexity of the scene, etc..

Basic classification

- Two basic classifications –based on either an object or projected image , which is going to be displayed

i) **Object-space Methods-** compares objects and parts of objects to each other within a scene definition to determine which surfaces are visible

ii) **Image-space Methods-** visibility is determined point-by-point at each pixel position on the projection plane

   Image-space Method is the most commonly used method

Back-face detection

- Object Space Method.
- To find the back faces of a polyhedron.
- Consider a polygon surface with parameters A,B,C and D.
- A point (x,y,z) is inside the polyhedrons' backface only if

$$Ax+By+Cz+D<0$$

- We can simply say that the z component of the polygon's normal is less than zero, then the point is on the back face.(Fig4.1)
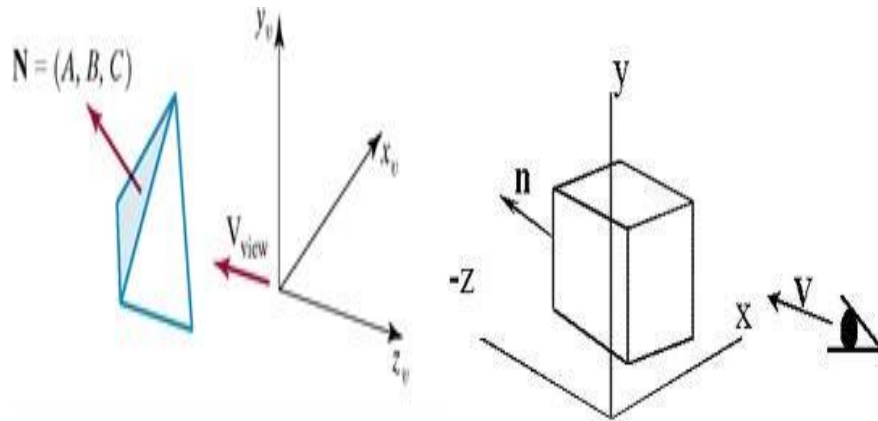
Fig 4.1 Back Face Detection

    ☐   If we take V as the vector in the viewing direction from the eye and N as the normal vector to the polygon's surface, then we can state the condition as

        I. $N > 0$

  then that face is the back face.

    ☐   It eliminates about half of the polygon surfaces in a scene from further visibility tests.

Depth buffer method or z-buffer method

    ☐   Image Space Method.

    ☐   Compares surface depths at each pixel position throughout the scene on the projection plane.

    ☐   It is usually applied to images containing polygons.
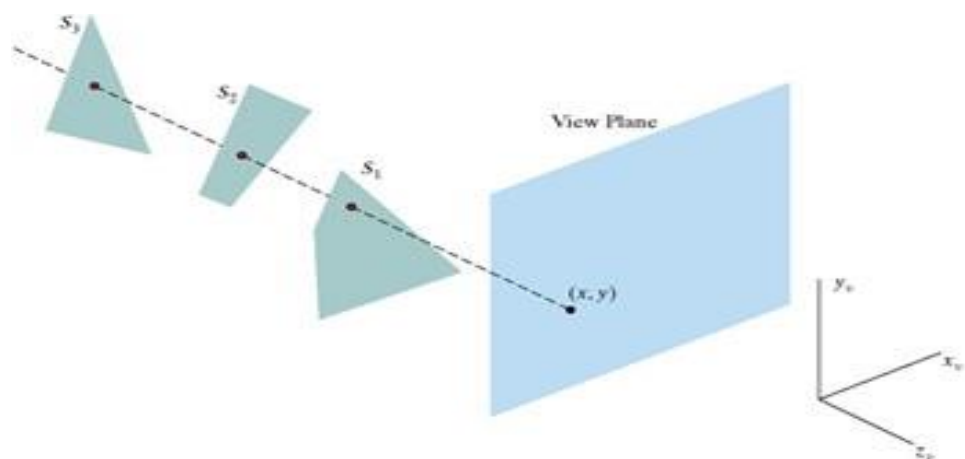
    ☐   Very fast method.

Fig 4.2 Example of Depth buffer method

Depth buffer algorithm

- For each projected (x, y) pixel position of a polygon, calculate the depth z (if not already known)(Fig 4.2)

- If z < depth(x, y), compute the surface colour at that position and set

    depth(x, y) = z

    frameBuff(x, y) = surfColour(x, y)

After all surfaces are processed depthBuff and frameBuff will store correct values

$$z = \frac{-Ax - By - D}{C}$$

$$z' = \frac{-A(x+1) - By - D}{C}$$

- The depth-buffer algorithm proceeds by starting at the top vertex of the polygon
- Then we recursively calculate the *x*-coordinate values down a left edge of the polygon
- The *x* value for the beginning position on each scan line can be calculated from the previous one.

A-buffer Method

- Extension of Depth-buffer -> **accumulation buffer(A-Buffer)**
- **A-** Antialiased, Area-averaged, Accumulation – Buffer.
- Drawback of Depth-buffer-can find one visible surface at each pixel position. Cannot accumulate intensity values for more than one surface.
- Each buffer position can reference a linked-list of surfaces.
- Each position has 2 fields.
- Depth field – stores a positive or negative real number
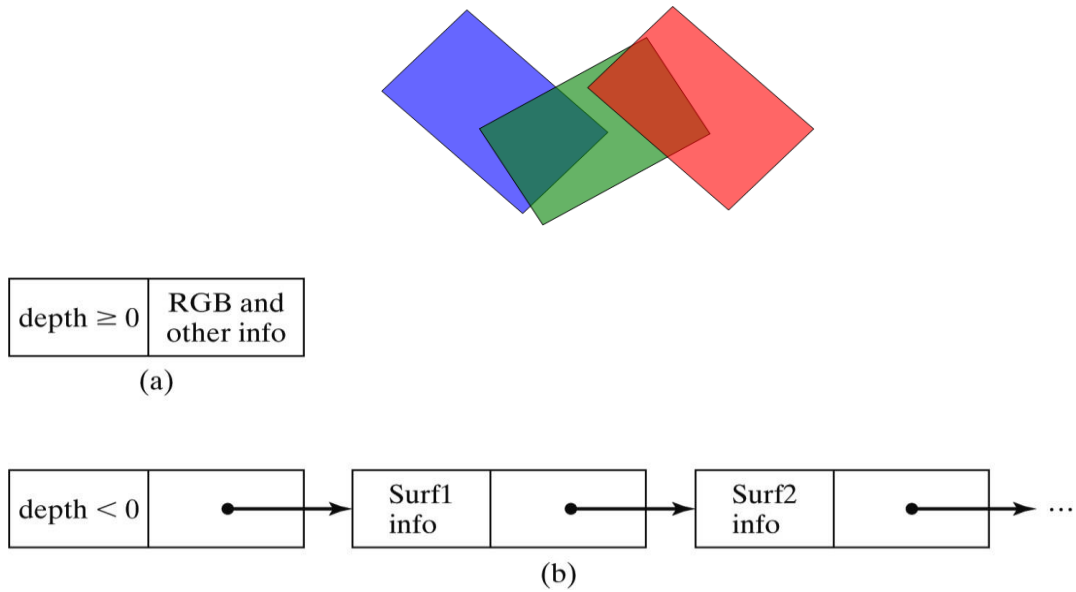- Intensity field- stores surface intensity information or a pointer value.

(a)

(b)

Fig 4.3 A-Buffer (a) Depth>=0 (b) Depth<0

- If depth is >= 0(single surface), then the surface data field stores the depth of that pixel position as before

- If depth < 0 (multiple surfaces)then the data filed stores a pointer to a linked list of surface data. (Fig4.3)

Scan-line method

- Idea is to intersect each polygon with a particular scanline. Solve hidden surface problem for just that scanline.

- Requires a depth buffer equal to only one scanline

- The cost of tiling scene is roughly proportional to its depth complexity
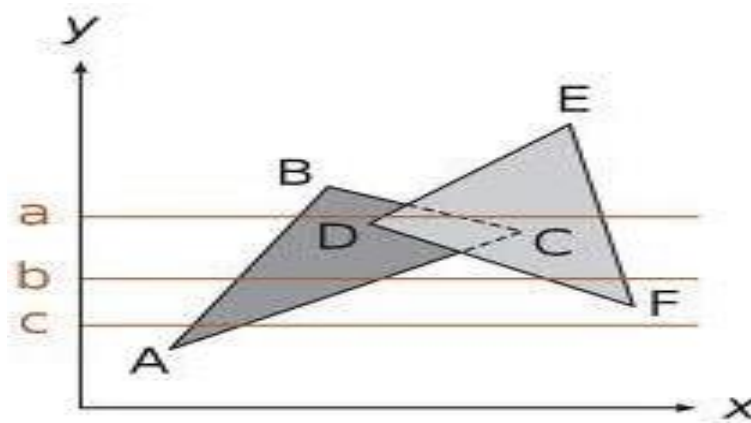
- Efficient way to tile shallowly-occluded scenes(Fig 4.4)

Fig 4.4 Scan Line method

5

Depth- sorting or painter's algorithm

- Sort the objects by distance from the viewer.

  - Draw objects in order from farthest to nearest.

  - Nearer objects will "overwrite" farther ones.

  - If 2 objects DO overlap

  - Need to find a plane to split one polygon by so that each new polygon is entirely in front of or entirely behind the other

  - Polygons may actually intersect, so then need to split each polygon by the other (Fig 4.5)
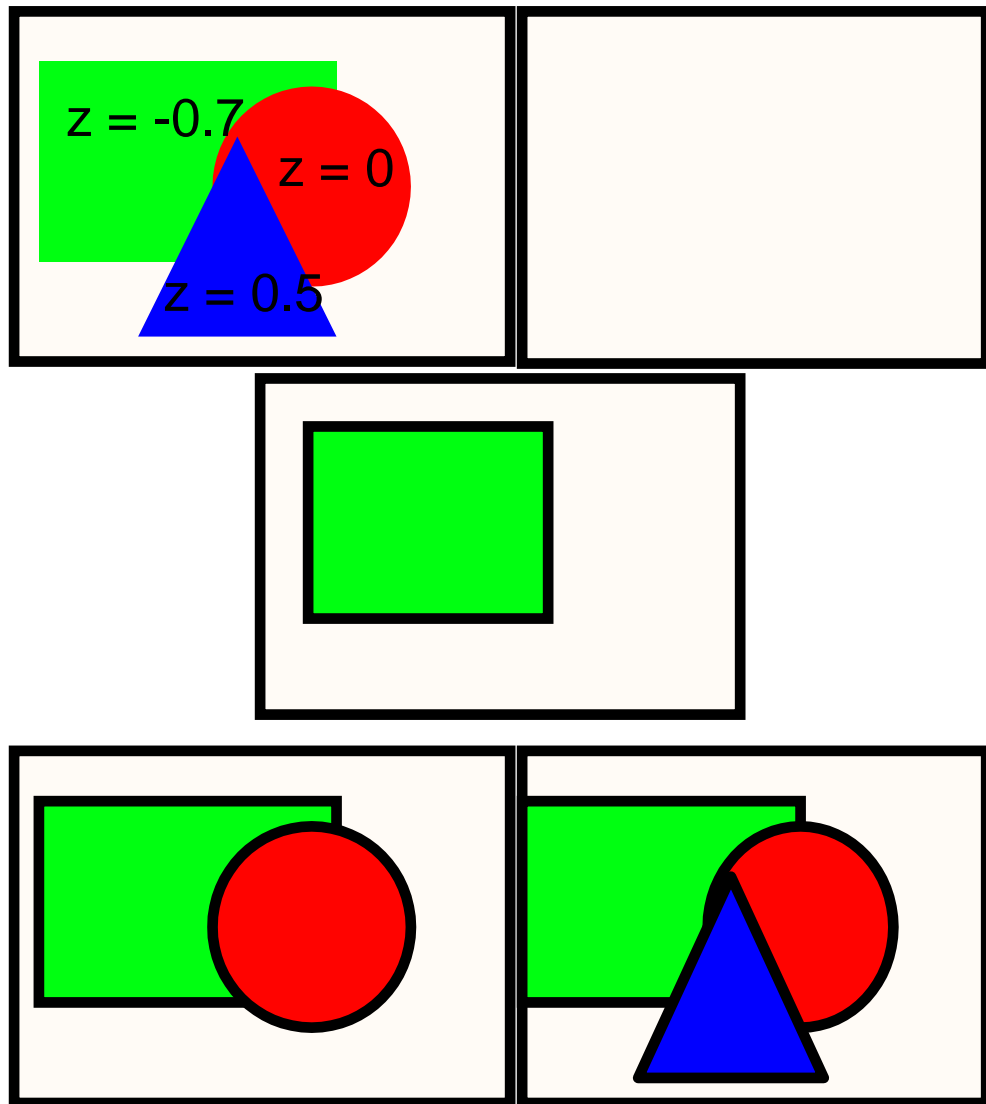

Fig 4.5 Depth Sorting method

BSP Tree Method

- A BSP (Binary Space-Partitioning) tree is formed by first choosing a triangle from the set of all triangles in the scene.

- The plane that contains this triangle is called P. Classify all other triangles into two groups: One group in front of P, and the other group behind P. All triangles that are intersected by P are split into multiple smaller triangles, each of which is either in front of P or behind P.

- Within each group, recursively pick another triangle and partition all the triangles in this group into two sub-groups.

- Do this until there is only one triangle in each group.
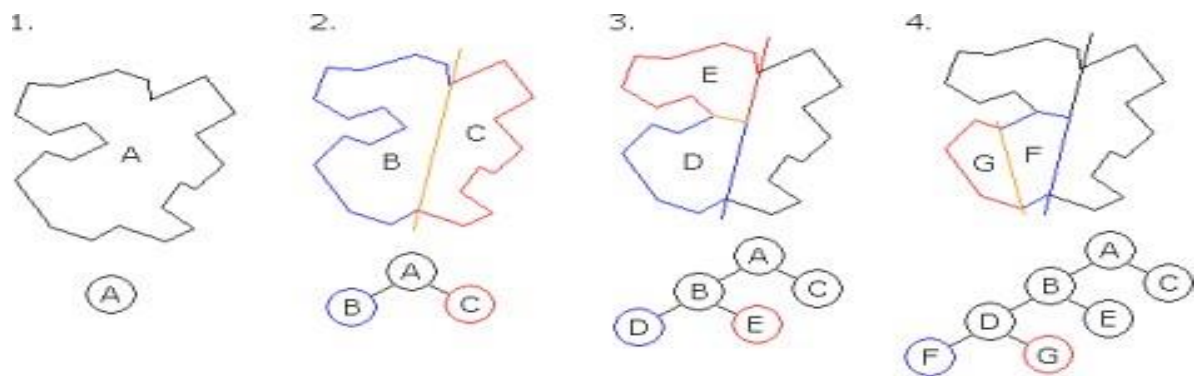
- The result is a tree.(Fig 4.6)

Fig 4.6 BSP Tree method

Area Subdivision Method

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.(Fig 4.7)

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

| Surrounding Surface | Overlapping Surface | Inside Surface | Outside Surface |

Possible relationships between polygon surfaces and a rectangular area.
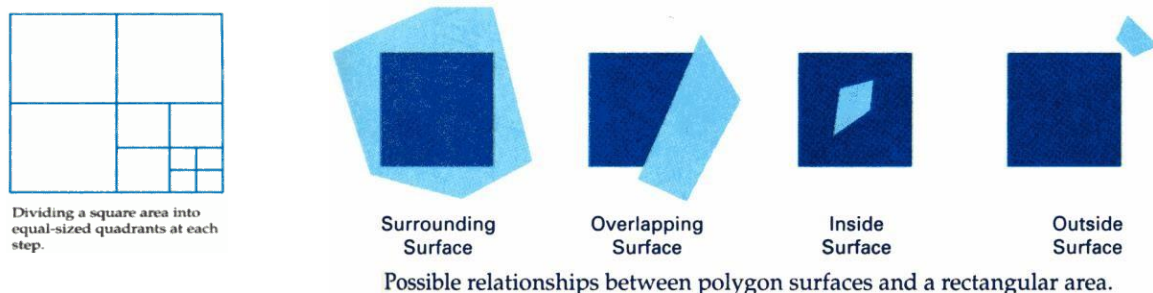
Fig 4.7 Area Subdivision method

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.

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 sub- octants of octant 0 are visited before the nodes for the four back sub-octants.(Fig 4.8)
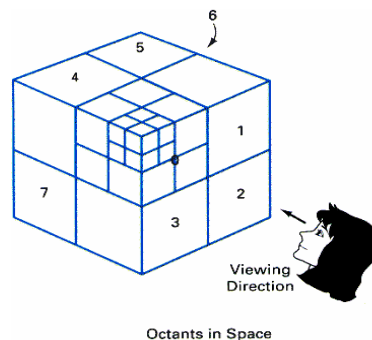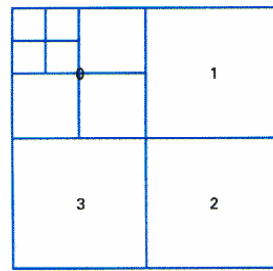

Fig 4.8 Octree method

When a colour is encountered in an octree node, the corresponding pixel in the frame buffer is painted only if no previous color has been loaded into the same pixel position.

Quadrants for
the View Plane

In most cases, both a front and a back octant must be considered in determining the correct color values for a quadrant. But

- If the front octant is homogeneously filled with some color, we do not process the back octant.
- If the front is empty, it is necessary only to process the rear octant.
- If the front octant has heterogeneous regions, it has to be subdivided and the sub-octants are handled recursively.

Ray Casting Method

The intensity of a pixel in an image is due to a ray of light, having been reflected from some objects in the scene, pierced through the centre of the pixel.

Center
of
projection
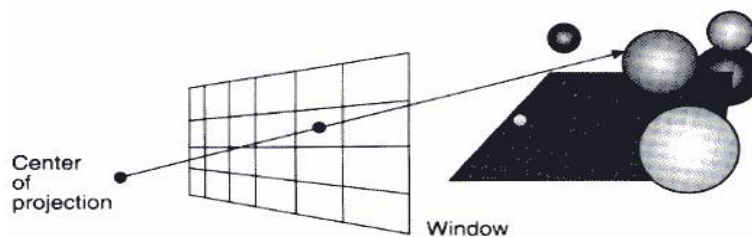
Window

A ray is fired from the center of projection through each pixel to which the window maps, to determine the closest object intersected.

Fig 4.9 Ray Casting Method

So, visibility of surfaces can be determined by tracing a ray of light from the centre of projection(Fig 4.9)
(viewer's eye) to objects in the scene. (backward-tracing).

- Find out which objects the ray of light intersects.
- Then, determine which one of these objects is closest to the viewer.
- Then, set the pixel color to this object.

The ray-casting approach is an effective visibility-detection method for scenes with curved surfaces, particularly spheres.

Speeding up the intersection calculation in ray tracing

For 1024x1024 pixels image and 100 objects in the scene, total number of object intersection

calculations is about 100 millions.

1. Bounding Volume Approach
- Test for intersection of ray with the bounding volume of the object.
- Typical bounding volumes are sphere, ellipsoid, rectangular solid. The intersection calculation for these bounding volumes are usually faster than the displayed object.
- If ray does not intersect bounding volume, no further processing of the object is needed.
- Other bounding volumes include convex polygons formed by a set of planes.

2. Using Hierarchies
- If a parent bounding volume does not intersect with a ray, all its children bounding volumes do not intersect with the ray and need not be processed
- Thus reduce the number of intersection calculations.
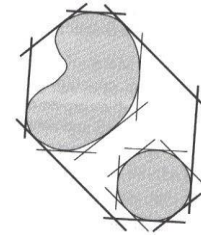
3. Space Partitioning Approach
- Partition the space into a regular grid of equal-size volumes.
- Each volume has associated with it a list of objects which are contained within or intersect the volume.
- Intersection calculation is only applied to those objects that are contained within the partition through which the ray passes.
- Objects lying within the partitions which do not intersect with the ray are not processed.
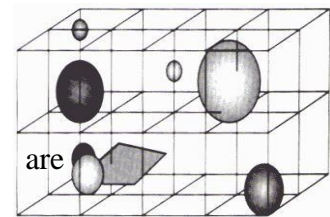
Summary and Comparison

The most appropriate algorithm to use depends on the scene
- depth-sort is particularly suited to scene with objects which are spread out along z-axis and/or with a small number of objects => rarely overlap in depth
- scan-line and area subdivision algorithms are suitable to scene where objects are spread out horizontally and/or scene with small number of objects (about several thousand surfaces).
- Z-buffer and subdivision algorithms perform best for scene with fewer than a few thousand
  surfaces.
- Octree is particularly good because it does not require any pre-sorting or intersection calculations.
- If parallel processing hardware is available, ray tracing would be a good choice (each processor handles a ray).

Illumination Models

• Motivation: In order to produce realistic images, we must simulate the appearance of surfaces under various lighting conditions

• Illumination Model: Given the illumination incident at a point on a surface, quantifies the reflected light(Fig 4.10)
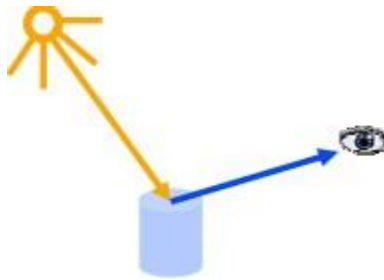

Fig 4.10 Illumination Model

Illumination Model Parameters

• Lighting effects are described with models that consider the interaction of light sources with object surfaces

• The factors determining the lighting effects are:

 – The light source parameters:

• Positions

• Electromagnetic Spectrum

 • Shape

 – The surface parameters

 • Position

• Reflectance properties

 • Position of nearby surfaces

 – The eye (camera) parameters

 • Position

• Sensor spectrum sensitivities

Illumination Models and Rendering

• An illumination model is used to calculate the intensity of the light that is reflected at a given point on a surface

• A rendering method uses intensity calculations from the illumination model to determine the light intensity at all pixels in the image(Fig 4.11)
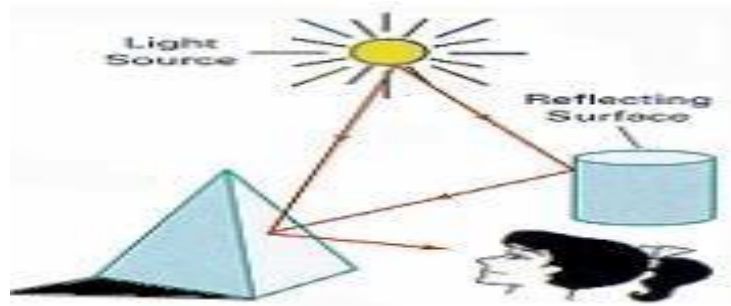
Fig 4.11 Illumination Model and Rendering

Light Source Models

 • Point Source (a): All light rays originate at a point and radially diverge. A reasonable approximation for sources whose dimensions are small compared to the object size

• Parallel source (b): Light rays are all parallel. May be modeled as a point source at infinite distance (the sun)

• Distributed source (c): All light rays originate at a finite area in space. It models a nearby source, such as a fluorescent light(Fig4.12)
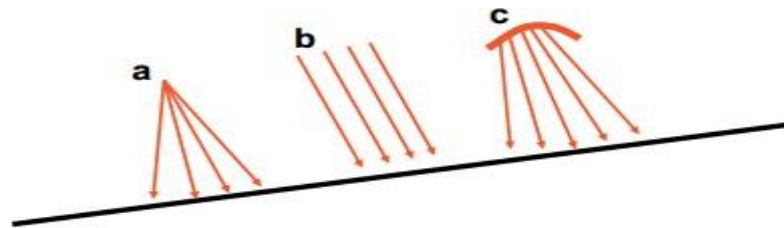
Fig 4.12 Light Source Model a)Point b)Parallel c)Distributed Source

• Simplified and fast methods for calculating surfaces intensities, mostly empirical

 • Calculations are based on optical properties of surfaces and the lighting conditions (no reflected sources nor shadows)

• Light sources are considered to be point sources

• Reasonably good approximation for most scenes

Simple Illumination Model

Surfaces in real world environments receive light in 3 ways:
1. Directly from existing light sources such as the sun or a lit candle
2. Light that passes and refracts through transparent objects such as water or a glass vase
3. Light reflected, bounced, or diffused from other exisiting surfaces in the environment

12

Diffuse illumination

Fig 4.13 Diffuse Illumination

Lambert's cosine law of reflection as shown in the above diagram:

1. **n**; a normal vector to the surface to be illuminated.

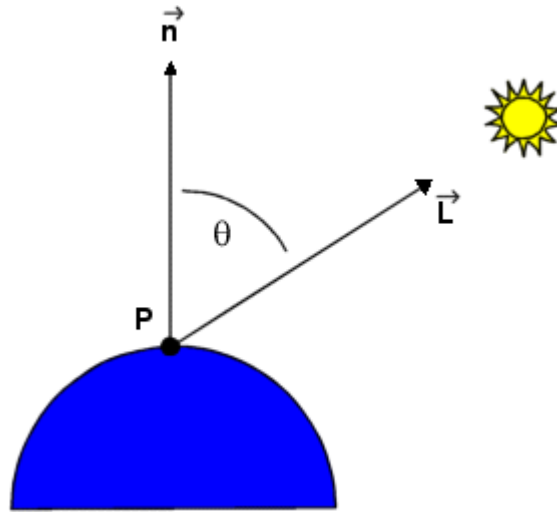2. **L,** a vector from the surface position that points towards the light source.

3. I_l, an intensity for a point light source.

4. k_d, a diffuse reflection constant.

(Fig 4.13)Equation gives the brightness of a surface point in terms of the brightness of a light source and its orientation relative to the surface normal vector, **n**,

$$I=I_l k_d \hat{\mathbf{n}} \bullet \hat{\mathbf{L}}=I_l k_d \cos\theta \qquad 0 \leq \theta \leq \pi/2$$

- *I* is the reflected intensity
    - Measures how bright the surface is at that point.
- Surface brightness varies as a function of the angle between **n** and **L**
    - When **n** and **L** coincide, the light source is directly overhead.
- *I* is at a maximum and $\cos_q = 1$.
    - As the angle increases to 90º, the cosine decreases the intensity to 0.
- All the quantities in the equation are normalized between 0 and 1.
- *I* is converted into frame buffer intensity values by multiplying by the number of shades available.
- With $2^8 = 256$ possible shades, we have 1 * 255, the brightest frame buffer intensity.
- For **n** and **L** at an angle of 45 º, I = cos 45 º * 256 = 181.
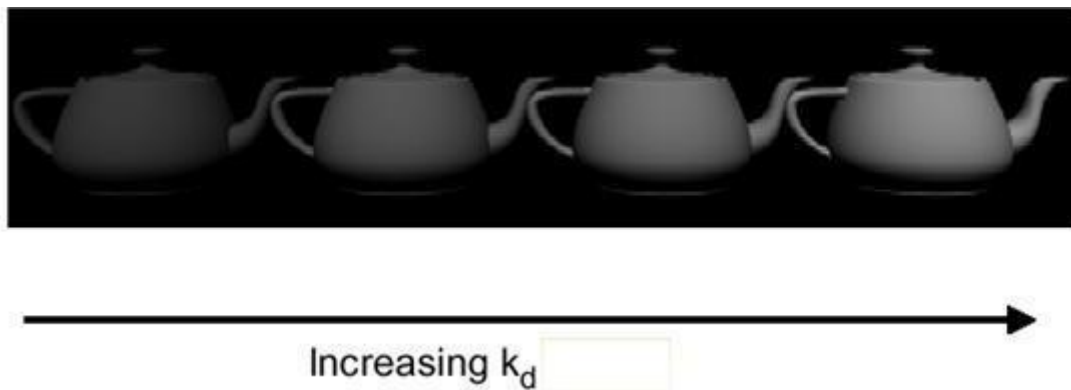
Fig 4.14 Lambertian Shader

- An image rendered with a Lambertian shader exhibits a dull, matte finish.(Fig 4.14)
- It appears as if it has been viewed by a coal miner with a lantern attached to his helmet.
- In reality, an object is not only subjected to direct illumination from the primary light source I_l, but secondary scattered light from all remaining surfaces.

Ambient illumination

- Simple illuminated model is unable to directly accommodate all scattered light
- It is grouped together as independent intensity, I_a.
- The formula becomes

$$I = I_a k_a + I_l k_d \cos\theta \qquad 0 \le \theta \le \pi/2$$

☐ $I_a k_a$ is the ambient illumination term, taking into account the additional environmental illumination, I_a, and the ability of the object to absorb it, k_a.(Fig4.15)

Fig. 4.15 Only ambient illumination
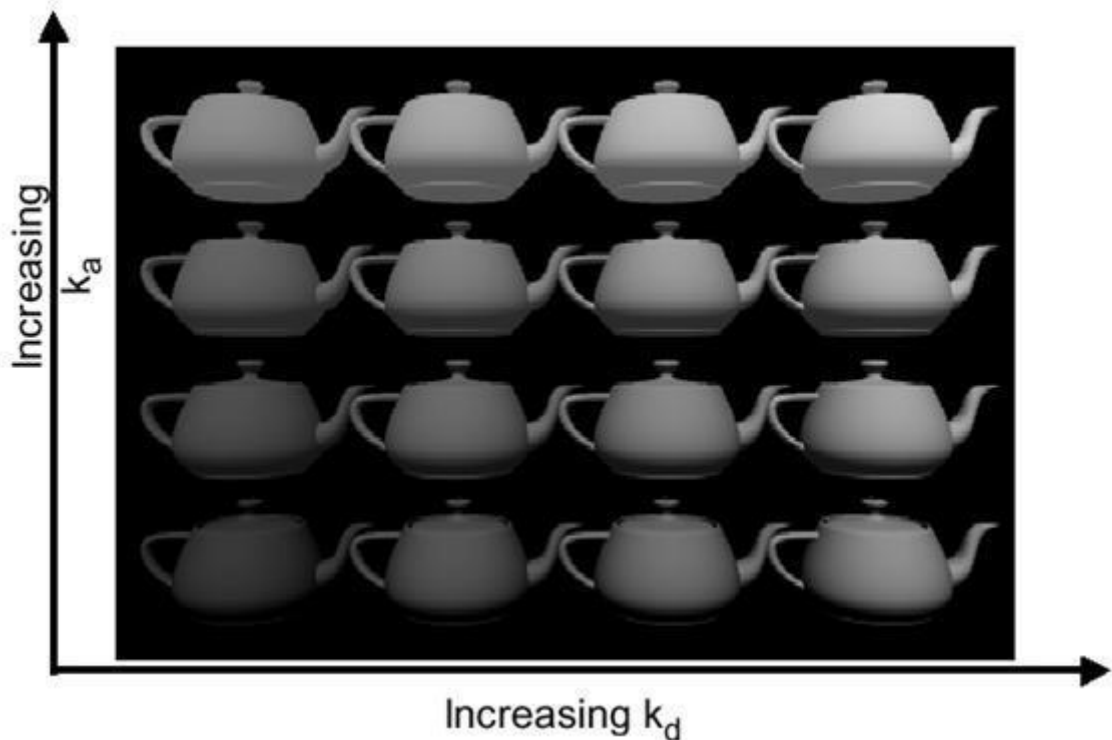
Fig. 4.16 Diffuse + ambient illumination

- Illumination decreases from the light source by I/d^2.
- Objects at a greater distance from the light source receive less illumination and appear darker.
- Above equation is distance independent.
- Dividing the Lambertian term by d^2 would seem to get the physics right, but it makes the intensity vary sharply over short distance.
- Modified distance dependence is employed, giving the following equation, where d is the distance from the light source to the object.(Fig 4.16)

$$I = I_a k_a + \frac{I_l k_d \cos\theta}{d+K} \qquad 0 \leq \theta \leq \pi/2$$

Specular Highlights – (Phong Reflection Model)

- Regions of significant brightness, exhibited as spots or bands, characterize objects that specularly reflect light.
- Specular highlights originate from smooth, sometimes mirrorlike surfaces
- Fresnel equation is used to simulate this effect.
- The Fresnel equation states that for a perfectly reflecting surface the angle of incidence equals the angle of reflection.(Fig 4.17)
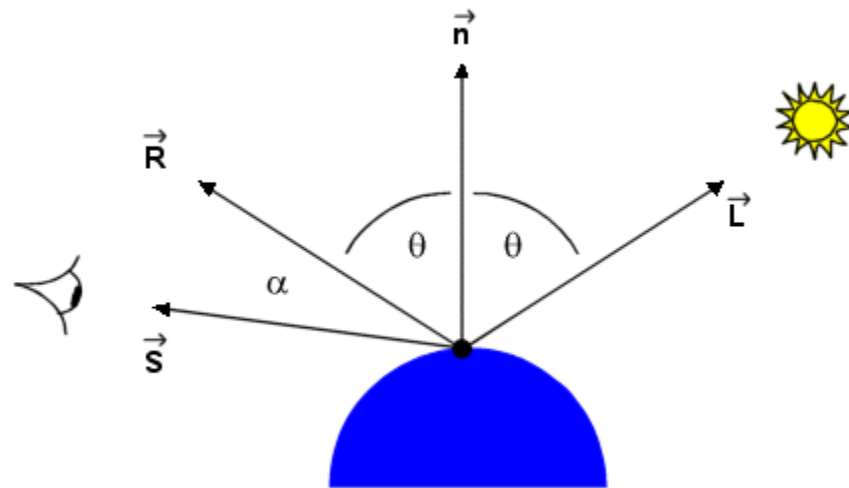
Fig 4.17 Phong Reflection Model

Shiny Surface
(Large n_s)

Dull Surface
(Small n_s)
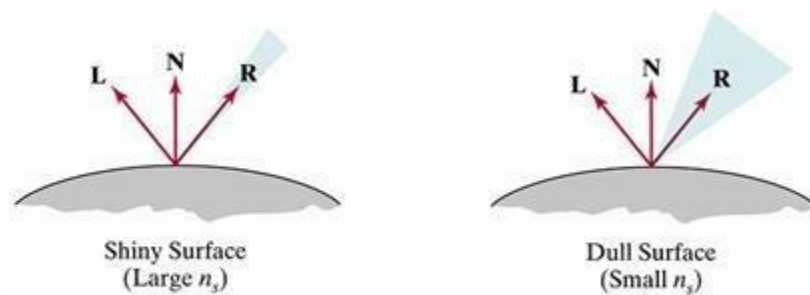
Modeling specular reflections (shaded area) with parameter n_s.

- Most objects are not perfect mirrors.
  - some angular scattering of light.
  - If the viewer increases the angle (a) between himself, the line of sight vector (**S**), and the reflectance vector (**R**), the bright spot gradually disappears.
  - Smooth surfaces scatter light less then rough surfaces.
  - This produces more localized highlights.

$$I_{l,specular} = W(\theta)I_l \cos^n \alpha$$
$$W(\theta) = specular\,reflection\,coefficient$$
$$n = specular\,reflection\,exponent$$
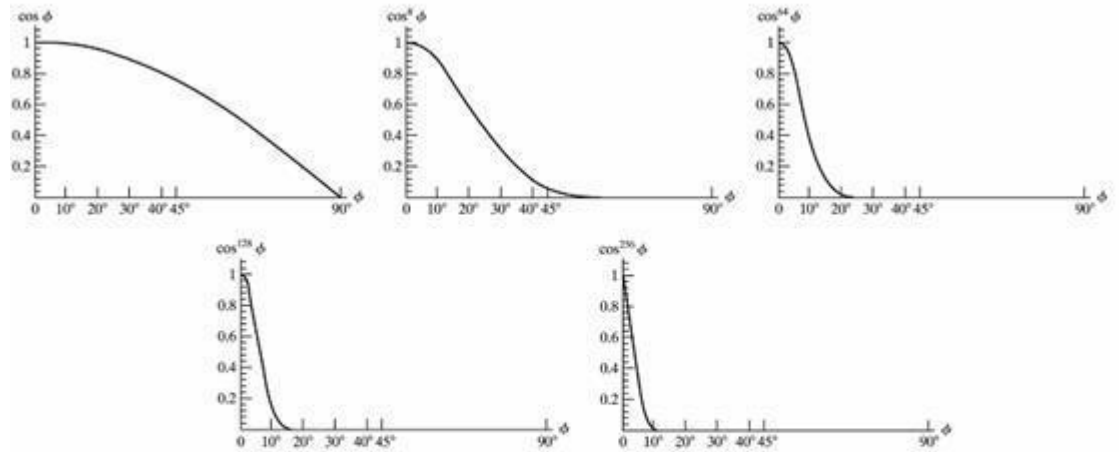
Plots of $\cos^{n_s} \phi$ using five different values for the specular exponent n_s.
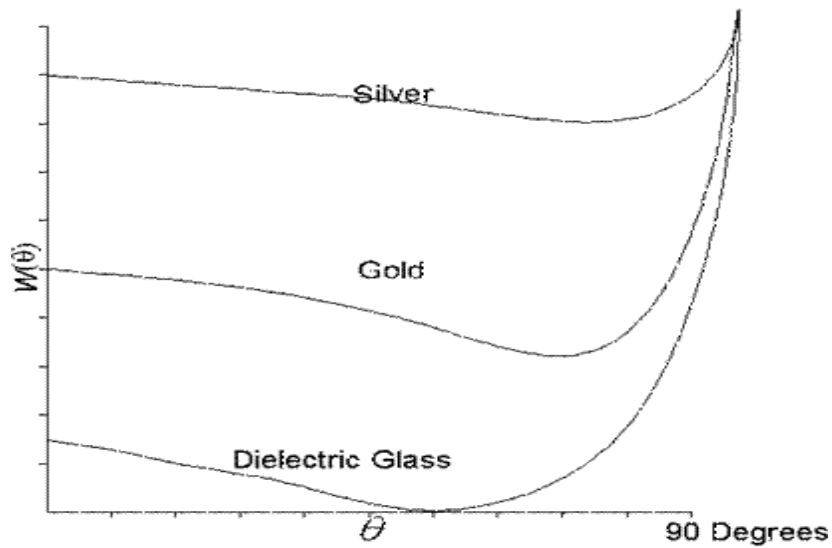
- Building this effect into the lighting model gives

$$I = I_a k_a + \frac{I_l}{d+K}\left(k_d \cos\theta + k_s \cos^n\alpha\right)$$

- Specular reflectance term possesses a specular reflectance constant, k_s.
- The cosine term is raised to the nth power.

  - Small values of n (e.g. 5) distribute the specular highlights, characteristic of glossy paper.
  - High values of n (e.g. 50) are characteristic of metals.(Fig 4.18)

17
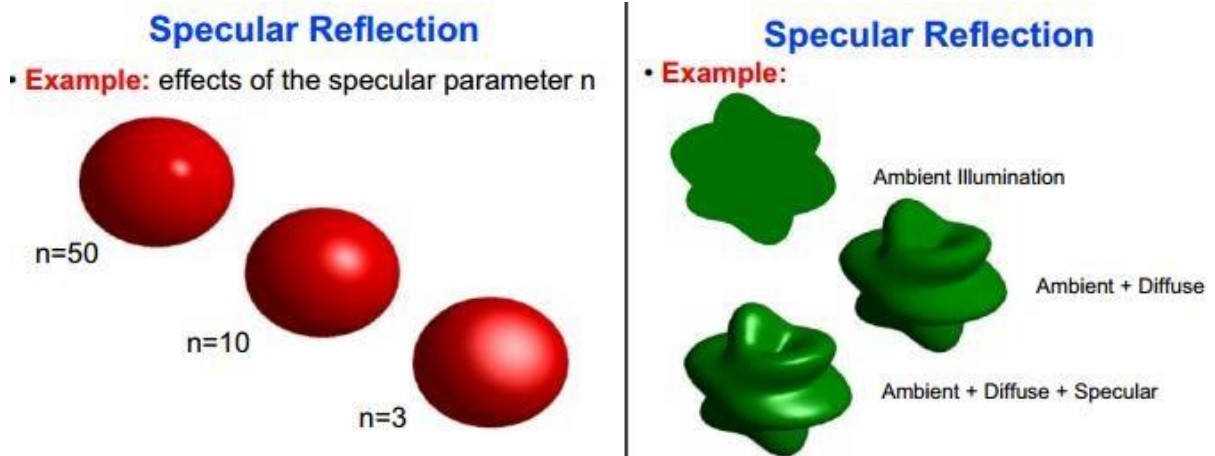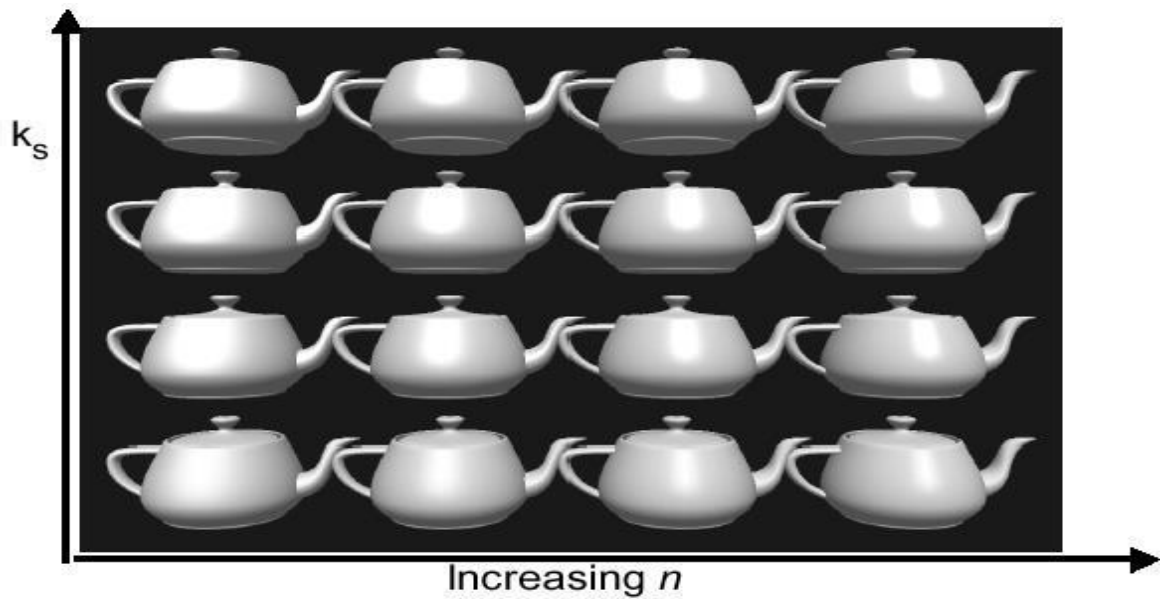
Fig 4.18 Example of Specular Reflection model

<u>Summary - Simple Illumination Model</u>

- Deficiencies
    - point light source
    - no interaction between objects
    - ad hoc, not based on model of light propagation
- Benefits
    - fast
    - acceptable results
    - hardware support

Polygon Rendering Methods

Flat Shading (also known as faceted shading)

Shades each polygon of an object based on the angle between the polygon's surface normal and the direction of the light source.

• A single intensity is calculated for each surface polygon

• Fast and simple method

• Gives reasonable result only if all of the following assumptions are valid:

    – The object is a polyhedron

    – Light source is far away from the surface so that N•L is constant over each polygon

    – Viewing position is far away from the surface so that V•R is constant over each polygon

Disadvantage - it gives low-polygon models a faceted look. Sometimes this look can be advantageous though, such as in modelling boxy objects. Artists sometimes use flat shading to look at the polygons of a solid model they are creating. Fig 4.19 shows flat shading.
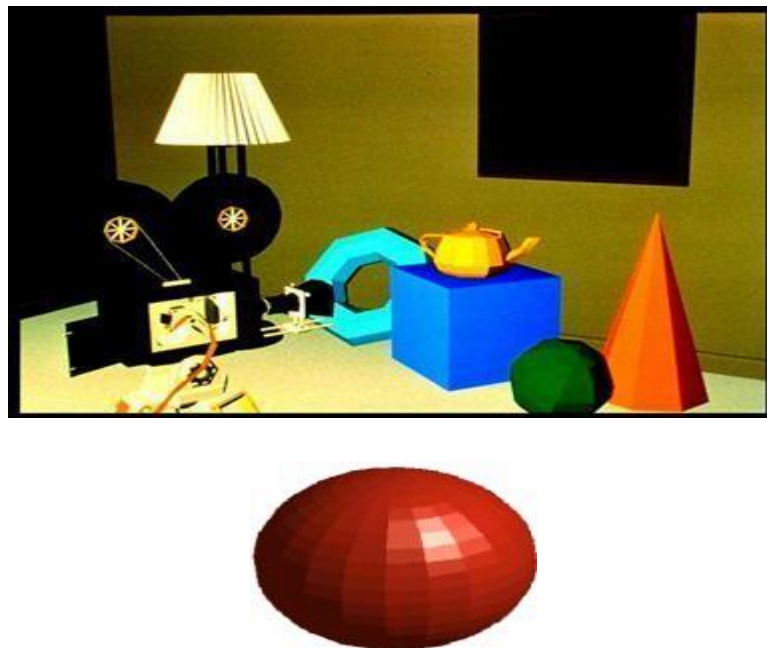
Fig 4.19 Example of Flat Shading model

Gouraud Shading

Gouraud shading (Henri Gouraud, 1971) is used to achieve smooth lighting on low-polygon surfaces using the Lambertiandiffuse lighting model.

    1.   Calculates the surface normals for the polygons.

2.   Normals are then averaged for all the polygons that meet at each vertex to produce a vertex normal.

3.   Lighting computations are then performed to produce intensities at vertices.

4.   These **intensities are interpolated** along the edges of the polygons.

5.   The polygon is filled by lines drawn across it that interpolate between the previously calculated edge intensities.

Advantage - Gouraud shading is superior to flat shading which requires significantly less processing than Gouraud, but gives low-polygon models a sharp, faceted look. Fig 4.20 shows Gouraud shading.
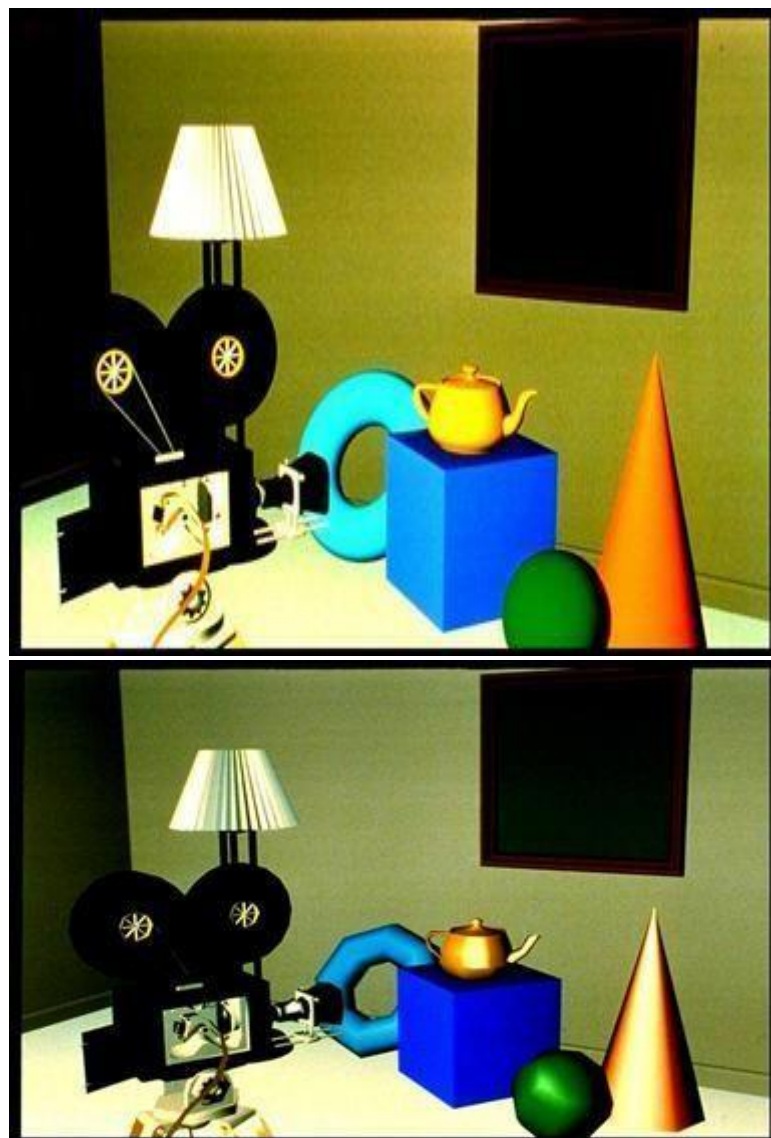
Fig 4.20 Example of Gouraud Shading model

Phong Shading (Phong Interpolation Model)

- An improved version of Gouraud shading that provides a better approximation to the Phong shading model(Fig 4.21) .
- Main problem with Gouraud shading - when a specular highlight occurs near the center of a large triangle, it will usually be missed entirely. Phong shading fixes the problem.

  1. Calculate the surface normals at the vertices of polygons in a 3D computer model.

  2. Normals are then averaged for all the polygons that meet at each vertex.

  3. These **normals are interpolated** along the edges of the polygons.

  4. Lighting computations are then performed to produce intensities at positions along scanlines.

Fig 4.21 Example of Phong Shading model
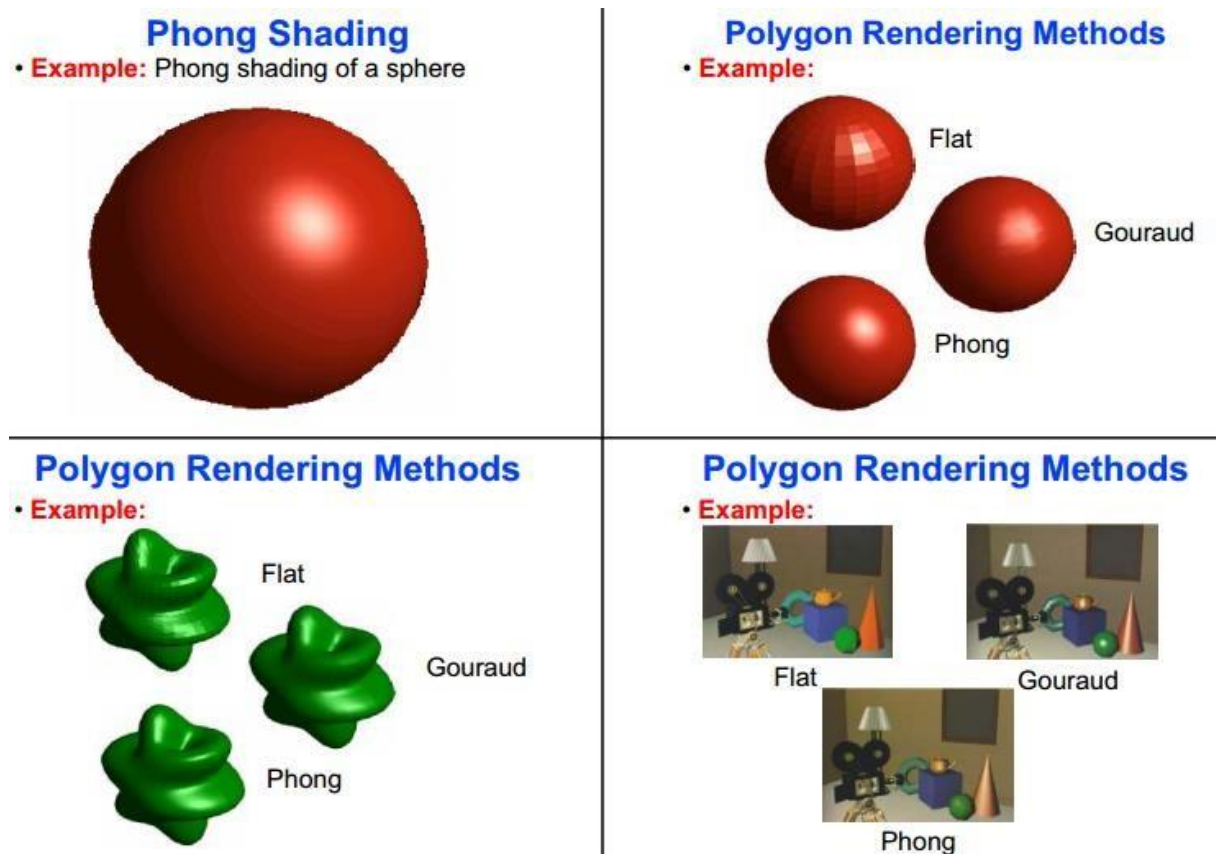
Fig 4.22 Comparison of Shading models

Fig 4.22 shows comparision of shading models.

Other Effects

Shadow

- 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. Fig 4.23 shows the shadow effect.

- 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.
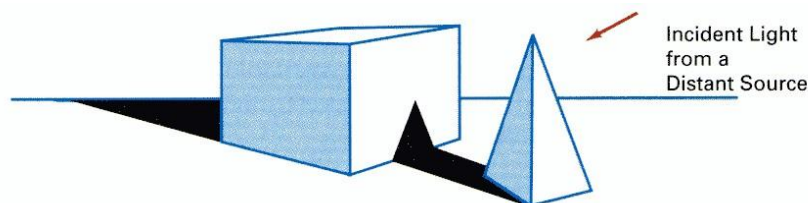
Fig 4.23 Shadow Effect

Texture Mapping

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(Fig 4.24).

- Texture pattern is defined in a MxN array of *texels* or a *texture map* indiced by (u,v) coordinates.

- For each pixel in the display:

  - Map the 4 corners of pixel back to the object surface (for cureved surfaces, these 4 points define a surface patch)

  - Map the surface patch onto the texture map (this mapping computes the source area in the texture map)

  - The pixel values is modified by weighted sum of the texels' color.

Fig 4.24 Texture Mapping

Bump Mapping

- To simulate surface roughness within the geometric description of the object, surface roughness can be generated by perturbing surface normals.
Fig 4.25 shows an example.
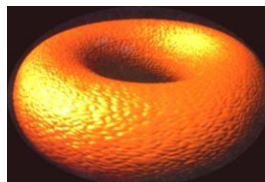
Fig 4.25 Bump Mapping

Fog Effect

- As an object is further away from the observer, the color of the object fades.

- Fog is a general term that describes similar forms of atmospheric effects. It can be used to simulate haze, mist, smoke, or pollution.

Ray Tracing Methods

- For each pixel on the image plane, a ray is projected from the center of projection through the pixel into the scene.

- The first object that the ray intersects is determined.

- The point at which the ray hits the object (ie. the ray intersection point) is also determined. The color value is then calculated according to the directions of the light sources to the surface normal.

- However, if there is another object located between a particular light source and the ray intersection point, then the point is in shadow and the light contribution from that particular light source is not considered.
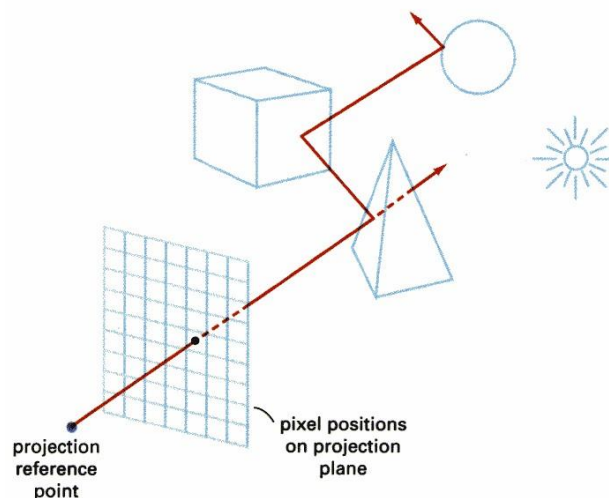
Fig 4.26 Ray Tracing

- The ray is then reflected and projected from the object until it intersects with another object as shown in Fig 4.26 (If the surface is a transparent surface, the ray is refracted as well as reflected.)
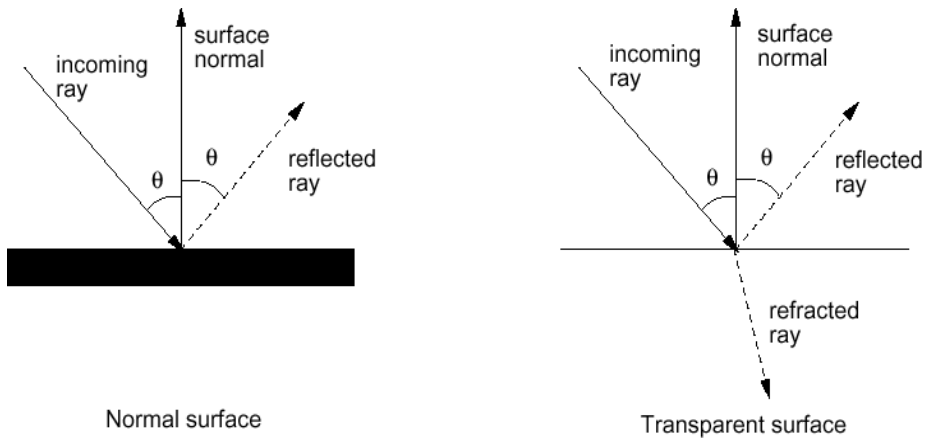
Fig 4.27 Ray Tracing Explained in Detail

- The point at which the reflected ray hits the second object are determined and the color value is again calculated in a similar way as the first object.

- The reflected ray is then reflected again from the second object as shown in Fig 4.27.. This process will continue until the color contribution of an intersected object is too small to be considered.

- In practical situation, se would specify the maximum number of reflections that a pixel can make to prevent spending too much processing at a particular pixel.

- All the color values calculated from the intersected objects will be weighted by the attenuation factors (which depend on the surface properties of the objects) and added up to produce a single color value. This color value becomes the pixel value.

Because ray-tracing method calculates the intensity value for each pixel independently, we can consider specular reflection and refraction in the calculation. Hence the method can generate very realistic images. The major problem of this method, however, is that it requires a lot of computations and therefore this method is slow.

To calculate the color of a pixel, consider the following diagram(Fig 4.28):
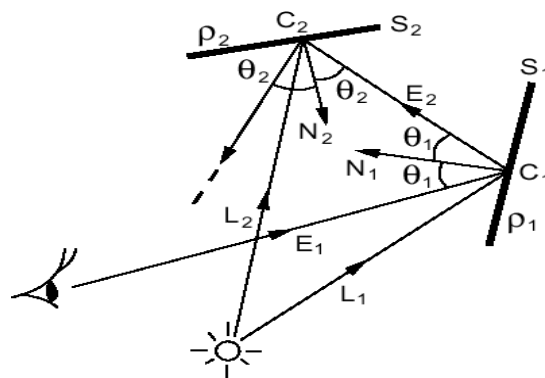
Fig 4.28 Ray Tracing Explained in Detail

- Assume that surfaces S_1 and S_2 have reflective indices $_1$ and $_2$ respectively.

- Given the surface normal vector, N_1, and the light vector, L_1, of surface S_1, we can calculate the color, C_1, of the surface at the point where the eye ray, E_1 intersects the surface.

- Similarly, given the surface normal vector, N_2, and the light vector, L_2, of surface S_2, we can calculate the color, C_2, of the surface at the point where the eye ray, E_2 intersects the surface.

- The color for the pixel is then calculated as: $C_p = C_1 + C_2 + \ldots$

Radiosity

Ray-tracing can model specular reflection very well, but not diffuse reflection. Why?
Recall that in diffuse reflection, although light is reflected with equal intensity in all directions, the amount of light energy received by another surface depends on the orientation of the surface relative to the source. Hence surfaces of different orientations may receive different amount of light.

Consider the diagram, assuming that all 3 surfaces are matte. Although A reflects light with equal intensity to B and C, B would receive more light energy than C because B has a smaller angle of incident (ie. higher cos).
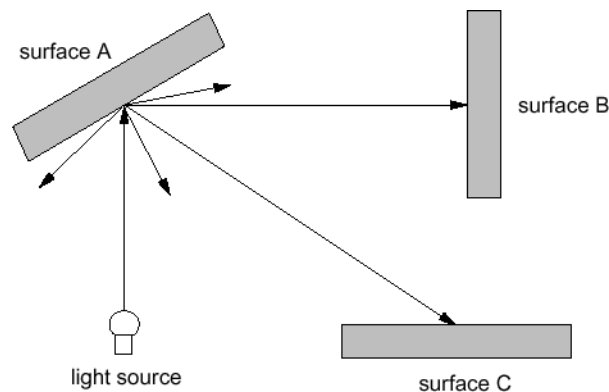
Fig 4.29 Radiocity

Radiosity is developed to model this kind of light interaction between surfaces as shown in Fig 4.29.
.

- The algorithm is based on the theory of the conservation of energy.

- In a closed environment such as a room, the rate at which energy leaves a surface, called its radiosity, is the sum of the rates at which the surface emits energy and it reflects

(or transmits) energy from other surfaces.

- To simplify the calculation, all surfaces in the scene are broken into small patches. Each of which is assumed to be of finite size, emitting and reflecting light uniformly over its entire area.

- The radiosity of a patch i, B_i, can be calculated as follows:

$$B_i = E_i + \rho_i \sum_{j \in all\ patches} L_{j \to i}$$

where $L_{j \to i}$ is the amount of light from patch j reaching patch i,
        E_i is the light emitted from patch i, and
        ρ_i is the reflectivity of patch i. (For an incident light, what ratio of energy is reflected)
- Once we have obtained a radiosity for each patch, we can render the scene with a scan-conversion method using the calculated radiosities as the intensities of the patches.

- Note that the radiosities calculated are view-independent. Hence, the radiosity method although can deal with diffuse reflection well, cannot deal with specular reflection. (Specular reflection is view-dependent).

- To be able to deal with both, we may combine the radiosity method with the ray-tracing method.

The prices are the added complexity of the algorithm and the increase in computational time. An example is a 2-pass approach that includes a view-independent radiosity process executed in the first pass, followed by a view-dependent ray-tracing approach in the second pass as shown in Fig 4.30.
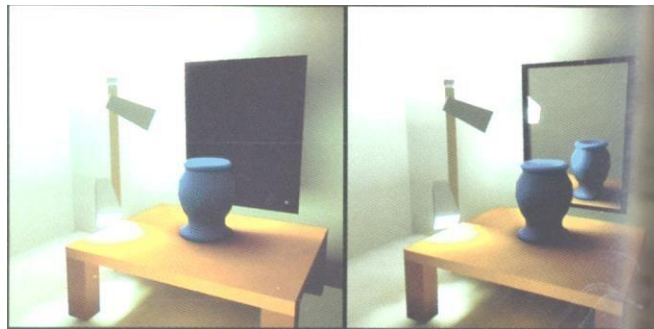
Fig 4.30 . Left: radiosity. Right: Diffuse first pass and ray-traing second pass.

Colour Image Processing

The human visual system can distinguish hundreds of thousands of different colour shades and intensities, but only around 100 shades of grey. Therefore, in an image, a great deal of extra information may be contained in the colour, and this extra information can then be used to simplify image analysis, e.g. object identification and extraction based on colour.

Three independent quantities are used to describe any particular colour. The *hue* is determined by the dominant wavelength. Visible colours occur between about 400nm (violet) and 700nm (red) on the electromagnetic spectrum, as shown in the below figure.
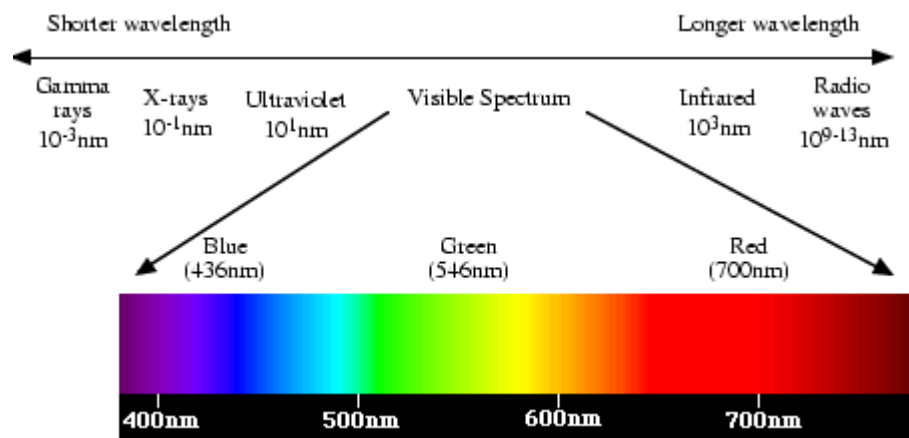
Fig 4.31. The visible spectrum

The *saturation* is determined by the excitation purity, and depends on the amount of white light mixed with the hue. A pure hue is fully saturated, i.e. no white light mixed in. Hue and saturation together determine the *chromaticity* for a given colour. Finally, the *intensity* is determined by the actual amount of light, with more light corresponding to more intense colours.Fig 4.31 shows the visible spectrum .

Achromatic light has no colour - its only attribute is quantity or intensity. Greylevel is a measure of intensity. The *intensity* is determined by the energy, and is therefore a physical quantity. On the other hand,*brightness* or *luminance* is determined by the perception of the colour, and is therefore psychological. Given equally intense blue and green, the blue is perceived as much darker than the green. Note also that our perception of intensity is nonlinear, with changes of normalised intensity from 0.1 to 0.11 and from 0.5 to 0.55 being perceived as equal changes in brightness.

Colour depends primarily on the reflectance properties of an object. We see those rays that are reflected, while others are absorbed. However, we also must consider the colour of the light source, and the nature of human visual system. For example, an object that reflects both red and green will appear green when there is green but no red light illuminating it, and conversely it will appear red in the absense of green light. In pure white light, it will appear yellow (= red + green).

Tristimulus theory of colour perception

The human retina has 3 kinds of cones. The response of each type of cone as a function of the wavelength of the incident light is shown in figure 2. The peaks for each curve are at 440nm (blue), 545nm (green) and 580nm (red). Note that the last two actually peak in the yellow part of the spectrum. Fig 4.32 shows the spectral response curve.
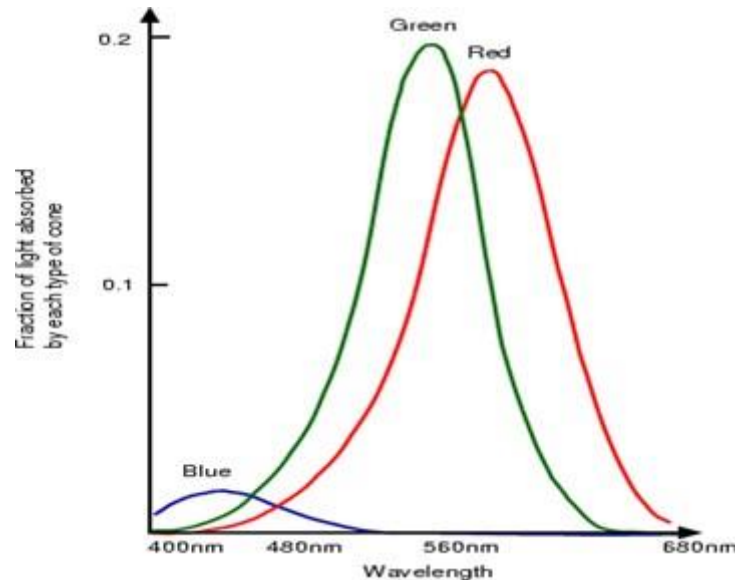
Fig 4.32 . Spectral response curves for each cone type. The peaks for each curve are at 440nm (blue), 545nm (green) and 580nm (red).

CIE primaries

The tristimulus theory of colour perception seems to imply that any colour can be obtained from a mix of the three primaries, red, green and blue, but although nearly all visible colours can be matched in this way, some cannot. However, if one of the primaries is added to one of these unmatchable colours, it can be matched by a mixture of the other two, and so the colour may be considered to have a negative weighting of that particular primary.

In 1931, the Commission Internationale de l'Éclairage (CIE) defined three standard primaries, called X, Y and Z, that can be added to form all visible colours. The primary Y was chosen so that its colour matching function exactly matches the luminous-efficiency function for the human eye, given by the sum of the three curves in the below figure.
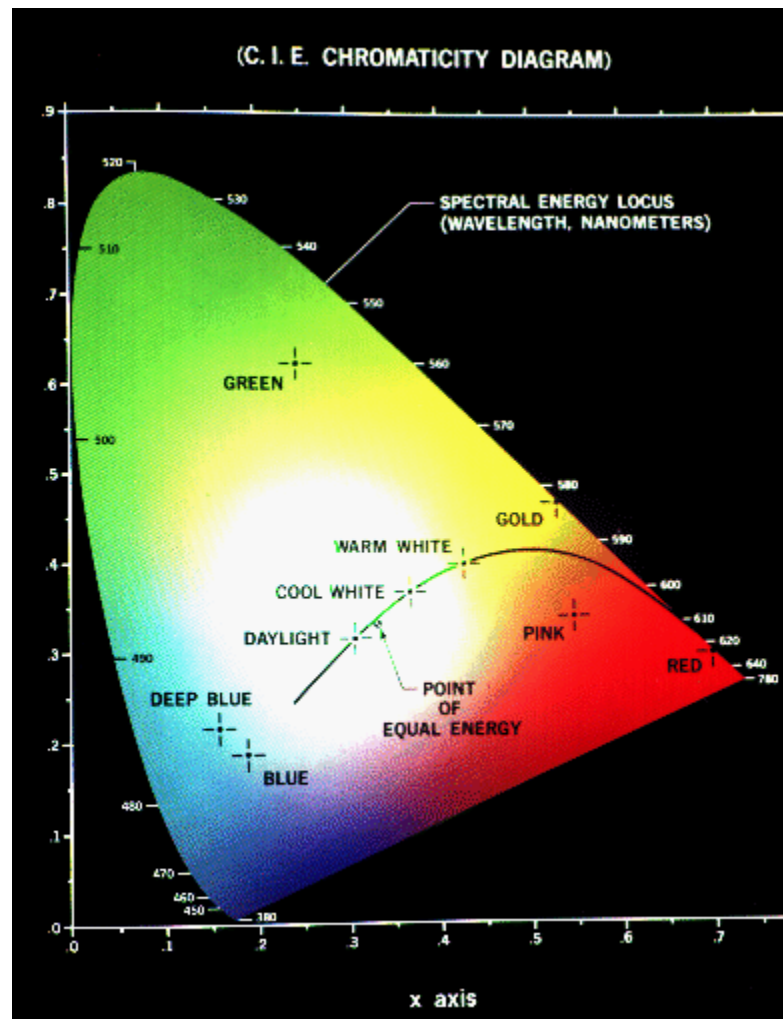
Fig 4.33 . CIE Chromaticity Curve

The CIE Chromaticity Diagram s (Fig 4.33) showing all visible colours. x and y are the normalised amounts of the X and Y primaries present, and hence z = 1 - x - y gives the amount of the Z primary required.

The CIE Chromaticity Diagram shows all visible colours. The x and y axis give the normalised amounts of the X and Y primaries for a particular colour, and hence z = 1 - x - y gives the amount of the Z primary required. Chromaticity depends on dominant wavelength and saturation, and is independent of luminous energy. Colours with the same chromaticity, but different luminance all map to the same point within this region.

The pure colours of the spectrum lie on the curved part of the boundary, and a standard white light has colour defined to be near (but not at) the point of equal energy x = y = z = 1/3. Complementary colours, i.e. colours that add to give white, lie on the endpoints of a line through this point. As illustrated in below figure, all the colours along any line in the chromaticity diagram may be obtained by mixing the colours on the end points of the line. Furthermore, all colours within a triangle may be formed by mixing the colours at the vertices. This property illustrates graphically the fact that all

visible colours cannot be obtained by a mix of R, G and B (or any other three visible) primaries alone, since the diagram is not triangular!
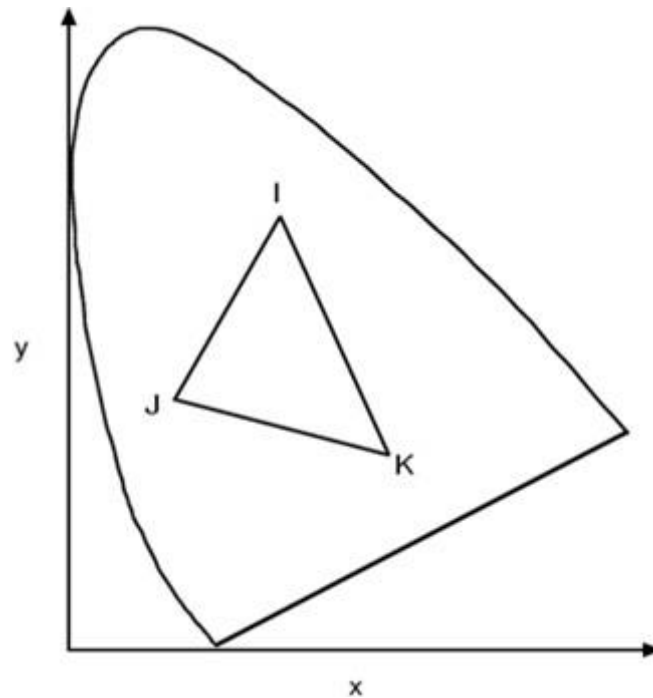
Fig 4.34 CIE chromaticity curve colour mixing

Mixing colours on the chromaticity diagram (Fig 4.34). All colours on the line IJ can be obtained by mixing colours I and J, and all colours in the triangle IJK can be obtained by mixing colours I, J and K.

Color Models

Colour models provide a standard way to specify a particular colour, by defining a 3D coordinate system, and a subspace that contains all constructible colours within a particular model. Any colour that can be specified using a model will correspond to a single point within the subspace it defines. Each colour model is oriented towards either specific hardware (RGB,CMY,YIQ), or image processing applications (HSI).

The RGB Model

In the RGB model, an image consists of three independent image planes, one in each of the primary colours: red, green and blue. Specifying a particular colour is by specifying the amount of each of the primary components present. Below figure shows the geometry of the RGB colour model for specifying colours using a Cartesian coordinate system. The greyscale spectrum, i.e. those colours made from equal amounts of each primary, lies on the line joining the black and white vertices.
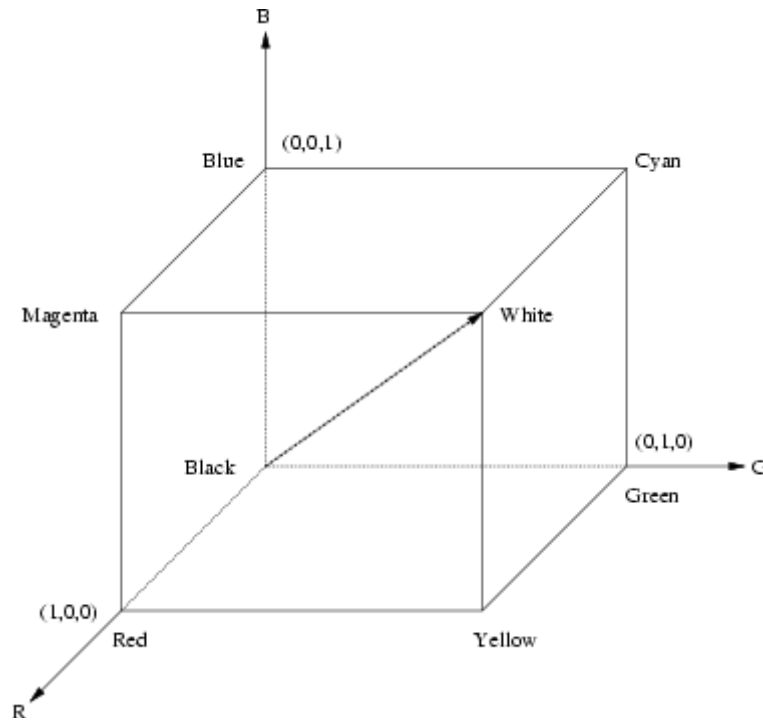
Fig 4.35. RGB colour cube.

The RGB colour cube is shown in (Fig 4.35)The greyscale spectrum lies on the line joining the black and white vertices.

This is an additive model, i.e. the colours present in the light add to form new colours, and is appropriate for the mixing of coloured light for example. The image on the left of figure 6 shows the additive mixing of red, green and blue primaries to form the three secondary colours yellow (red + green), cyan (blue + green) and magenta (red + blue), and white ((red + green + blue).

The RGB model is used for colour monitors and most video cameras.

The CMY Model

The CMY (cyan-magenta-yellow) model is a subtractive model appropriate to absorption of colours, for example due to pigments in paints. Whereas the RGB model asks what is added to black to get a particular colour, the CMY model asks what is subtracted from white. In this case, the primaries are cyan, magenta and yellow, with red, green and blue as secondary colours.

When a surface coated with cyan pigment is illuminated by white light, no red light is reflected, and similarly for magenta and green, and yellow and blue. The relationship between the RGB and CMY models is given by:

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}.$$

The CMY model is used by printing devices and filters.
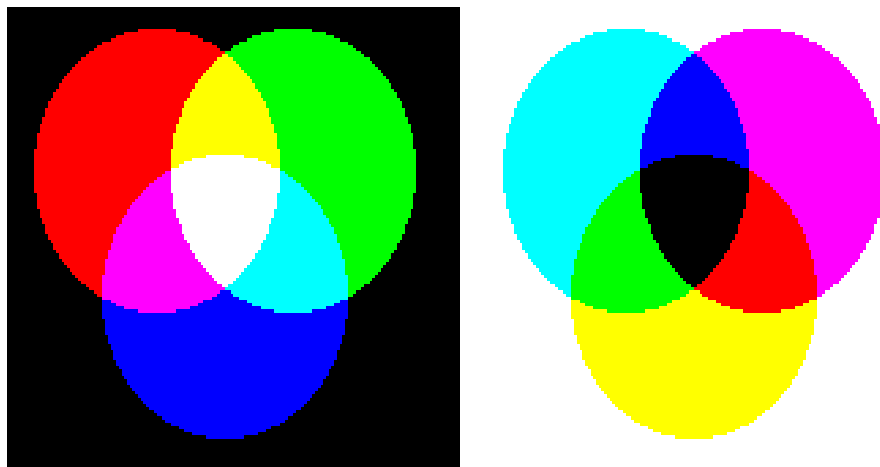
Fig 4.36 CMY Model

The Fig 4.36 on the left shows the additive mixing of red, green and blue primaries to form the three secondary colours yellow (red + green), cyan (blue + green) and magenta (red + blue), and white ((red + green + blue). The figure on the right shows the three subtractive primaries, and their pairwise combinations to form red, green and blue, and finally black by subtracting all three primaries from white.

The HSI Model

As mentioned above, colour may be specified by the three quantities hue, saturation and intensity. This is the HSI model, and the entire space of colours that may be specified in this way is shown in below figure.
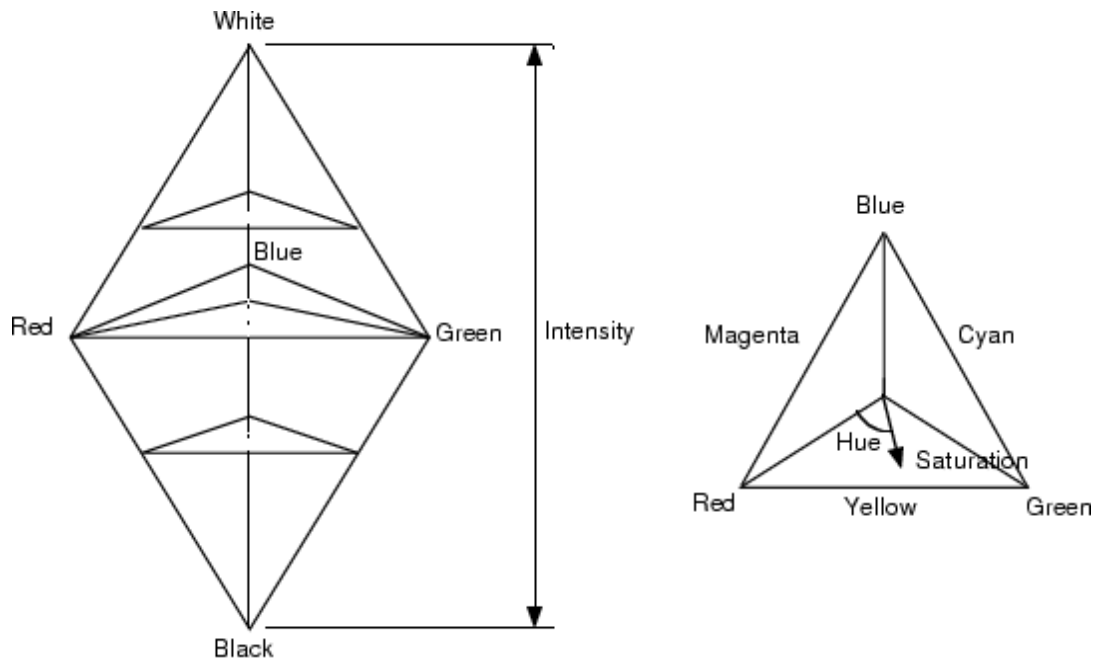
Fig 4.37 HSI Model

The HSI model, showing the HSI solid on the left, and the HSI triangle on the right, formed by taking a horizontal slice through the HSI solid at a particular intensity. Hue is measured from red, and saturation is given by distance from the axis. Colours on the surface of the solid are fully saturated, i.e. pure colours, and the greyscale spectrum is on the axis of the solid. For these colours, hue is undefined.(Fig 4.37)

Conversion between the RGB model and the HSI model is quite complicated. The intensity is given by,

$$I= (R+G+B)/3$$

where the quantities R, G and B are the amounts of the red, green and blue components, normalised to the range [0,1]. The intensity is therefore just the average of the red, green and blue components. The saturation is given by,

$$S= (min(R,G,B)) / I$$

where the min(R,G,B) term is really just indicating the amount of white present. If any of R, G or B are zero, there is no white and we have a pure colour.

The YIQ Model

The YIQ (luminance-inphase-quadrature) model is a recoding of RGB for colour television, and is a very important model for colour image processing. The conversion from RGB to YIQ is given by:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

The luminance (Y) component contains all the information required for black and white television, and captures our perception of the relative brightness of particular colours. That we perceive green as much lighter than red, and red lighter than blue, is indicated by their respective weights of 0.587, 0.299 and 0.114 in the first row of the conversion matrix above. These weights should be used when converting a colour image to greyscale if you want the perception of brightness to remain the same. This is not the case for the intensity component in an HSI image, as shown in below figure.

The Y component is the same as the CIE primary Y.

(a) Colour Image

(b) Intensity Image

(c) Luminance Image
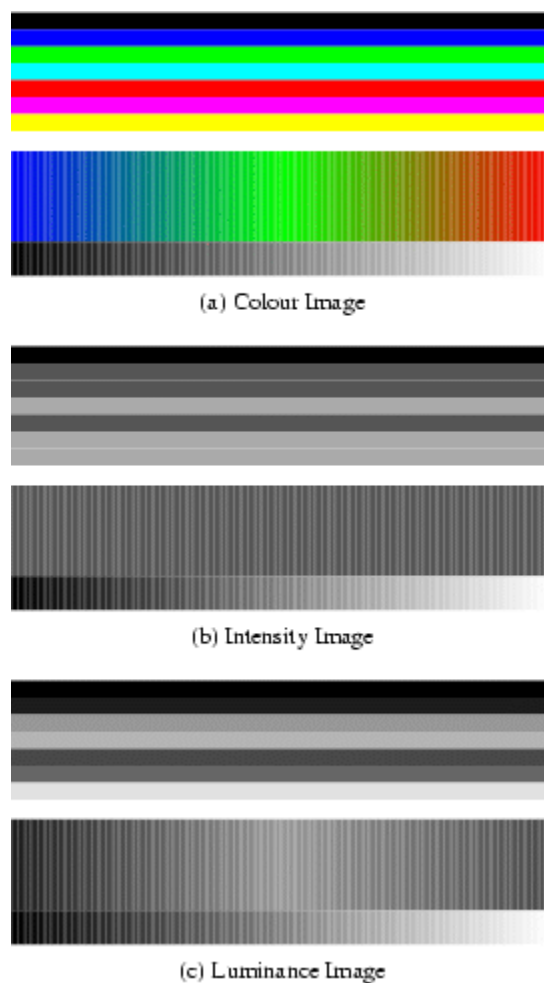
Fig 4.38 CIE Model

Fig 4.38 Image (a) shows a colour test pattern, consisting of horizontal stripes of black, blue, green, cyan, red, magenta and yellow, a colour ramp with constant intensity, maximal saturation, and hue changing linearly from red through green to blue, and a greyscale ramp from black to white. Image (b) shows the intensity for image (a). Note how much detail is lost. Image (c) shows the luminance. This third image accurately reflects the brightness variations preceived in the original image.

Applying Greyscale Transformations to Colour Images

Given all these different representations of colour, and hence colour images, the question arises as to what is the best way to apply the image processing techniques we have covered so far to these images? One possibility is to apply the transformations to each colour plane in an RGB image, but what exactly does this mean? If we want to increase the contrast in a dark image by histogram equalisation, can we just equalise each colour independently? This will result in quite different colours in our transformed image. In general it is better to apply the transformation to just the intensity component of an HSI image, or the luminance component of a YIQ image, thus leaving the chromaticity unaltered.

An example is shown in below Fig 4.39. When histogram equalisation is applied to each colour plane of the RGB image, the final image is lighter, but also quite differently coloured to the original. When histogram equalisation is only applied to the luminance component of the image in YIQ format, the result is more like a lighter version of the original image, as required.

Fig 4.39 YIQ Model

The top image is a very dark image of a forest scene. The middle image is the result of applying histogram equalisation to each of the red, green and blue components of the original image. The

bottom image is the result of converting the image to YIQ format, and applying histogram equalisation to the luminance component only.

Halftone Approximations

If we cannot display all the required intensity levels (e.g. on a printer) we need a trick using the spatial integration that our eyes performs: In normal light the eye can only detect about one arc minute (1/60 degree). This is called VISUAL ACUITY. Thus instead of gray dots a small black disk with radius varying according to the blackness (1-I) is printed. Usually, for newspaper 60-80 and for magazines 150-200 different radiuses are used. This process is called HALFTONING. (Fig 4.40)
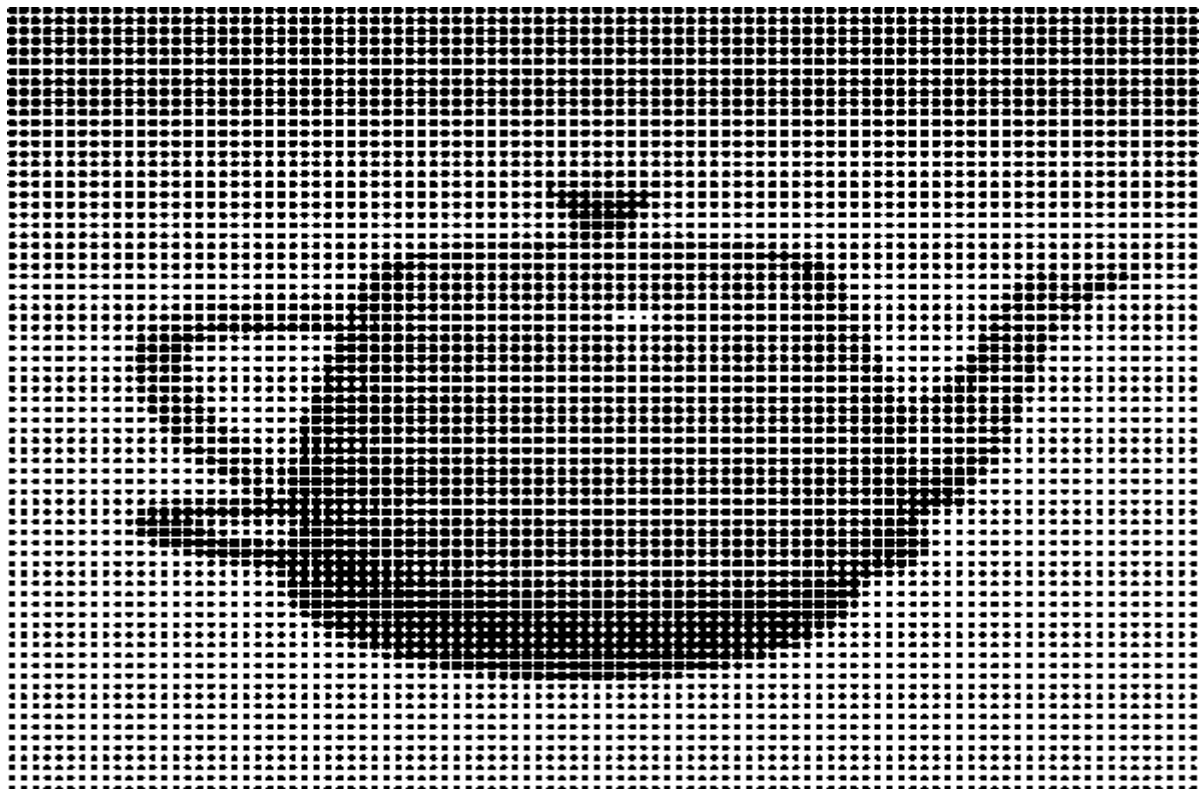
Fig 4.40 Halftoning

For computers this is implemented as CLUSTERED-DOT ORDERED DITHERING, e.g. the following patterns are used for each pixel of intensity $0 \ldots 9$: (Fig 4.41)
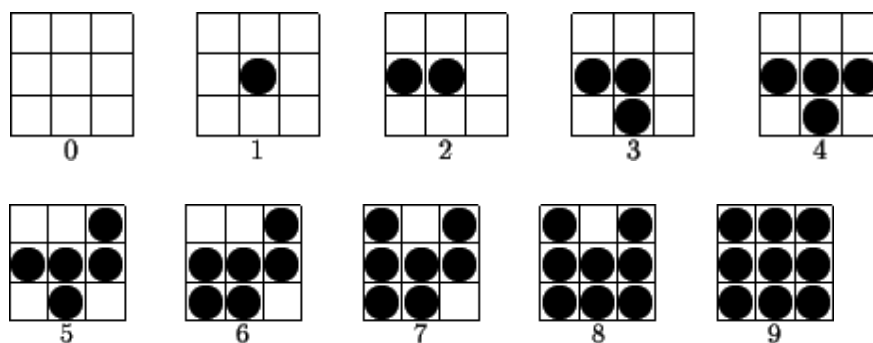
Fig 4.41 Clustered-dot ordered dithering
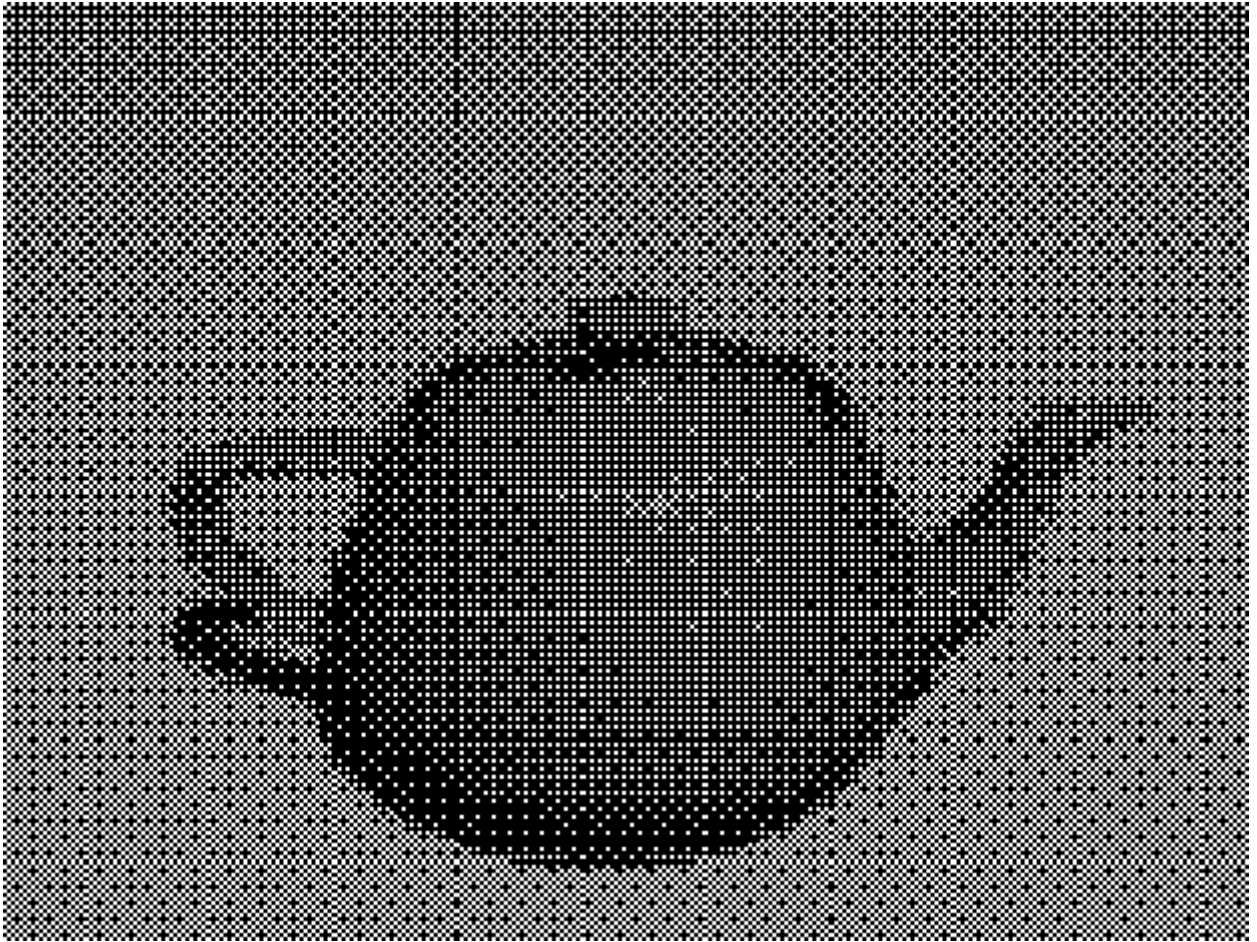
Fig 4.42 Clustered-dot ordered dithering

This can be described by a dither matrix

$$\begin{pmatrix} 6 & 8 & 4 \\ 1 & 0 & 3 \\ 5 & 2 & 7 \end{pmatrix}$$

saying that in order to display intensity I one should turn on all those pixels whose value in this matrix is less than I. It is chosen in such a way, that visual artifacts are avoided:

Growth sequence to minimize contour effects, i.e. the pattern are subsets of each other (hence the name ordered). They start in the center and expand towards the boundary.(Fig 4.42)

Keep the one's adjacent to each other (hence the name clustered dot), but this is not necessary for monitors.

For high-quality reproduction we need an 8x8 (64 gray levels) or even a 10x10 matrix and thus quite a highest resolution.

If such a high resolution is not available one can use ERROR DIFFUSION: There we use fewer intensity levels than desired, but we distribute the errors, (1-I$_i$) to the neighbouring pixels with the following weights:
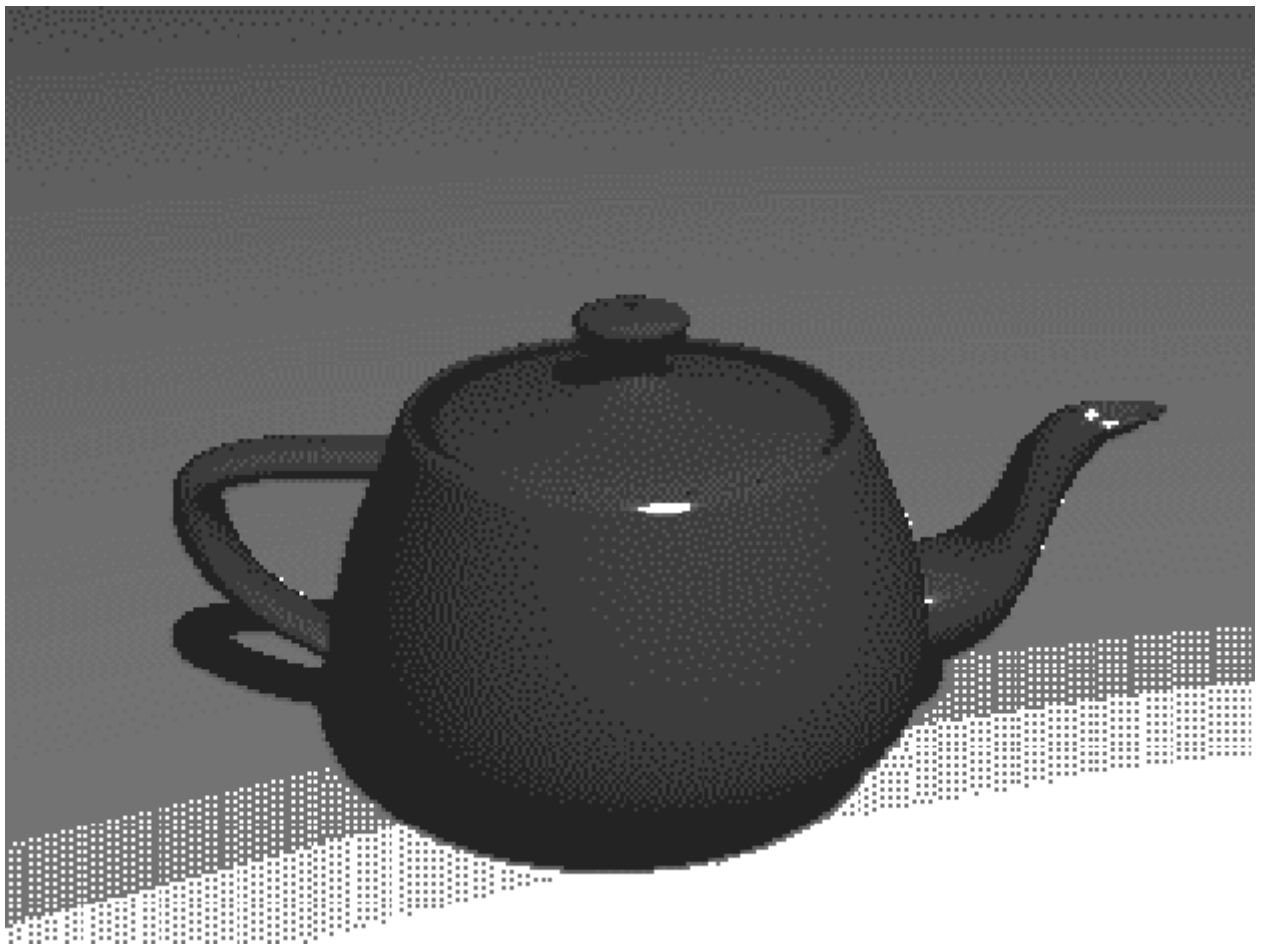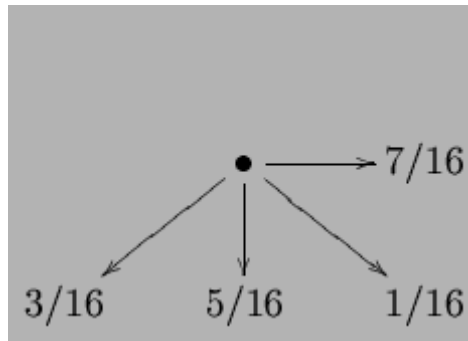
Fig 4.43- 8 colors with Floyd-Steinberg error diffusion

A similar trick can be used for enlarging a picture by taking interpolation values to neighbouring pixels as intermediate values. E.g. for doubling the size of the picture one inserts new rows and columns and takes as new values (Fig 4.43)

$$I'_{2i,2j} := I_{i,j} \qquad\qquad I'_{2i+1,2j} := \frac{1}{2}(I_{i,j} + I_{i+1,j})$$

$$I'_{2i,2j+1} \; := \frac{1}{2}(I_{2i,j} + I_{2i,j+1}) \qquad\qquad I'_{2i+1,2j+1} \; := \frac{1}{4}(I_{i,j} + I_{i,j+1} + I_{i+1,j} + I_{i+1,j+1})$$

This way one avoids that small square-size pixels can be seen. Disadvantage is that some blurring occurs.(Fig 4.44)
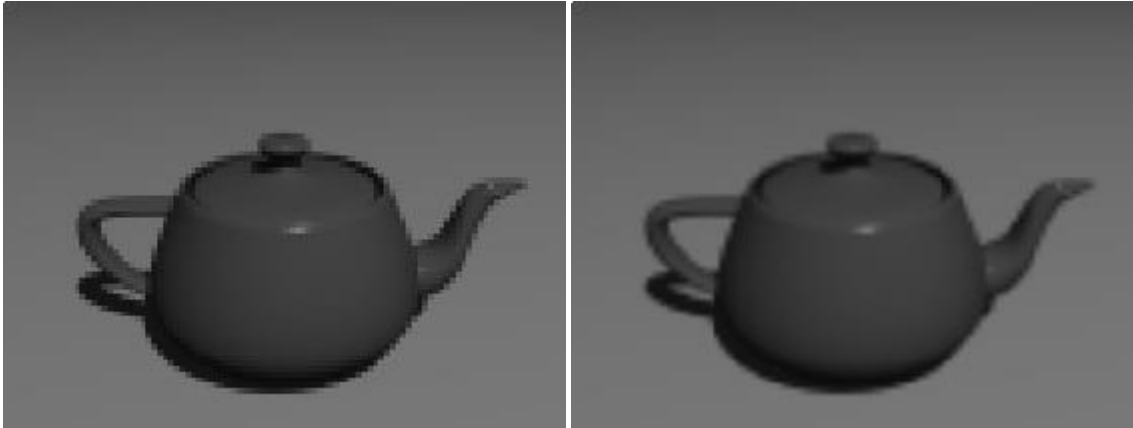
Fig 4.44 Enlarged versus enlarged with diffusion

Half toning- Main Points

- Many displays and hardcopy devices are bi-level
- They can only produce two intensity levels.
- In such displays or hardcopy devices we can create an apparent increase in the number of available intensity value.
- When we view a very small area from a sufficient large viewing distance, our eyes average fine details within the small area and record only the overall intensity of the area.
- The phenomenon of apparent increase in the number of available intensities by considering combine intensity of multiple pixels is known as half toning.
- The half toning is commonly used in printing black and white photographs in newspapers, magazines and books.
- The pictures produced by half toning process are called halftones.
- In computer graphics, halftone reproductions are approximated using rectangular pixel regions, say 22 pixels or 33 pixels.
- These regions are called halftone patterns or pixel patterns.
- Figure shows the halftone pattern to create number of intensity levels.(Fig 4.45)

2×2 Pixel patterns for creating five intensity levels

3×3 Pixel patterns for creating ten intensity levels
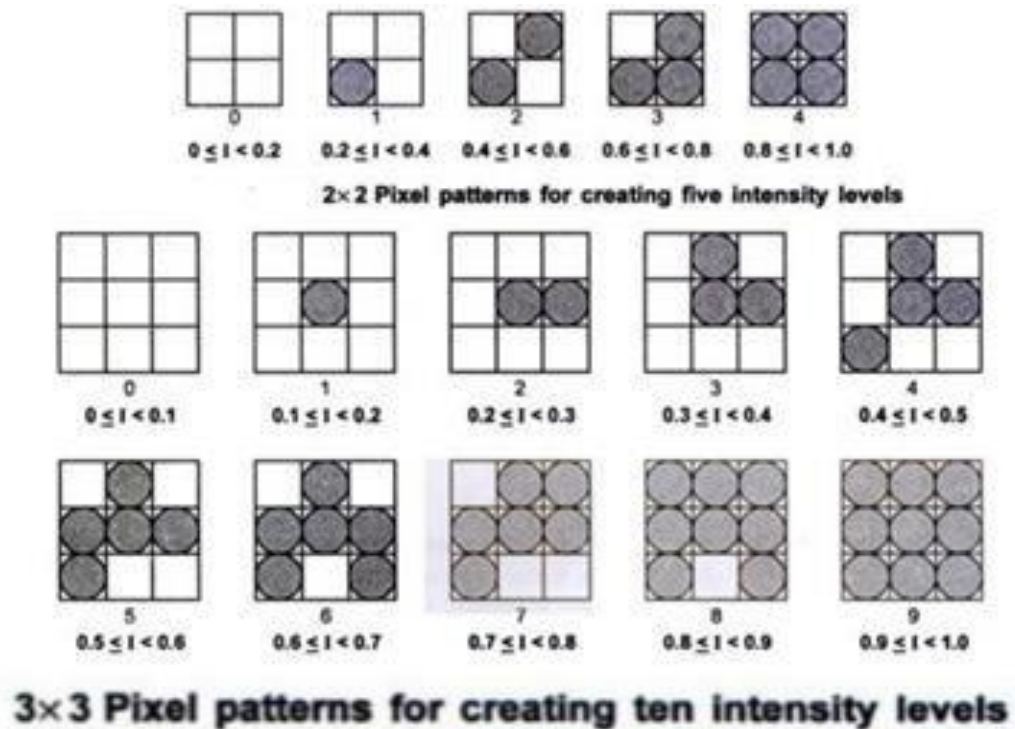
Fig 4.45. Halftoning Pixel patterns

Dithering Techniques

- Dithering refers to techniques for approximating halftones without reducing resolution, as pixel grid patterns do.
- The term dithering is also applied to halftone approximation method using pixel grid, and something it is used to refer to color halftone approximations only.
- Random values added to pixel intensities to break up contours are often referred as dither noise.
- Number of methods is used to generate intensity variations.
- Ordered dither methods generate intensity variations with a one-to-one mapping of points in a scene to the display pixel.
- To obtain n^2 intensity levels, it is necessary to set up an $n*n$ dither matrix D_n whose elements are distinct positive integers in the range of 0 to $n^2 - 1$.
- For example it is possible to generate four intensity levels with

$$D_2 = \begin{matrix} 3 & 1 \\ 0 & 2 \end{matrix}$$ and it is possible to generate nine intensity levels with

$$D_3 = \begin{matrix} 7 & 2 & 6 \\ 4 & 0 & 1 \\ 3 & 8 & 5 \end{matrix}$$

- The matrix element for D_2 and D_3 are in the same order as the pixel mask for setting up 22 and 33 pixel grid respectively.
- For bi-level system, we have to determine display intensity values by comparing input intensities to the matrix elements.
- Each input intensity is first scaled to the range 0< I < n2.
- If the intensity I is to be applied to screen position (x, y), we have to calculate numbers for the either matrix as

$$i=(x \bmod n)+1 \quad j=(y \bmod n)+1$$

- If I>Dn(i,j) the pixel at position (x, y) is turned on; otherwise the pixel is not turned on.
- Typically the number of intensity levels is taken to be a multiple of 2.
- High order dither matrices can be obtained from lower order matrices with the recurrence relation.

$$D_n = \begin{matrix} 4D_{n/2} + D_2(1,1)u_{n/2} & 4D_{n/2} + D_2(1,2)u_{n/2} \\ 4D_{n/2} + D_2(2,1)u_{n/2} & 4D_{n/2} + D_2(2,2)u_{n/2} \end{matrix}$$

- Another method for mapping a picture with mn points to a display area with mn pixels is error diffusion.
- Here, the error between an input intensity value and the displayed pixel intensity level at a given position is dispersed, or diffused to pixel position to the right and below the current pixel position

References

- R.C. Gonzales and R.E. Woods, Digital Image Processing, Addison-Wesley, Reading, 1992.

- J.D. Foley, A. van Dam, S.K. Feiner and J.F. Hughes, Computer Graphics, Principles and Practice, Addison-Wesley, Reading, 1990.

- Newton, Opticks, 4th Edition, Dover, New York, 1704/1952.

- E.B. Goldstein, Sensation and Perception, Brooks/Cole, California, 1989.

- http://www.mat.univie.ac.at/~kriegl/Skripten/CG/node11.html

- http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT14/lecture12.html

- http://www.mat.univie.ac.at/~kriegl/Skripten/CG/node6.html

- http://www.ques10.com/p/11466/explain-half-toning-and-dithering-techniques/

UNIT-V - SCSA1503- Computer Graphics and Multimedia Applications

Introduction to multimedia - Compression & Decompression - Data & File Format standards - Digital voice and audio - Video image and animation. Introduction to Photoshop – Workplace – Tools – Navigating window – Importing and exporting images – Operations on Images – resize, crop, and rotate. Introduction to Flash – Elements of flash document – flash environment – Drawing tools – Flash animations – Importing and exporting - Adding sounds-Publishing flash movies – Basic action scripts – GoTo, Play, Stop, Tell Target.

Multimedia Basics and Tools

Introduction to Multimedia

Multimedia is a combination of text, graphic art, and sound, animation and video elements.

The IBM dictionary of computing describes multimedia as "comprehensive material, presented in a combination of text, graphics, video, animation and sound. Any system that is capable of presenting multimedia, is called a multimedia system". A multimedia application accepts input from the user by means of a keyboard, voice or pointing device. Multimedia applications involve using multimedia teclmology for business, education and entertainment. Multimedia is now available on standard computer platforms. It is the best way to gain attention of users and is widely used in many fields as follows:

- Business - In any business enterprise, multimedia exists in the form of advertisements, presentations, video conferencing, voice mail, etc.
- Schools - Multimedia tools for learning are widely used these days. People of all ages learn easily and quickly when they are presented information with the visual treat.
- Home PCs equipped with CD-ROMs and game machines hooked up with TV screens have brought home entertainment to new levels. These multimedia titles viewed at home would probably be available on the multimedia highway soon.
- Public places - Interactive maps at public places like libraries, museums, airports and the stand-alone terminal
- Virtual Reality (VR) - This technology helps us feel a 'real life-like' experience. Games using virtual reality effect is very popular

Multimedia Elements

High-impact multimedia applications, such as presentations, training and messaging, require the use of moving images such as video and image animation, as well as sound (from the video images as well as overlaid sound by a narrator) intermixed with document images and graphical text displays. Multimedia applications require dynamic handling of data consisting of a mix of text, voice, audio components, video components, and image animation. Integrated multimedia applications allow the user to cut sections of all or any of these components and paste them in a new document or in another application such as an animated sequence of events, a. desktop publishing system, or a spreadsheet.

FacsimileFacsimile transmissions were the first practical means of transmitting document images overtelephone lines. The basic technology, now widely used, has evolved to allow higher scanning

density for better-quality fax

Document images Document images are used for storing business documents that must be retained forlong periods oftime or may need to be accessed by a large number of people. Providing multimedia access to such documents removes the need far making several copies ofthe original for storage or *distribution*

Photographic images Photographic images are used for a wide range of applications . such asemployee records for instant identification at a security desk, real estates systems with photographs of houses in the database containing the description of houses, medical case histories, and so on.

Geographic information systems map (GIS)
Map created in a GIS system are being used wildly for natural resources and wild life management as well as urban planning. These systems store the geographical information of the map along with a database containing information relating highlighted map elements with statistical or item information such as wild life statistics or details of the floors and rooms and workers in an office building

Voice commands and voice synthesis Voice commands and voice synthesis are used for hands-freeoperations of a computer program. Voice synthbsis is used for presenting the results of an action to the user in a synthesized voice. Applications such as a patient monitoring system in a surgical theatre will be prime beneficiaries of these capabilities. Voice commands allow the user to direct computer operation by spoken commands

Audio message Annotated voice mail already uses audio or voice message as attachments to memos anddocuments such as maintenance manuals.

Video messages Video messages are being used in a manner similar to annotated voice mail.

Holographic images

All of the technologies so for essentially present a flat view of information. Holographic images extend the concept of virtual reality by allowing the user to get "inside" a part, such as, an engine and view its operation from the inside.

Fractals

Fractals started as a technology in the early 1980s but have received serious attention only recently. This technology is based on synthesizing and storing algorithms that describes the information.

Compression and Decompression

Compression is the way of making files to take up less space. In multimedia systems, in order to manage large multimedia data objects efficiently, these data objects need to be compressed to reduce the file size for storage of these objects.
 Compression tries to eliminate redundancies in the pattern of data.
 For example, if a black pixel is followed by 20 white pixels, there is no need to store all 20 white pixels. A coding mechanism can be used so that only the count of the white pixels is stored. Once

such redundancies are removed, the data object requires less time for transmission over a network. This in turn significantly reduces storage and transmission costs.

Types of Compression

Compression and decompression techniques are utilized for a number of applications, such as facsimile system, printer systems, document storage and retrieval systems, video teleconferencing

systems, and electronic multimedia messaging systems. An important standardization of compression algorithm was achieved by the CCITT when it specified Group 2 compression for facsimile system. When information is compressed, the redundancies are removed. Sometimes removing redundancies is not sufficient to reduce the size of the data object to manageable levels. In such cases, some real information is also removed. The primary criterion is that removal of the real information should not perceptly affect the quality of the result. In the case of video, compression causes some information to be lost; some information at a delete level is considered not essential for a reasonable reproduction of the scene. This type of compression is called lossy compression. Audio compression, on the other hand, is not lossy. It is called lossless compression.

Lossless Compression

In lossless compression, data is not altered or lost in the process of compression or decompression. Decompression generates an exact replica of the original object. Text compression is a good example of lossless compression. The repetitive nature of text, sound and graphic images allows replacement of repeated strings of characters or bits by codes. Lossless compression techniques are good for text data and for repetitive data in images all like binary images and gray-scale images.

Some of the commonly accepted lossless standards are given below:
- Packpits encoding (Run-length encoding)
- CCITT Group 3 I D
- CCITT Group 3 2D
- CCITT Group 4
- Lempe l-Ziv and Welch algorithm LZW.

Lossy compression is that some loss would occur while compressing information objects. Lossy compression is used for compressing audio, gray-scale or color images, and video objects in which absolute data accuracy is not necessary. The idea behind the lossy compression is that, the human eye fills in the missing information in the case of video. But, an important consideration is how much information can be lost so that the result should not affect. For example, in a grayscale image, if several bits are missing, the information is still perceived in an acceptable manner as the eye fills in the gaps in the shading gradient. Lossy compression is applicable in medical screening systems, video tele-conferencing, and multimedia electronic messaging systems.

Lossy compressions techniques can be used alone only in combination with other compression methods in a multimedia object consisting of audio, color images, and video as well as other specialized data types. The following lists some of the lossy compression mechanisms:
- Joint Photographic Experts Group (JPEG)
- Moving Picture Experts Group (MPEG)
- Intel DVI
- CCITT H.261 (P * 24) Video Coding Algorithm
- Fractals.

Binary Image compression schemes

Binary Image Compression Scheme is a scheme by which a binary image containing black and white pixel is generated when a document is scanned in a binary mode. The schemes are used primarily for documents that do not contain any continuous-tone information or where the continuous-tone information can be captured in a black and white mode to serve the desired purpose. The schemes are applicable in office/business documents, handwritten text, line graphics,

engineering drawings, and so on. Let us view the scanning process.A scanner scans a document as sequential scan lines, starting from the top of the page. A scan line is complete line of pixels, of height equal to one pixel, running across the page. It scans the first line of pixels (Scan Line), then scans second "line, and works its way up to the last scan line of the page. Each scan line is scanned from left to right of the page generating black and white pixels for that scan line.

This uncompressed image consists of a single bit per pixel containing black and white pixels. Binary 1 represents a black pixel, binary 0 a white pixel. Several schemes have been standardized and used to achieve various levels of compressions. Let us review the more commonly used schemes.

1. PackpitsEncoding(Run-Length Encoding)

It is a scheme in which a consecutive repeated string of characters is replaced by two bytes. It is the simple, earliest of the data compression scheme developed. It need not to have a standard. It is used to compress black and white (binary) images. Among two bytes which are being replaced, the first byte contains a number representing the number of times the character is repeated, and the second byte contains the character itself.In some cases, one byte is used to represent the pixel value, and the other seven bits to represents the run length.

2. CCITT Group 3 1-D Compression

This scheme is based on run-length encoding and assumes that a typical scanline has long runs of the same color.This scheme was designed for black and white images only, not for gray scale or color images. The primary application of this scheme is in facsimile and early document imaging system.

Huffman Encoding

A modified version of run-length encoding is Huffman encoding. It is used for many software based document imaging systems. It is used for encoding the pixel run length in CCITT Group 3 1-dGroup 4. It is variable-length encoding. It generates the shortest code for frequently occurring run lengths and longer code for less frequently occurring run lengths.

Mathematical Algorithm for huffman encoding:

Huffman encoding scheme is based on a coding tree.It is constructed based on the probability of occurrence of white pixels or black pixels in the run length or bit stream. Table 5.1 below shows the CCITT Group 3 tables showing codes or white run lengths and black run lengths.

White Run Length	Code Word	Black Run Length	Code Word
0	00110101	0	0000110111
1	000111	1	010
2	0111	2	11
3	1000	3	10
4	1011	4	011
5	1100	5	0011
6	1110	6	0010
7	1111	7	00011
8	10011	8	000101
9	10100	9	000100
10	00111	10	0000100
11	01000	10	0000100

11	01000	10	0000100
11	01000	11	0000101
12	001000	12	0000111
13	000011	13	00000100
14	110100	14	00000111
15	110101	15	000011000
16	101010	16	0000010111
17	101011	17	0000011000
18	0100111	18	0000001000
19	0001100	19	0000 11 00 III
20	0001000	20	00001101000
21	0010111	21	00001101100
22	0000011	22	00000110111
23	0000100	23	00000101000
24	0101000	24	00000010111
25	0101011	25	00000011000
26	0010011	26	000011001010
27	0100100	27	000011001011
28	0011000	28	000011 00 11 00
29	00000010	29	000011001101
30	00000011	30	000001101000
31	00011010	31	000001101001
32	00011011	32	000001101010
33	00010010	33	000001101011
34	00010011	34	000011010010
35	00010100	35	000011 0 10011

Fig 5.1 CCITT Group 3 tables

For example, from the above table, the run-length code of 16 white pixels is 101010, and of 16 black pixels 0000010111. Statistically, the occurrence of 16 white pixels is more frequent than the occurrence of 16 black pixels. Hence, the code generated for 16 white pixels is much shorter. This allows for quicker decoding. For this example, the tree structure could be constructed.

36	00010101	36	000011010100
37	00010110	37	000011010101
38	000101 II	38	000011010110
39	00101000	39	000011 0 1 0 1 1
40	00101001	40	000001I01100
41	00101010	41	000001101101
42	00101011	42	000011011010
43	00101100	43	0000 11 0 1 1011
44	00101101	44	000001010100
45	00000100	45	00000 1010101
46	00000101	46	00000 1010110
47	00001010	47	000001010111
48	00001011	48	000001100100
49	01010010	49	000001100101
49	01010010	49	000001100101
50	010100II	50	000001010010
51	01010100	51	000001010011
52	01010101	52	000000100100
53	00100100	53	000000110111

Fig 5.2. CCITT Group 3 tables

The codes greater than a string of 1792 pixels are identical for black and white pixels. A new code indicates reversal of color, that is, the pixel Color code is relative to the color of the previous pixel sequence. The following table **shows** the codes for pixel sequences larger than 1792 pixels.

Run Length (Black and White)	Make-up Code
1792	00000001000
1856	00000001100
1920	00000001101
1984	000000010010
2048	000000010011
2112	000000010100
2176	000000010101
2240	000000010110
2304	000000010111
2368	000000011100
2432	000000011101
2496	000000011110
2560	000000011111

Fig 5.3 Black and white run length

CCITT Group 3 compression utilizes Huffman coding to generate a set of make-up codes and a set of terminating codes for a given bit stream. Make-up codes are used to represent run length in multiples of 64 pixels. Terminating codes are used to represent run lengths of less than 64 pixels. As shown in the above table, run-length codes for black pixels are different from the run-length codes for white pixels. For example, the run-length code for 64 white pixels is 11011. The run length code for 64 black pixels is 0000001111. Consequently, the run length of 132 white pixels is encoded by the following two codes: Makeup code for 128 white pixels - 10010 Terminating code for 4 white pixels - 1011

The compressed bit stream for 132 white pixels is 100101011, a total of nine bits. Therefore the compression ratio is 14, the ratio between the total number of bits (132) divided by the number of bits used to code them (9).

Joint Photographic Experts Group Compression (JPEG)

ISO and CCITT working committee joint together and formed Joint Photographic Experts Group. It is focused exclusively on still image compression. Another joint committee, known as the Motion Picture Experts Group (MPEG), is concerned with full motion video standards. JPEG is a compression standard for still color images and grayscale images, otherwise known as continuous tone images.

JPEG has been released as an ISO standard in two parts

Part I specifies the modes of operation, the interchange formats, and the encoder/decoder specifies for these modes along with substantial implementation guide lines.

Part 2 describes compliance tests which determine whether the implementation of an encoder or decoder conforms to the standard specification of part I to ensure interoperability of systems compliant with JPEG standards

Requirements addressed by JPEG

o The design should address image quality.
o The compression standard should be applicable to practically any kind of continuous-tone digital source image .

- It should be scalable from completefy lossless to lossy ranges to adapt it. It should provide sequential encoding .
- It should provide for progressive encoding .
- It should also provide for hierarchical encoding .
- The compression standard should provide the option of lossless encoding so that images can be guaranteed to provide full detail at the selected resolution when decompressed.

Definitions in the JPEG Standard
The JPEG Standards have three levels of definition as follows:
* Base line system
* Extended system
* Special lossless function.

The base line system must reasonably decompress color images, maintain a high compression ratio, and handle from 4 bits/pixel to 16 bits/pixel. The extended system covers the various encoding aspects such as variable-length encoding, progressive encoding, and the hierarchical mode of encoding. The special lossless function is also known as predictive lossless coding. It ensures that at the resolution at which the image is no loss of any detail that was there in the original source image.

Overview of JPEG Components JPEG Standard components are:
 (i) Baseline Sequential Codec (ii) OCT Progressive Mode
 (iii) Predictive Lossless Encoding (iv) Hierarchical Mode.

These four components describe four four different levels of jPEG compression. The baseline sequential code defines a rich compression scheme the other three modes describe enhancements to this baseline scheme for achieving different results. Some of the terms used in JPEG methodologies are:

Discrete Cosine Transform (OCT)

OCT is closely related to Fourier transforms. Fourier transforms are used to represent a two dimensional sound signal.DCT uses a similar concept to reduce the gray-scale level or color signal amplitudes to equations that require very few points to locate the amplitude in Y-axis X-axis is for locating frequency.

DCT Coefficients

The output amplitudes of the set of 64 orthogonal basis signals are called OCT Co-efficients. **Quantization** This is a process that attempts to determine what information can be safely discarded without a significant loss in visual fidelity. It uses OCT co-efficient and provides many-to-one mapping. The quantization process is fundamentally lossy due to its many-to-one mapping.

De Quantization This process is the reverse of quantization. Note that since quantization used a many-to-one mapping, the information lost in that mapping cannot be fully recovered

Entropy Encoder / Decoder Entropy is defined as a measure of randomness, disorder, or chaos, as well as a measure of a system's ability to undergo spontaneous change. The entropy encoder compresses quantized DCT co-efficients more compactly based on their spatial characteristics. The baseline sequential. codec uses Huffman coding. Arithmetic coding is another type of entropy encoding **Huffman Coding** Huffman coding requires that one or more sets of huff man code tables be specified by the application for encoding as well as decoding. The Huffman tables may be pre-defined and used within an application as defaults, or computed specifically for a given image.

JPEG Methodology The JPEG compression scheme is lossy, and utilizes forward discrete cosine transform (or forward DCT mathematical function), a uniform quantizer, and entropy encoding. The DCT function removes data redundancy by transforming data from a spatial domain to a frequency domain; the quantizer quantizes DCT co-efficients with weighting functions to generate quantized DCT co-efficients optimized for the human eye; and the entropy encoder minimizes the entropy of quantized DCT co-efficients. The JPEG method is a symmetric algorithm. Here, decompression is the exact reverse process of compression.

Moving Picture Experts Group Compression

The MPEG standards consist of a number of different standards. The MPEG 2 suite of standards consist of standards for MPEG2 Video, MPEG - 2 Audio and MPEG - 2 systems. It is also defined at different levels, called profiles. The main profile is designed to cover the largest number of applications. It supports digital video compression in the range of2 to 15 M bits/sec. It also provides a generic solution for television worldwide, including cable, direct broadcast satellite, fibre optic media, and optical storage media (including digital VCRs).

MPEG Coding Methodology

The above said requirements can be achieved only by incremental coding of successive frames. It is known as interframe coding. If we access information randomly by frame requires coding confined to a specific frame, then it is known as intraframe coding. The MPEG standard addresses these two requirements by providing a balance between interframe coding and intraframe coding. The MPEG standard also provides for recursive and non-recursive temporal redundancy reduction.

The MPEG video compression standard provides two basic schemes: discrete-transform-based compression for the reduction of' spatial redundancy and block-based motion compensation for the reduction of temporal (motion) redundancy. During the initial stages of DCT compression, both the full motion MPEG and still image JPEG algorithms are essentially identical. First an image is converted to the YUVcolor space (a luminance/chrominance color space similar to that used for color television). The pixel data is then fed into a discrete cosine transform, which creates a scalar quantization (a two-dimensional array representing various frequency ranges represented in the image) of the pixel data.

Following quantization, a number of compression algorithms are applied, including run-length and Huffman encoding. For full motion video (MPEG I and 2), several more levels of block based motion-compensated techniques are applied to reduce temporal redundancy with both causal and noncausal coding to further reduce spatial redundancy. The MPEG algorithm for spatial reduction is lossy and is defined as a hybrid which employs motion compensation, forward discrete cosine transform (DCF), a uniform quantizer, and Huffman coding. Block-based motion compensation

is *utilized* for *reducing temporal* redundancy (i.e. to reduce the amount of data needed to represent each picture in a video sequence). Motion-compensated reduction is a key feature of MPEG.

MPEG -2

It is defined to include current television broadcasting compression and decompression needs, and attempts to include hooks for HDTV broadcasting.
The MPEG-2 Standard Supports:
1. Video Coding: * MPEG-2 profiles and levels.
2. Audio Coding:*MPEG-l audio standard fro backward compatibility.
      * Layer-2 audio definitions for MPEG-2 and stereo sound.
      * Multichannel sound.
3. Multiplexing: MPEG-2 definitions

MPEG-2, "The Grand Alliance"
It consists of following companies AT&T, MIT, Philips, Sarnoff Labs, GI Thomson, and Zenith. The MPEG-2committee and FCC formed this alliance. These companies together have defined the advanced digital television system that include the US and European HDTV systems. The outline of the advanced digital television system is as follows:
 1. Format: 1080/2: 1160 or 720/1.1160
2. Video coding: MPEG-2 main profile and high level
3. Audio coding: Dolby AC3
4. Multiplexor: As defined in MPEG-2
Modulation: 8- VSB for terrestrial and 64-QAM for cable.

Vector Quantization

Vector quantization provides a multidimensional representation of information stored in look-up tables, vector quantization is an efficient pattern-matching algorithm in which an image is decomposed into two or more vectors, each representing particular features of the image that are matched to a code book of vectors. These are coded to indicate the best fit.
In image compression, source samples such as pixels are blocked into vectors so that each vector describes a small segment or sub block of the original image. The image is then encoded by quantizing each vector separately.

DATA AND FILE FORMATS STANDARDS

There are large number of formats and standards available for multimedia system. Let us discuss about the following file formats:

- Rich-Text Format (RTF)
- Tagged Image file Format (TIFF)
- Resource Image File Format (RIFF)
- Musical Instrument Digital Interface (MIDI)
- Joint Photographic Experts Group (JPEG)
- Audio Video Interleaved (AVI) Indeo file format
- TWAIN.

Rich Text Format

This format extends the range of information from one word processor application or DTP system to another. The key format information carried across in RTF documents are given below:
Character Set: It determines the characters that supports in a particular implementation.

Font Table: This lists all fonts used. Then, they are mapped to the fonts available in receiving application for displaying text.

Color Table: It lists the colors used in the documents. The color table then mapped for display by receiving application to the nearer set of colors available to that applications.

Document Formatting: Document margins and paragraph indents are specified here.

Section Formatting: Section breaks are specified to define separation of groups of paragraphs. Paragraph Formatting: It specifies style sheds. It specifies control characters for specifying paragraph justification, tab positions, left, right and first indents relative to document margins, and the spacing between paragraphs.

General Formatting: It includes footnotes, annotations, bookmarks and pictures.

Character Formatting: It includes bold, italic, underline (continuous, dotted or word), strike through, shadow text, outline text, and hidden text.

Special Characters: It includes hyphens, spaces, backslashes, underscore and so on

TIFF File Format

TIFF is an industry-standard file format designed to represent raster image data generated by scanners, frame grabbers, and paint/ photo retouching applications.

TIFF Version 6.0 .
It offers the following formats:

(i) Grayscale, palette color, RGB full-color images and black and white.
(ii) Run-length encoding, uncompressed images and modified Huffman data compression schemes.

The additional formats are:
(i) Tiled images, compression schemes, images using CMYK, $YCbC_r$ color models.

TIFF Structure
TIFF files consist of a header. The header consists of byte ordering flag, TIFF file format version number, and a pointer to a table. The pointer points image file directory. This directory contains table of entries of various tags and their information. Fig. 5.4 shows TIFF header structure.
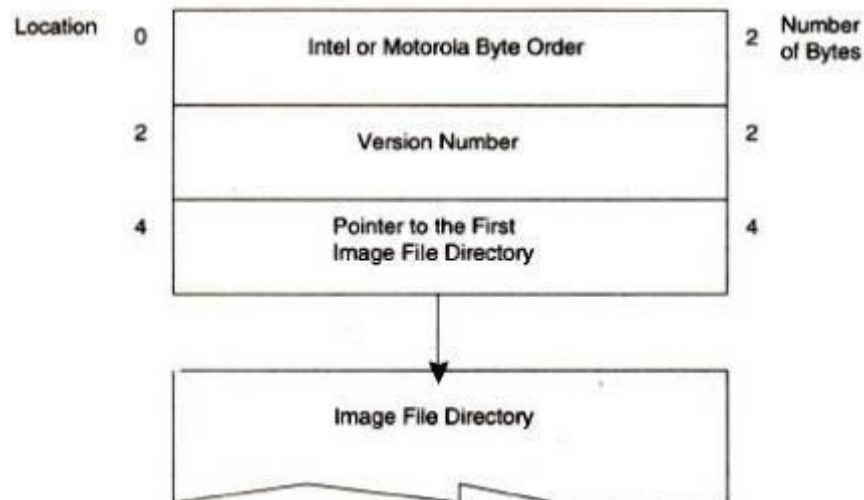
Fig 5.4 TIFF file format Header

TIFF Tags

The first two bytes of each directory entry contain a field called the Tag ID.

Tag IDs arc grouped into several categories. They are Basic, Informational, Facsimile, Document storage and Retrieval.

TIFF Classes: (Version 5.0)

It has five classes

1. Class B for binary images
2. Class F for Fax
3. Class G for gray-scale images
4. Class P for palette color images
5. Class R for RGB full-color images.

Resource Interchange File Format (RIFF)

The **RIFF** file formats consist' of blocks of data called chunks as shown in fig 5.5. They are RIFF Chunk - defines the content of the RIFF file.

List Chunk - allows to embed archival location copy right information and creating date. Subchunk - allow additional information to a primary chunk.
The first chunk in a RIFF file must be a RIFF chunk and it may contain one or more sub chunk.

The first four bytes of the RIFF chunk data field are allocated for the form type field containing four characters to identify the format of the data stored in the file: AVI, WAV, RMI, PAL and so.

File type	Form typl	File extension
Waveform Audio File	WAVE	.WAV
Audio Video Interleaved file	AVI	.AVI
MIDI File	RMID	.RMI
Device Independent Bitmap file	RDIB	.RDI
Pallette File	PAL	.PAL

Fig 5.5 RIFF file formats

The sub chunk contains a four-character ASCII string 10 to identify the type of data.

Four bytes of size contains the count of data values, and the data. The data structure of a chunk is same as all other chunks.

RIFF Chunk The first 4 characters of the RlFF chunk are reserved for the "RIFF" ASCII string. Thenext four bytes define the total data size.

The first four characters of the data field are reserved for form tyPe. The rest of the data field contains two subchunk:

(i)   fmt ~ defines the recording characteristics of the waveform.
(ii)   data ~ contains the data for the waveform.

LIST Chunk
*RlFF*chunk may contains one or more list chunks.

List chunks allow embedding additional file information such as archival location, copyright information, creating date, description of the content of the file.

_RlFF_MIDI FILE FORMAT

*RlFF*MIDI contains a*RlFF*chunk with the form type "RMID"and a subchunk called "data" for MIDIdata.

The 4 bytes are for ID of the *RlFF* chunk. 4 bytes are for size 4 bytes are for form type 4 bytes are for ID of the subchunk data and 4 bytes are for the size of MIDI data.

MIDI File Format

The MIDI file format follows music recording metaphor to provide the means of storing separate tracks of music for each instrument so that they can be read and syn~hronized when they are played.

The MIDI file format also contains chunks (i.e., blocks) of data. There are two types of chunks: (i) header chunks (ii) track chunks.

Header Chunk

It is made up of 14 bytes .

The first four-character string is the identifier string, "MThd" .

The second four bytes contain the data size for the header chunk. It is set to a fixed value of six bytes .

The last six bytes contain data for header chunk.

Track chunk

The Track chunk is organized as follows:

- **.:.** The first 4-character string is the identifier.
- **.:.** The second 4 bytes contain track length.

MIDI Communication Protocol

This protocol uses 2 or more bytes messages.

The number of bytes depends on the types of message. There are two types of messages:
    (i) Channel messages and (ii) System messages.

Channel Messages

A channel message can have up to three bytes in a message. The first byte is called a status byte, and other two bytes are called data bytes. The channel number, which addresses one of the 16 channels, is encoded by the lower nibble of the status byte. Each MIDI voice has a channel number; and messages are sent to the channel whose channel number matches the channel number encoded in the lower nibble of the status byte. There are two types of channel messages: voice messages and the mode messages.

Voice messages

Voice messages are used to control the voice of the instrument (or device); that is, switch the notes on or off and sent key pressure messages indicating that the key is depressed, and send control messages to control effects like vibrato, sustain, and tremolo. Pitch wheel messages are used to change the pitch of all notes .

Mode messages

Mode messages are used for assigning voice relationships for up to 16 channels; that *is,* to set the device to MOWO mode or POLY mode. Omny Mode on enables the device to receive voice messages on all channels.

System Messages

System messages apply to the complete system rather than specific channels and do not contain any channel numbers. There are three types of system messages: common messages, real-time messages, and exclusive messages. In the following, we will see how these messages are used.

Common Messages These messages are common to the complete system. These messages provide forfunctions such as select a song, setting the song position pointer with number of beats, and sending a tune request to an analog synthesizer.

System Real Time Messages

These messages are used for setting the system's real-time parameters. These parameters include the timing clock, starting and stopping the sequencer, ressuming the sequencer from a stopped position, and resetting the system.

System Exclusive messages

These messages contain manufacturer-specific data such as identification, serial number, model number, and other information. Here, a standard file format is generated which can be moved across platforms and applications.

JPEG Motion Image:

JPEG Motion image will be embedded in A VI RIFF file format. There are two standards available:

(i) MPEG ~ In this, patent and copyright issues are there.
(ii) MPEG 2 ~ It provide better resolution and picture quality.

TWAIN

To address the problem of custom interfaces, the TWAIN working group was formed to define an open industry standard interface for input devices. They designed a standard interface called a generic TWAIN interface. It allows applications to interface scanners, digital still cameras, video cameras.
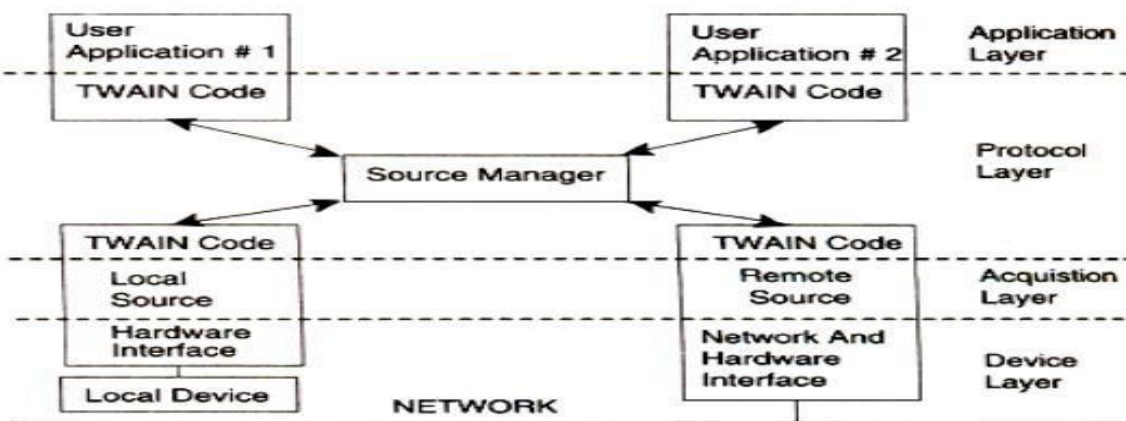
TWAIN ARCHITECHTURE:

Fig 5.6 TWAIN Architecture

o   The Twain architecture (as shown in Fig. 5.6) defines a set of application programming interfaces (APls) and a protocol to acquire data from input devices.

o   It is a layered architecture.

o   It has application layer, the protocol layer, the acquisition layer and device layer.

o   Application Layer: This layer sets up a logical connection with a device. The application layer interfaces with protocol layer.

o   Protocol Layer: This layer is responsible for communications between the application and acquisition layers.
o   The main part of the protocol layer is the source Manager.
o   Source manager manages all sessions between an application and the sources, and monitors data acquisition transactions. The protocol layer is a complex layer.

It provides the important aspects of device and application interfacing functions. **The Acquisition Layer:** It contains the virtual device driver.

It interacts directly with the device driver. This layer is also known as source. It performs the following functions:

1. Control of the device.
2. Acquisition of data from the device.
3. Transfer of data in agreed format.
4. Provision of user interface to control the device.

The Device Layer: The device layer receives software commands and controls the device hardware. **NEW WAVE RIFF File Format:** This format contains two subchunks:
(i) Fmt **(ii)** Data.
It may contain optional subchunks:
(i)   Fact **(ii)** Cue points **(iii)**Play list **(iv)** Associated datalist.

Fact Chunk: It stores file-dependent information about the contents of the WAVE file. **Cue Points Chunk:** It identifies a series of positions in the waveform data stream. **Playlist Chunk:** It specifies aplay order for series of cue points. **Associated Data Chunk:** It provides the ability to attach information, such as labels, to sections of the waveform data stream. **Inst Chunk:** The file format stores sampled sound synthesizer's samples.

Digital Voice and Audio

Digital Audio

Sound is made up of continuous analog sine waves that tend to repeat depending on the music or voice. The analog waveforms are converted into digital format by analog-to-digital converter (ADC) using sampling process as shown in fig. 5.7.
Sampling process
Sampling is a process where the analog signal is sampled over time at regular intervals to obtain the amplitude of the analog signal at the sampling time.
Sampling rate
The regular interval at which the sampling occurs is called the sampling rate.
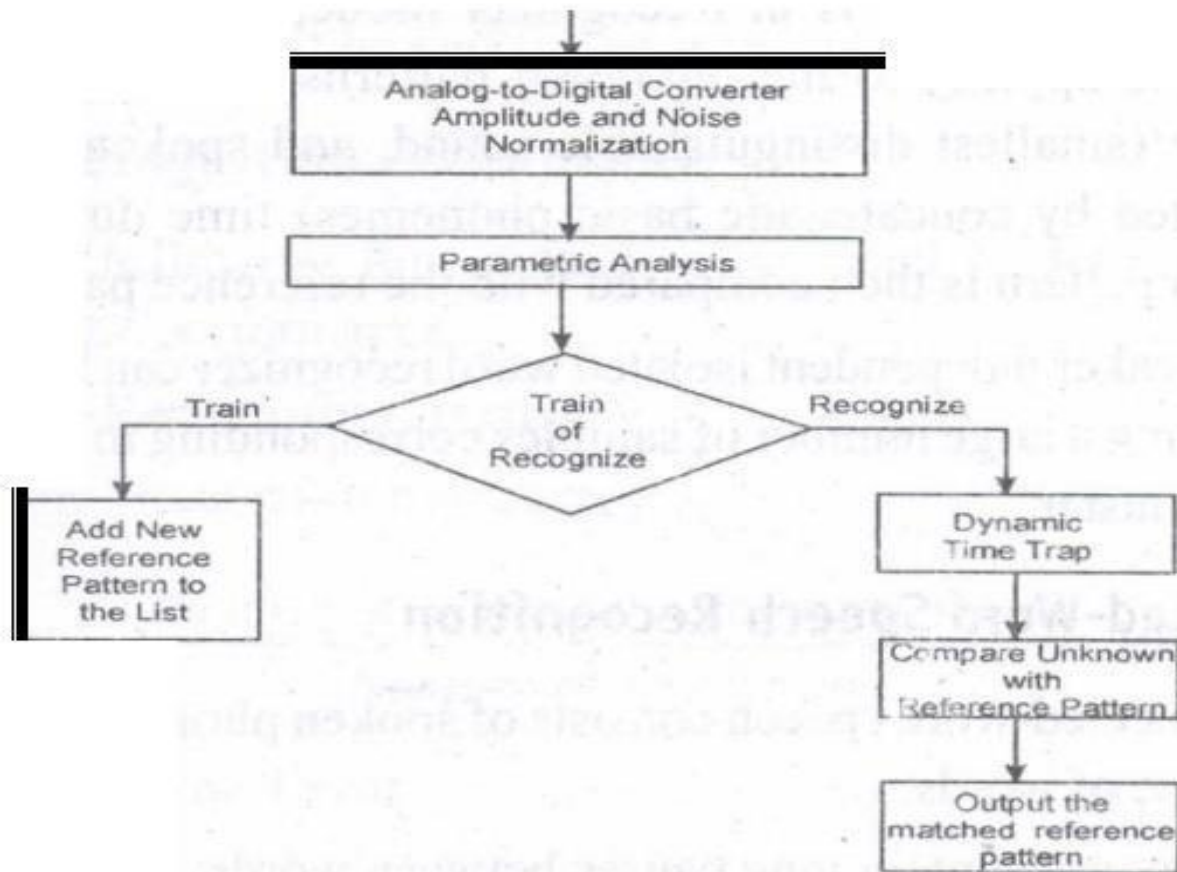
Fig 5.7 Digital Audio

Digital Voice

Speech is analog in nature and is converted to digital form by an analog-to-digital converter (ADC). An ADC takes an input signal from a microphone and converts the amplitude of the sampled analog signal to an 8, 16 or 32 bit digital value.

The four important factors governing the ADC process are sampling rate, resolution, linearity and conversion speed.

- **Sampling Rate:** The rate at which the ADC takes a sample of an analog signal.
- **Resolution:** The number of bits utilized for conversion determines the resolution of ADC.
- **Linearity:** Linearity implies that the sampling is linear at all frequencies and that the amplitude tmlyrepresents the signal.
- **Conversion Speed:** It is a speed of ADC to convert the analog signal into Digital signals. It must be fastenough.

VOICE Recognition System

Voice Recognition Systems can be classified into three types.

1. Isolated-word Speech Recognition.
2. Connected-word Speech Recognition.
3. Continuous Speech Recognition.

1. Isolated-word Speech Recognition.

It provides recognition of a single word at a time. The user must separate every word by a pause. The pause marks the end of one word and the beginning of the next word.

Stage 1: Normalization

The recognizer's first task is to carry out amplitude and noise normalization to minimize the variation in speech due to ambient noise, the speaker's voice, the speaker's distance from and position relative to the microphone, and the speaker's breath noise.

Stage2: Parametric Analysis

It is a preprocessing stage that extracts relevent time-varying sequences of speech parameters. This stage serves two purposes: (i) It extracts time-varying speech parameters. (ii) It reduces the amount of data of extracting the relevant speech parameters.

Training mode: In training mode of the recognizer, the new frames are added to the reference list. **Recognizer mode:** If the recognizer is in Recognizer mode, then dynamic time warping is applied to theunknown patterns to average out the phoneme (smallest distinguishable sound, and spoken words are constructed by concatenatic basic phonemes) time duration. The unknown pattern is then compared with the reference patterns.

A speaker independent isolated word recognizer can be achieved by groupi.ng a large number of samples corresponding to a word into a single cluster.

2. Connected-Word Speech Recognition:
Connected-word speech consists of spoken phrase consistingof a sequence of words. It may not contain long pauses between words.

The method using Word Spotting technique
It recognizes words in a connected-word phrase. In this technique, Recognition is carried out by compensating for rate of speech variations by the process called dynamic time warping (this process is used to expand or compress the time duration of the word), and sliding the adjusted connected-word phrase representation in time past a stored word template for a likely match.

Continuous Speech Recognition
This sytem can be divided into three sections:
  (i)   A section consisting of digitization, amplitude normalization, time normalization and parametric representation.
  (ii)   Second section consisting of segmentation and labeling of the speech segment into a symbolic string based on a knowledge based or rule-based systems.
  (iii)   The final section is to match speech segments to recognize word sequences.

Voice Recognition performance
It is categorized into two measures: Voice recognition performance and system performance. The following four measures are used to determine voice recognition performance.

1. Voice Recognition Accuracy

$$\text{Voice Recognition Accuracy} = \frac{\text{Number of correctly recognized words}}{\text{Number of test words}} \times 100$$

2. Substitution Error

$$\text{Substitution error} = \frac{\text{Number of substituted words}}{\text{Number of test words}} \times 100$$

3. No Response Error

$$\frac{\text{Number of no responses}}{\text{Number of test words}} \times 100$$

4. Insertion Error

$$\text{Insertion error} = \frac{\text{Number of insertion error}}{\text{Number of test words}} \times 100$$

Voice Recognition Applications

Voice mail integration: The voice-mail message can be integrated with e-mail messages to create an integrated message.

DataBase Input and Query Applications

A number of applications are developed around the voice recognition and voice synthesis function. The following lists a few applications which use Voice recognition.

- Application such as order entry and tracking

It is a server function; It is centralized; Remote users can dial into the system to enter an order or to track the order by making a Voice query.

- Voice-activated rolodex or address book

When a user speaks the name of the person, the rolodex application searches the name and address and voice-synthesizes the name, address, telephone numbers and fax numbers of a selected person. In medical emergency, ambulance technicians can dial in and register patients by speaking into the hospital's centralized system.

Police can make a voice query through central data base to take follow-up action ifhe catch any suspect.

Language-teaching systems are an obvious use for this technology. The system can ask the student to spell or speak a word. When the student speaks or spells the word, the systems performs voice recognition and measures the student's ability to spell. Based on the student's ability, the system can adjust the level of the course. This creates a self-adjustable learning system to follow the individual's pace.

Foreign language learning is another good application where"' an individual student can input words and sentences in the system. The system can then correct for pronunciation or grammar.

Musical Instrument Digital Interface (MIDI)

MIDI interface is developed by Daver Smith of sequential circuits, inc in 1982. It is an universal synthesizer interface

MIDI Specification 1.0
MIDI is a system specification consisting of both hardware and software components which define inter-connectivity and a communication protocol for electronic synthesizers, sequences, rythm machines, personal computers, and other electronic musical instruments. The inter-connectivity defines the standard cabling scheme, connector type and input/output circuitry which enable these different MIDI instruments to be interconnected. The communication protocol defines standard multibyte messages that allow controlling the instrument's voice and messages including to send response, to send status and to send exclusive.MIDI Input and output circuitry is shown in the Fig. 5.8
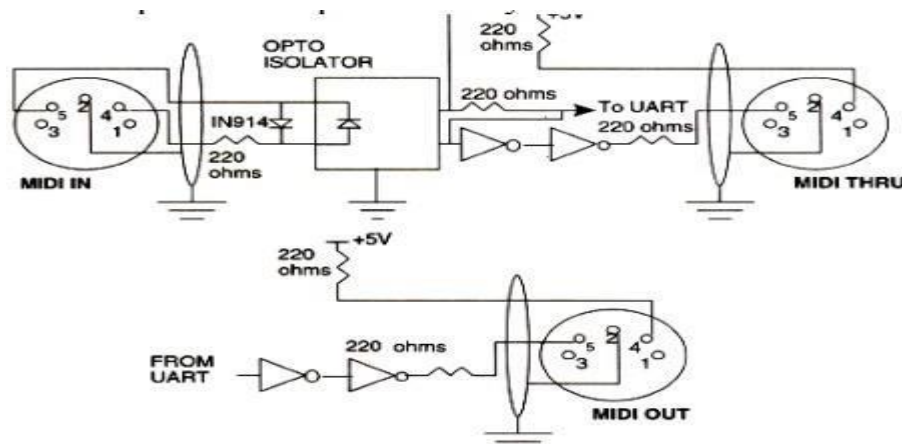
Fig 5.8 MIDI Input and output circuitry

MIDI Hardware Specification
The MIDI. hardware specification require five pin panel mount requires five pin panel mount receptacle DIN connectors for MIDI IN, MIDI OUT and MIDI THRU signals. The MIDI IN connector is for input signals The MIDI OUT is for output signals MIDI THRU connector is for daisy-chaining multiple MIDI instruments.

MIDI Interconnections
The MIDI IN port of an instrument receives MIDI ncssages to play the instrument's internal synthesizer. The MIDI OUT port sends MIDI messages to play these messages to an external synthesizer. The MIDI THRU port outputs MIDI messages received by the MIDI IN port for daisy-chaining external synthesizers.

Communication Protocol
The MIDI communication protocol uses multibyte messages; There are two types of messages:
   (i) Channel messages
   (ii)System messages

The *channel messages* have three bytes. The first byte is called a status byte, and the other two bytes are called data bytes. The two types of channel messages: (**i**) **Voice messages (ii) Mode messages**.

System messages: The three types of system messages.

Common message: These messages are common to the complete system. These messages provide for functions.

System real.time messages: These messages are used for setting the system's real-time parameters. These parameters include the timing clock, starting and stopping the sequencer, resuming the sequencer from a stopped position and restarting the system.

System exclusive message: These messages contain manufacturer specific data such as identification, serial number, model number and other information.

SOUND BOARD ARCHITECTURE
A sound card consist of the following components: MIDI Input/Output Circuitry, MIDI Synthesizer Chip, input mixture circuitry to mix CD audio input with LINE IN input and microphone input, analog-to-digital converter with a pulse code modulation circuit to convert analog signals to digital to create WAVfiles, a decompression and compression chip to compress and decompress audio files, a speech synthesizer to synthesize speech output, a speech recognition circuitry to recognize speech input and output circuitry to output stereo audio OUT or LINEOUT.

AUDIO MIXER
The audio mixer component of the sound card typically has external inputs for stereo CD audio, stereo LINE IN, and stereo microphone MICIN. These are analog inputs, and they go through analog-to-digital conversion in conjunction with PCM or ADPCM to generate digitized samples.

SOUND BOARD ARCHITECTURE:

MIDI sound board Architecture is shown in Fig 5.9.

Analog-to-Digital Converters: The ADC gets its input from the audio mixer and converts theamplitude of a sampled analog signal to either an 8-bit or 16-bit digital value.

Digital-to-Analog Converter (DAC): A DAC converts digital input in the 'foml of W AVE files, MIDIoutput and CD audio to analog output signals.

Sound Compression and Decompression: Most sound boards include a codec for sound compression and decompression. ADPCM for windows provides algorithms for sound compression.

CD-ROM Interface: The CD-ROM interface allows connecting u CD ROM drive.to the sound board.
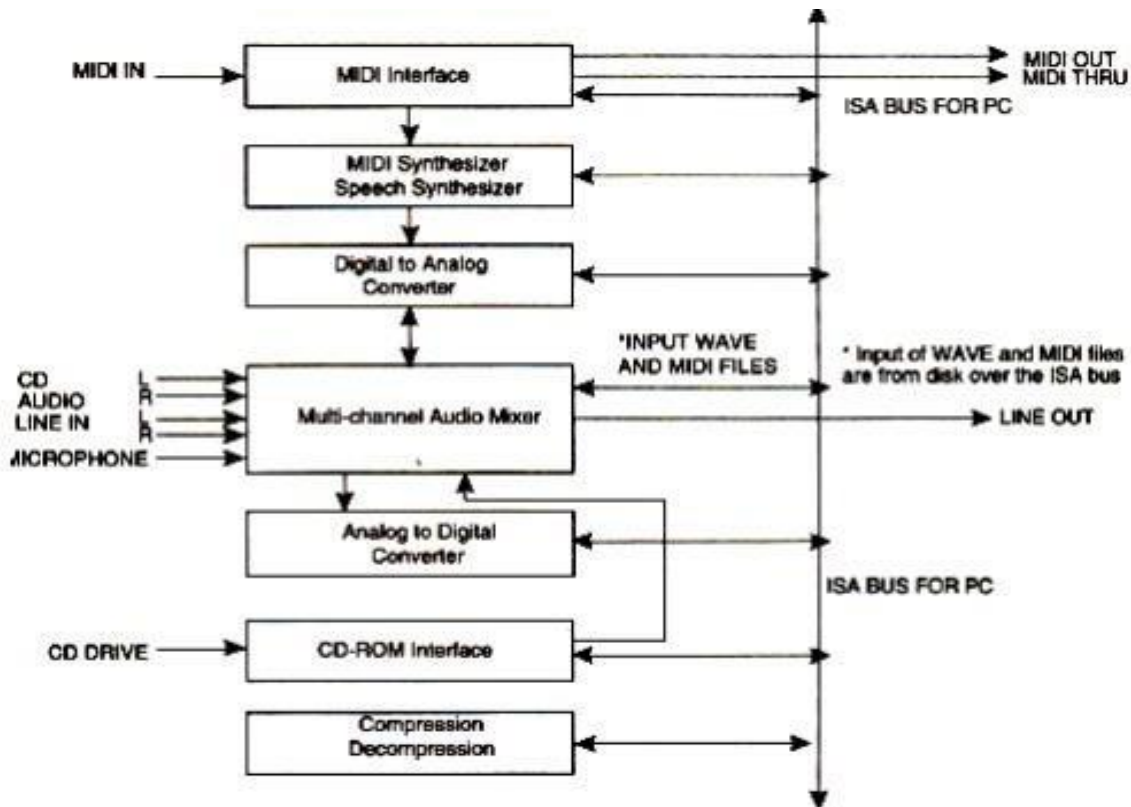
Fig 5.9 MIDI sound board Architecture

Video Images and Animation

Video Frame Grabber Architecture

A video frame grabber is used to capture, manipulate and enhance video images.
A video frame grabber card consists of video channel multiplexer, Video ADC, Input look-up table with arithmetic logic unit, image frame buffer, compression-decompression circuitry, output color look-up table, video DAC and synchronizing circuitry.

Video Channel Multiplexer:
A video channel multiplexer has multiple inputs for different video inputs. The video channel multiplexer allows the video channel to be selected under program control and switches to the control circuitry appropriate for the selected channel in a TV with multi – system inputs.

Analog to Digital Converter: The ADC takes inputs from video multiplexer and converts the amplitude of a sampled analog signal to either an 8-bit digital value for monochrome or a 24 bit digital value for color.

Input lookup table: The input lookup table along with the arithmetic logic unit (ALU) allowsperforming image processing functions on a pixel basis and an image frame basis. The pixel image-processing functions ate histogram stretching or histogram shrinking for image brightness and contrast, and histogram sliding to brighten or darken the image. The frame-basis image-processing functions perform logical and arithmetic operations.

Image Frame Buffer Memory: The image frame buffer is organized as a l024 x 1024 x 24 storagebuffer to store image for image processing and display.

Video Compression-Decompression: The video compression-decompression processor is used tocompress and decompress still image data and video data.

Frame Buffer Output Lookup Table: The frame buffer data represents the pixel data and is used to index into the output lookup table. The output lookuptable generates either an 8 bit pixel value for monochrome or a 24 bit pixel value for color.

SVGA Interface: This is an optional interface for the frame grabber. The frame grabber can bedesigned to include an SVGA frame buffer with its own output lookup table and digital-to-analog converter.

Analog Output Mixer: The output from the SVGA DAC and the output from image frame buffer DACis mixed to generate overlay output signals. The primary components involved include the display image frame buffer and the display SVGA buffer. The display SVGA frame buffer is overlaid on the image frame buffer or live video, This allows SVGA to display live video.

Video and Still Image Processing
Video image processing is defined as the process of manipulating a bit map image so that the image can be enhanced, restored, distorted, or analyzed.

The terms using in video and still image processing are,

Pixel point to point processing: In pixel point-to-point processing, operations are carried out onindividual pixels one at a time.

Histogram Sliding: It is used to change the overall visible effect of brightening or darkening of theimage. Histogram sliding is implemented by modifying the input look-up table values and using the input lookup table in conjunction with arithmetic logic unit.

Histogram Stretching and Shrinking: It is to increase or decrease the contrast.
In histogram shrinking, the brighter pixels are made less bright and the darker pixels are made less dark. **Pixel Threshold:** Setting pixel threshold levels set a limit on the bright or dark areas of a picture. Pixelthreshold setting is also achieved through the input lookup table.

Inter- frame image processing
Inter- frame image processing is the same as point-to-point image processing, except that the image processor operates on two images at the same time. The equation of the image operations is as follows:
Pixel output (x, y) = (Image l(x, y)
Operator (Image 2(x, y)
Image Averaging: Image averaging minimizes or cancels the effects of random noise.
Image Subtraction: Image subtraction is used to determine the change from one frame to the next for image comparisons for key frame detection or motion detection.

Logical Image Operation: Logical image processing operations are useful for comparing image framesand masking a block in an image frame.

Spatial Filter Processing The rate of change of shades of gray or colors is called spatial frequency.The process of generating images with either low-spatial frequency-components or high frequency components is called spatial filter processing.

Low Pass Filter: A low pass filter causes blurring of the image and appears to cause a reduction innoise.

High Pass Filter: The high-pass filter causes edges to be emphasized. The high-pass filter attenuateslow-spatial frequency components, thereby enhancing edges and sharpening the image.

Laplacian Filter: This filter sharply attenuates low-spatial-frequency components without affecting andhigh-spatial frequency components, thereby enhancing edges sharply.

Frame Processing: Frame processing operations are most commonly for geometric operations, imagetransformation, and image data compression and decompression Frame processing operations are very compute intensive many multiply and add operations, similar to spatial filter convolution operations.

Image scaling: Image scaling allows enlarging or shrinking the whole or part of an image.
Image rotation: Image rotation allows the image to be rotated about a center point. The operation canbe used to rotate the image orthogonally to reorient the image if it was scanned incorrectly. The operation can also be used for animation.

The rotation formula is:
 pixel output-(x, $y)$ = pixel input *(x,* $\cos Q + y \sin Q$, $- x \sin Q + Y \cos Q$) where, Q is the orientation angle and *x,* y are the spatial co-ordinates of the original pixel.

Image translation: Image translation allows the image to be moved up and down or side to side. Again,this function can be used for animation.
The translation formula is:
Pixel output *(x, y)* =Pixel Input *(x + Tx, y + Ty)* where
Tx and *Ty* are the horizontal and vertical coordinates. *x,* y are the spatial coordinates of the original pixel.
Image transformation: An image contains varying degrees of brightness or colors defined by thespatial frequency. The image can be transformed from spatial domain tothe frequency domain by using frequency transform.

Image Animation Techniques

Animation: Animation is an illusion of movement created by sequentially playing still image frames atthe rate of 15-20 frames per second.

- The illusion of motion created by the consecutive display of images of static elements.
- In multimedia, animation is used to further enhance / enriched the experience of the user to further understand the information conveyed to them.

  When you create an animation, organize its execution into a series of logical steps. First,

gather up in your mind all the activities you wish to provide in the animation; if it is complicated, you may wish to create a written script with a list of activities and required objects.

Choose the animation tool best suited for the job. Then build and tweak your Sequences; experiment with lighting effects. Allow plenty of time for this phase when you are experimenting and testing. Finally, post-process your animation, doing any special rendering and adding sound effects

Types of animation

1. Cel Animation
2. Computer animation
3. Kinematics
4. Morphing

1. Cel animation

The term cel derives from the clear celluloid sheets that were used for drawing each frame, which have been replaced today by acetate or plastic. Cel animation artwork begins with keyframes (the first and last frame of an action). For example, when an animated figure of a man walks across the screen, he balances the weight of his entire body on one foot and then the other in a series of falls and recoveries, with the opposite foot and leg catching up to support the body.

2. Computer Animation

Computer animation programs typically employ the same logic and procedural concepts as cel animation, using layer, keyframe, and tweening techniques, and even borrowing from the vocabulary of classic animators. The primary difference between the animation software program is in how much must be drawn by the animator and how much is automatically generated by the software

- In 2D animation the animator creates an object and describes a path for the object to follow. The software takes over, actually creating the animation on the fly as the program is being viewed by your user.
- In 3D animation the animator puts his effort in creating the models of individual and designing the characteristic of their shapes and surfaces.
- Paint is most often filled or drawn with tools using features such as gradients and anti-aliasing.

3. Kinematics

- It is the study of the movement and motion of structures that have joints, such as a walking man.
- Inverse Kinematics is in high-end 3D programs, it is the process by which you link objects such as hands to arms and define their relationships and limits.
- Once those relationships are set you can drag these parts around and let the computer calculate the result.

4. Morphing

Morphing is popular effect in which one image transforms into another. Morphing application and other modeling tools that offer this effect can perform transition not only between still images but often between moving images as well.

Animation File Formats

Some file formats are designed specifically to contain animations and the can be ported among application and platforms with the proper translators.

- Director *.dir, *.dcr
- AnimationPro *.fli, *.flc
- 3D Studio Max *.max
- SuperCard and Director *.pics
- CompuServe *.gif
- Flash *.fla, *.swf

Following is the list of few Software used for computerized animation:

- 3D Studio Max
- Flash
- AnimationPro

Toggling between image frames: We can create simple animation by changing images at display time.The simplest way is to toggle between two different images. This approach is good to indicate a "Yes" or "No" type situation.

Rotating through several image frames: The animation contains several frames displayed in a loop.Since the animation consists of individual frames, the playback can be paused and resumed at any time.

Photoshop

Introduction

Photoshop is a graphics based program created with images known as raster graphics. Other graphic applications, i.e. Illustrator, Corel Draw and Freehand, create vector graphics. Vector graphics are composed of solid lines, curves and other geometric shapes that are defined by a set of mathematical instructions. Vector images work best for type and other shapes that require clear crisp boundaries. Raster images work best with photographs. Raster graphics are comprised of a raster (a grid) of small squares called pixels. Objects in Photoshop are groups of many pixels – each of which can be a different color. Raster images require more memory and storage than vector images. Photoshop is a memory-hungry program.

Photoshop is unlike other common software interfaces which emulate virtual typewriters or graphing paper. Photoshop creates an artist's virtual studio/darkroom. When you open the program you see a toolbox on the left with tools you will use to manipulate your images, and on the right, a white square which is your "canvas" or work area. The gray area surrounding the canvas is not part of your image, but only defines its edges.

Workplace (Actually Workspace)

Generally, there are four components in your workspace that you will use while creating or modifying graphics. These components are as follows:

- The Menu Bar
- The Drawing Canvas
- The Toolbox

- Palettes (There are five palettes by default)

The figure 5.10 below shows each of these components similar to how they will appear on your screen
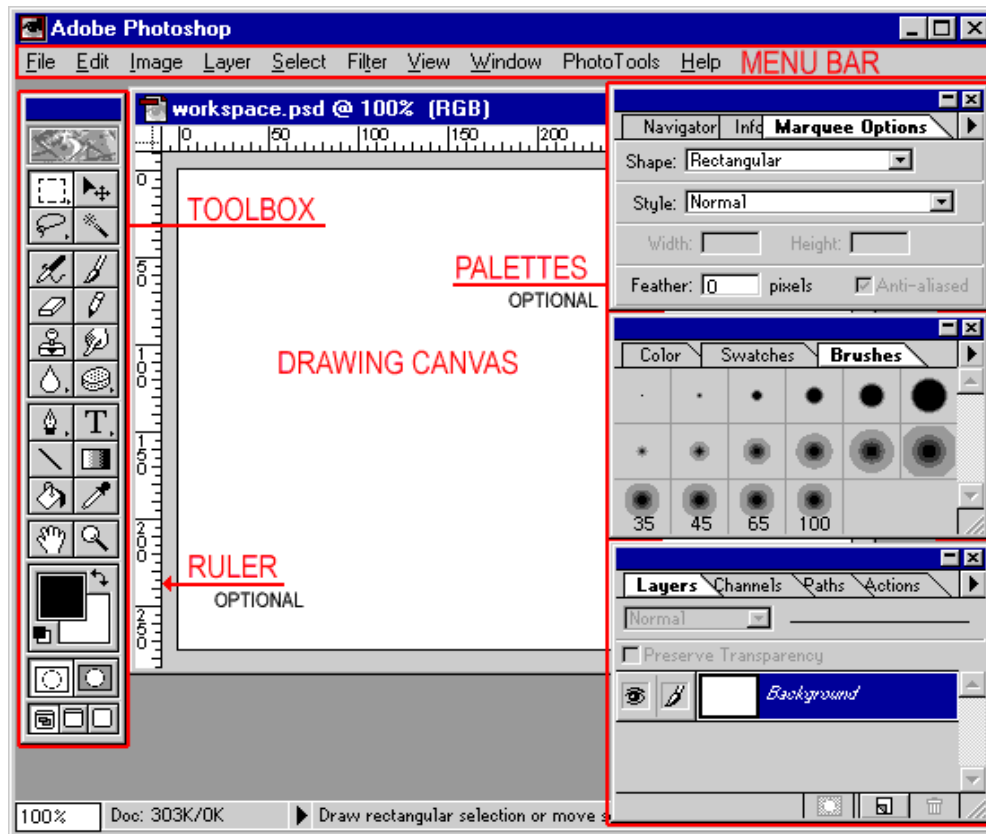

Fig 5.10 Photoshop Workspace

Navigation Window

It is a roadmap to your image document.

The Navigator panel is one panel that you probably want readily accessible. It's most useful when it's visible at all times. Undock the Navigator panel by pulling the tab of the panel to the left. Position the Navigator panel to one side of your image so it's ready for instant use.

View the thumbnail. The entire Navigator window shows the full document image, with an outline called a *View box* showing the amount of image visible in the document window at the current zoom level.

Change the view. Click anywhere in the thumbnail *outside* the View box to center the box at that position. The comparable view in your main document window changes to match.

Move the view. Click anywhere in the thumbnail inside the View box and then drag to move the box to a new position. The main document window changes to match the new view.

Zoom in or out. Click the Zoom In button (which has an icon of two large pyramids) or Zoom Out button (which has an icon of two smaller pyramids) to zoom in or out. Or drag the Zoom slider that resides between the two icons.

TOOLS

Figure 5.11 shows the Tools present in photoshop.

Marquee Tool : ⬚ The images in Photoshop are stored pixel by pixel, with a code indicating the color of each. The image is just a big mosaic of dots. Therefore, before you can do anything in Photoshop, you first need to indicate which pixels you want to change. The selection tool is one way of doing this. Click on this tool to select it, then click and drag on your image to make a dotted selection box. Hold shift while you drag if you want a perfect square or circle. Any pixels within the box will be affected when you make your next move. If you click and hold on this tool with your mouse button down, you will see that there is also an oval selection shape, and a crop tool ⬚.

Crop Tool: ⬚. To crop your image, draw a box with the crop tool. Adjust the selection with the selection points, and then hit return to crop.

Lasso Tool : ⬚ The lasso tool lets you select freeform shapes, rather than just rectangles and ovals.

Magic Wand: ⬚ Yet another way to select pixels is with the magic wand. When you click on an area of the image with this tool, all pixels that are the same color as the pixel you clicked will be selected. Double click on the tool to set the level of tolerance you would like (i.e. how similar in color the pixels must be to your original pixel color. A higher tolerance means a broader color range).

(M) Marquee tool — ⬚ ⬚ — Move Tool (V)
(L) Lasso tool — ⬚ ⬚ — Magic Wand (W)
(C) Crop tool — ⬚ ⬚ — Knife (K)
Healing Brush — ⬚ ⬚ — Pencil/paintbrush (B)
(S) Clone Stamp — ⬚ ⬚ — History Brush (Y)
(E) Eraser — ⬚ ⬚ — Gradient tool/ Paintbucket (G)
(R) Blur tool — ⬚ ⬚ — dodge and burn tool (O)
(A) Path compoment selection — ⬚ ⬚ — Type Tool (T)
(P) Pen tool for drawing paths — ⬚ ⬚ — Line tool/ Rectangle (U)
(N) Notes — ⬚ ⬚ — Eyedropper (I)
(H) Hand Tool — ⬚ ⬚ — Zoom (Z)
Foreground color — ⬚ — Switch colors (X)
— Background color
— Quickmask mode (Q)
— Full screen mode (F)
— Jump to ImageReady

Fig 5.11 Tools

The Move Tool: ⬚ This is a very important tool, because up until now all you have been able to do is select pixels, and not actually move them. The move tool not only allows you to move

29

areas you have selected, but also to move entire layers without first making a selection. If you hold the option (or alt) key while clicking and dragging with the move tool, you can copy the selection.

Airbrush **Paintbrush** and **Pencil** tools can be used to draw with the foreground color on whichever layer is selected. To change the foreground color, double-click on it in the toolbox. You will then see a palette of colors from which to choose. Select one and click OK. To change the brush size, go to Window > Show Brushes.

Eraser Tool: Erases anything on the selected layer. You can change the eraser size by going to Window > Show Brushes.

Line Tool: Can be used to draw straight lines. Click on the tool to select it, then click with the tool on the canvas area and drag to draw a line. When you release the mouse button, the line will end. You can change the thickness of the line or add arrowheads to it by double clicking on the tool to see this dialog box shown in Fig 5.12:
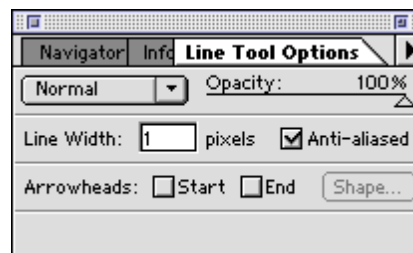
Fig 5.12 Line Tool

Text tool: Click on this tool to select it, then click in the Canvas area. You will be given a dialog box in which to type your text, and choose its attributes. Each new block of text goes on its own layer, so you can move it around with the Move Tool. Once you have placed the text, however, it is no longer editable. To correct mistakes, you must delete the old version (by deleting its layer) and replace it.

Eyedropper: Click with this tool on any color in the canvas to make that color the foreground color. (You can then paint or type with it).

Magnifier: Click with this tool on a part of your image you want to see closer, or drag with it to define the area you want to expand to the size of the window. Hold down the Option or Alt key to make it a "reducer" instead and zoom back out.

Grabber: Click with this and drag to move the entire page for better viewing.

 Options Bar

Fig 5.13 Options Bar

The Options bar(as shown in Fig 5.13) appears at the top of the screen and is context sensitive, changing as you change tools. The tool in use is shown in the left corner, and options relating to the tool appear to the right of that.

 Import to Photoshop

        Flash can import still images in many formats, but you usually use the native Photoshop PSD format when importing still images from Photoshop into Flash.

When importing a PSD file, Flash can preserve many of the attributes that were applied in Photoshop, and provides options for maintaining the visual fidelity of the image and further modifying the image. When you import a PSD file into Flash, you can choose whether to represent each Photoshop layer as Flash layers or individual key frames.

Save and Export

There are two methods to saving a photo in Photoshop, and each has a specific purpose. One way is to use the typical **Save As...** dialogue, the other is a special feature in Photoshop called **Save for Web & Devices...** which is used to save your photos in preparation for publication to the Web.
1) Save as: Use this method when saving your photo for archiving or if you plan to work on it later. We recommend saving the file type as a **Photoshop** or **.PSD** file, which will also save extra Photoshop-specific information about your photo.
**2) Save for Web:** Use this when you are ready to export your photo for publication to the Web. While it's possible to save a photo with the regular "save as..." option and still publish it to the Web, the Photoshop built-in "Save for Web" feature specifically prepares your photo for the Web and has added features that allow you to see how it will appear once it's on a Web site. This ensures your photos will show up properly on the Web.

Operations On Images – Resize, Crop, Rotate

Crop images

Cropping is the process of removing portions of an image to create focus or strengthen the composition. You can crop an image using the Crop tool 🔲 and the Crop command. You can also trim pixels using the Crop And Straighten and the Trim commands.
Crop an image using the Crop tool | CS5
 1. Select the Crop tool 🔲.
 2. (Optional) Set resample options in the options bar.
    - To crop the image without resampling (default), make sure that the Resolution text box in the options bar is empty. You can click the Clear button to quickly clear all text boxes.
    - To resample the image during cropping, enter values for height, width, and resolution in the options bar. To switch the height and width dimensions, click the Swaps Height And Width icon ⇄.
    - To resample an image based on the dimensions and resolution of another image, open the other image, select the Crop tool, and click Front Image in the options bar. Then make the image you want to crop active.
 3. Drag over the part of the image you want to keep to create a marquee.
 4. Select Hide to preserve the cropped area in the image file. You can make the hidden area visible by moving the image with the Move tool ⊹. Select Delete to discard the cropped area.
 5. To complete the crop, press Enter (Windows) or Return (Mac OS), click the Commit button ✓ in the options bar, or double-click inside the cropping marquee.
 6. To cancel the cropping operation, press Esc or click the Cancel button ⊘ in the options bar.

Crop an image using the Crop command
1. Use a selection tool to select the part of the image you want to keep.
2. Choose Image > Crop.

Resize & Rotate

Creating an aesthetically pleasing design in Adobe Photoshop is dependent on the ability to properly composite and arrange image layers. Therefore, it's vital to know how to resize and rotate layers using Photoshop's "Free Transform" command. Free Transform combines several "Transform" options into one tool, letting you quickly scale and rotate layers. This command has been a continuous and preferred feature in Photoshop

STEPS:
1. Open a file containing layers.
2. Click on the layer you want to resize and rotate.
3. Go to "Edit," "Free Transform" or press "Ctrl" and "T."
4. Click and drag the corner handles to resize the layer. Hold down the "Shift" key while you do this to constrain the image's proportions. You can also scale the layer using the top, bottom or side handles.
5. Hold the mouse pointer just outside one of the corner handles until it becomes a double-sided arrow. Click and drag to rotate the layer. "Shift" will limit the rotation to every 15 degrees.
6. Double-click within the "Free Transform" box to apply the changes to the layer.

FLASH

Introduction

Flash can be used for creating games, making presentations, animations, visualizations, webpage components, and many other interactive applications. Flash is a multimedia software that is used to design user interfaces and applications. Flash packs a lot of functionality into one easy-to-use program.
In Flash you can:
- create animated movies from scratch
- import graphic content created in other programs and use Flash to animate it.
- how the site will function and appear regardless of the browser used.

Flash Elements

To design and deliver Flash documents, you work within the Adobe Flash authoring environment. The Quick Start page provides easy access to your most frequently used actions. The stage contains the visible elements, such as text, components, and images of the document. The Timeline represents different phases, or frames, of an animation. The Tools panel provides the tools used to create and manipulate objects on the stage. The Edit bar tells you what you are currently working on and gives you the ability to change the magnification level of the stage. The panels provide access to a wide variety of authoring tools. The below Fig 5.14 shows the workspace of Flash environment.
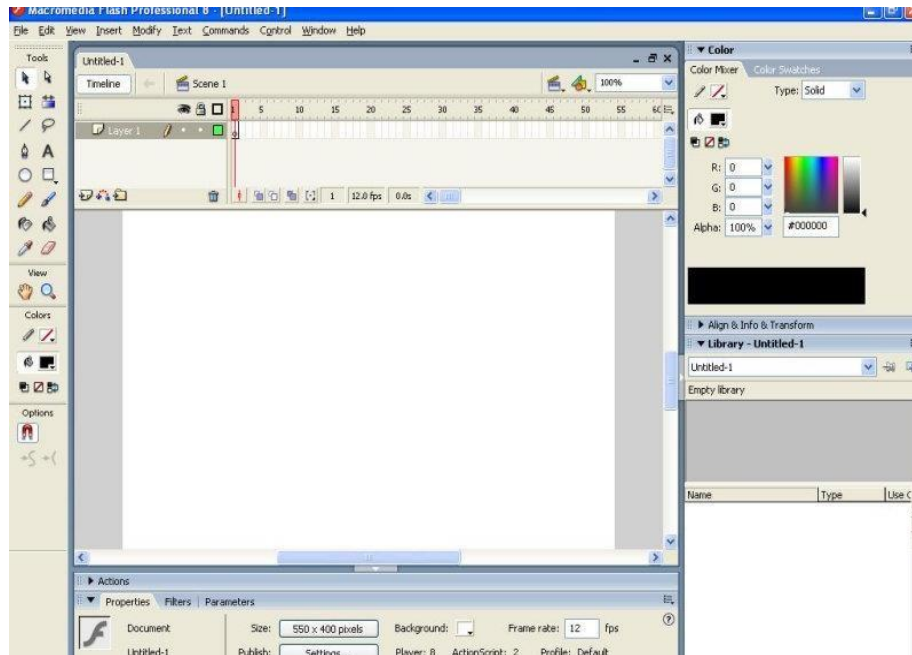
Fig 5.14   Flash elements

The Tools Panel
The Tools panel(as shown in Fig 5.15) is divided into four sections. You can use the tools in the Tools panel to draw, paint, select, and modify artwork, as well as change the view of the stage.

Area On Tools
Tools Contain drawing, painting, and selection tools.
View Contains zooming and panning tools.
Colors Contains tools for setting stroke and fill colors.
Options Displays options for the selected tool that affect the tool's painting or editing operations.

Expanded Stage Work Area
All software from the Adobe CS3 family have similar user interfaces, similar tools, familiar icons, and customizable workspaces that enable you to move smoothly between Flash and other Adobe design software. The gray area around the stage can be used as the expanded stage to store graphics for future use.

Panels
Panels are Flash screen elements that give you easy access to the most commonly used features in Flash. Panels help you view, organize, and change elements in your Flash document. Panels can have different appearances or states and are categorized into different types based on function.
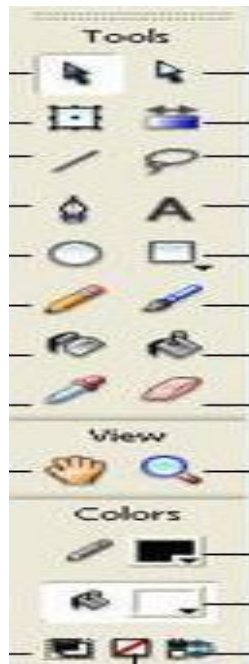
33

Fig 5.15 Tools

Panel States
Panels have three states:
1. Open—Visible in the interface with a content window.
2. Collapsed—Visible in the interface as only a title bar with the content window hidden.
3. Not visible in the interface.
Panel Types
Panels are divided into three different types.
Panel Type Included Panels
Design
- Align
- Color
- Swatches
- Info
- Scene
- Transform

Development
- Actions
- Behaviors
- Components
- Component Inspector
- Debugger
- Output
- Web Services

Other
- Accessibility
- History
- Movie Explorer
- Strings
- Common Libraries

Flash File Formats

There are two main file types used in Flash.
1. **FLA** is the file type you create in the authoring environment. You can think of it as your source file. The term document is associated with this file type.
2. When you publish your document, a **SWF** file is created. The creation process is similar to a compilation process in other languages. This file is actually viewed by users of a web page or Flash application. The term application is associated with this file type. When you publish a Flash document, an HTML file is also created along with the SWF.

How to Produce a Flash Application File

To produce a Flash application file:
1. If necessary, launch Adobe Flash CS3.
2. Open an FLA file.
   a. Choose File→Open or in the Quick Start page, click Open.
   b. In the Open dialog box, navigate to the location where the desired FLA file is stored.
   c. Select the file, and click Open to open the file.
3. Choose File→Publish to publish the file.
4. Minimize the Adobe Flash CS3 Professional window.
5. View the published file.
   a. Navigate to the location where the original FLA file is stored. A SWF file is generated with the same name as that of the FLA file.
   b. Double-click the SWF file to launch and test the application.
   c. Close the Adobe Flash Player 9 window.
6. Restore the Adobe Flash CS3 Professional window.
7. Close the FLA file

Flash Animations

Tweening: Short for *in-betweening,* the process of generating intermediate frames between two images to give the appearance that the first image evolves smoothly into the second image. Tweening is a key process in all types of animation, including computer animation. Sophisticated animation software enables you to identify specific objects in an image and define how they should move and change during the tweening process.

Import

When you work with Flash, you'll often import assets into a document. Perhaps you have a company logo, or graphics that a designer has provided for your work. You can import a variety of assets into Flash, including sound, video, bitmap images, and other graphic formats (such as PNG, JPEG, AI, and PSD). Imported graphics are stored in the document's library. The library stores both the assets that you import into the document, and symbols that you create within Flash. A symbol is a vector graphic, button, font, component, or movie clip that you create once and can reuse multiple times.
So you don't have to draw your own graphics in Flash, you can import an image of a pre-drawn gnome from the tutorial source file. Before you proceed, make sure that you save the source files for this tutorial as described in "Open the finished FLA file", and save the images to the same directory as your banner.fla file.

1. Select File > Import > Import to Library to import an image into the current document.34 Basic Tasks: Creating a banner, Part 1 You'll see the Import dialog box which enables you to browse to the file you want to import. Browse to the folder on your hard disk that contains an image to import
into your Flash document.

2. Navigate to the directory where you saved the tutorial's source files, and locate the bitmap image saved in the FlashBanner/Part1 directory.

3. Select the gnome.png image, and click Open (Windows) or Import (Macintosh). The image is imported into the document's library.

4. Select Window > Library to open the Library panel. You'll see the image you just imported, gnome.png, in the document's library.

5. Select the imported image in the library and drag it onto the Stage. Don't worry about where you put the image on the Stage, because you'll set the coordinates for the image later. When you drag something onto the Stage, you will see it in the SWF file when the file plays.

ADDING SOUNDS

The first step of this process is to prepare the audio file you will be importing. Flash is most compatible with uncompressed audio formats such as WAV and AIFF files, and automatically compresses the audio to MP3 when you publish your movie. If Flash returns an error when you attempt to import an MP3 file, convert it to either WAV or AIFF in a 3rd-party audio conversion application and try again.

Once your audio file is ready, the next step file will be to make a new layer and name it audio. This naming system is of course only a suggestion.
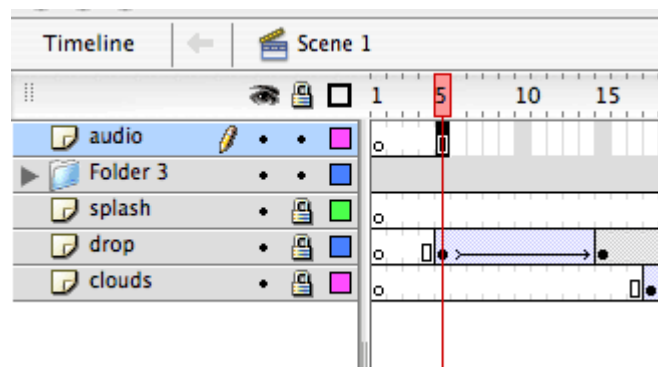

Fig 5.16 Adding sounds

In the next step we will locate the file we wish to import. Go to 'File>Import>Import to Library' and browse to your audio file. Selecting it once it is located will import the file into the library and a new audio icon will appear with your other files.

Once the audio file has been imported into the library, select the audio layer in your timeline, and drag your file from the library directly onto the work area of your main stage. You should see a thin waveform appear in those few frames in the audio layer of your timeline. Now select the end frame and drag it further down the timeline, this should reveal the rest of the waveform in the audio layer.

These waveform spikes will assist you in timing to your audio to your animation. If you want to move the beginning of the audio around in the layer, you can select the first frame, and drag it to the frame where you would like the audio to begin. Then just grab the last frame and drag it to where you want your audio to end. One thing to know is that the audio file on the layer cannot have a Keyframe between the beginning and the end of the waveform. If you insert

35

one it will stop the audio at that point and clear any audio after it. You can still move the frame at the end of the audio to reveal the waveform again however. Also it would likely make it easier if you reserve this layer only for the audio you have applied to it.

If you are importing a longer audio file that requires a fade out you will have to do that. If you do not cut the audio down, your .swf file will get to the end of the animation and repeat while the previous audio continues to play. This will result in multiple layers of audio playing at once, and will distort the audio. To fade the audio out simply select it in the "audio" layer, and you will notice the right hand side of your properties inspector will display the audio file name.
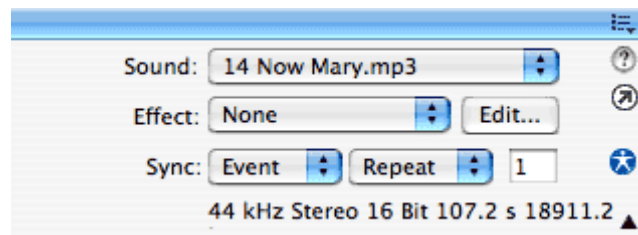

Fig 5.17:audio inspector

Select edit from your properties inspector as shown in fig.5.17. This will open the edit envelope window, which displays the left and right channel waveforms up close. The first thing to do is check and bottom of the window and make sure frames is selected instead of time. Doing this will change to numbers between the waves to correspond to the frame on which the audio is happening instead of at what time it happens.

In Figure 5.18 the dark line across the top of the waveform represents the audio level. If you click on it, you will create a control point. I created two control points, then left one at the top around frame 158, and dragged the other to the bottom around frame 171.
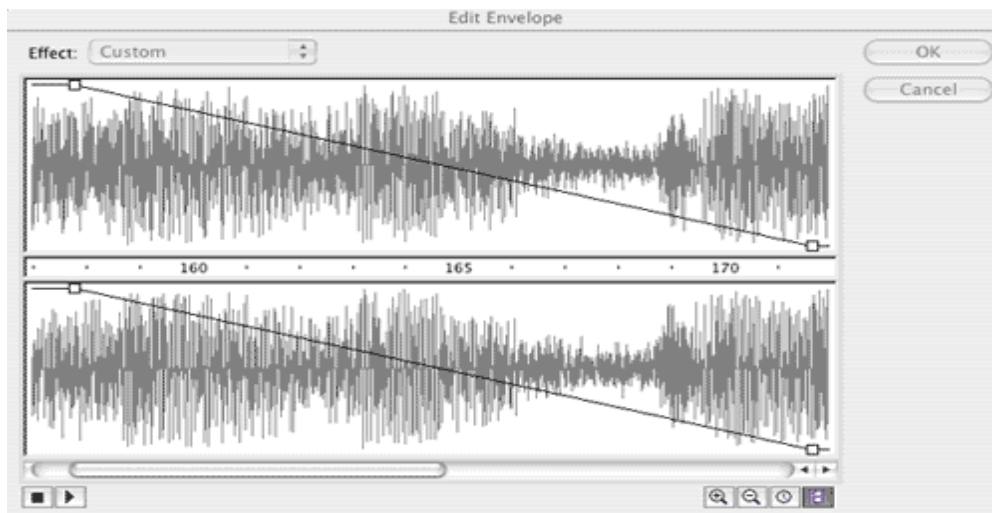

Fig 5.18:fading the audio out

This will cause the audio to fade out when it gets to these frames. My audio is now where I want it to be. You can also use this technique to edit the level of the left and right channels throughout your project if you wanted certain parts to be louder or softer.

Action Scripts

You'll need to give Flash instructions. Among other things, you'll tell Flash to stop the movie, play the movie, or jump to a specific place in the timeline. These tasks that Flash will perform at your request are called "actions".

There are two types of actions.
Button actions perform a specific task when a button is clicked. Among other things, you can use button actions to jump to a different part of the timeline, stop the movie, or instruct a movie clip to start playing. We're not going to mess with buttons too much right now, but well get to them later.
The other type of action is a frame action. A frame action is automatically triggered when the playback head reaches a certain frame. Among other things, frame actions can be used to stop the movie, jump the playback head to a different part of the timeline, or preload frames to ensure a smooth playing animation.

GO TO & PLAY / GO TO & STOP

Using Frame Actions

In this section, well go through an exercise to use the stop action to keep a movie from looping. The stop action can also be used to halt the movie while youre waiting for a user to make a choice about what button they want to click or what they're going to do next.
1. Build a small, twenty-frame movie. It doesn't have to be anything special. A symbol animating across the screen will do fine.
2. Hit F12 to test the movie.
   Notice that it loops.
Now we're going to place a stop action on the last frame of our twenty-frame movie. The stop action will keep halt the playback head when it reaches frame 20, thus keeping the movie from looping.
3. Add a new layer and name it Actions as shown in Fig. 5.19.
   Remember that frame actions should get their own top layer.
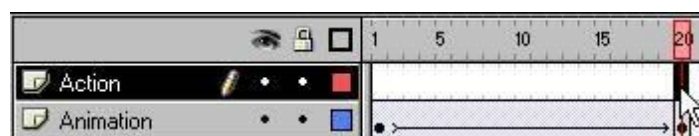4. On the Actions Layer, add a keyframe (F6) on frame 20

Fig 5.19 Frame action

5. Open the Actions Palette as in figure 5.20.
   (WINDOW-> ACTION)

Fig 5.20 Frame Action

6. Open the Basic Actions By clicking on the plus sign. You'll be able to see the actions as demonstrated in figure 5.21.
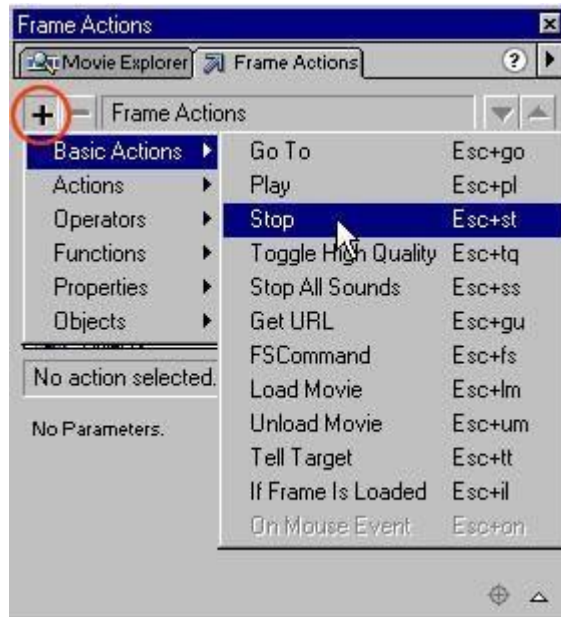

Fig 5.21Opening Basic Actions

7. Choose Stop as shown in figure 5.22.
   You will notice that coding for your action was added to the right side of the Frame Action Palette.
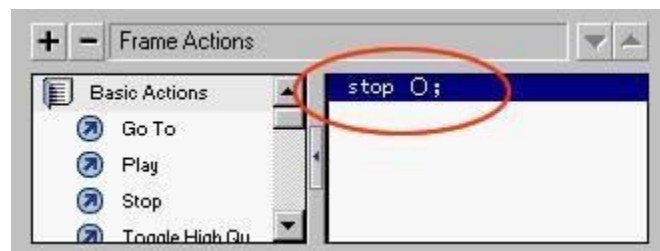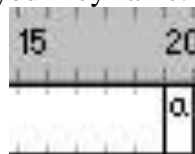

Fig 5.22 Stop frame action

You may also notice that a small "a" appeared in your keyframe.

8. Hit F12 to test your movie.It should stop at the end!

References:
http://www.readorrefer.in/article/Multimedia-Basics_10188/