Introduction of computer animation

- Computer animation is referred to any time sequence of visual changes in a picture
  - For example, transforming a car of motor oil into an auto-mobile machine
- Considerations in computer-generated animations
  - Camera parameters: position, orientation or focal length
  - Lighting effects or other parameters and procedures associated with illumination and rendering
  - Realism
- Basic method for constructing a motion sequence
  - Real-time animation, each stage of the sequence is viewed as created
  - Frame-by-frame animation, each frame of the motion is separately generated and stored

Raster methods for computer animation (1)

- Animation sequence can be produced on a raster-system one frame at a time
- Each completed frame could be saved in a file for later viewing
- Generate animation in real time
  - Produce the motion frames quickly enough so that a continuous motion sequence is displayed
  - One frame of the animation could take most of the refresh time cycle time to construct in complex scene
  - Lead to erratic motions and fractured frame displays for a very complex animations

Raster methods for computer animation (2)

- Method for producing a real-time animation
  - Double buffering
    - Employ two refresh buffers
    - Create a frame for the animation in one of the buffers
    - Construct the next frame in the other buffer, while the screen is being refreshed
    - Switch the role of two buffers when the second frame is complete
  - Using raster operations
    - Using block transfers of a rectangular array of pixel values
    - Often used in game-playing programs
    - Color-table transformation are used to animate object along 2D motion paths

Design of animation sequence (1)

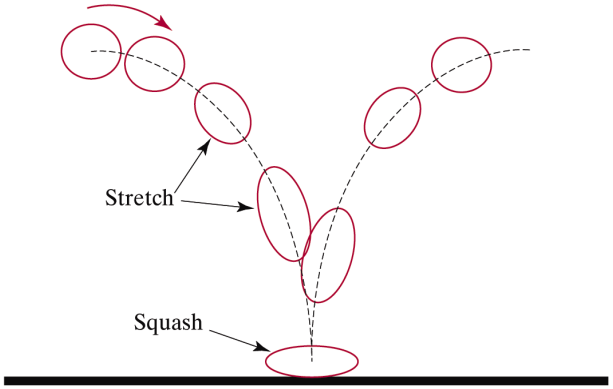
- Development stages used to design animation sequences
  - Storyboard layout
  - Object definitions
  - Key-frame specifications
  - Generations of in-between frames
- Storyboard
  - An outline of the action
  - Defines the motion sequence as a set of basic events that are to take place
- Object definition
  - Given for each participant in the action
  - Objects can be defined in terms of basic shapes
  - A description is often given of the movements that are to be performed by each character or object

Design of animation sequence (2)

- Key frame
  - Detailed drawing of the scene at a certain time in the animation sequence
  - Within a key frame, each object is positioned according to the time for that frame
  - Example, one frame from the short film Tins Toy
- In-between
  - Intermediate frames between the key frames
  - The total number of in-between frames is determined by the display
    media that is to be used

Traditional animation techniques (1)

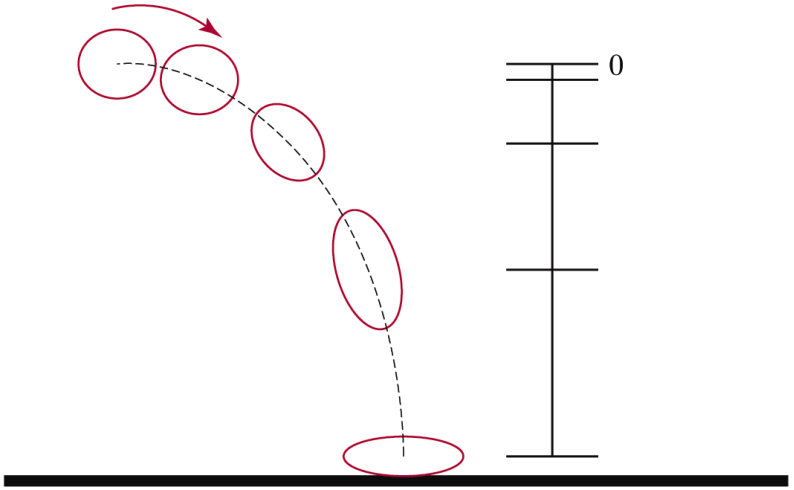
- Film animators use a variety of methods for depicting and emphasizing motion sequences
  - Including object deformations, spacing between animation frames, motion anticipation and follow-through, and action focusing
- Squash and stretch
  - Technique for simulating acceleration effects, particularly for non-rigid objects
  - Example, a bouncing-ball illustration



Computer Graphics with Open GL, Third Edition, by Donald Hearn and M.Pauline Baker.

Traditional animation techniques (2)

- Timing
  - Refers to the spacing between motion fames
  - A slower moving object is represented with more closely spaced frames and a faster moving object us displayed with fewer frames
    over the path of the motion
  - Example, motion frames for bouncing-ball

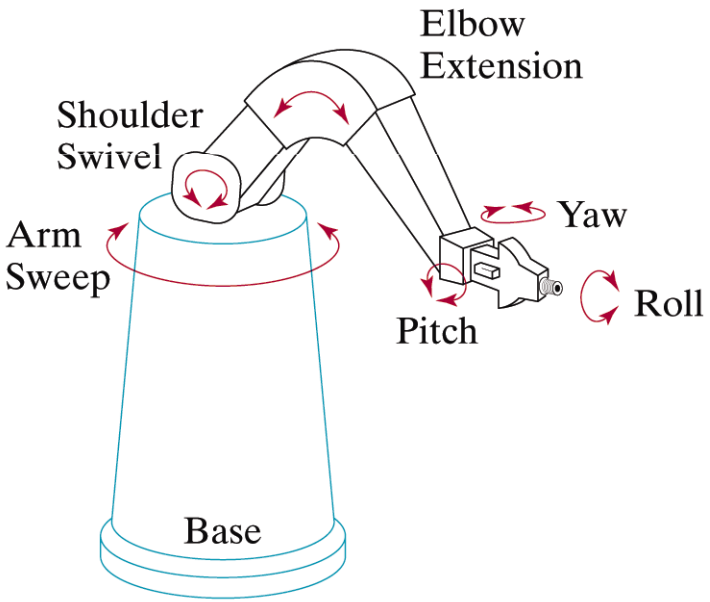


General computer-animation functions

- Software package are developed for general animation design or performing specialized animation tasks
- Typical animation functions
  - Include managing object motions, generating views of objects, producing camera-motions and the generation of in-between frames
- Some animation package provide functions for both the overall
  animation design and the processing of the individual objects
  - For example: Wavefront
- A set of routines is often provide for storage and managing the object database
  - Object shapes and associated parameters are stored and updated in the database

Computer-animation languages (1)

- Develop routines to design and control animation sequences within a general-purpose programming language
- Specialized animation language have been developed



Computer Graphics with Open GL, Third Edition, by Donald Hearn and M.Pauline Baker.

Computer-animation languages (2)

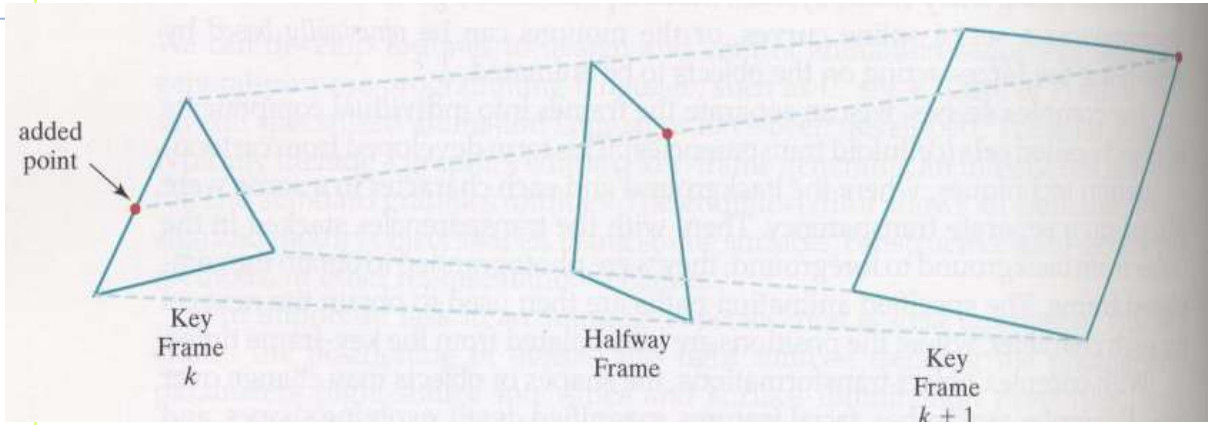
- Animation description
  - Scene description, includes the positioning of objects and light sources, defining the photometric parameters and setting the camera parameters
  - Action description, involves the layout of motion paths for the objects and camera
- Key-frame systems
  - Originally designed as a separate set of animation routines for generating the in-between from the user-specified key frames
- Parameterized systems
  - Allow object motion characteristics to be specified as part of the object definitions
- Scripting systems
  - Allow object description and animation sequences to be defined with a user-input script

Key-frame systems (1)

- A set of in-between can be generated from the specification of two (or more) key frames
- For complex scene, separate the frame into individual components or object called cels
  - Transparencies stacked in the order from background to foreground
  - Obtain complete frame by photographing transparencies
  - Obtain the next cels for each character with the specified animations paths
- For complex object transformations
  - The shapes of objects may change overtime
  - Example: clothes, facial features and evolving shapes
- For surfaces described with polygon meshes
  - Changes can result in significant changes in polygon shape

Key-frame systems (2)

- **Morphing**
  - Transformation of object shapes from one form to another
  - For two key frames with a different number of line segments specifying an object
    - Adjust the object description in one of the frames
    - To balance the number of polygon edges or the number of vertices
  - Example
    - Linear interpolation for transforming a triangle to a quadrilateral



Key-frame systems (3)

- **Edge equalization**
  - **The maximum and minimum number of lines to be equalized**

$$L_{\max} = \max(L_k, L_{k+1})$$

$$L_{\min} = \min(L_k, L_{k+1})$$

  - **Where L_k and L_{k+1} denote the number of line segments in two successive frames**
  - **Compute the quantities**

$$N_e = L_{\max} \bmod L_{\min}$$

$$N_s = \text{int}\left(\frac{L_{\max}}{L_{\min}}\right)$$

  - **Preprocessing steps for edge equalization**
    - **Divide N_e edges if $\text{keyframe}_{\min}$ into N_s +1 sections**
    - **Divide the remaining lines of $\text{keyframe}_{\min}$ into N_s sections**

Key-frame systems (4)

- **Vertex equalization**
  - **The maximum and minimum number of vertices**

$$V_{\max} = \max(V_k, V_{k+1})$$

$$V_{\min} = \min(V_k, V_{k+1})$$

  - **Where V_k and V_{k+1} denote the number of vertices in two successive frames**
  - **Compute the quantities**

$$N_{ls} = (V_{\max} - 1) \bmod (V_{\min} - 1)$$

$$N_p = \text{int}\left(\frac{V_{\max} - 1}{V_{\min} - 1}\right)$$

  - **Preprocessing steps for vertex equalization**
    - **Add N_p points to N_{ls} line section of $\text{keyframe}_{\min}$**
    - **Add N_p-1 points to remaining edges of $\text{keyframe}_{\min}$**

Key-frame systems (5)

- Simulating accelerations
  - Use curve-fitting techniques to specify the animation paths between key frames
  - Positions can be fitted with the paths by given vertex position at the key frames
- For a motion with constant speed
  - The time interval between the key frames is divided into n+1 equal sub-interval

$$\Delta t = \frac{t_2 - t_1}{n+1}$$

  - The time for the j th in-between is

$$tB_j = t_1 + j\Delta t$$

$$j = 1,2,\ldots., n$$

Key-frame systems (6)

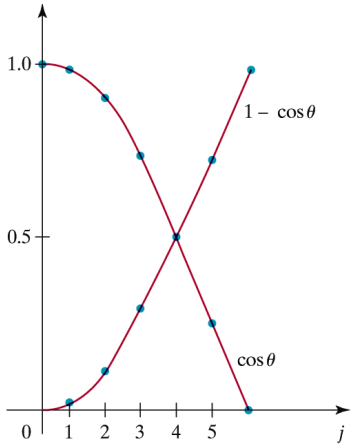
- For a motion with increasing speed (positive acceleration)
  - Obtain an increasing size for the time interval with the function

$$1 - \cos\theta \qquad 0 < \theta < \pi / 2$$

  - The time for the j th in-between is

$$tB_j = t_1 + \Delta t\left[1 - \cos\frac{j\pi}{2(n+1)}\right]$$

$$j = 1,2,\ldots,n$$

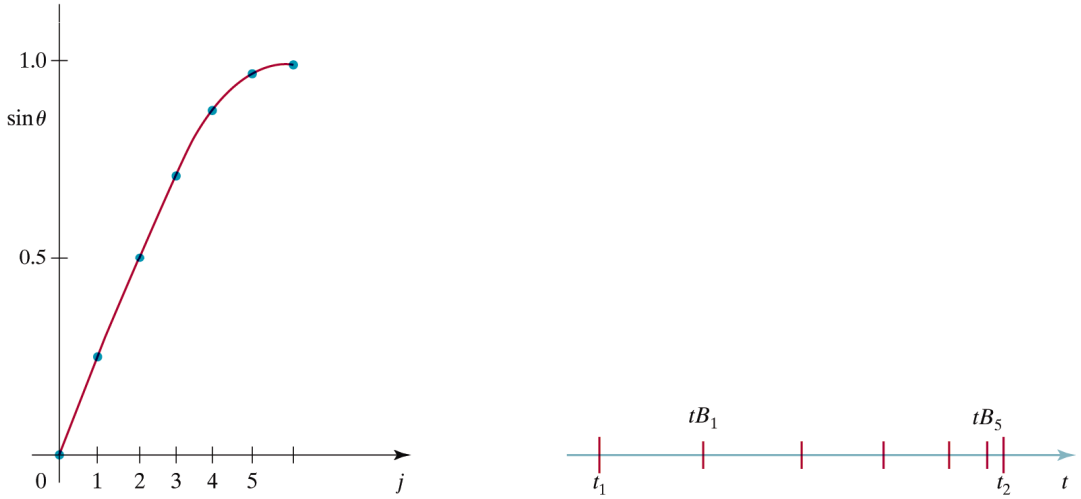


Key-frame systems (7)

- For a motion with increasing speed (negative acceleration)
  - The time for the j th in-between is

$$tB_j = t_1 + \Delta t \sin\frac{j\pi}{2(n+1)}$$

$$j = 1,2,\ldots,n$$



Key-frame systems (8)

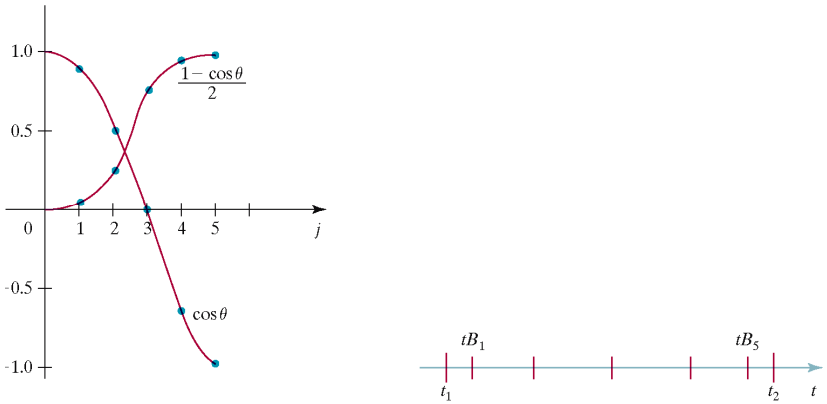
- For a motion with both speedups and slowdowns
  - A function to accomplish the time changes

$$\frac{1}{2}(1-\cos\theta) \qquad 0 < \theta < \pi/2$$

  - The time for the j th in-between is

$$tB_j = t_1 + \Delta t\left\{\frac{1-\cos[j\pi/(n+1)]}{2}\right\}$$

$$j = 1,2,\ldots,n$$



Motion specifications (1)

- Direct motion specification
  - The most straightforward method for defining an animation
    - set the values for the rotation angle and translation vectors
    - Apply transformation matrices to transform coordinate positions
  - Approximate equation involving parameters to specify certain kind of motions
    - For example, approximate the path of bouncing ball with a damped rectified, as sine curve

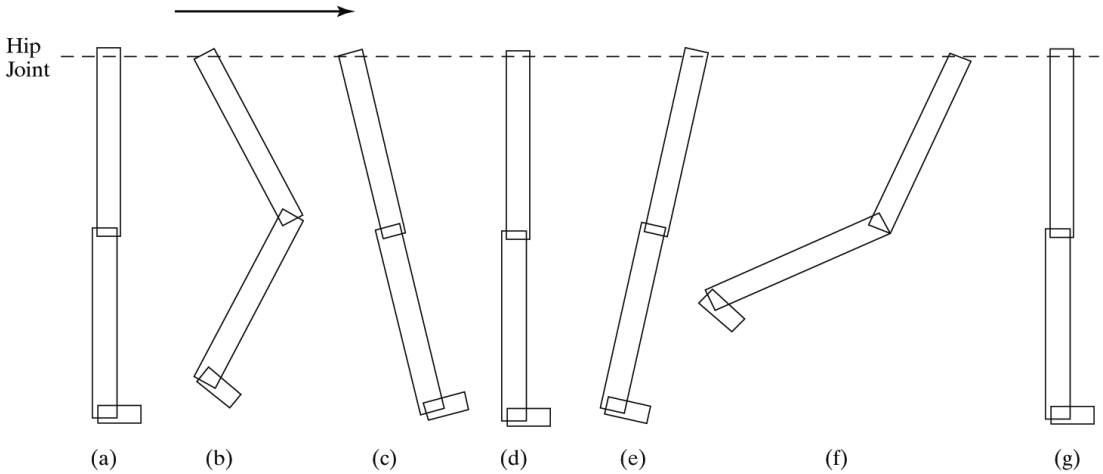
$$y(x) = A\left|\sin(\omega x + \theta_0)\right| e^{-kx}$$

Motion specifications (2)

- Goal-Directed systems
  - An animation is specified in terms of the final stage of the motions
  - For example. Specify an object to “walk” or to “run” to a particular destination
- Kinematics and Dynamics
  - Kinematics description, specify the animation by giving motion parameters (position, velocity and acceleration)
    - No reference to causes or goals of the motions
  - Inverse kinematics, specify the initial and final positions of object at specified times and motion parameters are computed by system
  - Dynamic description, require the specification of the forces that produce the velocities abs accelerations
    - Referred as physically based modeling

Articulated figure animation

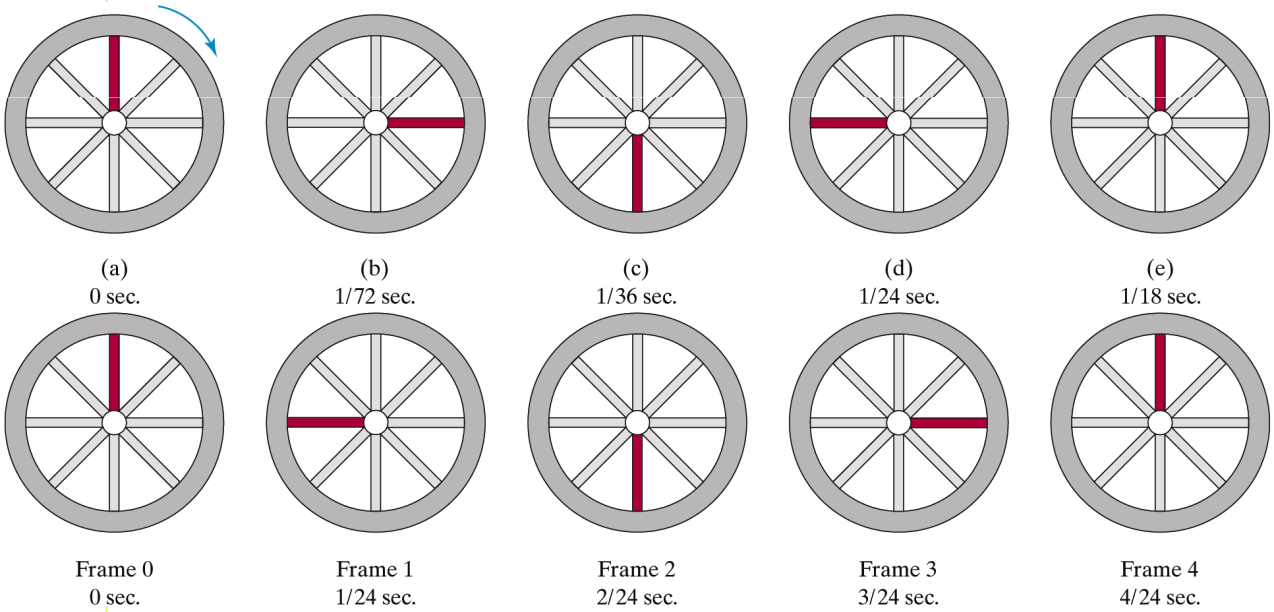
- A basic technique for animating people or other critters
  - Model animated objects as moving stick figures or simplified skeletons
  - Connection points or hinges are placed to the shoulders, hips, knees and other skeletal joints
  - A series of walking led motions



Computer Graphics with Open GL, Third Edition, by Donald Hearn and M.Pauline Baker.

Periodic motions

- Construct an animation with repeated motion patterns
  - For example: rotating object
  - The motion must be synchronized with the frame-generation rate
  - Typical example of an undersampled periodic motion



Computer Graphics with Open GL, Third Edition, by Donald Hearn and M.Pauline Baker.
ISBN 0-13-0-15390-7 © 2004 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

OpenGL animation procedure (1)

- Core library
  - Include raster operations and color-index assignment functions
- GLUT
  - Provide routine for changing color-table values
- Activate double-buffering operations

  glutInitDisplayMode (GLUT_DOUBLE);
- Interchange the role of buffers

  glutSwapBuffers ();
- To determine the availability of double buffering operation

  glutGetBooleanv (GLUT_DOUBLEBUFFER, status);
- Continuously execute animation

  glutIdleFunc (animationFcn);

Computer Graphics

Color Models

Represented By: Haitham Abdel-atty Abdullah
Donia Gamal El-din

Pre-Master 2014-2015

Supervised by: Prof. Taha El-Areef

Outline:

Properties of Light

Color Models

Standard primaries and the chromaticity diagram

The RGB color model

The YIQ and related color model

The HSV color model

The HSL Color model

The CMY and CMYK Color models

Color Models Applications

Dithering VS Half-toning

Properties of Light

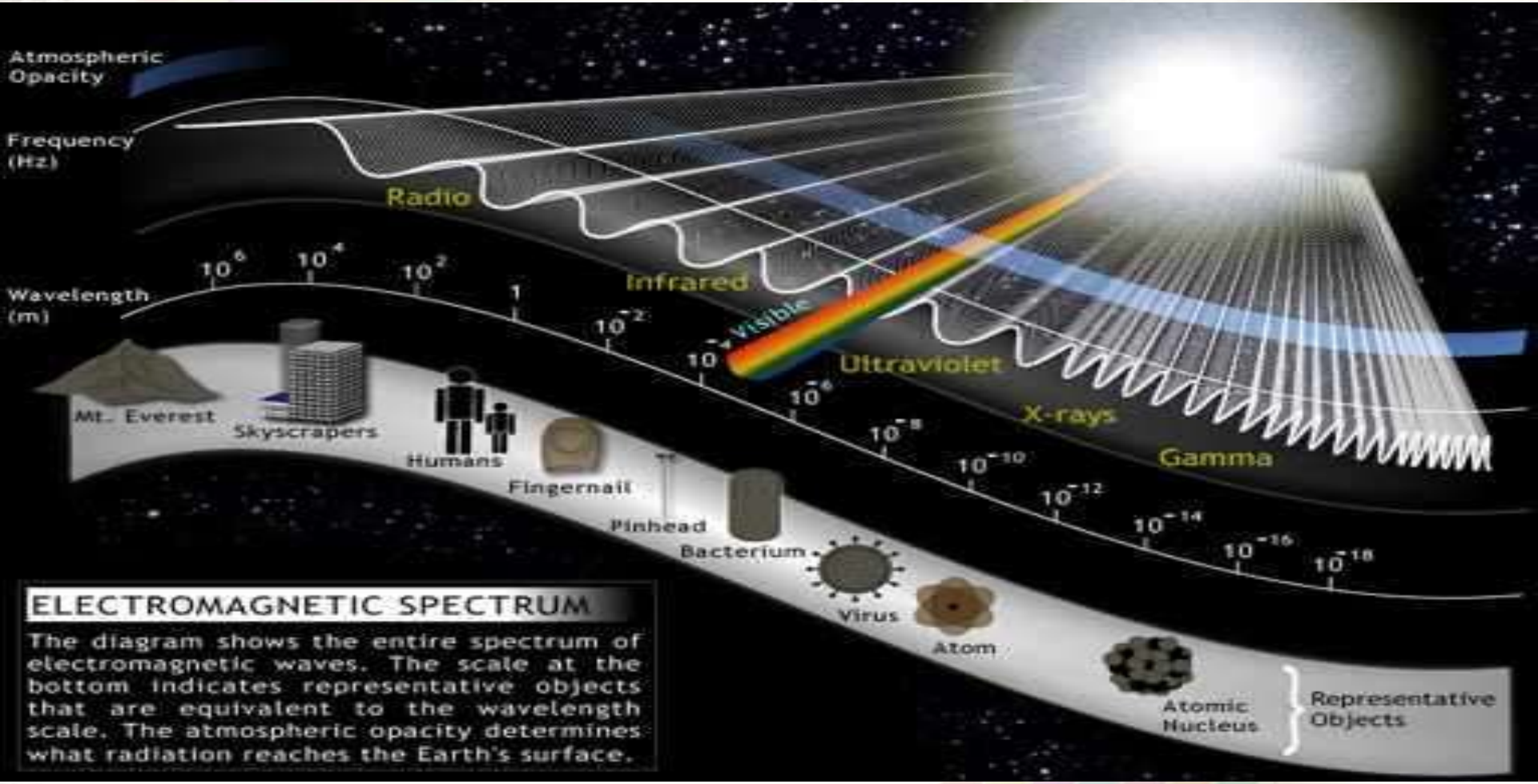
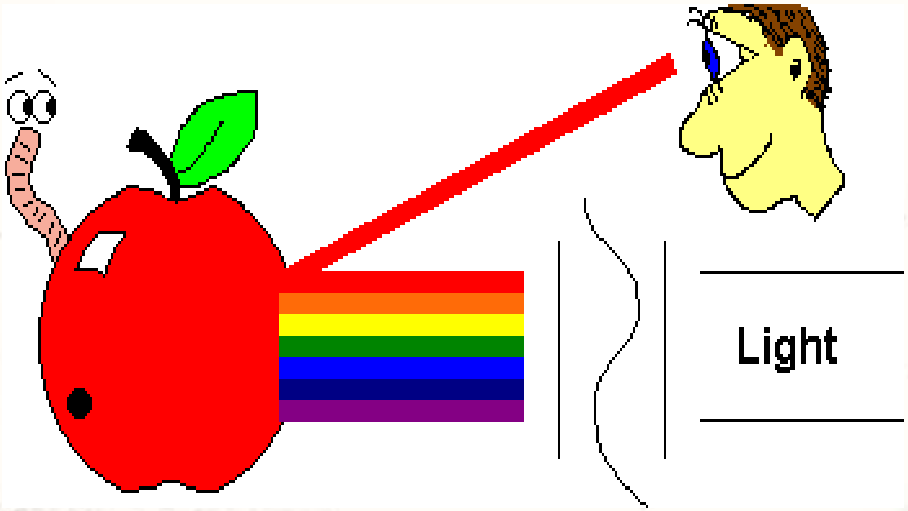


FIGURE-1

Each frequency value within the visible region of the electromagnetic spectrum corresponds to a distinct **spectral color.**

Properties of Light

- When white light is incident on an opaque object , some frequencies are reflected and some are absorbed.
- The combination of frequencies present in the reflected in the reflected light determines the color of the object that we see.(**Dominant frequency** or **Hue**)



Properties of Light

- In the wave model of electromagnetic radiation, Light can be described as oscillating transverse electric & magnetic fields propagating through space.

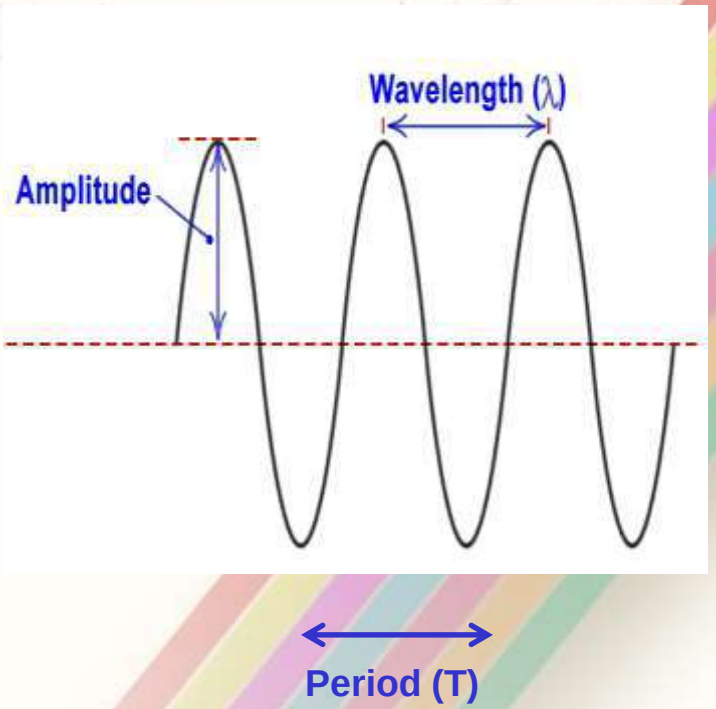
<u>For each Spectral Color :</u>

- Rate of oscillation = frequency = f
- Time between any two consecutive positions on the wave that have same amplitude = Period of wave= T
- Period is the inverse of frequency

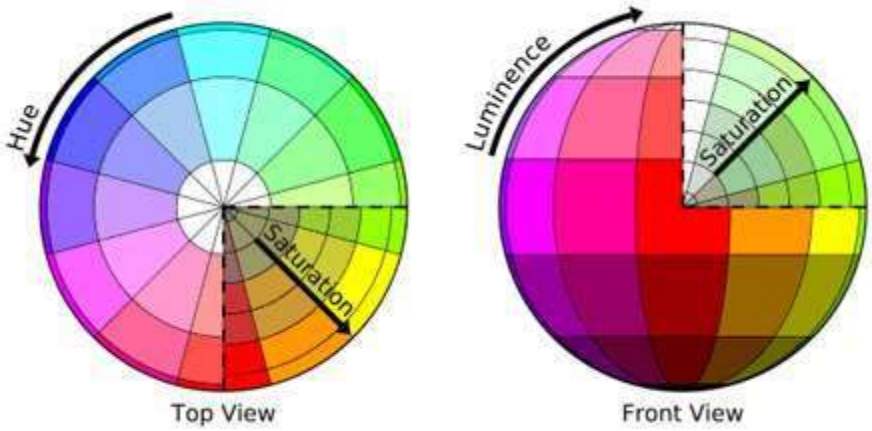
$$T = 1/f$$

- Distance travelled from beginning of one oscillation to the beginning of the next oscillation = wavelength = λ
- The wavelength and frequency are inversely proportional to each other with the proportionality constant as the speed of light (c)

$$c = \lambda f$$



Characteristics of Color



1. Dominant Frequency (Hue)
   The color we see (red, green, purple).

2. Brightness
   The total light energy, how bright is the color (How bright are the lights illuminating the object?)

3. Purity (Saturation)
   Purity describes how close a light appears to be to a pure spectral color, such as pink is less saturated than red.

Chromaticity refers to the two properties (purity & hue) together.

Color Model

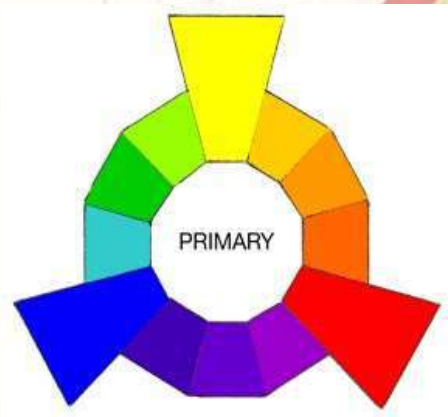
•A color model is an abstract mathematical model describing the way colors can be represented as tuples of numbers, typically as three or four values or color components. *[Wikipedia]*

•Any method for explaining the properties or behavior of color within some particular context is called a Color Model.*[Hearn, Baker ,computer graphics with OpenGL]*

Color Model

Primary Colors

Sets of colors that can be combined to make a useful range of colors



Color Gamut

Set of all colors that we can produce from the primary colors.

Complementary Colors

Pairs of colors which, when combined in the right proportions, produce white.

Example, in the RGB model: red & cyan , green & magenta , blue & yellow.

- No finite set of real primary colors can be combined to produce all possible visible colors.

- However, given a set of three primary colors, we can characterize any fourth color using color-mixing processes.

Color Model

Shades , Tints & Tones

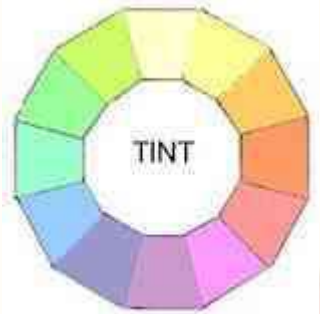
- A shade is produced by “dimming ” a hue.[Adding black].

  Dark Blue = pure blue + black

- A tint is produced by "lightening" a hue. [Adding white].

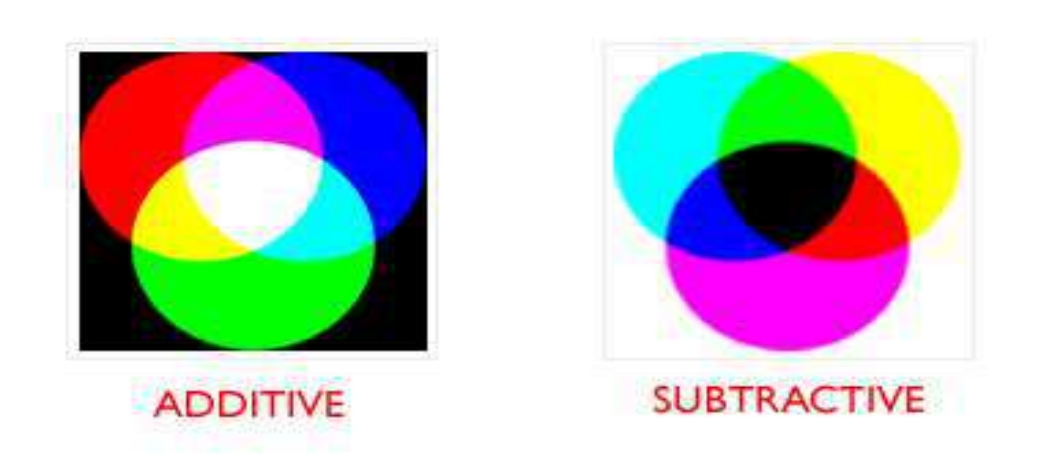
  Pastel red = pure red + white

- Tone refers to the effects of reducing the "colorfulness" of a hue. [adding gray] or [adding black & white].



Thus, shading takes a hue toward black, tinting takes a hue towards white, and tones cover the range between.

Color Model



Additive color

Uses light to display color. Mixing begins with black and ends with white; as more color is added, the result is lighter and tends to white. Used for computer displays

Example: The RGB colors are light primaries and colors are created with light.

A subtractive color

Uses ink to display color. Mixing means that one begins with white and ends with black; as one adds color, the result gets darker and tends to black. Used for printed material

It is called 'subtractive' because its wavelength is less than sum of the wavelengths of its constituting colors.

Example: The CMYK color system is the color system used for printing.

Standard Primaries & the chromaticity diagram

•This is an international standard for primary colors established in 1931.

• It allows all other colors to be defined as weighted sum of the three "primary" colors.

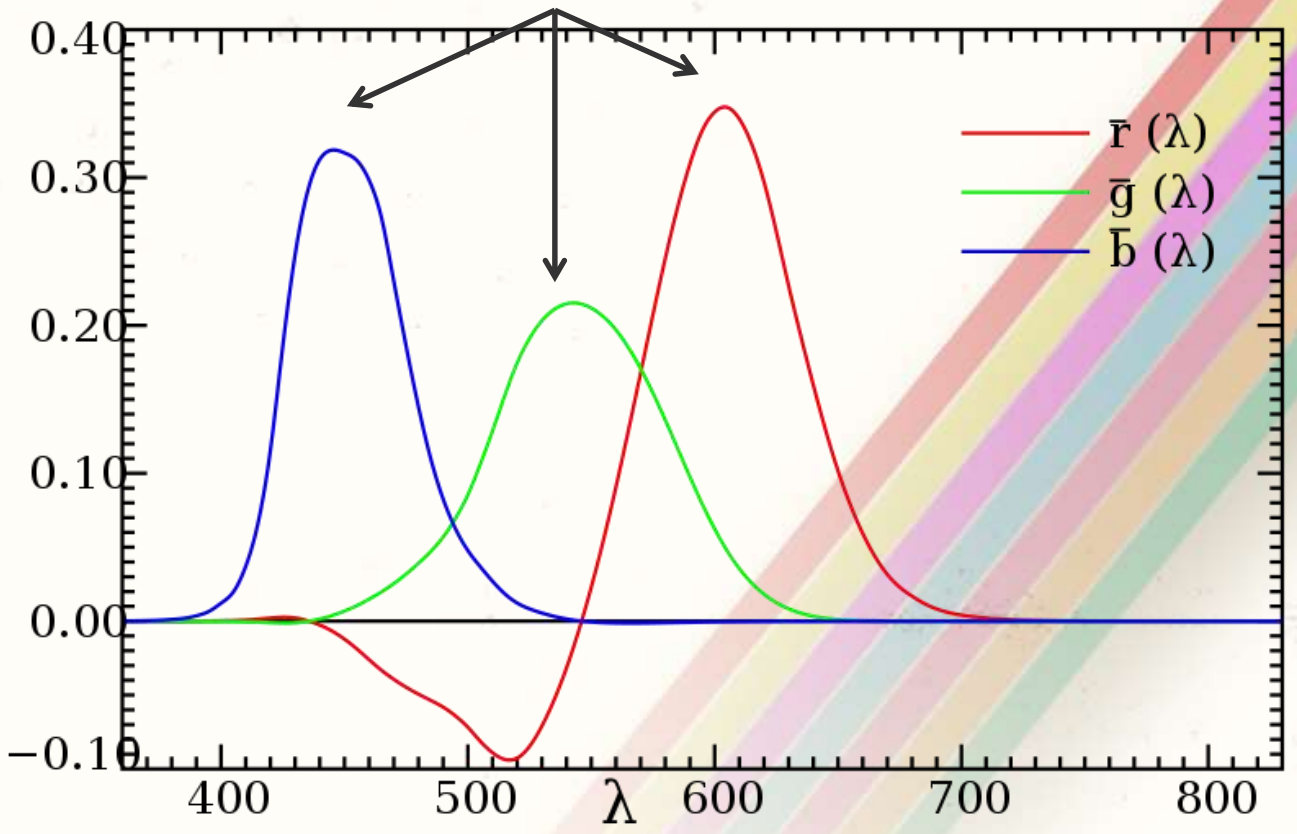
• There are no real three colors that can be combined to give all possible colors. Therefore the Three standard primaries are selected [imaginary numbers].

•They are defined mathematically with *positive color-matching functions* that specify the amount of each primary needed to describe any spectral color.

Standard Primaries & the chromaticity diagram

A color in the vicinity of 500nm can be matched only but subtracting an amount of red light from a combination of blue and green lights.

Color-matching functions



Standard Primaries & the chromaticity diagram

- **The XYZ Color Model**
- X,Y & Z are the amount of each primary needed to produce a selected color.

$$C(\lambda) = (X,Y,Z)$$

- X,Y & Z are calculated from the color-matching functions

$$X = k \int_{visible\lambda} f_x(\lambda) I(\lambda) d\lambda$$

$$Y = k \int_{visible\lambda} f_y(\lambda) I(\lambda) d\lambda$$

$$Z = k \int_{visible\lambda} f_z(\lambda) I(\lambda) d\lambda$$

K = 683 lumens/watt

$I(\lambda)$: light intensity in a particular direction

f_y : chosen such that Y is the luminance for that color.

Standard Primaries & the chromaticity diagram

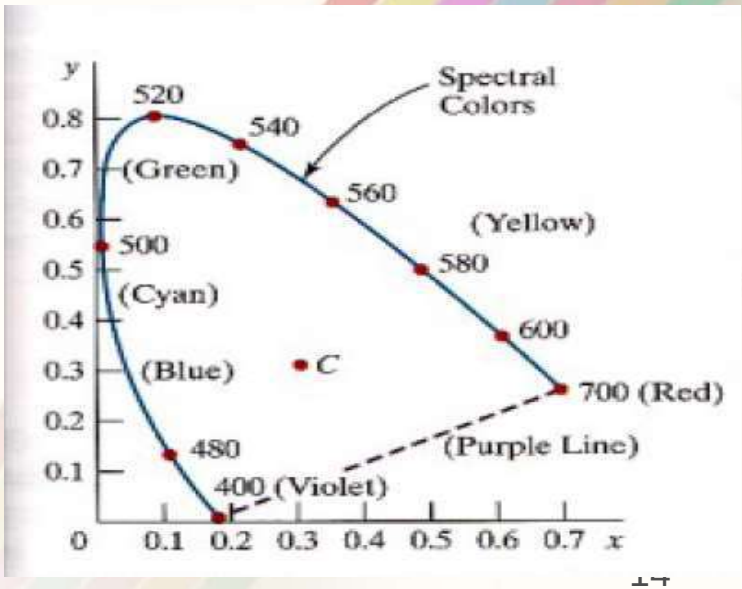
To define a color in CIE model, provide weights for the X, Y and Z primaries, just as you would for an RGB display (e.g. color = xX + yY + zZ).

- X, Y and Z form a three dimensional color volume.

- We can ignore the dimension of luminance by normalizing with total light intensity, x+y+z = 1.

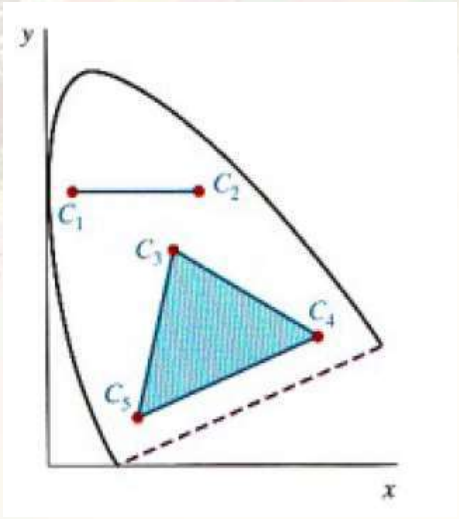
$$x = \frac{X}{X+Y+Z} \qquad y = \frac{Y}{X+Y+Z}$$

$$z = \frac{Z}{X+Y+Z} = 1 - x - y$$

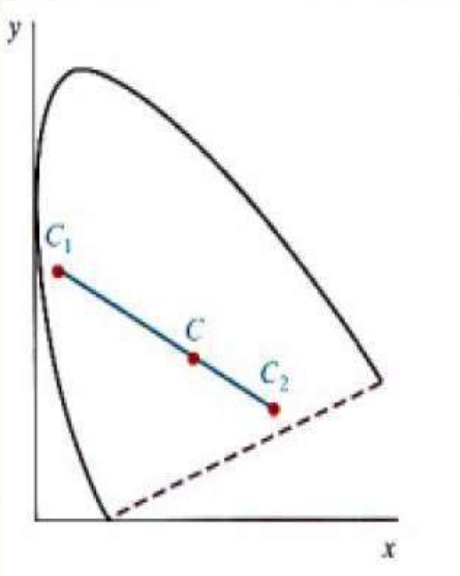


CIE chromaticity diagram.

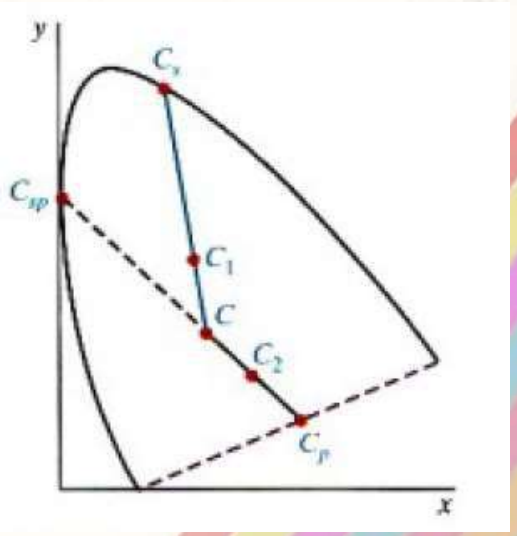
Standard Primaries & the chromaticity diagram



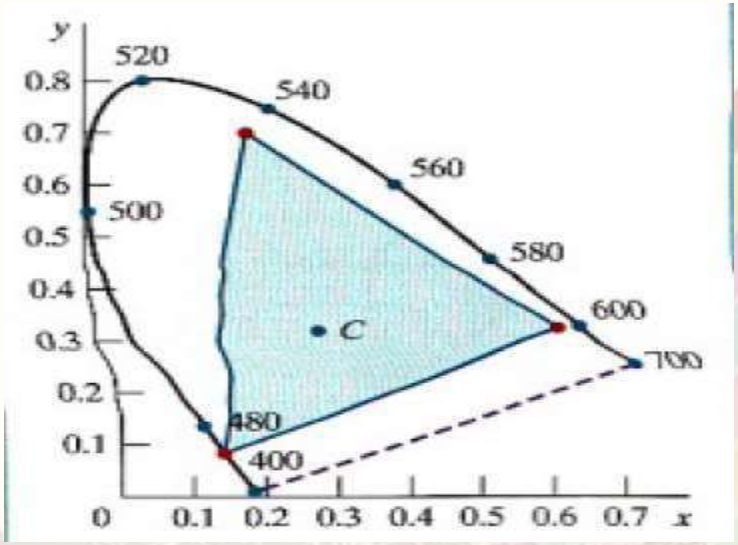
Gamut Color



Complementary Color



Dominant Wavelength
&
Purity



RGB Model

RGB Model

•The red, green, and blue (RGB) color space is widely used throughout computer graphics.

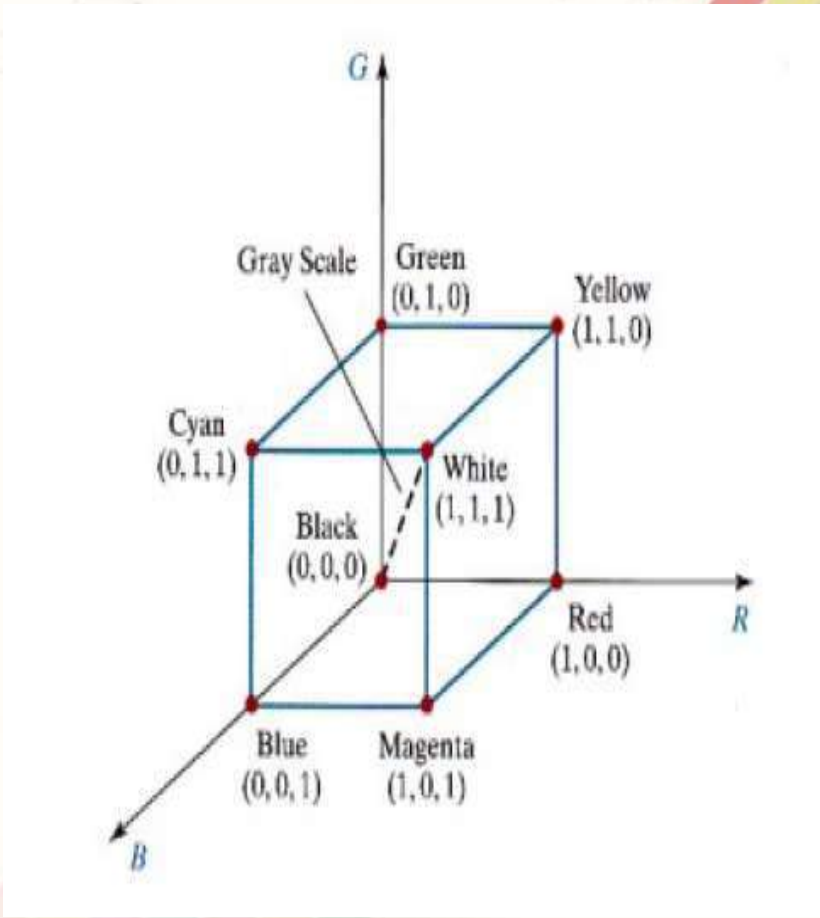
•Additive Color Model.

•Unit Cube defined on R, G & B axes.

•The Origin (0,0,0) represents black and the diagonally opposite vertex (1,1,1) is White.

•Vertices of the cube on the axes represent primary colors, and the remaining vertices are the complementary color points for each of the primary colors.

•Shades of gray are represented along the main diagonal.



RGB Model

Each color point within the unit cube can be represented as w weighted vector sum of the primary colors, using unit vectors **R**,**G** and **B**.

$$C(\lambda) = (R,G,B) = R\mathbf{R} + G\mathbf{G} + B\mathbf{B}$$

Where R,G, and B are assigned values in the range from 0 to 1.0.

For example , the magenta vertex is obtained by adding the maximum red and blue values to produce : (1,0,1)

YIQ model

•YIQ model is used for US TV broadcast.

•This model was designed to separate chrominance (I and Q) from luminance (Y).

•This was a requirement in the early days of color television when black-and-white sets still were expected to pick up and display what were originally color pictures

•The Y-channel contains luminance information (sufficient for black-and-white television sets) while the I and Q channels carried the color information.

YIQ model

•A color television set would take these three channels, Y, I, and Q, and map the information back to R, G, and B levels for display on a screen.

•The advantage of this model is that more bandwidth can be assigned to the Y-component (luminance) because the human visual system is more sensitive to changes in luminance than to changes in hue or saturation

Convert From RGB To YIQ

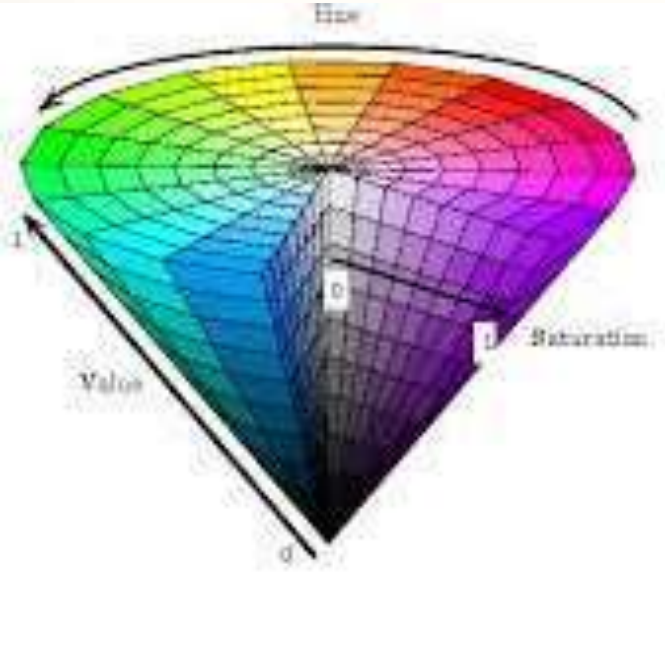
$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.595716 & -0.274453 & -0.321263 \\ 0.211456 & -0.522591 & 0.311135 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

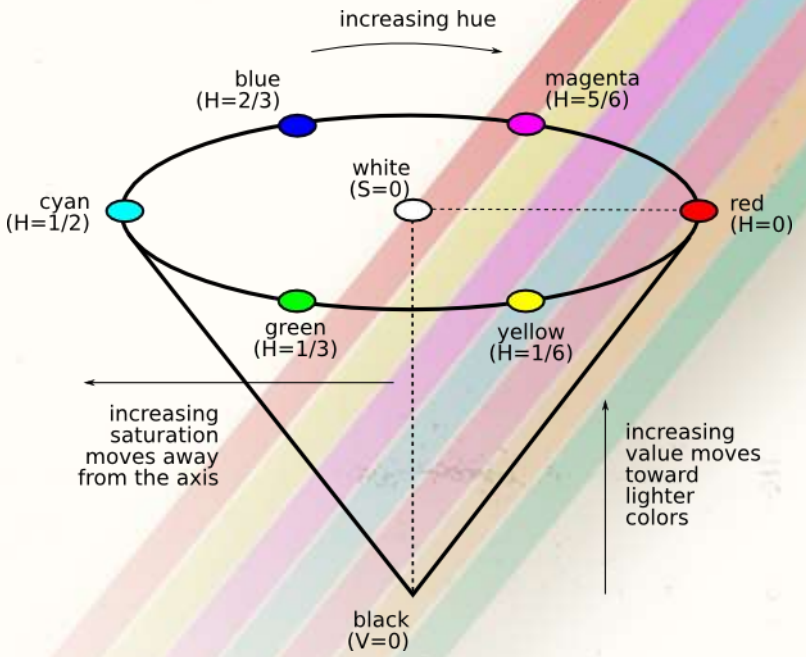
Convert From YIQ To RGB

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0.9563 & 0.6210 \\ 1 & -0.2721 & -0.6474 \\ 1 & -1.1070 & 1.7046 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$

HSV Model

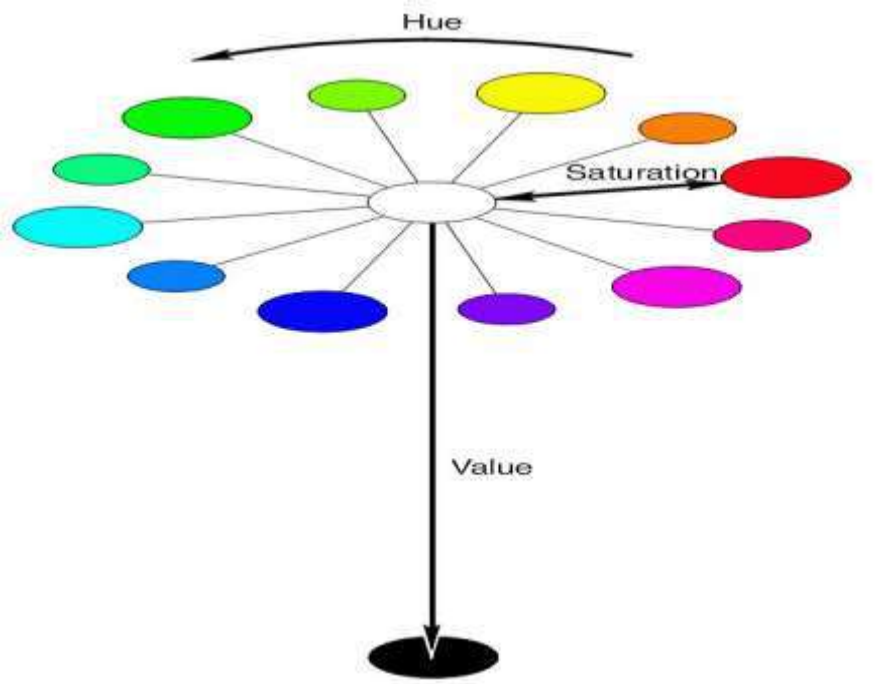
Every color is represented by three components Hue (H), Saturation (S) and Value (V)

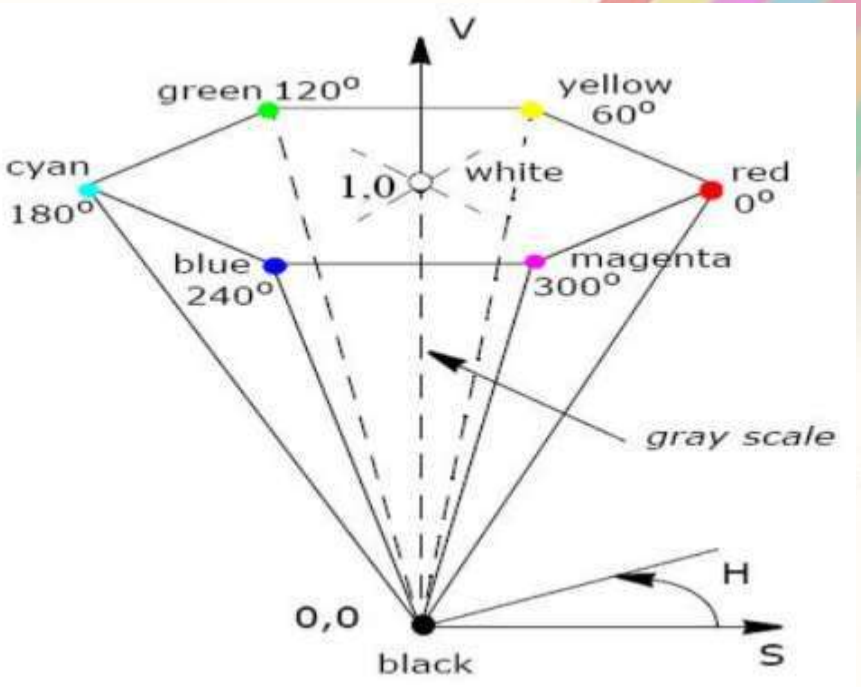




HSV Model

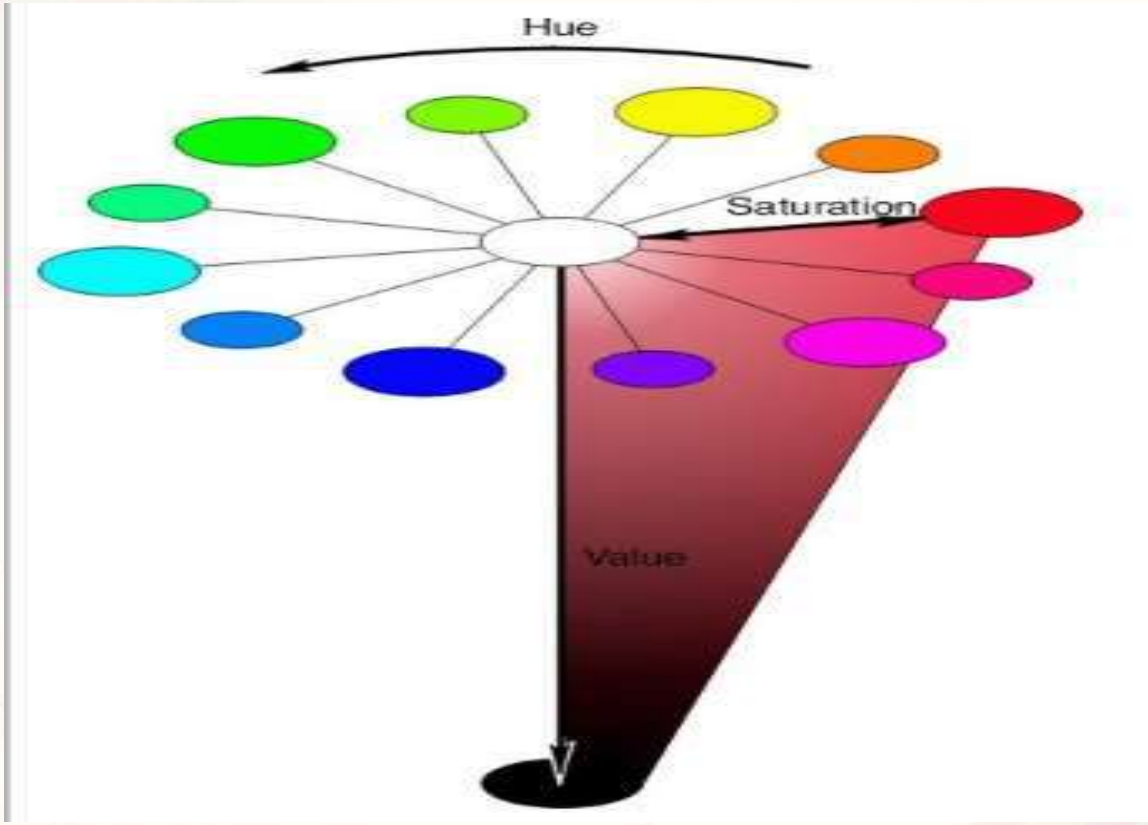
The **Hue (H)** of a color refers to which pure color it resembles. All tints, tones and shades of red have the same hue. (simply the color we see)





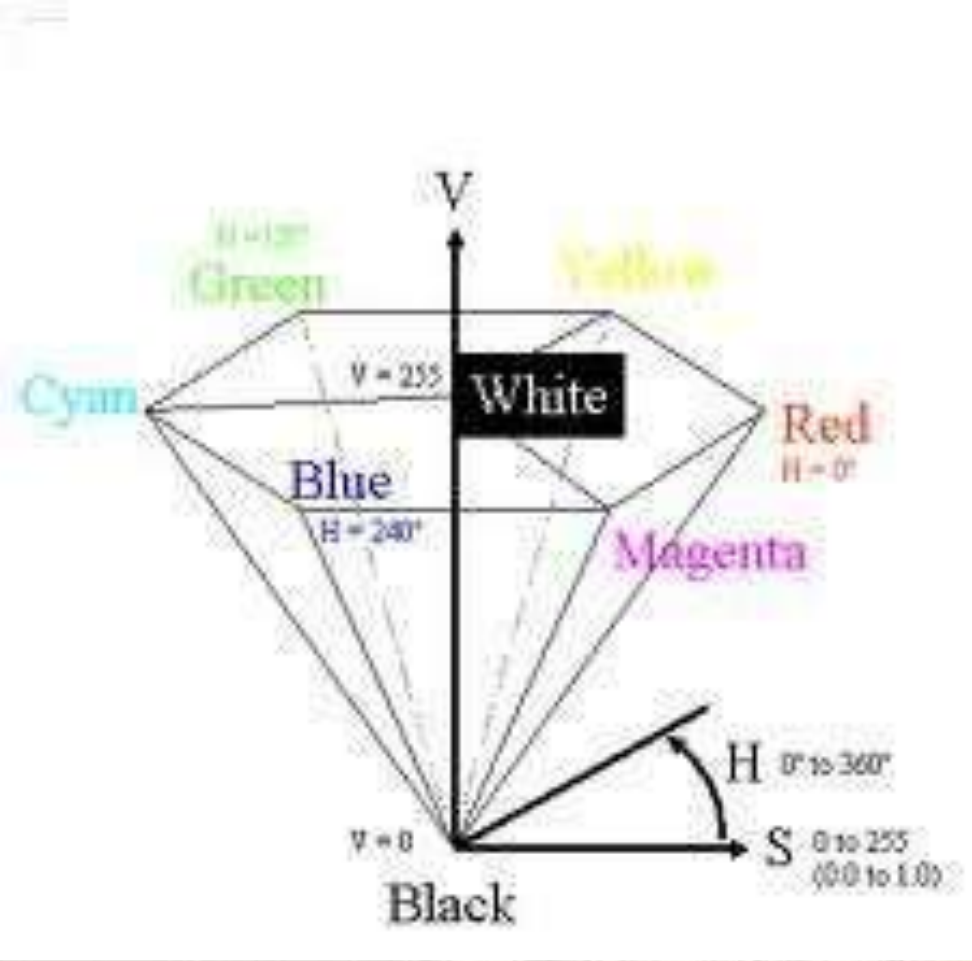
HSV Model

The **Saturation (S)** of a color describes how white the color is. Or the amount of white added to the color. A pure red is fully saturated (S=1) means no white added



HSV Model

The **Value (V)** of a color, also called its lightness, describes how dark the color is. A value of 0 is black, with increasing lightness moving away from black.

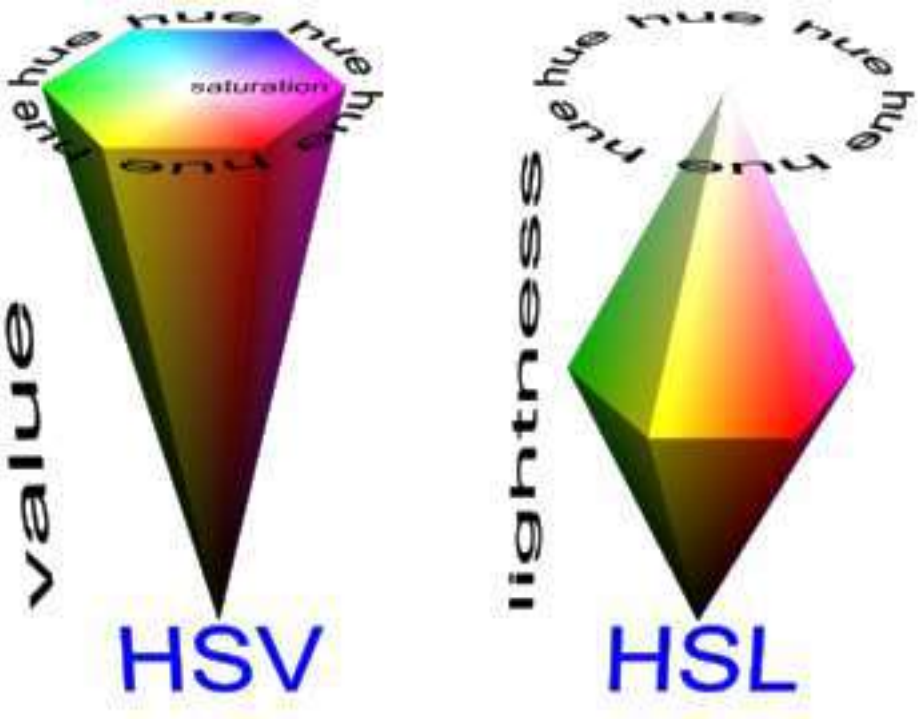


HSL Model

- Double-cone Representation

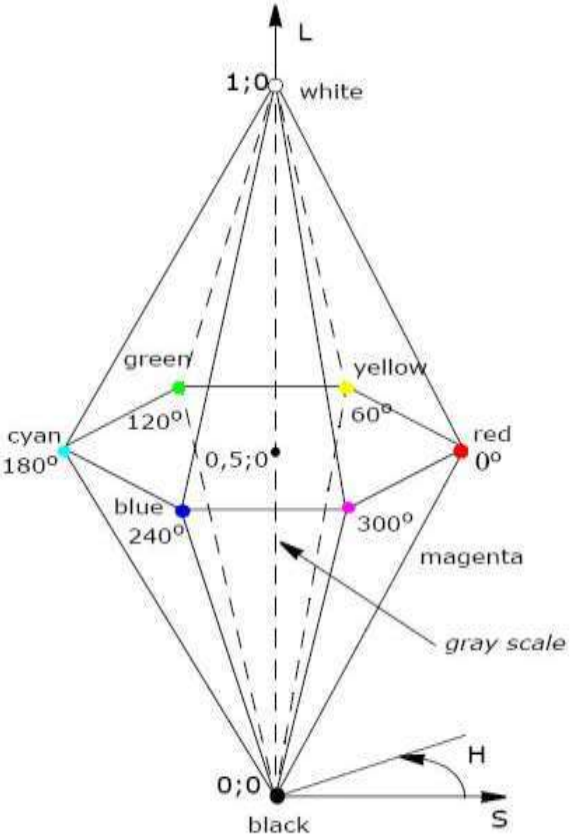
Parameters are :

- Hue (H)
- Lightness (L)
- Saturation (S)



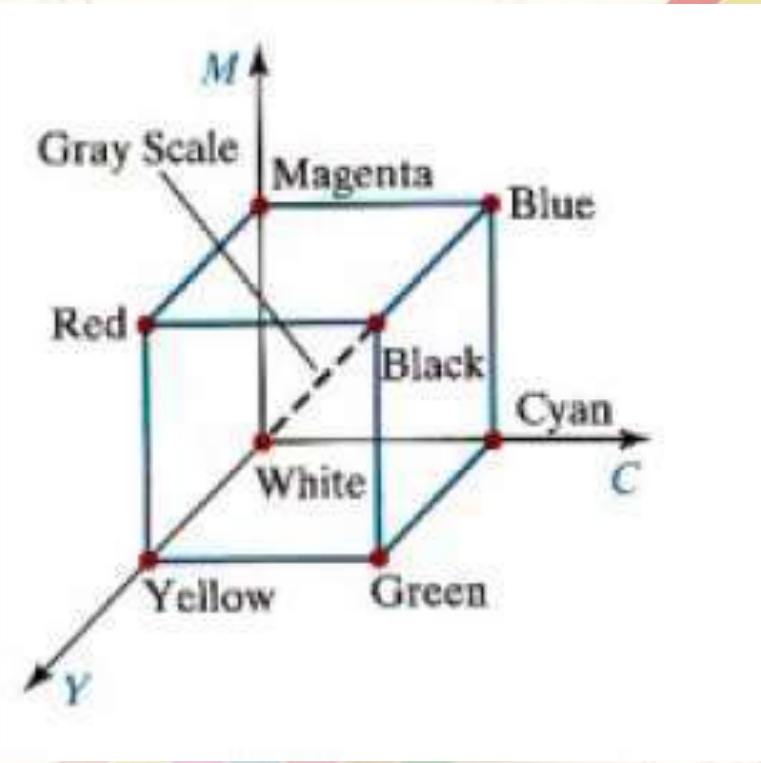
HSL Model

- Vertical Axis is called Lightness(L).
- At L=0 we have black , and at L=1 we have white
- Grayscale values are along the L axis
- The pure colors lie at the axis where L=0.5 and S=1.0



CMY and CMYK Model

- Subtractive Color Model.
- Stands for cyan-magenta-yellow.
- Used for hardcopy devices (ex. Printers).
- A printed color that looks red absorbs the other two components G and B and reflects R.
- Thus the C-M-Y coordinates are just the complements of the R-G-B coordinates.



CMY and CMYK Model

In additive color models such as RGB, white is the “additive” combination of all primary colored lights, while black is the absence of light.

In the CMYK model, it is the opposite: white is the natural color of the paper or other background, while black results from a full combination of colored inks.

RGB To CMY

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

CMY To RGB

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

CMY and CMYK Model

CMYK Color Model

Although cyan, magenta and yellow inks might be expected be sufficient for color printing, most actual color printing uses black ink in addition.

This is partly because a mixture of the first three inks may not yield a black that is neutral enough, or dark enough, but also because the use of black spares the use of the more expensive colored inks, and also reduces the total amount of ink used, thus speeding drying times.

K used instead of equal amounts of CMY

- called under color removal
- richer black
- less ink deposited on paper – dries more quickly
- First approximation – nonlinearities must be accommodated:

$K = \min(C, M, Y)$

$C' = C - K$

$M' = M - K$

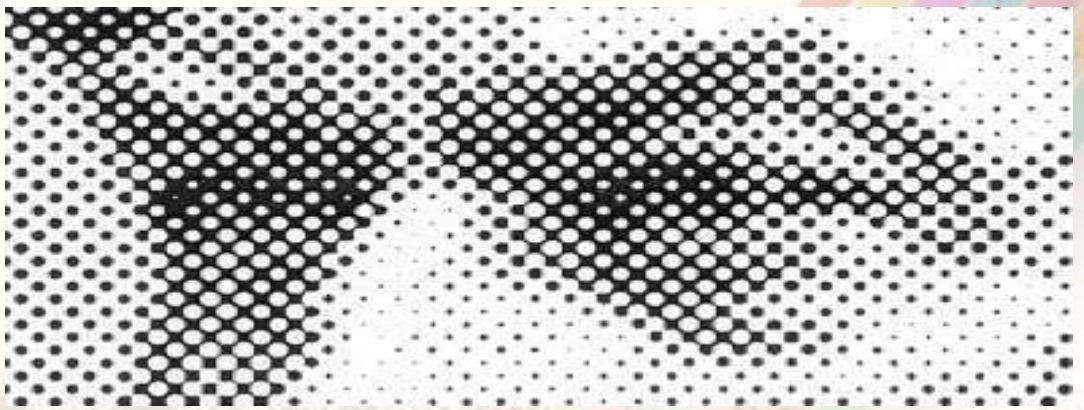
$Y' = Y - K$

Color Models Applications

Color Model Application Area	Color Model Application Area
RGB	- Computer graphics - Image processing - Image Analysis - Image Storage
CMY(K)	Printing
HSV, HSL	- Human visual perception - Computer graphics processing - Computer Vision - Image Analysis - Design image - Human vision - Image editing software - Video editor
YIQ	- TV broadcasting - Video system

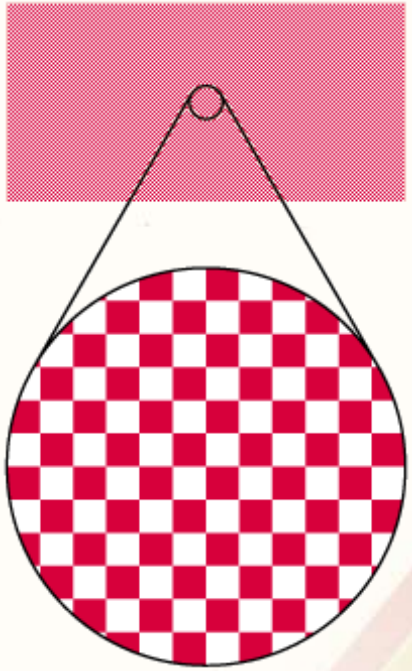
Halftone

- A technique used in newspaper printing
  Only two intensities are possible, blob of ink and no blob of ink. But, the size of the blob can be varied



Dithering

The process of approximating colors you don't have by mixing colors you do have.



Halt-Toning Vs. Dithering

Half toning is the reproduction of grayscale images using dots but with varying size.

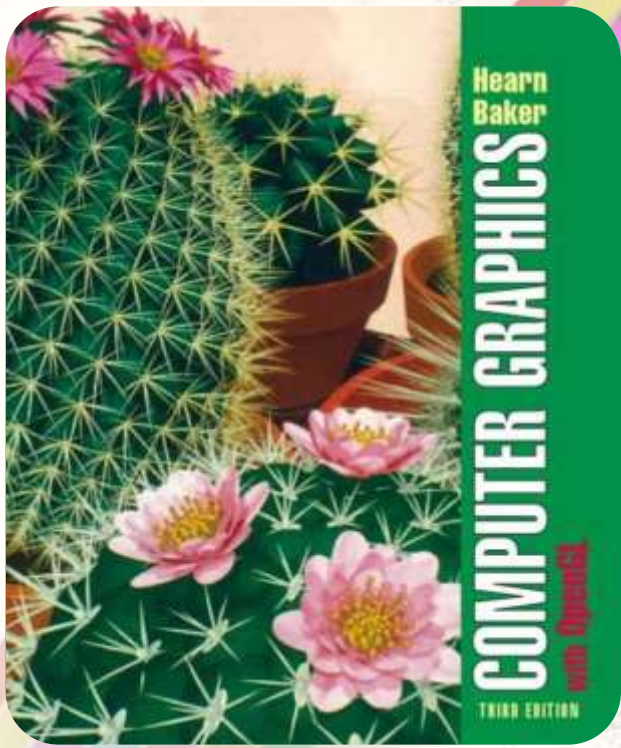
Typical Application Laser printer.

If a monitor can't show a certain color, dithering approximates the color by placing close together pixels in colors that the computer can display.

Typical Application Web graphic designers often limit their images to 256 colors and use dithering to imply other colors.

References

- Hearn, Baker and Carithers, Computer Graphics with OpenGL
- https://www.siggraph.org/education/materials/HyperGraph/color/colorcm.htm
- www.wikipedia.com
- http://www.cs.sun.ac.za/~lvzijl/courses/rw778/grafika/OpenGLtuts/Big/graphicsnotes008.html



Q&A

You have Questions

We have Answers


Thank You!

Content - Fourth Year

Computer Graphics

UNIT – III

Attributes of Output Primitives

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.

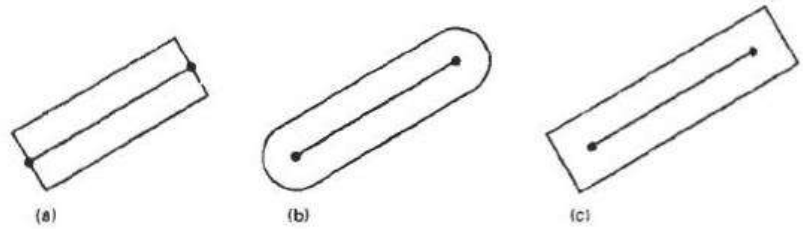
There are three types of line cap. They are

* 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.

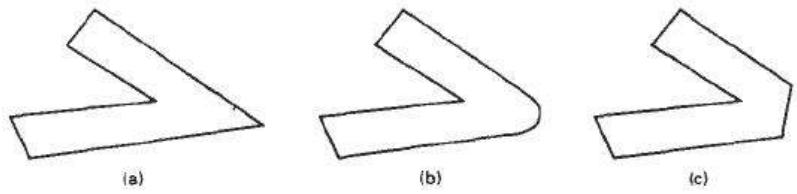
Three possible methods for smoothly joining two line segments

* 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.

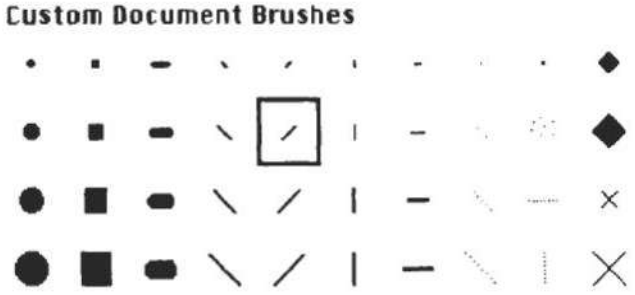
A bevel join is generated by displaying the line segment with but caps and filling in tri angular gap where the segments meet.



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



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

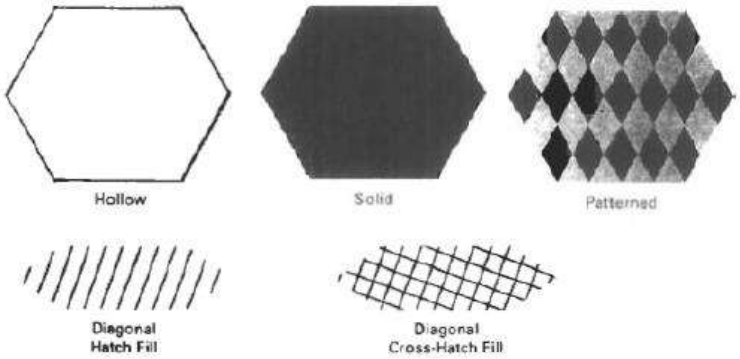
Options for filling a defined region include a choice between a solid color or a pattern fill and choices for particular colors and patterns

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

setInteriorStyle (fs)

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



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

setInteriorColourIndex (fc)

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 05

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

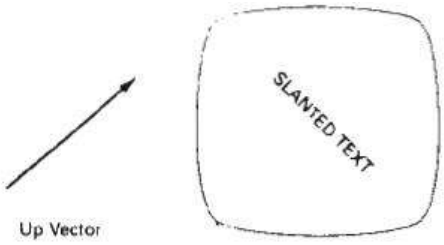
Spacing 0.5

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.



To arrange character strings vertically or horizontally

setTextPath (tp)

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

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

setTextPrecision (tpr)

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

setMarkerType (mt)

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

setMarkerSizeScaleFactor (ms)

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

setPolymarkerColourIndex (mc)

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

setIndividualASF(attributeptr, flagptr)

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

setPolylineRepresentation (ws, li, lt, lw, lc)

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

setPolylineRepresentation (1, 3, 2, 0.5, 1)

setPolylineRepresentation (4, 3, 1, 1, 7)

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

setInteriorRepresentation (ws, fi, fs, pi, fc)

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

setTextRepresentation (ws, ti, tf, tp, te, ts, tc)

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.

SUMMARY OF ATTRIBUTES

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

Two Dimensional Geometric Transformations

Changes in orientations, size and shape are accomplished with geometric transformations that alter the coordinate description of objects.

Basic transformation

* Translation
  > T(tx, ty)
  > Translation distances
* Scale
  > S(sx,sy)
  > Scale factors
* Rotation
  > R( )
  > Rotation angle

Translation

A translation is applied to an object by representing it along a straight line path from one coordinate location to another adding translation distances, tx, ty to original coordinate position (x,y) to move the point to a new position (x',y') to

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

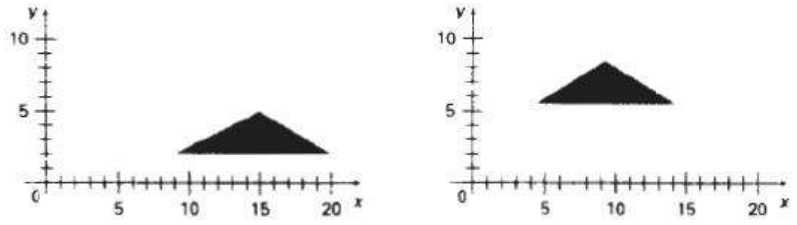
The translation distance point (tx,ty) is called translation vector or shift vector.

Translation equation can be expressed as single matrix equation by using column vectors to represent the coordinate position and the translation vector as

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

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

$$P' = P + T$$

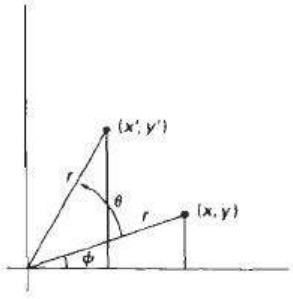


Moving a polygon from one position to another position with the translation vector (-5.5, 3.75)

Rotations:

A two-dimensional rotation is applied to an object by repositioning it along a circular path on xy plane. To generate a rotation, specify a rotation angle θ and the position (x_r, y_r) of the rotation point (pivot point) about which the object is to be rotated.

Positive values for the rotation angle define counter clock wise rotation about pivot point. Negative value of angle rotate objects in clock wise direction. The transformation can also be described as a rotation about a rotation axis perpendicular to xy plane and passes through pivot point



Rotation of a point from position (x, y) to position (x', y') through angle θ relative to coordinate origin

The transformation equations for rotation of a point position P when the pivot point is at coordinate origin. In figure r is constant distance of the point positions Φ is the original angular of the point from horizontal and θ is the rotation angle.

The transformed coordinates in terms of angle θ and Φ

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

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

The original coordinates of the point in polar coordinates

$$x = r\cos\Phi,\ y = r\sin\Phi$$

the transformation equation for rotating a point at position (x,y) through an angle θ about origin

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

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

Rotation Equation

$$P' = R \cdot P$$

Rotation Matrix

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

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

Note: Positive values for the rotation angle define counterclockwise rotations about the rotation point and negative values rotate objects in the clockwise.

Scaling

A scaling transformation alters the size of an object. This operation can be carried out for polygons by multiplying the coordinate values (x, y) to each vertex by scaling factor Sx & Sy to produce the transformed coordinates (x', y')

$$x' = x.Sx \qquad y' = y.Sy$$

scaling factor Sx scales object in x direction while Sy scales in y direction.

The transformation equation in matrix form

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

(or)

P′ = S. P

Where S is 2 by 2 scaling matrix

Turning a square (a) Into a rectangle (b) with scaling factors sx = 2 and sy = 1.

Any positive numeric values are valid for scaling factors sx and sy. Values less than 1 reduce the size of the objects and values greater than 1 produce an enlarged object.

There are two types of Scaling. They are

* Uniform scaling
* Non Uniform Scaling

To get uniform scaling it is necessary to assign same value for sx and sy. Unequal values for sx and sy result in a non uniform scaling.

Matrix Representation and Homogeneous Coordinates

Many graphics applications involve sequences of geometric transformations. An animation, for example, might require an object to be translated and rotated at each increment of the motion. In order to combine sequence of transformations we have to eliminate the matrix addition. To achieve this we have represent matrix as 3 X 3 instead of 2 X 2 introducing an additional dummy coordinate h. Here points are specified by three numbers instead of two. This coordinate system is called as Homogeneous coordinate system and it allows to express transformation equation as matrix multiplication

Cartesian coordinate position (x, y) is represented as homogeneous coordinate triple (x, y, h)

* Represent coordinates as (x, y, h)
* Actual coordinates drawn will be (x/h, y/h)

For Translation

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

$$P' = T(t_x, t_y) \cdot P$$

For Scaling

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

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

For Rotation

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

$$P' = R \cdot P$$

Composite Transformations

A composite transformation is a sequence of transformations; one followed by the other. we can set up a matrix for any sequence of transformations as a **composite transformation matrix** by calculating the matrix product of the individual transformations

Translation

If two successive translation vectors (tx1,ty1) and (tx2,ty2) are applied to a coordinate position P, the final transformed location P′ is calculated as

P′ = T(tx2, ty2). {T(tx1, ty1).P}

= {T(tx2, ty2).T(tx1,ty1)}.P

Where P and P′ are represented as homogeneous-coordinate column vectors.

$$\begin{bmatrix} 1 & 0 & tx2 \\ 0 & 1 & ty2 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & tx1 \\ 0 & 1 & ty1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_3 \\ 0 & 1 & t_3 \\ 0 & 0 & \end{bmatrix}$$

(or)

T(tx2, ty2).T(tx1,ty1) = T(tx1+tx2, ty1+ty2)

Which demonstrated the two successive translations are additive.

Rotations

Two successive rotations applied to point P produce the transformed position

P′ = R(θ2).{R(θ1).P} = {R(θ2).R(θ1)}.P

By multiplying the two rotation matrices, we can verify that two successive rotation are additive

R(θ2).R(θ1) = R(θ1 + θ2)

So that the final rotated coordinates can be calculated with the composite rotation matrix as

P′ = R(θ1 + θ2).P

$$\begin{bmatrix} \cos\theta 2 & -\sin\theta 2 \\ \sin\theta 2 & \cos\theta 2 \\ 0 & 0 \end{bmatrix}$$

Scaling

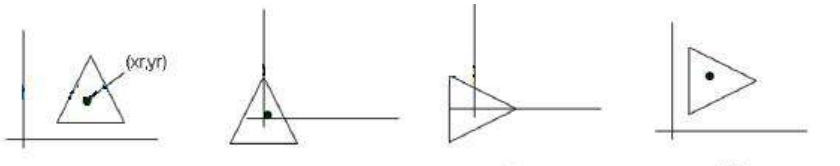
Concatenating transformation matrices for two successive scaling operations produces the following composite scaling matrix

$$\begin{bmatrix} sx2 & 0 & 0 \\ 0 & sy2 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} sx1 & 0 & 0 \\ 0 & sy1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

General Pivot-Point Rotation

1. Translate the object so that pivot-position is moved to the coordinate origin
2. Rotate the object about the coordinate origin

Translate the object so that the pivot point is returned to its original position



(a) Original Position of Object and pivot point

(b) Translation of object so that pivot point (xr,yr)is at origin

(c) Rotation was about origin

(d) Translation of the object so that the pivot point is returned to position (xr,yr)

The composite transformation matrix for this sequence is obtain with the concatenation

$$\begin{bmatrix} 1 & 0 & xr \\ 0 & 1 & yr \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -xr \\ 0 & 1 & -yr \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \cos\theta & -\sin\theta & xr(1-\cos\theta)+ yr\sin\theta \\ \sin\theta & \cos\theta & yr(1-\cos\theta) - xr\sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

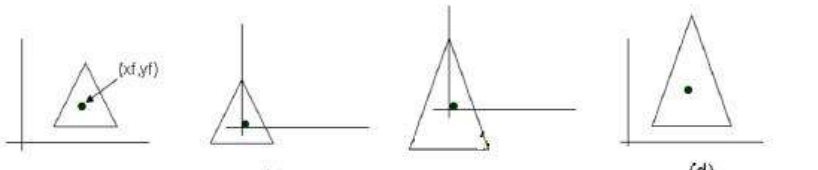
Which can also be expressed as T(xr, yr).R(θ).T(-xr, -yr) = R(xr, yr, θ)

General fixed point Scaling

1. Translate object so that the fixed point coincides with the coordinate origin

2. Scale the object with respect to the coordinate origin

Use the inverse translation of step 1 to return the object to its original position



(a) Original Position of Object and Fixed point

(b) Translation of object so that fixed point (xf,yf)is at origin

(c) scaling was about origin

(d) Translation of the object so that the Fixed point is returned to position (xf,yf)

Concatenating the matrices for these three operations produces the required scaling matix

$$\begin{bmatrix} 1 & 0 & xf \\ 0 & 1 & yf \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -xf \\ 0 & 1 & -yf \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} sx & 0 & xf(1-sx) \\ 0 & sy & yf(1-sy) \\ 0 & 0 & 1 \end{bmatrix}$$

Can also be expressed as T(xf, yf).S(sx, sy).T(-xf, -yf) = S(xf, yf, sx, sy)

Note: Transformations can be combined by matrix multiplication

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \left(\begin{bmatrix} 1 & 0 & rx \\ 0 & 1 & ry \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\Theta \\ \sin\Theta \\ 0 \end{bmatrix} \right)$$

Implementation of composite transformations

```
#include <math.h>
```

```
#include <graphics.h>
typedef float Matrix3x3 [3][3];
Matrix3x3 thematrix;
void matrix3x3SetIdentity (Matrix3x3 m)
{
int i,j;
for (i=0; i<3; i++)
for (j=0: j<3; j++ )
m[il[j] = (i == j);
}
/ * Multiplies matrix a times b, putting result in b */
void matrix3x3PreMultiply (Matrix3x3 a. Matrix3x3 b)
{
int r,c:
Matrix3x3 tmp:
for (r = 0; r < 3: r++)
for (c = 0; c < 3; c++)
tmp[r][c] =a[r][0]*b[0][c]+ a[r][1]*b[l][c] + a[r][2]*b[2][c]:
for (r = 0: r < 3: r++)
for Ic = 0; c < 3: c++)
b[r][c]=- tmp[r][c]:
}
void translate2 (int tx, int ty)
{
Matrix3x3 m:
rnatrix3x3SetIdentity (m) :
m[0][2] = tx;
m[1][2] = ty:
matrix3x3PreMultiply (m, theMatrix);
}
vold scale2 (float sx. float sy, wcPt2 refpt)
(
Matrix3x3 m.
matrix3x3SetIdentity (m);
m[0] [0] = sx;
m[0][2] = (1 - sx)* refpt.x;
m[l][l] = sy;
m[10][2] = (1 - sy)* refpt.y;
matrix3x3PreMultiply (m, theMatrix);
}
void rotate2 (float a, wcPt2 refPt)
{
Matrix3x3 m;
matrix3x3SetIdentity (m):
a = pToRadians (a);
m[0][0]= cosf (a);
m[0][1] = -sinf (a) ;
m[0] [2] = refPt.x * (1 - cosf (a)) + refPt.y sinf (a);
m[1] [0] = sinf (a);
m[l][l] = cosf (a];
```

```
m[l] [2] = refPt.y * (1 - cosf (a) - refPt.x * sinf ( a ) ;
matrix3x3PreMultiply (m, theMatrix);
}
void transformPoints2 (int npts, wcPt2 *pts)
{
int k:
float tmp ;
for (k = 0; k< npts: k++)
{
tmp = theMatrix[0][0]* pts[k] .x * theMatrix[0][1] * pts[k].y+ theMatrix[0][2];
pts[k].y = theMatrix[1][0]* pts[k] .x * theMatrix[1][1] * pts[k].y+ theMatrix[1][2];
pts[k].x =tmp;
}
}
void main (int argc, char **argv)
{
wcPt2 pts[3]= { 50.0, 50.0, 150.0, 50.0, 100.0, 150.0};
wcPt2 refPt ={100.0. 100.0};
long windowID = openGraphics (*argv,200, 350);
setbackground (WHITE) ;
setcolor (BLUE);
pFillArea(3, pts):
matrix3x3SetIdentity(theMatrix);
scale2 (0.5, 0.5, refPt):
rotate2 (90.0, refPt);
translate2 (0, 150);
transformpoints2 ( 3 , pts)
pFillArea(3.pts);
sleep (10);
closeGraphics (windowID);
}
```

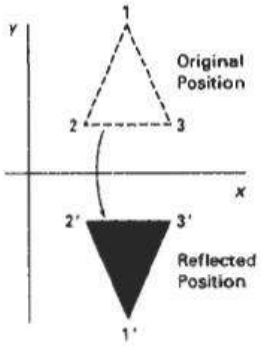
Other Transformations

1. Reflection
2. Shear

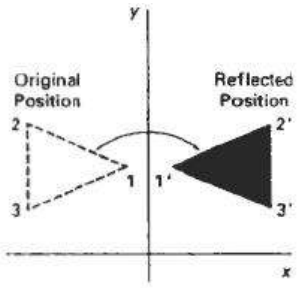
Reflection

A reflection is a transformation that produces a mirror image of an object. The mirror image for a two-dimensional reflection is generated relative to an axis of reflection by We can choose an axis of reflection in the xy plane or perpendicular to the xy plane or coordinate origin

Reflection of an object about the x axis



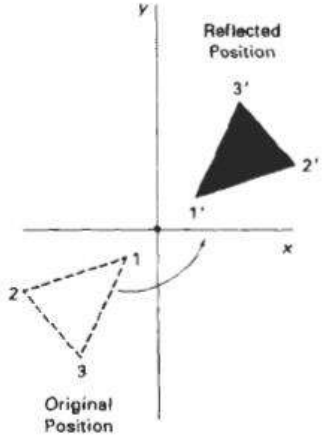
Reflection the x axis is accomplished with the transformation matrix



Reflection the y axis is accomplished with the transformation matrix

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

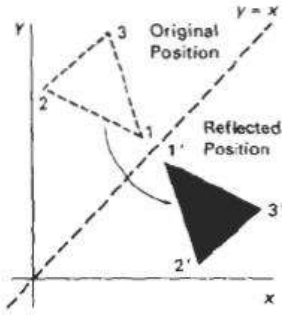
Reflection of an object about the coordinate origin



Reflection about origin is accomplished with the transformation matrix

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

Reflection axis as the diagonal line y = x



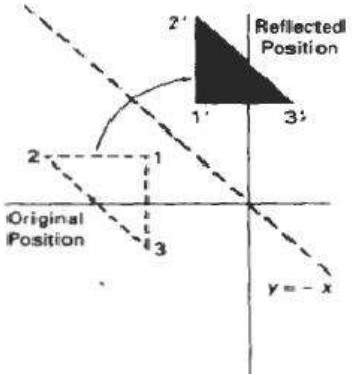
To obtain transformation matrix for reflection about diagonal y=x the transformation sequence is

1. Clock wise rotation by 45^0
2. Reflection about x axis
3. Counter clock wise by 45^0

Reflection about the diagonal line y = x is accomplished with the transformation matrix

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

Reflection axis as the diagonal line y = -x



To obtain transformation matrix for reflection about diagonal y = -x the transformation sequence is

1. Clock wise rotation by 45^0
2. Reflection about y axis
3. counter clock wise by 45^0

Reflection about the diagonal line y = -x is accomplished with the transformation matrix

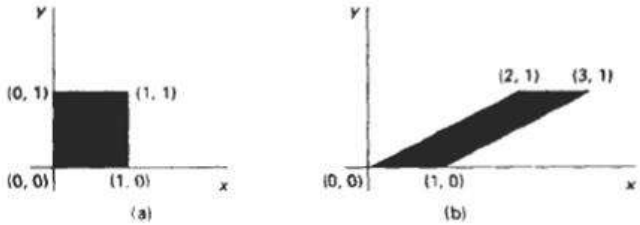
$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Shear

A Transformation that slants the shape of an object is called the shear transformation. Two common shearing transformations are used. One shifts x coordinate values and other shift y coordinate values. However in both the cases only one coordinate (x or y) changes its coordinates and other preserves its values.

X - Shear

The x shear preserves the y coordinates, but changes the x values which cause vertical lines to tilt right or left as shown in figure



The Transformations matrix for x-shear is

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

which transforms the coordinates as

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

$$y' = y$$

Y - Shear

The y shear preserves the x coordinates, but changes the y values which cause horizontal lines which slope up or down

The Transformations matrix for y-shear is

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

which transforms the coordinates as

$$x' = x$$

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

XY - Shear

The transformation matrix for xy-shear

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & sh_x & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

which transforms the coordinates as

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

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

Shearing Relative to other reference line

We can apply x shear and y shear transformations relative to other reference lines. In x shear transformations we can use y reference line and in y shear we can use x reference line.

X - shear with y reference line

We can generate x-direction shears relative to other reference lines with the transformation matrix

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

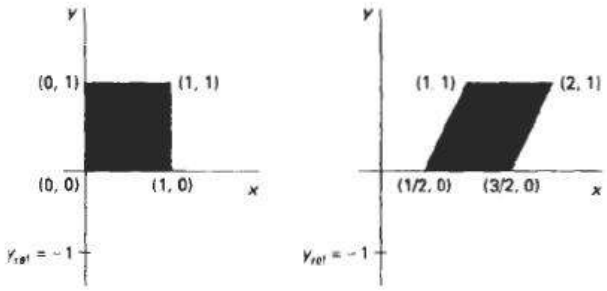
which transforms the coordinates as

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

$$y' = y$$

Example

$$Sh_x = ½ \text{ and } y_{ref} = -1$$



Y - shear with x reference line

We can generate y-direction shears relative to other reference lines with the transformation matrix

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

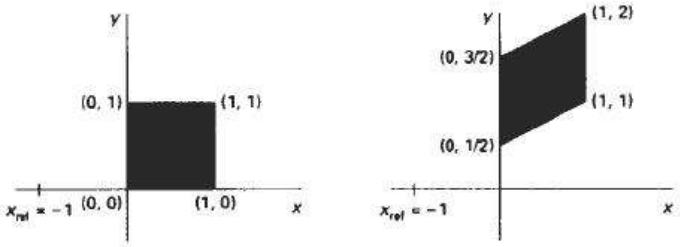
which transforms the coordinates as

$$x' = x$$

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

Example

$$Sh_y = ½ \text{ and } x_{ref} = -1$$



::Back::

LIANG BARSKY LINE CLIPPING:

A point is considered to be within a rectangle, if

$$xw_{min} \le x_1 + u\Delta x \le xw_{max}$$

$$yw_{min} \le y_1 + u\Delta y \le yw_{max}$$

Four Inequalities can be expressed as

$$up_k \le q_k \quad , \quad k = 1, 2, 3, 4$$

where parameters p and q are defined as

$$xw_{min} \le x_1 + u\Delta x$$

$$xw_{min} - x_1 \le u\Delta x$$

(ii) $$-u\Delta x \le x_1 - xw_{min}$$

$$\therefore p_1 = -\Delta x \quad q_1 = x_1 - xw_{min}$$

$$x_1 + u\Delta x \le xw_{max}$$

$$u\Delta x \le xw_{max} - x_1$$

Compare with $up_k \le q_k$

$$\therefore p_2 = \Delta x \quad q_2 = xw_{max} - x_1$$

Similarly

$$p_3 = -\Delta y \quad q_3 = y_1 - yw_{min}$$

$$p_4 = \Delta y \quad q_4 = yw_{max} - y_1$$

(i.e)

$$p_1 = -\Delta x \quad q_1 = x_1 - xw_{min}$$

$$p_2 = \Delta x \quad q_2 = xw_{max} - x_1$$

$$p_3 = -\Delta y \quad q_3 = y_1 - yw_{min}$$

$$p_4 = \Delta y \quad q_4 = yw_{max} - y_1$$

Based on these four inequalities, we can find the following conditions of line clipping:

(i) If $P_k = 0$, the line is parallel to the corresponding clipping boundary.

$k = 1 \rightarrow$ Left $k = 3 \rightarrow$ Bottom

$k = 2 \rightarrow$ Right $k = 4 \rightarrow$ TOP.

(ii) If for any k, for which $P_k = 0$:

* $q_k < 0$, the line is completely outside the boundary

* $q_k \geq 0$, Line is inside the parallel clipping boundary.

(iii) If $P_k < 0$, the line proceeds from the outside to the inside of the particular clipping boundary.

(iv) If $P_k > 0$, the line proceeds from inside to outside of the particular clipping boundary.

In Both (iii) and (iv) cases, the intersection parameter is calculated as

$$u = q_k / P_k$$

Example: Find the clipping coordinates for a line P_1P_2 where $P_1 = (10,10)$ and $P_2 = (60,30)$ against window with $(xw_{min}, yw_{min}) = (15,15)$ and $(xw_{max}, yw_{max}) = (25,25)$.

$x_1 = 10$ $y_1 = 10$ $xw_{min} = 15$ $xw_{max} = 25$

$x_2 = 60$ $y_2 = 30$ $yw_{min} = 15$ $yw_{max} = 25$

$p_1 = -\Delta x$

$= -(x_2 - x_1)$

$= -(60-10)$

$\boxed{p_1 = -50}$

$q_1 = x_1 - xw_{min}$

$= 10 - 15$

$= -5$

$\boxed{q_1 = -5}$

$p_2 = \Delta x$

$\boxed{p_2 = 50}$

$q_2 = xw_{max} - x_1$

$= 25 - 10 = 15$

$\boxed{q_2 = 15}$

$p_3 = -\Delta y$

$= -(y_2 - y_1)$

$= -(30-10)$

$\boxed{p_3 = -20}$

$q_3 = y_1 - yw_{min}$

$= 10 - 15$

$= -5$

$\boxed{q_3 = -5}$

$p_4 = \Delta y$

$\boxed{p_4 = 20}$

$q_4 = yw_{max} - y_1$

$= 25 - 10$

$\boxed{q_4 = 15}$

* None of the P_k value is $= 0$.

* P_1 and $P_3 < 0$ line proceeds from outside to inside

P_2 and $P_4 > 0$ line proceeds from inside to outside.

* Find Intersection parameters $u = \frac{q_k}{p_k}$, $k = 1, 2, 3, 4$

$$\frac{q_1}{p_1} = \frac{-5}{-50} = 0.1 \qquad \frac{q_3}{p_3} = \frac{-5}{-20} = 0.25$$

$P_k < 0$

$$\frac{q_2}{p_2} = \frac{15}{50} = 0.3 \qquad \frac{q_4}{p_4} = \frac{15}{20} = 0.75$$

$P_k > 0$

If $P_k < 0$ (P_1 and P_3) update u_1 as:

$$u_1 = \max\left[0, (q_k/p_k)\right], \quad k = 1 \text{ to } 4$$

$$\therefore u_1 = \max\left[0, \frac{q_1}{p_1}, \frac{q_3}{p_3}\right] = \max[0, 0.1, 0.25]$$

$$\therefore \boxed{u_1 = 0.25}$$

If $P_k > 0$ (P_2 and P_4) update u_2 as:

$$u_2 = \min\left[1, (q_k/p_k)\right], \quad k = 1 \text{ to } 4$$

$$u_2 = \min\left[1, \frac{q_2}{p_2}, \frac{q_4}{p_4}\right] = \min[1, 0.3, 0.75]$$

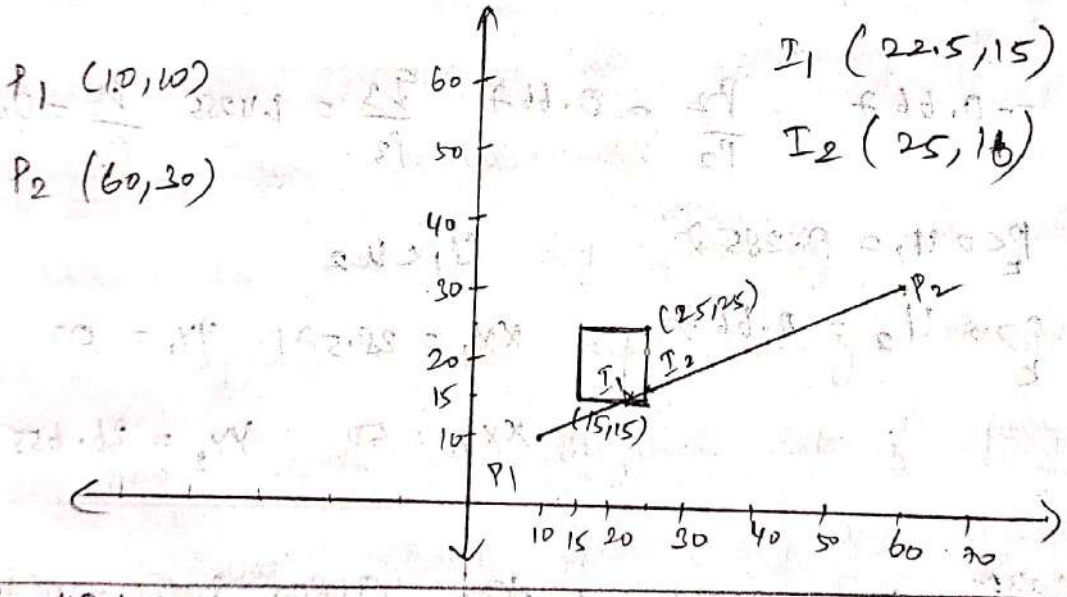
$$\boxed{u_2 = 0.3}$$

* Check condition if $u_1 < u_2$ then the endpoints of the clipped line are:

$$I_1 \begin{cases} xx_1 = x_1 + u_1 \Delta x = 10 + 0.25(50) = 22.5 \\ yy_1 = y_1 + u_1 \Delta y = 10 + 0.25(20) = 15 \end{cases}$$

$$I_2 \begin{cases} xx_2 = x_1 + u_2 \Delta x = 10 + 0.3 \times 50 = 25 \\ yy_2 = y_1 + u_2 \Delta y = 10 + 0.3(20) = 16. \end{cases}$$

* If $u_1 > u_2$ (Reject the line).



Algorithm:

(i) Find p_k and q_k, $k = 1$ to 4

(ii) check If $p_k = 0$, then check $q_k < 0$ for which $k = 1, 2, 3, 4$.
(Reject)
$q_k \geq 0$ Accept

(iii) Check if $p_k < 0$ or $p_k > 0$. Calculate Intersection parameters $u = \frac{q_k}{p_k}$, $k = 1$ to 4

If $p_k < 0$ $u_1 = \max(0, q_k/p_k)$ for which $p_k < 0$

$p_k > 0$ $u_2 = \min(1, q_k/p_k)$ for which $p_k > 0$

(iv) check $u_1 > u_2$ (reject)

If $u_1 < u_2$ then

$xx_1 = x_1 + u_1 \Delta x$ $\quad yy_1 = y_1 + u_1 \Delta y$

$xx_2 = x_1 + u_2 \Delta x$ $\quad yy_2 = y_1 + u_2 \Delta y$

Example :- Apply the Liang - Barsky algorithm to the line pair with coordinates (30, 60) and (60, 25) against the window $(x_{min}, y_{min}) = (10, 10)$ and $(x_{max}, y_{max}) = (50, 50)$.

$p_1 = -80$ $\quad p_2 = 80$ $\quad q_1 = 20$ $\quad q_3 = 50$

$p_3 = 35$ $\quad p_4 = -35$ $\quad q_2 = 20$ $\quad q_4 = -10$

$\frac{q_1}{p_1} = -0.667$ $\quad \frac{p_2}{p_2} = 0.667$ $\quad \frac{q_3}{p_3} = 1.4285$ $\quad \frac{p_4}{p_4} = 0.2857$

$p_k < 0$ $u_1 = 0.2857$ $\quad u_1 < u_2$

$p_k > 0$ $u_2 = 0.667$ $\quad xx_1 = 38.571$ $\quad yy_1 = 50$

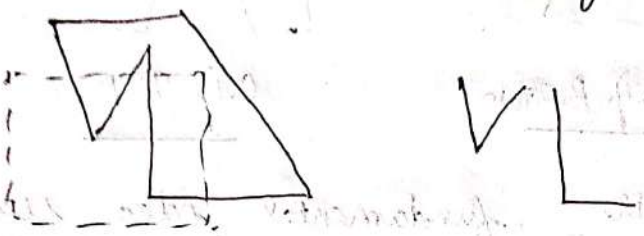
$xx_2 = 50$ $\quad yy_2 = 36.655$

Advantages:-

- More efficient than Cohen Sutherland Line clipping since intersection calculations are reduced.
- It requires only one division to update parameters u_1 and u_2.
- window intersections of the line are computed only once, when the final values of u_1 and u_2 have been computed.
- Can be extended to 3D clipping.

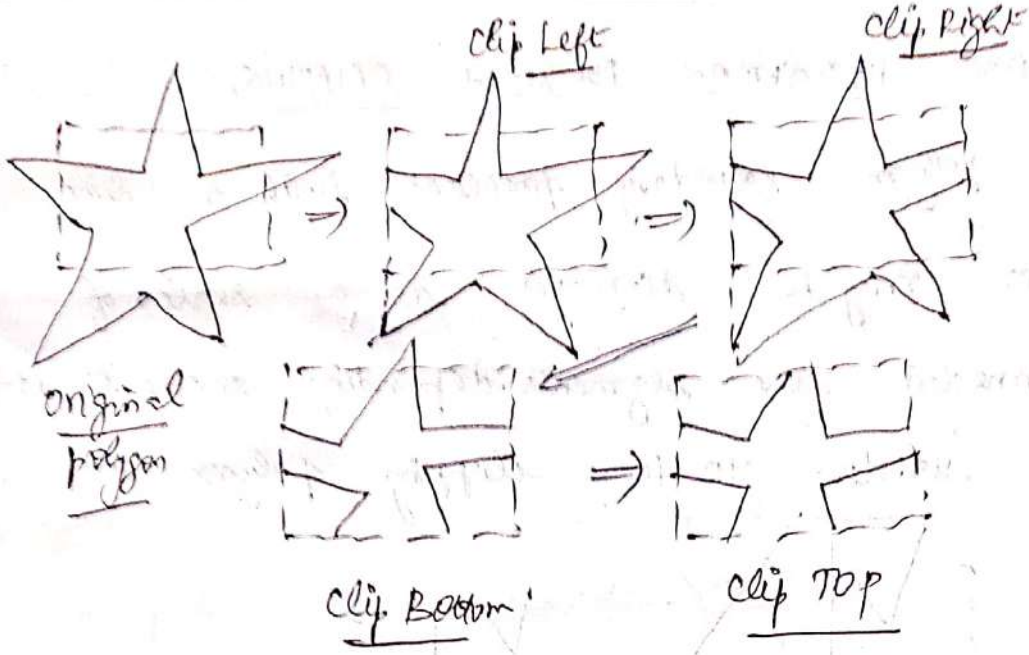
SUTHERLAND HODGEMAN POLYGON CLIPPING:

* A polygon boundary processed with a line clipper may be displayed as a series of unconnected line segments depending on the orientation of the window to the clipping polygon.



Before clipping After clipping.

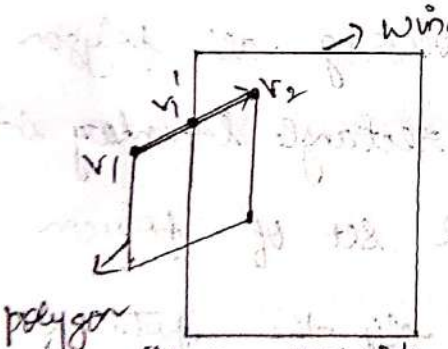
* In polygon clipping, a polygon can be clipped by processing its boundary as a whole against each window edge.
* This is achieved by processing all polygon vertices against each clip rectangle boundary in turn.
* Beginning with the original set of polygon vertices, we could first clip the polygon against the left rectangle boundary to produce a new sequence of vertices.
* The new set of vertices could then be successively passed to a right boundary clipper, a Bottom Boundary clipper and a top boundary clipper. At each step a new set of polygon vertices is generated and passed to the next window boundary clipper.



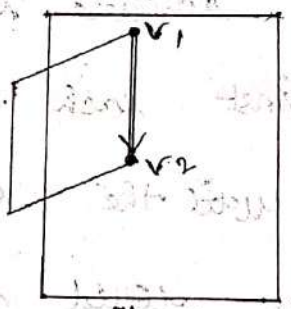
* This is the fundamental idea used in the Sutherland - Hodgeman Algorithm.

* There are 4 possible cases when processing vertices in a sequence around the perimeter of a polygon.

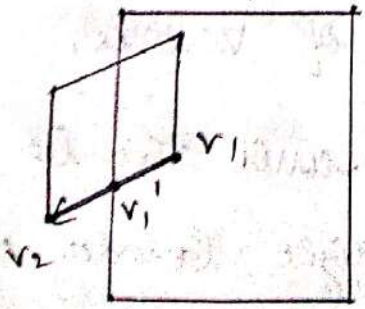
Processing of edges of the polygon against left window boundary.



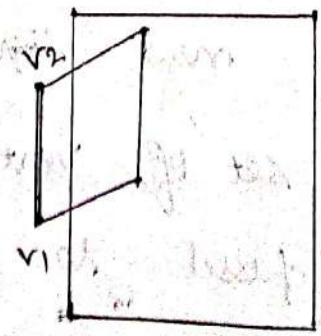
(i) v_1 outside
v_2 Inside
Save v_1' and v_2



(ii) v_1 Inside
v_2 Inside
Save v_2



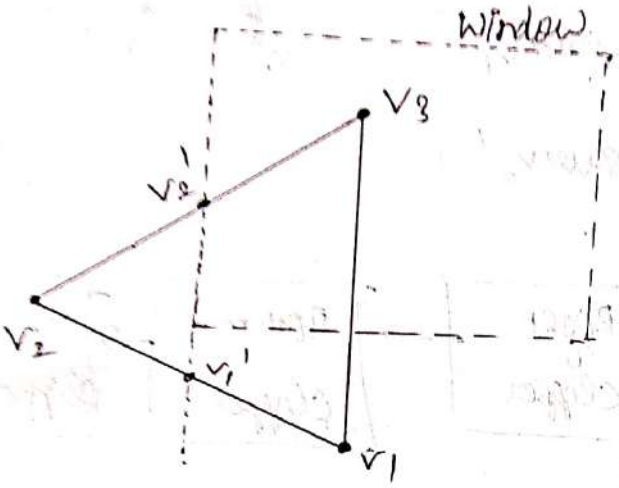
(iii) v_1 Inside
v_2 outside
Save v_1'



(iv) v_1 outside
v_2 outside
Save Nothing

Example:

Clip the polygon using Sutherland-Hodgeman polygon clipping algorithm.



Convex polygon

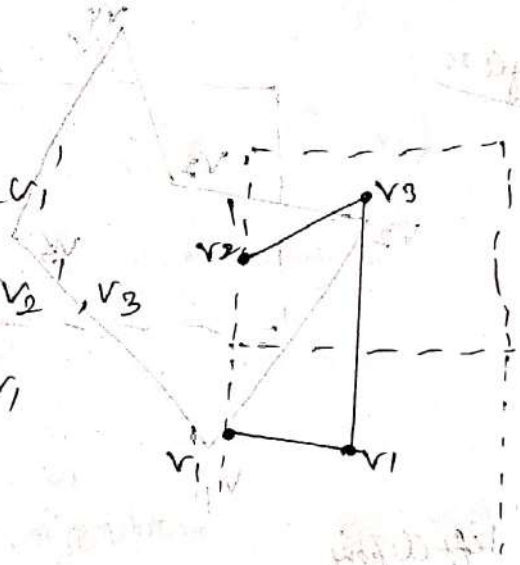
Interior angle $< 180°$

Concave polygon

One or more Interior angles $> 180°$

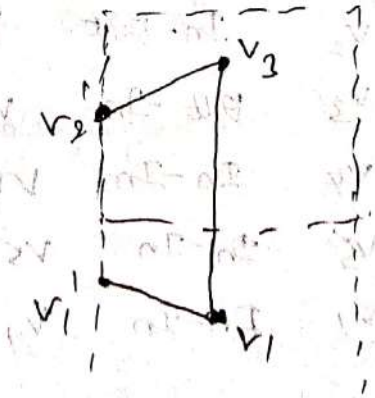
left clipping:

V_1 V_2	In – out	Save V_1'
V_2 V_3	out – In	Save V_2, V_3
V_3 V_1	In – In	Save V_1



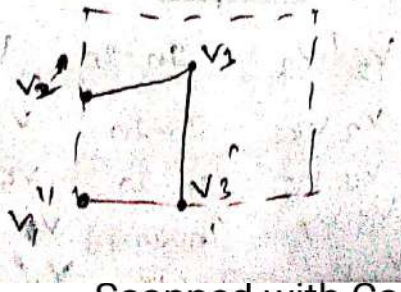
Right clipping:

V_2' V_3	In – In	Save V_3
V_3 V_1	In – In	Save V_1
V_1 V_1'	In – In	Save V_1'
V_1' V_2'	In – In	Save V_2'



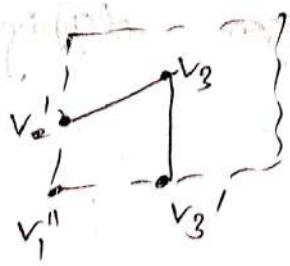
Bottom clipping

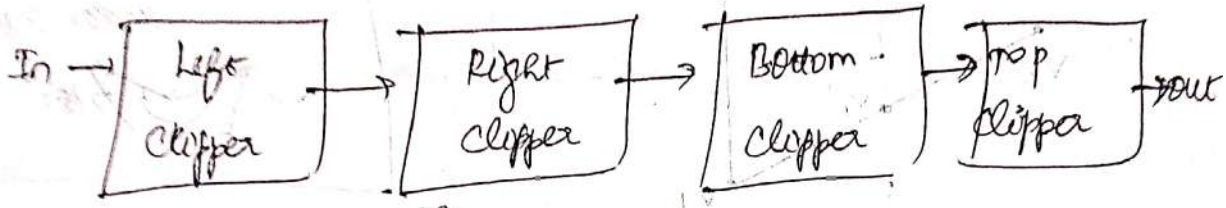
V_2' V_3	In – In	Save V_3
V_3 V_1	In – out	Save V_3'
V_1 V_1'	out – out	Save nothing
V_1' V_2'	out – In	Save V_1'', V_2'



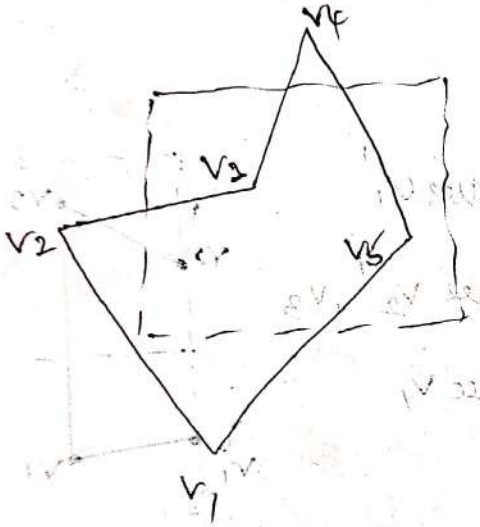
Top clipping

V_2' V_3	In-In	Save V_3
V_3 V_2'	In-In	Save V_3'
V_3' V_1''	In-In	Save V_1''
V_1'' V_2'	In-In	Save V_2'



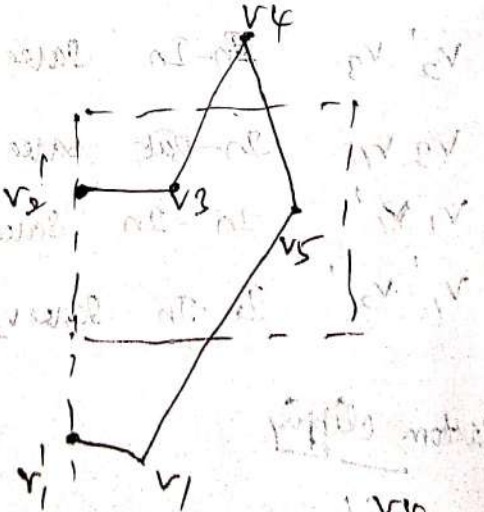


Example :



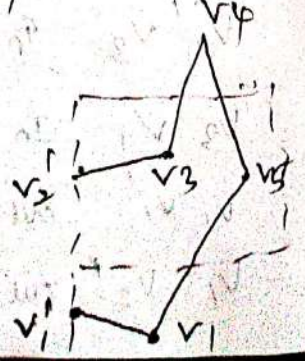
Left clipping

V_1 V_2	In-Out	V_1'
V_2 V_3	Out-In	V_2' V_3
V_3 V_4	In-In	V_4
V_4 V_5	In-In	V_5
V_5 V_1	In-In	V_1



Right clipping

V_2' V_3	In-In	V_3	
V_3 V_4	In-In	V_4	V_1' V_2' In-In V_2'
V_4 V_5	In-In	V_5	
V_5 V_1	In-In	V_1	
V_1 V_1'	In-In	V_1'	

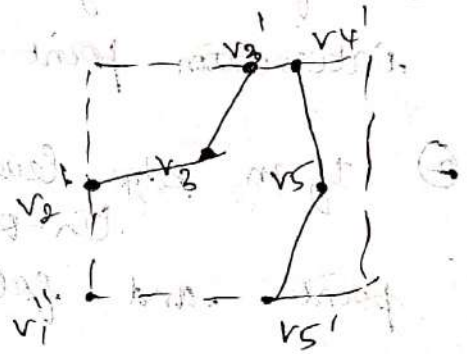


Top clipping:

$v_2'\ v_3$	In-In	v_3
$v_3\ v_4$	In-out	v_3'
$v_4\ v_5$	out-In	$v_4'\ v_5$
$v_5\ v_1$	In-In	v_1
$v_1\ v_1'$	In-In	v_1'
$v_1'\ v_2'$	In-In	v_2'

Bottom clipping

$v_2'\ v_3$	In-In	v_3
$v_3\ v_3'$	In-In	v_3'
$v_3'\ v_4'$	In-In	v_4'
$v_4'\ v_5$	In-In	v_5
$v_5\ v_1$	In-out	v_5'
$v_1\ v_1'$	out-out	(nothing)
$v_1'\ v_2'$	out-In	$v_1''\ v_2'$



- Sutherland Hodgeman algorithm clips convex polygons correctly, but in case of concave polygons, clipped polygon may be displayed with extraneous lines.

- The problem can be overcome by separating concave polygon into two or more convex polygons and processing each convex polygon separately.

WEILER ATHERTON POLYGON CLIPPING:-

* To display Concave polygons correctly use Weiler Atherton polygon clipping algorithm.

* Select any arbitrary vertex of the polygon and trace boundary of polygon in clockwise direction until an intersection with the window occurs.

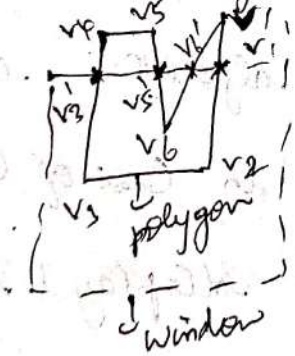
Cases: (Out - In)

① If an edge enters the window, record & save intersection point and follow the polygon boundary.

② If an edge leaves the window, save intersection (In - Out) point and follow the window boundary.

* When all the edges of the polygon are processed only once, the algorithm stops.

Example:



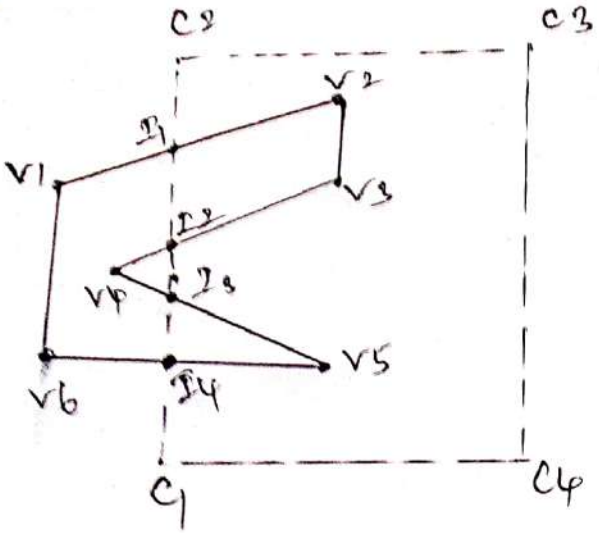
① Start arbitrarily at V_1' check for intersection, (V_1, V_2) out - In V_1' Continue following polygon boundary

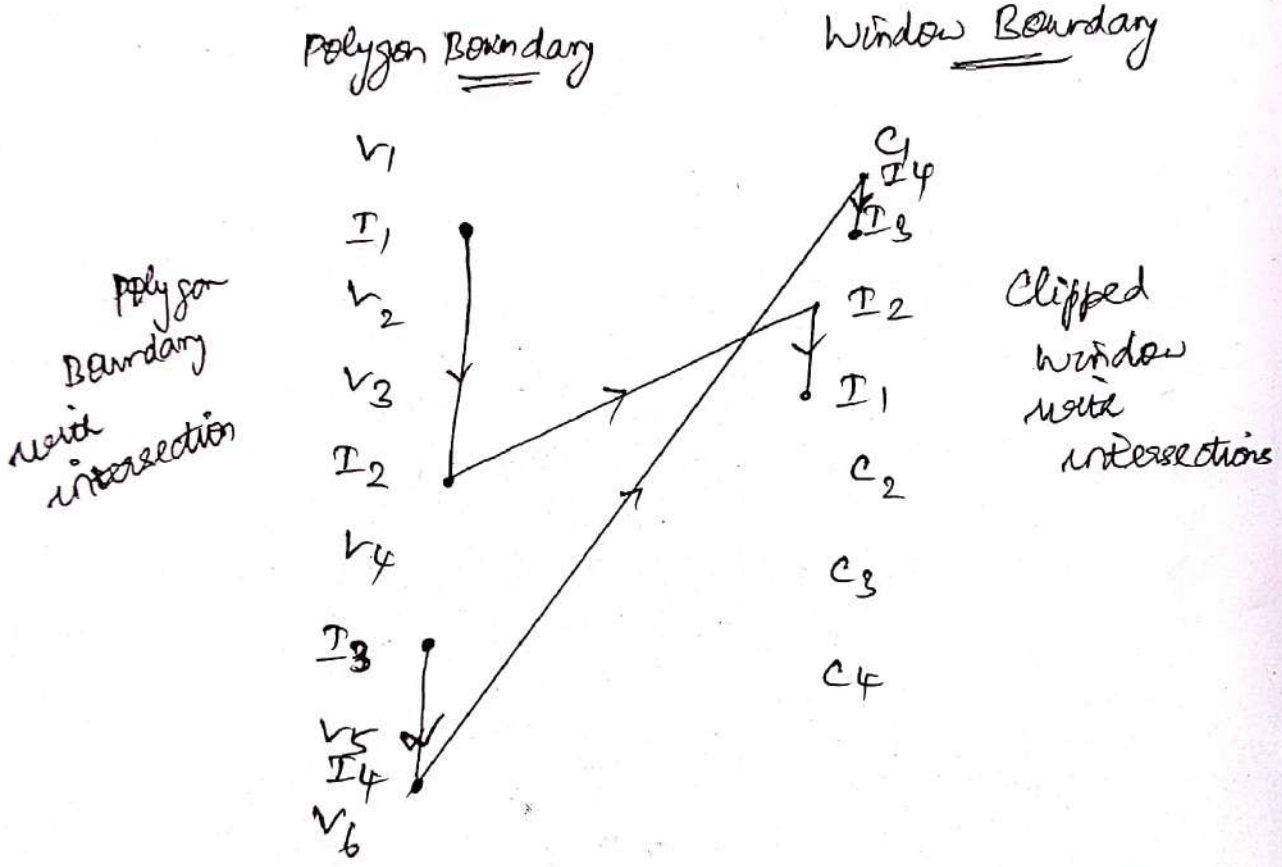
(V_2, V_3) In - In (No Intersection)

$[V_3, V_4)$ In - out follow window boundary, save V_3' (Intersection occurs)

(V_3', V_5') Intersection occurs save V_5' (window becomes polygon polygon becomes window)

Example



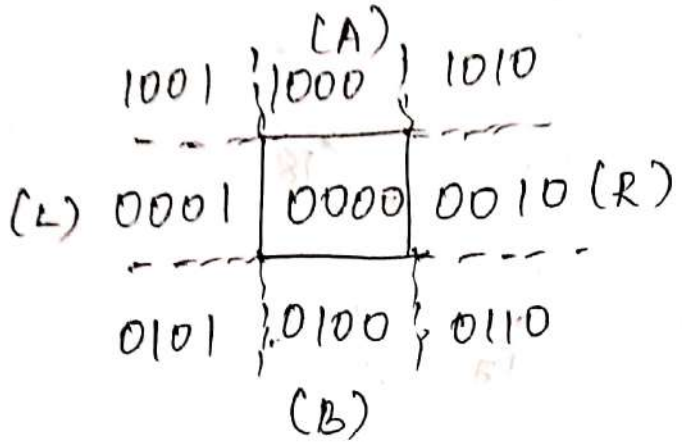


(i) Start with vertex V_1, find the intersection point

(ii) Find the edge that enters the clipping window. In this (V_1 to V_2) I_1 enters, proceed till where an edge leaves clipping window. (Intersection point)

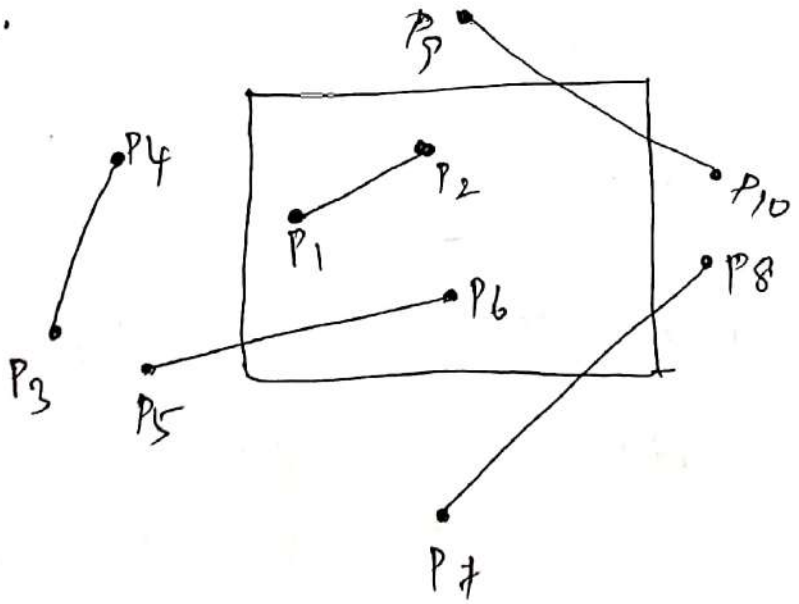
COHEN SUTHERLAND LINE CLIPPING:-

4 Bit codes → Region Codes or Outcodes

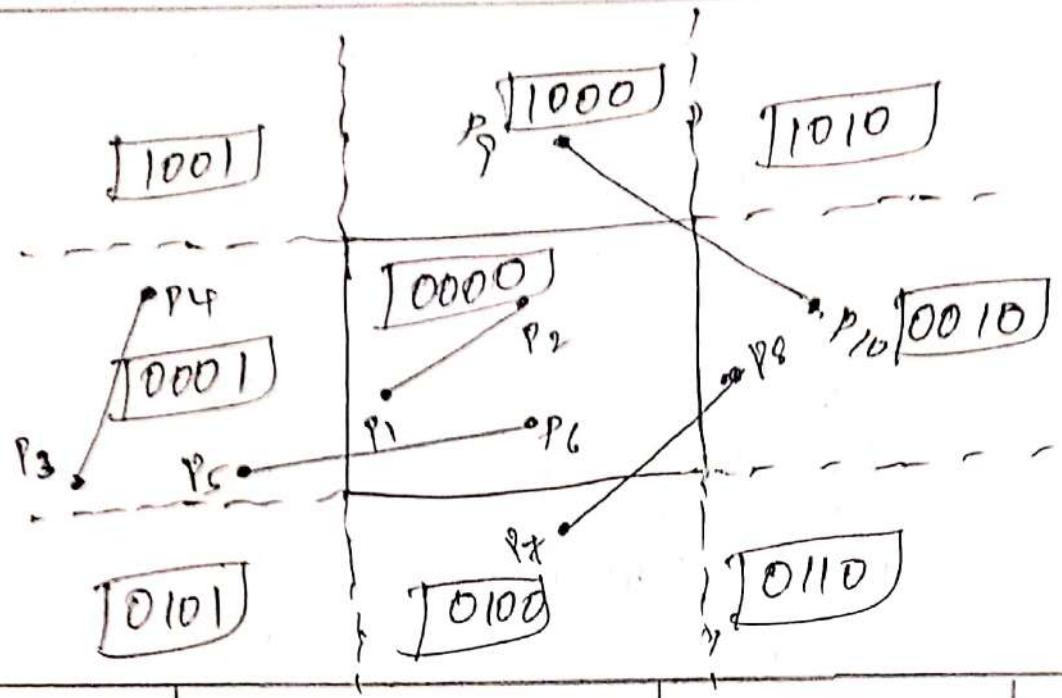


Bit 1 → Left
Bit 2 → Right
Bit 3 → Below
Bit 4 → Above

Example:- Consider the clipping window and the lines shown in Figure. Find the region codes for each end point and identify whether the line is completely visible, partially visible or completely invisible.



① Compare the diagram with the region codes.



Line	End point codes		logical AND	Result
P1 P2	0000	0000	0000	completely visible
P3 P4	0001	0001	0001	completely Invisible
P5 P6	0001	0000	0000	partially visible
P7 P8	0100	0010	0000	partially visible
P9 P10	1000	0010	0000	Partially visible

Alg' (i) If P_1 & P_2 are 0000 → Completely visible

(ii) If P1 and P_2 are Non zero → logical AND → Non-zero → Completely visible.

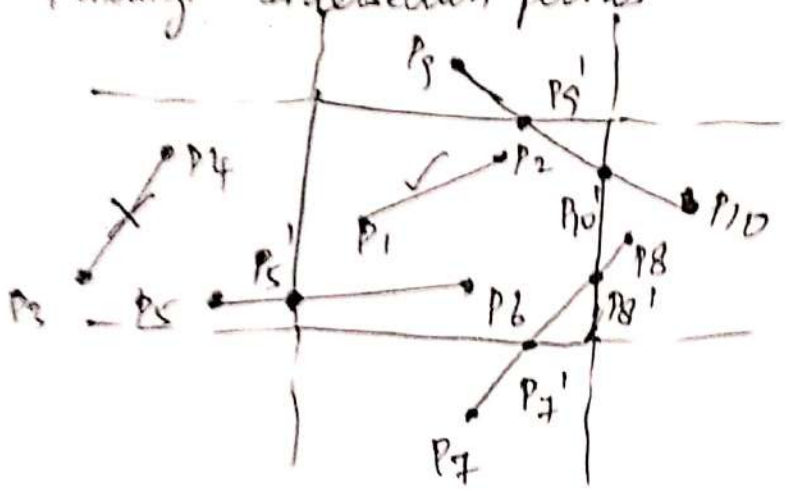
(iii) If any One of P1 and P_2 is zero → logical AND → zero [0000]

If Both P1 and P_2 are Non zero → logical AND → zero [0000]

↓

Partially visible (Find Intersection point)

Finding Intersection points



(x_1, y_1) (x_2, y_2)

y values vertical line (x, y) Intersection point.

$$\frac{y - y_1}{x - x_1} = m \Rightarrow y - y_1 = m(x - x_1)$$

$$\boxed{y = y_1 + m(x - x_1)}$$

↓ Slope between (x_1, y_1) and (x, y) — x can be x_{min} or x_{max}

x-value horizontal line

$$\frac{y - y_1}{x - x_1} = m \Rightarrow y - y_1 = m(x - x_1)$$

$$\frac{y - y_1}{m} = x - x_1 \Rightarrow \boxed{x = x_1 + \frac{1}{m}(y - y_1)}$$

↓ y can be y_{min} or y_{max}

$$\boxed{m = \frac{y_2 - y_1}{x_2 - x_1}}$$

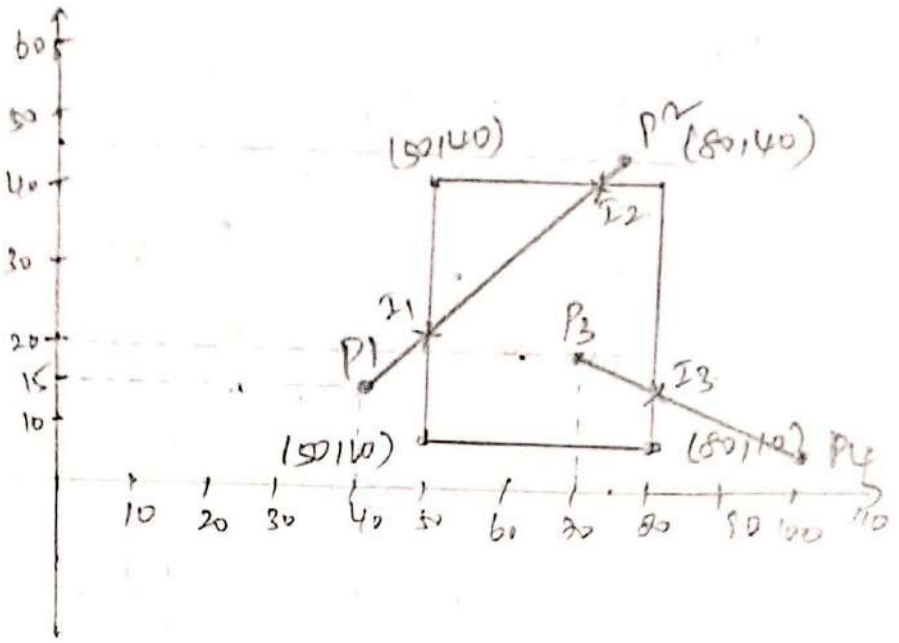
Example: Use the Cohen Sutherland Outcode algorithm to clip two lines $P_1(40, 15)$ - $P_2(75, 45)$ and $P_3(70, 20)$ - $P_4(100, 10)$ against a window A(50, 10), B(80, 10), C(80, 40), D(50, 40).

$x_{min} = 50$

$x_{max} = 80$

$y_{min} = 10$

$y_{max} = 40$



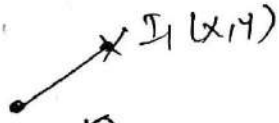
Line 1

	outcode
P1	0001
P2	1000

Logical AND → zero → Partially visible [0000]

↓ find Intersection point

TO find I_1



x value is known (50)

y value ?

(40, 15) and (75, 45)

x_1 y_1 x_2 y_2

$y = y_1 + m(x - x_1) \Rightarrow y_1 + m(x_{min} - x_1)$

$m = \frac{45-15}{75-40} = \frac{30}{35_7}$

$m = 6/7$

$15 + 6/7 (50 - 40)$

↳ since it intersects with x_{min}

$\Rightarrow y = 15 + 6/7 (10)$

$= 15 + \frac{60}{7} = \frac{105+60}{7} = \frac{165}{7} = 23.57$

$\therefore I_1 = (50, 23.57)$

To find I_2

(x,y) (75,45)

x_1 y_1

$x = x_1 + 1/m (y - y_1)$

$= 75 + 7/6 (40 - 45)$ ↳ y_{max}

$= 75 - 35/6 = 69.17$

$I_2 = (69.17, 40)$

Line 2 $P_3(70,20)$ $P_4(100,10)$

Logical AND

P_3 0000
P_4 0010

0000 → Partially visible.
(Find Intersection point)

To find I_3

$$m = \frac{10-20}{100-70} = \frac{-10}{30} = -\frac{1}{3} \Rightarrow \boxed{m = -\frac{1}{3}}$$

$P_3(70,20)$ $I_3(x,y)$
$x_1\ y_1$

$$y = m(x - x_1) + y_1$$

x_{max}

$$= -\frac{1}{3}(80-70) + 20 = -\frac{1}{3}(10) + 20$$

$$= -\frac{10}{3} + 20 = \frac{60-10}{3} = 16.67$$

$$\therefore \boxed{I_3 = (80, 16.67)}$$

We can also take P_4 I_3
$(100,10)$ (x,y)
$x_1\ y_1$

$$y = -\frac{1}{3}(80-100) + 10$$

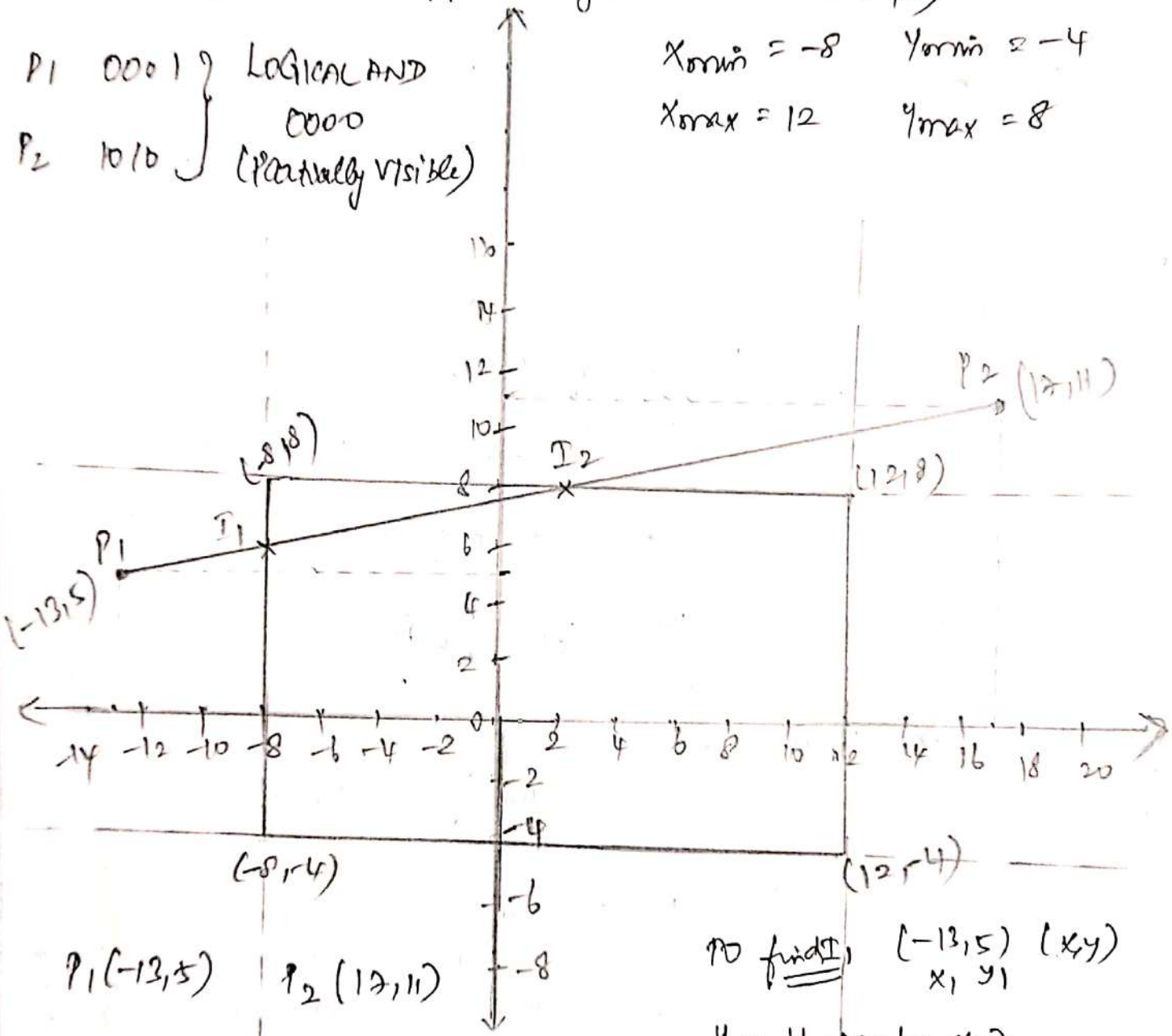
$$= -\frac{1}{3}(-20) + 10 = \frac{20}{3} + 10 = \frac{50}{3} = 16.67$$

$$\therefore \boxed{I_3 = (80, 16.67)}$$

Example: Use Region Code Based Line clipping method to clip a line starting from $(-13, 5)$ and ending at $(17, 11)$ against the window having its lower left corner at $(-8, -4)$ and upper right corner at $(12, 8)$

P_1 0001 } LOGICAL AND
P_2 1010 } 0000
(Partially visible)

$x_{min} = -8$ $y_{min} = -4$
$x_{max} = 12$ $y_{max} = 8$



$P_1(-13, 5)$ $P_2(17, 11)$

$$m = \frac{11-5}{17+13} = \frac{6}{30} = \frac{1}{5}$$

To find I_1 $(-13, 5)$ (x, y)
(x_1, y_1)

$$y = y_1 + m(x - x_1)$$
$$= 5 + \frac{1}{5}(-8 + 13)$$
$$= 5 + \frac{1}{5}(5) = 6.$$

$I_1 = (-8, 6)$

To find I_2 $(17, 11)$ (x, y)
(x_1, y_1)

$$x = x_1 + \frac{1}{m}(y - y_1)$$
$$= 17 + 5(8 - 11) = 17 + 5(-3) = 17 - 15 = 2$$

$I_2 = (2, 8)$

Content - Fourth Year

Computer Graphics

UNIT – IV

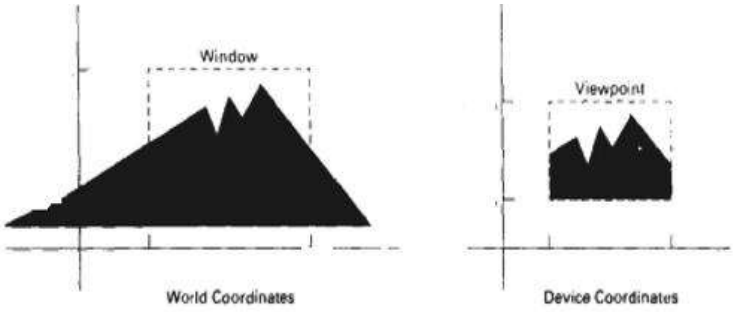
Two Dimensional Viewing

The viewing pipeline

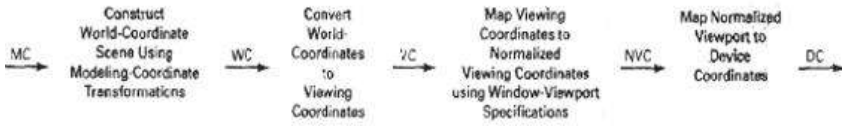
A world coordinate area selected for display is called a window. An area on a display device to which a window is mapped is called a view port. The window defines what is to be viewed the view port defines where it is to be displayed.

The mapping of a part of a world coordinate scene to device coordinate is referred to as viewing transformation. The two dimensional viewing transformation is referred to as window to view port transformation of windowing transformation.

A viewing transformation using standard rectangles for the window and viewport



The two dimensional viewing transformation pipeline



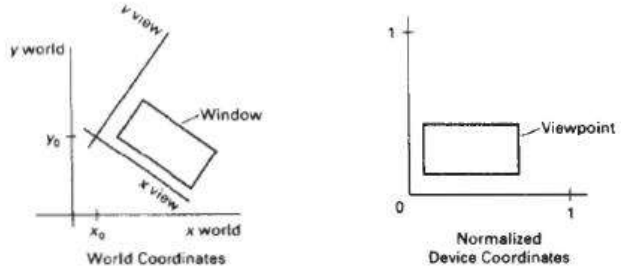
The viewing transformation in several steps, as indicated in Fig. First, we construct the scene in world coordinates using the output primitives. Next to obtain a particular orientation for the window, we can set up a two-dimensional viewingcoordinate system in the world coordinate plane, and define a window in the viewingcoordinate system.

The viewing- coordinate reference frame is used to provide a method for setting up arbitrary orientations for rectangular windows. Once the viewing reference frame is established, we can transform descriptions in world coordinates to viewing coordinates.

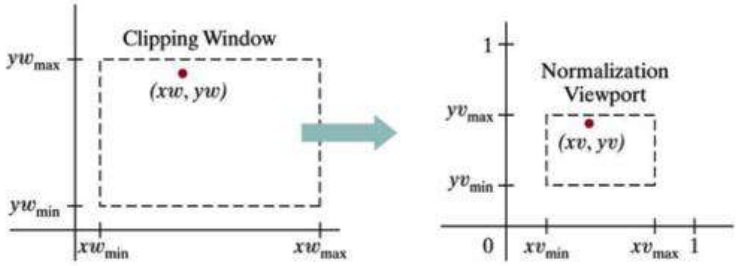
We then define a viewport in normalized coordinates (in the range from 0 to 1) and map the viewing-coordinate description of the scene to normalized coordinates.

At the final step all parts of the picture that lie outside the viewport are clipped, and the contents of the viewport are transferred to device coordinates. By changing the position of the viewport, we can view objects at different positions on the display

area of an output device.



Window to view port coordinate transformation:



A point (xw, yw) in a world-coordinate clipping window is mapped to viewport coordinates (xv, yv), within a unit square, so that the relative positions of the two points in their respective rectangles are the same.

A point at position (xw,yw) in a designated window is mapped to viewport coordinates (xv, yv) so that relative positions in the two areas are the same. The figure illustrates the window to view port mapping.

A point at position (xw,yw) in the window is mapped into position (xv,yv) in the associated view port. To maintain the same relative placement in view port as in window

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

$$\frac{yv - yv_{min}}{yv_{max} - yv_{min}} = \frac{yw - yw_{min}}{yw_{max} - yw_{min}}$$

solving these expressions for view port position (x_v,y_v)

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

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

where scaling factors are

$sx = xv_{max} - xv_{min}$ $\qquad$ $sy = yv_{max} - yv_{min}$

$xw_{max} - xw_{min}$ $\qquad$ $yw_{max} - yw_{min}$

The conversion is performed with the following sequence of transformations.

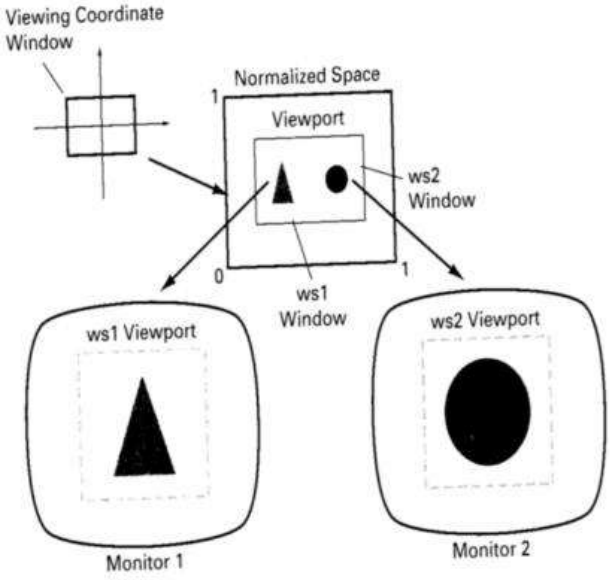
1. Perform a scaling transformation using point position of (xw min, yw min) that scales the window area to the size of view port.
2. Translate the scaled window area to the position of view port. Relative proportions of objects are maintained if scaling factor are the same(Sx = Sy).

Otherwise world objects will be stretched or contracted in either the x or y direction when displayed on output device. For normalized coordinates, object descriptions are mapped to various display devices.

Any number of output devices can be open in particular application and another window view port transformation can be performed for each open output device. This mapping called the work station transformation is accomplished by selecting a window

area in normalized apace and a view port are in coordinates of display device.

Mapping selected parts of a scene in normalized coordinate to different video monitors with work station transformation.



Two Dimensional viewing functions

Viewing reference system in a PHIGS application program has following function.

evaluateViewOrientationMatrix (x_0, y_0, x_v, y_v, error, viewMatrix)

where x_0, y_0 are coordinate of viewing origin and parameter x_v, y_v are the world coordinate positions for view up vector. An integer error code is generated if the input parameters are in error otherwise the view matrix for world-to-viewing transformation is calculated. Any number of viewing transformation matrices can be defined in an application.

To set up elements of window to view port mapping

evaluateViewMappingMatrix ($x_w$min, $x_w$max, $y_w$min, $y_w$max, $x_v$min, $x_v$max, $y_v$min, $y_v$max, error, viewMappingMatrix)

Here window limits in viewing coordinates are chosen with parameters $x_w$min, $x_w$max, $y_w$min, $y_w$max and the viewport limits are set with normalized coordinate positions $x_v$min, $x_v$max, $y_v$min, $y_v$max.

The combinations of viewing and window view port mapping for various workstations in a viewing table with

setViewRepresentation (ws, viewIndex, viewMatrix, viewMappingMatrix, xclipmin, xclipmax, yclipmin, yclipmax, clipxy)

Where parameter ws designates the output device and parameter view index sets an integer identifier for this window-view port point. The matrices viewMatrix and viewMappingMatrix can be concatenated and referenced by viewIndex.

setViewIndex (viewIndex)

selects a particular set of options from the viewing table.

At the final stage we apply a workstation transformation by selecting a work station window viewport pair.

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

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

where was gives the workstation number. Window-coordinate extents are specified in

the range from 0 to 1 and viewport limits are in integer device coordinates.

Clipping operation

Any procedure that identifies those portions of a picture that are inside or outside of a specified region of space is referred to as clipping algorithm or clipping. The region against which an object is to be clipped is called clip window.

Algorithm for clipping primitive types:

* Point Clipping
* Line Clipping (Straight-line segment)
* Area Clipping
* Curve Clipping
* Text Clipping

Line and polygon clipping routines are standard components of graphics packages.

Point Clipping

Clip window is a rectangle in standard position. A point P=(x,y) for display, if following inequalities are satisfied:

$$xw_{min} <= x <= xw_{max}$$

$$yw_{min} <= y <= yw_{max}$$

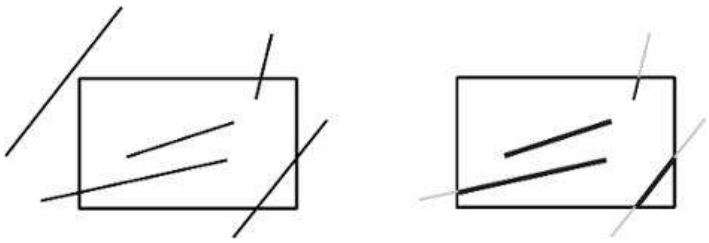
where the edges of the clip window **(xw_{min}, xw_{max}, yw_{min}, yw_{max})** can be either the world-coordinate window boundaries or viewport boundaries. If any one of these four inequalities is not satisfied, the point is clipped (not saved for display).

Line Clipping

A line clipping procedure involves several parts. First we test a given line segment whether it lies completely inside the clipping window. If it does not we try to determine whether it lies completely outside the window . Finally if we can not identify a line as completely inside or completely outside, we perform intersection calculations with one or more clipping boundaries.

Process lines through "inside-outside" tests by checking the line endpoints. A line with both endpoints inside all clipping boundaries such as line from P1 to P2 is saved. A line with both end point outside any one of the clip boundaries line P3P4 is outside the window.

Line clipping against a rectangular clip window



All other lines cross one or more clipping boundaries. For a line segment with end points (x_1, y_1) and (x_2, y_2) one or both end points outside clipping rectangle, the parametric representation

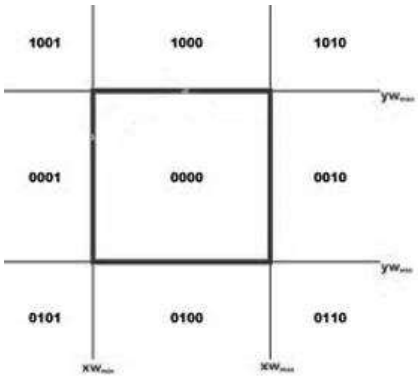
$$x = x_1 + u(x_2 - x_1)$$
$$y = y_1 + u(y_2 - y_1) \quad 0 \le u \le 1$$

could be used to determine values of u for an intersection with the clipping boundary coordinates. If the value of u for an intersection with a rectangle boundary edge is outside the range of 0 to 1, the line does not enter the interior of the window at that boundary. If the value of u is within the range from 0 to 1, the line segment does indeed cross into the clipping area. This method can be applied to each clipping boundary edge in to determined whether any part of line segment is to displayed.

Cohen-Sutherland Line Clipping

This is one of the oldest and most popular line-clipping procedures. The method speeds up the processing of line segments by performing initial tests that reduce the number of intersections that must be calculated.

Every line endpoint in a picture is assigned a four digit binary code called a **region code** that identifies the location of the point relative to the boundaries of the clipping rectangle.



Binary region codes assigned to line end points according to relative position with respect to the clipping rectangle.

Regions are set up in reference to the boundaries. Each bit position in region code is used to indicate one of four relative coordinate positions of points with respect to clip window: to the left, right, top or bottom. By numbering the bit positions in the region code as 1 through 4 from right to left, the coordinate regions are corrected with bit positions as

bit 1: left
bit 2: right
bit 3: below
bit4: above

A value of 1 in any bit position indicates that the point is in that relative position. Otherwise the bit position is set to 0. If a point is within the clipping rectangle the region code is 0000. A point that is below and to the left of the rectangle has a region code of 0101.

Bit values in the region code are determined by comparing endpoint coordinate values (x, y) to clip boundaries. Bit1 is set to 1 if $x < xw_{min}$.

For programming language in which bit manipulation is possible region-code bit values can be determined with following two steps.

1. Calculate differences between endpoint coordinates and clipping boundaries.
2. Use the resultant sign bit of each difference calculation to set the corresponding value in the region code.

bit 1 is the sign bit of $x - xw_{min}$

bit 2 is the sign bit of $xw_{max} - x$

bit 3 is the sign bit of $y - yw_{min}$

bit 4 is the sign bit of $yw_{max} - y$.

Once we have established region codes for all line endpoints, we can quickly determine which lines are completely inside the clip window and which are clearly outside.

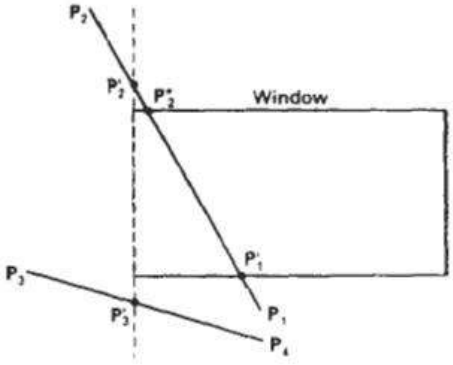
Any lines that are completely contained within the window boundaries have a region code of 0000 for both endpoints, and we accept these lines. Any lines that have a 1 in the same bit position in the region codes for each endpoint are completely

outside the clipping rectangle, and we reject these lines.

We would discard the line that has a region code of 1001 for one endpoint and a code of 0101 for the other endpoint. Both endpoints of this line are left of the clipping rectangle, as indicated by the 1 in the first bit position of each region code.

A method that can be used to test lines for total clipping is to perform the logical and operation with both region codes. If the result is not 0000,the line is completely outside the clipping region.

Lines that cannot be identified as completely inside or completely outside a clip window by these tests are checked for intersection with window boundaries.



Line extending from one coordinates region to another may pass through the clip window, or they may intersect clipping boundaries without entering window.

Cohen-Sutherland line clipping starting with bottom endpoint left, right , bottom and top boundaries in turn and find that this point is below the clipping rectangle.

Starting with the bottom endpoint of the line from P_1 to P_2, we check P_1 against the left, right, and bottom boundaries in turn and find that this point is below the clipping rectangle. We then find the intersection point P_1' with the bottom boundary and discard the line section from P_1 to P_1'.

The line now has been reduced to the section from P_1' to P_2, Since P_2, is outside the clip window, we check this endpoint against the boundaries and find that it is to the left of the window. Intersection point P_2' is calculated, but this point is above the window. So the final intersection calculation yields P_2'', and the line from P_1' to P_2'', is saved. This completes processing for this line, so we save this part and go on to the next line.

Point P_3 in the next line is to the left of the clipping rectangle, so we determine the intersection P_3', and eliminate the line section from P_3 to P_3'. By checking region codes for the line section from P_3' to P_4 we find that the remainder of the line is below the clip window and can be discarded also.

Intersection points with a clipping boundary can be calculated using the slopeintercept form of the line equation. For a line with endpoint coordinates (x_1, y_1) and (x_2, y_2) and the y coordinate of the intersection point with a vertical boundary can be obtained with the calculation.

$$y = y_1 + m (x - x_1)$$

where x value is set either to xwmin or to xwmax and slope of line is calculated as

$$m = (y_2 - y_2) / (x_2 - x_1)$$

the intersection with a horizontal boundary the x coordinate can be calculated as

$$x = x_1 + (y - y_1) / m$$

with y set to either to yw_{min} or to yw_{max}.

Implementation of Cohen-sutherland Line Clipping

```
#define Round(a)          ((int)(a+0.5))
#define LEFT_EDGE       0x1
#define RIGHT_EDGE      0x2
#define BOTTOM_EDGE  0x4
#define TOP_EDGE 0x8
#define TRUE 1
#define FALSE 0

#define INSIDE(a) (!a)
#define REJECT(a,b) (a&b)
#define ACCEPT(a,b) (!(a|b))

unsigned char encode(wcPt2 pt, dcPt winmin, dcPt winmax)
unsigned char code=0x00;
if(pt.x<winmin.x)
code=code|LEFT_EDGE;
if(pt.x>winmax.x)
code=code|RIGHT_EDGE;
if(pt.y<winmin.y)
code=code|BOTTOM_EDGE;
if(pt.y>winmax.y)
code=code|TOP_EDGE;
return(code);
}
void swappts(wcPt2 *p1,wcPt2 *p2)
{
wcPt2 temp;
tmp=*p1;
*p1=*p2;
*p2=tmp;
}
void swapcodes(unsigned char *c1,unsigned char *c2)
{
unsigned char tmp;
tmp=*c1;
*c1=*c2;
*c2=tmp;
}
void clipline(dcPt winmin, dcPt winmax, wcPt2 p1,ecPt2 point p2)
{
unsigned char code1,code2;
int done=FALSE, draw=FALSE;
float m;
while(!done)
{
code1=encode(p1,winmin,winmax);
code2=encode(p2,winmin,winmax);
if(ACCEPT(code1,code2))
```

```
{
done=TRUE;
draw=TRUE;
}
else if(REJECT(code1,code2))
done=TRUE;
else
if(INSIDE(code1))
{
swappts(&p1,&p2);
swapcodes(&code1,&code2);
}
if(p2.x!=p1.x)
m=(p2.y-p1.y)/(p2.x-p1.x);
if(code1 &LEFT_EDGE)
{
p1.y+=(winmin.x-p1.x)*m;
p1.x=winmin.x;
}
else if(code1 &RIGHT_EDGE)
{
p1.y+=(winmax.x-p1.x)*m;
p1.x=winmax.x;
}
else if(code1 &BOTTOM_EDGE)
{
if(p2.x!=p1.x)
p1.x+=(winmin.y-p1.y)/m;
p1.y=winmin.y;
}
else if(code1 &TOP_EDGE)
{
if(p2.x!=p1.x)
p1.x+=(winmax.y-p1.y)/m;
p1.y=winmax.y;
}
}
}
if(draw)
lineDDA(ROUND(p1.x), ROUND(p1.y), ROUND(p2.x), ROUND(p2.y));
}
```

Liang – Barsky line Clipping:

Based on analysis of parametric equation of a line segment, faster line clippers have been developed, which can be written in the form:

$$x = x_1 + u\,\Delta x$$

$$y = y_1 + u\,\Delta y \qquad 0 <= u <= 1$$

where $\Delta x = (x_2 - x_1)$ and $\Delta y = (y_2 - y_1)$

In the Liang-Barsky approach we first the point clipping condition in parametric form:

$$xw_{min} <= x_1 + u\,\Delta x <= xw_{max}$$

$$yw_{min} <= y_1 + u\,\Delta y <= yw_{max}$$

Each of these four inequalities can be expressed as

$$\mu p_k <= q_k \qquad k = 1, 2, 3, 4$$

the parameters p & q are defined as

$$p_1 = -\Delta x \qquad q_1 = x_1 - xw_{min}$$

$$p_2 = \Delta x \qquad q_2 = xw_{max} - x_1$$

$$P_3 = -\Delta y \qquad q_3 = y_1 - yw_{min}$$

$$P_4 = \Delta y \qquad q_4 = yw_{max} - y_1$$

Any line that is parallel to one of the clipping boundaries have $p_k = 0$ for values of k corresponding to boundary k = 1, 2, 3, 4 correspond to left, right, bottom and top boundaries. For values of k, find $q_k < 0$, the line is completely out side the boundary.

If qk > = 0, the line is inside the parallel clipping boundary.

When $p_k < 0$ the infinite extension of line proceeds from outside to inside of the infinite extension of this clipping boundary.

If $p_k > 0$, the line proceeds from inside to outside, for non zero value of p_k calculate the value of u, that corresponds to the point where the infinitely extended line intersect the extension of boundary k as

$$u = qk / pk$$

For each line, calculate values for parameters u1and u2 that define the part of line that lies within the clip rectangle. The value of u1 is determined by looking at the rectangle edges for which the line proceeds from outside to the inside (p < 0).
For these edges we calculate

$$r_k = qk / pk$$

The value of u1 is taken as largest of set consisting of 0 and various values of r. The value of u2 is determined by examining the boundaries for which lines proceeds from inside to outside (P > 0).

A value of rkis calculated for each of these boundaries and value of u2 is the minimum of the set consisting of 1 and the calculated r values.

If u1 > u2, the line is completely outside the clip window and it can be rejected.

Line intersection parameters are initialized to values u1 = 0 and u2 = 1. for each clipping boundary, the appropriate values for P and q are calculated and used by function.

Cliptest to determine whether the line can be rejected or whether the intersection parameter can be adjusted.

When p < 0, the parameter r is used to update u1.

When p > 0, the parameter r is used to update u2.

If updating u1 or u2 results in u1 > u2 reject the line, when p = 0 and q < 0, discard the line, it is parallel to and outside the boundary. If the line has not been rejected after all four value of p and q have been tested, the end points of clipped lines are determined from values of u1 and u2.

The Liang-Barsky algorithm is more efficient than the Cohen-Sutherland algorithm since intersections calculations are reduced. Each update of parameters u1 and u2 require only one division and window intersections of these lines are computed only once.

Cohen-Sutherland algorithm, can repeatedly calculate intersections along a line path, even through line may be completely outside the clip window. Each intersection calculations require both a division and a multiplication.

Implementation of Liang-Barsky Line Clipping

```
#define Round(a) ((int)(a+0.5))
int clipTest (float p, float q, gfloat *u1, float *u2)
{
float r;
int retval=TRUE;
if (p<0.0)
{
r=q/p
if (r>*u2)
retVal=FALSE;
else
if (r>*u1)
*u1=r;
}
else
if (p>0.0)
{
r=q/p
if (r<*u1)
retVal=FALSE;
else
if (r<*u2)
*u2=r;
}
else
if )q<0.0)
retVal=FALSE
return(retVal);

void clipLine (dcPt winMin, dcPt winMax, wcPt2 p1, wcpt2 p2)
{
float u1=0.0, u2=1.0, dx=p2.x-p1.x,dy;
if (clipTest (-dx, p1.x-winMin.x, &u1, &u2))
if (clipTest (dx, winMax.x-p1.x, &u1, &u2))
{
dy=p2.y-p1.y;
if (clipTest (-dy, p1.y-winMin.y, &u1, &u2))
if (clipTest (dy, winMax.y-p1.y, &u1, &u2))
{
if (u1<1.0)
{
p2.x=p1.x+u2*dx;
p2.y=p1.y+u2*dy;
}
if (u1>0.0)
{
p1.x=p1.x+u1*dx;
p1.y=p1.y+u1*dy;
}
```

```
lineDDA(ROUND(p1.x),ROUND(p1.y),ROUND(p2.x),ROUND(p2.y));
}
}
}
```

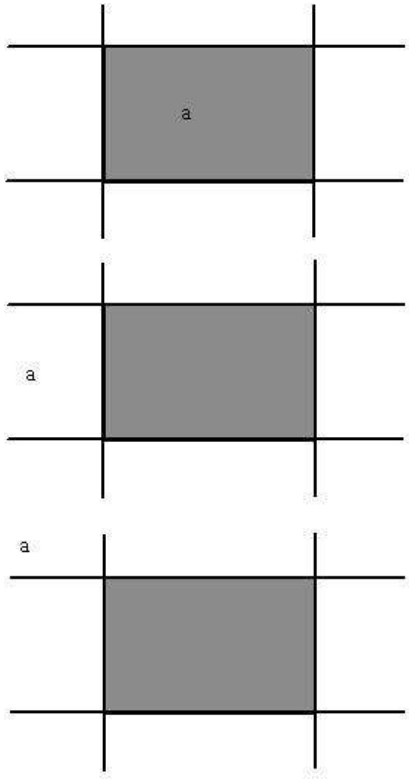
Nicholl-Lee-Nicholl Line clipping

By creating more regions around the clip window, the Nicholl-Lee-Nicholl (or NLN) algorithm avoids multiple clipping of an individual line segment. In the Cohen-Sutherland method, multiple intersections may be calculated.These extra intersection calculations are eliminated in the NLN algorithm by carrying out more region testing before intersection positions are calculated.

Compared to both the Cohen-Sutherland and the Liang-Barsky algorithms, the Nicholl-Lee-Nicholl algorithm performs fewer comparisons and divisions. The trade-off is that the NLN algorithm can only be applied to two-dimensional dipping, whereas both the Liang-Barsky and the Cohen-Sutherland methods are easily extended to threedimensional scenes.

For a line with endpoints P1 and P2 we first determine the position of point P1, for the nine possible regions relative to the clipping rectangle. Only the three regions shown in Fig. need to be considered. If P1 lies in any one of the other six regions, we can move it to one of the three regions in Fig. using a symmetry transformation. For example, the region directly above the clip window can be transformed to the region left of the clip window using a reflection about the line y = **-x,** or we could use a **90 degree** counterclockwise rotation.

Three possible positions for a line endpoint p1(a) in the NLN algorithm



Case 1: p1 inside region

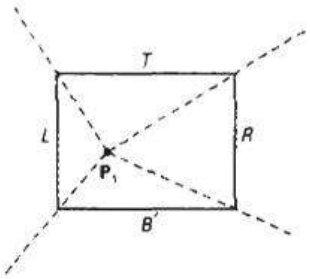
Case 2: p1 across edge

Case 3: p1 across corner

Next, we determine the position of P2 relative to P1. To do this, we create some new regions in the plane, depending on the location of P1. Boundaries of the new regions are half-infinite line segments that start at the position of P1 and pass through the window corners. If P1 is inside the clip window and P2 is outside, we set up the

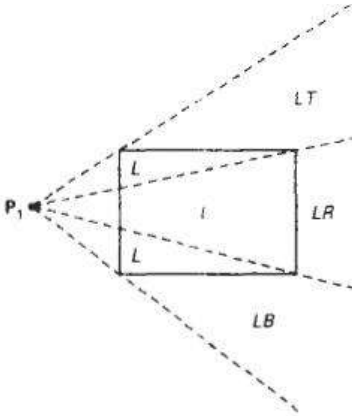
four regions shown in Fig

The four clipping regions used in NLN alg when p1 is inside and p2 outside the clip window



The intersection with the appropriate window boundary is then carried out, depending on which one of the four regions (L, T, R, or B) contains P2. If both P1 and P2 are inside the clipping rectangle, we simply save the entire line.

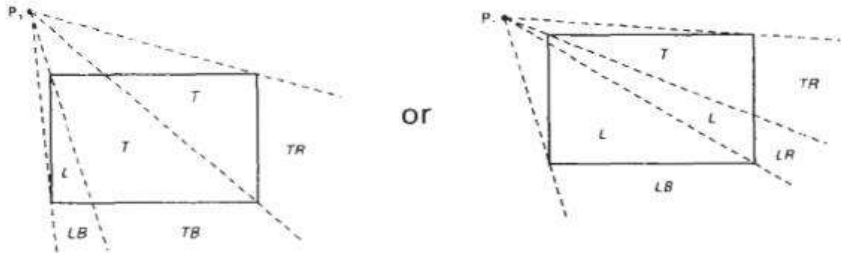
If P1 is in the region to the left of the window, we set up the four regions, **L, LT, LR,** and **LB,** shown in Fig.



These four regions determine a unique boundary for the line segment. For instance, if P2 is in region L, we clip the line at the left boundary and save the line segment from this intersection point to P2. But if P2 is in region LT, we save the line segment from the left window boundary to the top boundary. If P2 is not in any of the four regions, L, LT, LR, or LB, the entire line is clipped.

For the third case, when P1 is to the left and above the clip window, we usethe clipping regions in Fig.

The two possible sets of clipping regions used in NLN algorithm when P1 is above and to the left of the clip window



In this case, we have the two possibilities shown, depending on the position of P1, relative to the top left corner of the window. If P2, is in one of the regions T, L, TR, TB, LR, or LB, this determines a unique clip window edge for the intersection calculations. Otherwise, the entire line is rejected.

To determine the region in which P2 is located, we compare the slope of the line to the slopes of the boundaries of the clip regions. For example, if P1 is left of the clipping rectangle (Fig. a), then P2, is in region LT if

$$\text{slope}\overline{P_1P_{TR}} < \text{slope}\overline{P_1P_2} < \text{slope}P_1P_{TL}$$

(or)

$$\frac{y_T - y_1}{\overline{x_R - x_1}} < \frac{y_2 - y_1}{\overline{x_2 - x_1}} < \frac{y_T - y_1}{\overline{x_L - x_1}}$$

And we clip the entire line if

$(y_T - y_1)\,(x_2 - x_1) < (x_L - x_1)\,(y_2 - y_1)$

The coordinate difference and product calculations used in the slope tests are saved and also used in the intersection calculations. From the parametric equations

$$x = x_1 + (x_2 - x_1)u$$

$$y = y_1 + (y_2 - y_1)u$$

an x-intersection position on the left window boundary is $x = x_L$,, with

$u = (x_L - x_1)/(x_2 - x_1)$ so that the y-intersection position is

$$y = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x_L - x_1)$$

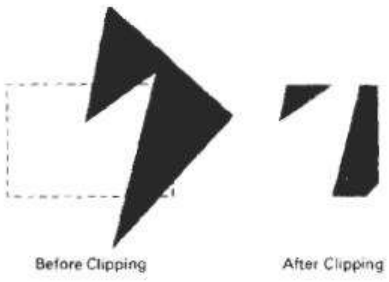
And an intersection position on the top boundary has $y = y_T$ and $u = (y_T - y_1)/(y_2 - y_1)$ with

$$x = x_1 + \frac{x_2 - x_1}{y_2 - y_1}(y_T - y_1)$$

POLYGON CLIPPING

To clip polygons, we need to modify the line-clipping procedures. A polygon boundary processed with a line clipper may be displayed as a series of unconnected line segments (Fig.), depending on the orientation of the polygon to the clipping window.

Display of a polygon processed by a line clipping algorithm



For polygon clipping, we require an algorithm that will generate one or more closed areas that are then scan converted for the appropriate area fill. The output of a polygon clipper should be a sequence of vertices that defines the clipped polygon boundaries.

Sutherland – Hodgeman polygon clipping:

A polygon can be clipped by processing the polygon boundary as a whole against each window edge. This could be accomplished by processing all polygon vertices against each clip rectangle boundary.

There are four possible cases when processing vertices in sequence around the perimeter of a polygon. As each point of adjacent polygon vertices is passed to a window boundary clipper, make the following tests:

1. If the first vertex is outside the window boundary and second vertex is inside, both the intersection point of the polygon edge with window boundary and second vertex are added to output vertex list.

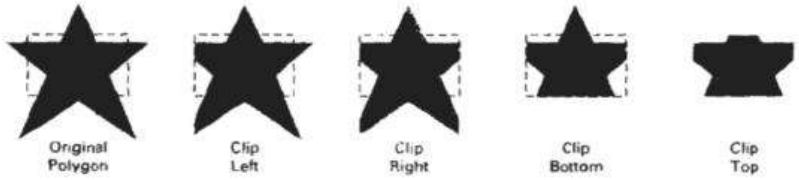
2. If both input vertices are inside the window boundary, only the second vertex is

added to the output vertex list.

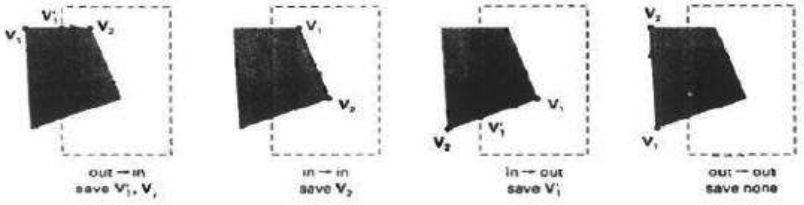
3. If first vertex is inside the window boundary and second vertex is outside only the edge intersection with window boundary is added to output vertex list.

4. If both input vertices are outside the window boundary nothing is added to the output list.

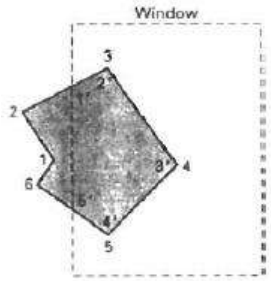
Clipping a polygon against successive window boundaries.



Successive processing of pairs of polygon vertices against the left window boundary



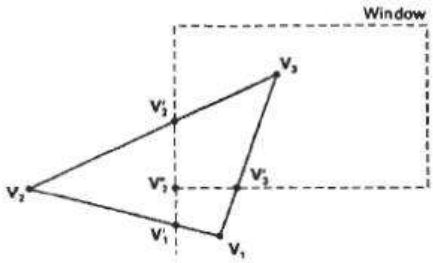
Clipping a polygon against the left boundary of a window, starting with vertex 1. Primed numbers are used to label the points in the output vertex list for this window boundary.



Vertices 1 and 2 are found to be on outside of boundary. Moving along vertex 3 which is inside, calculate the intersection and save both the intersection point and vertex 3. Vertex 4 and 5 are determined to be inside and are saved. Vertex 6 is outside so we find and save the intersection point. Using the five saved points we repeat the process for next window boundary.

Implementing the algorithm as described requires setting up storage for an output list of vertices as a polygon clipped against each window boundary. We eliminate the intermediate output vertex lists by simply by clipping individual vertices at each step and passing the clipped vertices on to the next boundary clipper.

A point is added to the output vertex list only after it has been determined to be inside or on a window boundary by all boundary clippers. Otherwise the point does not continue in the pipeline.



A polygon overlapping a rectangular clip window

Processing the vertices of the polygon in the above fig. through a boundary clipping pipeline. After all vertices are processed through the pipeline, the vertex list is {v2", v2', v3,v3'}

Implementation of Sutherland-Hodgeman Polygon Clipping

```
typedef enum { Left,Right,Bottom,Top } Edge;
#define N_EDGE 4
#define TRUE 1
#define FALSE 0

int inside(wcPt2 p, Edge b,dcPt wmin,dcPt wmax)
{
switch(b)
{
case Left: if(p.x<wmin.x) return (FALSE); break;
case Right:if(p.x>wmax.x) return (FALSE); break;
case bottom:if(p.y<wmin.y) return (FALSE); break;
case top: if(p.y>wmax.y) return (FALSE); break;
}
return (TRUE);
}
int cross(wcPt2 p1, wcPt2 p2,Edge b,dcPt wmin,dcPt wmax)
{
if(inside(p1,b,wmin,wmax)==inside(p2,b,wmin,wmax))
return (FALSE);
else
return (TRUE);
}
wcPt2 (wcPt2 p1, wcPt2 p2,int b,dcPt wmin,dcPt wmax )
{
wcPt2 iPt;
float m;
if(p1.x!=p2.x)
m=(p1.y-p2.y)/(p1.x-p2.x);
switch(b)
{
case Left:
ipt.x=wmin.x;
ipt.y=p2.y+(wmin.x-p2.x)*m;
break;
case Right:
ipt.x=wmax.x;
ipt.y=p2.y+(wmax.x-p2.x)*m;
break;
case Bottom:
ipt.y=wmin.y;
if(p1.x!=p2.x)
ipt.x=p2.x+(wmin.y-p2.y)/m;
else
ipt.x=p2.x;
break;
```

```
case Top:
ipt.y=wmax.y;
if(p1.x!=p2.x)
ipt.x=p2.x+(wmax.y-p2.y)/m;
else
ipt.x=p2.x;
break;
}
return(ipt);
}
void clippoint(wcPt2 p,Edge b,dcPt wmin,dcPt wmax, wcPt2 *pout,int *cnt, wcPt2
*first[],struct point *s)
{
wcPt2 iPt;
if(!first[b])
first[b]=&p;
else
if(cross(p,s[b],b,wmin,wmax))
{
ipt=intersect(p,s[b],b,wmin,wmax);
if(b<top)
clippoint(ipt,b+1,wmin,wmax,pout,cnt,first,s);
else
{
pout[*cnt]=ipt;
(*cnt)++;
}
}
s[b]=p;
if(inside(p,b,wmin,wmax))
if(b<top)
clippoint(p,b+1,wmin,wmax,pout,cnt,first,s);
else
{
pout[*cnt]=p;
(*cnt)++;
}
}
void closeclip(dcPt wmin,dcPt wmax, wcPt2 *pout,int *cnt,wcPt2 *first[], wcPt2 *s)
{
wcPt2 iPt;
Edge b;
for(b=left;b<=top;b++)
{
if(cross(s[b],*first[b],b,wmin,wmax))
{
i=intersect(s[b],*first[b],b,wmin,wmax);
if(b<top)
clippoint(i,b+1,wmin,wmax,pout,cnt,first,s);
else
```

```
{
pout[*cnt]=i;
(*cnt)++;
}
}
}
}
int clippolygon(dcPt point wmin,dcPt wmax,int n,wcPt2 *pin, wcPt2 *pout)
{
wcPt2 *first[N_EDGE]={0,0,0,0},s[N_EDGE];
int i,cnt=0;
for(i=0;i<n;i++)
clippoint(pin[i],left,wmin,wmax,pout,&cnt,first,s);
closeclip(wmin,wmax,pout,&cnt,first,s);
return(cnt);
}
```

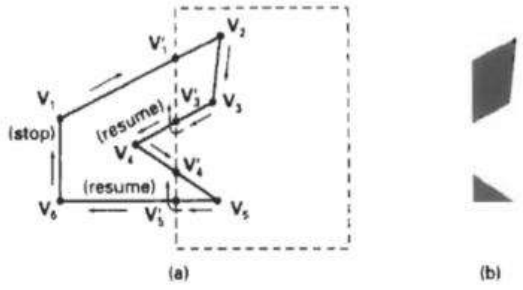
Weiler- Atherton Polygon Clipping

This clipping procedure was developed as a method for identifying visible surfaces, and so it can be applied with arbitrary polygon-clipping regions.

The basic idea in this algorithm is that instead of always proceeding around the polygon edges as vertices are processed, we sometimes want to follow the window boundaries. Which path we follow depends on the polygon-processing direction (clockwise or counterclockwise) and whether the pair of polygon vertices currently being processed represents an outside-to-inside pair or an inside- to-outside pair. For clockwise processing of polygon vertices, we use the following rules:

* For an outside-to-inside pair of vertices, follow the polygon boundary.
* For an inside-to-outside pair of vertices,. follow the window boundary in a clockwise direction.

In the below Fig. the processing direction in the Weiler-Atherton algorithm and the resulting clipped polygon is shown for a rectangular clipping window.



An improvement on the Weiler-Atherton algorithm is the Weiler algorithm, which applies constructive solid geometry ideas to clip an arbitrary polygon against any polygon clipping region.

Curve Clipping

Curve-clipping procedures will involve nonlinear equations, and this requires more processing than for objects with linear boundaries. The bounding rectangle for a circle or other curved object can be used first to test for overlap with a rectangular clip window.

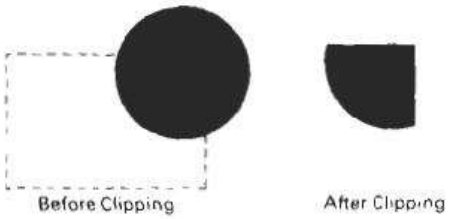
If the bounding rectangle for the object is completely inside the window, we save the object. If the rectangle is determined to be completely outside the window, we discard the object. In either case, there is no further computation necessary.

But if the bounding rectangle test fails, we can look for other computation-saving

approaches. For a circle, we can use the coordinate extents of individual quadrants and then octants for preliminary testing before calculating curve-window intersections.

The below figure illustrates circle clipping against a rectangular window. On the first pass, we can clip the bounding rectangle of the object against the bounding rectangle of the clip region. If the two regions overlap, we will need to solve the simultaneous linecurve equations to obtain the clipping intersection points.

Clipping a filled circle

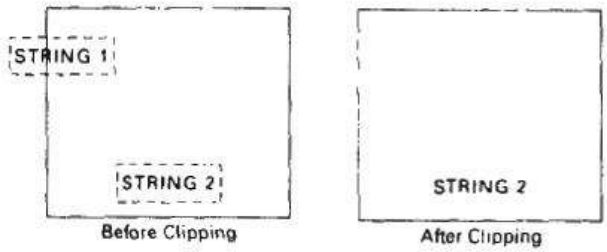


Text clipping

There are several techniques that can be used to provide text clipping in a graphics package. The clipping technique used will depend on the methods used to generate characters and the requirements of a particular application.

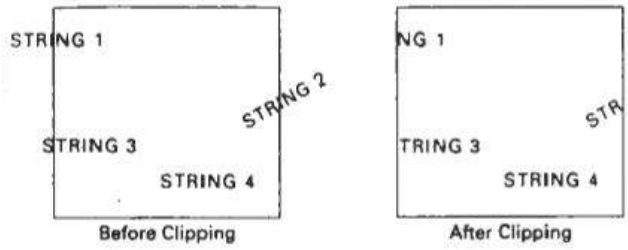
The simplest method for processing character strings relative to a window boundary is to use the **all-or-none string-clipping** strategy shown in Fig. If all of the string is inside a clip window, we keep it. Otherwise, the string is discarded. This procedure is implemented by considering a bounding rectangle around the text pattern. The boundary positions of the rectangle are then compared to the window boundaries, and the string is rejected if there is any overlap. This method produces the fastest text clipping.

Text clipping using a bounding rectangle about the entire string



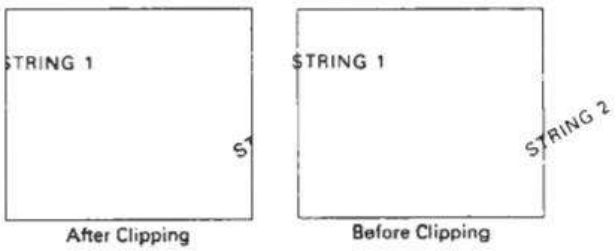
An alternative to rejecting an entire character string that overlaps a window boundary is to use the **all-or-none character-clipping** strategy. Here we discard only those characters that are not completely inside the window .In this case, the boundary limits of individual characters are compared to the window. Any character that either overlaps or is outside a window boundary is clipped.

Text clipping using a bounding rectangle about individual characters.



A final method for handling text clipping is to clip the components of individual characters. We now treat characters in much the same way that we treated lines. If an individual character overlaps a clip window boundary, we clip off the parts of the character that are outside the window.

Text Clipping performed on the components of individual characters



Exterior clipping:

Procedure for clipping a picture to the interior of a region by eliminating everything outside the clipping region. By these procedures the inside region of the picture is saved. To clip a picture to the exterior of a specified region. The picture parts to be saved are those that are outside the region. This is called as exterior clipping.

Objects within a window are clipped to interior of window when other higher priority window overlap these objects. The objects are also clipped to the exterior of overlapping windows.

::Back::

MULTIMEDIA APPLICATIONS

➢ DOCUMENT IMAGING:

The fundamental concepts of storage, compression and decompression, and display technologies used for multimedia systems were developed for document image management. Organizations such as insurance agencies law offices, country and state governments, and the federal government manage large volumes of documents.

➢ IMAGE PROCESSING AND RECOGNITION:

Image processing involves image recognition, Image enhancement, image synthesis, and image reconstruction.

An image processing system may actually alter the contents of the image itself. Image processing systems employ the compression and decompression techniques, a wide range of algorithm for object recognition, comparing images of objects with pre-defined objects, extrapolating finer details to view edges more clearly, gray-scale balancing and gray-scale and color adjustments.

➢ IMAGE ENHANCEMEWNT:

Most image display systems feature some level of image adjustment. Increasing the sensitivity and contrast makes the picture darker by making borderline pixels black or increasing the Gray-scale level of pixels.

Capabilities built in the compression boards might include the following

- **Image calibration**: The overall image density is calibrated, and the image pixels are adjusted to a predefined level.
- **Real time alignment**: The image is aligned in real-time for skewing caused by improper feeding of paper.
- **Gray-Scale normalization**: The overall gray level of an image or picture is evaluated to determine if it is skewed in one direction and if it needs cOlTection.
- **RGB hue intensity adjustment**: Too much color makes picture garish and fuzzy. Automatic hue intensity adjustment brings the hue intensity within pre-defined ranges.
- **Colour Separation**: A picture with very little color contrast can be dull and may not bring out the details. The hardware used can detect and adjust the range of color separation.
- **Frame averaging**: The intensity level of the frame is averaged to overcome the effects of very dark or very light areas by adjusting the middle tones.

➢ IMAGE ANIMATION:

Computers-created or scanned images can be displayed sequentially at controlled display speeds to provide image animation that simulates real processes.

The basic concept of displaying successive images at short intervals to give the perception of motion is being used successfully in designing moving parts such as automobile engines.

➢ IMAGE ANOTATION:

Image annotation can be performed in one of two ways: as a text file stored along with the image or as a small image stored with the original image. The annotation is overlayed over the original image for display purposes. It requires tracking multiple image components associated with a single page, decompressing all of them, and ensuring correct spatial alignment they are overlayed.

➢ OPTICAL CHARACTER RECOGNITION:

Data entry is the most expensive component of data processing, because it requires extensive clerical staff work to enter data.

Automating data entry, both typed and handwritten, is a significant application that can provide high returns. Optical Character Recognition (OCR) technology is used for data entry by scanning typed or printed words in a form.

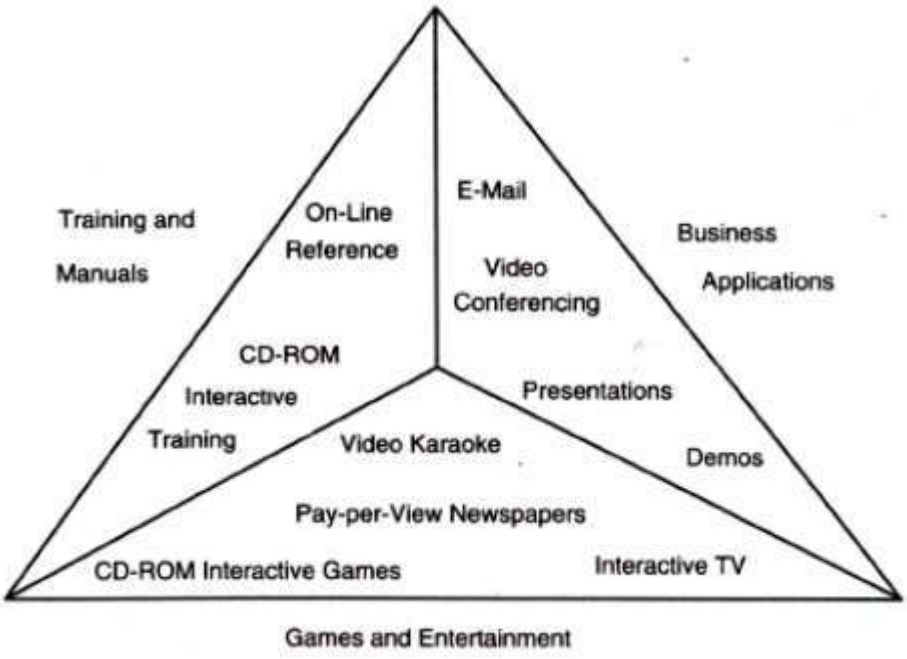
Initially, people used dedicated OCR scanners. Now, OCR Technology is available in software. OCR technology, used as a means of data entry, may be used for capturing entire paragraphs of text. The capturing text is almost a\ways entered as a field in a database or in an editable document

➢ FULL-MOTION DIGITAL VIDEO APPLICATION:

Full motion video has applications in the games industry and training, as well as the business world. Full motion video is the most complex and most demanding component of multimedia applications.

Some core requirements are:

- Full-motion video clips should be sharable but should have only one sharable copy.
- It should be possible to attach full-motion video clips to other documents such as memos, chapter text, presentation, and so on.



The following features should be available:

(a) Features, of a VCR metaphor, such as, rewind, fast-forward, play, and search.(b) Ability to move and resize the window displaying the video clip. (c) Ability to view the same clip on a variety of display terminal types with varying resolution capabilities without the need for storing multiple copies in different form (d) Ability to adjust the contrast and brightness of the video clip. (e) Ability to adjust the volume of the associated sound. (I) It should enable the users to place their own indexing marks to locate segments in video clip.

- **ELECTRONIC MESSAGING:**

➢ A Universal Multimedia Application:

It is an application that works on universal data type. This means that the application manipulates datatypes that can be combined in a document, displayed 'on a screen, or printed, with no special manipulations that the user needs to perform. The application is truly distributed in nature.

An important consideration for such a universal application is the methodology for dissemination of the information on a network.

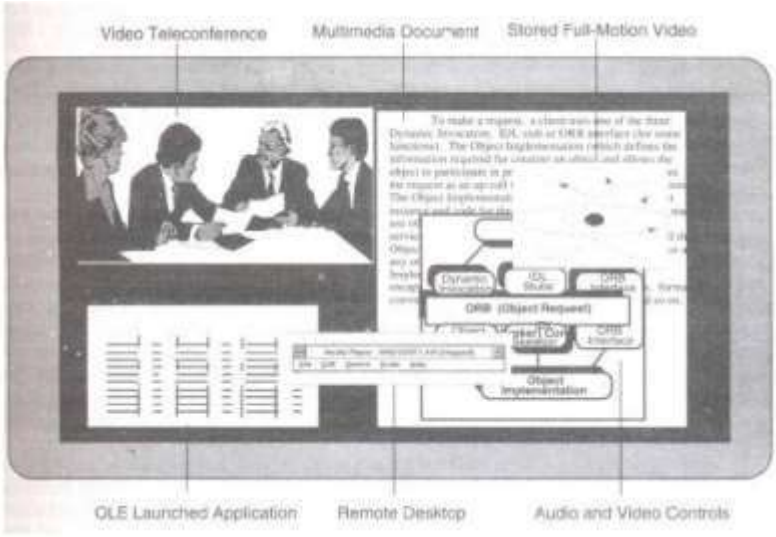


Figure describes the user screen for a universal multimedia application. **In** this screen, mix of windows for displaying still video and document images, a video conference window with a live session in progress, a remote live desk top, and a couple of other windows for applications such as electronic mail and desk top publishing.

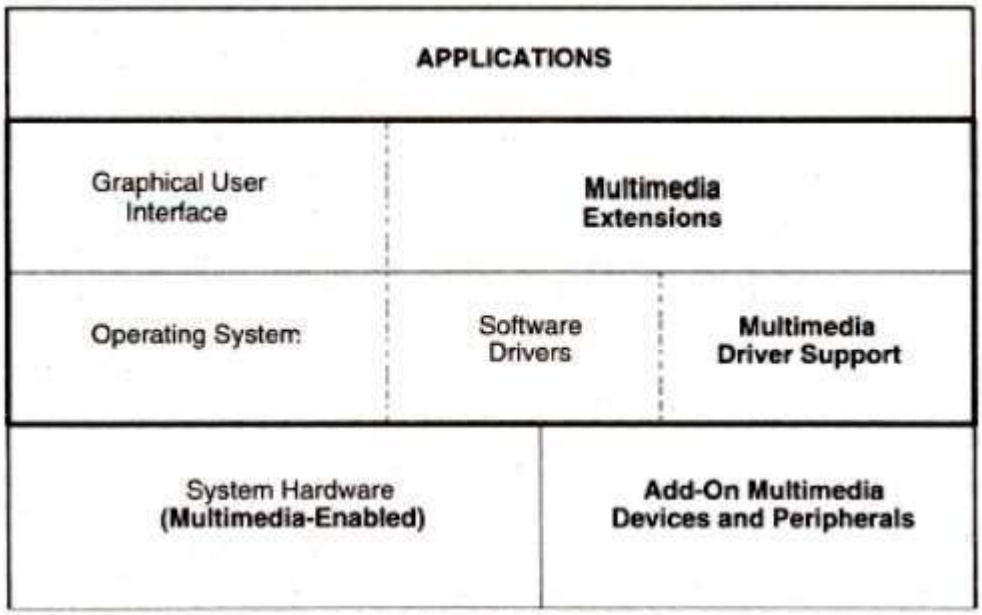
To maintain all of these windows requires a substantial amount of CPU power. Digital Signal Processing assistance is needed to manage the multiple simultaneous decompressions for JPEG, MPEG and windows applications.

MULTIMEDIA SYSTEM ARCHITECTURE

➢ MULTIMEDIA WORKSTATION ARCHITECTURE:

Multimedia encompasses a large variety of technologies and integration of multiple architectures interacting in real time. All of these multimedia capabilities must integrate with the standard user interfaces such as Microsoft Windows.

The following figure describes the architecture of a multimedia workstation environment. In this diagram.



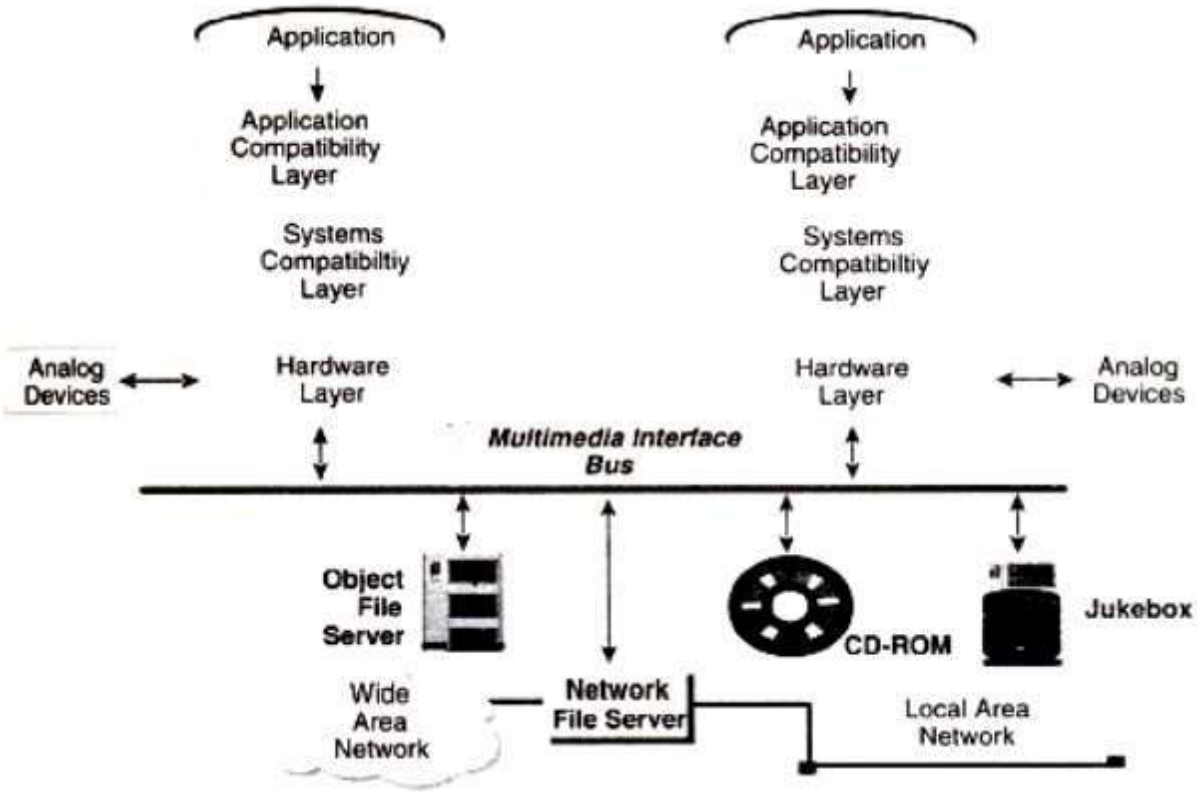
The right side shows the new architectural entities required for supporting multimedia applications.

For each special devices such as scanners, video cameras, VCRs and sound equipment-, a software device driver is need to provide the interface from an application to the device. The GUI require control extensions to support applications such as full motion video

- **THE IMA ARCHETECTURE FRAMEWORK:**

The Interactive Multimedia Association has a task group to define the architectural framework for multimedia to provide interoperability. The task group has C0ncentrated on the desktops and the servers. Desktop focus is to define the interchange formats. This format allows multimedia objects to be displayed on any work station.

The architectural approach taken by IMA is based on defining interfaces to a multimedia interface bus. This bus would be the interface between systems and multimedia sources. It provides streaming I/O service"s, including filters and translators **The Figure** describes the generalized architectural approach



➢ NETWORK ARCHITECTURE:

Multimedia systems need special networks. Because large volumes of images and video messages are being transmitted.

Asynchronous Transfer Mode technology (A TM) simplifies transfers across LANs and W ANs.

- **Task based Multi-level networking**

Higher classes of service require more expensive components in the' workstations as well as in the servers supporting the workstation applications.

Rather than impose this cost on all work stations, an alternate approach is to adjust the class of service to the specific requirement for the user. This approach is to adjust the class of services according to the type of data being handled at a time also.

We call this approach task-based multilevel networking.

- **High speed server to server Links:**

❖ **Duplication:** It is the process of duplicating an object that the user can manipulate. There is no requirement for the duplicated object to remain synchronized with the source (or master) object.

❖ **Replication:** Replication is defined as the process of maintaining two or more copies of the same object in a network that periodically re-synchronize to provide the user faster and more reliable access to the data Replication is a complex process.

EVOLVING TECHNOLOGIES FOR MULTIMEDIA SYSTEM

- **HYPERMEDIA DOCUMENTS:**

  Hypermedia documents are documents which have text, embedded or linked multimedia objects such as image, audio, hologram, or full-motion video.

- **HYPERTEXT:**

  Hypertext systems allow authors to link information together, create information paths through a large volume of related text in documents.

  It also allows to annotate existing text, and append notes.

  It allows fast and easy searching and reading of selected excerpts.

- **HYPERSPEECH:**

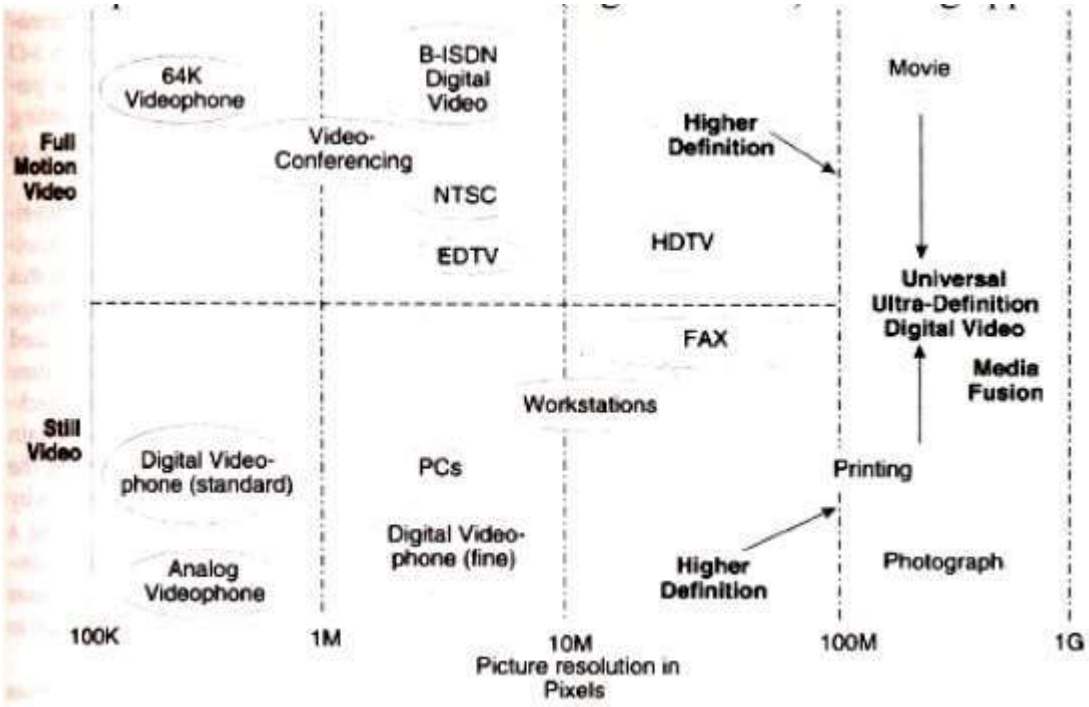
  Multimedia stimulated the development of general-purpose speech interfaces. Speech synthesis and speech recognition are fundamental requirement for hyperspeech systems. Speech recognition is nothing but converting the analog speech into a computer action and into ASCII text. Speech-recognition systems cannot segment a stream of sounds without breaks into meaningful units. The user must speak in a stilted fashion. He should make sure to interpose silence between each word.

- **HDTV AND UDTV:**

  HDTV is an acronym of High-Definition Television.

  The broadcasting standards such as NTSC, PAL, SECAM, NHK have an idea of bringing the world together on a single high-definition Television broadcasting standard.

The japanese broadcasting services developed a 1125-line, along MUSE system. A competing standard in the U.S. changed direction from analog to digital technology:A 1125-line digital HDTV has been developed and is being commercialized. NHK of Japan is trying to leapfrog the digital technology to develop ultra definition television (digital UDTV) featuring approximately 3000 lines



- **3-D TECHNOLOGIES AND HOLOGRAPHY:**

  Three-dimensional technologies are concerned with two areas: pointing devices and displays. 3-D pointing devices are essential to manipulate object in a 3-D display system. 3-D displays are achieved using holography techniques.

- **DIGITAL SIGNAL PROCESSING (DSP) :**

  Digital Signal Processing are used in applications such as digital servos in hard disk drives, and fax/modems. DSP technology is used in Digital wireless communications, such as personal communication networks (pens), wireless local area networks and digital cordless phones.

DSP Architectures and Applications

A typical DSP operating system architecture would contain the following subsystems:

- **Memory Management:** DSP architectures provide dynamic allocation of arrays from multiple segments, including RAM, SRAM and DRAM.
- **Hardware-Interrupt handling:** A DSP operating system must be designed to minimize hardware-interrupt latency to ensure fast response to real time events for applications, such as servo systems.
- **Multitasking:** DSPs need real-time kernels that provide pre-emptive multitasking and user-defined and dynamic task prioritization
- A
- B
- c

Defining objects for multimedia systems

- **TEXT:**

  It is the simplest of data types and requires the least amount of storage. Text is the base element of a relational database.

  It is also the basic building of a document.

  The major attributes of text include paragraph styling, character styling, font families and sizes, and relative location in a document

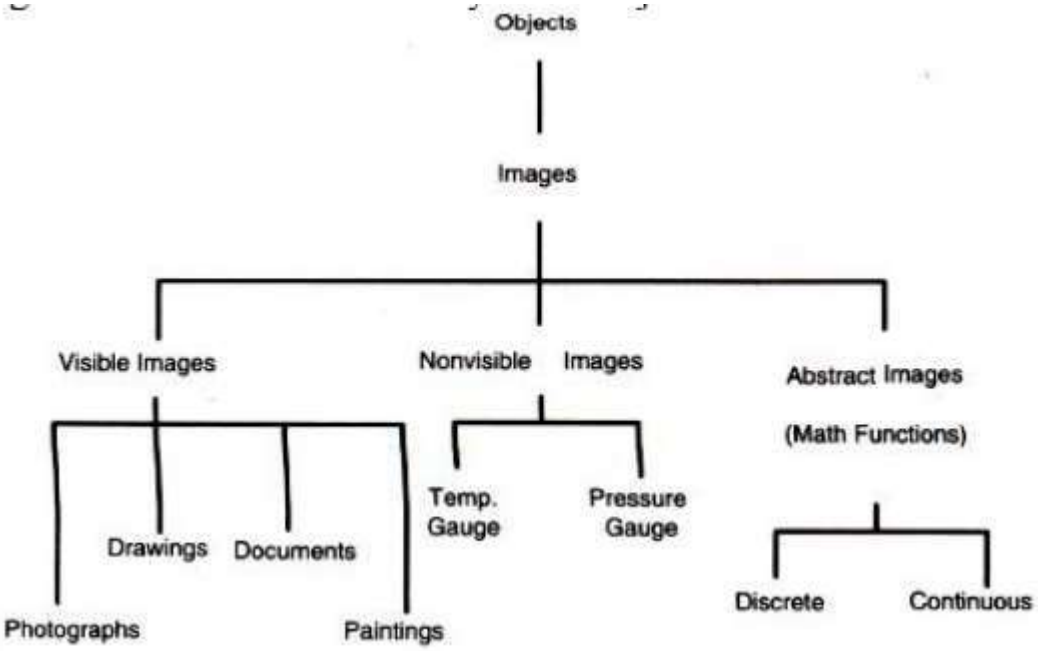
- **IMAGES:**

  Image object is an object that is represented in graphics or encoded form. Image object is a subobject of the hypermedia document object. In this object, there is no direct relationship between successive representations in time.

  The image object includes all data types that are not coded text. It do not have a temporal property associated with them.

  Thc data types such as document images, facsimile systems, fractals, bitmaps, meta files, and still pictures or still video frames are grouped together.

  Figure describes a hierarchy of the object classes



Non-Visible: This type of images are not stored as images. But they are displayed as images. Example: Pressure gauges, and temperature gauges.

Abstract: Abstract images are computer-generated images based on some arithmetic calculations. They are really not images that ever existed as real-world objects. Example of these images is fractals.

- **AUDIO AND VIDEO:**

Stored-Audio and Video objects contain compressed audio information. This can consist of music, speech, telephone conversation and voice commands. An Audio object needs to store information about thc sound clip.

Information here means length of the sound clip, its compression algorithm, playback characteristics, and any annotations associated with the original clip.

- **FULL-MOTION AND LIVE VIDEO:**

Full motion video refers to pre-stored video clips. Live video refers to live and it must be processed while it is being captured by the camera. . From a storage perspective, we should have the information about the coding algorithm used for compression. It need decoding also.

From a processing perspective, video should be presented to user with smooth and there should not be any unexpected breaks.

Hence, video object and its associated audio object must be transferred over the network to the decompression unit. It should be then played at the fixed rate specified for it.

Multimedia Data interface standards

VIDEO PROCESSING STANDARDS: -

- **INTEL'S DVI:**

DVI is an achronym of Digital Video Interface.

DVI standard is to provide a processor independent specification for a video interface. That video interface should accomodate most compression algorithms for fast multimedia displays. An example of custom-designed chip which supports DVI is Intel's i750 B. This chip is designed for enhancing low-end, software based PC video.

Advantages of the DVI Chip:

- It can operate software video processing real time.
- It can share the processing with the host CPU.
- It can handle additional vector-quantization-type algorithms in conjunction with host processing. DVI silicon chip relies on a programmable video processor. It gives potential to DVI chips to run a range of compression algorithms.

- **APPLE'S QUICKTIME:**

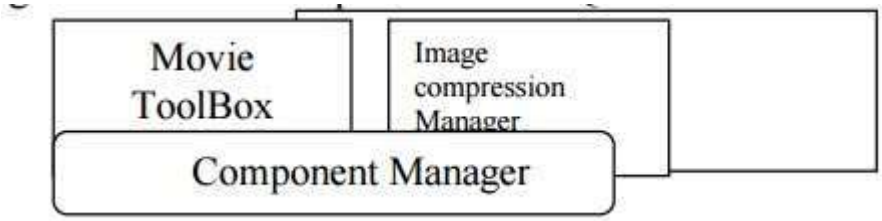
Quick Time standard is developed by Apple Computer. It is designed to Support multimedia applications. It is integrated with the operating system. Quick time refers to both the extensions to the Mac Operating system and to the compression/decompression functionality

Of the environment. Quick Time is designed to be the graphics standard for timebased graphic data types.

Quick Time's definition has been extended to include:

- System Software.
- File Formats.
- Compression! decompression algorithms.
- Human Interface Standards.

Figure Shows the components in the Quick Time Architecture.



Quick Time adjust automatically to the hardware being used by the user. MPEG is another competing standard which is comparitively higher-end, hardware-assisted standard. It can produce better resolutions at faster rates.

- **MICROSOFT'S AVI:**

AVI is an acronym for Audio Video Interleave Standard. It is similar to Apple's Quick Time. It offers low-cost, low-resolution video processing for the average desktop user. It is a layered product. A VI is scalable. It allows users to set parameter such as window size, frame rate, quality and compression algorithm through a number of dialog boxes. AVI-compatible hardware allows enhancing performance through hardware-accelerated compression algorithms such as DVI and MPEG. A VI supports several compression algorithms

MULTIMEDIA DATABASES

Images, sounds and movies can be stored, retrieved and played by many databases. In future, multimedia databases will become a main source of interaction between users and multimedia elements. **Multimedia storage and retrieval** Multimedia storage is characterized by a number of considerations. They are:

- massive storage volumes
- large object sizes
- multiple related objects
- temporal requirements for retrieval

- **Massive Data Volumes**

  A single multimedia document may be a combination of different media Hence indexing of documents, films and tapes is more complex. Locating massive data volumes requires searching through massive storage files.

  Locating and indexing systems can be understood only by a few key staff personnel. Hence it requires a major organizational eff0l1 to ensure that they are returned in proper sequence to their original storage location.

- **storage technologies**

There are two major mass storage technologies used currently for storage of multimedia documents.

(i) Optical disk storage systems. (ii) High-speed magnetic storage.

Advantages of Optical disk storage systems:

(i) Managing a few optical disk platters in a juke box is much simpler than man;lging a large magnetic disk farm. (ii) Optical disk storage is excellent storage system for off line archival of old and infrequently referenced documents for significant periods of time

- **Multimedia object storage**

Multimedia object storage in an optical medium serves its original purpose, only if it can be located fast and automatically. A key issue here is random keyed Access t6 various components of hypermedia database record. Optical media providcs very dense storage. Speed of retrieval is another consideration.

Retrieval speed is a direct result of the storage latency, size of the data relative to display resolution, transmission media and speed, and decompression efficiency. Indexing is important for fast retrieval of information. Indexing can be at multiple levels.

- **Multimedia document retrieval**

The simplest form of identifying a multimedia document is by storage platter identification and its relative position on the platter (file number). These objects can then be grouped using a database in folders (replicating the concept of paper storage in file folders) or within complex objects representing hypermedia documents.

The capability to access objects using identifiers stored in a database requires capability in the database to perform the required multimedia object directory functions. Another important application for sound and full motion video is the ability to clip parts of it and combine them with another set.

Indexing of sound and full-motion video is the subject of intense debate and a number of approaches have been used.

➢ Database Management Systems for Multimedia Systems

Since most multimedia applications are based primarily on communications technologies, such as electronic mail, the database system must be fully distributed. A number of database storge choices are available.

The choices available are:

- Extending the existing relational database management systems, (RDBMSs) to support the various objects for multimedia as binary objects.
- Extending RDBMSs beyond basis binary objects to the concepts of inheritance and classes. RDBMSs supporting these . features provide extensions for object-programming front ends and/or C++ support.
- Converting to a full fledged object oriented database that supports the standard SQL language.
- Converting the database and the application to an objectoriented database and using an object-oriented language, or an object-enabled SQL for development.

Multimedia Compression

TYPES OF COMPRESSION: -

- **LOSSLESS COMPRESSION:**

  In lossless compression, data is not altered or lost in the process of compression or decompression. Decompression generates an exact replica ofthe 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:**

  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 o~' 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 COMPRESSION

➢ PACKBITS ENCODING(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.

➢ 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 below shows the CCITT Group 3 tables showing codes or white run lengths and black run lengths.

White Run	Code	Black Run	Code
Length	**Word**	**Length**	**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

- **ADVANTAGES:**
  - It is simple to implement in both hardware and software .
  - It is a worldwide standard for facsimile which is accepted for document imaging application. This allows document imaging applications to incorporate fax documents easily.
- **DISADVANTAGES:**

➢ CCITT GROUP 3 2-D COMPRESSION:

It is also known as modified run length encoding. It is used for software based imaging system and facsimile.

It is easier to decompress in software than CCITT Group 4. The CCITT Group 3 2D scheme uses a "k" factor where the

image is divided into several group of k lines. This scheme is based on the statistical nature of images; the image data across the adjacent scanline is redundant.

If black and white transition occurs on a given scanline, chances are the same transition will occur within + or - 3 pixels in the next scanline.

Necessity of k factor·

When CCITT Group 3 2D compression is used, the algorithm embeds Group 3 1 D coding between every k groups of Group 3 2D coding, allowing the Group 3 1 D coding to be the synchronizing line in the event of a transmission error.

Therefore when a transmission error occurs due to a bad communication link, the group 3 I D can be used to synchronize and correct the error.

Data formatting for CClIT Group 3 2D

The 2D scheme uses a combination of additional codes called vertical code, pass code, and horizontal code to encode every line in the group of k lines.

The steps for pseudocode to code the code line are:

(i) Parse the coding line and look for the change in the pixel value. (Change is found at all location).

(ii) Parse the reference line and look for the change in the pixel value. (Change is found at bl location).

(iii) . Find the difference in location between bland a 1: delta = b1- al

- **ADVANTAGES:**
  - The implementation of the k factor allows error-free transmission.
  - Compression ratio achieved is better than CClTT Group 3 1 D.
  - It is accepted for document imaging applications.
- **DISADVANTAGES:**
  - It doesn't provide dense compression
  - Complex and difficult to implement in software
  - Not as efficient as Group 4 2-D.

➢ CCITT GROUP 4 2-D COMPRESSION:

CClTT Group 4 compression is the two dimensional coding scheme without the k-factor.

In this method, the first reference line is an imaginary all-white line above the top of the image.The first group of pixels (scanline) is encoded utilizing the imaginary white line as the reference line.

The new coded line becomes the references line for the next scan line. The k-factor in this case is the entire page of line. In this method, there are no end-of-line (EOL) markers before the start of the compressed data

COLOR

- **COLOR CHARACTERISTICS:**
  - **LUMINANCE OR BRIGHT:**

    This is the measure of the brightness of the light emitted or reflected by an object; it depends on the radiant, energy of the color band.
  - **HUE:**

    This is the color sensation produced in an observer due to the presence of certain wavelengths of color. Each wavelength represents a different hue.
  - **SATURATION:**

    This is a measure of color intensity, for example, the difference between red and pink.
- **COLOR MODELS:**
  - **RGB MODEL:**
  - **HIS MODEL:**
  - **YUV MODEL:**

- **JOINT PHOTOGRAPHIC EXPERTS GROUP COMPRES5ION (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.

Requirements addressed by JPEG

- The design should address image quality.
- 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.

- **Overview of JPEG Components:**

  - **Baseline Sequential Codec:**

    It consists of three steps: Formation of DCT co-efficients quantization, and entropy encoding. It is a rich compression scheme.

  - **DCT Progressive Mode:**

    The key steps of formation of DCT co-efficients and quantization are the same as for the baseline sequential codec. The key difference is that each image component is coded in multiple scans instead ofa single scan.

  - **Predictive Lossless Encoding:**

    It is to define a means of approaching lossless continuous-tone compression. A predictor combines sample areas and predicts neighboring areas on the basis of the sample areas. The predicted areas are checked against the fully loss less sample for each area.

    The difference is encoded losslessly using huffman on arithmetic entropy encoding .

- **Hierarchical Mode.**

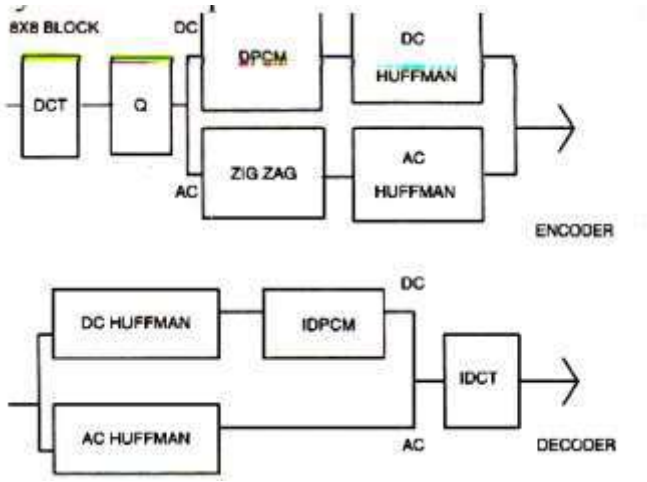
  The hierarchical mode provides a means of carrying multiple resolutions. Each successive encoding of the image is reduced by a factor of two, in either the horizontal or vertical dimension.

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.

Figure below describes a typical DCT based encoder and decoder. Symmetric Operation of DCT based Codec



❖ Quantization

Quantization is a process of reducing the precision of an integer, thereby reducing the number of bits required to store the integer, thereby reducing the number of bits required to store the integer.

The baseline JPEG algorithm supports four color quantization tables and two huffman tables for both DC and AC DCT co-efficients. The quantized co-efficient is described by the following equation:

$$\text{Quantized Co-efficient (i, j)} = \frac{DCT(i,j)}{Quantum(i,j)}$$

❖ ZigZag Sequence

Run-length encoding generates a code to represent the C0unt of zero-value OCT co-efficients. This process of run-length encoding gives an excellent compression of the block consisting mostly of zero values.

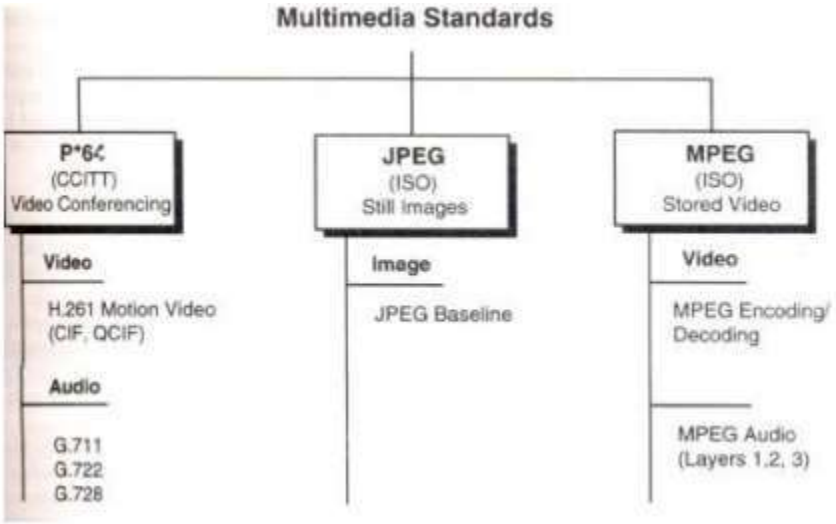
Further empirical work proved that the length of zero values in a run can be increased to give a further increase in compression by reordering the runs. JPEG came up with ordering the quantized OCT co-efficients in a ZigZag sequence

❖ Entropy Encoding

Entropy is a term used in thermodynamics for the study of heat and work. Entropy, as used in data compression, is the measure of the information content of a message in number of bits. It is represented as

Entropy in number of bits = log2 (probability of Object)

VIDEO IMAGE COMPRESSION



- **CCITT H.261 Video Coding Algorithms (P x 64)**

The linear quantizer uses a step algorithm that can be adjusted based on picture quality and coding efficiency. The H.261 is a standard that uses a hybrid of OCT and OPCM (differential pulse Code Modulation) schemes with motion estimation.

It also defines the data format. Each MB contains the OCT coefficients (TCOEFF) ofa block followed by an EOB (a fixed length end-of-block marker). Each MB consists of block data and an MB header. A GOB (Group of Blocks) consists of a GOB header. The picture layer consists of a picture header. The H.261 is designed for dynamic use and provides a fully contained organization and a high level of interactive control.

MPEG Coding Methodology

Very high level of compression 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.

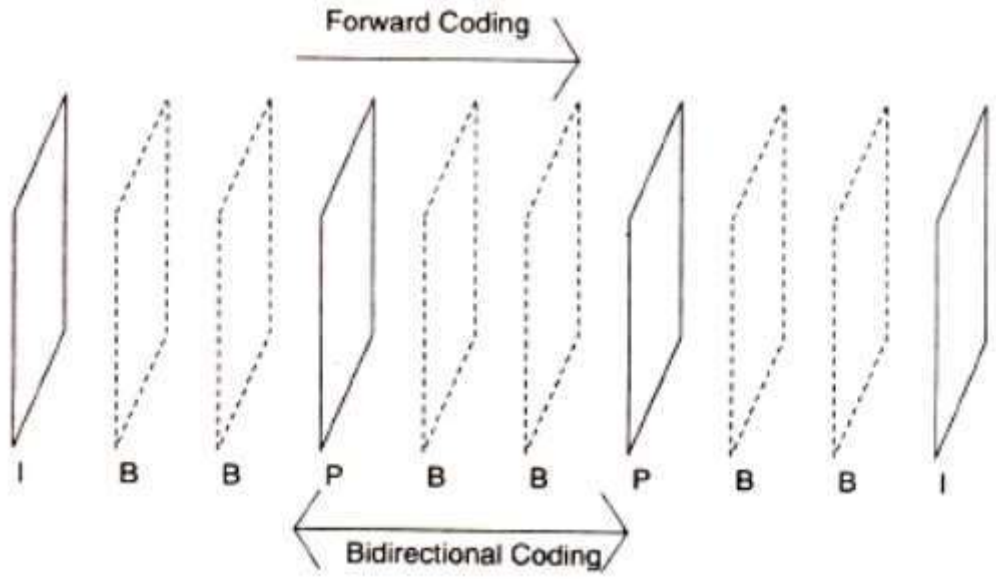
❖ **Moving Picture Types**

Moving pictures consist of sequences of video pictures or t1'ame'S that are played back a fixed number of frames per second. To achieve the requirement of random access, a set of pictures can be defined to form a group of pictures (GOP) consisting of one or more of the following three types of pictures.

- ✓ **Intra pictures (I)**
- ✓ **Unidirectionally predicted pictures (P)**
- ✓ **Bidirectionally predicted pictures (B)**

A Gap consists of consecutive pictures that begin with an intrapicture. The intrapicture is coded without any reference to any other picture in the group.

PrediGted pictures are coded with a reference to a past picture, either an intrapicture or a unidirectionally predicted picture. Bidirectionally predicted picture is never used as referencesMotion Compensation for Coding MPEG



Let us review the concept of Macroblocks and understand the role they play in compression

❖ MACRO BLOCKS

For the video coding algorithm recommended by CCITT, CIF and QCIF are diviqed into a hierarchical block structure consisting of pictures, groups of blocks (GOBs), Macro Blocks(MBs), and blocks. Each picture frame is divided into 16 x 16 blocks. Each Macroblock is composed of four 8 x 8 (Y) luminance blocks and two 8 x 8 (Cb and Cn) chrominance blocks. This set of six blocks, called a macroblock; is the basic hierarchical component used for achieved a high level of compression.

❖ Motion compensation

Motion compensation is the basis for most compression algorithms for visual telephony and full-motion video. Motion compensation assumes that the current picture is some translation of a previous picture. This creates the opportunity for using prediction and interpolation. Prediction requires only the current frame and the reference frame.

Based on motion vectors values generated, the prediction approach attempts to find the relative new position of the object and confirms it by comparing some block exhaustively. In the interpolation approach, the motion vectors are generated in relation to two reference frames, one from the past and the next predicted frame.

The best-matching blocks in both reference frames are searched, and the average is taken as the position of the block in the current frame. The motion vectors for the two reference, frames are averaged.

❖ Picture Coding Method

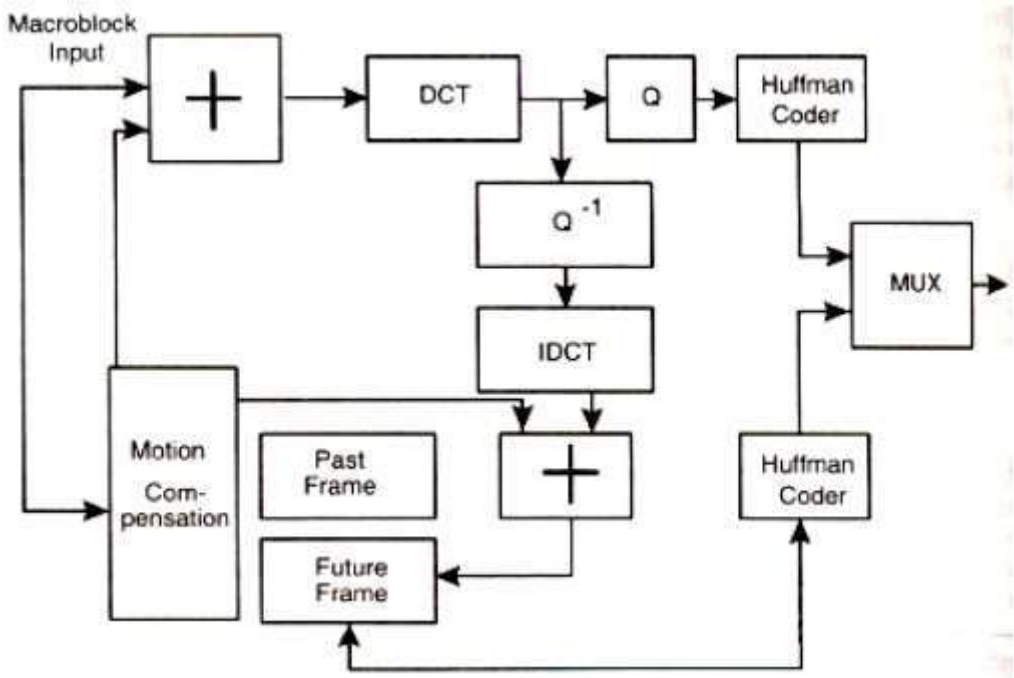
In this coding method, motion compensation is applied bidirectionally. In MPEG terminology, the motion-compensated units are called macro blocks (MBs).

MBs are 16 x 16 blocks that contain a number of 8 x 8 luminance and chrominance blocks. Each 16 x 16 macro block can be of type intrapicture, forward-predicted, backward predicted, or average.

❖ MPEG Encoder

Figure below shows the architecture of an MPEG encoder. It contains DCT quantizer, Huffman coder and Motion compensation. These represent the key modules in the encoder.

Architecture of MPEG Encoder:



AUDIO COMPRESSION

Audio consists of analog signals of varying frequencies. The audio signals are converted to digital form and then processed, stored and transmitted. Schemes such as linear predictive coding and adaptive differential. pulse code modulation (ADPCM) are utilized for compression to achieve 40-80% compression.

What is RTF?

- Stands for Rich Text Format.
- RTF documents are designed to transfer documents between word processing software.
- These files use .rtf filename extensions
- While the text formatting options are as "rich" as those used by Word, RTF files have limited page layout options.
- For example, you cannot create columns, add page numbers, headers, or footers
- The WordPad word processor included with Windows defaults to creating RTF documents.
- RTF or Rich Text File format was formalized in 1987 by Charles Simonyi, Richard Brodie, and David Luebbert from the Microsoft Word development team, for document interchange over cross-platforms.
- RTF files basically consist of commands written in ASCII codes. A single file consists of only 7-bit ASCII characters.

ADVANTAGES OF RTF

- **File Compatibility:** The most important feature of an RTF file is its compatibility with numerous operating systems and word processing applications.
- **Free from Viruses:** Unlike .doc files, .rtf files don't contain macros or viruses that can hamper the word documents.
- **File Size:**. RTF files use text-based encoding. This is advantageous, because smaller files are easier to download/upload, and also save disk storage space.

DISADVANTAGES OF RTF

- **Security:** RTF files cannot be password protected. If the file contains confidential data which needs security, then it is advisable to use Word.

File Size: If a file contains images, Word-Art, etc, the size of the file is incredibly larger than corresponding .doc file. This is disadvantageous as it consumes more download/upload time as well as occupies more disk space.

WHAT IS TIF?

- The file extension, ‘TIF’ means, ‘Tagged image file format’. And the type of file TIF is a raster file which then means it uses pixel data.
- It was developed by aldus corporation in 1986, specifically for scanners, frame grabbers, and paint/photo-editing programs.
- TIF is which is considered the highest quality format for commercial work. The TIF format is not necessarily any "higher quality" per se (the same RGB image pixels, they are what they are), and most formats other than JPG are lossless too. TIF simply has no JPG artifacts, no additional losses or JPG artifacts to degrade and detract from the original. And TIF is the most versatile, except that web pages don't show TIF files. For other purposes however, TIF does most of anything you might want, from 1-bit to 48-bit color, RGB, CMYK, LAB, or Indexed color. Most any of the "special" file types (for example, camera RAW files, fax files, or multipage documents) are based on TIF format, but with unique proprietary data tags - making these incompatible unless expected by their special software.
- TIF supports lossless LZW compression which also makes it a good archive format for photoshop documents.
- TIF is a flexible,adaptable file format for handling images and data within a single file.
- Originally designed as a common format for scanners but now a popular professional format for colour images, photos, etc

- Can hold various “tags” as well associated with the image eg: photographer, copyright, subject details, ...
- Supports several compression formats – mostly lossless Commonest is LZW, others include ZIP and JPEG and NONE!!
- **Advantages: –**
  - .tiff format isn’t compressed and it means that no data is lost;
  - The images are accurate and high quality.
  - it can handle images and data within a single file
  - What makes the format more flexible and adaptable is that it can be rendered in any classes, including gray scale, color palette, or RGB full color.
- **Disadvantages: –**
  - It’s still not supported everywhere; –
  - The size of the images is incompatible with web usage; –
  - Due to the bigger size there are some other negative consequences: it’s harder to email a tiff image and the digital cameras can’t store many images.
- **TIFF file structure:**

  TIFF files have a three-level hierarchy
  - A file header:

    Of 8 bytes
  - One or more IFDs (image file directories):

    contain codes and their data
  - Data:

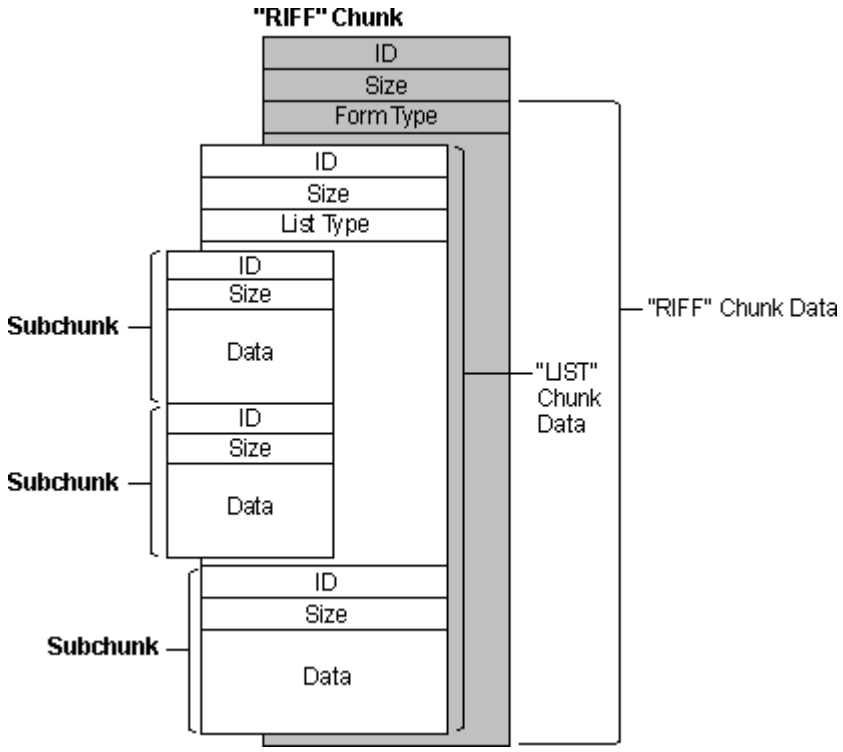
    The data information

RIFF

- The Resource Interchange File Format (RIFF) is a tagged-file specification designed for the storage of multimedia data. Data types ranging from C++ objects to full-motion video can be stored based on the RIFF specification, and new data types can be added.
- The Resource Interchange File Format (RIFF) is a generic file container format for storing data in tagged chunks. It is primarily used to store multimedia such as sound and video, though it may also be used to store any arbitrary data.
- RIFF was introduced in 1991 by Microsoft and IBM, and was presented by Microsoft as the default format for Windows 3.1 multimedia files.
- It is based on Electronic Arts' Interchange File Format, introduced in 1985 on the Commodore Amiga
- RIFF files consist entirely of "chunks". The overall format is identical to IFF, except for the endianness as previously stated, and the different meaning of the chunk names.
- RIFF (Resource Interchange File Format) is a tagged file structure for multimedia resource files. Strictly speaking, RIFF is not a file format, but a file structure that defines a class of more specific file formats, some of which are listed here as subtypes. The basic building block of a RIFF file is called a chunk. Chunks are identified by four-character codes and an application such as a viewer will skip chunks with codes it does not recognize. The basic chunk is a RIFF chunk, which must start with a second four-character code, a label that identifies the particular RIFF "form" or subtype. Applications that play or render RIFF files may ignore chunks with labels they do not recognize. Chunks can be nested. The RIFF structure is the basis for a few important file formats, but has not been used as the wrapper structure for any file formats developed since the mid 1990s.
- several file formats based on the RIFF specification.
  - CPPO- APPS Software International C++ Object Format
  - PAL Palette File Format

- RDIB Device Independent Bitmap Format
- RMID MIDI Audio Format
- RMMP Multimedia Movie File Format
- WAVE Waveform Audio Format

RIFF File Structure



The first chunk in a RIFF file must be a RIFF chunk with a chunk ID consisting of the four-character code RIFF.

The RIFF chunk contains at least one other chunk, These chunks are known as "subchunks" of the RIFF chunk.

RIFF chunks have a special code at the start of the data area that specifies the form type--the type of data in the file and its format. A RIFF form is also defined by:

A list of mandatory chunks (which must be present to make up a valid file of the aforementioned form type).

A list of optional chunks, some or all of which may be present.

Optionally, an order in which to store some or all of the chunks.

MIDI

- Musical Instrument Digital Interface (MIDI)
- it is a standard protocol for the interchange of musical information between musical instruments, synthesizers, keyboard controllers, sound cards, computers and all other electronic instruments.
- A MIDI file is very small, often as small as 10 KB for a 1-minute playback, This is because it doesn’t contain audio waves like audio file formats do, but instructions on how to recreate the music.
- The MIDI is used as the foundation for Internet music playback
- A MIDI file is a sequence of chunks.
- A chunk has a 4-character type, a 4-byte length code (in MSB format), and some data
- These chunk have the same general format as the chunks used by AIFF, IFF and WAVE;
- MIDI files are a lot smaller than MP3s,WMAs,WAVs.
- Extension is .MID
- MIDI is nothing more than data -- a set of instructions. MIDI data contains a list of events or messages that tell an electronic device how to generate a certain sound.

Structure of MIDI

- MIDI files are structured into chunks.
- Each chunk consists of:
  - A 4-byte chunk type (ASCII)
  - A 4-byte length (32 bits, msb first)
  - length bytes of data
- There are two types of chunks:
  - Header Chunks
  - Track Chunks
- A MIDI file consists of a single header chunk followed by one or more track chunks.

Advantages of MIDI

- The music can be modified conveniently.
- Generally used in games – as background music
- General MIDI specifies 175 instruments

JPEG

- JPEG, which stands for Joint Photographic Experts Group.
- Jpeg is a commonly used method of compression for digital images .
- JPEG (.jpeg or .jpg) format is used when a smaller file size is more important than maximum quality.
- It uses a lossy compression algorithm.
- JPEG typically achieves 10:1 compression with little perceptible loss in image quality.
- uses a combination of discrete cosine transform, quantization, run-length and Huffman coding.
- Commonly used For web usage, where the amount of data used for an image is important.
- also the most common format saved by digital cameras.

ADVANTAGES OF JPEG

There are several advantages of JPEG image format:

- The JPEG format has been in use for a long time and is extremely portable;
- The JPEG format is compatible with almost every image processing application;
- The JPEG format is compatible with most of the hardware devices.
- JPEG format can be used to store high-resolution fast-moving images.
- Size of JPEG images can be reduced and compressed.

DISADVANTAGES OF JPEG

- it doesn’t render the sharp contrasts well;
- It doesn’t support animation;
- It’s not recommended to save a JPEG image multiple times; each new save means the information discard.

AVI

- **AVI** stands for **Audio Video Interleaved**
- AVI is a multimedia container format introduced by Microsoft in November 1992 as part of its Video for Windows software.
- **AVI** is a file format used to store audio and/or video information digitally for playback purposes.
- AVI works as a container to hold videos and audios created and compressed using several codecs like DivX and XviD.
- It uses the .avi filename extension
- It is the nominal standard for personal computers using Windows but can also run in Macintosh and Linux.
- AVI does not need substantial compression to store files and requires more space compared to other video formats like MOV and MPEG.

STRUCTURE OF AVI

- AVI file format structure is based on the Resource Interchange File Format (RIFF).
- This file format allocates all the information in a particular file into blocks or "chunks". Each chunk can be tagged as AVI.
- The AVI file itself then is divided into parts:
  - **Header:** contains metadata of the file, such as the file size, frame rates etc.
  - **Chunk:** store actual audio and video information.

MPEG

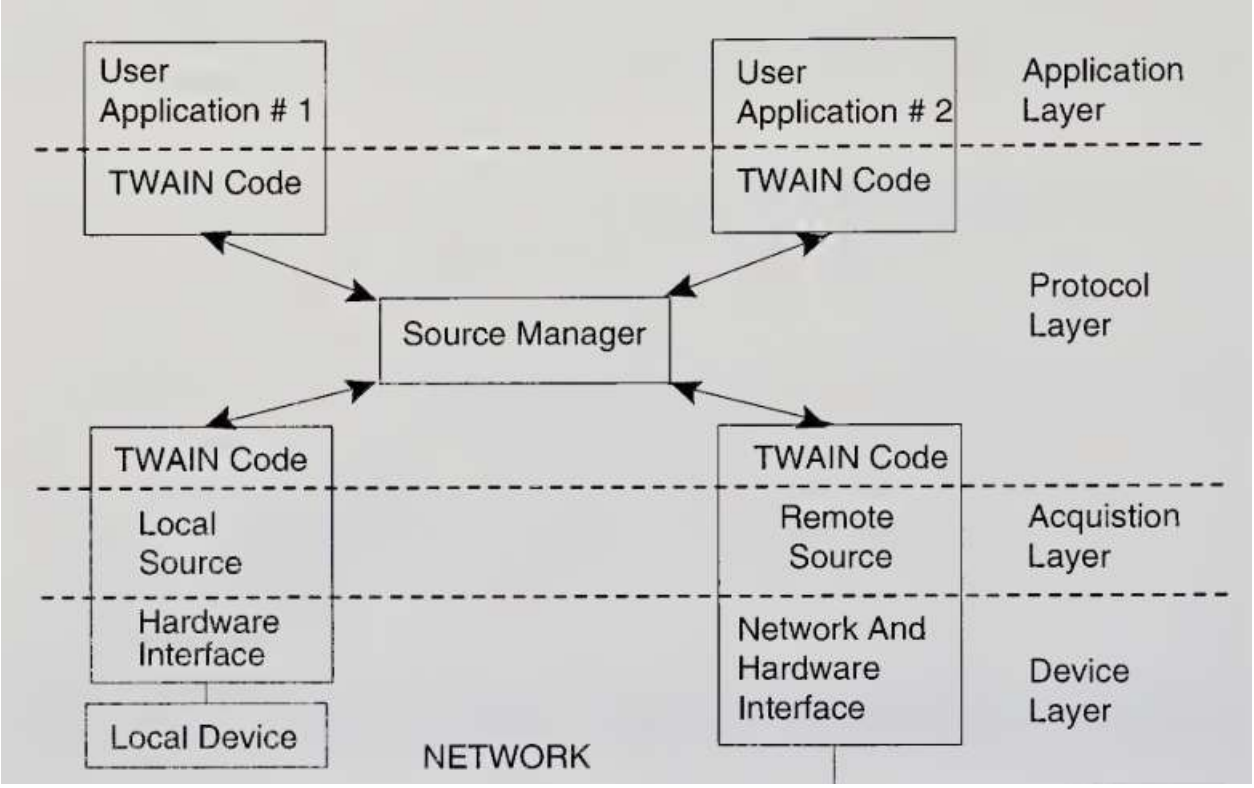
- Short for Moving Picture Experts Group
- MPEG is an organization that develops standards for encoding digital audio and video.
- It works with the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) to ensure media compression standards are widely adopted and universally available.
- Using MPEG compression, the file size of a multimedia file can be significantly reduced with little noticeable loss in quality.
- MPEG achieves its high compression rate by storing only the changes from one frame to another, instead of each entire frame.
- The video information is then encoded using a technique called Discrete Cosine Transform (DCT).
- MPEG uses a type of lossy compression.
- The MPEG organization has produced a number of digital media standards since its inception in 1998. Examples include:
    - **MPEG-1** is initial video and audio compression standard. The video quality of this standard is almost as good as a VHS tape.
    - **MPEG-2** compared with MPEG-1, it has better quality of coding, multi-channel sound and higher image resolution. Due to all this MPEG-2 became standard in digital television and DVD industry.
    - **MPEG Audio Layer-3** is used for audio compression and creates almost CD quality sound. Previous versions were MPEG Audio Layer 1 and 2. Today most people know this format as MP3.

- **MPEG-4** was created to stream DVD quality video at lower data rates and smaller file sizes. MPEG-4 supports video/audio "objects", 3D content, sprites, text and other media types. MPEG-4 has become extremely popular due to the ability to fit a two-hour movie to a CD file retaining comparatively high quality.

TWAIN

- TWAIN is a widely-used program that lets you scan an image (using a scanner) directly into the application (such as PhotoShop) where you want to work with the image.
- The standard interface was designed to allow applications to interface with different types of input devices, such as scanners, digital still cameras, video cameras, and so on, using a generic TWAIN interface without creating a device-specific driver.

TWAIN ARCHITECTURE



- **Application Layer**
  - It sets up a logical connection with a device.
  - TWAIN does not impose any rules on the design of an application.

- It does set guidelines for the user interface to select sources from a given list of sources and also specifies user interface guidelines to acquire data from the selected sources.

➢ **The Protocol Layer**

- The application layer interfaces with the protocol layer.
- The protocol layer is responsible for communications between the application and acquisition layers.
- It specifies the services provided by sources, including: establishing a session with a device, data creation, and data transfer.
- The heart of the protocol layer is the Source Manager. It manages all sessions between an application and the sources, and monitors data acquisition transactions.

➢ **The Acquisition Layer**

- Control of the device.
- Acquisition of data from the device.
- Transfer of data in agreed (negotiated) format. This can be transferred in native format or another filtered format.
- Provision of a user interface to control the device

➢ **The Device Layer**

- The purpose of the device driver is to receive software commands and control the device hardware accordingly.
- This is generally developed by the device manufacturer and shipped with the device.

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 fornlat by analog-to-digital converter (ADC) using sampling process.
- **Sampling** - 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.
- A **sampling rate** is the number of times the analog sound is taken per second.
- A higher sampling rate implies that more samples are taken during the given time interval and ultimately, the quality of reconstruction is better.
- The sampling rate is measured in terms of Hertz, Hz in short, which is the term for Cycle per second.

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:** 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 tmly represents the signal.
- **Conversion Speed:** It is a speed of ADC to convert the analog signal into Digital signals. It must be fast enough.

VOICE RECOGNITION

Voice Recognition Systems can be classified into three types.

- **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.

- **Connected-Word Speech Recognition**

Connected-word speech consists of spoken phrase consisting of 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 system can be divided into three sections:

- A section consisting of digitization, amplitude normalization, time nominalization and parametric representation.
- Second section consisting of segmentation and labeling of the speech segment into a symbolic string based on a knowledgebased or rule-based systems.
- The final section is to match speech segments to recognize word sequences.

VIDEO AND IMAGE ANIMATION

While the data representation of still video images may be the same as for document images, the input technology is very different. Still video uses specialized circuitry for selecting the video input source and capturing video frames.

VIDEO FRAME GRABBER ARCHITECTURE

A video frame grabber is used to capture, manipulate and enhance video images.

A video frame grabber card consists of the following:-

- **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 aTV 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 colour.
- **Input lookup table**: The input lookup table along with the arithmetic logic unit (ALU) allows performing image processing functions on a pixel basis and an image frame basis. 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 storage buffer to store image for image processing and display.
- **Video Compression-Decompression**: The video compression-decompression processor is used to compress 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 lookup table generates either an 8-bit pixel value for monochrome or a 24-bit pixel value for colour.
- **SVGA Interface:** This is an optional interface for the frame grabber. The frame grabber can be designed 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 DAC is mixed to generate overlay output signals. The primary components involved include the display image frame buffer and the display SVGA buffer.

Image Animation Techniques

Animation: Animation is an illusion of movement created by sequentially playing still image frames at the rate of 15-20 frames per second.

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.

FULL MOTION VIDEO

- Most modem cameras use a CCD for capturing the image.
- HDTV video cameras will be all-digital, and the capture method will be significantly different based on the new NTSC HDTV Standard.

Full-Motion Video Controller Requirements

Video Capture Board Architecture:

A full-motion video capture board is a circuit card in the computer that consists of the following components:

(i) Video input to accept video input signals.

(ii) S- Video input to accept RS 170 input.

(iii) Video compression-decompression processor to handle different video compression-decompression algorithms for video data.

(iv) Audio compression-decompression processor to compress and decompress audio data.

(v) Analog to digital converter.

(vi) Digital to analog converter.

(vii)Audio input for stereo audio LINE IN, CD IN.

(viii) Microphone.

Video Channel Multiplexer:

It is similar to the video grabber's video channel multiplexer.

Video Compression and Decompression:

The video compression and decompression processor contains multiple stages for compression and decompression. The stages include forward discrete cosine transformation and inverse discrete cosine transformation, quantization and inverse quantization, ZigZag and Zero run-length encoding and decoding, and motion estimation and compensation.

Audio Compression: MPEG-2 uses adaptive pulse code modulation (ADPCM) to sample the audio signal. The method takes a difference between the actual sample value and predicted sample value. The difference is then encoded by a 4-bit value or 8-bit value depending upon the sample rate

STORAGE AND RETRIVAL TECHNOLOGY

Multimedia systems require storage for large capacity objects such as video, audio and images.

- **MAGNETIC MEDIA TECHNOLOGY**
  - Magnetic hard disk drive storage is a mass storage medium.
  - It has high capacity and available in low cost.
- **HARD DISK TECHNOLOGY**
  - **ST506 and MFM Hard drives**: It is developed by seagate. It is used to control platter speed and the movement of heads for a drive.
  - **ESDI Hard Drive**: ESDI (Enhanced Small Device Interface) was developed by a consortium of several manufacturers. It converts the data into serial bit streams.
  - **IDE**: Integrated Device Electronics (IDE) contains a,n integrated controller with drive.
- **RAID (Redundant Array of Inexpensive Disks)**
  - It is an alternative to mass storage for multimedia systems that combines throughput speed and reliability improvements.
  - RAID is an array of multiple disks.
  - In RAID the data is spread across the drives.
  - It achieves fault tolerance, large storage capacity and performance improvement.

- There are six levels of RAID available.
  - **RAID Level 0 Disk Striping**
  - **RAID Level 1 Disk Mirroring**
  - **RAID Level 2, - Bit interleaving of Data**
  - **RAID Level-3 Parallel Disk Array**
  - **RAID Level-4 Sector Interleaving**
  - **RAID Level-5 Block Interleaving**

- **Optical Media**

Optical Media can be classified by technology as follows:

  - CD-ROM - Compact Disc Read Only Memory
  - WORM - Write Once Read Many
  - Rewritable - Erasable
  - Multifunction - WORM and Erasable.

- **Magneto-Optical Technology**
  - It uses a combination of magnetic and laser technology to achieve read/write capability.
  - The disk recording layer uses a weak magnetic field to record data under high temperature.
  - High temperature is achieved by laser beam.
  - When the beam is on, it heats the spot on the magneto optical disk to its curie temperature.

- The rise in temperature makes the spot extra sensitive to the magnetic field of bias field.
- Magneto-optical drives require two passes to write data; in the first pass, the magneto optical head goes through an erase cycle, and in the second pass, it writes the data.

➢ **Phase change Rewritable optical Disk**

In phase change technology the recording layer changes the physical characteristics from crystalline to amorphous and back under the influence of heat from a laser beam.

Multimedia Authoring Systems

Multimedia authoring systems are designed with two primary target users:

- Professionals who prepare documents, audio or sound tracks, and full motion video clips for wide distribution.
- Average business users preparing documents, audio recordings, or full motion video clips for stored messages' or presentations.

Types of Multimedia Authoring Systems

- **Dedicated Authority Systems**
  - Dedicated authoring systems are designed for a single user and generally for single streams.
  - Designing this type of authoring system is simple, but if it should be capable of combining even two object streams, it becomes complex.
  - The authoring is performed on objects captured by the local video camera and image scanner or an objects stored.
- **TimeLine –based authoring**
  - In a timeline-based authoring system, objects are placed along a timeline.
  - The timeline can be drawn on the screen in a window in a graphic manner, or using a script.
  - But the user must specify a resource object and position it in the timeline.
  - On playback, the object starts playing at that point in the time Scale.
  - In most timeline-based approaches, once the multimedia object has been captured in a timeline, it is fixed in location and cannot be manipulated easily.

- So, a single timeline causes loss of information about the relative time lines for each individual object.

➢ **Structured Multimedia Authoring**

This approach consists of two stages:

- ❖ The construction of the structure of a presentation.
- ❖ Assignment of detailed timing constraints.

A successful structured authoring system must provide the following capabilities for navigating through the structure of presentation.

- ❖ Ability to view the complete structure.
- ❖ Maintain a hierarchy of objects.
- ❖ Capability to zoom down to any specific component.
- ❖ View specific components in part or from start to finish.
- ❖ Provide a running status of percentage full of the designated length of the presentation.
- ❖ Clearly show the timing relations between the various components.
- ❖ Ability to address all multimedia types including text, image, audio, video and frame based digital images.

➢ **Programmable Authoring Systems:**

programmable authoring system has improved in providing powerful functions based on image processing and analysis and embedding program interpreters to use image-processing functious.

The capability of this authoring system is enhanced by Building user programmability in the authoring tool to perform the analysis and to manipulate the stream based on the analysis results and also manipulate the stream based on the analysis results. The programmability allows the following tasks through the program interpreter rather than manually. Return the time stamp of the next frame. Delete a specified movie segment. Copy or cut a specified movie segment to the clip board . Replace the current segment with clip board contents.

- **Multisource Multi-user Authoring Systems**
  - We can have an object hierarchy in a geographic plane; that is, some objects may be linked to other objects by position, while others may be independent and fixed in position".
  - We need object data, and information on composing it. Composing means locating it in reference to other objects in time as Well as space.
  - Once the object is rendered the author can manipulate it and change its rendering information.
  - If there are no limits on network bandwidth and server performance, it would be possible to assemble required components on cue at the right time to be rendered.
  - A multi user authoring system must provide resource allocation and scheduling of multimedia objects.

Telephone Authoring systems

- ❖ This can be used as a reading device by providing fill text to-speech synthesis capability so that a user on the road can have electronic mail messages read out on the telephone.
- ❖ The phone can be used for voice command input for setting up and managing voice mail messages. Digitized voice clips are captured via the phone and embedded in electronic mail messages.
- ❖ As the capability to recognize continuous speech is deploy phones can be used to create electronic mail messages where the voice is converted to ASCII text on the fly by high-performance voice recognition engines.

Hypermedia Application Design

- ➢ hypermedia applications are applications consisting of compound objects that include
- ➢ multimedia objects.
- ➢ An authoring application may use existing multimedia objects or call upon a media editor to create new objects.
- ➢ The primary role performed by the authoring application is to structure multimedia documents or database records by coordinating the actions of media editors and combining them with existing objects.
- ➢ The authoring application does not manipulate the media directly.
- ➢ The authoring application does allow the user to combine different media streams on a timeline.

User Interface Design

User interface design for multimedia applications is more involved than for other applications due to the number of types of interactions with the user. Consequently, rather than a simple user interface dialogue editor, multimedia applications need to use four different kinds of user interface development tools. We can classify these as the following:

1. Media editors
2. An authoring application
3. Hypermedia object creation
4. Multimedia object locator and browser

One would think that developing a user interface should be fairly simple and intuitive. However, that is not the case. There is never a "correct" UI; a UI can be good or bad. The cor¬ rectness of a user interface is a perception of a user. Different users have different perceptions of what is correct. But a good user interface can be designed by following some structured design guidelines, as follows:

1. Planning the overall structure of the application
2. Planning the content of the application
3. Planning the interactive behavior
4. Planning the look and feel of the application

HYPER MEDIA MESSAGING

- Messaging is one of the major multimedia applications.
- Messaging started out as a simple text-based electronic mail application.
- Multimedia components have made messaging much more complex.
- We see how these components are added to messages.

Mobile Messaging

Mobile messaging represents a major new dimension in the users interaction with the messaging system. With the emergence of remote access from users using personal digital assistants and notebook computers, made possible by wireless communications developments supporting wide ranging access using wireless modems and cellular telephone links, mobile messaging has significantly influence messaging paradigms.

Hypermedia messaging is not restricted to the desktops; it is increasingly being used on the road through mobile communications in metaphors very different from the traditional desktop metaphors.

Hypermedia Message Components

- The user may have watched some video presentation on the material and may want to attach a part of that clip in the message. While watching it, the user marks possible quotes and saves an annotated copy.
- Some pages of the book are scanned as images. The images provide an illustration or a clearer analysis of the topic
- The user writes the text of the message using a word processor. The text summarizes the highlights of the analysis and presents conclusions.

These three components must be combined in a message using an authoring tool provided by the messaging system. The messaging system must prompt the user to enter the name of the addressee for the message.

Creating Hypermedia Messages

- Hypermedia message is a complex collection of a variety of objects.
- It is an integrated message consisting of text, rich text, binary files, images, bitmaps. voice and sound, and full motion video.
- Creating of a hypermedia message requires some preparation. It requires the following steps:
    - Planning
    - Creating each component
    - Integrating components

The planning phase for preparing the hypermedia can include any of the following:

- ❖ A text report prepared in a word-processing system.
- ❖ A spreadsheet in a spreadsheet program.
- ❖ Some diagrams from a graphics program.
- ❖ Images of documents.
- ❖ Sound dips.
- ❖ Video clips.

Integrated Multimedia Message Standards

- **Vendor Independent Messaging (VIM)**
  - The VIM interface makes mail and messages services available through a well-defined interface.
  - A messaging service enables its clients to communicate with each other in a store-and-forward manner.
  - VIM-aware applications may also use one-or-more address books.
  - Address books are used to store information about users, groups, appl ications, and so on. **VIM**

Messages:

The objects transported by a messaging system are called messages. The message, along with the address is sent to the messaging system.

Message Definition:

Each message has a message type. The message type defines the syntax of the message and the type of information that can be contained in the message.

Mail Message:

It is a message of a well-defined type that must include a message header and may include note parts, attachments, and other application-defined components.

Message Delivery:

If message is delivered successfully, a delivery report is generated and send to the sender of the message if the sender requested the delivery report. If a message is not delivered, a non-delivered report is sent to the sender.

Message Container:

Multiple users or applications can access one message container. Each message in a message container has a reference number associated with it for as long as the message remains stored in the message container.

VIM Services: The VIM interface provides a number of services Some of them are:

- ❖ Electronic message composition and submission.
- ❖ Electronic message sending and receiving.
- ❖ Message extraction from mail system.
- ❖ Address book services.

- **MAPI Support (Multimedia Application Programmable Interface)**

MAPI provides a layer of functionality between applications and underlying messaging systems.

The primary goals of MAPI are:

Separate client applications from the underlying messaging services.

Make basic mail enabling a standard feature for all applications.

Support message-reliant workgroup applications.

- **MAPI Architecture:**

MAPI Architecture provides two perspectives

A client API

A service provider interface.

The Client API provides the link between the client applications andMAPI. The service provider interface links MAPI to the messaging system.

- **Telephony API (TAPI)**

TAPI standard has been defined by Microsoft and Intel. The telephone can be used for reading e-mail as well as for entering e-mail messages remotely.

Integrated Document Management

- It provides Integrated document management for messaging Lotus Notes messaging system.
- The user can attach embed or link a variety of multimedia objects.
- When document is forwarded to other users, all associated multimedia objects are also forwarded and available to the new receivers of the forward message.

Multimedia Object Server and Mail Server Interactions:

- The mail server is used to store in e-mail messages.
- It consists of a file server with mail files for each user recipient.
- This file server act as a mail box.
- All received mail is dropped in the user's mail file.
- The user can review or delete these mails.
- When mail messages include references to multimedia objects, mail file contains only link information.