Lecture 1

Introduction to computer graphic

Is a set of all means to deal with pictures which can be seen by man then became aware of stimulus in a certain part of his brain (visual perception), he analysed and interception of pictures depends on the cultural environment.

A graphic system is defined as any collection of H/W and S/W design to make it easy to use graphic input device and output device in a computer program.

Picture: is the reflection of a three dimensional scene on the plane.

Luminous for any point based on two factors:

1. Incident light
2. Reflectance light

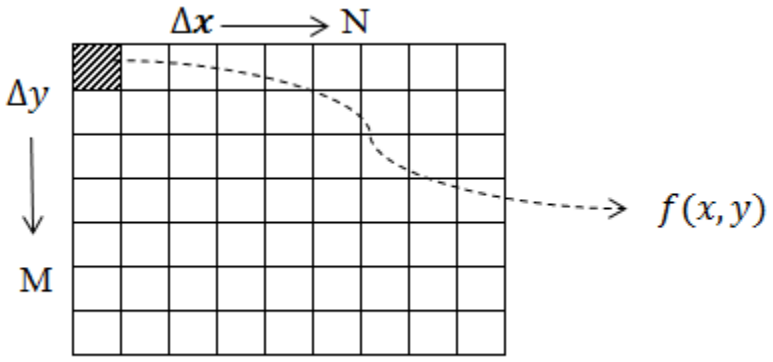
$$F(x,y) = i(x,y) * r(x,y), \quad 0 <= i(x,y) <= \infty, \quad 0 \le r(x,y) \le 1$$

$$L_{min} \le f(x,y) \le L_{max}$$

The term image refers to a two dimensional light intensity function $f(x,y)$, where x and y denote spatial coordinates and the value of f at any point (x,y) is proportional to the brightness (or gray level) of the image at that point.

A digital image can be considered a matrix whose row and column. The elementsof such a digital array are called image elements, picture elements, pixels.

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1)\ldots\ldots\ldots\ldots f(0, n-1) \\ f(1,0) & f(1,1)\ \ldots\ldots\ldots\ldots f(1, n-1) \\ & \vdots \\ f(M-1, 0) & \ldots\ldots\ldots\ldots\ldots\ldots (M-1, N-1) \end{bmatrix}$$



$x = 0 \rightarrow M - 1$ *rows*

$y = 0 \rightarrow N - 1$ *Column*

Monochrome image (0 -255) = 256 (gray level)

$256 = 2^8$ = 8 bit = 1 byte

Image of two bits (0,1) = binary image

$$no.\ of\ levels = 2^n$$

10 gray level (0 – 9) , 9= 1001 , 4 bits

That means, the image sends with 4 bits for each pixel

Max of 4 bits = 1111 = 15

Digitization

Digitization is the process of converting information into a digital format. In this format, information is organized into discrete units of data (called bits) that can be separately addressed (usually in multiple-bit groups called bytes). This is the binary data that computers and many devices with computing capacity (such as digital camera and digital hearing aid) can process.

Text and images can be digitized similarly: a scanner captures an image (which may be an image of text) and converts it to an image file, such as a bitmap. An optical character recognition (OCR) program analyse a text image of light and dark areas in order to identify each alphabetic letter or numeric digit, and converts each character into ASCII code.

Audio and video digitization uses one of many analog to digital conversion processes in which a continuously variable (analog) signal is changed, without altering its essential content, into a multi-level (digital) signal.

The process of sampling measures the amplitude (signal strength) of an analog waveform at evenly spaced time markers and represents the samples as numerical values for input as digital data.

Digitization information makes it easier to preserve, access and share. For example, an original historical document may only be accessible to people who

visit its physical location, but if the document content is digitized, it can make available to people worldwide.

To be suitable for computer processing, an image function $f(x,y)$ must be digitized both spatially and amplitude.

The digitization process requires decisions about values for N , M and G (is the no. of gray levels) allowed for each pixel.

$$N = 2^n \ , M = 2^n \ , \quad G = 2^m$$

$$b = N * M * m$$

$$if \ \ N = M:$$

$$b = N^2 M$$

Example:-

128 x 128 image dimension, with 64 gray level.

b= $(128)^2$ * 6 = 98304

b= $\frac{N^2 m}{8}$

b: the no. of bits required to store a digitized image.

Application of CG

1. Line drawing
2. Animation
3. Picture processing
4. Pattern recognition
5. Games, planning and business

Computer Graphic Mode

There are two modes of CG:

1. Passive mode:

   The passive mode is non-interactive.

   In this mode the system operates in a batch environment. The input device is card disk, tape file and output device is line printer, plotter.

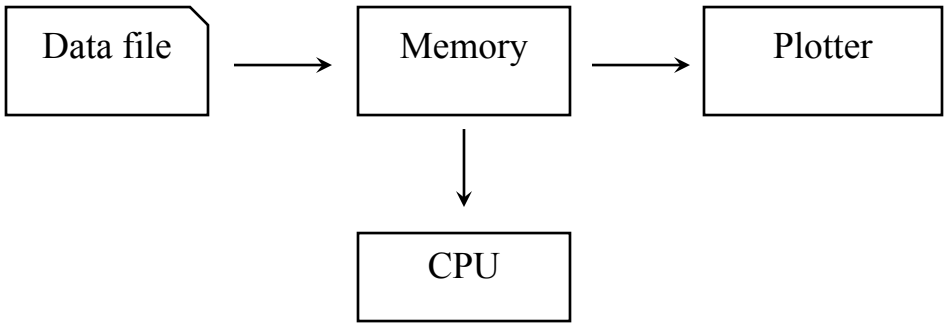


Figure 1. Passive graphic mode

2. Interactive mode:

   The user and computer are interactive all converse on-line.

   The user should communicate quickly and effectively with the computer.

   The input device used must provide for:

   - Issuing commands
   - Position symbols on the screen
   - Specifying item to change or delete

The input devices are keyboard, joystick, mouse and light pen.

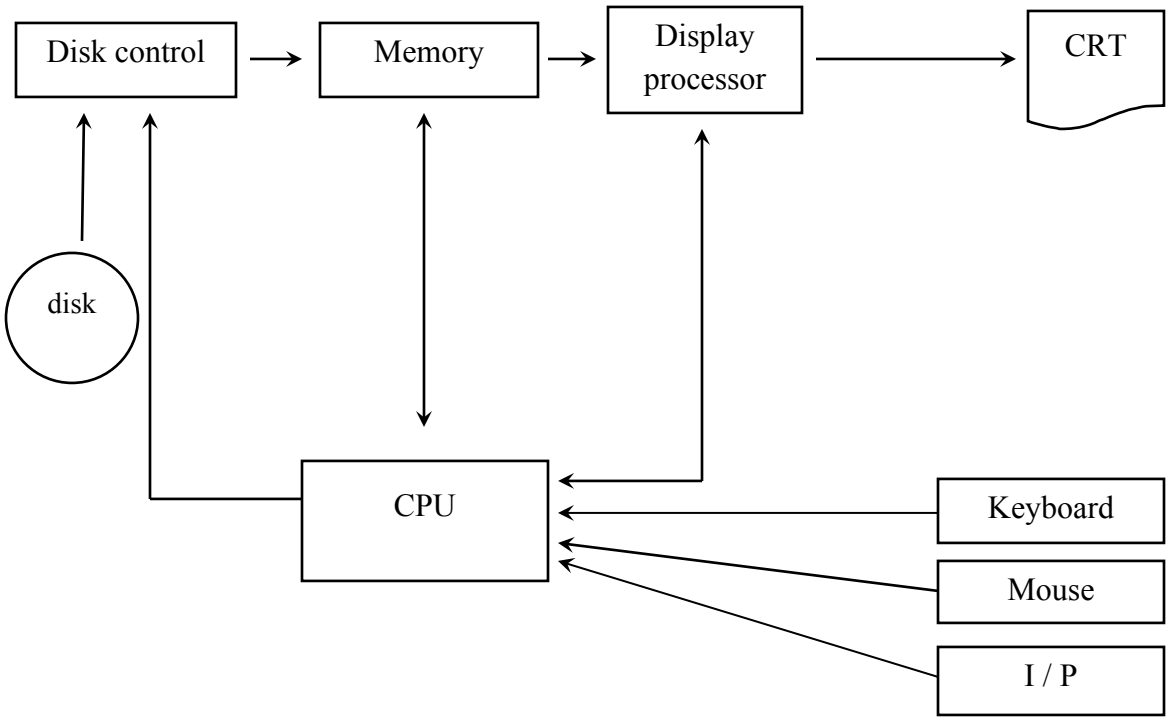


Figure 2. Interactive mode

Lecture 2

Display Devices

Video Display Generation

Most video display screens are the same type of cathode ray tubes (CRT) that are used in home television.

The electron gun contains a cathode that when heated emitted a beam of negatively charged electrons towards a positively charged phosphor-coated screen.

Along the way, the electron beam passes through the focusing and deflecting system, which consists of an electrostatic or magnetic field.

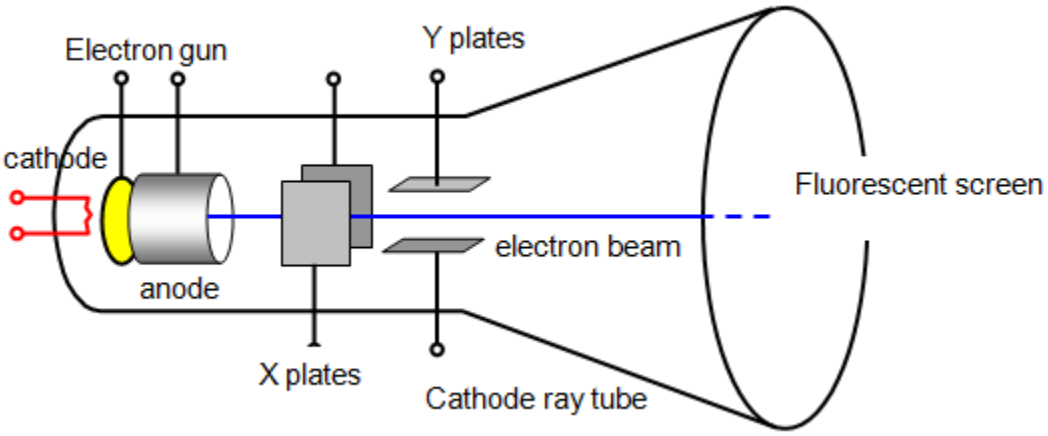


Figure 1. Cathode Ray Tube (CRT)

The focusing system:

Concentrates the beam so that by the time the electron reaches the screen, they have converged to a small dot.

The deflection system:

Which consists of two pairs of deflection plates (horizontal and vertical), directs the electron beam to any point on the screen. Both pairs of plates have equal voltages but with opposite signs, that is, one has a positive charge, the other negative charge.

When negatively charged electron passes the plates, it is attracted to the positively charged plate, resulting in a deflection of electrons.

- The amount of deflection depends on the magnitude of the voltage on the plates.
- By varying the voltage on the horizontal and vertical plates, the electron beam can strike any point on the screen.
- When the focused electron beam strikes the screen, the phosphor emits a spot of visible light whose intensity depends on the no. of electrons in the beam.
- A blank spot on the screen corresponds to no (or very few) electrons have been sent to that location
- The light on the display screen starts to fade as soon as the beam moves to another location. The duration of this light, called persistence, depends on the type of phosphor that coats the screen.
- Normally, visible light lasts for only a fraction of a second.
- In order to give the viewer the appearance of a continuous flicker-free image, each illuminated dot on the screen must be intensified many times per second.

This type of video display is called a refresh CRT.
Two types of refresh CRT_s are available: raster scan and random vector.

Although both are currently in use, the raster-scan system is preferred for most microcomputers and for applications that require colour or shade.

Color CRT

- A colour CRT has a three electron guns, one for each of the three primary colors: red, green and blue. A delta-gun system arranges the three guns in a

triangular pattern with a perforated metal grid or shadow mask placed between the guns and the face of the display screen.

- Each pixel is composed of a triangular pattern of a red, green and blue phosphor dot.

- The holes in the shadow mask are aligned so that each electron gun excites its corresponding phosphor dot.

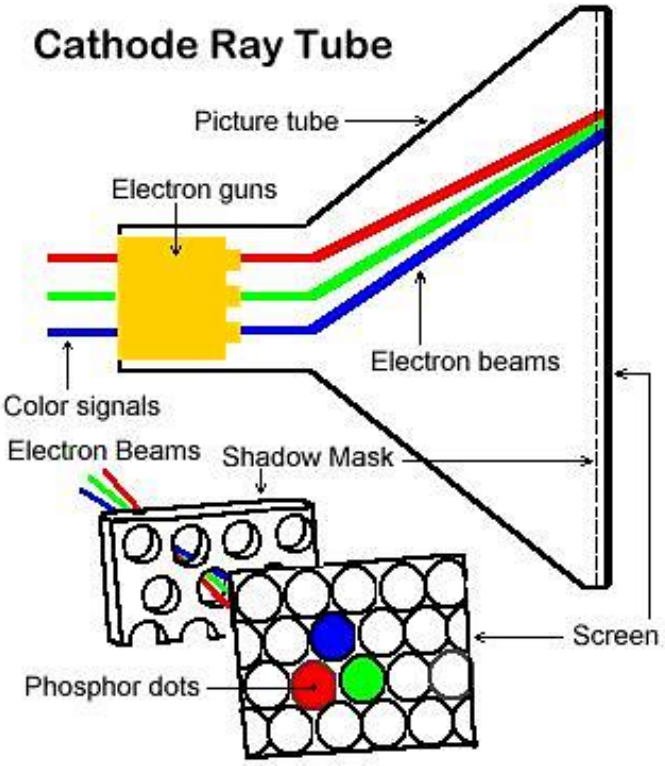


Figure 2. Color CRT

Raster Scan Display

- This video display screen used by most microcomputers is divided into very small dots.
- These dots are referred to as picture elements or pixels.
- CRT screen consists of a grid of vertical and horizontal lines, where each horizontal line is made up of pixels.

- These horizontal lines are called raster-scan lines and video display is referred as a raster –scan display.
- The quality of a raster display is often described based on its resolution.
- The great resolution means great details of an image.
- Low resolution displays have about 300 scan lines, each containing about 400 pixels.
- High resolution displays have at least 1000 scan lines with over 1000 pixels per line.

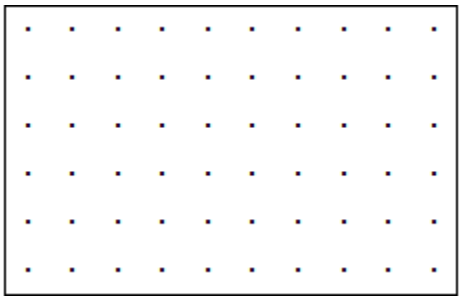


Figure 3. Raster scan display

Raster Scan System

A raster-scan system displays the image on the CRT in a certain fixed sequence as explained below:

1. Initially the vertical and horizontal deflection plates are set to display a pixel at the upper-left corner of the screen.
2. The voltage on the horizontal plates is then continuously changed steering the electron beam across the scan line.
3. When the beam at the end of the scan line, the electron gun is turned off (no visible beam) and both pairs of deflection plates are set to send the beam to the leftmost pixel in the second raster-scan line.
4. The blanking of the video signal is called the horizontal retrace.
5. This left-to-right scanning pattern continues until the electron beam reaches the bottom-right corner of the screen. At that time, the beam was turned off and is steered to the top-left corner of the screen, a process called vertical retrace.

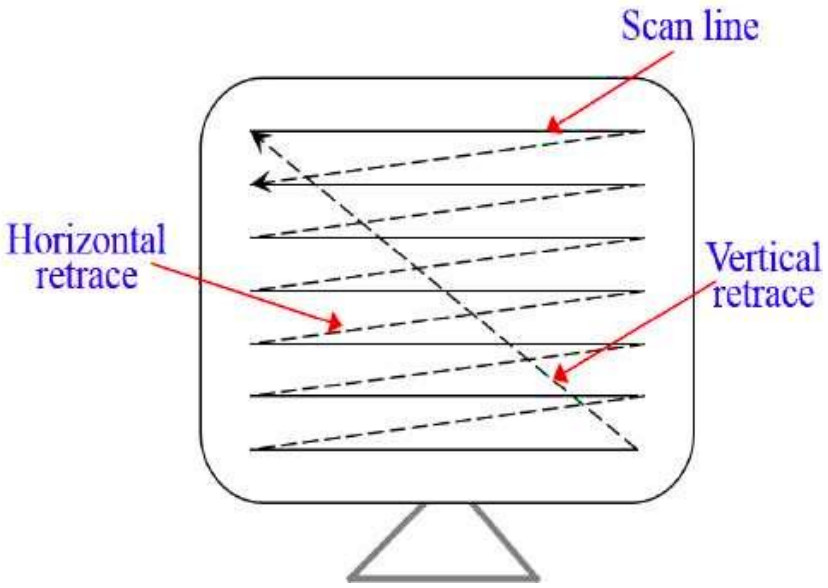


Figure 4. A scan line pattern

Illuminated phosphor dot emits light for only a friction of the second, this left to right, top-to-bottom scanning pattern must be repeated many times per second.

The refresh rate: is the no. of complete images, or frames drawn on the screen in 1 second.
Frame time: is the time between each complete scan.

- Although a pixel is actually off for a longer time than it is on, a fast enough refresh rate gives the eye the impression of a continuous illuminated image.

- Most expensive display systems have a refresh rate of 30 times/sec.

Interlacing

The scanning process is usually divided into two phases:

1. The first phase: in this phase the odd-numbered raster lines are displayed.
2. The second phase: in this phase the even-numbered raster lines are displayed.

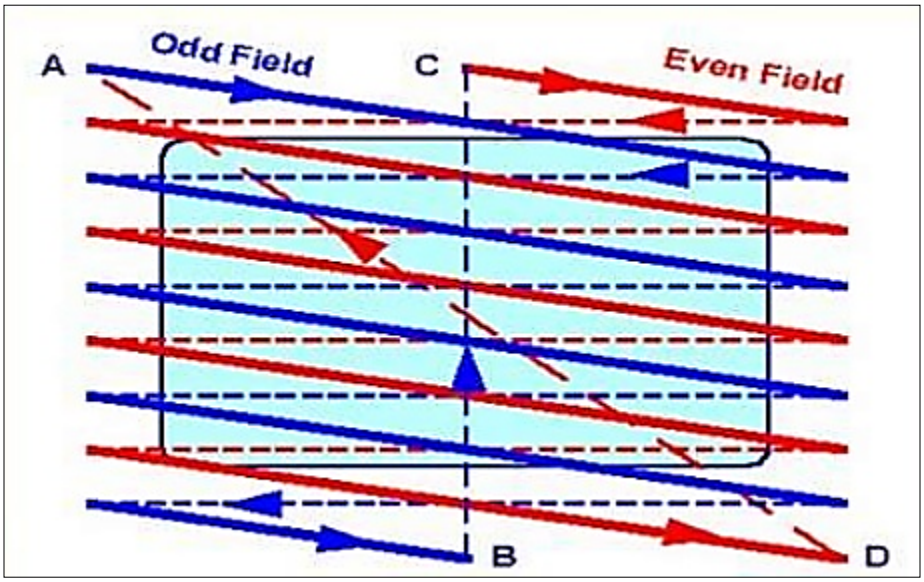


Figure 5. Interlacing

- Whenever an image is displayed on the screen, it is necessary to illuminate the appropriate pixels.

Q1: how does the display screen hardware know which pixels should be turned on?

Q2: how this information sent to the display screen?

Lecture 3

Illumination in Raster Scan System

The task of illuminating the appropriate pixels requires a display unit composed of three components:

1. Frame Buffer
2. Display Controller
3. Scan Conversion Algorithms

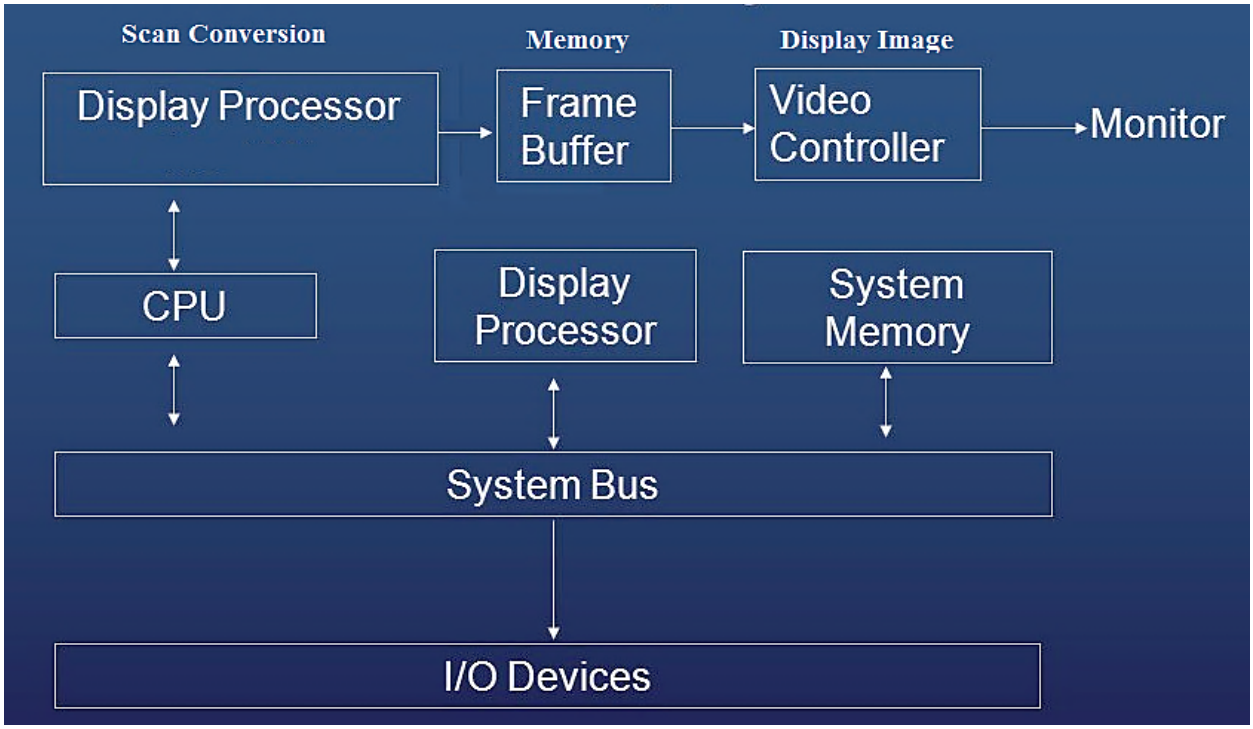


Figure 6. A raster scan display unit

Scan Conversion

Is the process of converting this abstract representation of an image into the appropriate pixel values in the frame buffer.

- Images are usually, defined in terms of equations, for example x+y=5 or graphic descriptions such as “draw a line from A to B”.
- Raster graphics systems have a separate display processor that performs scan conversion and other screen update functions.
- Inexpensive microcomputer graphics systems use the CPU and library of software routines to perform scan conversion.
- It is important to realize that each time an image is modified, hundreds of scan conversion calculations may be required.

Frame Buffer

- Each screen pixel corresponds to a particular entry in two dimensioned array residing in memory. This memory called a frame buffer or bitmap.
- Some graphic systems have a frame buffer memory distinct from main memory.
- The current is to have a frame buffer accessible to the CPU of the main computer, thus allowing rapid update of the stored image.
- The no. of rows in the frame buffer array equals the no. of raster lines on the display screen.
- The no. of columns in the array equals the no. of pixels on each raster line.
- The term pixel is also used to describe the row and column location in the frame buffer array that corresponds to the screen location.
- A 512 x 512 display screen requires 262,144 pixel memory locations.
- Whenever we wish to display a pixel on the screen, a specific value is placed in the corresponding memory location in the frame buffer array.

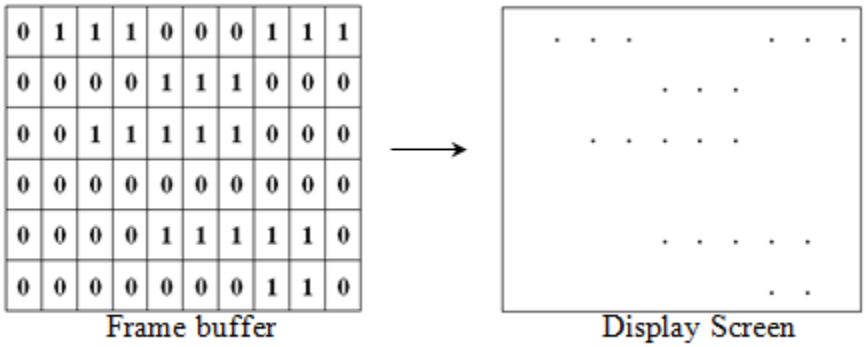


Figure 7. Frame buffer

Black/white Frame buffer

- Value of 1 placed in a location in the frame buffer results in the corresponding pixel being displayed on the screen.n
- Each screen location and corresponding memory location in the frame buffer is accessed by an (x,y) integer coordinate pair. The x value refers to the column position, the y value refers to the row position.
- The origin of this coordinate system is usually positioned at the bottom-left corner of the screen, although the image is still displayed in scan-line order from top to bottom.
- Each pixel in the frame buffer array is composed of no. of bits.
- A black and white image has only two intensity levels, on/off, has a single – bit-plane frame buffer.

frame buffer (color display)

- To display a color or a black-and-white quality image with shades of gray, additional bit planes are needed. 3 bit-plane frame buffer.
- The eight distinct values from 3-bit planes are interpreted as intensity levels between 0 and $2^3 - 1 = 7$.
- A black and white television quality image requires 8-bit planes, giving 2^8 or 256 intensity levels.
- High quality color display systems have frame buffers with up to 24 bit planes.
- Each of the three primary colors red-green-blue uses 8 bit planes to drive its color electron gun, producing a total of $2^{24} = 16,777,216$ colors.
- A full color frame buffer with 512 x 512 resolution requires 512 x 512 x 24 = 6,291,456 bits of memory.

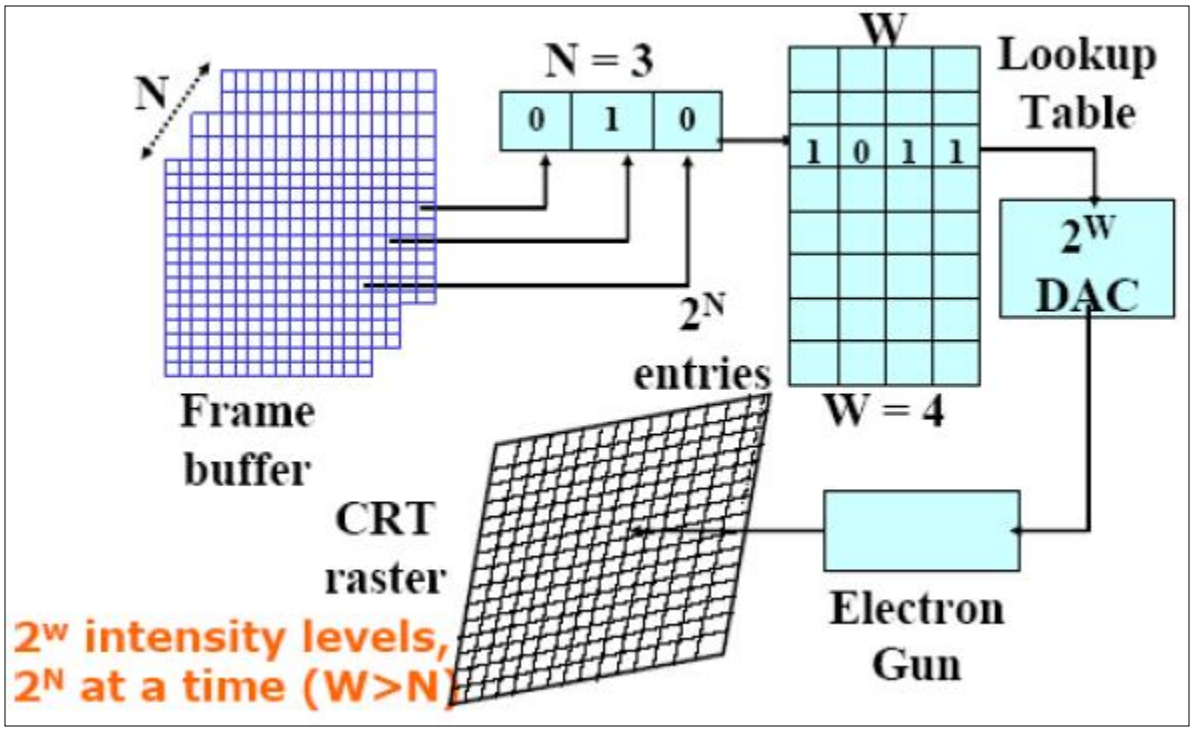


Figure 8. frame buffer (color CRT)

Display Controller

- This hardware device reads the contents of the frame buffer into a video buffer, which then converts the digital representation of a string of pixel values into analog voltage signals that are sent serially to the video display screen.
- Whenever the display controller encounters a value of 1 in a single bit-plate frame buffer, a high voltage signal is sent to the CRT, which turns on the corresponding screen pixel.
- The display controller sequentially cycles through the frame buffer, satisfying the refresh rate
- *The frame time, 1/refresh rate, is the sum of the vertical retrace time (two retraces if interlaced) , the total horizontal retrace time (one per scan line) , and the time to display all pixels.*

$$Pixel\ access\ time\ =\ \frac{[frame\ time\ -\ (2\times vertical\ ret.)]\ -\ horizontal\ ret.}{no.\ of\ pixels\ per\ can\ ine}$$

Random Vector Display

- We have seen how a raster scan system uses a frame buffer to display an image in scan line order: from top to bottom, left to right.
- There is a line-drawing graphics system, if we wish to draw only lines.
- Line drawing systems, also called random-vector or calligraphic systems.

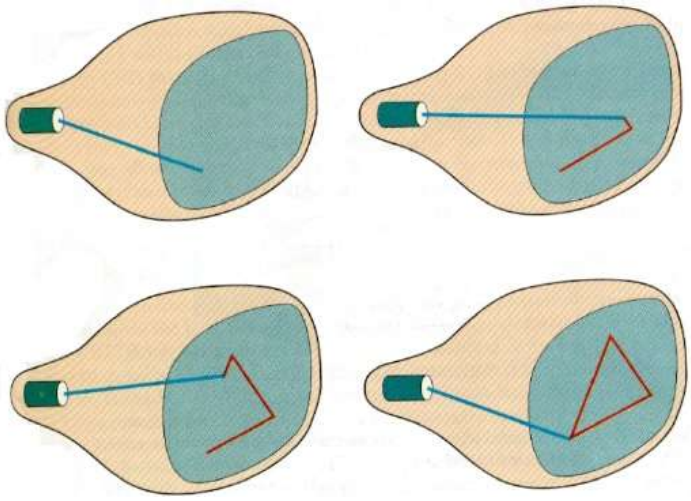


Figure 9. Random vector display

Line Drawing System (random vector)

- There is a memory region called a display file, consisting of line-drawing commands such as “draw line from A to B”.
- CPU or special display processor inserts the correct line drawing command into the display file.
- Vector generating hardware processor interprets these commands and sends the correct voltages to the deflection system of the CRT, moving the electron beam in a straight line from the initial to the final point.

- In RVD not necessary to direct the electron beam over the entire screen during each refresh cycle.
- The beam is moved only over those parts of the screen where there is a straight line or vector.
- Random- vector systems usually have a noninterlaced refresh rate of 60 times/sec.

Disadvantages

- Is the inability to fill areas and create realistic color or shaded images.
- The complexity of an image is limited in random-vector systems.

The low cost of memory and enhanced color and shading capabilities, have made raster scan displays the choice for most graphics display systems.

Lecture 4

Plotting points

- In order to draw a picture on a raster display, we must determine the corresponding points in the frame buffer that make up the picture.
- To perform this task we must write scan conversion point-plotting algorithm.
- Both the frame buffer and the display screen are given a two-dimensional coordinate system with the origin at the lower-left corner.

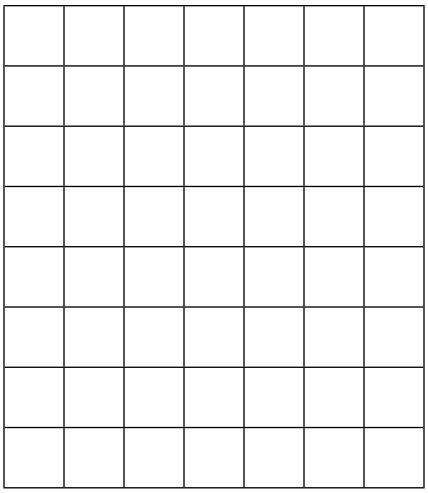


Figure 1.

- Each pixel is accessed by a non-negative integer (x,y) coordinate pair.
- The x values start at the origin 0 and increase from left to right; the y values start at 0 and increase from bottom to top (there is no standard for the location of the origin).
- None the x or y coordinate values should be beyond the boundaries of the display device.
- To prevent this undesirable effect, simple solution is to assign zero to a coordinate having a negative value and assign the maximum screen value to a large.

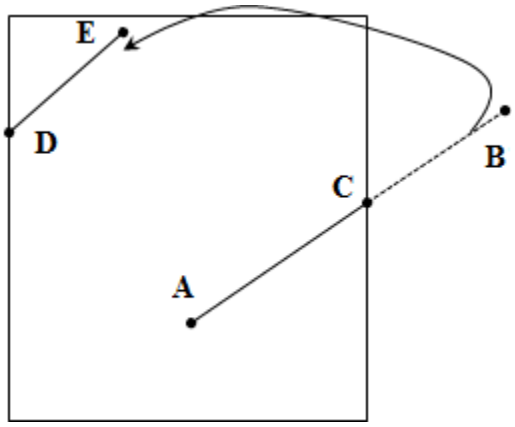


Figure 2.

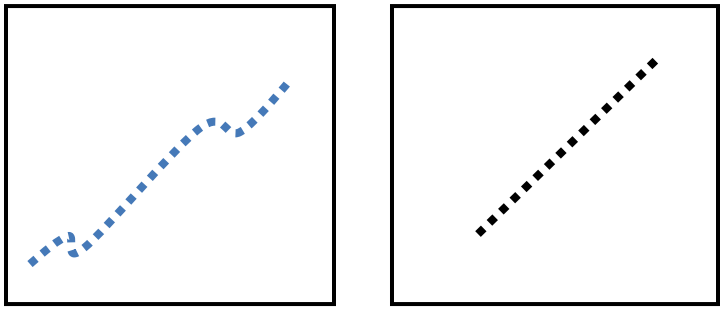
- To draw a point on the display screen, a point-plotting procedure is required:

  Plotpoint(x,y)

- The parameters x and y must be non-negative integer constants or expressions.

Line Drawing

- Many computer generated pictures are composed of straight line segments.
- A line segment is displayed by turning on a string of adjacent pixels.
- The accuracy and quality of the displayed line depend on the resolution of the display device.



Low resolution High resolution

Figure 3

Horizontal and vertical lines:

```
For x:= xstart to xend do
        Plotpoint(x,y);
```

Diagonal lines:

```
        X:=xstart;
        Y:=ystart;
        i:=0;
        while (x+i) <= xend do
            begin
               plotpoint (x+I , y+i)
               i:=i+1
            end;
```

Drawing lines with arbitrary slope creates several problems, because the display screen can eliminate only at pixel locations.

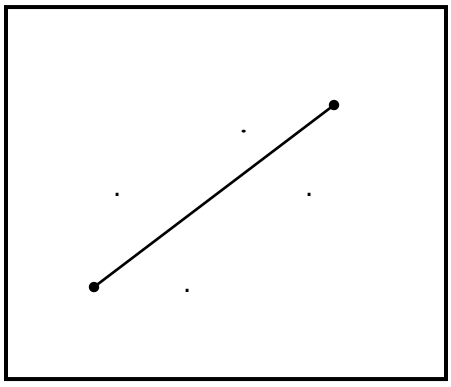


Figure 4

Simple DDA (simple digital differentiate Analyzer)

- Is a line drawing procedure that is easy to program and produce good lines.
- Draw a line with endpoints (xstart, ystart) and (xend, yend) having slope:

$$m = \frac{(yend - ystart)}{xend - xstart}$$

- Any two consecutive points (x1, y1), (x2, y2) lying on this line satisfies the equation:-

$$\frac{(y2 - y1)}{(x2 - x1)} = m$$

- For abs(m) < 1 and xstart < xend increment x by one unit until xend is reached.

- If xstart > xend swap the two endpoints

$$x2 = x1 + 1$$

$$\frac{(y2 - y1)}{1} = m \quad or \quad y2 = y1 + m$$

- If abs(m) > 1 and ystart > yend , generate line by reversing this procedure.
- Increment the y value by one unit until yend reached and solve for x.
- If (ystart > yend) , swap the two endpoints.

Y2=y1+1 or y2-y1=1

$$\frac{1}{(x2 - x1)} = m \quad or \quad x2 = x1 + \frac{1}{m}$$

- Use round function to convert the increment to integer.

```
Procedure Simple DDA (xstart, ystart, xend, yend:integer);
    var
        Xi, yi, xr, yr, m, minverse: real;

    begin
        m:=(yend-ystart) / (xend – ystart);
        if (abs(m) <1) and (xstart  > xend) or (abs(m) >1) and (ystart > yend) then
            swap(xstart, ystart, xend, yend);
        plotpoint(xstart, ystart);
        if abs (m)  < 1 then
            begin
                yr:=ystart;
                for xi:= (xstart+1) to (xend-1) do
                    begin
                        yr:=yr+m;
                        plotpoint(xi, round (yr))
                    end

            end

        else
           begin
              minverse:= 1/m;
              xr:=xstart;
              for yi:= (ystart+1) to (yend-1) do
                    begin
                        xr=xr+minverse;
                        plotpoint (round xr, yi)
                    end end;
        plotpoint (xend, yend);
    end;

    procedure Swap(x1, y1, x2, y2:integer);
        var
            temp:integer;
        begin
            temp:=x2; x2:=x1; x1:= temp;
            temp:=y2; y2:=y1; y1:=temp;
```

```
end;
```

Lecture 5

Draw Line Using Integer DDA:

- The simple DDA has disadvantages of using two operations that are expensive in computational time: floating point addition and the round function.
- Several good line drawing algorithms avoid these problems by using only integer arithmetic. This algorithm called the integer differential Analyzer (integer DDA).
- Its replace the real valued slope by its integer components.

Procedure Digital DDA (x1, x2, y1, y2:real);

```
var
    x,y, xinc, yinc :real;
    length, I : integer;

begin
    length:= abs (x2-x1);
    if  abs(y2-y1)  > length   then
        length:= abs(y2-y1);

    xinc:= (x2-x1) / length;
    yinc:= (y2-y1) / length;
    x:= x1+ 0.5;
    y:= y1+ 0.5;

    for I:= 1 to length do
        begin
            putpixel (trunc(x), trunc(y), color);
            x:= x + xinc;
            y:= y+ yinc;
        end;
     putpixel(x2, y2, color)
end;
```

Example:- Draw line by using integer DDA with start point (3,5) , end point (6,10)

Answer:-

Length= abs (x2-x1)
= (6-3) = 3

If abs (10-5) > 3 then length = 5

xinc = (6-3) / 5
= 3 /5 = 0.6

yinc = (10-5) / 5 = 1

x= x1+0.5 = 3.5
y= y1+0.5 = 5.5

putpixel (t(3.5), t(5.5), 3) □ (3,5,3)
x = 3.5 + 0.6 = 4.1
y = 5.5 + 1 = 6.5

putpixel (t(4.1), t(6.5), 3) □ (4,6,3)
x= 4.1 +0.6 = 4.7
y= 6.5 + 1 = 7.5

putpixel (t(4.7), t(7.5), 3) □ (4,7,3)
x= 4.7+ 0.6 = 5.3
y= 7.5 +1 = 8.5

putpixel (t(5.3), t(8.5), 3) □ (5,8,3)
x= 5.3+ 0.6 = 5.9
y= 8.5 + 1= 9.5

putpixel (t(5.9), t(9.5), 3) □ (5,9,3)

putpixel (6,10,3)

This procedure used to draw the line. Each line consist of many dots. You should determine the start point (x, y) then multiply each point with a fixed value. Draw point after trunk it.

Lecture 6

Circle Drawing

The circle is probably the most used curves in elementary graphics. A circle is specified by the coordinates of its center (xc, yc) and its radius r

$$(x - xc)^2 + (y - yc)^2 = r^2 \qquad (1)$$

If the center of the circle is at the origin (0,0)

$$x^2 + y^2 = r^2 \qquad (2)$$

$$y = yc \pm \sqrt{r^2 - (x - xc)^2} \qquad (3)$$

To draw a circle, increment the x values by one unit from $-r$ to $+r$ and use the equation (1) to solve for the two y values at each step.

This method of drawing a circle is inefficient for several reasons.

- If we have calculated one point to be on the circle, we shall see that seven other points are immediately known to lie on the circle.
- The amount of processing time required to perform the squaring and square root operation repeatedly.

Parametric Polar Representation

$$x = xc + r\cos(\theta)$$

$$y = yc + r\sin(\theta)$$

Where θ is measured in radians from 0 to 2π. The length are equals to $(r * \theta)$ and r the radius is constant, equal the increments of θ, $d\theta$, result in equal spacing (arc length) between successively plotted points.

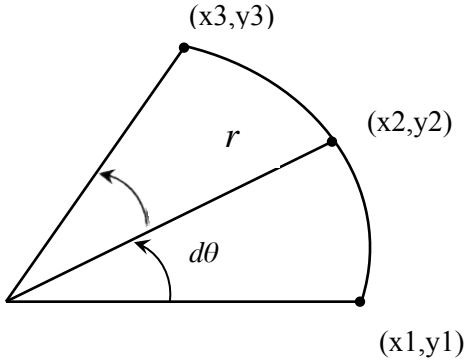
The problem of using polar representation is the repeated calculations of sin(θ)and cos(θ).

Incremental Drawing

To speed up the computation, we used the incremental drawing which calculates the points on a circle from the coordinates of the previously calculated points.

- This technique requires only an initial calculation of the sin and cos.
- The center of the circle is placed at the origin.
- Two consecutive points (x1, y1) and (x2, y2) on the circle are related by:

$$x1 = r\cos(\theta)$$
$$y1 = r\sin(\theta)$$
$$x2 = r\cos(\theta + d\theta)$$
$$y2 = r\sin(\theta + d\theta) \qquad \ldots. (1)$$



$d\theta$: is a fixed angular step size.

We get:

$$x2 = r\cos(\theta)\cos(d\theta) - r\sin(\theta)\sin(d\theta)$$
$$y2 = r\sin(\theta)\cos(d\theta) + r\cos(\theta)\sin(d\theta)$$

By substituting $x1$ and $y1$ from the equation (1) we have:

$$x2 = x1\cos(\theta) - y1\sin(d\theta)$$
$$y2 = y1\cos(\theta) + x1\sin(d\theta)$$

1. Starting the circle at $x1 = 0,\ y1 = r$ and fixed angle increment $d\theta$, we can compute all points on the circle by calculating $cos\ \ d\theta$ only once.

2. Before using these equations to draw a circle. Lets discuss symmetric:

   - If a point (a,b) lies on the circle $x^2 + y^2 = r^2$ Centred at the origin, then do so, other seven points: (-a, b), (a, -b) , (-a, -b) , (b, a) , (-b, a) , (b, -a), (-b, -a) to verify this, substitute all eight points in the circle equation.

   - We take advantage of this symmetry by calculating values only for first eight points of the circle, an angular interval of 45 degrees. If the first point is (0, r) the calculations are terminated when $y = x$.

   - To find the symmetric points on a circle centered at (xc, yc) , add xc to the first coordinate and yc to the second coordinate for all the eight points.

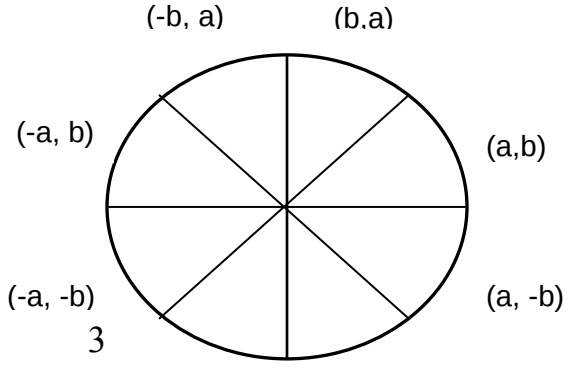
   - The following procedure circle, calculates points on the circle centered about the origin then adds xc to x values and yc to y values, moving the center to (xc, yc).

   - To make the circle look circular, the y values are multiplied by the aspect ratio before plotting.

   - The radius is the integer distance from the center of the circle (otherwise, use round function).

   - The spacing between adjacent points is one unit; therefore $d\theta$ should be 1 / radius.

   - Before using this procedure, we should test that the circle lies within the screen boundaries by using the calling procedure, by comparing $xc \pm r$ and $yc \pm r$ with the maximum and minimum horizontal and vertical screen coordinates.



(-b, -a) (b, -a)

Procedure Circle Drawing (Xc, Yc, radius: integer);

```
Const
    ar = 1.33;

var
    dtheta, ct, st, x, y, xtemp: real;

begin
    dtheta:= 1/ radius;
    ct:= cos (dtheta);
    st:= sin (dtheta);
    x:= 0;
    y:= radius;

    while y >= x do
        begin
            putpixel (round(xc+ x), round (yc+ y * ar));
            putpixel (round(xc- x), round (yc+ y * ar));
            putpixel (round(xc+ x), round (yc- y * ar));
            putpixel (round(xc - x), round (yc- y * ar));
            putpixel (round(xc+ y), round (yc+ x * ar));
            putpixel (round(xc - y), round (yc+ y * ar));
            putpixel (round(xc+ y), round (yc- x * ar));
            putpixel (round(xc- y), round (yc-  x * ar));

            xtemp:=x;
            x:= (x * ct – y * st);
            y:= (y * ct + xtemp* st);
        end;
end;
```

A circle also can be drowned by using many small line segments.

Setting the number line segments equal to a constant times the radius.

Larger circles will look as smooth as smaller ones.

The greater is the value of the constant multiplier.

The smoother represents the circle more and smaller line segments.

Procedure Circ_Draw (xc, yc, radius:integer);

```
Const
    two-pi= 6.28318;
var
    dtheta, ct, st : real;
    count, x, y, xtemp: integer;
begin
    dtheta:= two-pi / 16* radius;
    ct:= cos(dtheta);
    st:= sin(dtheta);
    x:=0;
    y:= radius;
    moves(round(xc+x), round (yc+y*ar));
    for count:= 1 to 16*radius do
        begin
            xtemp:= x;
            x:= (x*ct – y* st);
            y:= (y*ct + xtemp* st);
            draws (round (xc+x), round (yc+y* ar))
        end;
end;
```

Ellipses

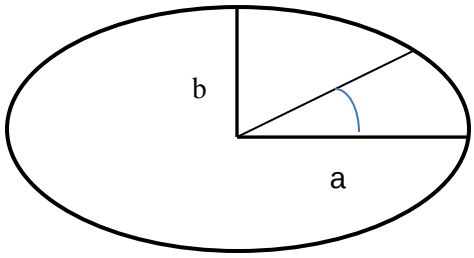
- An ellipse is a variation of a circle. Stretching a circle in one direction produces an ellipse.
- The ellipses are starched in the x or y direction.
- The equations for this type of ellipse centered at (xc, yc) are:

$$\left.\begin{aligned} x &= xc + a * \cos(\theta) \\ y &= yc + b * \sin(\theta) \end{aligned}\right\} \quad 1$$

- The angle θ assumes values between 0 and 2π radians
- The values of a and b affect the shape of the ellipse
- If $b > a$, the ellipse is longer in the y direction
- If $b < a$, the ellipse is longer in the x direction
- The ellipse can be drawn using four-points symmetry: if (c,d) lies on the ellipse; so do the points (-c, d), (c, -d), and (-c, -d).

$$\left.\begin{aligned} x2 &= x1\cos(d\theta) + (a/b)\, y1 \sin(d\theta) \\ y2 &= y1\cos(d\theta) - (b/a)x1\sin(d\theta) \end{aligned}\right\} \quad 2$$

- The procedure Circ-Draw can be modified to plot an ellipse by initializing x to 0 and y to b and replacing the circle equations by equation(2).
- Also can draw ellipse by changing the aspect ratio in the procedure Circ-Draw.



Lecture 8

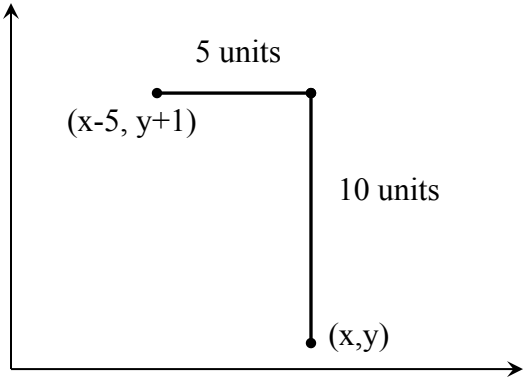
Two Dimensional Transformations

- Transformations provides means by which an image can be constructed or modified.
- The transformations are: translation, rotation, scaling, reflection and shearing. Each one is first described geometrically, and then its matrix representation is given.

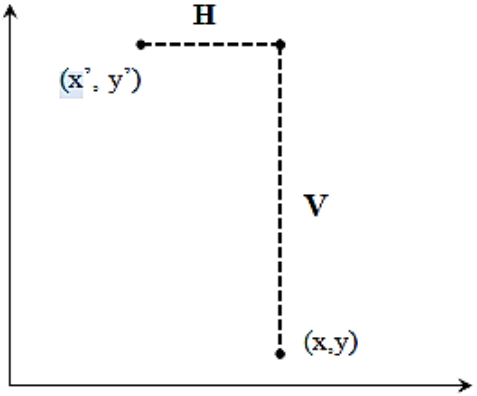
1. Translation

- A point in the world coordinates system is transformed to another position by modifying its x and y coordinates.

example:



- In general, a point (x, y) was translated into a new position (x', y') by moving it H units in the horizontal direction and V units in the vertical direction.



$$\acute{x} = x + H$$

$$\acute{y} = y + V$$

- The H and V represent the horizontal and vertical displacement or distance that the point has moved.
- If H is positive, the point moves to the right; if H is negative, the point moves to the left. Similarly, positive V moves the point up; a negative V moves it down.
- To translate, one or more objects in an image, we must translate every point defining the object.

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ H & V & 1 \end{bmatrix}$$

2. Rotation

- Another useful transformation is the rotation of an object about a specified pivot point.
- Any point (x,y) can be represented by its radius r, from the origin and angle θ.

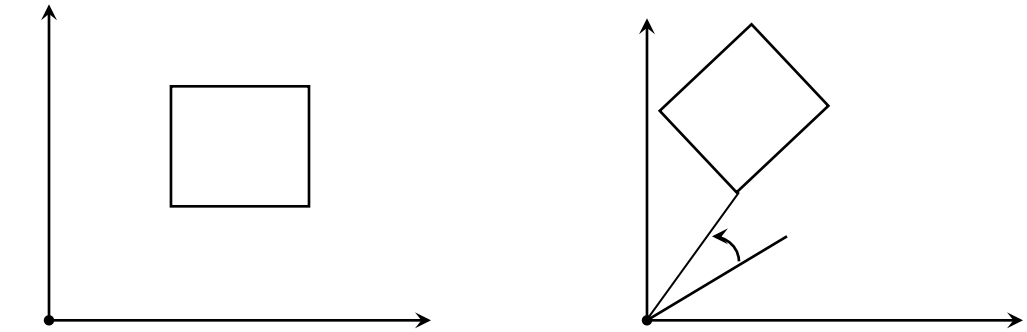
$$x = r\cos(\theta)$$
$$y = r\sin(\theta)$$

- If (x,y) is rotated an angle θ in the <u>counter-clockwise</u> direction. The transformed point $(\acute{x},\acute{y})$:

$$\acute{x} = r\cos(\theta + \theta)$$
$$\acute{y} = r\sin(\theta + \theta)$$

$\acute{x} = r\cos(\theta)\cos(d\theta) - r\sin(\theta)\sin(d\theta)$

$$\acute{y} = r\sin(\theta)\cos(d\theta) + r\cos(\theta)\sin(d\theta)$$



$$\acute{x} = x\cos(\theta) - y\sin(\theta)$$

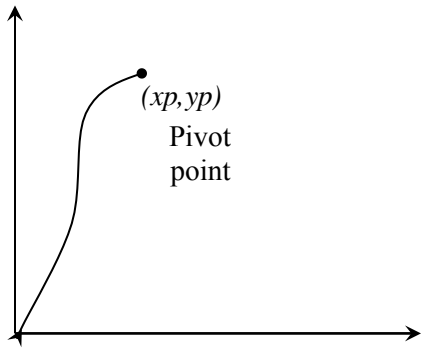
$$\acute{y} = y\cos(\theta) + x\sin(\theta)$$

Rotation Steps

To rotate an object an angle θ about a pivot point rather than the origin:

1. Translate the pivot point (xp, yp) to the origin.

$$\acute{x} = x - xp$$
$$\acute{y} = y - yp$$



origin

Translate

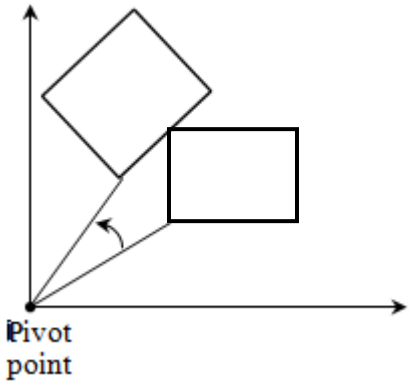
This translation sends the pivot point (xp, yp) to (0,0).

2. Rotate these translated points $(\acute{x}, \acute{y})$ θ degrees about the origin to obtain the new point $(\ddot{x}, \ddot{y})$.

$$\ddot{x} = \acute{x}\cos(\theta) - \acute{y}\sin(\theta)$$
$$\ddot{y} = \acute{y}\cos(\theta) + \acute{x}\sin(\theta)$$

Substituting the above values for $\acute{x}$ and $\acute{y}$ into these equations:

$$\ddot{x} = (x - xp)\cos(\theta) - (y - yp)\sin(\theta)$$
$$\ddot{y} = (y - yp)\cos(\theta) + (x - xp)\sin(\theta)$$



3. Translate the center of rotation back to the pivot point (xp, yp), thus the point $(\ddot{x}, \ddot{y})$ is translated to $(\dddot{x}, \dddot{y})$:

$$\dddot{x} = \ddot{x} + xp$$
$$\dddot{y} = \ddot{y} + yp$$

Substituting the values of $\ddot{x}$ and $\ddot{y}$ from the above equation

$$\dddot{x} = (x - xp)\cos(\theta) - (y - yp)\sin(\theta) + xp$$

$$\dddot{y} = (y - yp)\cos(\theta) + (x - xp)\sin(\theta) + yp$$

To rotate the clock wise direction, change the angle θ *to* $-\theta$:

$$\cos(-\theta) = \cos(\theta)$$

$$\sin(-\theta) = -\sin(\theta)$$

Then the rotate matrix is:

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

Lecture 9

3. Scaling

- An object can be made larger by increasing the distance between the points which describe the object.

- We can change the size of an object, or image by multiplying the distances between points by enlargement or reduction factor.

- This factor is called the scaling factor.

- If the scaling factor greater than 1, the object is enlarged; if less than 1, the object is made smaller. If equal 1 no effect on the object.

- Whenever a scaling is performed, there is one point that remains at the same location. This is called the fixed point of the scaling transformation.

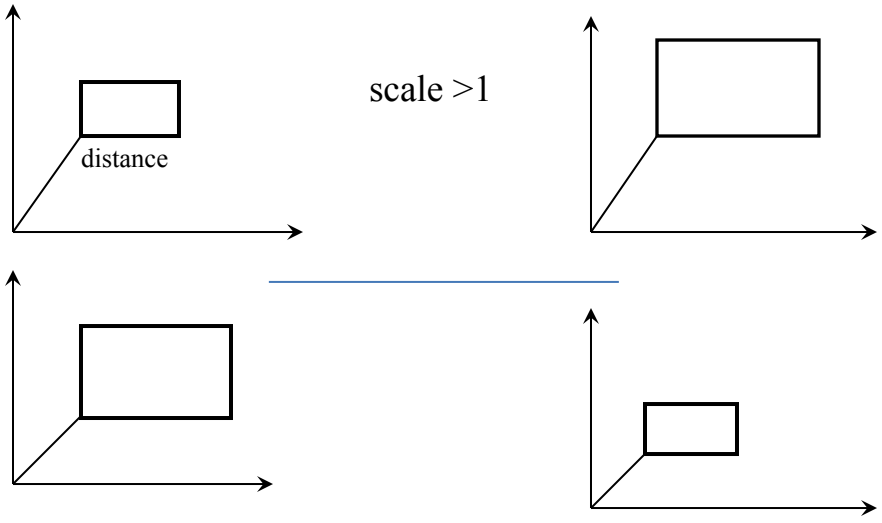
- If the fixed point at the origin (0,0), a point (x,y) can be scaled by a factor Sx in the x direction and Sy in the y direction to the new point $(\acute{x}, \acute{y})$.

$$\acute{x} = x \,.\, Sx$$
$$\acute{y} = y \,.\, Sy$$

Sx and Sy are called the horizontal and vertical scale factors.

- If $Sx \neq Sy$, the resulting object is a distortion of the original.

- If both scaling factors are greater than 1, the scaled object is enlarged and moved further away from the fixed point.



scale <1

Scaling Steps

1. Translate the point *(xp, yp)* to the origin. Every point *(x,y)* moved to a new point $(\acute{x}, \acute{y})$.

$$\acute{x} = x - xp$$
$$\acute{y} = y - yp$$

2. Scale these translated points with the origin as the fixed point.

$$x'' = \acute{x} \ . \ Sx$$
$$y'' = \acute{y} \ . \ Sy$$

3. Translate the origin back to the fixed point *(xp, yp)*.

$$\dddot{x} = x'' + xp$$
$$\dddot{y} = y'' + yp$$

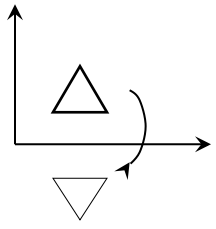
Then, substitute the values of $\acute{x}$, $\acute{y}$, x'' *and* y'' into the last equation:

$$\dddot{x} = (x - xp).\ Sx + xp$$
$$\dddot{y} = (y - yp).\ Sy + yp$$

Scale Matrix : $\begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$

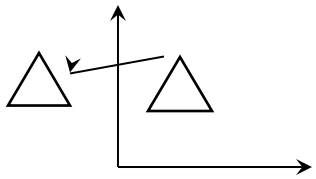
Reflection

1. On *x - axis*



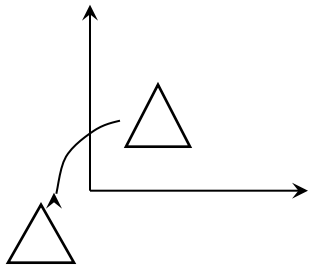
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2. On y - *axis*



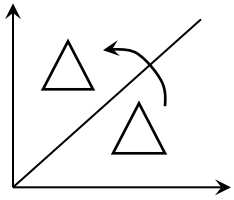
$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3. On Origin



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

4. On $y = x$



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

Shearing

A shearing transformation produces a distortion of an object or the entire image.

- There are two types of shears, x-shear and y-shear.

y- shear

$$\acute{x} = x$$
$$\acute{y} = y + shy * x \quad , \quad shy \neq 0$$

- A *y - shear* moves a vertical line up or down, depending on the sign of the shear factor *shy*.
- A horizontal line distorted into a slanted line with slope *shy*.

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

x- shear

$$\acute{y} = y$$
$$\acute{x} = x + shx * y$$

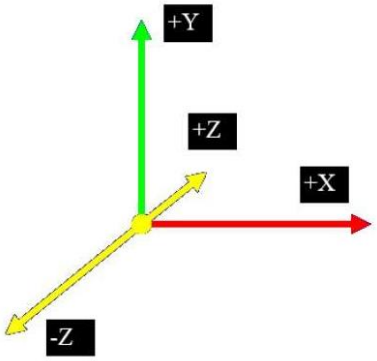
$$\begin{bmatrix} 1 & shx & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Lecture 10

3D Computer Graphics

In the 2D system, we use only two coordinates X and Y but in 3D, an extra coordinate Z is added. 3D graphics techniques and their application are fundamental to the entertainment, games, and computer-aided design industries. It is a continuing area of research in scientific visualization.

Furthermore, 3D graphics components are now a part of almost every personal computer and, although traditionally intended for graphics-intensive software such as games, they are increasingly being used by other applications.



Three Dimensional Transformations

- Our world is composed of three dimensional images.
- Objects have height, width and depth.

1. Translation

Translate a point (x, y, z) T_x units in the x direction, T_y units in the y direction and T_z units in the Z direction to a new point (x`, y`, z`).

$$T_r = (T_x, T_y, T_z) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}$$

$$\acute{x} = x + T_x$$

$$\acute{y} = y + T_y$$

$$\acute{Z} = Z + T_z$$

2. Rotation

- A three dimensional rotation is performed about an axis.

***Rotation on*: $Z - axis$**

$$R = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\acute{x} = xcos\theta - ysin\theta$$

$$\acute{y} = xsin\theta + ycos\theta$$

$$\acute{Z} = Z$$

***Rotation on* $y - axis$**

$$R = \begin{bmatrix} cos\theta & 0 & sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -sin\theta & 0 & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$\acute{x} = xcos\theta + zsin\theta$

$\acute{y} = y$

$\acute{Z} = -xsin\theta + xcos\theta$

$\boldsymbol{Rotation\ on\ X-axis}$

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & sin\theta & 0 \\ 0 & -sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$\acute{x} = x$

$\acute{y} = ycos\theta - zsin\theta$

$\acute{z} = ysin\theta + zcos\theta$

3. Scaling

A three dimensional scaling allows for a contraction or stretching in any of $x, y\ or\ \ z$ directions.

$$S = \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$\acute{x} = x * Sx$

$\acute{y} = y * Sy$

$\acute{z} = z * Sz$

4. Shearing

The shear matrix transform is:

$$Sh = \begin{bmatrix} 1 & b & c & 0 \\ d & 1 & f & 0 \\ h & i & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$\acute{x} = x + yd + zh$

$\acute{y} = xb + y + zi$

$\acute{z} = xc + fy + z$

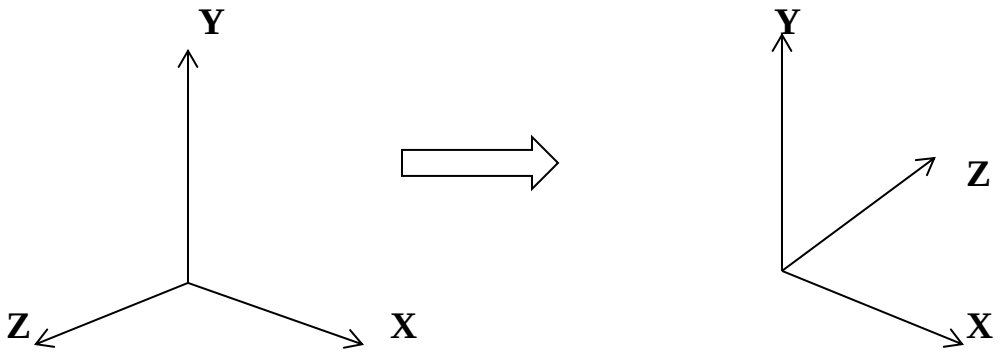
If $d, h \neq 0$ the shear in X-axis

If $b, i \neq 0$ the shear in y-axis

Else the shear in Z

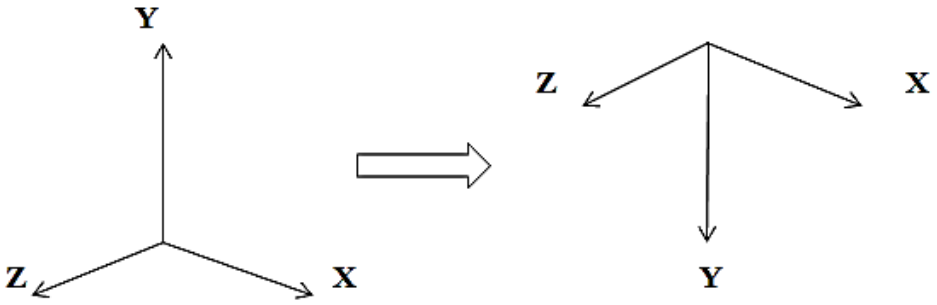
5. Reflection

a. Reflection relative to the XY plane



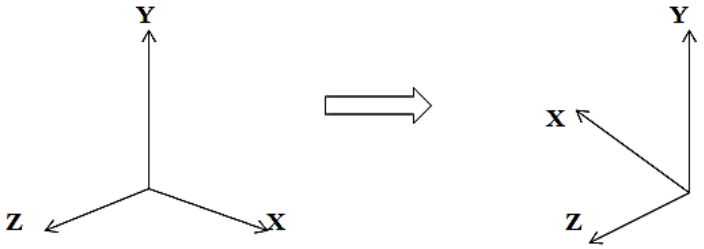
$$\begin{bmatrix} \acute{X} \\ \acute{Y} \\ \acute{Z} \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

b. Reflection relative to the XZ plane



$$\begin{bmatrix} \acute{X} \\ \acute{Y} \\ \acute{Z} \\ \acute{1} \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

c. Reflection relative to the YZ plane

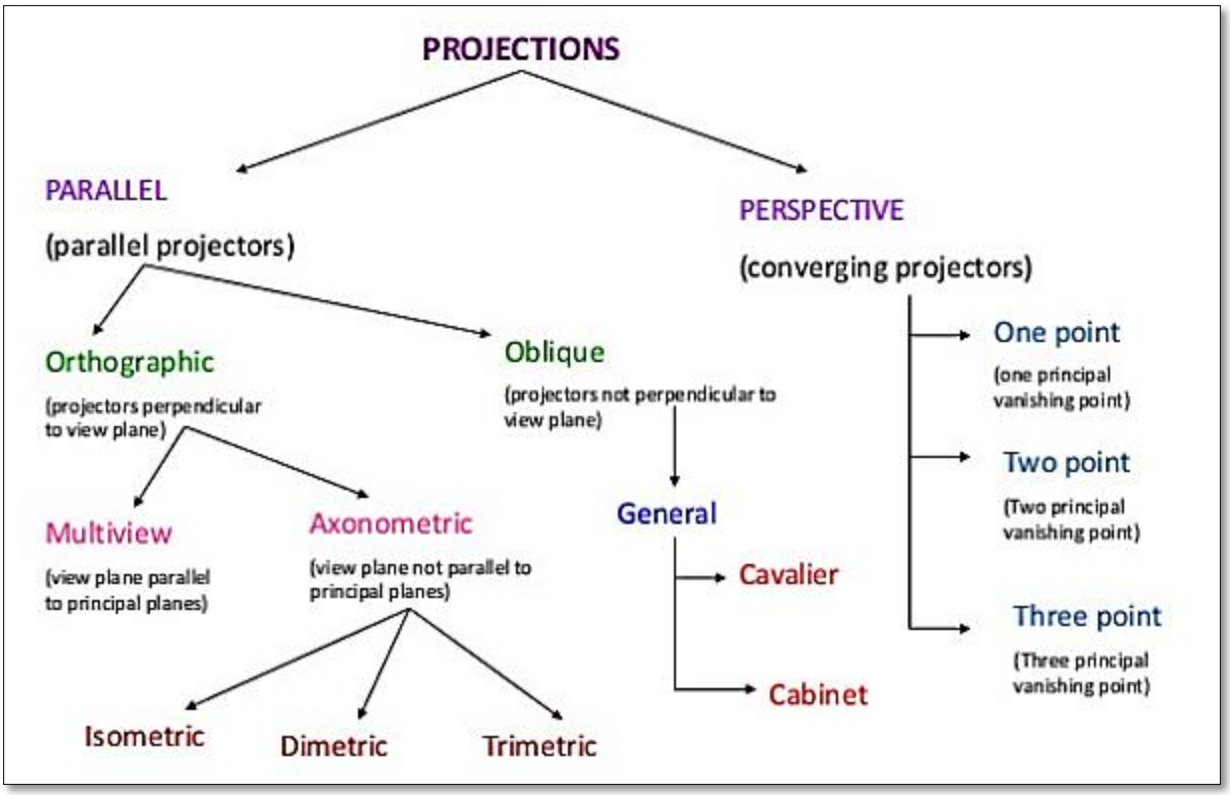


$$\begin{bmatrix} \acute{X} \\ \acute{Y} \\ \acute{Z} \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Lecture 11

Projections

- In computer graphics , a three –dimensional object is viewed on a two –dimensional display.
- A projection is a transformation that performs this conversion from a three-dimensional representation to a two-dimensional representation.
- This process is not unique to the field of computer graphics, artists use perspective projection to create realistic pictures; architects use several parallel projections to depict different views of buildings, draw maps of the world.
- There are two types of projections: parallel and perspective.

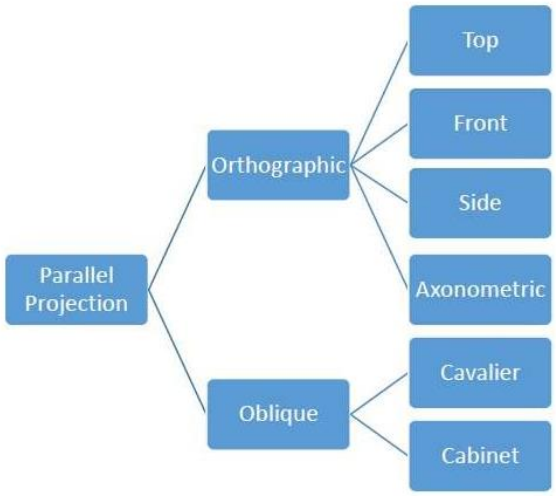


Parallel Projection

Parallel projection discards z-coordinate and parallel lines from each vertex on the object are extended until they intersect the view plane. In parallel projection, we specify a direction of projection instead of center of projection.

In parallel projection, the distance from the center of projection to project plane is infinite. In this type of projection, we connect the projected vertices by line segments which correspond to connections on the original object.

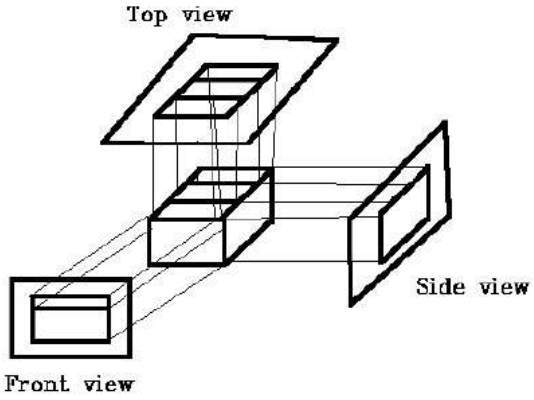
Parallel projections are less realistic, but they are good for exact measurements. In this type of projections, parallel lines remain parallel and angles are not preserved. Various types of parallel projections are shown in the following hierarchy.



Orthographic Projection

In orthographic projection the direction of projection is normal to the projection of the plane. There are three types of orthographic projections −

- Front Projection
- Top Projection
- Side Projection

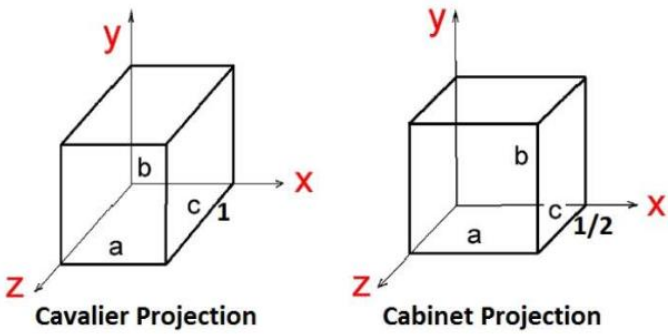


Oblique Projection

In orthographic projection, the direction of projection is not normal to the projection of plane. In oblique projection, we can view the object better than orthographic projection.

There are two types of oblique projections − Cavalier and Cabinet. The Cavalier projection makes 45° angle with the projection plane. The projection of a line perpendicular to the view plane has the same length as the line itself in Cavalier projection. In a cavalier projection, the foreshortening factors for all three principal directions are equal.

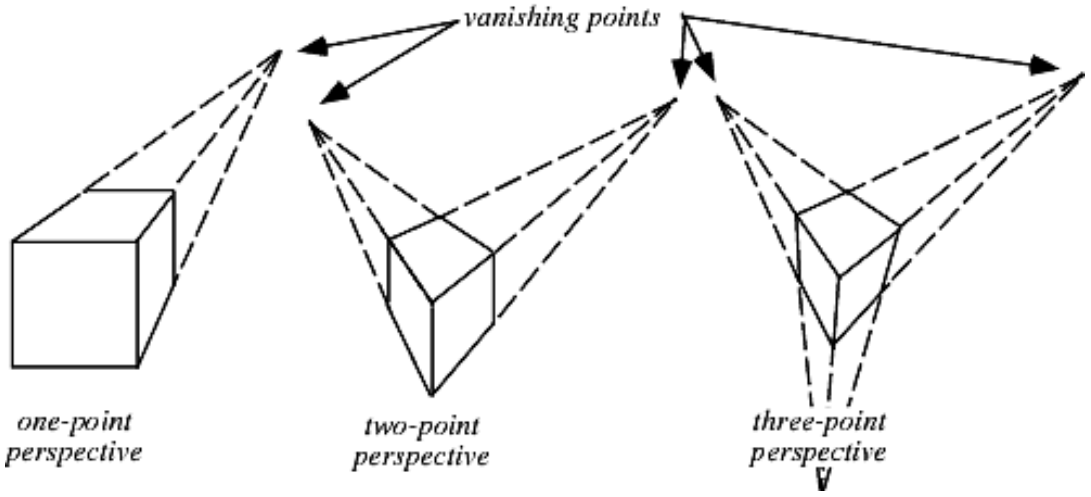
The Cabinet projection makes 63.4° angle with the projection plane. In Cabinet projection, lines perpendicular to the viewing surface are projected at ½ their actual length. Both the projections are shown in the following figure.

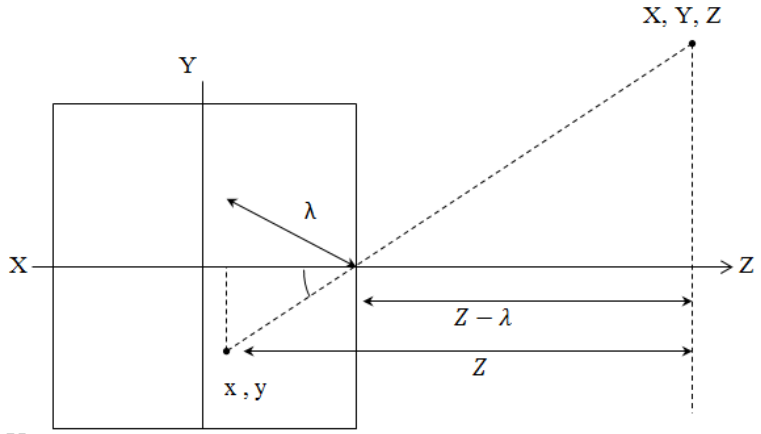


Perspective Projection

- Is projection of three dimensional scene information onto a two dimensional plane.
- In perspective projection, the distance from the center of projection to project plane is finite and the size of the object varies inversely with the distance which looks more realistic.
- The distance and angles are not preserved and parallel lines do not remain parallel. Instead, they all converge at a single point called the center of projection or projection reference point. There are 3 types of perspective projections, which are shown in the following chart.
  - One point perspective projection: is simple to draw.
  - Two point perspective projection: gives a better impression of depth.
  - Three point perspective projection: is most difficult to draw.

The following figure shows all the three types of perspective projection –





$$\frac{-x}{\lambda} = \frac{X}{Z-\lambda}$$

$$\frac{-y}{\lambda} = \frac{Y}{Z-\lambda}$$

$$x = \frac{X\lambda}{\lambda - Z}$$

$$y = \frac{Y\lambda}{\lambda - Z}$$

Let P is the perspective projection matrix

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

We have to suggest new coordinate model called homogenous coordinate

$$c = \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix} \qquad c_h = \begin{bmatrix} kx \\ ky \\ 0 \\ k \end{bmatrix}$$

by dividing each element of the 4th one (k):

c_h ⟶ c

$$W = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \qquad W_h = \begin{bmatrix} kX \\ kY \\ kZ \\ k \end{bmatrix}$$

Convertion from 3D to 2D:

$$\boldsymbol{c_h = P\ W_h}$$

$$\begin{bmatrix} kx \\ ky \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{\lambda} & 1 \end{bmatrix} \begin{bmatrix} kX \\ kY \\ kZ \\ k \end{bmatrix}$$

$$= \begin{bmatrix} kX \\ kY \\ kZ \\ -\frac{kZ}{\lambda} + k \end{bmatrix}$$

Divide on the 4th factor.

$$\begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{\lambda X}{\lambda - Z} \\ \frac{\lambda Y}{\lambda - Z} \\ \frac{\lambda Z}{\lambda - Z} \\ 1 \end{bmatrix}$$

Convertion from 2D to 3D:

$$\boldsymbol{W_h = P^{-1}\, c_h}$$

$$P^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{\lambda} & 1 \end{bmatrix}$$

$$W_h = \begin{bmatrix} kX \\ kY \\ kZ \\ k \end{bmatrix} \quad , \quad c_h = \begin{bmatrix} kx \\ ky \\ 0 \\ k \end{bmatrix}$$

$$\begin{bmatrix} kX \\ kY \\ kZ \\ k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{\lambda} & 1 \end{bmatrix} \begin{bmatrix} kx \\ ky \\ 0 \\ k \end{bmatrix}$$

$$= \begin{bmatrix} kx \\ ky \\ kz \\ \frac{kz}{\lambda} + k \end{bmatrix}$$

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{\lambda x}{\lambda + z} \\ \frac{\lambda y}{\lambda + z} \\ \frac{\lambda z}{\lambda + z} \\ 1 \end{bmatrix}$$

Lecture 12

Windows and Viewports

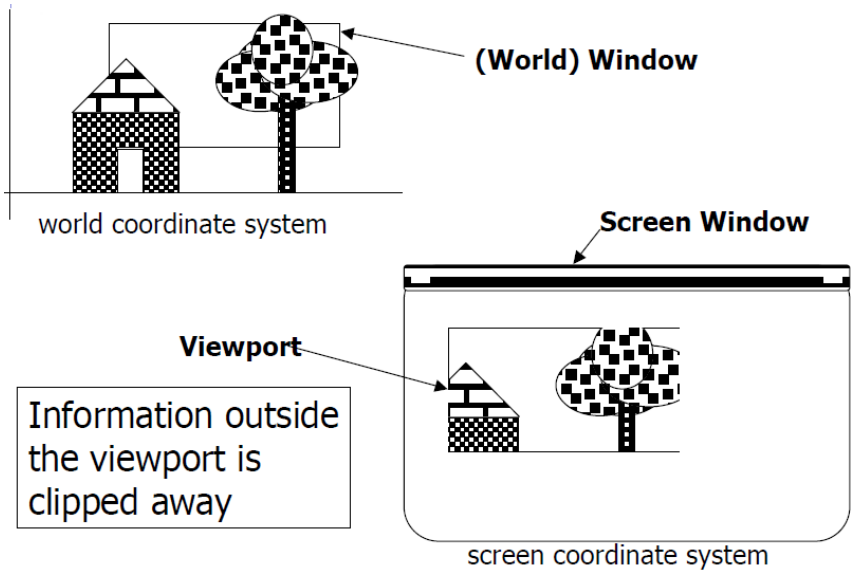
- **World Coordinate System** (Object Space): The space in which the application model is defined. The representation of an object is measured in some physical or abstract units.
- **Screen Coordinate System** (Image Space): The space in which the image is displayed. Usually measured in pixels, but could use any units.

- **(World) Window**: The rectangle defining the part of the world we wish to display.

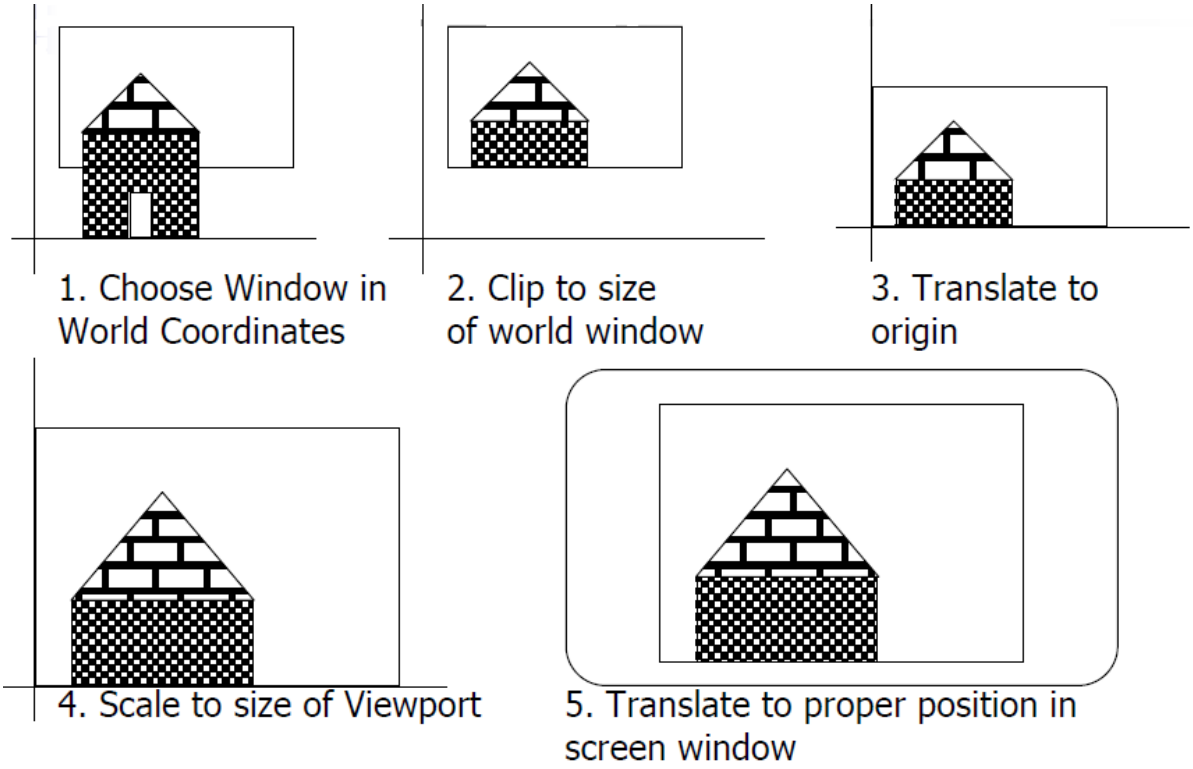
- **(Screen) Window**: The visual representation of the screen coordinate system for "windowed" displays (coordinate system moves with screen window).

- **Viewport**: The rectangle within the screen window defining where the image will appear (Default is usually entire interface window).

- Window-Viewport Transformation: The process of mapping from a window in world coordinates to a viewport in screen coordinates.



Steps of Windows to Viewport Transformation



$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -w_{xl} & -W_{yl} & 1 \end{bmatrix}$$

Translate to origin

$$\begin{bmatrix} \frac{v_{xh}-v_{xl}}{w_{xh}-w_{xl}} & 0 & 0 \\ 0 & \frac{v_{yh}-v_{yl}}{w_{yh}-w_{yl}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

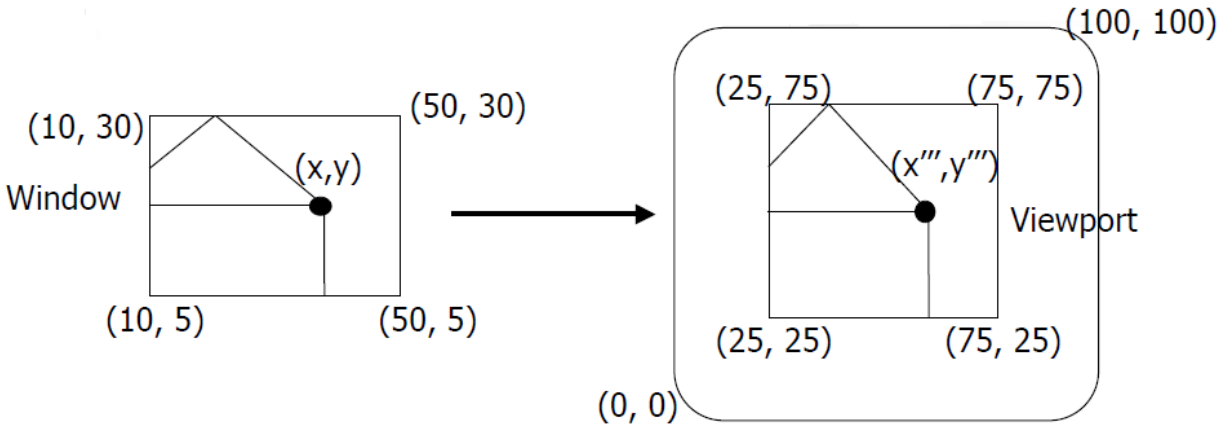
Scale from Window to Viewport

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ v_{xl} & v_{yl} & 1 \end{bmatrix}$$

Translate to the proper position

Example:

Window with corners (10,5) , (50,30) and viewport with corners (25,25), (75,75). Do window to viewport transformation.



Answer:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -10 & -5 & 1 \end{bmatrix} \begin{bmatrix} \frac{75-25}{50-10} & 0 & 0 \\ 0 & \frac{75-25}{30-5} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 25 & 25 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -10 & -5 & 1 \end{bmatrix} \begin{bmatrix} 1.25 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 25 & 25 & 1 \end{bmatrix}$$

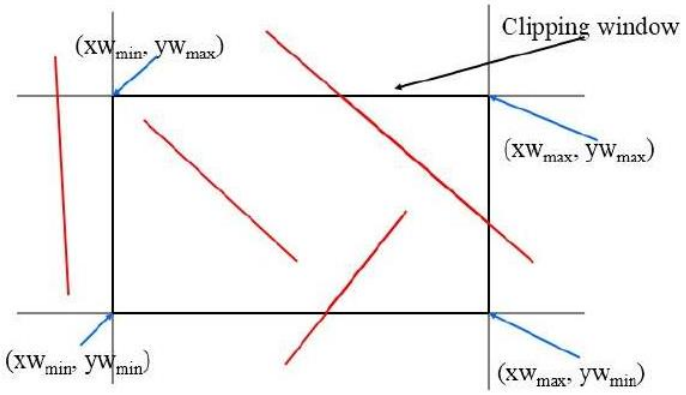
$$= \begin{bmatrix} 1.25 & 0 & 0 \\ 0 & 2 & 0 \\ -12.5 & -10 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 25 & 25 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1.25 & 0 & 0 \\ 0 & 2 & 0 \\ 12.5 & 15 & 1 \end{bmatrix}$$

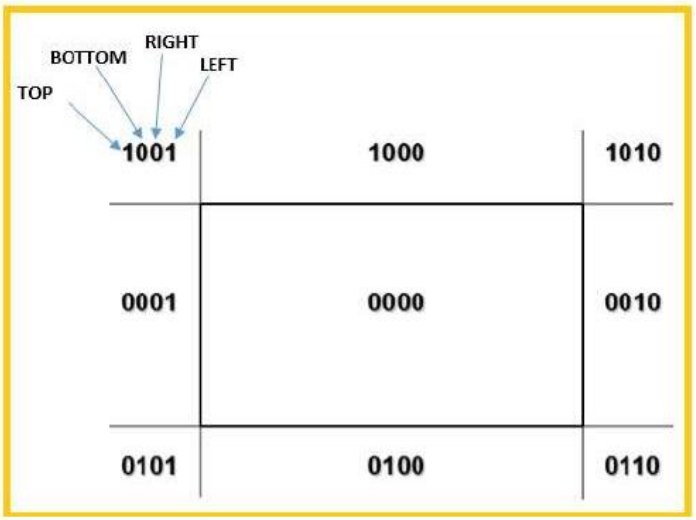
Cohen-Sutherland Line Clippings

In line clipping, we will cut the portion of the line which is outside of the window and keep only the portion that is inside the window.

This algorithm uses the clipping window as shown in the following figure. The minimum coordinates of the clipping region is (XW min, YW min) (XWmin,YWmin) and the maximum coordinate for the clipping region is (XW max, YW max) (XWmax,YWmax).

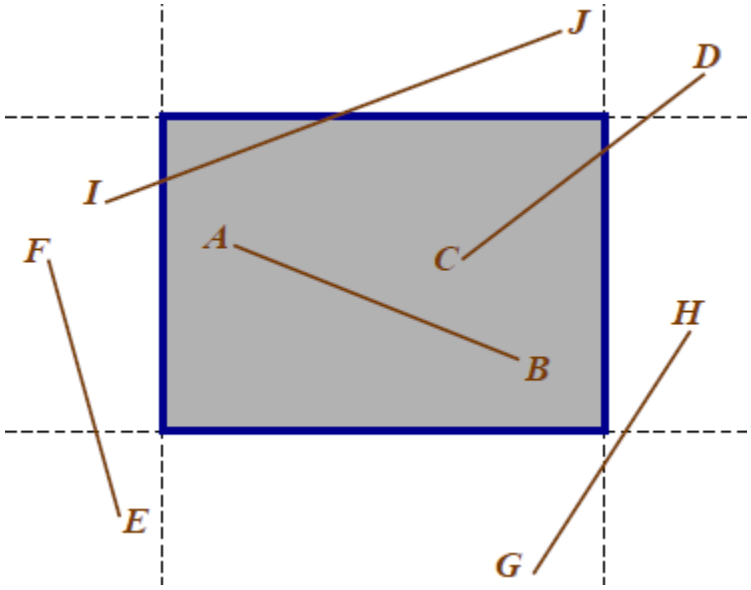


We will use 4-bits to divide the entire region. These 4 bits represent the Top, Bottom, Right, and Left of the region as shown in the following figure. Here, the TOP and LEFT bit is set to 1 because it is the TOP-LEFT corner.



There are 3 possibilities for the line –

- Line can be completely inside the window (This line should be accepted).
- Line can be completely outside of the window (This line will be completely removed from the region).
- Line can be partially inside the window (We will find intersection point and draw only that portion of line that is inside region).



Algorithm

Step 1 − Assign a region code for each endpoint.

Step 2 − If both endpoints have a region code 0000 then accept this line.

Step 3 − Else, perform the logical AND operation for both region codes.

Step 3.1 − If the result is not 0000, then reject the line.

Step 3.2 − Else you need clipping.

Step 3.2.1 − Choose an endpoint of the line that is outside the window.
Step 3.2.2 − Find the intersection point at the window boundary (base of region code).
Step 3.2.3 − Replace endpoint with the intersection point and update the region code.

Step 3.2.4 − Repeat step 2 until we find a clipped line either accepted or rejected.

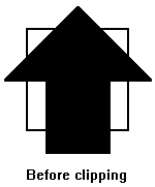
Step 4 − Repeat step 1 for other lines.

Example:

Window is defined as A(10,20), B(20,20), C(20,10) , D(10,10). Find visible portion of line p(15,15) and p(15,5).

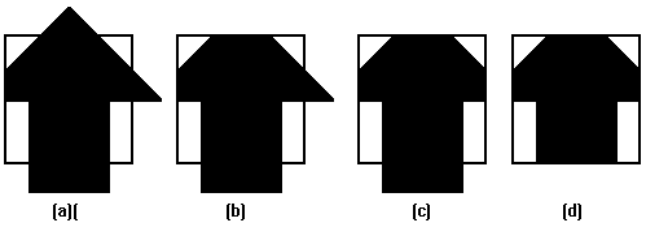
Sutherland - Polygon Clipping

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



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

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



a.Clipping against the left side of the clip window.

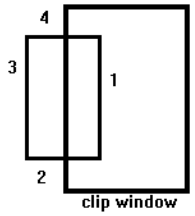
b.Clipping against the top side of the clip window.

c.Clipping against the right side of the clip window.

d.Clipping against the bottom side of the clip window.

Four Types of Edges

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



The four types of edges are:

1.Edges that are totally inside the clip window. - add the second inside vertex point

2.Edges that are leaving the clip window. - add the intersection point as a vertex

3.Edges that are entirely outside the clip window. - add nothing to the vertex output list

4.Edges that are entering the clip window. - save the intersection and inside points as vertices

How To Calculate Intersections

Assume that we're clipping a polgon's edge with vertices at (x1,y1) and (x2,y2) against a clip window with vertices at (xmin, ymin) and (xmax,ymax).

- The location (IX, IY) of the intersection of the edge with the left side of the window is:
  1) IX = xmin
  2) IY = slope*(xmin-x1) + y1, where the slope = (y2-y1)/(x2-x1)

- The location of the intersection of the edge with the right side of the window is:

1) IX = xmax
2) IY = slope*(xmax-x1) + y1, where the slope = (y2-y1)/(x2-x1)

- The intersection of the polygon's edge with the top side of the window is:
  1) IX = x1 + (ymax - y1) / slope
  2) IY = ymax

- Finally, the intersection of the edge with the bottom side of the window is:

1) IX = x1 + (ymin - y1) / slope
2) IY = ymin