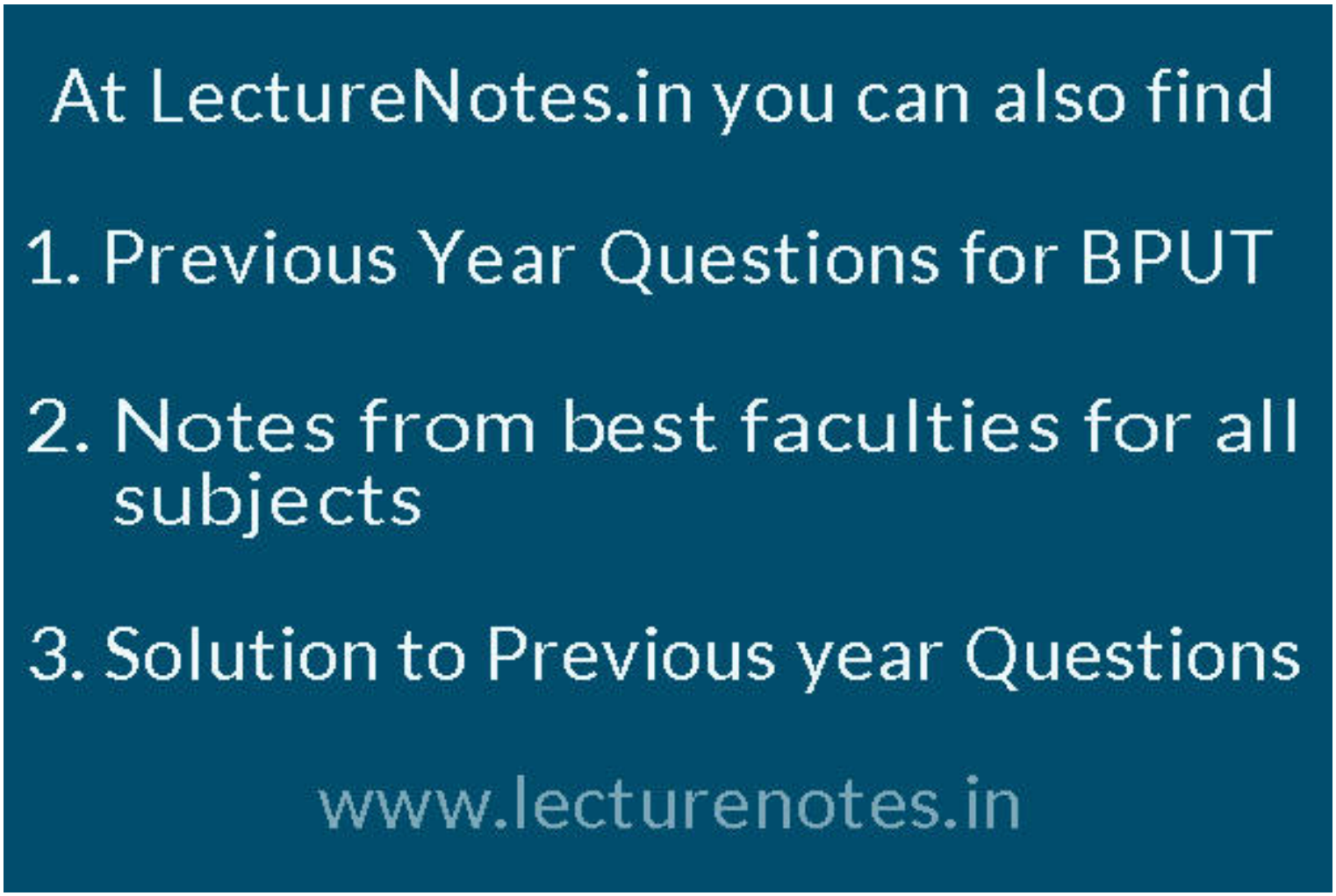
Computer Graphics And Multimedia

Contributed By:
Ankita Jiyani

Disclaimer



Computer Graphics And Multimedia

Topic:

Introduction To Subject

Contributed By:

Ankita Jiyani

Computer Graphics and Multimedia Techniques

Introduction to subject :-

The computer graphics refers to anything involved in the creation & manipulation of images on computer.

Computer graphics is an art of drawing pictures on computer screens with the help of programming. It involves computation, creation & manipulation of data.

Application of computer graphics :-

CAD – 3D modeling of objects

Entertainment → Games, movies.

Virtual Reality.

Visualization for Science & Business.

Hospitals device for tracking patient's body.

Image Processing.

Computer Graphics And Multimedia

Topic:

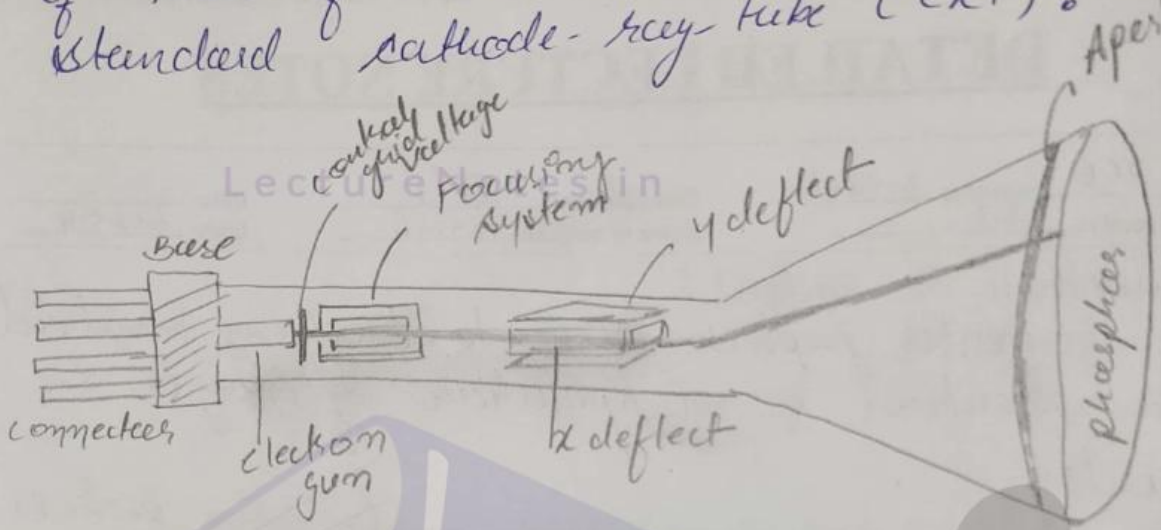
Video Display Device

Contributed By:

Ankita Jiyani

Video Display Devices:-

The primary output devices in graphics system is a video monitor. The operation of most of the devices is based on the standard cathode-ray tube (CRT).



operation of CRT:-

1. Electron gun emits a beam of electron.
2. Electron beam passes through focusing & deflecting system, that direct it towards the specified position on the phosphor coated screen
3. When beam hits the screen, the phosphor emits a small spot of light at each position contacted by electron beam.
4. It redraws the picture by directing the electron beam back over the same screen points quickly
5. Focusing system & x & y deflects are anodes, which are magnetized to push & pull the electrons in a specific direction.

6. There are three different color phosphors. (R G B) for each pixel & the color of pixel depends on the phosphor on which electron strikes.

7. Aperture grill is a metal sheet consisting of holes, which controls the brightness on the screen.

8. Different level of phosphors are available for CRT, but major difference b/w them is the persistence, how long they continue to emit light.

9. The max. no. of points that can be displayed without overlap on a CRT is referred as resolution.

10. Aspect Ratio :- This no. gives the ratio of vertical points to horizontal points necessary to produce equal length lines in both direction on screen.

Aspect ratio 3/4 means vertical line plotted with 3 points has same length as a horizontal line plotted with four points.

Computer Graphics And Multimedia

Topic:

Types Of Display

Contributed By:

Ankita Jiyani

There are two ways by which we can display an object on the screen:-

1 Rastes Scan Display :-

1. The image which is to be displayed is stored in a binary format of 0 & 1 in refresh buffer memory.
2. A video controller is used to scan each & every line of the Refresh Buffer memory.
3. Each line is scanned from left to right and the electron beam of video controller is turned on & off to create a pattern of illuminated spots.
4. The refresh buffer or frame buffer stores various information:-
   a) coordinates of screen points
   b) intensity values of screen points.
5. These information is retrieved from the buffer & painted on the screen one row at a time
6. Each point is referred as pixel or pel (picture element).
7. Printers are the example of raster scan display.

8. In black & white system one bit is needed to control the intensity of screen position.

for bilevel : 2 bits are needed. (0 & 1)

1 → beam intensity on

0 → " " off.

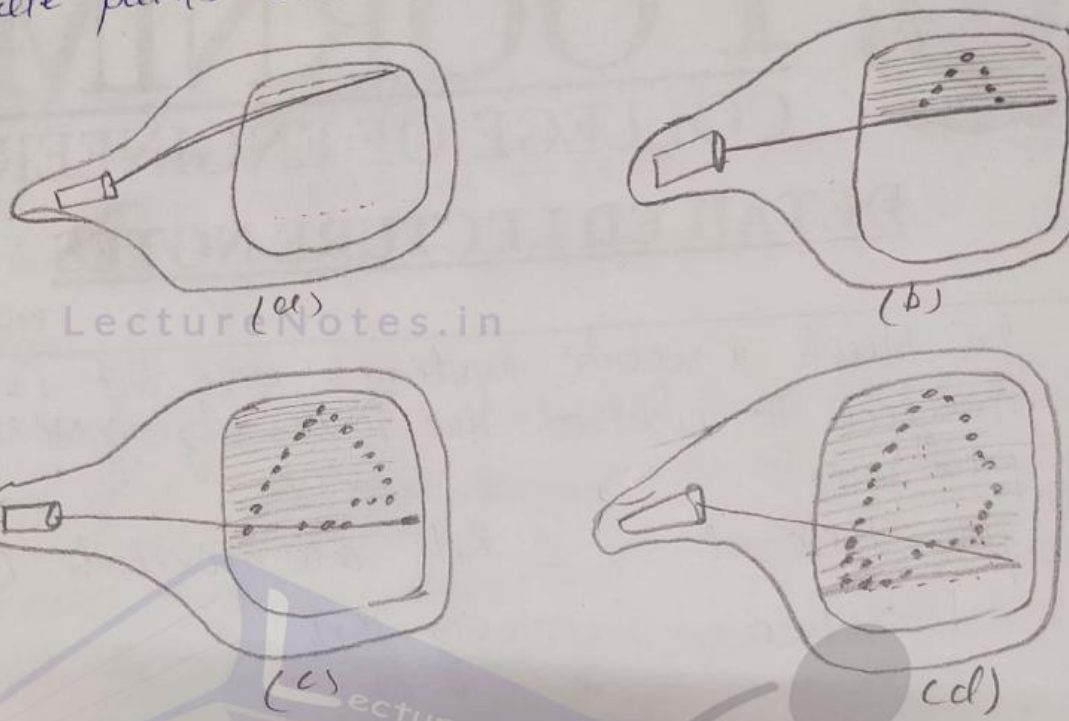
24 bits per pixel are included in high-quality systems.

24 bits per pixel system need screen resolution of 1024 x 1024 are 3 MB of frame buffer.

To refresh buffer, refresh rate is 60-80 frames per second.

After scanning all the horizontal lines, at the end of each frame, the vertical retrace returns to the top left corner of the screen to begin next frame.

A raster scan system displays an object as a se
discrete points across each scan line.



2. Random scan display

In this technique, the beam is directed only to the part of the screen where the picture is to be drawn rather than scanning from left to right.

The picture definition is stored as a set of line drawings commands in refresh display file (buffer). The video controller here, cycles through the set of commands in the display file, drawing each component line in turn. After all line drawing commands are processed, the system cycles back to the first line in the list. Designed to draw component lines of a
[illegible] 30 to 60 times each second.

Computer Graphics And Multimedia

Topic:

Color Monitors

Contributed By:

Ankita Jiyani

Color Monitors
Beam Penetration
Shadow Masking

Color Monitors :-

There are 2 ways to produce color with CRT

(1) Beam penetration method :-

Two phosphor of Red & Green are coated inside of CRT screen.

The color displayed depends on how far electron beam penetrates into the layer.

Slow electron only penetrates to Red.

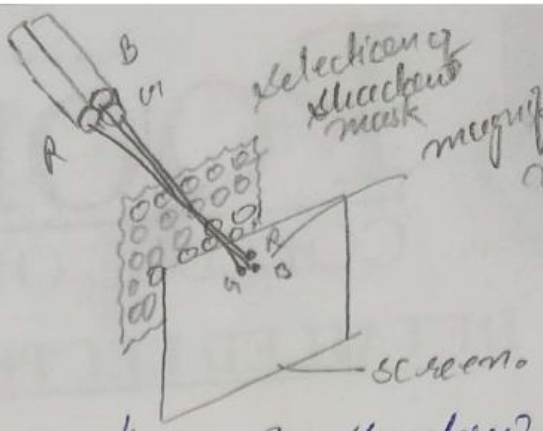
Excited electron penetrates to green.

The speed of electrons is controlled by the beam acceleration voltage. With this only 4 colors are possible.

(2) Shadow mask :

Produces wider range of colors.

It has three phosphor RGB emits R, G, B light & have three electron gun one for each light and a shadow mask grid just behind the phosphor coated screen.



↳ Delta-delta shadow mask method.

when 3 beams passes through shadow mask, they activate a dot triangle, which appears as a small color spot on the screen.

Another method is, 3 electron guns are placed inline arrangement. This is commonly used in high resolution color CRTs.

Different intensity of color can be obtained by turning on & off one or two or three electron guns.

Storage tubes :-

Alternate method for maintaining a screen image is to store the picture info inside the CRT instead of refreshing the screen. A storage tube stores the picture info as a charge distribution just behind the phosphor coated screen.

Two electron guns are used :-

1 → primary gun - store picture pattern.

2 → flood gun - maintain picture display.

Disadv :- Do not display color. It can't be erased. The erasing & redrawing process can take several seconds for a complex picture.

Difference between Raster and Random Scan Systems

Properties	Raster Scan System	Random Scan System
Resolution	Less Resolution because picture definition is stored as a intensity value.	High Resolution picture definition is stored as a set of line of commands.
Electron Beam.	directed from top to bottom & one row at a time on screen	directed to only that part of the screen where picture is required to be drawn.
Cost	less expensive	Costlier.
Refresh Rate	60 to 80 fps	Rate depends on no. of lines to be displayed 30-40 fps.
Picture Definition	Store picture definition in Refresh buffer.	Stored as a set of line of commands for Refresh display file.

Page No.:

Properties	Raster Scan System	Random Scan system
Line Drawing	Zig-Zag line is produced because plotted value are discrete.	Smooth line is produced because directly the line path is followed by electron beam.
Image Drawing	It uses pixels along scan lines for drawing an Image.	It is designed for line drawing apps & uses various mathematical function to draw.

Difference b/w Beam penetration & Shadow Masking.

Properties	Beam Penetration	Shadow Mask
Mainly used	used with random scan system to display colors.	used with raster scan system to display colors.
Colors.	It can display only 4 color, Red, green, orange & yellow	It can display million of colors.

Difference between Beam Penetration and Shadow Masking

Properties	Beam penetration	Shadow Mask
Color Dependency	Color depends on the level of electron penetration into the phosphorus layer.	Color depends on the type of gun.
Cost	less expensive	Costlier.
Picture Quality	Not good	Good quality picture, Realistic.
Resolution	High	Low
Criteria	Color display depends on how far electron enters the outer Red layer & then Green layer.	There is no such criteria for producing colors. It used in computer, T.V.

Computer Graphics And Multimedia

Topic:

Interlacing Techniques

Contributed By:

Ankita Jiyani

Interlacing Technique

Interlacing :- It is a description how the picture is created. In interlaced monitors, the electron beam takes two passes to form a complete image, it skips every other row on the first pass & then goes back & fills the missing rows. Interlaced monitors are easier to build & therefore cheaper. The problem is that all things being equal as non-interlaced monitors, it takes twice as long to create the complete screen image. That's long enough to spoil the illusion that you are looking at steady picture & the image on the screen flickers annoyingly.

Interlaced monitors are needed because earlier, mechanical CRT display devices scans hundred horizontal lines in a frame, And the refresh rate is slower so, it causes headache & eye fatigue & also producing distracting screen flicker.

Flickering :- The appearance of flickering, flashing or unsteadiness in an image may occur

on a display screen. A flicker issue can occur when the refresh rate is too low, or other video related issues.

Refresh Rate :- It is the no. of times per second pixels are recharged so the image doesn't flicker.

A refresh rate of 60 Hz means image is redrawn 60 times a second. The higher the refresh rate, the lower is the flickering. A refresh rate below 75 Hz can produce an often imperceptible flicker that can cause eye strain after long viewing.

Calculation of refresh rate :-

$$f_V = f_H / \#\text{ of horizontal lines} \times 0.95$$

f_V - vertical scanning freq.
f_H - horizontal scanning freq.

Eg:- a monitor with horizontal frequency of 96 kHz at a resolution 1280 × 1024 would have following refresh rate.

$$f_V = \frac{96 \times 1000}{1024 + 0.95}$$

$$f_V = 89.06 = \underline{\underline{89\,Hz}}$$

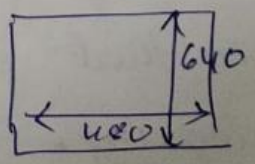
with 1600 × 1200.

$$f_V = \frac{96 \times 1000}{1200 + 0.95} = 76\,Hz.$$

Image Resolution:- It refers to pixel spacing, the distance b/w two pixels.

Screen Resolution:- It refers to the no. of pixels in the horizontal & vertical direction on the screen.

Screen Resolution with 640 × 480.



Increasing the resolution, the image will become smaller due to screen displaying more pixels per inch.

13 inch	640 × 480
15 inch	800 × 600
17 inch	1024 × 768
21 inch	1024 × 768

Computer Graphics And Multimedia

Topic:

Raster Scan Systems

Contributed By:

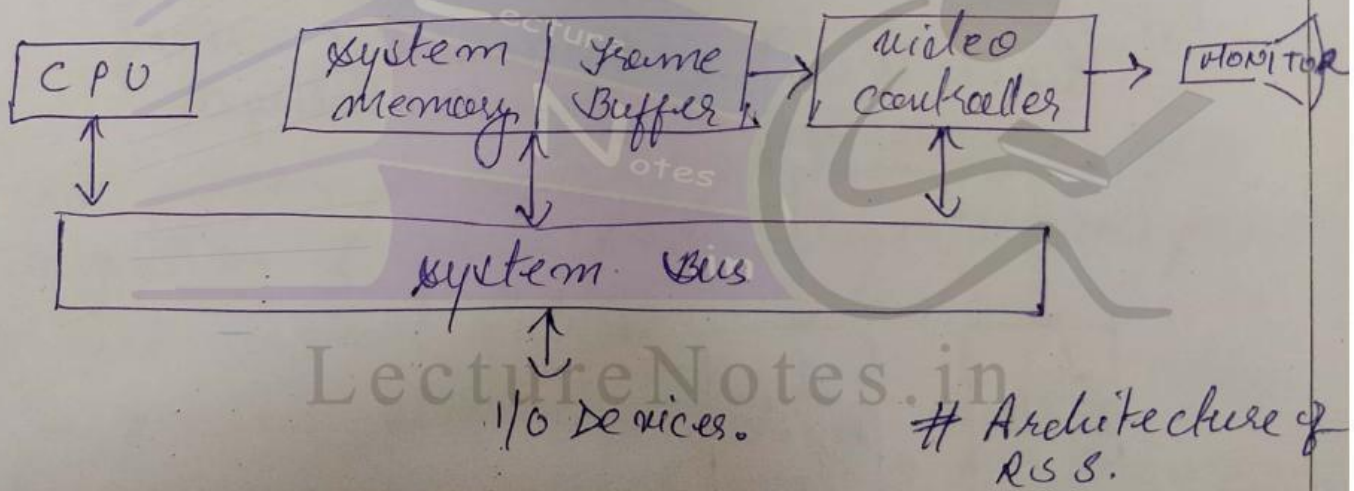
Ankita Jiyani

Raster Scan Systems

Raster scan systems :-

Interactive - allows user to interact with the graphical info presented on the display using some of the input devices.

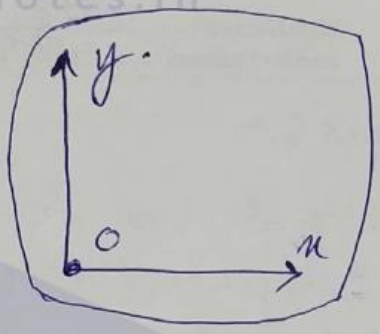
Eg :- video game controller.



Architecture of RSS.

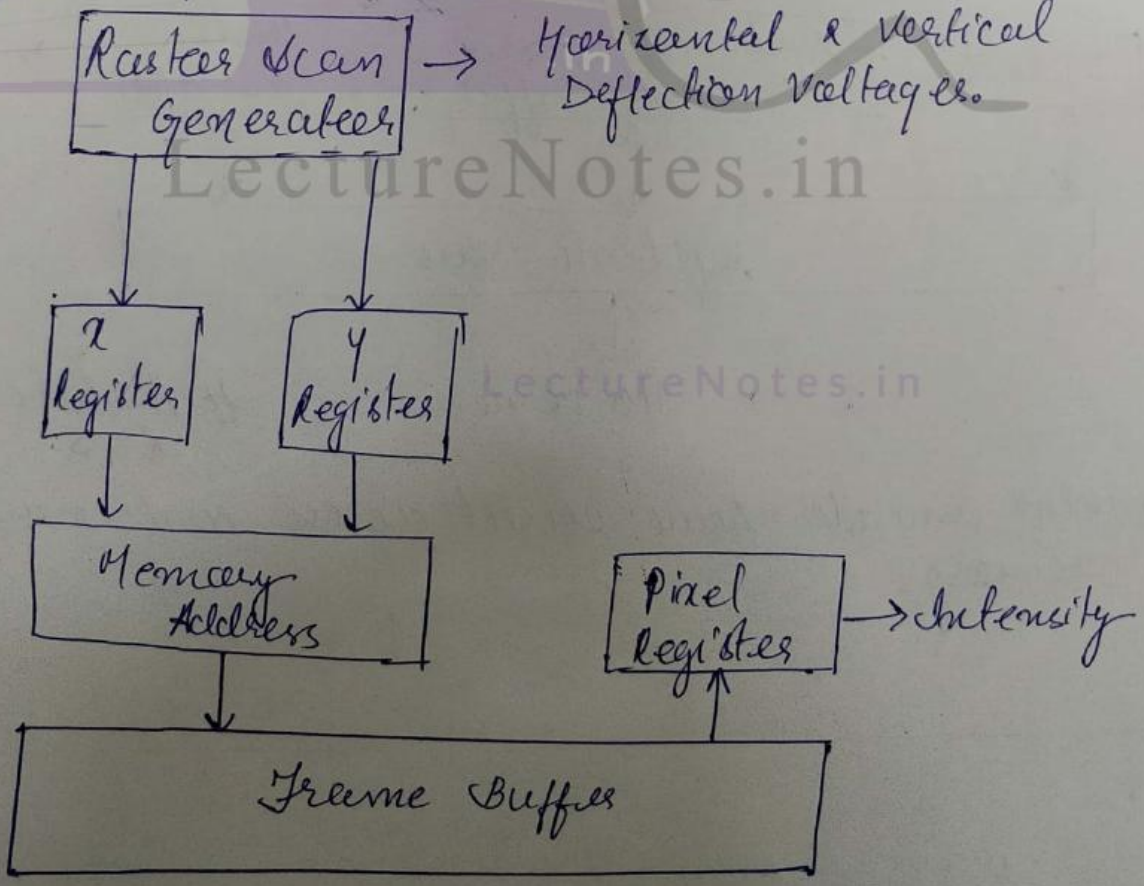
→ video controller have direct access to frame buffer memory.

Frame Buffer - Stores the screen position in form of pixels. And pixels are further stored as a cartesian co-ordinates. Origin of the screen is bottom-left corner.



Origin of the coordinate system.

Refresh Operations;



Refresh Operations

Two registers are used to store coordinates x & y respectively.

x register is set to 0

y register is set to ymax

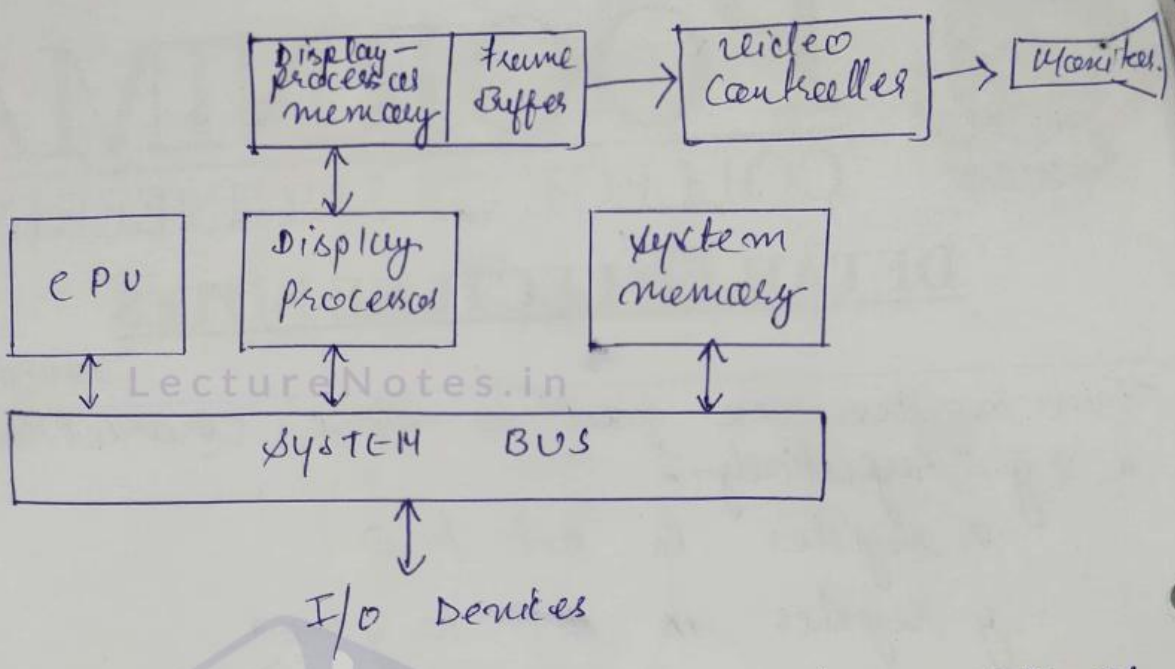
The value stored in frame buffer for the pixel position x is retrieved & used to set the intensity of the CRT beam.

The x register is incremented by 1.

& y register is decremented by 1.

And process is repeated for each pixel. After completion of every pixel, registers are reset to their original values.

The process shown, generally is very slow. To achieve the refresh rate of 60 frames per second. video controller retrieve multiple pixel values from refresh buffer. A separate register is used to store the intensity values.



Display processor - specialized I/O processor which is used to convert digital information from the CPU into the analog value by the display device. It will interpret a display list / frame buffer to produce picture and characters on the screen. which is known as scan conversion.

Display processor also performs various operations:-

generating lines styles
display color areas
perform transformation & manipulation

Computer Graphics And Multimedia

Topic:

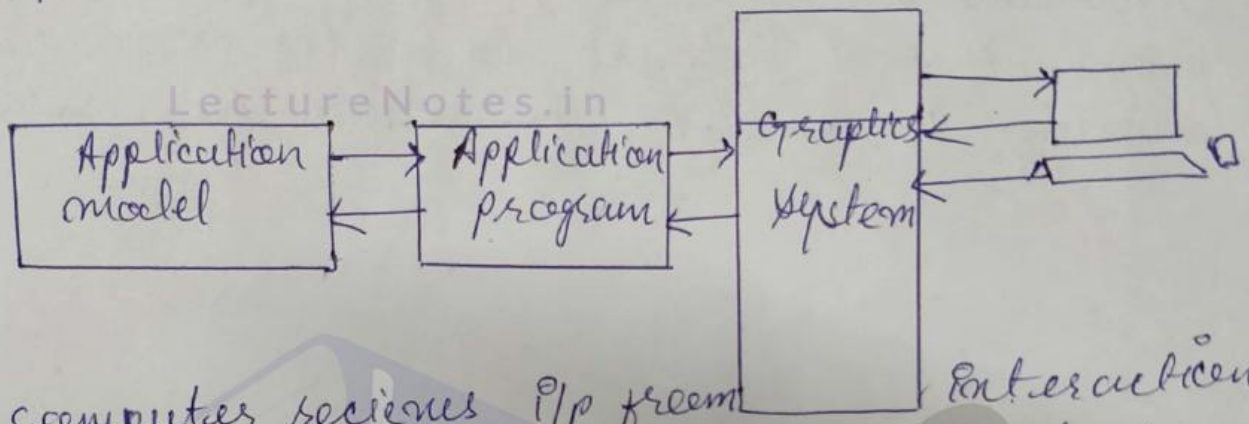
Frame Work For Interactive Graphics

Contributed By:

Ankita Jiyani

Framework for Interactive Graphics

Conceptual framework for interactive graphics:-



A computer receives i/p from device & output image to a display device.

The framework has 3 components;

1. → Application program; It stores the data, retrieves the data, it creates the data to be stored in application model. It handles user i/p.
2. Application model; It will store data or object to be pictured on the screen.

App program, on receiving i/p from user it produces views by sending it to the graphics system.

Cg graphics system, deals with the actual representation of the object on the screen, what to be appeared & how it should appear.

Cgs is intermediate b/w app program & display hardware.

The fundamental task of interactive system is to specify what data or objects are to be generated and how they will be present pictorially. And also, how the user & app program will interact in order to create & modify the model & its visual representation.

Computer Graphics And Multimedia

Topic:

Scan Conversion Or Rsterrization

Contributed By:

Ankita Jiyani

Scan conversion or Rasterisation

Scan conversion :-

The process of representing continuous graphics objects as a collection of discrete pixels is c/a scan conversion.

It is the responsibility of graphics system or the app program to convert each primitive from its geometric definition into a set of pixels.

This conversion task is c/a scan conversion or rasterization.

Scan converting a point :-

The mathematical point (x, y) where x & y are real no. within an image area needs to be scan converted to a pixel at a location (x', y').

$x' = \text{Floor}(x)$ $\quad y' = \text{Floor}(y)$.

Page No.:

The origin is lower left corner of the pixel grid.

points must satisfy following condition:-

$$x' \le x \le x'+1 \quad \&$$

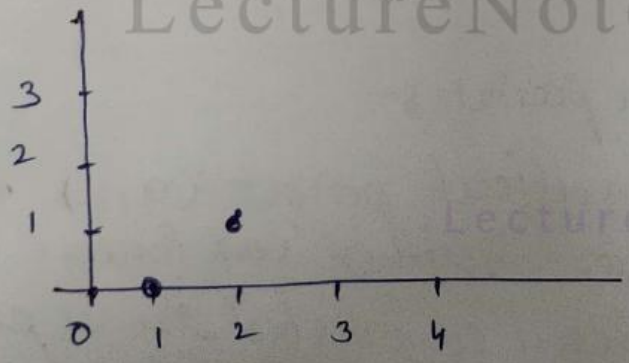
$$y' \le y \le y'+1$$

Eg:- $P_1(1.7, 0.8)$ $P_2(2.2, 1.3)$

$P_3(2.8, 1.9)$

Scan conversion.

	Floor(x, y)
$P_1(1.7, 0.8)$	$P_1(1, 0)$
$P_2(2.2, 1.3)$	$P_2(2, 1)$
$P_3(2.8, 1.9)$	$P_2(2, 1)$



Second Approach;

$x' = \text{Floor}(x + 0.5)$

$y' = \text{Floor}(y + 0.5)$

Digital Diffrenetial Analyzer (DDA) Line Algorithm

PAGE NO. .....

Now for some example :-

$P_1(1.7, 0.8)$ → $P_1(2, 1)$

$P_2(2.2, 1.3)$ → $P_1(2, 1)$

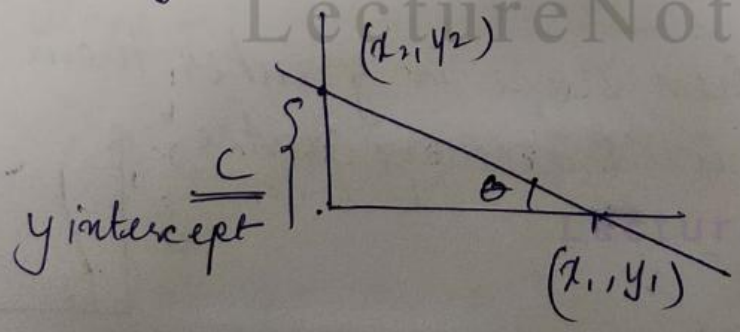
$P_3(2.8, 1.9)$ → $P_3(3, 2)$

y(int)(x+0.5); i=int(y+0.5)

Screen consists of line :-

DDA (Digital differential Analyzer) Algorithm.

$y = mx + c$.



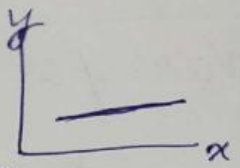
m - slope = $\tan\theta = \frac{P}{B}$.

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

$$m = \frac{\Delta y}{\Delta x}$$

Now we are considering 3 cases;

when slope is less than or equal to one which means, x co-ordinate is increasing faster than y co-ordinate.



so, x is always increasing by 1.

$$x_2 - x_1 = 1$$

or, $x_{k+1} - x_k = 1$

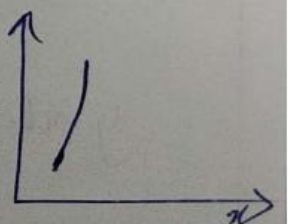
or, $\Delta x = 1.$

now, $m = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{1}$

or, $m = \frac{y_{k+1} - y_k}{1}$

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

Second case; when slope is greater than one. which means y is increasing faster;



here, $y_2 - y_1 = 1$

$\Delta y = 1$

now, $m = \frac{\Delta y}{\Delta x} = \frac{1}{x_2 - x_1}$

DDA Line Drawing Example

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

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

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

Example :-

Suppose we have following co-ordinates, for the line :- $P_1(2, 4)$ & $P_2(5, 7)$.

$x_1 = 2, \quad x_2 = 5$

$y_1 = 4, \quad y_2 = 7.$

Slope = $m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{7-4}{5-2} = \frac{3}{3} = 1.$

Case 1, here, x is increasing faster.

$y_{k+1} = y_k + m = 4 + 1 = 5$

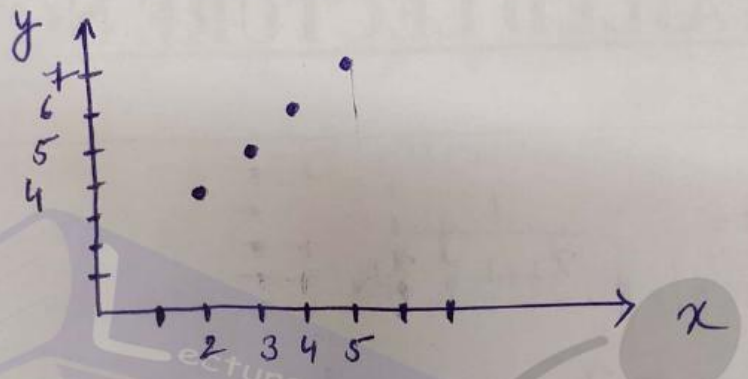
$y_{k+1} = y_k + m = 5 + 1 = 6$

$y_{k+1} \quad y_k + m = 6 + 1 = 7$

points for the pixels are :-

$P_1(2, 4)$ $P_2(3, 5)$ $P_3(4, 6)$

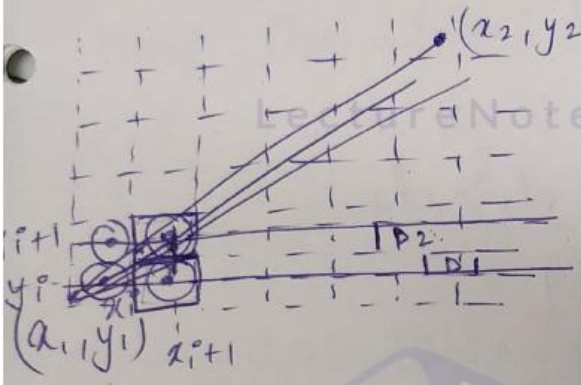
$P_4(5, 7)$.



Bresenhem's Line Drawing Algorithm

Name of Faculty:

Bresenham's Line Algorithm :-



Assumptions before starting the derivation of algo :-

Assumption 1 - two endpoints (x_1, y_1) & (x_2, y_2)

Assumption 2 - Slope is $< 45°$

$0 \leq m \leq 1$

Assumption 3, $x_1 < x_2$

Assumption 4, $y = mx + b$

For, $x_i + 1$

y for $x_i + 1$ is :- $y = m(x_i + 1) + b$

& $m = \frac{\Delta y}{\Delta x}$

$$\boxed{d_1 = y - y_i = m(x_i + 1) + b - y_i}$$

Distance b/w y & y_i

Page No.:

$d_2 = y_{i+1} - y = y_{i+1} - m(x_i+1) + b$

$d_1 - d_2 = m(x_i+1) + b - y_i - (y_{i+1} - m(x_i+1))$

$= 2m(x_i+1) - 2y_i + 2b - 1$

This difference will help us to know which pixel is nearer to the line.

If ; $d_1 - d_2 < 0$; i.e $d_1 < d_2$

choose pixel, y_i

so, $y_{i+1} = y_i$

If ; $d_1 - d_2 > 0$; i.e $d_1 > d_2$

choose pixel, $y_i + 1$

so, $y_{i+1} = y_i + 1$.

$P_i = \Delta x (d_1 - d_2)$; multiply with Δx to avoid fractional Calculation.

$= (2m(x_i+1) - 2y_i + 2b - 1) \times \Delta x$

$\Delta x \left(2 \times \frac{\Delta y}{\Delta x} (x_i+1) - 2y_i + 2b - 1 \right)$

$$P_i = 2\Delta y\,(x_i) - 2\Delta x \cdot y_i + \underbrace{2\Delta y + 2\Delta x\, b - \Delta x}_{\text{constant}}$$

$$P_i = 2\Delta y\, x_i - 2\Delta x \cdot y_i + c$$

next iteration;

$$P_{i+1} = 2\Delta y\, x_{i+1} - 2\Delta x \cdot y_{i+1} + c$$

subtract; $P_{i+1} - P_i$

$$= 2\Delta y\, x_{i+1} - 2\Delta x\, y_{i+1} + c - 2\Delta y x_i + 2\Delta x y_i - c$$

$$= 2\Delta y\,(x_{i+1} - x_i) - 2\Delta x(y_{i+1} - y_i)$$

$x_{i+1} - x_i = 1$

$$= 2\Delta y - 2\Delta x(y_{i+1} - y_i)$$

$$P_{i+1} = P_i + 2\Delta y - 2\Delta x(y_{i+1} - y_i)$$

if $P_i < 0$ $\quad P_{i+1} = P_i + 2\Delta y$

$P_i > 0$ $\quad P_{i+1} = P_i + 2\Delta y - 2\Delta x$.

Initial parameters;

$$P_0 = 2\Delta y - \Delta x.$$

$$P_0 = 2\Delta y\, x_0 - 2\Delta x y_0 + 2\Delta y + 2\Delta x\, b - \Delta x$$

$$b = y_0 - \left(\frac{\Delta y}{\Delta x}\right) x_0$$

$$= 2\Delta y x_0 - 2\Delta x y_0 + 2\Delta y + 2\Delta x * \left(y_0 - \left(\frac{\Delta y}{\Delta x}\right) x_0\right) +$$

$$= 2\Delta y x_0 - 2\Delta x y_0 + 2\Delta y + 2\Delta x y_0 -$$

$$2\Delta x \times \left(\frac{\Delta y}{\Delta x}\right) x_0 - \Delta x$$

$$= 2\Delta y x_0 - 2\Delta x y_0 + 2\Delta y + 2\Delta x y_0 - 2\Delta y x_0 - \Delta x$$

$$\boxed{= 2\Delta y - \Delta x}$$

Example of Bresenham's Algo :-

$(x_1, y_1) = (20, 10)$ $(x_2, y_2) = (30, 18)$

$m = 0.8$

$\Delta x = 10,$ $\Delta y = 8$

$(30 - 20)$ $(18 - 10)$

Bresenhem's Line Drawing Example 1

$P_0 = 2\Delta y - \Delta x$

$= 2 \times 8 - 10 = 16 - 10 = 6.$

i	p_i	x_{i+1}, y_{i+1}
0	6	21, 11
1	2	22, 12
2	-2	23, 12
3	14	24, 13
4	10	25, 14
5	6	26, 15
6	2	27, 16
7	-2	28, 16
8	14	29, 17
9	10	30, 18

$p_i > 0$

$y_{i+1} = y_i + 1$

$= y_1 + 1 = 10 + 1 =$

Example :- Draw a line b/w (5,12) & (15,20) with BLDA.

Step1- calculate slope;

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{8}{10} = 0.8 < 1$$

$P_i = 2\Delta x - \Delta x$

$P_{i+1} = P_i + 2\Delta y - 2\Delta x (y_{i+1} - y_i)$

$P_i \geq 0$

$y_{i+1} = y_i + 1$

$P_i < 0$

$y_{i+1} = y_i$

If $m > 1$

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

$P_{i+1} = P_i + 2\Delta x - 2\Delta y (x_{i+1} - x_i)$

$P_i \geq 0$

$x_{i+1} = x_i + 1$

$P_i < 0$

$x_{i+1} = x_i$

$P_0 = 2 \times 8 - 10 = 6$

$P_1 = P_0 + 2\Delta y - 2\Delta x (y_{i+1} - y_i)$

$6 + 2 \times 8 - 2 \times 10 (13 - 12) = 6 + 16 - 20 = 2$

$P_2 = P_1 + 2 \times 8 - 2 \times 10 (14 - 13) = 2 + 16 - 20 = -2$

$P_3 = P_2 + 2 * 8 - 2 * 10 (14 - 14) = -2 + 16 - 20 * 0 = 14$

$P_4 = P_3 + 2 * 8 - 2 * 10 (15 - 14) = 14 + 16 - 20 = 10$

$P_5 = P_4 + 2 * 8 - 2 * 10 (16 - 15) = 10 + 2 * 8 - 20 = 6$

$P_6 = P_5 + 16 - 20 (17 - 16) = 6 + 16 - 20 = 2$

$P_7 = 2 + 16 - 20 (18 - 17) = -2$

$P_8 = -2 + 16 - 20 (0) = 14$

$P_9 = 14 + 16 - 20 (19 - 18)$

$14 + 16 - 20 = 10.$

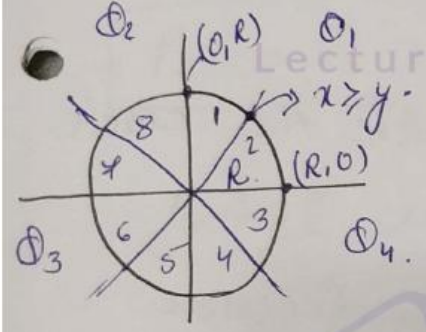
PAGE NO. .....

Tables for the points.

Nc.	x_i	y_i	p_i	x_{i+1}	y_{i+1}
0	5	12	6	6	13
1	6	13	2	7	14
2	7	14	-2	8	14
3	8	14	14	9	15
4	9	15	10	10	16
5	10	16	6	11	17
6	11	17	2	12	18
7	12	18	-2	13	18
8	13	18	14	14	19
9	14	19	10	15	20
10					
11					

Mid Point Circle Drawing Algorithm

Mid Point circle Algo :-



circle is having 8 fold symmetry property.

Divide the circle into 8 octets.

Equation of circle :-

$$x^2 + y^2 - R^2 = 0$$

Here, x - increase unit interval.

y - can increase/decrease.

Next points can be :-

$$(x_k + 1, y_k) \text{ or } (x_k + 1, y_k - 1)$$

which point to choose depends on decision parameter.

Mid Point is :- $\frac{x_k + 1 + x_k + 1}{2}, \frac{y_k + y_k - 1}{2}$

$$= x_k + 1, y_k - \frac{1}{2}$$

Page No.:

$P_k = f\left(x_k + 1, y_k - \frac{1}{2}\right)$

$f(x,y) = x^2 + y^2 - R^2$

put the value of P_k in $f(x,y)$

$$(x_k+1)^2 + \left(y_k - \frac{1}{2}\right)^2 - R^2 = 0$$

$P_k < 0$, mid point is inside the circle

y_k is closes.

if $P_k \geq 0$, mid point is outside the circle

$y_k - 1$ is closes.

Similarly, we need successive decision parameters.

$$P_{k+1} = f\left(x_{k+1} + 1, y_{k+1} - 1/2\right)$$

$$= (x_{k+1}+1)^2 + (y_{k+1} - 1/2)^2 - R^2$$

$$P_{k+1} - P_k = (x_{k+1}+1)^2 + (y_{k+1} - 1/2)^2 - R^2 - (x_k+1)^2 - (y_k - 1/2)^2 + R^2$$

$$((x_k+1)+1)^2 + (y_{k+1} - 1/2)^2 - (x_k+1)^2 - (y_k - 1/2)^2$$

$$(x_k+1)^2 + 1 + 2(x_k+1) + (y_{k+1})^2 + \frac{1}{4} - 2 \times \frac{1}{2}(y_{k+1}) - (x_k+1)^2 - (y_k)^2 + \frac{1}{4} + y_k$$

$$P_{k+1} - P_k = 2(x_k + 1) + (y_{k+1})^2 - (y_k)^2 - y_{k+1} + y_k + 1$$

$$= 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$$

$$\boxed{P_{k+1} = P_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1}$$

Initial decision parameters:-

$$x_k = 0, \quad y_k = r$$

$$P_k = (x_k + 1)^2 + (y_k - 1/2)^2 - r^2$$

$$(0 + 1)^2 + (r - 1/2)^2 - r^2$$

$$1 + r^2 + \frac{1}{4} - 2 \times \frac{1}{2} r - r^2$$

$$1 + \frac{1}{4} - r = \frac{5}{4} - r$$

$$\boxed{P_0 = \frac{5}{4} - r}$$

or, $\frac{1.25 - r}{(1)}$

$1 - r$

Mid Point Circle Drawing Example

$P_k \geq 0$, NC $= y_{k+1} = y_k - 1$

$(x_k + 1, y_k - 1)$

$P < 0$, NC $= y_{k+1} = y_k$

$(x_k + 1, y_k)$

Examples:

$P_k = 1 - r$.

$P_{k+1} = P_k + 2(x_k + 1) + (y_{k+1}^2 - y_k)^2 - (y_{k+1} - y_k) + 1$

Given radius = 8.

$P_{10} = 1 - r = 1 - 8 = -7$

$P_{k+1} = -7 + 2(0+1) + (8^2 - 8^2) - (8 - 8) + 1$

$-7 + 2 + 0 + 0 + 1 = -7 + 3 = -4$

$P_2 = -4 + 2(1+1) + 0 - 0 + 1$

$-4 + 4 + 1 = 1$

$P_3 = 1 + 2(2+1) + (49 - 64) - (7 - 8) + 1$

$1 + 6 - 15 + 1 + 1 = -6$

$P_4 = -6 + 2(3+1) + (7^2 - 7^2) - (7 - 7) + 1$

$-6 + 8 + 1 = 3$

$P_5 = 3 + 2(4+1) + (36 - 49) - (6 - 7) + 1$

$3 + 10 - 13 + 1 + 1 = 2$

k iteration	x_k	y_k	P_k	x_{k+1}	y_{k+1}
0	0	8	−7	1	8
1	1	8	−4	2	8
2	2	8	1	3	7
3	3	7	−6	4	7
4	4	7	3	5	6
5	5	6	2	6	5

$(x > y)$ end of 1st octet.

- whenever, $x > y$, (−x, 4).

Quadrant 1. (x, y)

(0, 8)	(7, 4)
(1, 8)	(7, 3)
(2, 8)	(8, 2)
(3, 7)	(8, 1)
(4, 7)	(8, 0)
(5, 6)	
(6, 5)	

Quadrant 2.

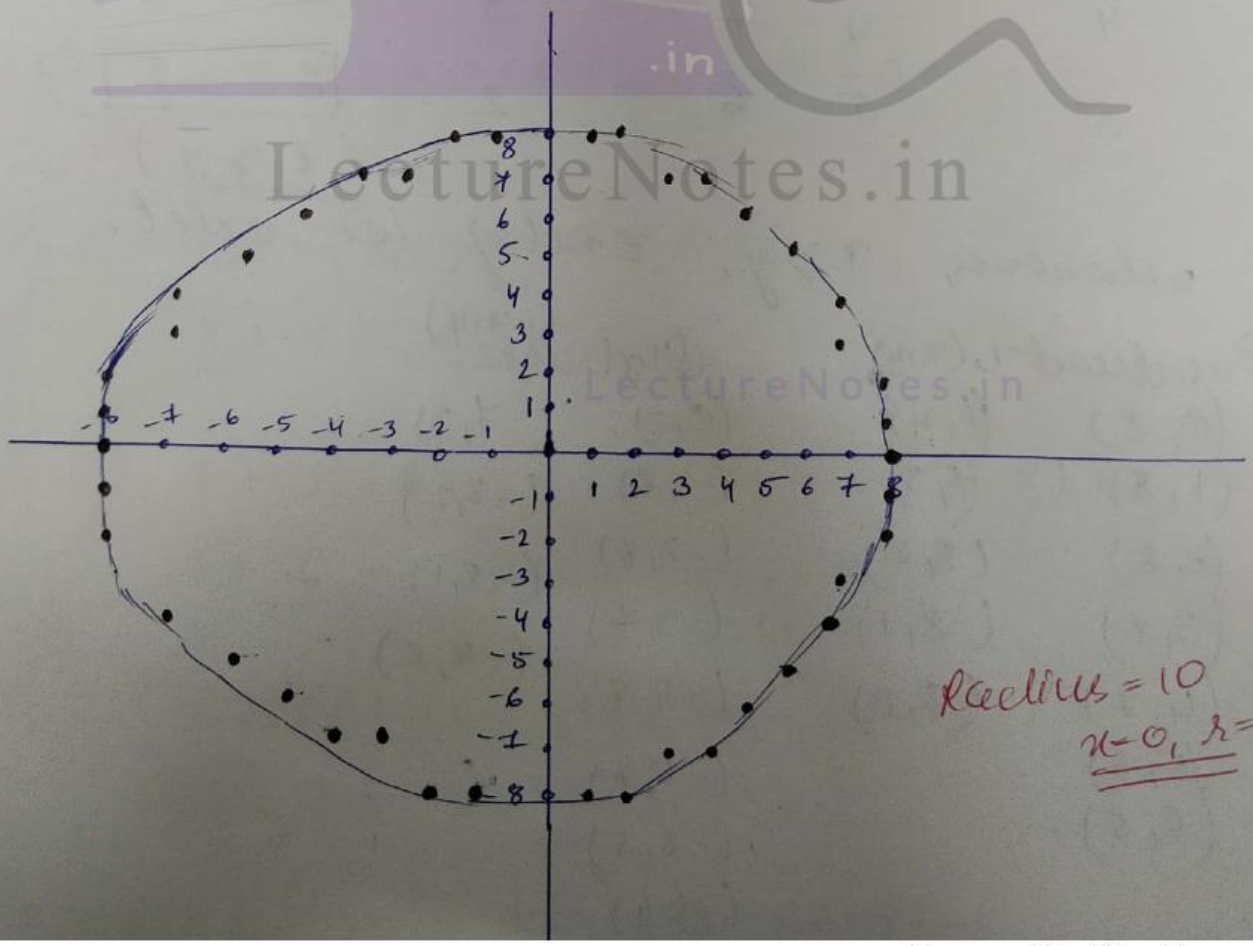
(0, 8)	(−7, 3)
(−1, 8)	(−8, 2)
(−2, 8)	(−8, 1)
(−3, 7)	(−8, 0)
(−4, 7)	
(−5, 6)	
(−6, 5)	
(−7, 4)	

Quadrant 3 $(-x, -y)$

$(0, 8)$
$(-1, -8)$
$(-2, -8)$
$(-3, -7)$
$(-4, -7)$
$(-5, -6)$
$(-6, -5)$
$(-7, -4)$
$(-7, -3)$
$(-8, -2)$
$(-8, -1)$
$(-8, 0)$

Quadrant 4 $(x, -y)$

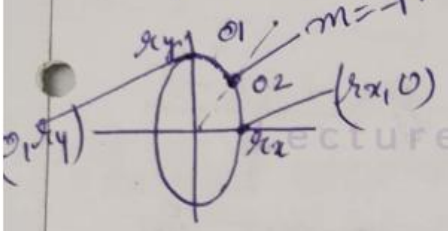
$(0, 8)$
$(1, -8)$
$(2, -8)$
$(3, -7)$
$(4, -7)$
$(5, -6)$
$(6, -5)$
$(7, -4)$
$(7, -3)$
$(8, -2)$
$(8, -1)$
$(8, 0)$



Radius = 10
$x = 0, x = y$

Mid Point Ellipse Drawing Algorithm

Midpoint Ellipse Drawing Algorithm :-



Follow 2 different procedures :-

O1, slope, $m < 1$

O2, slope, $m > 1$

If $m < 1$, x moves unit interval

y_{i+1} ?, y_i, $y_i - 1$

NC → (x_{i+1}, y_i) $(x_{i+1}, y_i - 1)$ } O1

If $m > 1$, y moves unit interval.

x_{i+1} ?, x_i, $x_i + 1$

NC → $(x_i, y_i + 1)$ $(x_i + 1, y_i - 1)$ } O2

Now where to stop octet 1 & start octet 2

Eq^n of ellipse :-

$$\frac{x^2}{r_x^2} + \frac{y^2}{r_y^2} = 1$$

Page No.:

$$x^2 r_y^2 + y^2 r_x^2 - r_x^2 r_y^2 = 0 \quad \text{--- (1)}$$

$$m = \frac{dy}{dx}$$

$$2x r_y^2 \dot{x} \geq 2 r_x^2 y \dot{}$$

Octet I. $m < 1$

Expected points, $(x_k + 1, y_k)$ $(x_k + 1, y_k - 1)$

midpoint is :-

$$\frac{x_k + 1 + x_k + 1}{2}, \frac{y_k + y_k - 1}{2}$$

$$= \frac{x_k + 1}{2}, y_k - 1/2 \rightarrow \text{put in eq. 1.}$$

Decision parameter :-

$$P_{1K} = (x_k + 1)^2 r_y^2 + (y_k - 1/2)^2 r_x^2 - r_x^2 r_y^2 = 0$$

$$P_{1k+1} = \left(\frac{x_{k+1}}{x_k + 1} + 1\right)^2 r_y^2 + (y_{k+1} - 1/2)^2 r_x^2 - r_x^2 r_y^2 = 0$$

$$(x_{k+1} + 2)^2 r_y^2 + (y_{k+1} - 1/2)^2 r_x^2 - r_x^2 r_y^2 = 0$$

$$P_{1k+1} - P_{1K} = (x_{k+1})^2 r_y^2 + r_y^2 + 2(x_k + 1) r_y^2 + y_{k+1}^2 r_x^2 + 1/4 r_x^2 - y_{k+1} r_x^2 - r_x^2 r_y^2 - (x_k + 1)^2 r_y^2 - y_k^2 r_x^2 - \frac{1}{4} r_x^2 + y_k r_x^2 + r_x^2 r_y^2$$

$$p_{ik+1} = p_{ik} + r_y^2 + 2(x_k+1)r_y^2 + r_x^2(y_{k+1}^2 - y_k^2) - r_x^2(y_{k+1} - y_k)$$

→ Decision parameter for 1st octet.

Initial decision parameter:-

substitute $(0, r_y)$ in p_{ik}

$$p_{ik} = (0+1)^2 r_y^2 + (r_y - 1/2)^2 r_x^2 - r_x^2 r_y^2$$

$$= r_y^2 + r_y^2 r_x^2 + \frac{r_x^2}{4} - r_y r_x^2 - r_x^2 r_y^2$$

$$p_0 = r_y^2 + \frac{r^2 x}{4} - r_y r_x^2$$

step when, $2r_y^2 x \geq 2r_x^2 y$.

$p_{ik} \geq 0$, (x_k+1, y_k-1)

$p_{ik} < 0$ (x_k+1, y_k)

Octet 2 :- $m > 1$,

Expected points :- (x_k, y_k-1)

(x_k+1, y_k-1)

midpoint :- $(x_k + 1/2, y_k - 1)$

put this in eq 1.

$$P_{2k} = (x_k + 1/2)^2 r_y^2 + (y_k - 1)^2 r_x^2 - r_x^2 r_y^2 = 0$$

$$x_k^2 r_y^2 + \frac{1}{4} r_y^2$$

$$P_{2k+1} = (x_{k+1} + 1/2)^2 r_y^2 + (y_{k+1} - 1)^2 r_x^2 - r_x^2 r_y^2 = 0$$

$y_k - 1$

$$(x_{k+1} + 1/2)^2 r_y^2 + ((y_k - 1) - 1) r_x^2 - r_x^2 r_y^2 = 0$$

$$P_{2k+1} - P_{2k} = x_{k+1}^2 r_y^2 + \frac{r_y^2}{4} + x_{k+1} r_y^2 + (y_k - 1)^2 r_x^2 + r_x^2 - 2(y_k - 1) r_x^2 - r_x^2 r_y^2 - x_k^2 r_y^2 - \frac{r_y^2}{4} - x_k r_y^2 - (y_k - 1)^2 r_x^2 + r_x^2 r_y^2$$

$$P_{2k+1} = P_{2k} + r_x^2 - 2 r_x^2 (y_k - 1) + r_y^2 (x_{k+1}^2 - x_k^2) + r_y^2 (x_{k+1} - x_k)$$

Can't derive critical decision parameters.

If $P_{2k} \geq 0$, $(x_k, y_k - 1)$

$P_{2k} < 0$, $(x_{k+1}, y_k - 1)$

Mid Point Ellipse Drawing Example

Example :-

Region 1 :-

$$P_{1k} = r_x^2 y + \frac{r_x^2}{4} - r_y r_x^2$$

$$P_{1k+1} = P_{1k} + r_y^2 + 2(x_k+1) r_y^2 + r_x^2 (y_{k+1}^2 - y_k^2) - r_x^2 (y_{k+1} - y_k)$$

$P_{1k} \geq 0 \quad (x_{k+1}, y_k - 1)$

$P_{1k} < 0 = (x_k + 1, y_k)$

Region 2 :-

$$P_{2k} = (x_k + 1/2)^2 r_x^2 + (y_k - 1)^2 r_y^2 - r_x^2 r_y^2$$

$$P_{2k+1} = P_{2k} + r_x^2 - 2 r_x^2 (y_k - 1) + r_y^2 (x_{k+1}^2 - x_k^2) + r_y^2 (x_{k+1} - x_k)$$

$P_{2k} \geq 0 \Rightarrow (x_k, y_k - 1)$

$P_{2k} < 0 \Rightarrow (x_k + 1, y_k - 1)$

$r_x = 8, \quad r_y = 6.$

Region 1. $\quad P_{10} = -332 < 0.$

k	x_k, y_k	P_{ik}	x_{k+1}, y_{k+1}	$2x_{k+1} r_y^2$	$2y_{k+1} r_x^2$
0	0 6	-332	1 6	72	768
1	1 6	-224	2 6	144	768
2	2 6	-44	3 6	216	768
3	3 6	208	4 5	288	640
4	4 5	-108	5 5	360	640
5	5 5	288	6 4	432	512
6	6, 4	244	7 .3	504 >	384

stop.

Region 2 $\quad P_{20} = -23 < 0$

7	7, 3	-23	8 2	
8	8 2	361	8 1	
9	8 1	297	8, 0.	

Computer Graphics And Multimedia

Topic:

Aliasing And Antialiasing

Contributed By:

Ankita Jiyani

Aliasing and Antialiasing

In computer graphics, antialiasing is a s/w technique for diminishing jaggies, stair step like lines that should be smooth. Jaggies occur because the o/p device the monitor or printer doesn't have enough resolution to represent a smooth line.

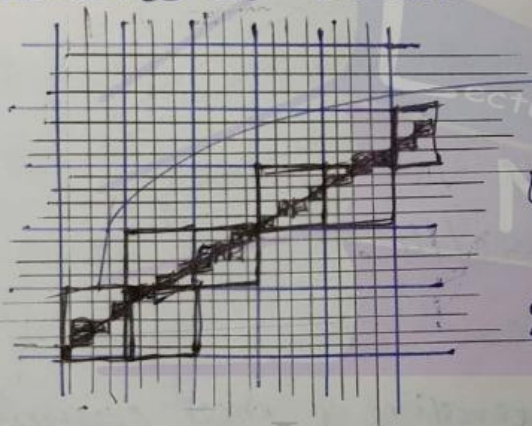
Anticaliasing reduces the prominence of jaggies by surrounding the staircases with intermediate shades of gray, or color. It reduces the jagged appearance of the lines but also makes them fuzzier.

Another method for reducing jaggies is of smoothening in which the printer changes the size & horizontal alignment of dots to make curves smoother.

PAGE NO. .....

Aliasing & Anti Aliasing.

When scan converting the pictures, jaggies appear on the screen, is k/a aliasing. The digital information from the CPU is sampled at low frequency due to which this phenomena occurs & visible on the screen.



→ Jaggies

→ No smooth edges.

→ e.g. playing high graphics game at low resolution.

The application or techniques which removes aliasing is referred as antialiasing.

Antialiasing Techniques

Anti Aliasing techniques :-

1. Super sampling or Post filtering
2. Area sampling or pre filtering.

Super sampling deals with fine grid. we are increasing the frequency of the sampling.

It will smooth the line, remove some of the jaggies.

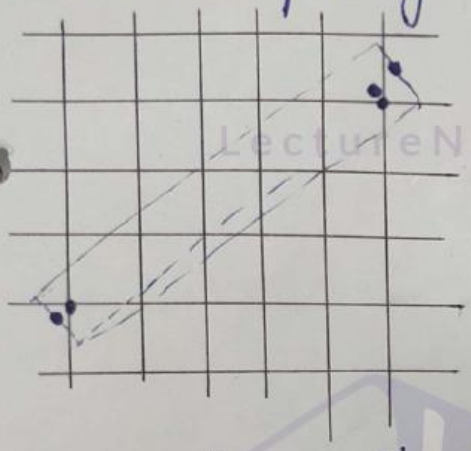
The line passes through twice as many columns, least hits twice many jags, the jag is half as the previous one.

Disadvantages :-

- It requires more cost for memory
- more memory bandwidth.
- Scan conversion time.
- Higher resolution is expensive & not completely removing the problem of jags.

Area Sampling

Area Sampling :-

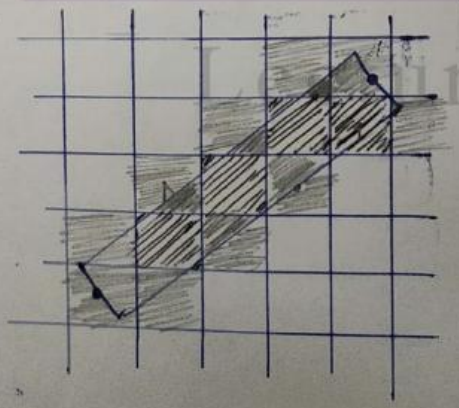


we consider the line as a rectangle of required thickness covering a portion of a grid.

we assume a line contributes to each pixel's intensity an amount proportional to the percentage of the pixel's tile it covers.

Fully covered pixel colored black

partially covered pixel colored gray



This blurring makes a line look better at a distance.

This technique is k/a unweighted area sampling.

Computer Graphics And Multimedia

Topic:

Basic Transformations In Graphics

Contributed By:

Ankita Jiyani

Transformations in Graphics

Basic Transformations :-

Translation :-

It is applied to the object by repositioning it along a straight line path from one co-ordinate location to other.

· A 2D object can be translated by adding a translation distances, t_x & t_y to other position.

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

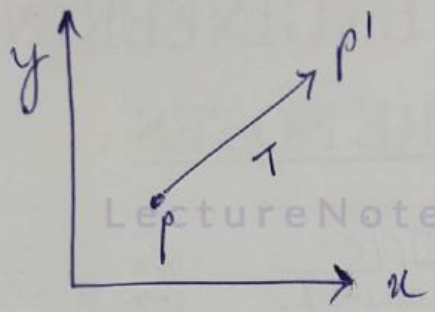
$t_x, t_y \rightarrow$ translation vectors or shift vectors.

$$P = \begin{bmatrix} x \\ y \end{bmatrix} \quad P' = \begin{bmatrix} x' \\ y' \end{bmatrix} \quad T = \begin{bmatrix} t_x \\ T_y \end{bmatrix}$$

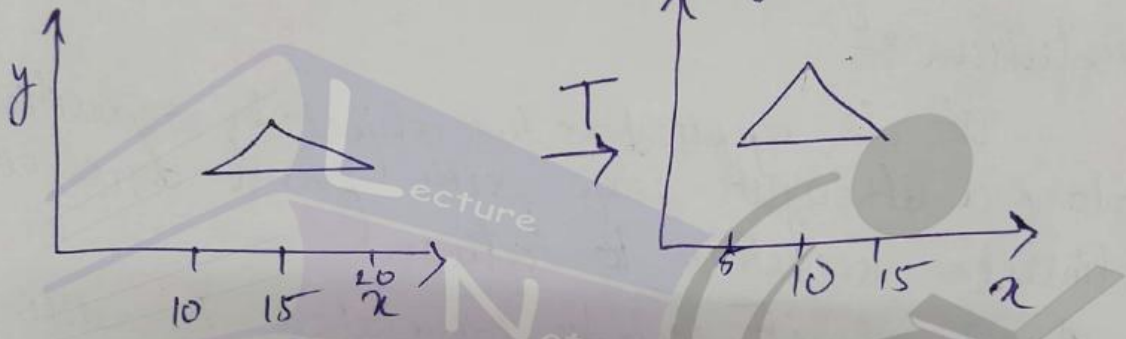
$$P' = P + T$$

Translation is rigid body transformation, that moves the object without deformation

Every point is translated by the same



A line is translated point



If we want to translate a circle, then we translate the centre co-ordinates & redraw the figure in new location.

Rotation :-

It is applied along a circular path in an xy plane.

For rotation, we specify an angle θ the rotation angle.

a, position the rotation point (x_r, y_r)

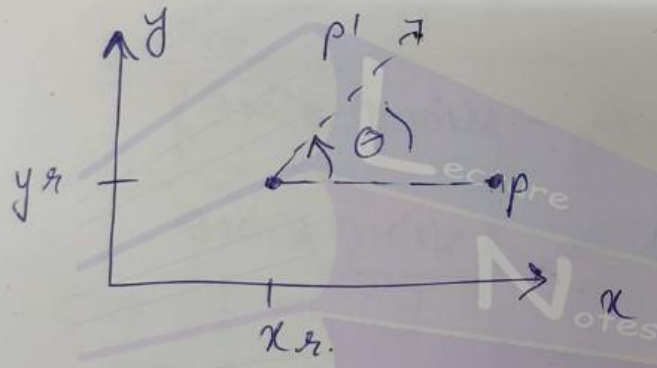
Rotation in Graphics

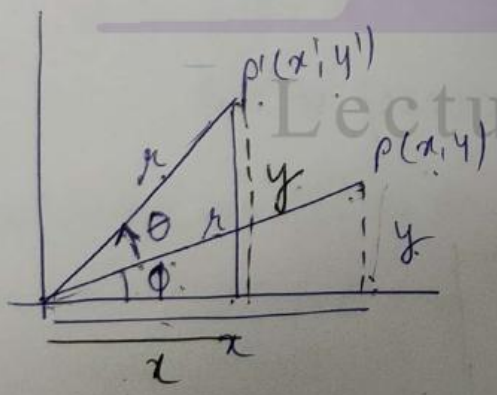
PAGE NO. .....

Also k/a pivot point, about which the object is supposed to be rotated.

positive values - counter clockwise direction

negative values - clockwise direction.





$$\cos\phi = \frac{x}{r} = \frac{B}{H}$$

$$x = r\cos\phi$$

$$\sin\phi = \frac{y}{r} = \frac{P}{H}$$

$$y = r\sin\phi$$

New Angel is :- $\phi + \theta$

P to P'

$\cos(\phi+\theta) = \frac{x'}{r}$

$\sin(\phi+\theta) = \frac{y'}{r}$

$\cos(A+B) = \cos A \cos B - \sin A \sin B$

$\sin(A+B) = \sin A \cos B + \cos A \sin B$

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

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

$x' = r[\cos\phi\cos\theta - \sin\phi\sin\theta]$

$\frac{r}{r}\cos\phi\cos\theta - \frac{r}{y}\sin\phi\sin\theta$

$y' = r[\sin\phi\cos\theta + \cos\phi\sin\theta]$

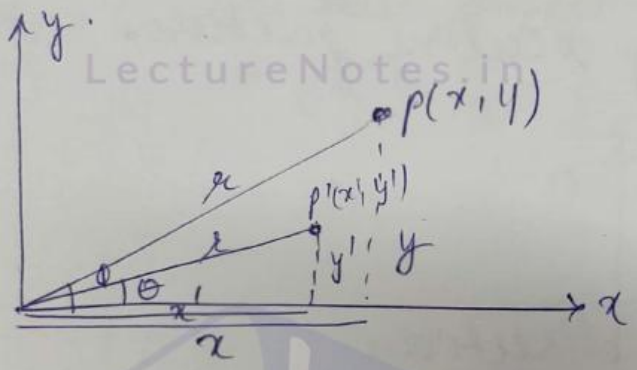
$= r\sin\phi\cos\theta + r\cos\phi\sin\theta$

$$\boxed{\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \end{aligned}}$$

$$P' = R \cdot P$$

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

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



clockwise rotation.

New Angle → $\phi - \theta$

$$\cos(\phi - \theta) = \frac{x'}{r}$$

$$\cos(A-B) = \cos A \cos B + \sin A \sin B$$

$$\sin(\phi - \theta) = \frac{y'}{r}$$

$$\sin(A-B) = \sin A \cos B - \cos A \sin B$$

$$x' = r\cos\phi\cos\theta + r\sin\phi\sin\theta$$

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

$$y' = r\sin\phi\cos\theta - r\cos\phi\sin\theta$$

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

$$P' = R \cdot P$$

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

Scaling :-

It alters the size of an object. It is usually carried out by multiplying the vertex values by scaling factors.

$$x' = x \cdot S_x$$
$$y' = y \cdot S_y$$

S_x - scales in x direction.
S_y - scales in y direction.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

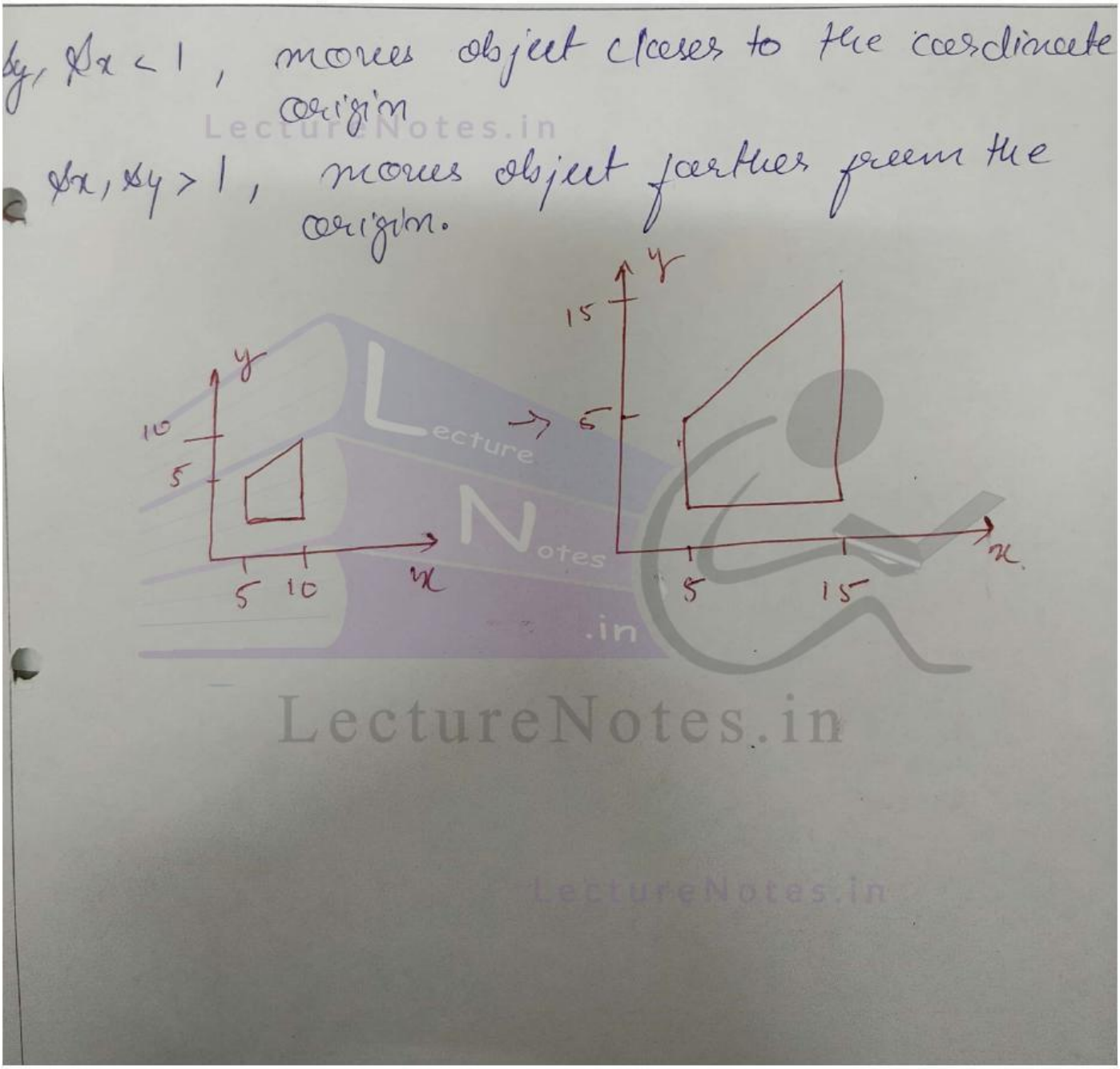
$$P' = S \cdot P$$

$S_x, S_y < 1$ → Reduce the size of object

$S_x, S_y > 1$ → Increase the size of the object.

$S_x, S_y = 1$ → Uniform scaling

Scaling in Graphics



Computer Graphics And Multimedia

Topic:

Homogeneous Matrix Representation

Contributed By:

Ankita Jiyani

Homogenous Matrix Representation

while designing the application, we perform translation, rotation, scaling and other transformations to fit the picture components into their proper position.

So, we are constructing the matrices, so that these transformation sequence can be efficiently processed.

Each transformation can be expressed in the general form as:-

$$P' = M_1 \cdot P + M_2 \quad - (1).$$

where P & P' are column vectors

$M_1 \rightarrow$ 2 x 2 matrix containing multiplicative factors.

$M_2 \rightarrow$ 2 element column matrix containing translational terms.

For Translation

$M_1 \rightarrow$ identity matrix

For Rotation/Scaling.

$M_2 \rightarrow$ contains translational terms.
associated with pivot point as scaling point.

Suppose, we want to perform a series of transformation, i.e scaling, rotating, translating etc. So an efficient way to do this task is to combine the transformations so that final co-ordinate positions are obtained directly from the initial co-ordinates, & eliminating the calculation of intermediate co-ordinate values.

→ combine multiplicative & translational terms for 2D geometric transformation into a single matrix representation by expanding 2x2 matrix into 3x3 matrix.

→ To represent any 2D transformation as a matrix multiplication, we represent each cartesian co-ordinate with homogeneous coordinate triple (x_h, y_h, h).

where,

$x = \frac{x_h}{h}$, $y = \frac{y_h}{h}$

Homogeneous co-ordinate representation →
$(h.x, h.y, h)$

h → homogeneous parameter
→ non zero value.

There can be infinite no. of equivalent homogeneous representation for each coordinate point (x, y).

→ easy choice ⇒ $h = 1$.

→ each point is represented as → $(x, y, 1)$.

→ we are representing 2D-coordinates as homogeneous co-ordinates because, it's implementation for multiplication matrix is easy.

→ Homogeneous coordinates are invented because of vanishing point. Projective space can't be efficiently represented by cartesian x-y coordinate system.

Matrix Representation of Transformations

Representation of Transformation in Homogeneous Co-ordinates.

① Translation :

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$P' = T(tx, ty) \cdot P.$$

② Rotation :-

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

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

③ Scaling. :-

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

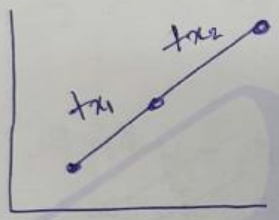
Composite Transformations

Composite Transformation :-

Two successive of any transformation, by concatenation, or composition.

(1) Translation :-

two & more than two. translation.



$$P' = T \cdot P$$

$$\begin{bmatrix} x_1' \\ y_1' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$P' = T_1 \cdot T_2 \cdot P$$

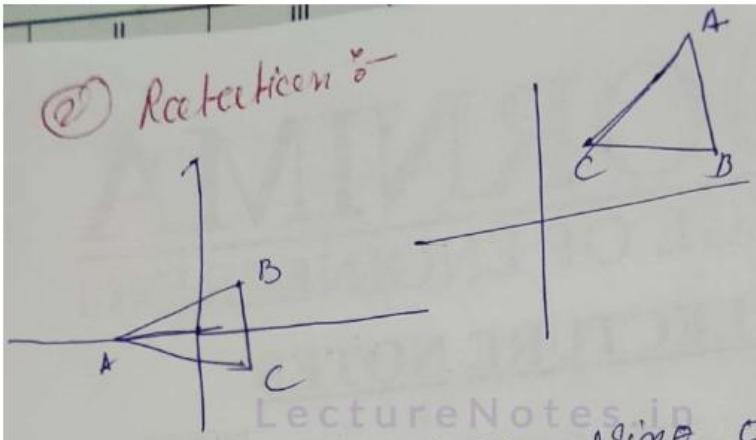
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & tx_1 \\ 0 & 1 & ty_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & tx_2 \\ 0 & 1 & ty_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & tx_1 + tx_2 \\ 0 & 1 & ty_1 + ty_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} x + tx_1 + tx_2 \\ y + ty_1 + ty_2 \\ 1 \end{bmatrix}$$

Two successive translation are additive.

(2) Rotation :-



$P' = RP$

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

$P' = R_1 R_2 P$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x\cos\theta_2 - y\sin\theta_2 \\ x\sin\theta_2 + y\cos\theta_2 \\ 1 \end{bmatrix}$$

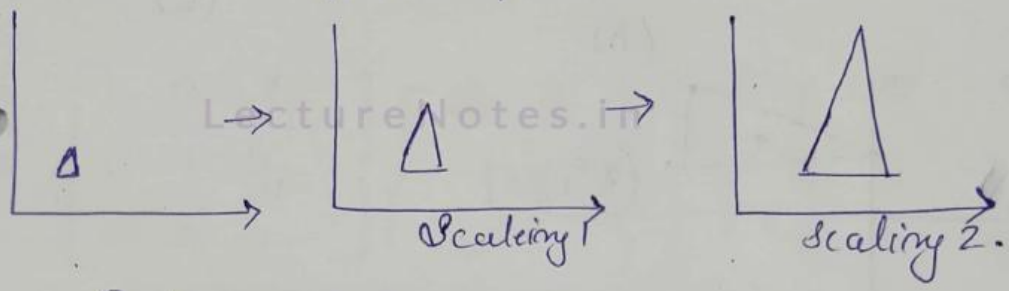
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_1 (x\cos\theta_2 - y\sin\theta_2) + (-\sin\theta_1)(x\sin\theta_2 + y\cos\theta_2) \\ \sin\theta_1 (x\cos\theta_2 - y\sin\theta_2) + \cos\theta_1 (x\sin\theta_2 + y\cos\theta_2) \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta_1 + \theta_2) & -\sin(\theta_1 + \theta_2) & \\ \sin(\theta_1 + \theta_2) & \cos(\theta_1 + \theta_2) & \\ & & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Two successive rotations are additive

Scaling :-

$P' = S \cdot P$



$$\begin{bmatrix} x_1' \\ y_1' \\ 1 \end{bmatrix} = \begin{bmatrix} S_{x_1} & 0 & 0 \\ 0 & S_{y_1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_{x_1} & 0 & 0 \\ 0 & S_{y_1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_{x_2} & 0 & 0 \\ 0 & S_{y_2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_{x_1} \cdot S_{x_2} & 0 & 0 \\ 0 & S_{y_1} \cdot S_{y_2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

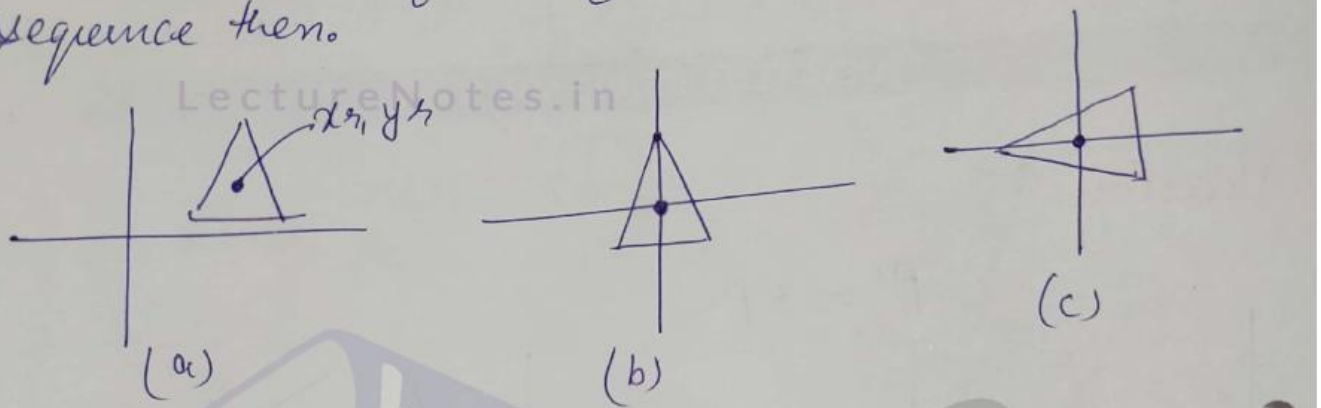
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x \cdot S_{x_1} \cdot S_{x_2} \\ y \cdot S_{y_1} \cdot S_{y_2} \\ 1 \end{bmatrix}$$

Two successive scaling are multiplicative.

Pivot-Point Rotation:-

① Translate ② Rotate ③ Translate

If above series of transformations performed in sequence then.



$$\begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & x_r(1-\cos\theta) + y_r \sin\theta \\ \sin\theta & \cos\theta & y_r(1-\cos\theta) - x_r \sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

$$T(x_r, y_r) \cdot R(\theta) \cdot T(-x_r, -y_r) = R(x_r, y_r, \theta)$$

Fixed Point Scaling

PAGE NO. .....

Fixed point scaling :-

① Translate ② Scaling. ③ Inverse Translate.

$$= \begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} s_x & 0 & x_f(1-s_x) \\ 0 & s_y & y_f(1-s_y) \\ 0 & 0 & 1 \end{bmatrix}$$

$$= T(x_f, y_f) \cdot S(s_x, s_y) \cdot T(-x_f, -y_f) =$$

$$= S(x_f, y_f, s_x, s_y).$$

Reflection in Graphics

Reflection :-

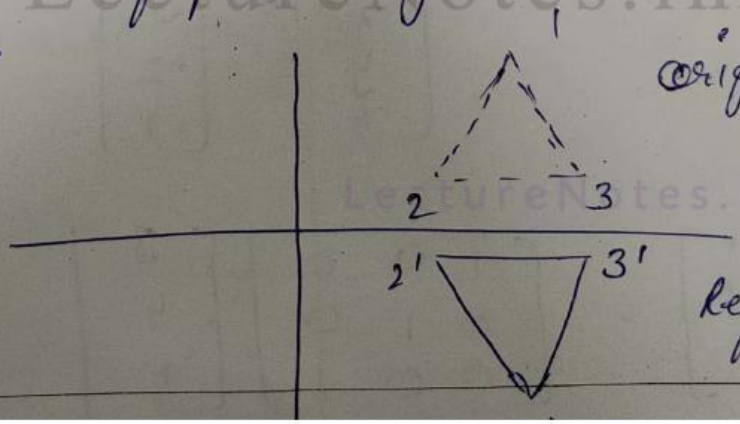
It is a transformation which produces mirror image of an object.

A mirror image is relative to ~~a~~ axis of reflection by rotating the object about reflection axis. 180°

Reflection about the line $y = 0$, the x-axis is with transformation matrix

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

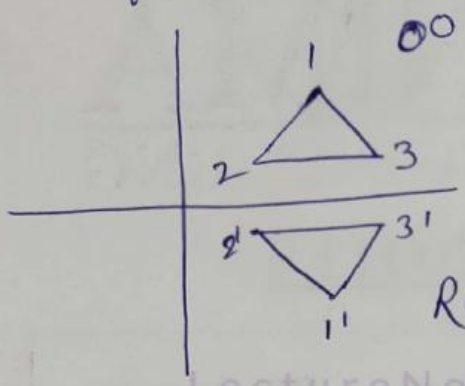
This matrix, flips the y value & keeping x as same.



Page No.:

① Reflection w.r.t x axis.

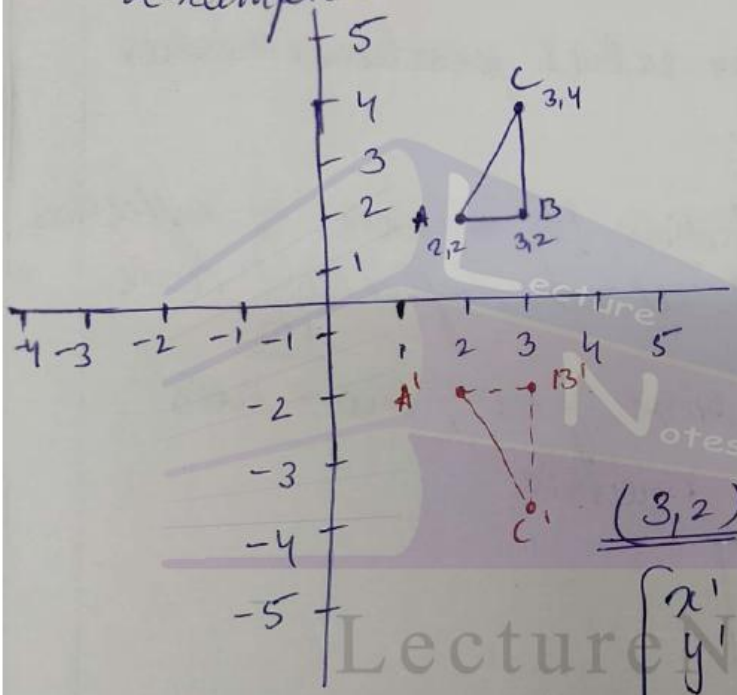
$P' = R'P.$



RO

$$\begin{bmatrix} x' \\ y' \\ h \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ h \end{bmatrix}$$

Example :-



(2,2)

$$\begin{bmatrix} x' \\ y' \\ h \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ h \end{bmatrix} = \begin{bmatrix} 2 \\ -2 \\ 1 \end{bmatrix}$$

(3,2)

$$\begin{bmatrix} x' \\ y' \\ h \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ h \end{bmatrix} = \begin{bmatrix} 3 \\ -2 \\ 1 \end{bmatrix}$$

(3,4)

$$\begin{bmatrix} x' \\ y' \\ h \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 4 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ h \end{bmatrix} = \begin{bmatrix} 3 \\ -4 \\ 1 \end{bmatrix}$$

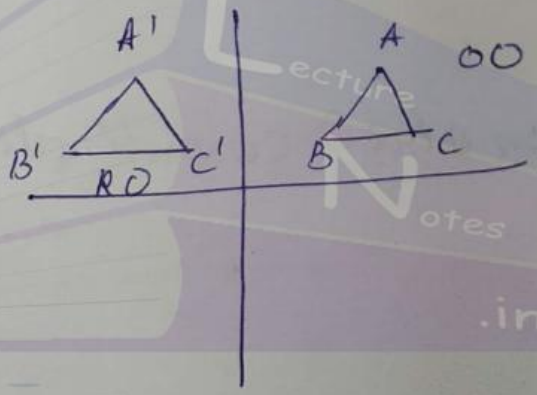
Reflection in Graphics

New - Co-ordinates :-

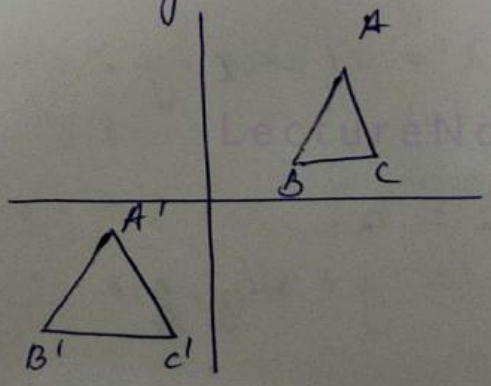
$A'(2, -2)$ $B' = (3, -2)$ $C' = (3, -4)$

Reflection w.r.t y axis :-

$$\begin{bmatrix} x' \\ y' \\ h \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ h \end{bmatrix}$$

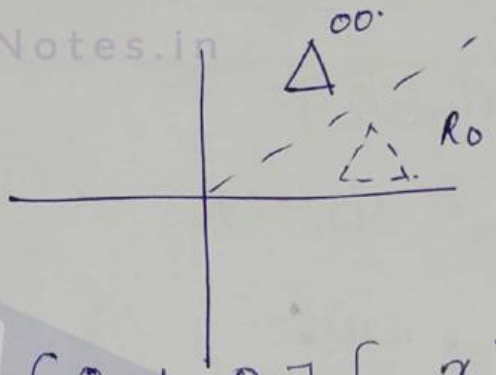


Reflection w.r.t origin :-



$$\begin{bmatrix} x' \\ y' \\ h \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ h \end{bmatrix}$$

Reflection w.r.t, when $y = x$.



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

SHEARING :-

Some distort on other coordinate :-

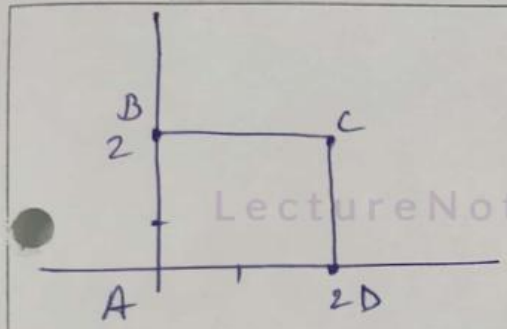
x- shears } shearing.
y- shears }

In x shears; $y' = y$
$x' = x + sh_x \cdot y$.

In y shears; $x' = x$.
$y' = y + sh_y \cdot x$

Shearing in Graphics

PAGE NO. .....

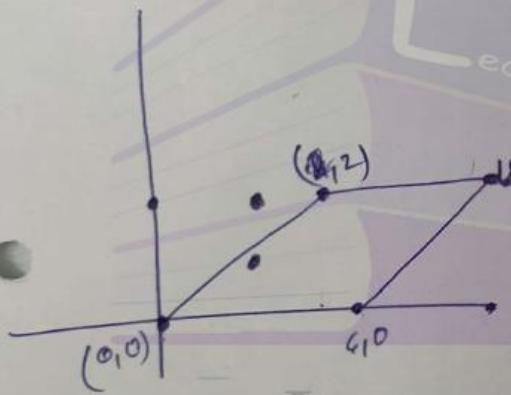


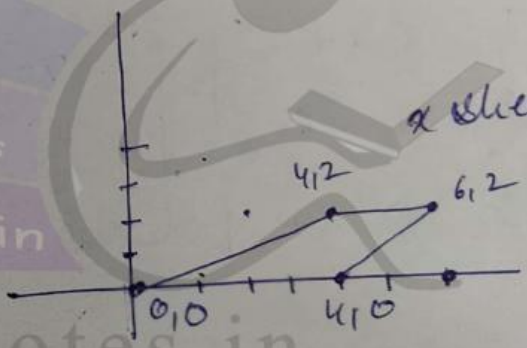
(0,0), (0,2) (2,2) (2,0)

Shx = 2 units

0,0	0,2	2,2	2,0
= 0+2.0	0+2.2	2+2.0	2+2.0
= 0	= 4	= 6	= 2

(x', y') = (0,0) (4,2) (6,0) (4,0)

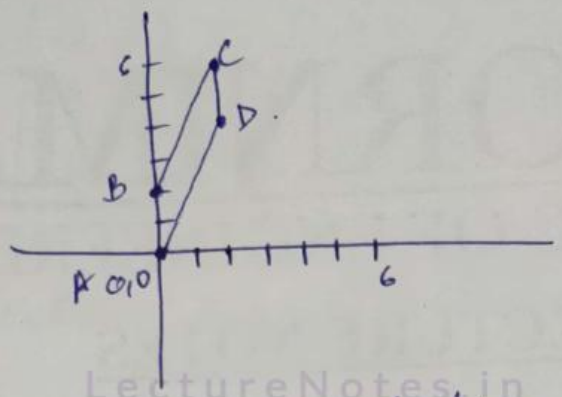




y- shear :-

Shy = 2 units.

A (0,0)	B (0,2)	C (2,2)	D (2,0)
y + Shy.x	2 + 2.0	2 + 2.2	0 + 2.2
0 + 2.0	2	6	4
= 0			
(0,0)	(0,2)	(2,6)	(2,4)



MATRIX Representation :-

x shear :-

$$\begin{bmatrix} 1 & Sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ h \end{bmatrix}$$

y shear :-

$$\begin{bmatrix} 1 & 0 & 0 \\ Sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ h \end{bmatrix}$$

Computer Graphics And Multimedia

Topic:

Polygon Filling Algorithm

Contributed By:

Ankita Jiyani

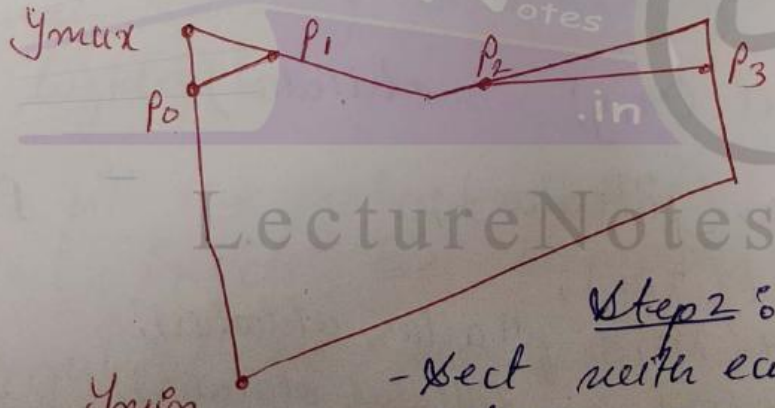
Polygon Filling Algorithms

Polygon Filling Algo:-

To fill polygon with particular colors, you need to determine the pixels falling on the border of the polygon & those which fall inside the polygon.

Scan line Algorithm:-

This algo works by intersecting scan line with polygon edges & fills the polygon b/w pairs of intersection.



Step 1- find out Ymax & Ymin from the given below;

Step 2: Scan line intersect with each edge of the polygon from Ymin to Ymax. Name each intersection point of polygon. P_0, P_1, P_2, P_3.

Page No.:

Step 3 : Sort the intersection point in the increasing order of x-coordinate.
(P_0, P_1) (P_1, P_2) (P_2, P_3).

Step 4 : Fill all those pair of coordinates that are inside the polygon & ignore the alternate pairs.

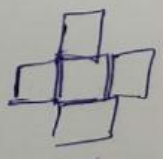
Flood Fill Algorithm :-

Disadvantages of Scan conversion, scan fill algorithms are :-

- More complex
- It is discrete algorithm

Sometimes we want to fill an color area that is not defined within a single color boundary

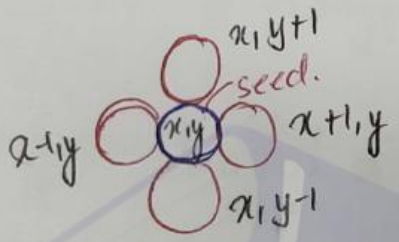

```
Procedure flood fill ( x, y, oldcolor, fillcolor)
{
    if (getpixel (x,y) == oldcolor)
    {
        set color (fillcolor);
        Set Pixel (x,y);
        floodfill 4 (x+1, y, fillcolor, oldcolor);
        floodfill4 (x-1, y, fillcolor, oldcolor);
        floodfill4 (x, y+1, fillcolor, oldcolor);
        floodfill4 (x, y-1, fillcolor, oldcolor);
    }
}
```



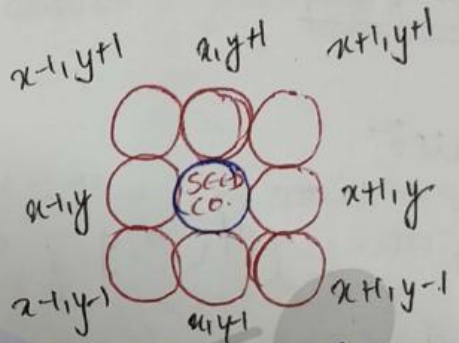
Boundary Filling Algorithm

Example :- MS paint.

- Boundary filling Algorithm.



4 way connected



8 way connected.

- ↳ First boundary drawn, and boundary pixels are highlighted.
- ↳ To fill the polygon, select any one pixel inside the boundary.
- ↳ Highlight the connected pixel, until boundary pixel is reached.
- ↳ The procedure tests neighboring pixels to determine whether they are of boundary color.

This can be done in 2 ways;
4 way connected, 8 way connected.

An 8 way connected would correctly fill the interior of the circle. A 4 way will fill it partially.

in 4 way

Procedure:-

```
boundary-fill( x,y, f-colour, b-colour).
{ if (getpixel(x,y) != bcolour && getpixel != fcolour
  { putpixel(x,y,f-colour)
    boundary-fill ( x+1,y,f-colour, b-colour)
    boundary-fill ( x, y+1, fcolour, bcolour)
    boundary-fill ( x-1, y, f-colour, bcolour)
    boundary-fill (x, y-1, f-colour, bcolour)
  }
}
```

Difference between Seed Fill and Boundary Fill

Disadvantages of Seedfill.

- If an pixel is in some other color, then the fill terminates a polygon remain unfilled
- Seed fill method doesn't work for large polygons.

Difference b/w Flood Fill & Boundary Fill.

Flood Fill	Boundary Fill.
colors the entire area in an enclosed figure through interconnected pixels using a single color.	Area gets covered with pixels of a choosen color as boundary fill's giving technique
So, Flood fill is one in which all connected pixel of a selected color get replaced by a fill color.	when the color of boundary is found. It stops
It can use unpredictable amount of memory to finish because it isn't known, how many subfills will be spawned.	Usually more complicated but it is a linear algo.
Time consuming.	less time consuming.

Computer Graphics And Multimedia

Topic:

Window To Viewport Transformation

Contributed By:

Ankita Jiyani

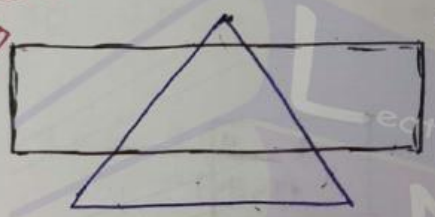
Window to Viewport Transformations

Point & Line Clipping :-

Clipping :- The objects to be displayed, which is inside the window.

First we will study : window to viewport transformation :

window

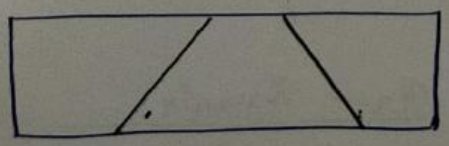


→ we want to capture this image. or a portion of the image

→ the portion or part in which object or image is displayed, c/a window.

→ WINDOW - WHAT IS TO BE DISPLAYED.

VIEWPORT - where object is displayed.



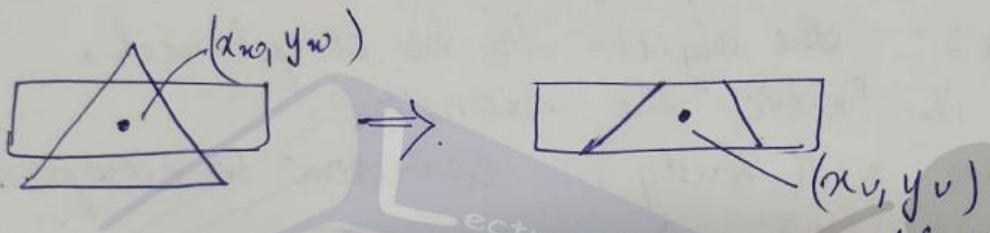
→ viewport

For a window, we have boundary values,

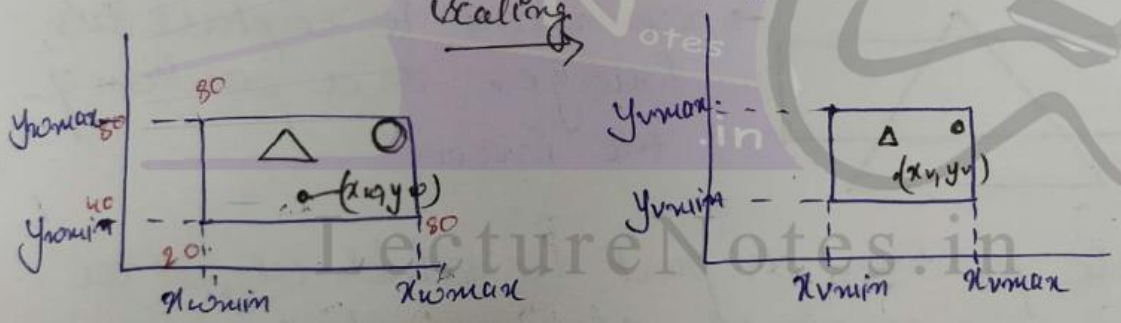
x_{wmin}, x_{wmax} → for x-axis
y_{wmin}, y_{wmax} → for y-axis
} window coordinate

For a viewport;

x_{vmin}, x_{vmax} → for x-axis
y_{vmin}, y_{vmax} → for y-axis
} device co-ordinates



window to viewport transformation;



WINDOW VIEWPORT.

The relative position is not changed.
Only size is changed.

x coordinate:
$$\frac{x_w - x_{wmin}}{x_{wmax} - x_{wmin}} = \frac{x_v - x_{vmin}}{x_{vmax} - x_{vmin}}$$

y coordinate:
$$\frac{y_w - y_{wmin}}{y_{wmax} - y_{wmin}} = \frac{y_v - y_{vmin}}{y_{vmax} - y_{vmin}}$$

Window to Viewport Transformation Example

Example:-

$x_{wmin} = 20$
$x_{wmax} = 80$
$y_{wmin} = 40$
$y_{wmax} = 80$

$(x_w, y_w) = (30, 80)$

$x_{vmin} = 30$
$x_{vmax} = 60$
$y_{vmin} = 40$
$y_{vmax} = 60$

$(x_v, y_v) = ?$

$$\frac{30-20}{80-20} = \frac{x_v - 30}{60-30} \Rightarrow \frac{10}{60} = \frac{x_v - 30}{30}$$

$x_v = 35$

$$\frac{80-40}{80-40} = \frac{y_v - 40}{60-40} \Rightarrow \frac{40}{40} = \frac{y_v - 40}{20}$$

$y_v = 20 + 40 = 60.$

$(x_v, y_v) = (35, 60)$

Point Clipping :- Discarding the object which are outside of the window.

↳ Clip a point.

↳ points lies outside the window are discarded.

↳ a point say (x,y) lies inside the window if it satisfies following property :-

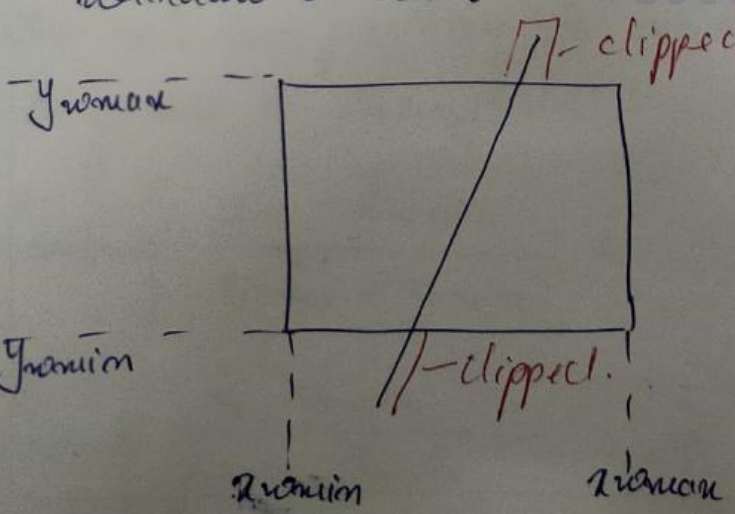
① $x_{wmin} \leq x \leq x_{wmax}$

② $y_{wmin} \leq y \leq y_{wmax}$

If it doesn't satisfy the above property, discard the point.

Line Clipping :- Discarding the part of line which is outside the window.

There are various algorithm to check whether the portion of a line exist inside the window or not.



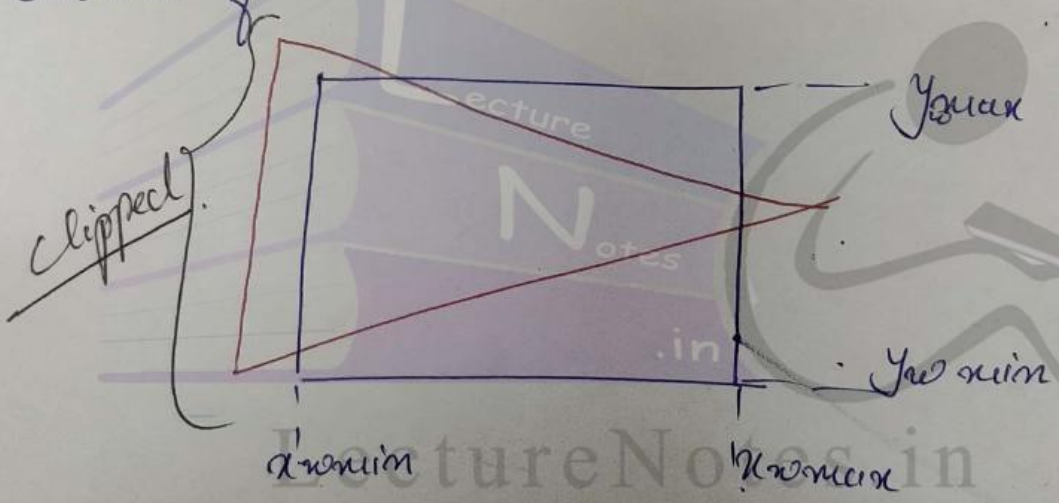
Types of Clipping Algorithms

Algorithms are :-

1. Cohen- Sutherland line clipping Algo
2. Liang- Busky line clipping Algo.

Polygon Clipping :-

It is erasing the part of the polygon, outside of the window.



Algorithms are :-

1. Sutherland- Hodgeman.

Computer Graphics And Multimedia

Topic:

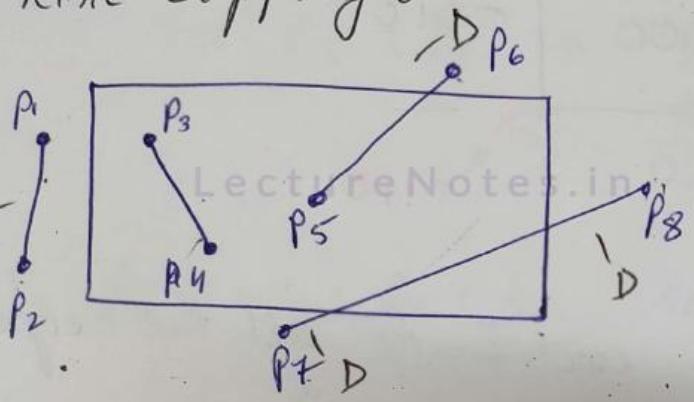
Line Clipping Algorithm

Contributed By:

Ankita Jiyani

Line Clipping Algorithm

Line Clipping :-



Discard lines outside of the window.

P_1P_2 - Reject → check endpoints for every line.
P_3P_4 - consider
P_5P_6 - clipping Reqd
P_7P_8 - clipping Reqd.

if both point lies outside Reject the line.

if both points inside accept the line.

But this method has disadvantage;

if P_7P_8 → lies outside → Reject.

P_5P_6 → one point inside & one is outside.

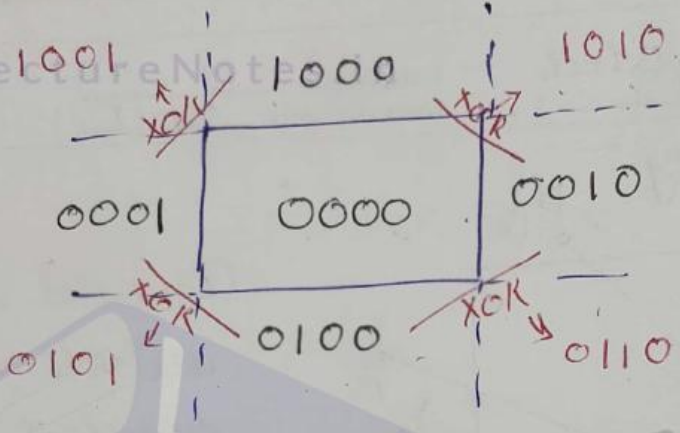
so not efficient.

Page No.:

COHEN-SUTHERLAND LINE CLIPPING ALGORITHM.

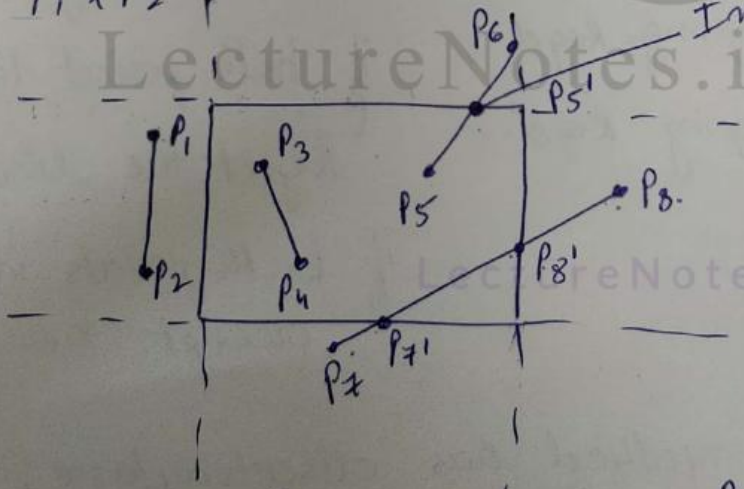
-> This algorithm is based on 4 bit region code.

Below
ABRL - Left
Above Right



→ We have given the codes to all the neighbor of the rectangle.

→ Four P_1 & P_2 :-



Code for P_1 & P_2 - 0001 — P_1
0001 — P_2

0001 & Non-zero.

If result is non-zero, then line doesn't

Cohen Sutherland Line Clipping Algorithm

lie in the region.

For P_3 & P_4. → 0 0 0 0 ✓
0 0 0 0 ✓ AND
0 0 0 0 → Zero Result.

P_3, P_4 lies in the region 0000.

For P_5 P_6 → 0 0 0 0 — zero.
1 0 0 0 AND
0 0 0 0 → clipping Requd.

Some part lies inside & some outside.

→ Find intersection point on window.

P_5' P_6 → 0 0 0 0
0 0 0 0
0 0 0 0 → All zero.

→ P_5' - P_6 → clipped.

for P_7 P_8.

$$\begin{array}{r} 0100 \\ 0010 \\ \hline 0000 \end{array} \rightarrow \text{zero Result.}$$

P_7' P_8' =

$$\begin{array}{r} 0000 \\ 0000 \\ \hline 0000. \end{array} \rightarrow \text{zero Result.}$$

P_7 P_8' =

$$\begin{array}{r} 0100 \\ 0000 \\ \hline 0000 \end{array} \rightarrow \text{zero. Result.}$$

Points.	Region code	Result.	Intersection point	Decision.
P_1 P_2	0001 ?? 0001	0001	No.	Reject.
P_3 P_4	0000 ?? 0000	0000	No	Accept.
P_5 P_6	0000 ?? 1000	0000	P_5'	P_5' P_6 clipped.
P_7 P_8	0100 ?? 0010	0000	P_7'	P_7 P_7' clipped.
P_7' P_8	0000 ?? 0010	0000	P_8'	P_8' P_8 clipped.

Pseudo code for Cohen Sutherland Algorithm

Pseudo code :-

Step 1 - Assign a region code for 2 endpoints.

Step 2 - If both endpoints have region code, 0000 then given line is completly inside

Step 3 - else perform logical AND operation for both endpoints.

Step 3.1 : if result is non zero, then line is completly outside.

Step 3.2 : else line is particlly inside.

3.2.1 : choose end point that is outside the given rectangle.

3.2.2 : find intersection point of the rectangular boundary.

3.2.3 : Replace end point with intersection point & update the region code.

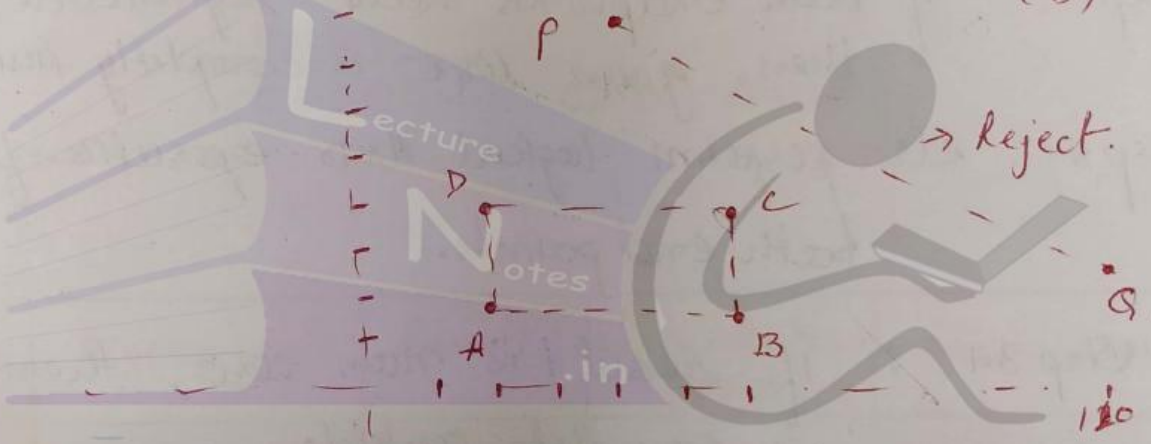
3.2.4 : Repeat step 2 untill we find a clipped line.

Step 4 - Repeat step 1 for other lines.

Questions :-

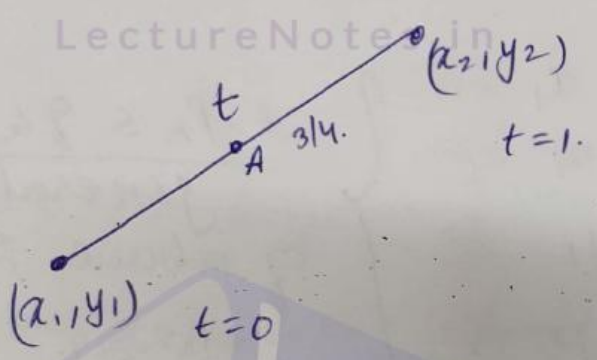
(1) use cohen-sutherland line clipping algo to find the visible portion of the line P(40,80), Q(120,30) inside the rectangle. the rectangle is defined as A(20,20) B(60,20), C(60,40) & D(20,40). RTU-2013. (8)

Sol



Liang Basky Line Clipping Algorithm

LIANG-BASKY LINE CLIPPING ALGORITHM.



line A travel from (x_1, y_1) to (x_2, y_2) in time interval $t=0$ to $t=1$.

$$x = t \cdot x_2 + (1-t)x_1 \qquad 1/4\, x_1 + \frac{3}{4} x_2$$

$$= x_1 + t(x_2 - x_1)$$

$$\boxed{x = x_1 + t(\Delta x)}$$

find x coordinate at particular time interval t. , $0 < t < 1$

$$\boxed{y = y_1 + t * \Delta y.} \qquad 0 < t < 1$$

The intermidiate point is the intersection point on the window.

The liang Barsky algorithm also considers following properties :-

$$x_{wmin} \le x_1 + t\Delta x \le x_{wmax}$$

$$y_{wmin} \le y_1 + t\Delta y \le y_{wmax}$$

$$t\Delta x \ge x_{wmin} - x_1 \quad (1)$$
$$t\Delta x \le x_{wmax} - x_1 \quad (2)$$
$$t\Delta y \ge y_{wmin} - y_1 \quad (3)$$
$$t\Delta y \le y_{wmax} - y_1 \quad (4)$$

$t \cdot p_k \le q_k$ general form of above inequalities $k = 1, 2, 3, 4$.

eq 1 & eq 3 should be multiplied with (-1)

$$-t\Delta x \le x_1 - x_{wmin}$$
$$t\Delta x \le x_{wmax} - x_1$$
$$-t\Delta y \le y_1 - y_{wmin}$$
$$t\Delta y \le y_{wmax} - y_1$$

All inequalities are converted into lesser than equal to.

$p_1 = -\Delta x$ $\quad p_2 = \Delta x$

$p_3 = -\Delta y$ $\quad p_4 = \Delta y$.

$q_1 = x_1 - x_{wmin}$

$q_2 = x_{wmax} - x_1$

$q_3 = y_1 - y_{wmin}$

$q_4 = y_{wmax} - y_1$

If $p_k = 0$ → line is parallel to window

Corresponding $q_k < 0$ – line is outside

$q_k > 0$ – line is inside. or, partially inside.

$q_k = 0$ – within the boundary. or partially inside.

If $p_k < 0$ → find (x, y) at t.

$t_1 = \max\left(0, \frac{q_k}{p_k}\right)$ // Intersection point

$x = x_1 + t_1 \cdot \Delta x$

$y = y_1 + t_1 \cdot \Delta y$.

If $P_k > 0$, time interval at t_2 //

$$t_2 = \min\left(1, \frac{q_K}{P_k}\right)$$ // intersection point.

$$x = x_1 + t_2 \cdot \Delta x$$
$$y = y_1 + t_2 \cdot \Delta y$$

If $t_1 = 0$ → doesn't change
no change in initial point.
& initial point is inside the window

If $t_1 \neq 0$ → changed.
Change in initial point.
& final or end point doesn't lie in the window.

If $t_2 = 1$, second point inside the window.
If $t_2 \neq 1$. if not outside the window.

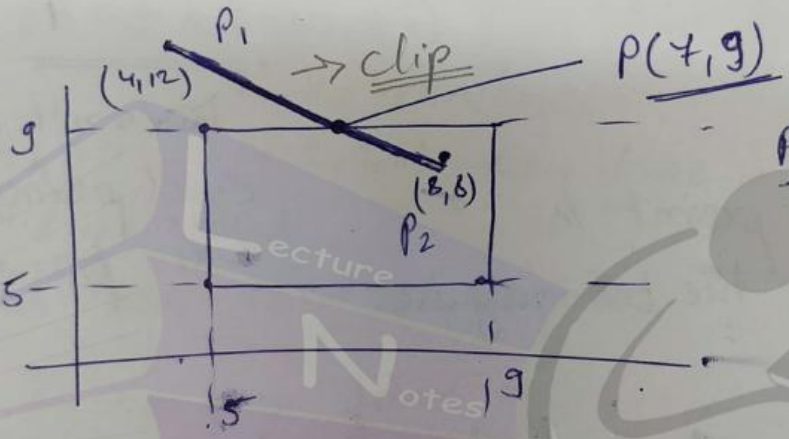
Liang Basky Example

Example:-

$x_{wmin} = 5$ $y_{wmin} = 5$

$x_{wmax} = 9$ $y_{wmax} = 9$.

Line points $P_1(4, 12)$ $P_2(8, 8)$



$P(7,9)$ & $P_2(8,8)$ Inside the boundary.

$P_1 = -4$ $q_1 = 4 - 5 = -1$

$P_2 = 4$ $q_2 = 9 - 4 = 5$

$P_3 = -(-4) = 4$ $q_3 = 12 - 5 = 7$

$P_4 = -4$ $q_4 = 9 - 12 = -3$

For P_1 & P_4.

$P_k < 0$; $t_1 = \max\left(0, \frac{q_1}{P_1}, \frac{q_4}{P_4}\right)$

p_2 & $p_3 > 0$.

$$t_2 = \min\left(1, \frac{Q_2}{P_2}, \frac{Q_3}{P_3}\right)$$

p_1 & p_4

$$t_1 = \max\left(0, \frac{-1}{-4}, \frac{-3}{-4}\right)$$

$$t_1 = \max\left(0, \frac{1}{4}, \frac{3}{4}\right)$$

$$t_1 = \frac{3}{4}$$

$t_1 \neq 0$.

Starting point is not inside the boundary.

p_2 & p_3

$$t_2 = \min\left(1, \frac{5}{4}, \frac{7}{4}\right)$$

$$t_2 = 1$$

Doesn't change, end point inside the boundary.

$$x = x_1 + t_1 \cdot \Delta x$$

$$4 + \frac{3}{4} \times 4$$

$$= 7.$$

$$y = y_1 + t \cdot \Delta y$$

$$12 + \frac{3}{4} \times -4 = 12 - 3$$

$$= 9.$$

$$(x, y) = (7, 9)$$

Values at $t_1 = \frac{3}{4}$, $(x, y) = (7, 9)$

Pseudo Code for Liang Basky

Pseudo code :-

Step1 :- Get the line endpoints (x_1, y_1) to (x_2, y_2)

Step2 : Find Δx, Δy, $P_1 - P_4$, $Q_1 - Q_4$

Step3 Assign $t_1 = 0$, & $t_2 = 1$

i) if $P_k = 0$ $(k = 1, 2, 3, 4)$ then line is parallel to the window.

ii) if $q_k < 0$ $(k = 1, 2, 3, 4)$ then line is outside the window.

iii) For non-zero value of P_k.

if $P_k < 0$ then find t_1

$$t_1 = \max(0, q_k / P_k)$$

else $P_k > 0$ then find t_2

$$t_2 = \min(1, q_k / P_k)$$

if $t_1 > t_2$ then line is completely outside - Reject.

or.

else find new set of (x, y) if t_1, t_2 is changed.

$x = x_1 + t \cdot \Delta x$

$y = y_1 + t \cdot \Delta x$

Solution :- $P_1(-1,-2)$ $P_2(2,4)$

$2-(-1)$
$2+1=3$
$4-(-2)=4+2=6$

$x_{wmin} = 0$ $y_{wmin} = 0$

$x_{wmax} = 1$ $y_{wmax} = 1$

$P_1 = -3$ $q_1 = -1-0 = -1$ $q_1/p_1 = 1/3$

$P_2 = 3$ $q_2 = 1-(-1) = 2$ $q_2/p_2 = 2/3$

$P_3 = -6$ $q_3 = -2-0 = -2$ $q_3/p_3 = 1/3$

$P_4 = 6$ $q_4 = 1-(-2) = 3$ $q_4/p_4 = 1/2$

$P_1 \& P_3 < 0$. $P_2, P_4 > 0$

$t_1 = \max(0, \frac{1}{3}, \frac{1}{3})$ $t_2 = \min(1, \frac{2}{3}, \frac{1}{2})$

$t_1 = 1/3$ $t_2 = 1/2$

$t_1 \neq 0$ $t_2 \neq 0$.

$x = -1 + \frac{1}{3} \times 3 = 0$ $x = -1 + \frac{1}{2} \times 3 = \frac{1}{2}$

$y = -2 + 6 \cdot \frac{1}{3} = 0$ $y = -2 + \frac{1}{2} \times 6 = 1$

$(x,y) = (0,0)$ $(x,y) = (1/2, 1)$

Point inside the region is
$P_1(0,0)$ $P_2(1/2, 1)$

Computer Graphics And Multimedia

Topic:

Ploygon Clipping Algorithm

Contributed By:

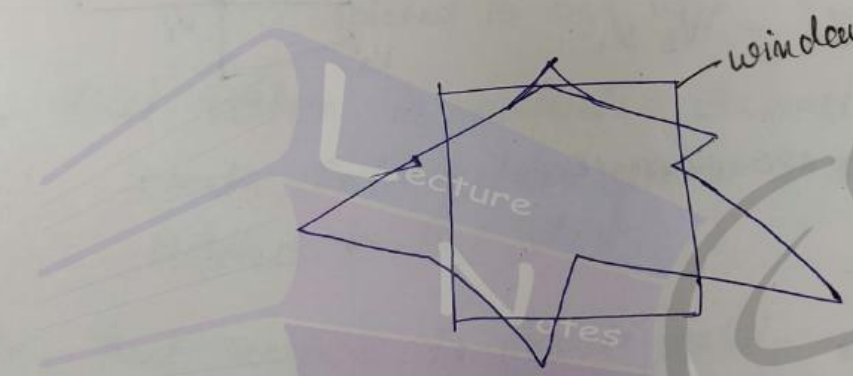
Ankita Jiyani

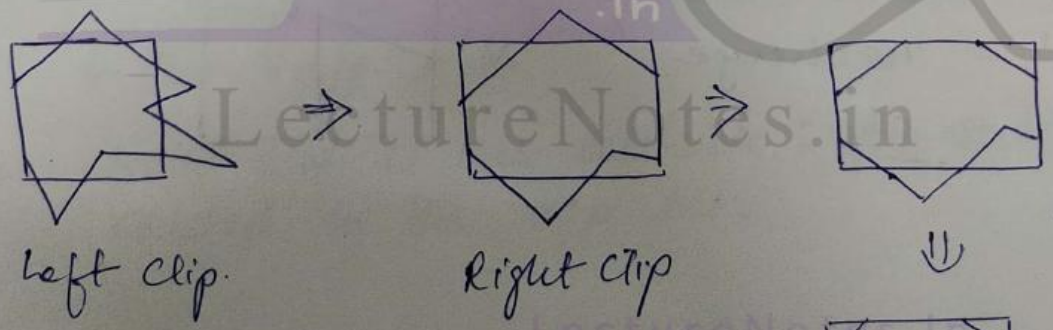
Sutherland Hodgeman Polygon Clipping Algorithm

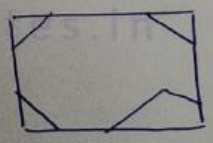
SUTHERLAND - HODGEMAN POLYGON CLIPPING ALGORITHM.

Four procedures;

- left clip.
- Right clip.
- Top clip.
- Bottom clip.

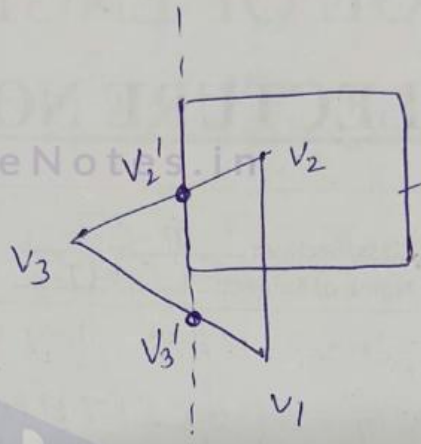






Page No.:

one output is input to the other clip function.

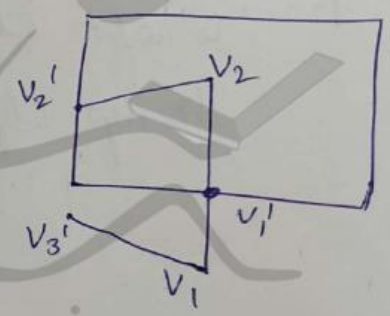


Left clip

V_1V_2 — In - In — V_2

V_2V_3 — In - out — V_2'

V_3V_1 — out - In - $V_3'V_1$



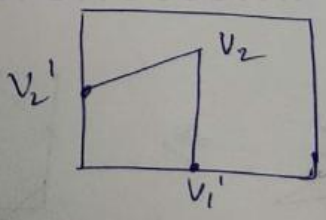
Right-clip. → no change.

Bottom-clip.

V_1V_2 — ~~out~~ - In — V_1V_1'

V_2V_2' — no change

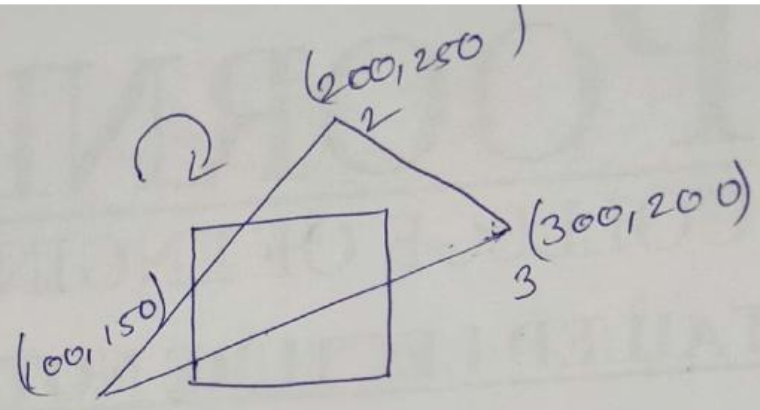
$V_3'V_1$ → clip



Clip polygon using Sutherland-Hodgeman Algo,
Input is in the form of vertices of the polygon in clockwise order.

There are 4 possible cases for any given edge of given polygon against different clipping edge e.

1. Both vertices are inside : only second vertex added to the o/p list.
2. First vertex is outside & second is inside :- both intersection point & second vertex is added to the list.
3. First vertex is inside while second is outside; only the point of intersection is added to the o/p list.
4. Both vertices are outside : no vertex are added to the o/p list.



vertex list (100, 150) (200, 250) (300, 200)

a polygon start from (x_1, y_1) & ends at (x_2, y_2). This for clipper polygon.

$$P = (x_2 - x_1)(y_0 - y_1) - (y_2 - y_1)(x_0 - x_1)$$

if $P < 0$, point is right side of the line.

$P = 0$, point is on the line

$P > 0$, point is left side of the line.

Computer Graphics And Multimedia

Topic:

Hidden Surface Problem

Contributed By:

Ankita Jiyani

Hidden surface problem

Introduction:

When we view a picture containing non-transparent objects & surfaces, then we can't see those objects from view which are behind from objects closer to eye. We must remove these hidden surfaces to get a realistic screen image. The identification & removal of these surfaces is called Hidden surface problem.

The HSR is the process to determine which surfaces and parts of the surfaces are not visible from certain viewpoint.

There are 2 approaches for removing hidden surface problems:

1. object space method.

This method is implemented in physical coordinate system.

An object-space method compares objects & part of objects to each other within the scene definition to determine which surfaces as a whole we should label as visible.

↳ Image space method.

It is implemented in screen coordinate system.

In this method, visibility is decided point-by-point at each pixel position on the projection plane.

Most of the algorithms uses, image space method for detecting visible surface.

Algo. uses image space method, is much more efficient than object space method.

But object space methods are much more functional than image space method.

Image & Object Space Method

PAGE NO. .....

Image space method :-

The representation of graphics in form of pixels is very flexible in raster systems keeps refreshing the screen by taking the values stored in frame buffer. Image space algorithms are very simple & efficient as their data structure is very similar to that of frame buffer. Commonly used algo is Z-Buffer Algo.

Object space method :-

Space object algo has advantage of retaining the relevant data & because of this ability the interaction of algo with the object becomes easier. The calculation done for the colors is done only once. This algo also allows shadow generation to increase the depth of the 3D object on the screen. Easy to implement as s/w rather than h/w

Difference between Image and Object Space Method

Object space model	Image space Model.
This model defines the object with real world dimensions & volume.	This model defines the object with dimensions of image, created to be displayed on the display device.
The dimensions may be in km, m or even in microns	It makes use of scaled dimensions.
less efficient than image space method.	More efficient than object space method.
More functional than image space method.	less functional than object space method.
color calculation is done only one time & is retained by it	color calculation once done & is overwritten later.

Object Space Method Pseudo code

object space method :-

For each object in the scene do;

Begin;

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

End.

→ Compare each object with all other objects to determine the visibility of object parts.

→ If n objects in scene, $O(n^2)$ - complexity.

→ Calculations performed at the resolution in which objects are defined.

→ Computationally expensive than image space because step 1 is more complex.

→ Suitable for scene with smaller no. of objects.

Image Space Method Pseudo code

Image space method:-

Begin:

1. Determine the object closest to the viewer that is pierced by the projector through the pixel
2. Draw the pixel in the object colour.

End.

→ For each pixel, all n objects to determine the one closest to the viewer.

→ If there are p pixels in an image, then complexity is O(np)

→ Accuracy is bounded by the display resolution.

→ A change in resolution requires re-calculation.

Computer Graphics And Multimedia

Topic:

Visible Surface Detection Algorithm

Contributed By:

Ankita Jiyani

Visible Surface Detection Algorithm

Depth Buffer or Z Buffer Method

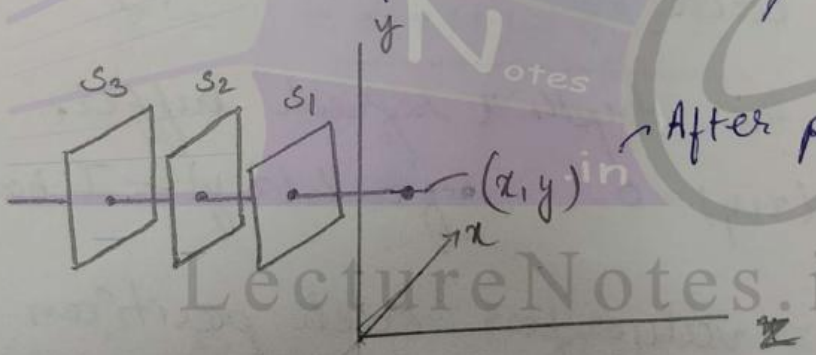
Visible surface detection Algorithm;

Depth Buffer Method or Z Buffer Method

→ This is based on Image Space method.

→ It will compare surface depth at each pixel position on projection plane.

→ Calculate the intensity value of the surface which is nearer to viewplane.



S_1 surface is nearer to the x-y plane.

Z Buffer Algorithm

Algorithm :-

We are using 2 buffers,
- one buffer for storing z-value.
- one buffer for refreshing, I storing intensity value of x, y coordinate for which we have calculated z value.
- we are taking normalized value for z-coordinate.

$0 < z < 1$ — front clipping plane / Back clipping plane

Algo pseudo code :-

1. Initialize the depth & refresh buffer.
   depth(x, y) = 0. refresh(x, y) = I background
2. calculate z value for each position in the surface & then
   if $z > depth(x, y)$
   set, depth(x, y) = z., &
   refresh(x, y) = I surf(x, y)

3. After processing all the surfaces, we will get visible surface in depth (x,y). and intensity values in refresh (x,y).

z buffer or z value is calculated as;

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

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

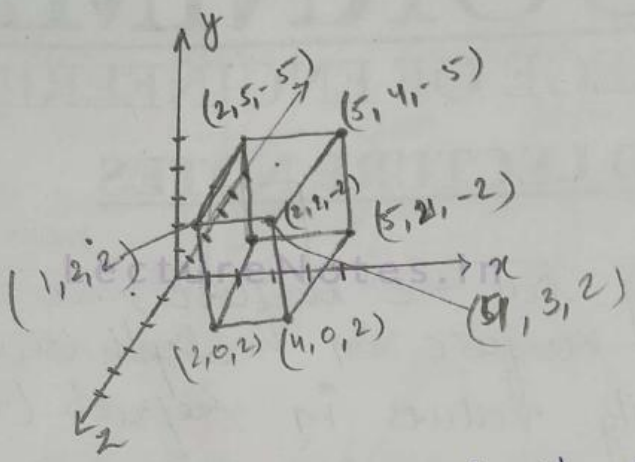
$$z' = \frac{-A(x+1) - By - D}{C}$$

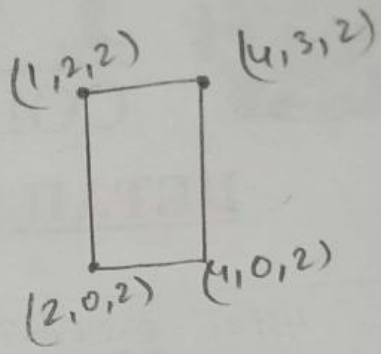
$$z' = z - \frac{A}{C}$$

→ It can't be use for transparent objects.

Z Buffer Algorithm Example

Example :-





depth (x, y) = 0.

Initialize depth(x,y) = 0.

0	0	0
0	0	0
0	0	0
0	0	0
0	0	0

z = 2.

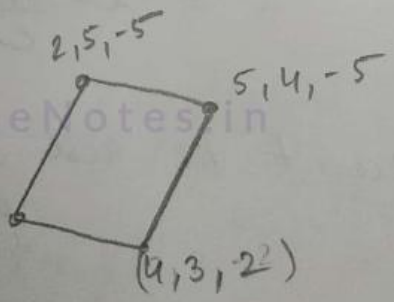
z > 0.

update.

depth (x, y) = 2.

2	2	2
2	2	2
2	2	2
2	2	2
2	2	2

depth(x, y) = 2.



-5	-5	-5
-4	-4	-4
-3	-3	-3
-2	-2	-1
-1	-1	-1
0	0	0
1	1	1
2	2	2

depth buffer (x, y)

Computer Graphics And Multimedia

Topic:

Back Face Detection

Contributed By:

Ankita Jiyani

Back Face Detection Method

BACK FACE DETECTION

It is method to detect visible surfaces, It is based on inside-outside test which is defined as :-

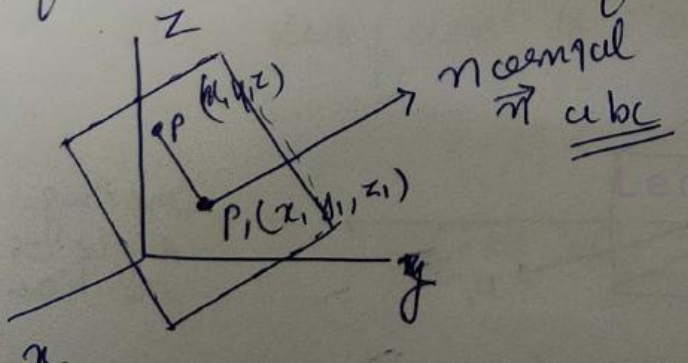
$$Ax + By + Cz + D < 0. \quad -(1)$$

The equation of normal plane is :-

$$Ax + By + Cz + D = 0. \quad -(2).$$

A point $A(x, y, z)$ is inside the polygon if it satisfies equation 1.

If it doesn't satisfies, the point is outs



$\vec{p} - \vec{p_1} = (x-x_1, y-y_1, z-z_1)$

since $\vec{p}-\vec{p_1}$ & $\vec{n}$ are perpendicular to each other.

Dot product is 0.

$\vec{n} \cdot (\vec{p} - \vec{p_1}) = 0.$

This dot product becomes the eq. of plane.

$(a, b, c) \cdot (x-x_1, y-y_1, z-z_1) = 0$

$a(x-x_1) + b(y-y_1) + c(z-z_1) = 0$

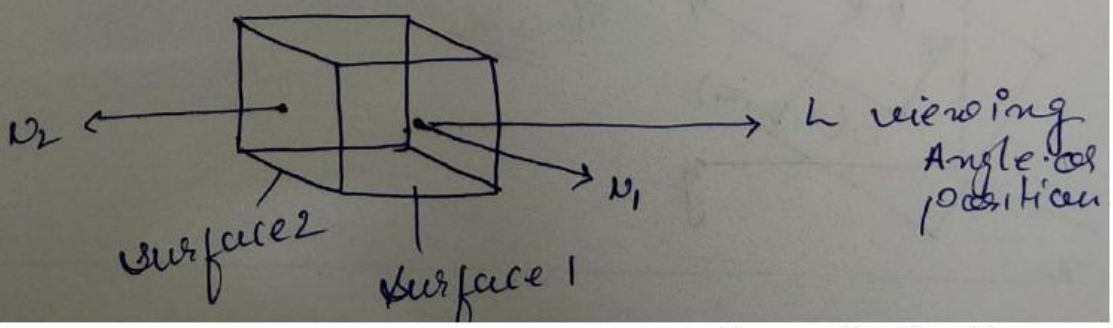
$ax + by + cz - (ax_1 + by_1 + cz_1) = 0$

$ax + by + cz + D = 0.$

$D = -(ax_1 + by_1 + cz_1)$ ... (1).

BACK FACE PROCEDURE :-

Consider the 2 faces (surface) of a cube & their normal vectors



Now Depending on the angel between
L and N_1 , L and N_2
The surface S_1 & S_2 may or may not
be visible to the viewer.

→ The surface is visible from the position L,
iff.

$$0 \leq \theta \leq 90^\circ$$

or

$$0 \leq \cos\theta \leq 1$$

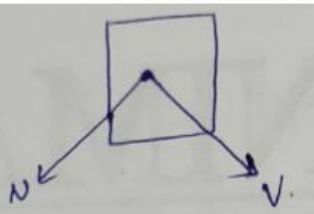
→ Calculate $\cos\theta$ using;

$$L \cdot N = |L||N| \cos\theta.$$

→ The surface is visible; iff,

$L \cdot N < 0$. Front face.

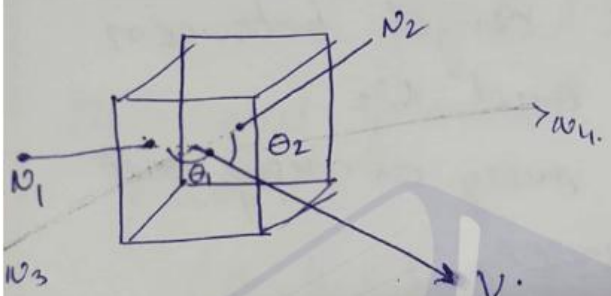
$L \cdot N > 0$ Back face.



$0 \le \cos\theta \le \pi/2$

$\cos\theta > 0$.

It is Back face of the object.



$V \cdot N_2 = |V|\,|N_2| \cos\theta_2$

$V \cdot N_1 = |V|\,|N_1| \cos\theta_1$

So, $\cos\theta_1$ & $\cos\theta_2$ are > 0.

So, these are Back faces

$V \cdot N_3 = |V|\,|N_3| \cos\theta_3$

$V \cdot N_4 = |V|\,|N_4| \cos\theta_4$.

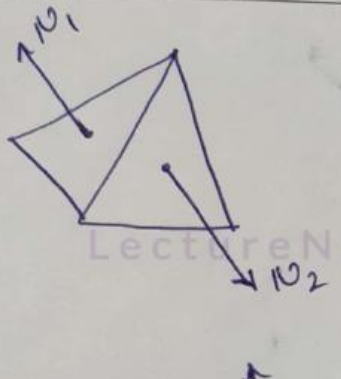
So, here $\cos\theta_3$ & $\cos\theta_4 < 0$.

So, these are the front faces.

If. $\pi/2 \le \cos\theta < \pi$

$\cos\theta < 0$

It will be considered as front face of the object.



N1 & V are both in same direction.

So, here

$V \cdot N_1 > 0$.

Back face surface.

N2 & V are in opposite direction

$V \cdot N_2 < 0$

Front face surface.

Now we are projecting the object into 2D are our viewing is along the z axis.

So, view coordinate will be,

$V = (0, 0, V_z)$ $N(A, B, C)$

V.N will become:-

$V \cdot N = A \cdot 0 + B \cdot 0 + V_z \cdot C$.

$V \cdot N = V_z \cdot C$ (It depends on, sign of C).

Computer Graphics And Multimedia

Topic:

Bezier Curve And It's Properties

Contributed By:

Ankita Jiyani

Bezier Curves & it's properties

Bezier Curves :-

Properties :-

1. They follow the shape of the control polygon which consist of the segments joining the control points.

2. They always pass through first & last control points.

3. They are contained in the convex hull of their defining control points.

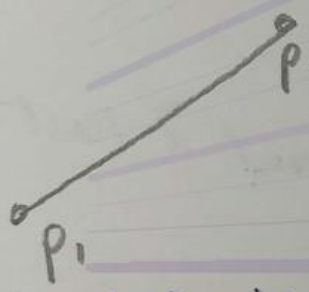
4. The degree of the polynomial defining the curve segment is one less than the no. of defining polygon point. Therefore for 4 control points, the degree of the polynomial is 3. i.e cubic polynomial.

Page No.:

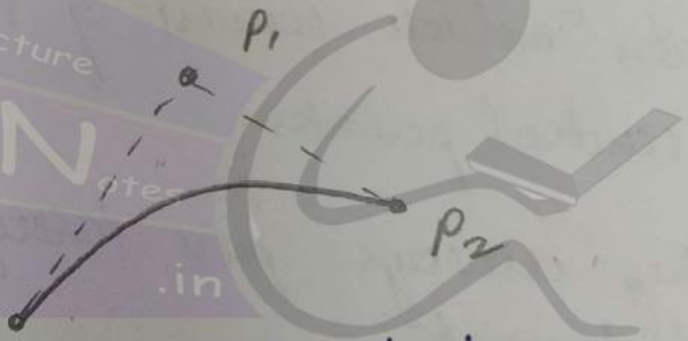
5. A Bezier curve generally follows the shape of defining polygon.

6. Bezier curves exhibit global control, means moving a control point alters the shape of a curve.

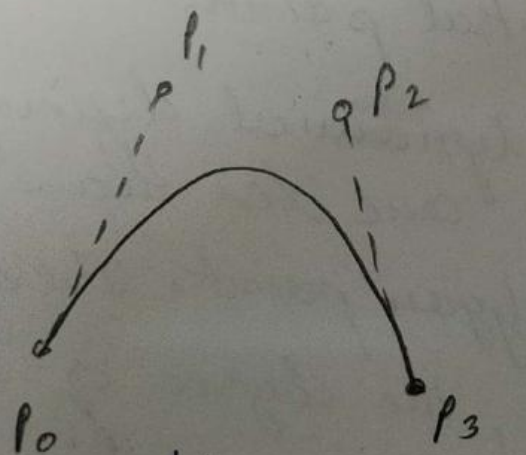
7. These curves are generated under the control of other points.



Simple Bezier curve



Quadratic Bezier curve.

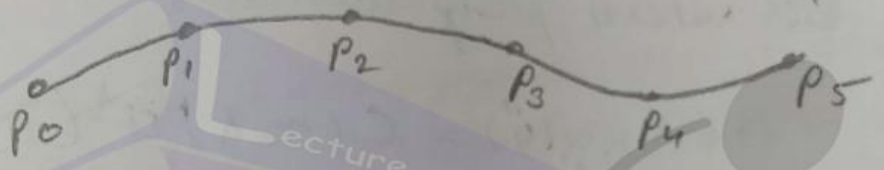


Cubic Bezier Curve

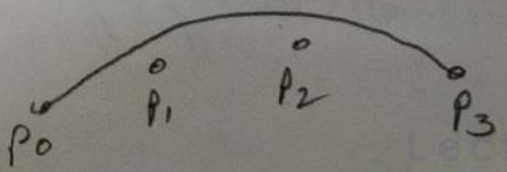
Interpolation and Approximation Curve

There are two types of curves;

1) Interpolation curve; All points of the basic figure are located on the created figure c/a interpolation curve segment.



2). Approximation curve; All points of the basic figure need not be located on the created figure c/a approximation curve segment.



Bezier curve is the approximation curve.

Blending Function for Bezier Curve

Functions of Bezier curve,
Suppose we have $n+1$ control points.
position : $P_k = (x_k, y_k, z_k)$
with $k = 0$ to n

These co-ordinates can be blended to produce the vector $P(u)$ which describes the path of an approximation Bezier polynomial function between P_0 & P_n.

$$P(u) = \sum_{k=0}^{n} P_k \, BEZ_{k,n}(u), \quad 0 \le u \le 1$$

The $BEZ_{k,n}(u)$ is bezier Blending function, and Bernstein polynomials :-

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

where, $C(n,k)$ are the binomial coefficients.

$$C(n,k) = \frac{n!}{k!(n-k)!}$$

we can define Bezier Blending function with recursive calculation :-

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

$$n > k \geq 1.$$

Example for Bezier Curve

with $BEZ_{k,k} = u^k$ & $BEZ_{0,k} = (1-u)^k$,
the set of parametric equation,

$$x(u) = \sum_{k=0}^{n} x_k \; BEZ_{k,n}(u)$$

$$y(u) = \sum_{k=0}^{n} y_k \; BEZ_{k,n}(u)$$

$$z(u) = \sum_{k=0}^{n} z_k \; BEZ_{k,n}(u).$$

Ques Determine eleven points on a bezier curve with equidistant parametric values control points $(x_0, y_0) = (50, 180)$, $(x_1, y_1) = (250, 100)$ $(x_2, y_2) = (600, 300)$ & $(x_3, y_3) = (500, 50)$ distributed over a screen of resolution 640×350.

Sol, Standard Bezier curve is given by;

$$P(u) = [x(u), y(u)]$$

$$= \sum_{k=0}^{n} \frac{n!}{(n-k)!\,k!} \cdot u^k (1-u)^{n-k} \cdot p^k$$

control points – 4.
Degree = 4-1 = 3 = n.

$$x(u) = \sum_{k=0}^{3} \frac{3!}{(3-k)!\,k!} \cdot u^k (1-u)^{3-k} \cdot x_k$$

$$= (1)\,u^0 (1-u)^3 x_0 + 3u(1-u)^2 \cdot x_1 + 3u^2(1-u)\,x_2 + 1 \cdot u^3 x_3$$

$$x(u) = 50(1-u)^3 + 750(1-u)^2 + 1800u^2(1-u) + 500u^3$$

$$y(u) = \sum_{k=0}^{3} \frac{3!}{(3-k)!\,k!} u^k (1-u)^{3-k} \cdot y_k$$

$$= 1u^0 (1-u)^3 y_0 + 3u(1-u)^2 \cdot y_1 + 3u^2(1-u)\,y_2 + 1 \cdot u^3 \cdot y_3$$

$$= 180(1-u)^3 + 300u(1-u)^2 + 900u^2(1-u) + 50u^3$$

For eleven point, u must be varied from 0 to 1 stepping.

Point	u	x[u], y[u]
1	0	(50, 180)
2	0.1	(113.9, 163.67)
3	0.2	(183.2, 159.76)
4	0.3	(254.3, 163.89)
5	0.4	(323.6, 171.68)
6	0.5	(387.5, 178.75)
7	0.6	(442.4, 180.72)
8	0.7	(484.7, 173.2)
9	0.8	
10	0.9	

Construct a Bezier curves for the points.

$P_0(4,2)$ $P_1(8,8)$ $P_2(16,4)$.

Soln $\sum_{k=0}^{2} P_k B_{k,n}(u)$ $\quad 0 \le u \le 1$.

$$P_k \frac{n!}{(n-k)!\,k!} u^k (1-u)^{n-k}$$

$$P_0 B_{0,2}(u) + P_1 B_{1,2}(u) + P_2 B_{2,2}(u)$$

$$\rightarrow x(u) = x_0 B_{0,2}(u) + x_1 B_{1,2}(u) + x_2 B_{2,2}(u)$$

$$\rightarrow y(u) = y_0 B_{0,2}(u) + y_1 B_{1,2}(u) + y_2 B_{2,2}(u)$$

$$B_{0,2}(u) = \frac{n!}{k!(n-k)!} u^k (1-u)^{n-k} = \frac{2!}{2!\,0!} = 1 \cdot u^0 (1-u)^0$$

$$1 \cdot (1-u)^2 = (1-u)^2$$

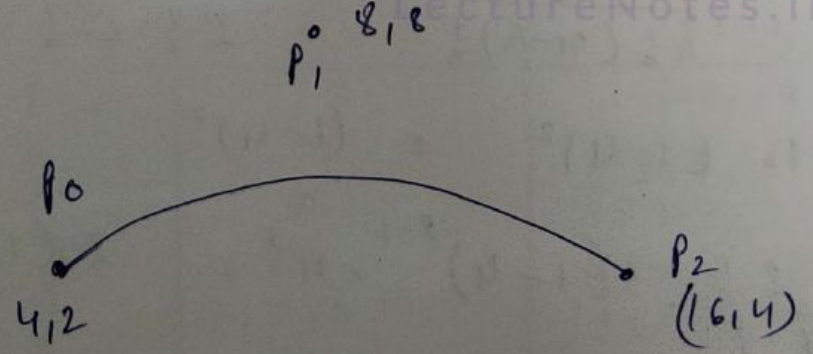
$$B_{1,2}(u) = \frac{2!}{1!\,1!} (1-u)^{2-1} \cdot u^1 = 2u(1-u)$$

$$B_{2,2}(u) = \frac{2!}{2!\,0!} u^2 (1-u)^0 = u^2$$

$$x(u) = 4(1-u)^2 + 8(1-u)2u + 16u^2$$
$$= 4 + 4u^2 - 8u + 16u - 16u^2 + 16u^2$$
$$= 4 + 4u^2 + 8u.$$

$$y(u) = 2(1-u)^2 + 8(1-u)2u + 4u^2$$
$$= 2 + 2u^2 - 4u + 16u - 12u^2$$
$$= -10u^2 + 12u + 2.$$

u	$x(u)$	$y(u)$
0	4	2
0.2	5.76	4
0.4	7.84	5.20
0.6	10.24	5.6
0.8	12.96	5.2
1.	16	4.



Example for Bezier Curve

Ques A cubic Bezier curve segment is described by control points $P_0(2,2)$ $P_1(4,8)$ $P_2(8,8)$ $P_3(9,5)$. Another curve segment is described by $Q_0(a,b)$ $Q_1(c,2)$ $Q_2(15,2)$ & $Q_3(18,2)$. Determine the value of a, b, c & d so that curve segment join smoothly.

Sol. For C^0 continuity at the point of contact $P_3(9,5)$ & $Q_0(a,b)$ must be same.

$$P_3(9,5) = Q_0(a,b)$$
$$a = 9, \quad b = 5$$

As the curve join smoothly at the point of coincident C_1 continuity at this point should be same.

Thus. $P_3' = Q_0'$ that the tangent vectors at the end point of the first curve should be identical to the tangent vectors at the starting point of the second curve.

$P_3 - P_2 = Q_1 - Q_0$

$\{(9,5) - (8,8)\} = \{(c,2) - (a,b)\}$

$(c,2) = 2(9,5) - (8,8)$

$= 10,2$

$c = 2.$

Thus, $a = 9, b = 5, c = 2.$

Computer Graphics And Multimedia

Topic:

B-Spline Curve And Its Properties

Contributed By:

Ankita Jiyani

B-Spline Curve & it's Properties

Properties of B-spline curves;
Basis-splines.

1. The sum of the B-spline basis functions for any parameter value is 1.
2. The max. order of the curve is equal to the no. of vertices of defining polygono
3. The degree of B-spline polynomial is independent on the no. of vertices defining polygon.
4. B-spline allows the local control over the curve surface because each vertex affects the shape of a curve only over a range of parameter values where its associated basis function is non zero.
5. The curve generally follows the shape of defining polygon.

Page No.:

6. B-spline curve is made of $n+1$ control points & order of the curve is k, where k is the segments form in the curve. The value of k is decided by the user.

Suppose, if we have segments no, k then it will give us the polynomial of degree $k-1$.

$k=3$, degree $- x^2$

$k=4$, $p(x) = x^3$

$2 \leq k \leq n+1$

7. The no. of segments, $n-k+2$.

$n=6$, control point $\rightarrow 7\ (n+1)$

segment; $k=3$.

$6-3+2 = 5$

Total segments $= 5$.

Variable, $u \Rightarrow 0$ to 5.

8. B-spline curve allow us to change the no. of control points without changing the degree of the polynomial.

Blending Function for B-Spline Curve

Blending function for B-spline curve :-

$$P(u) = \sum_{i=0}^{n} B_{i,k}\, u\, (P_k)$$

where, $P_k(x_k, y_k, z_k)$

$$x(u) = \sum_{i=0}^{n} B_{i,k}(u)\, x_k$$

$$y(u) = \sum_{i=0}^{n} B_{i,k}(u)\, y_k$$

$$\begin{cases} 0 \le u \le n-k+2 \\ 0 \le k \le n+1 \end{cases}$$

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

t_i $(0 \le i \le n+k)$ - knot value.

$t_i = 0$ if $i < k$

$t_i = i - k + 1$ if $k \le i \le n$

$t_i = n - k + 2$ if $i > n$

$B_{i,k}(u) = 1$ if $t_i < u \le t_{i+1}$

$= 0$ otherwise

Ques Construct a B-spline curve, with 6 control points & 3 segments.

$n+1 = 6$.

$n = 5$, $k = 3$

$0 \le ti \le 5+3$ = $0 \le ti \le 8$

knot values.

$P(u) = Bi,k(u)\, pi$

For, $B_{0,3}$, $t_0 = 0$, $0 < 3$, $i < k$.

For, $B_{1,3}$, $t_1 = 0$ $1 < 3$ $i < k$

For $B_{2,3}$, $t_2 = 0$ $2 < 3$ $i < k$

For $B_{3,3}$ $t_3 = 1$, $i-k+1$, $3-3+1$, $k \le i \le n$.

For $B_{4,3}$ $t_4 = 2$, $i-k+1$, $4-3+1$, $k \le i \le n$.

For $B_{5,3}$ $t_5 = 3$, $i-k+1$, $5-3+1$, $k \le i \le n$.

For $B_{6,3}$ $t_6 = 4$, $n-k+2$, $5-3+2$, $i > n$

For $B_{7,3}$ $t_7 = 4$ $n-k+2$, $5-3+2$, $i > n$

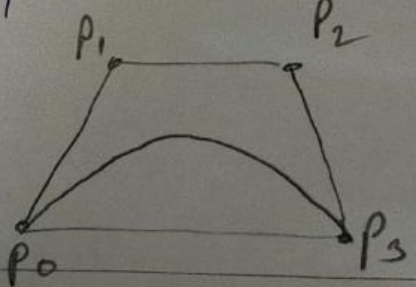
For, $B_{8,3}$ $t_8 = 4$ $n-k+2$ $5-3+2$, $i > n$

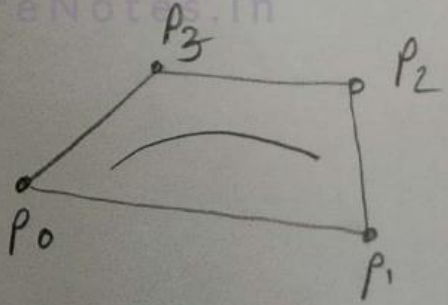
$B_{0,3}(u) = (1-u)^2 N_{2,1}(u)$.

Difference between Bezier and B-Spline Curves

Difference b/w Bezier & B-spline curve.

Bezier curve	B-spline curve.
4. It has global control over the control points. Changing one control point effects whole curve.	It has local control over the control points. Changing one control point effect only one part of the curve.
The degree of curve depends on the no. of control points. degree = n-1	The degree of curve depends on the no. of segments. degree = k-1.
It is defined as curve lies in the convex hull of their control points.	It is defined as curve lies within the convex hull of their control points.





Bezier curves

The curve touches it first & last control point.

The range of u, lies b/w, $0 \leq u \leq 1$

B-spline curves

The curve here is not necessarily touches the first & last control points.

The range of u, lies.

b/w $0 \leq u \leq n-k+2$

& $0 \leq k \leq n+1$

Computer Graphics And Multimedia

Topic:

Illumination Model

Contributed By:

Ankita Jiyani

Illumination Model

Illumination Model :-

- It is used to calculate the intensity of light of a point on the surface of an object.
- It is also known as lighting model or shading model.
- The calculation done by the model is used by surface rendering method to determine the light intensity for all projected pixels for various surfaces in a scene.
- The model will calculate the intensity projected from a particular surface point in a specified viewing direction using the following parameters.
  i) optical properties of a surface.
  ii) The relative positions of the surfaces in a given scene.

iii) the color & positioning of light sources.
iv) positions & orientation of the viewing plane.

Light sources :-
↳ Light source is the source which emits the light.
↳ when we view an opaque object, we see the reflected light from the surface of the object
↳ total reflected light = light directly from light source / light emitting source.
+ light from reflecting surface / light reflecting src

Point source :-

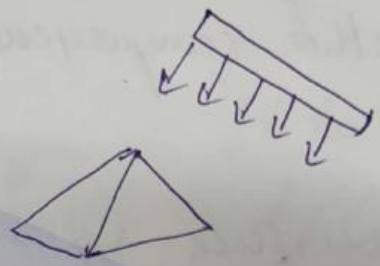
↳ Rays from the source follow radially diverging paths from the source.
↳ point source dimensions are small as compared to size of the objects in the scene.
Eg :- Sun (sufficiently far from the scene can be accurately modeled as point source).

Types of Light Sources

Distributed light source:

A area of the source is not small compared to the surfaces in the scene.

eg: A fluorescent light



Parallel light source:

A source of light which is far away and the rays coming from the source is considered as parallel light source.

There are 3 types Reflection:-

1. Ambient Reflection:

The light is coming from different source. And the light is reflected rays from other object.

Or, The object is visible when nearby objects are illuminated.

Intensity = Ia Ka

$0 \leq Ka \leq 1$

Ia → Intensity of Reflection

Ka → Diffuse Reflectivity. Coefficient.

Diffuse Reflection :-

Reflection from the rough surfaces are scattered. They are scattered in all directions.

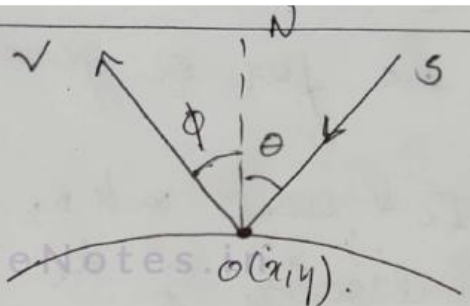
Eg: A blue object illuminated by a white light source reflects the blue component of the white light & absorbs all other component.

Specular Reflection :

The shiny surface appearance, when light is reflected from any object is called as specular reflection.

or Highlights or bright spots appearence is a specular reflection.

It mostly found on shiny surfaces



Intensity diffuse $= f(\mathcal{I}_I, \theta, S, \Phi) + A(V)$

$\mathcal{I}_I \rightarrow$ intensity of light source.

$S \rightarrow$ reflectivity coefficient

$A(V) \rightarrow$ Ambient light vector.

For ideal diffuse reflection;
the light coming from light source, after hitting any surface, part of it reflected back at same angel. So discarding Φ from above function.

$$I_{diff} = f(\mathcal{I}_I, \theta, S) + A(V)$$

Also known as, Lambertian reflector.

Computer Graphics And Multimedia

Topic:

Lambert's Cosine Law

Contributed By:

Ankita Jiyani

Lambert's Cosine Law

Lambert's cosine law :-

How much light the surface receives from a light source depends on the angle b/w its normal and vector from the surface point to the light.

Lambert's law : The radiant energy Id from a small surface da for a given light source is :

$$Id = I_L * \cos\theta * k_d$$

I_L - Intensity of the light source.

θ - angle b/w the surface normal N & light vector L.

↳ A surface is illuminated by a point source only if the angle of incidence is in the range 0° to 90°, i.e $0 \le \cos\theta \le 1$

↳ when $\cos\theta$ is -ve, the light source is behind the surface.

↳ The total reflection is :- ambient + diffuse

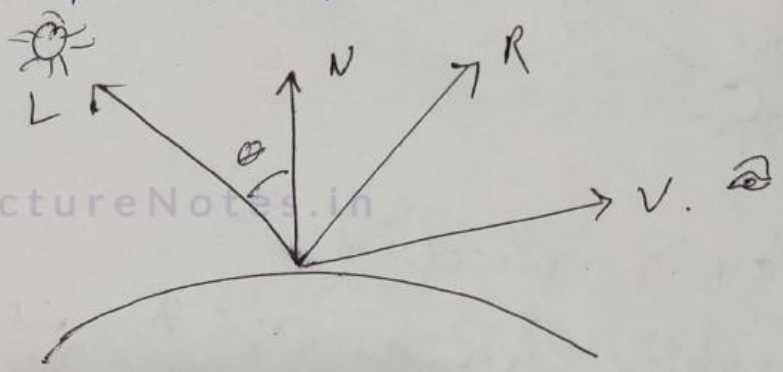
$$I_{diff} = k_a * I_a + k_d * I_L * (N * L)$$

$$\cos\theta = N \cdot L$$

N → Normal vectors to surface

L = The unit direction vector to the point light src.

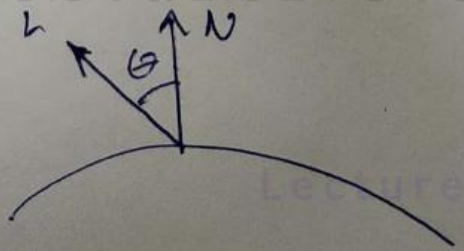
$0 \leq k_d, \quad k_a \leq 1$



Types of Illumination Model :-

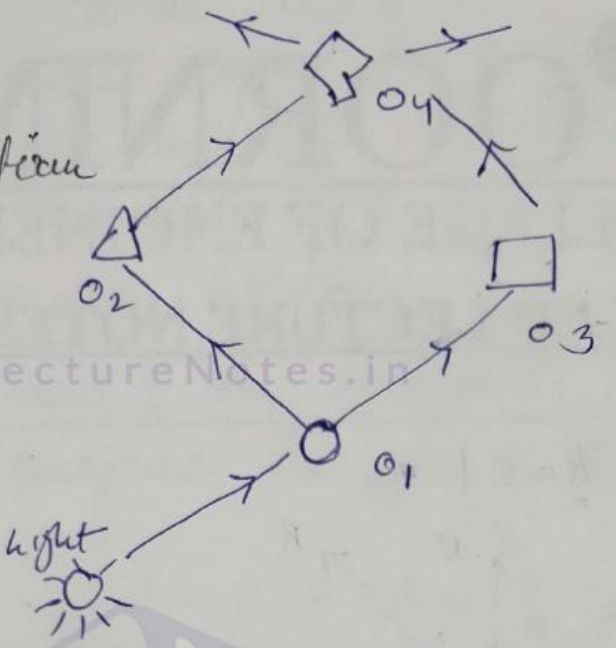
Global Illumination model :- It takes into account the interaction of light from all the surfaces in the scene.

Local Illumination model, only considers the light, the observers position & object material properties.



Global Illumination Model

Global Illumination Model.



Illumination vs Shading.

→ Illumination determines the color of a surface point by simulating scene light attributes.

→ Shading model applies the illumination model at a set of points ? color the whole image.

Computer Graphics And Multimedia

Topic:

Shading And It's Types

Contributed By:

Ankita Jiyani

Shading and it's types

Shading : By using intensity calculated by illumination model, we need to color the whole object.

Types of Shading :-

(1) Flat shading or Constant shading.

It is a technique used in 3D graphics. It shades each polygon of an object based on the angle between the polygon's surface normal & the direction of light source, their respective colors & the intensity of light source.

It is usually used for high speed rendering where more advanced shading techniques are too computationally expensive.

Disadvantage :

→ It produce unrealistic look.

→ It produce much bund effect.

Page No.:

→ Flat shading of polygon facets provides an accurate rendering for an object if all of the following assumptions are valid :-

1) The object is a polyhedron & is not an approximation of an object with a curved surface.

2). All light sources illuminating the object are sufficiently far from the surface so that $N.L$ & the attenuation function are constant over the surface.

3). The viewing position is sufficiently far from the surface so that $V.R$ is constant over the surface.

Application :

- The algorithm is applied to the scene when both light source & viewer are far distant from the object.
- To display fast moving object in a scene, this algorithm is applicable.

Gouraud Shading

The mach band effect occures, if Intensity varies uniformly.

Gaurand shading overcomes this limitation

Gaurand Shading :-

↳ It is an interpolation technique developed by Henri Gouraud in 1976.

↳ Intensity levels are calculated at each vertex and interpolated across the surface of a polygon.

↳ It eliminates the discontinuties of the Intensity, as intensity values for each polygon are matched with the values of adjacent polygons along the common edges.

↳ To renders a polygon, following steps are performed;

1) Determine the average unit normal vectors at each polygon vertex.

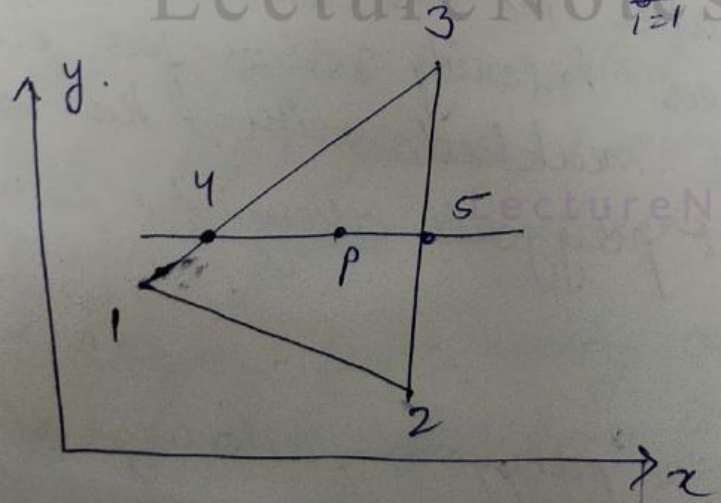
2) Apply an illumination model to each vertex to calculate the intensity of the vertex.

3). Interpolate the vertex intensity in linear manner, over the surface of the polygon.

The average unit normal vector at v is given as :-

$$N_v = \frac{N_1 + N_2 + N_3 + N_4}{|N_1 + N_2 + N_3 + N_4|}$$

More generally :- $N_v = \frac{\sum_{i=1}^{n} N_i}{\sum_{i=1}^{n} N_i}$



The intensity at point 4 can be interpolated from intensity 1 & 3.

$$I_4 = \frac{y_4 - y_3}{y_1 - y_3} I_1 + \frac{y_1 - y_4}{y_1 - y_3} I_3$$

The intensity at point 5. can be interpolated from intensity 2 & 3.

$$I_5 = \frac{y_5 - y_2}{y_3 - y_2} I_3 + \frac{y_3 - y_5}{y_3 - y_2} I_2$$

The intensity at point p, can be interpolate from intensity 4 & 5.

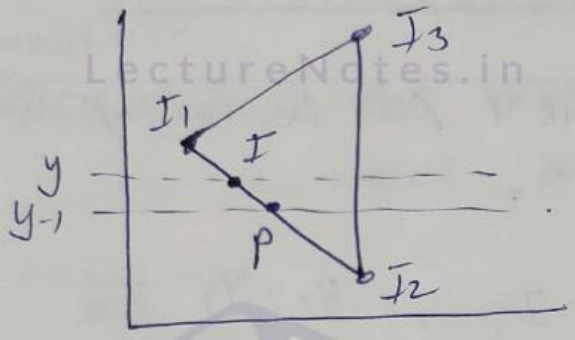
$$I_p = \frac{x_5 - x_p}{x_5 - x_4} I_4 + \frac{x_p - x_4}{x_5 - x_4} I_5$$

If the intensity at edge position (x, y) is interpolated as:

$$I = \frac{y - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y}{y_1 - y_2} I_2$$

then we can obtain the intensity along this edge for the next scan line $y-1$ as:

$$I' = I + \frac{I_2 - I_1}{y_1 - y_2}$$



Advantages:-

It removes the discontinuities associated with flat shading model.

Disadvantages:-

Highlights on the surface are sometimes displayed with anomalies shape, and linear interpolation can cause bright or dark intensity streaks called Mach bands to appear on the surface. This is overcome by Phong shading.

Phong Shading

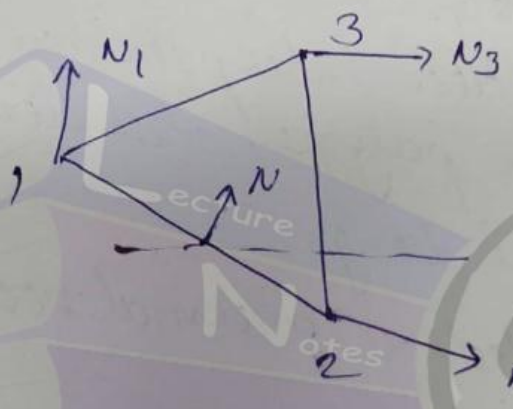
PHONG SHADING.

- A more accurate interpolation based approach for rendering a polygon was developed by Phong Bui Tong.
- It is also called as normal vector interpolation rendering.
- It interpolates normal vectors instead of intensity values.
- The rendering of a polygon is performed by using following steps :-
  1) Determine the average unit normal vector at each vertex of the polygon.
  2) Linearly interpolate the vertex normal over the projected area of the polygon.

Page No.:

3) Apply an illumination model at positions along scan lines to calculate pixel intensities using the interpolated normal vectors.

- The normal vector N for the scan-line intersection point along the edge b/w vertices 1 & 2 is obtained as:-



$$N = \frac{y - y_2}{y_1 - y_2} N_1 + \frac{y_1 - y}{y_1 - y_2} N_2$$

- Incremental methods are used to evaluate normals between scan lines & along each individual scan-lines.

↳ At each pixel position along a scan line, the illumination model is applied.

Advantages :-

- Intensity calculations are more accurate
- Displays more realistic highlights on a surface.
- Reduces Mach Band effect.
- Suitable for shiny surfaces.

Dis advantages :-

It require more calculations and greatly increases the cost of shading steeply