Computer Graphics

Contributed By:
Verified Writer

Computer Graphics

Topic:

Overview Of Graphics System

Contributed By:

Verified Writer

23/6/14

Computer Graphics

Computer graphics involve display, manipulation, storage of picture and experimental data for proper visualization.

A computer graphics system consisting of

1) A host computer
2) Processor
3) Memory
4) Frame buffer
5) Display device
6) A set of input device

System Model / CG Systems

Application Model → Application Program → Graphics System → Monitor

Computer Graphics Package

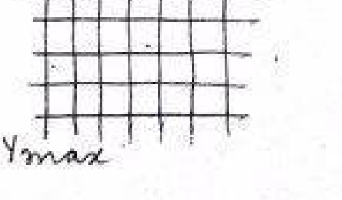
1) Turbo C
2) Core graphics
3) Graphics Kernel System
4) Open GL (Open graphics library)

Set of Computer Graphics Devices

1) CRT Monitor
2) EGA - Extended Graphics Adapter
3) CGA - Colour graphics adapter
3) VGA - Vector graphics adapter

How to display / draw an image in a computer?

1) Raster scan display
2) Random scan display



Raster scan display is a point plotting device, or pixel plotting device, each dot - smallest dot in the picture → is a pixel

~~Pixel~~ store the information in terms of pixel. It store the display primitive like line, rectangle, character or any type of geometrical object in refresh buffer or frame buffer.

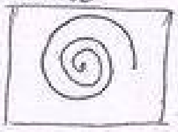
Refresh buffer also called as frame buffer which store the geometrical object in terms of pixel / point component. The whole screen is a matrix of pixel in case of Raster scan display. All info stored in row & column format. Each pixel brightness can be controlled by electron beam. Refresh buffer can be visualized as a set of

(2) horizontal raster line for a row of individual pixels

Random Scan Display

Random scan also known as vector scan, in random scan display, the electron beam directed only to the part of the screen where the picture is to be drawn. In this, a picture can be drawn in terms of line, one at a time

In this the refresh rate depends on the number of line to be displayed.



In this, all the system having high resolution as compared to the raster scan display.

Resolution - no of pixel in row/column

24/6/19

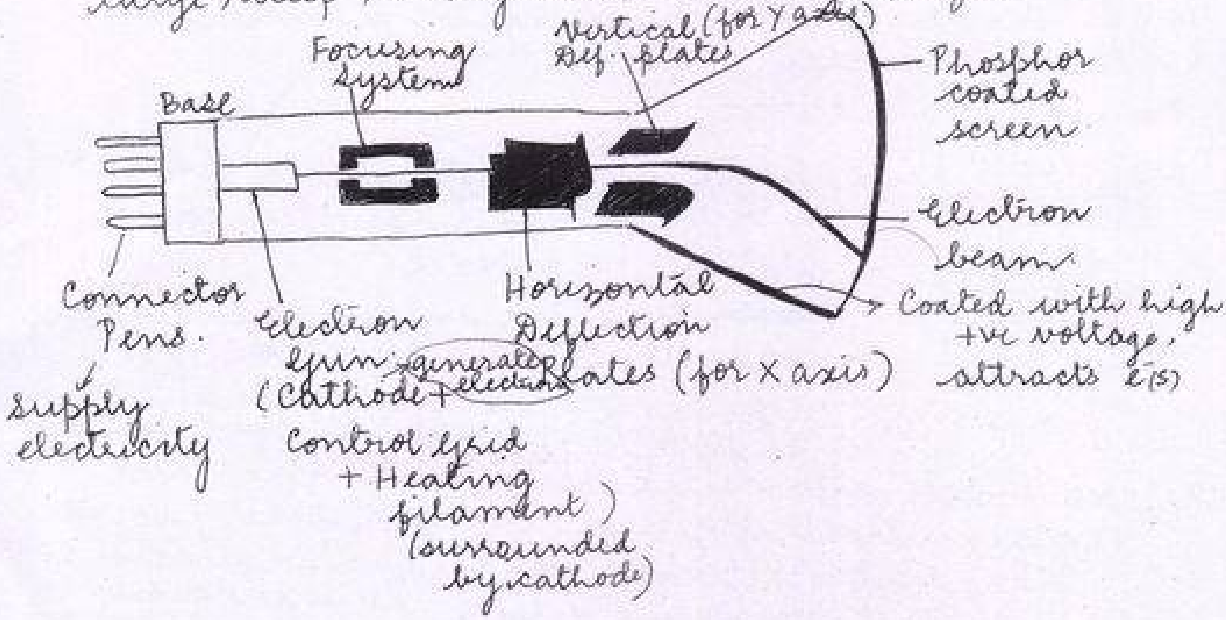
Overview (Display Devices)

- Raster Scan Display
- Random Scan Displays
- Color CRT Monitors

The display system mainly consists of Monitor

Cathode Ray Tube (CRT)

The CRT uses an evacuated glass envelope which is large, deep, heavy and relatively fragile

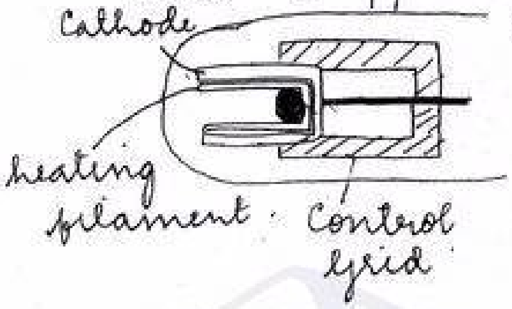


Refresh CRT.

- Light emitted by the Phosphor fades ~~stopping~~ very rapidly
- Refresh CRT One way to keep the phosphor glowing is to redraw the picture repeatedly by quickly directing the electron beam back over the same points.

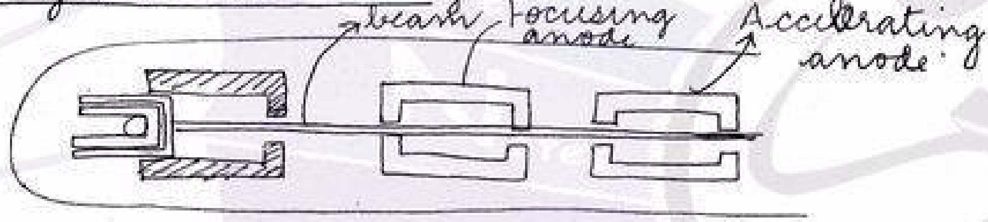
Electron gun.

Heat is supplied to the cathode by the filament
Cathode emits e^- beam.



The free e^-(s) are then accelerated by +ve charged plate

High Positive Voltage

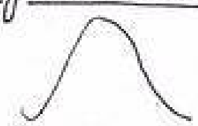


Persistance

The time it takes the emitted light from the screen to decay one tenth of its original intensity

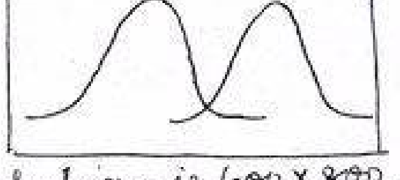
Intensity Distribution

Refer slides



Resolution (Spots of light)

The maximum number of points can be displayed without overlap on a CRT



If resolution is 600 x 800, then total pixels are 600 x 800 = 480000

Resolution of a CRT is dependent on:
The type of phosphor
The intensity to be displayed
The focussing & deflection system

Aspect Ratio :-

This numbers gives the ratio of vertical points to horizontal points necessary to produce equal length lines in both direction the screen.
Standard aspect ratio is 4:3 (i.e 800 1600 that is

$$\frac{\text{vertical points}}{\text{horizontal points}}$$

Raster Scan Desplays

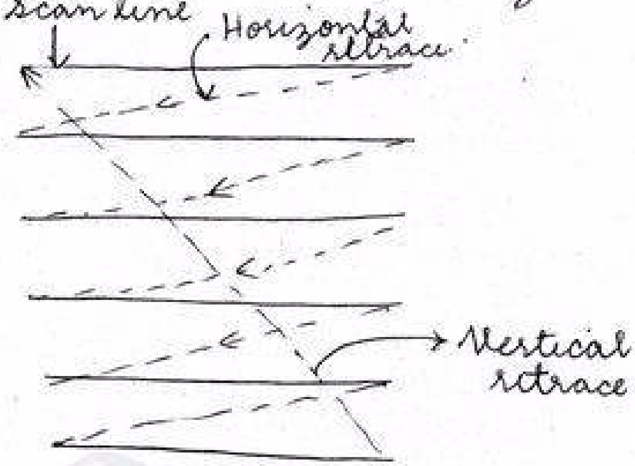
Raster : A rectangular array of points or dots
Pixel : One dot or picture element of a raster
Scan Line : A row of pixel.

- In a raster scan system, the electron beam is swept across the screen, one row at a time from top to bottom.
- Horizontal retrace - scan the line horizontally
- Vertical retrace - scan the line vertically

- As electron beam moves across each row, the beam intensity is turned on and off to create a pattern of illuminated spots.
- Refresh buffer | frame buffer : This memory area holds the set of intensity values for all the screen points.
- On a black & white system with one bit per pixel, the frame buffer is called bitmap
- For system with multiple bits per pixels, the frame buffer is called pixmap.
- Sometimes refresh rates are described in unit of cycles per second or Hertz (HZ).
  No of frame displayed / second - refresh rate

- Refreshing on raster scan displays is carried out a the rate 60 to 80 frame per second. (5)

- Horizontal Retrace – the return of to the left of the screen, after refreshing each scan line.



Vertical retrace: At the end of each frame (displayed in $1/80^{th}$ to $1/60^{th}$ of a second) the electron beams returns to the top left corner of the screen to begin the next frame.

25/6/14. Interlacing Scan.

To reduce flicker, divide frame into two fields – one consisting of even scan line and the other of odd scan lines.

Even and odd fields are scanned out ~~all~~ alternately to produce an interlaced image.

On an older, 30-frame per-second, noninterlaced display, some flicker is noticeable.

An effective technique for avoiding flicker.

Q Find out the aspect ratio of a raster system using 8 X 10 inches screen and 100 pixels per inches.

80
800X1000
4:5

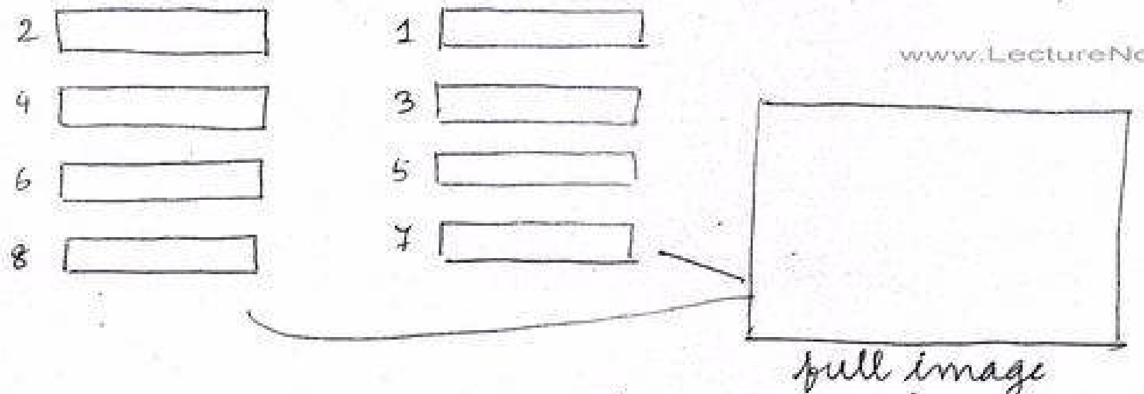
~~aspect ratio~~.

Resolution = 800 x 1000

Aspect ratio = 4 : 5.

6

interlaced



Field 1 + Filed 2 = Frame (Complete image)

Display rate : 60 fields/sec (North America).

Interlacing

Random Scan Display

- It is the use of geometrical primitives such as points, lines, curves, and polygons, which are all based up on mathematic eqn
- To display a

Q A raster scan system with resolution of 1024 x 1024 is given. What is the size of the raster in bytes needed to store 4 bit/pixel. How much storage is required if 8 bit/pixel is used.

ans. 524288 bytes.

$\frac{1024 \times 1024 \times 4}{8}$ bytes

$\frac{1024 \times 1024 \times 8}{8}$ bytes.

1024 X 1024 - total pixels.
each pixels requires 4 bits
so 1024 X 1024 X 4.
dividing by 8 gives us answer in bytes.

Q. How long will it take to load 640 x 480 frame buffer with 12 bits/pixel. If 10^5 bits can be transferred per second, find out the total time.

$$\frac{640 \times 480 \times 12}{10^5} = 36.86 \text{ seconds}.$$

640 X 480 – total pixel
640 X 480 X 12 – total bits.

Q. Find the refresh rate of a 512 x 512 frame buffer if the access time for each pixel is 200ns.

$$\frac{512 \times 512 \cancel{\times 200 \times 10^{-9}}}{200 \times 10^{-9}}$$

$$\text{refresh rate} = \frac{1}{\text{Total acc. time}}$$

$$= \frac{1}{200 \times 10^{-9} \times 512 \times 512}$$

$$= \frac{10^9}{200 \times 512 \times 512}$$

$= 19$ (approx)

Q. Find the access time per pixel for a raster system 640 X 480 with refresh rate ~~60Hz~~ 60 frame/sec.

640 X 480 total pixels

$$60 = \frac{1}{\text{total access time}}$$

$$60 = \frac{1}{\text{acc. time per pixel} \times 640 \times 480}$$

$$\text{acc. time} = \frac{1}{60 \times 640 \times 480}$$

$$= 54.2 \text{ ns}$$

Computer Graphics

Topic:

Line Drawing Algorithms DDA Bresenhams

Contributed By:

Verified Writer

Digital Differential Analyzer Algorithm

$y = mx + c$

Let us consider eq. of line
where m = slope of line
c = y intercept

Let us assume two end points of the line is
(x_i, y_i) and (x_{i+1}, y_{i+1}).

$$m = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$$

DDA for line with slope < 1

Let us $\Delta x = 1$ incase slope < 1

where $\boxed{\Delta x = x_{i+1} - x_i}$

$$\boxed{x_{i+1} = x_i + 1} \quad - (1)$$

So $m = \frac{y_{i+1} - y_i}{1}$

$= y_{i+1} - y_i$

$$\Rightarrow \boxed{y_{i+1} = y_i + m} \rightarrow (2)$$

Q. Calculate the intermediate point of the given line having end point (0,0) and (11,6).

$$m = \frac{6-0}{11-0}$$

$$= \frac{6}{11} = 0.5$$

Intermediate points

$x_0 = 0$, $y_0 = 0$. ⇒ initial point (the left most point).

i	x_{i+1}	y_{i+1}
0	1	6/11
1	2	12/11
2	3	18/11
3	4	24/11
⋮	⋮	⋮
10	11	66/11 = 6

30/6/14. DDA Line Drawing Algorithm.

Case-I

$m < 1$

$$\boxed{\begin{aligned} x_{i+1} &= x_i + 1 \\ y_{i+1} &= y_i + m \end{aligned}}$$

Case 2: $m > 1$

(x_i, y_i) & (x_{i+1}, y_{i+1})

$$m = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} = \frac{\Delta y}{\Delta x}$$

For $m > 1$, assume $\Delta y = 1$

$$\Delta y = y_{i+1} - y_i = 1$$

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

$$m = \frac{\Delta y}{\Delta x} = \frac{1}{\Delta x}$$

$$m = \frac{1}{\Delta x}$$

$$\Delta x = \frac{1}{m}$$

$$x_{i+1} - x_i = \frac{1}{m} \Rightarrow \boxed{x_{i+1} = x_i + \frac{1}{m}}$$

DDA Alogorithm

1.) Input two end points of the line (x_1, y_1) & (x_2, y_2)
2). Calculate $\Delta x = x_2 - x_1$ and $\Delta y = y_2 - y_1$
3) If ~~absolute value of Δx~~ abs(Δx) > abs(Δy) then ~~define~~
4) ~~another variable~~ step = abs(Δx

~~4. Abs(Δx) = step~~

5) else
6) step = abs(Δy)
7) $x_{inc} = \frac{dx}{step}$
8) $y_{inc} = \frac{dy}{step}$
9) $x = x_1$, & $y = y_1$
10) putpixel(x, y, color)

standard graphics func^n used to plot a point on the graphic screen
x → x-co-ordinate
y → y-co-ordinate
color → color of the pixel

11) for (k = 0 ; k < step ; k++)
12) x = x + x_{inc}
13) y = y + y_{inc}
14) putpixel(x, y, color)
15) end

(10) Using DDA line drawing algorithm, calculate the points b/w two end points (6, 9) and (11, 19).

$\Delta x = 11 - 6 = 5$

$\Delta y = 19 - 9 = 10$

abs(Δy > abs(Δx)

so step = 10.

$x_{inc} = \frac{5}{10} = \frac{1}{2}$

$y_{inc} = \frac{10}{10} = 1$

~~put~~ $x = 6$ $y = 9$.

k	x	y
0	6.5	10
1	7	11
2	7.5	12
3	8	13
4	8.5	14
5	9	15
6	9.5	16
7	10	17
8	10.5	18
9	11	19
~~10~~	~~11.5~~	~~20~~

Q Calculate intermediate points for end points ~~for~~ (0, 0) & (-8, -4).

Take lesser value as ~~end~~ initial point always

$\Delta x = -8$

$\Delta y = -4$

$\frac{\Delta x}{\Delta y} = \frac{-8}{-4} = 2$

abs(Δx) > abs(Δy) so step = abs(Δx) = 8

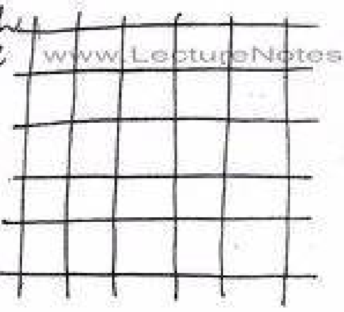
$x_{inc} = \frac{8}{8} = 1$ $x = -8$ $y = -4$

$y_{inc} = \frac{4}{8} = \frac{1}{2}$

	x	y
0	-7	-3.5
1	-6	-3
2	-5	-2.5
3	-4	-2
4	-3	-1.5
5	-2	-1
6	-1	-0.5
7	0	0

Drawbacks of DDA Algorithm

(round(x), round(y), color).



1. It uses floating point operation which is expensive
2. It produces aliasing effect

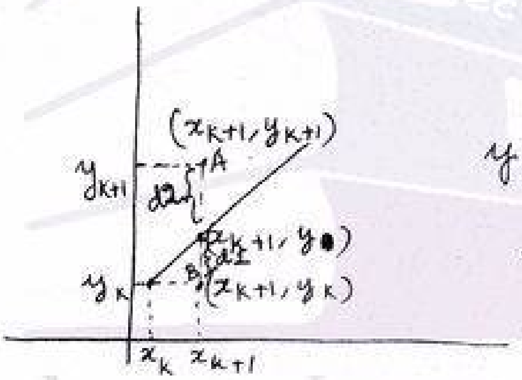
→ The coordinates are always integer values but by using this algorithm we get floating point values which the computer screen does not allow.

∴ putpixel (round(x), round(y), color)
round() used for floating point operation

→ exact location / exact line is not found, we get a zigzag line.

1/7/19.

Bresenham's Line Drawing Algorithm



$m < 1$

$x_{K+1} = x_K + 1$

$y = mx + c$
$= m(x_K + 1) + c$
$= m(x_K + 1) + c$.

$d1 = y - y_K$
$= m(x_K + 1) + c - y_K$

$d2 = y_{(K+1)} - y = y_K + 1 - y$
$= y_{K+1} - m(x_K + 1) - c$.

$d1 - d2 = 2m(x_K + 1) - 2y_K + 2c - 1$

$$\boxed{d1 - d2 = 2m(x_K + 1) - 2y_K + 2c - 1}$$

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

$d1 - d2 = 2\frac{\Delta y}{\Delta x}(x_K + 1) - 2y_K + 2c - 1$

Multiply Δx on both sides:

$\Delta x(d1 - d2) = 2\Delta y(x_K + 1) - 2y_K\Delta x + 2c\Delta x - \Delta x$

$$\boxed{P_K = 2\Delta y(x_K + 1) - 2y_K\Delta x + 2c\Delta x - \Delta x}$$

Case 1.

If $P_K > 0$ then $d1 > d2$.
choose the next point A with co-ordinates (x_{K+1}, y_{K+1})

Case 2

If $P_K < 0$ then $d1 < d2$.
choose the next point as B with co-ordinates $(x_K + 1, y_K)$

Case 3

If $P_K = 0$.
then A and B are on the line since A = B.
Choose either A or B

$$P_{K+1} = \cancel{2\Delta y(x_{K+1}+1) + 2\Delta y + 2\Delta x y_{K+1} + 2c\Delta x - \Delta x}$$

$$= 2\Delta y(x_K + 1 + 1) - 2y_{K+1}\Delta x + 2c\Delta x - \Delta x$$

$$= 2\Delta y(x_K + 1) + 2\Delta y - 2y_{K+1}\Delta x + 2c\Delta x - \Delta x.$$

$$P_{K+1} - P_K = 2\Delta y - 2\Delta x(y_{K+1} - y_K)$$

$$\Rightarrow \boxed{P_{K+1} = P_K + 2\Delta y - 2\Delta x(y_{K+1} - y_K)}$$

Case I :

We are choosing point A.

$y_{K+1} = y_K + 1$

$P_{K+1} = P_K + 2\Delta y - 2\Delta x$.

Case II

We are choosing point B.

$y_{K+1} = y_K$

$P_{K+1} = P_K + 2\Delta y$.

Case III

Choose either A or B.

Calculation of P_0 value.

$y = mx + c$

Assume that the line is passing through initial point (x_0, y_0)

$\rightarrow y_0 = mx_0 + c$

$c = y_0 - mx_0$

$= y_0 - \frac{\Delta y}{\Delta x} x_0$.

$\Rightarrow \Delta x c = y_0 \Delta x - \Delta y x_0$

$\Rightarrow 2c\Delta x = 2y_0 \Delta x - 2\Delta y x_0$

In P_K expression put $k=0$

Imp. $P_0 = 2\Delta y x_0 - 2\Delta x y_0 + 2\Delta y + 2y_0\Delta x - 2\Delta y x_0 - \Delta x$

$\boxed{P_0 = 2\Delta y - \Delta x}$ Eq^n for P_0 (Decision Variable) can be defined as the distance b/w actual line and nearest pixel

Q Generate the intermediate point b/w (20,10) and (30,18) using Bresenham's Line Drawing Algorithm

$\Delta y = 8$

$\Delta x = 10$

$m = \frac{\Delta y}{\Delta x} = \frac{8}{10} < 1$

K	P_K	x_{K+1}	y_{K+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

2/7/14

Q Find the intermediate points b/w two end point (0,2) and (4,5) using Bresenham's line drawing algorithm

$m = \frac{\Delta y}{\Delta x} = \frac{5-2}{4-0} = \frac{3}{4} < 1$ $P_0 = 2\Delta y - \Delta x$

$\Delta y = 3$

$\Delta x = 4$

K	P_K	x_{K+1}	y_{K+1}
0	2	1	3
1	0	2	4 → Pt (A)
2	-2	3	4
3	4	4	5

(14) $|m| < 1$

1) Input the two end point of the line and store the end point as x_0 and y_0.

2) Plot the first point (x_0, y_0) using putpixel $(x_0, y_0, color)$

3) Calculate Δx and Δy value

4) Calculate the initial decision parameter value P_0 as $\boxed{2\Delta y - \Delta x}$

5) For each x_K starting $K=0$, perform the following test

Case 1 : If $P_K > 0$, the next point is (x_K+1, y_K+1)

$$\boxed{P_{K+1} = P_K + 2\Delta y - 2\Delta x}$$

Case 2 : If $P_K < 0$

Next point is (x_K+1, y_K)

$$\boxed{P_{K+1} = P_K + 2\Delta y}$$

Case 3 : If $P_K = 0$

Choose either (x_K+1, y_K+1) or (x_K+1, y_K)

$$\boxed{P_{K+1} = P_K + 2\Delta y - 2\Delta x \quad \text{or} \quad P_{K+1} = P_K + 2\Delta y}$$

6. Repeat step no. 5, Δx times

<u>Pseudo Code</u>

```
void BLDA(int x1, int y1, int x2, int y2, color)
{
    int d, x, y;
    dx = x2 - x1
    dy = y2 - y1
    x = x1
    y = y1
    PutPixel(x, y, color)
    d = 2*dy - dx
    while(x <= x2)
    {
        If(d <= 0)
        {   x++
            d = d + 2*dy
        }
```

```
else
 {
   x++;
   y++
   d = d + 2*dy - 2*dx
 }
putpixel (x, y, color)

}
}
```

BLDA for slope greater than one

$|m| > 1$

1) Put the two end points of line and store the left end point as x_0, y_0
2) plot the first using putpixel (x_0, y_0, color)
3) calculate dx and dy.
4) Calculate initial decision parameter $\boxed{P_0 = 2*dx - dy}$
5) For each y_K value starting at $K=0$ perform the following test:

   Case I: If $P_K > 0$ then the next point is
   (x_K+1, y_K+1)
   $P_{K+1} = P_K + 2*dx - 2*dy$

   Case II: If $P_K < 0$ then
   (x_K, y_K+1)
   $P_{K+1} = P_K + 2*dx$

   Case III: $P_K = 0$
   (x_K+1, y_K+1) or (x_K, y_K+1)
6) Repeat step no 5, for dy times
7) end.

Q Generate the intermediate point using BLDA b/w two end points (3, 2) and (4, 6).

$m = \frac{\Delta y}{\Delta x} = \frac{4}{1} = 4 > 1$

$P_0 = 2 \times 1 - 4 = -2$

K	P_K	x	y
0	-2	3	3
1	0	4	4
2	-6	4	5
3	-4	4	6.

P_K	x	y
-2	3	3
0	3	4
2	4	5
-4	4	6

Computer Graphics

Topic:

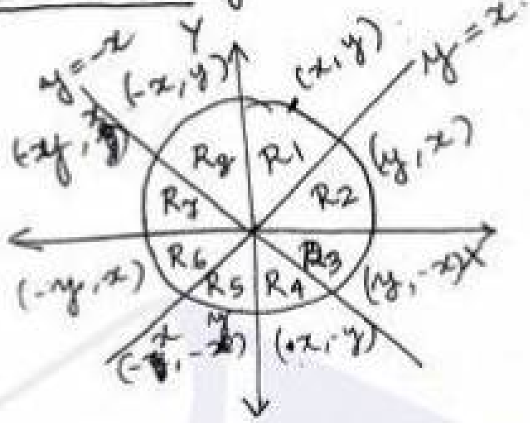
Circle Drawing Algorithms

Contributed By:

Verified Writer

7/7/14

Circle Drawing



Mid Point Circle Drawing algorithm

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

Circle is 8-way symmetric in nature.

0+2−2×4
2−8
=−6.

−6+2×1
=−6+2
=−4

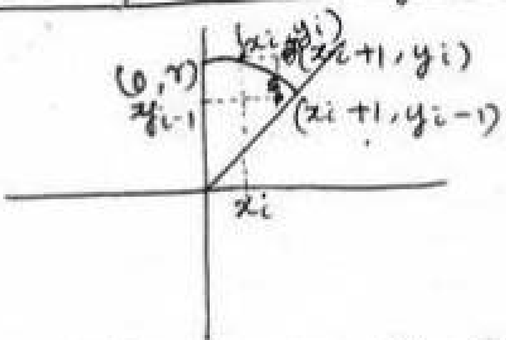
Mirror point reflection procedure

void mirrorpt(x, y, color).

{ putpixel(x, y, color)
putpixel (y, x, color).

putpixel(−x, y, color).

Calculate point on region one.



Let us take a mean point b/w T and S.

midpoint $= (x_i+1, y_i - 1/2)$

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

$f(x_i+1, y_i-1/2) = (x_i+1)^2+(y_i-1/2)^2-r^2$

$= (x_i+1)^2+(y_i-1/2)^2-r^2$

$\Rightarrow P_i = (x_i+1)^2+(y_i-1/2)^2-r^2$

If P_i is negative midpoint is inside the circle.

$x_i+1 = x_i+1$

$P_i < 0$

If P_i is positive, then mean point is outside the circle, so point S is close to the circle boundary then S is next point. If P_i is zero choose S or T.

$P_{i+1} = (x_{i+1}+1)^2+(y_{i+1}-1/2)^2-r^2$

$= ((x_i+1)+1)^2+(y_{i+1}-1/2)^2-r^2$

$$\boxed{P_{i+1}-P_i = 2(x_i+1)+1+(y_{i+1}^2-y_i^2)-(y_{i+1}-y_i)}$$

$$\boxed{P_{i+1} = P_i+2(x_i+1)+1+(y_{i+1}^2-y_i^2)-(y_{i+1}-y_i)}$$

If P is -ve choose the pixel P then the new value of y.

$y_{i+1} = y_i$

$P_{i+1} = P_i+2(x_i+1)+1$

$$\boxed{P_{i+1} = P_i+2x_i+3}$$

on choosing point S, the P_{i+1} value is:

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

$$\boxed{P_{i+1} = P_i+2(x_i-y_i)+5}$$

Calculation of initial decision variable value

$P_i = (x_i+1)^2+(y_i-1/2)^2-r^2$

$P_0 = (x_0+1)^2+(y_0-1/2)^2-r^2$

$x_0=0, y_0=r$

$\boxed{P_0 = \frac{5}{4}-r} \simeq \boxed{P_0 = 1-r}$

Find the point of the first region of the given circle $x^2+y^2=10$ using midpoint circle drawing algorithm

$x^2+y^2=100$

$x_0=0,\ y=10$

$P_0=1-r=1-10=-9$

i	P_i	x_{i+1}	y_{i+1}
0	-9	1	10
1	-6	2	10
2	-1	3	10
3	6	4	9
4	6	.	.
.		.	.
		7	7

$P_{i+1}=P_i+2x_i+3$

$P_1=P_0+2x_0+3$

$=-9+2\cdot0+3$

$=-6.$

$P_4=P_3+2x_3+3$

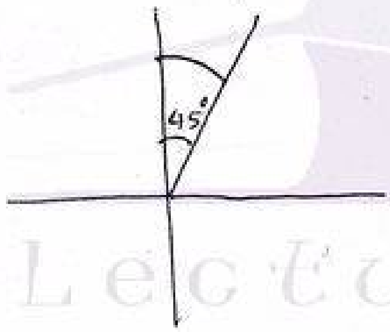
$=-1+4+3=6$

As.

$P_{i+1}=P_i+2(x_i-y_i)+5$

8/7/14

Find the points in the first octant from 90°-45° using above algo. given circle $x^2+y^2=64$.



$x_i=0,\ y_i=$

i	p_i	x_{i+1}	y_{i+1}
0	-7	1	8
0	-4	2	8
2	1	3	7
3	-6	4	7
4	3	5	6
5	2	6	5

$(x_i, y_i)=(20,30)$

$(x-20)+(y-30)=64$

Midpoint Circle Drawing Algorithm

Step I – assume that the centre is origin and radius is r. Obtain the first point of the circle by taking $x_0 = 0$ and $y_0 = r$.

Step II – Find the initial decision parameter $P_0 = 5/4 - r$.

Step III – At each x_K position perform the following test

Case 1 – if $P_K < 0$, then the next point is $(x_K + 1, y_K)$

$P_{K+1} = P_K + 2x_K + 3$.

Case 2: If $P_K > 0$

$(x_K + 1, y_K - 1)$

$P_{K+1} = P_K + 2(x_K - y_K) + 5$

Case 3 If $P_K = 0$

4) mirrorpt

5) $y >$

Pseudocode

```
void mid-pt-circle(int r, color)
{
   int x, y
   float d
   x = 0
   y = r
   d = 5/4 - r
    mirrorpt (x, y, color)
   while( y >= x)
   {
    if (d < 0)
      {
       d = d + 2*x + 3;
       x++;
      }
    else
      {
       d = d + 2(x - y) + 5
       x++;
       y--;
      }
   }
```

Bresenham's Circle Drawing Algorithm

Assume that centre is origin and radius is r obtain the first point on the circle.

1) $x_0 = 0, y_0 = r$
2) $P_0 = 3 - 2r$ // Derive
3) At each x_K posn perform the following check

Case 1 If $P_K < 0$
the next point on the circle is (x_K+1, y_K)
Corresponding $\boxed{P_{K+1} = P_K + 4x_K + 6}$ // Derive.

Case 2 If $P_K > 0$
the next point on the circle is (x_K+1, y_K-1)
$P_{K+1} = P_K + 4(x_K - y_K) + 10$ // Derive.

Case 3 If $P_K = 0$
Choose either case 1 or case 2

4) Same as mid-point circle drawing algorithm
5) Repeat step (3) & (4) until $y \geq x$.

Q Using Bresenham's Circle Drawing Algo calculate 1st 5 pixel in first octant, given: centre of circle is (50,60), radius is 10.

here,
centre is (50,60)
radius = 10

So, $x_0 = 0$
$y_0 = 10$

$P_0 = 3 - 2r = 3 - 20 = -17$

i	Pi	xi+1	yi+1
0	-17	1	10
1	-11	2	10
2	3	3	9
3	-11	4	9
4	11	5	8

Computer Graphics

Topic:

Two Dimensional Geometric Transformation

Contributed By:

Verified Writer

8/7/14.

2-D Transformations

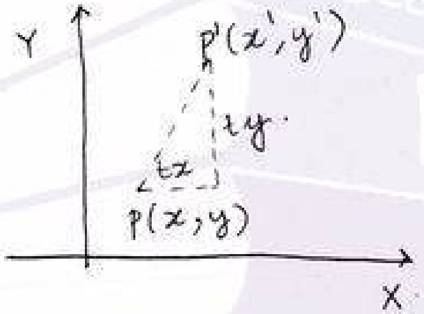
2-D transformation means a change in either position or size or shape or the rotation of any geometrical object like circle, ellipse, rectangle, line polygon, etc

There are five types of transformation

1. Translation
2. Scaling
3. Rotation
4. Reflection
5. Shearing

Translation:-

Translation means changing the position of an object / point from one position to another location.



$$\boxed{\begin{aligned} x' &= x + tx \\ y' &= y + ty \end{aligned}}$$

tx = translation factor along x ~~...~~ axis

ty = " " y axis

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

$$\boxed{[X'] = [X] + [T]}$$

Suppose A (0,8)

B (9,12)

$tx = 6$

$ty = -3$

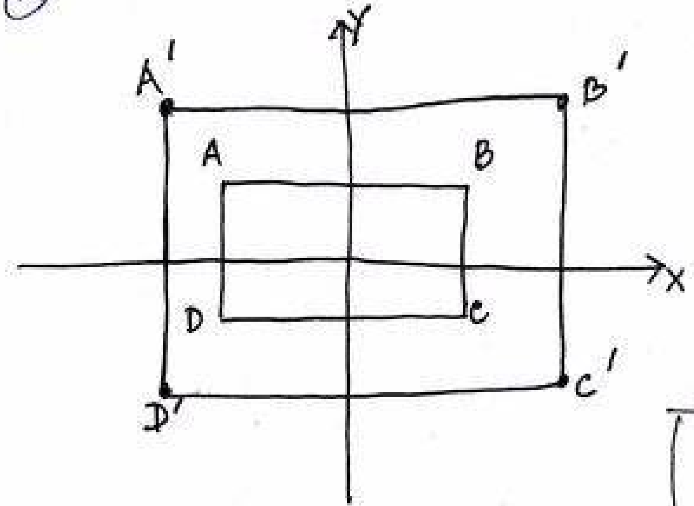
$A' = (6, 5)$

$B' = (15, 9)$

Scaling: Means a transformation that changes the size of an object

Scaling w.r.t Origin

(92)



Scaling w.r.t point

$P(x, y) \rightarrow$ original point

$S_x, S_y \rightarrow$ Scaling factor of X and Y.

$P'(x, y)$ $\rightarrow$ Scaling factor in X-axis

$$\boxed{\begin{aligned} x' &= S_x \cdot x \\ y' &= S_y \cdot y \end{aligned}}$$

Scaling in Matrix Form.

$$\boxed{\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}}$$ — w.r.t ORIGIN

Q A triangle is defined as with co-ordinate point

$$\begin{array}{ccc} A & B & C \\ \end{array}$$
$$\begin{pmatrix} 2 & 4 & 4 \\ 2 & 2 & 4 \end{pmatrix}$$. Scale the triangle twice its size with respect to origin

Find out A', B' and C'

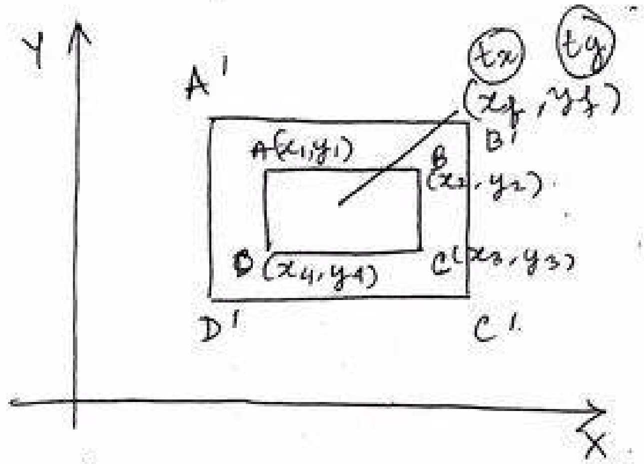
Sol^n here $S_x = 2$ $S_y = 2$

$A' = 2 \cdot 2 = 4$
$\quad\; 2 \cdot 2 = 4$

$B' = 4 \cdot 2 = 8$
$\quad\; 2 \cdot 2 = 4$

$C' = 4 \cdot 2 = 8$
$\quad\; 4 \cdot 2 = 8$

Scaling w.r.t any arbitrary point.



Procedure

1. Translate the fixed point to the origin
2. At origin scale the object
3. Retranslate the scaling object to the original position
4. Find the new position of the scaling object.

$A' = x_1 - x_f$
$\quad y_1 - y_f$

$A'' = \begin{cases}(x_1 - x_f)\,s_x \\ (y_1 - y_f)\,s_y\end{cases}$

$B' = x_2 - x_f$
$\quad y_2 - y_f$

$B'' = (x_2 - x_f)\,s_x$
$\quad (y_2 - y_f)\,s_y$

$C' = x_3 - x_f$
$\quad y_3 - y_f$

$C'' = (x_3 - x_f)\,s_x$
$\quad (y_3 - y_f)\,s_y$

$D' = x_4 - x_f$
$\quad y_4 - y_f$

$D'' = (x_4 - x_f)\,s_x$
$\quad (y_4 - y_f)\,s_y$

New position of the scaling object

$A''' = (x_1 - x_f)\cdot s_x + x_f$
$\quad (y_1 - y_f)\cdot s_y + y_f$

$B''' = (x_2 - x_f)\,s_x + x_f$
$\quad (y_2 - y_f)\,s_y + y_f$

$C''' = (x_3 - x_f)\,s_x + x_f$
$\quad (y_3 - y_f)\,s_y + y_f$

$D''' = (x_4 - x_f)\,s_x + x_f$
$\quad (y_4 - y_f)\,s_y + y_f$

Scaling of a fixed point w.r.t a fixed point

$$\boxed{\begin{aligned} x' &= (x - x_f)s_x + x_f \\ y' &= (y - y_f)s_y + y_f \end{aligned}}$$

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

Q. A triangle is defined for co-ordinate point $\begin{pmatrix} \overset{A}{2} & \overset{B}{4} & \overset{C}{4} \\ 2 & 2 & 4 \end{pmatrix}$ Scale the Δ twice its size w.r.t a fixed point $(5,2)$.

$s_x = 2$ $\quad x_A' = (2-5)2+5 = -1$

$s_y = 2$

$x_f = 5$ $\quad y_A' = (2-2)2+2 = 2$

$y_f = 2$ $\quad x_B' = 5+(4-5)2 = 3$ $\qquad A'\begin{pmatrix} -1 \\ 2 \end{pmatrix}$

$y_B' = 2+(2-2)2 = 2$ $\qquad B\begin{pmatrix} 3 \\ 2 \end{pmatrix}$

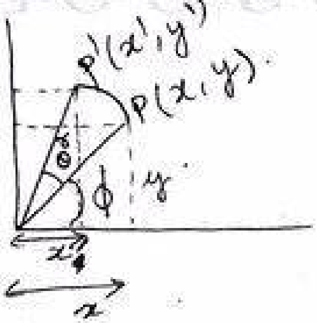
$x_C' = 5+(4-5)2 = 3$

$y_C' = 4+(4-2)2 = 6$ $\qquad C\begin{pmatrix} 3 \\ 6 \end{pmatrix}$

$A' = \begin{pmatrix} -1 \\ 2 \end{pmatrix}$ $\quad B' = \begin{pmatrix} 3 \\ 2 \end{pmatrix}$

$C' = \begin{pmatrix} 3 \\ 6 \end{pmatrix}$

Rotation (Anti-Clockwise Rotation)



$\frac{x}{r} = \cos\phi$ $\qquad$ Total angle $= \theta + \phi$

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

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

$$\boxed{x' = r\cos\phi\cos\theta - r\sin\phi\sin\theta}$$

Across Sh 18 $\frac{1}{30}$ Ch 43.5 Sleeve 28

$$\boxed{x' = x\cos\theta - y\sin\theta}$$

$y' = r\cos\phi \cdot \sin\theta + r\sin\phi \cdot \cos\theta$

$$\boxed{y' = x\sin\theta + y\cos\theta}$$

Write x' & y' in Matrix form.

$$\boxed{\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}} \rightarrow \text{Rotation of a point in ACW direction}$$

Rotation of an Object in CW direction.

Rotate an object 90° about the origin and the object coordinates are

A (3,1)
B (2,1)
C (3,2)
D (2,2)

$x_A' = 2\cos 90^\circ - 1\sin 90^\circ = -1$

$y_A' = 2\sin 90^\circ + 1\cos 90^\circ = 2$

$x_B' = 3\cos 90^\circ - 1\sin 90^\circ = -1$

$y_B' = 3\sin 90^\circ + 1\cos 90^\circ = 3$

$x_C' = 3\cos 90^\circ - 2\sin 90^\circ = -2$

$y_C' = 3\sin 90^\circ + 2\cos 90^\circ = 3$

$x_D' = 2\cos 90^\circ - 2\sin 90^\circ = -2$

$y_D' = 2\sin 90^\circ + 2\cos 90^\circ = 2$

$A' = \begin{pmatrix} -1 \\ 2 \end{pmatrix}$

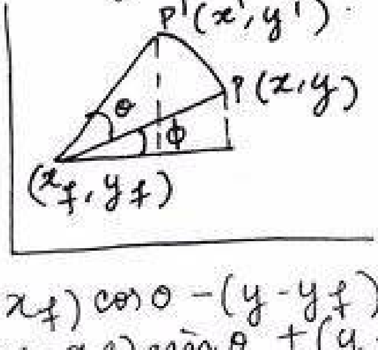
$B' = \begin{pmatrix} -1 \\ 3 \end{pmatrix}$

$C' = \begin{pmatrix} -2 \\ 3 \end{pmatrix}$

$D' = \begin{pmatrix} -2 \\ 2 \end{pmatrix}$

Rotation of an Object About a fixed point

1. Bring the fixed point to the origin
2. At origin rotate the object
3. Retranslate the rotate object to the original pos^n
4. After bringing the fixed point to the origin co-ordinate of the point $P = (x - x_f), (y - y_f)$.



$x' = (x - x_f)\cos\theta - (y - y_f)\sin\theta$

$y' = (x - x_f)\sin\theta + (y - y_f)\cos\theta$

After rotation it is retranslated i.e the co-ordinate pt will be added,

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

Q Previous question but fixed point is (2,1).

$x_A' = (2-2)\cos 90^\circ - (1-1)\sin 90^\circ + x_f = 2$.
$y_A' = (2-2)\sin 90^\circ + (1-1)\cos 90^\circ + y_f = 1$
$x_B' = (3-2)\cos 90^\circ - (1-1)\sin 90^\circ + x_f = 2$.
$y_B' = (3-2)\sin 90^\circ + (1-1)\cos 90^\circ + y_f = 2$.
$x_C' = (3-2)\cos 90^\circ - (2-1)\sin 90^\circ + x_f = 1$.
$y_C' = (3-2)\sin 90^\circ + (2-1)\cos 90^\circ + y_f = 2$.
$x_D' = (2-2)\cos 90^\circ - (2-1)\sin 90^\circ + x_f = 1$
$y_D' = (2-2)\sin 90^\circ + (2-1)\cos 90^\circ + y_f = 1$

$A' = \begin{pmatrix}2\\1\end{pmatrix}$ $B' = \begin{pmatrix}2\\2\end{pmatrix}$ $C' = \begin{pmatrix}1\\2\end{pmatrix}$ and $D' = \begin{pmatrix}1\\1\end{pmatrix}$.

Homogenous Co-ordinate System

To express any 2-D transformation as a matrix multiplication we represent each cartesian co-ordinate $(x, y) = (xh, yh, h)$
where h represents a homogenous parameter which is a non zero value.

Generally for homogenous co-ordinate system $h = 1$, so $(x, y) = (x, y, 1)$

Q $(6, 12, 6) \rightarrow$ homogenous coordinate system
Its corresponding cartesian coordinate is (1, 2).
$(5, 10, 4) \rightarrow (5/4, 10/4)$.

Representation of Translation in Homogenous System

$x' = x + \Delta x$
$y' = y + \Delta y$.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}.$$

cartesian → homogenous

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

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

⇒ equivalent to cartesian co-ordinate system.

Scaling in Homogenous format

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

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

$\Downarrow$

Homogenous format

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

Rotation in homogenous format

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

$\Downarrow$ homogenous

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

⇒ Homogenous Co-ordinate format.

Inverse Transformation

$$T^{-1} = \begin{bmatrix} 1 & 0 & -\Delta x \\ 0 & 1 & -\Delta y \\ 0 & 0 & 1 \end{bmatrix}$$

⇒ Inverse Translation

$$S^{-1} = \begin{bmatrix} \frac{1}{S_x} & 0 & 0 \\ 0 & \frac{1}{S_y} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

→ Inverse Scaling

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

→ Clockwise Rotation

(28)

Q. Dericve the composite transformation matrix for scaling ~~animation~~ object about a fixed point in Homogenous system

$(x_f, y_f) \rightarrow \Delta x \quad \Delta y$

$$T = \underbrace{\begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix}}_{T} \underbrace{\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{S} * \underbrace{\begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix}}_{T^{-1}}$$

$$\boxed{[T_{comp}] = T \cdot S \cdot T^{-1}}$$

$$T_{composite} = \begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 1. \\ 0 & S_y & -S_y y_f \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_{comp} = \begin{bmatrix} S_x & 0 & x_f(1-S_x) \\ 0 & S_y & y_f(1-S_y) \\ 0 & 0 & 1 \end{bmatrix}$$

Q. Consider a triangle ABC, A = (0,0), B(1,1), C(4,2). Magnify the triangle A, B, C twice its size keeping the point C fixed. What is the final co-ordinate of Δ after magnification

A(0,0)

B(1,1)

C(4,2)

$A' = T(comp) * A$

$B' = T(comp) \cdot B$

$C' = T(comp) \cdot C$

$$T_{comp} = \begin{bmatrix} 2 & 0 & -4 \\ 0 & 2 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

C(4,2) fixed :-

$S_x = 2 \quad x_f = 4$

$S_y = 2 \quad y_f = 2$

$$A' = \begin{bmatrix} 2 & 0 & -4 \\ 0 & 2 & -2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -4 \\ -2 \\ 1 \end{bmatrix}$$

$$B' = \begin{bmatrix} 2 & 0 & -4 \\ 0 & 2 & -2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2-4 \\ 2-2 \\ 1 \end{bmatrix} = \begin{bmatrix} -2 \\ 0 \\ 1 \end{bmatrix}$$

$$C = T_{comp} \begin{bmatrix} 4 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 8-4 \\ 4-2 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \\ 1 \end{bmatrix} = [4, 2]$$

Derieve the comp. transformation matrix to rotate an object about a fixed point in homogenous coordinate system (29)

assume anti rotation

$$T_{comp} = \begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_{comp} = \begin{bmatrix} \cos\theta & -\sin\theta & x_f(1-\cos\theta)+y_f\sin\theta \\ \sin\theta & \cos\theta & y_f(1-\cos\theta) - x_f\sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

Q Rotate a triangle ABC with angle 90 about a fixed point (-1, 1) where the co-ordinate of the triangle is given A(5, 0), B(10, 2), C(7, 4).

$A', B', C' = ?$

$x_f = -1,\ y_f = 1,\ \theta = 90^\circ$

$$T_{comp} = \begin{bmatrix} \cos 90 & -\sin 90 & -1(1-\cos\theta)+1\sin 90 \\ \sin 90 & \cos 90 & 1(1-\cos 90)+1\sin 90 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$A' = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 7 \\ 1 \end{bmatrix} = (0, 7)$$

$$B' = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} -2 \\ 12 \\ 1 \end{bmatrix} = (-2, 12)$$

$$C' = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 7 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} -4 \\ 9 \\ 1 \end{bmatrix} = (-4, 9)$$

Q Prove that 2-D rotation & scaling is commutative, if $S_x = S_y$ or $\theta = n\pi$, $n = 1, 2, 3, \ldots$

$x_f(1-\cos\theta)+y_f\sin\theta$
$y_f(1-\cos\theta) - x_f\sin\theta$

16/7/2014.

Prove that 2-D Rotation & Scaling is commutative if $S_x = S_y$ or

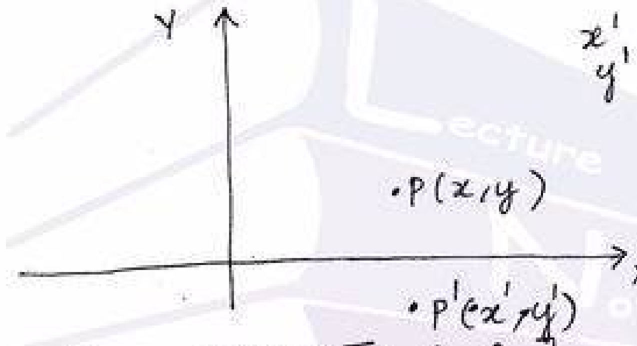
$\theta = n\pi$

$n = 1, 2, 3, \ldots$

Reflection

It is a transformation that produce a mirror image of an object. Reflection also defined as the rotation of an object by $180°$.
In reflection image is of same size but in opposite region.

Reflection of a point about X-axis.



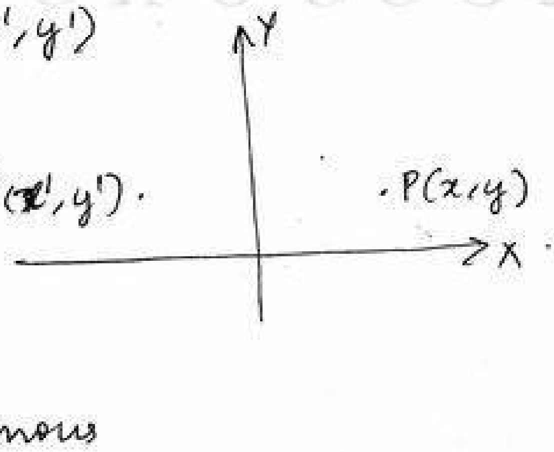
$x' = x$
$y' = -y$.

$$Ref_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \xrightarrow{\text{homogenous}} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Reflection about Y axis

$P(x, y) \longrightarrow P'(x', y')$

where $x' = -x$.
$y' = y$.

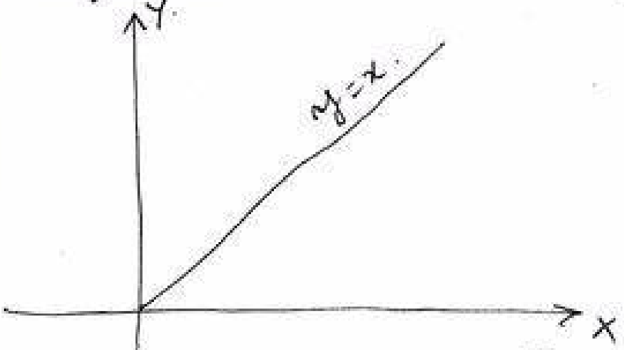


$$Ref_y = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

$\Downarrow$

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$ in homogenous format.

Reflection about a line $y = x$.



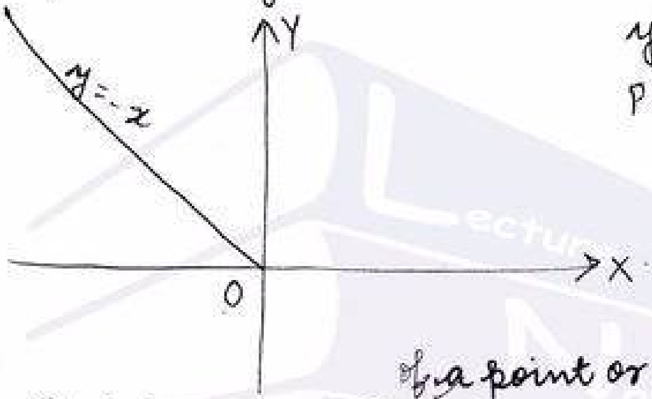
$P(x,y) \downarrow P'(x',y')$

$x' = y$

$y' = x$.

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \xRightarrow{\text{homogenous}} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

if line is $y = -x$.



$y = -x$

$P(x,y) \longrightarrow P(x',y')$.

$x' = -y$

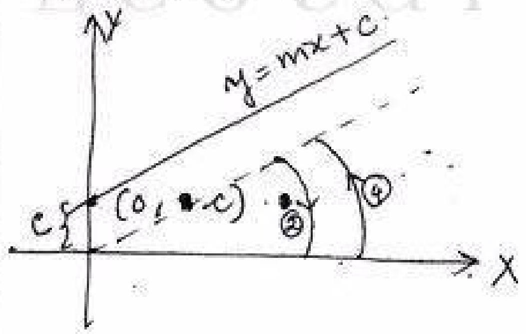
$y' = -x$

$$\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \xRightarrow{\text{homogenous}} \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Reflection of a point or object about any arbitrary line

Derieve the composite matrix to reflect the point/object about any arbitrary line.

Assume:



Procedure:

Step-1 : Bring the line to origin i.e Translate the line to the origin

$x_f = 0$

$y_f = -c$

When line bought to origin $y = mx$

if $y = x$ then $m = 1$

Step 2 : Rotate the line in clockwise direction, so that the line will co-incide with X axes

Step 3 : Reflect the line about X axis

Step 4 : Rotate this line in ACW direction

Step 5 : Retranslate the rotated line to the original position.

32

1. $T^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -c \\ 0 & 0 & 1 \end{bmatrix}$

2. Rotation in CW direc^n.

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

3. Reflect the line about x axis

$Ref_{(x)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

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

5. $T_{forward} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & c \\ 0 & 0 & 1 \end{bmatrix}$

Multiply all five matrix to get the result

$$\boxed{T_{comp} = T \cdot R(\theta)_{acw} \cdot Ref_{x} \cdot R(\theta)_{cw} \cdot T^{-1}}$$

$m = \tan\theta$

$\sin\theta = m\cos\theta = \frac{m}{\sqrt{m^2+1}}$

$\cos\theta = \frac{\sin\theta}{m} = \frac{1}{\sqrt{m^2+1}}$

$\frac{\sin^2\theta}{\cos^2\theta} = m^2$.

$\sin\theta = m\cos\theta$

$\sin^2\theta + \cos^2\theta = 1$

$m^2\cos^2\theta + \cos^2\theta = 1$

$m^2+1 = \frac{1}{\cos^2\theta}$

$\cos\theta = \frac{1}{\sqrt{m^2+1}}$

$$T_{comp} = \begin{bmatrix} \frac{1-m^2}{m^2+1} & \frac{2m}{m^2+1} & \frac{-2cm}{m^2+1} \\ \frac{2m}{m^2+1} & \frac{m^2-1}{m^2+1} & \frac{2c}{m^2+1} \\ 0 & 0 & 1 \end{bmatrix}$$

Q. Reflect point P(12, 21) about Y = -21.

$P(12, 21) \rightarrow P' \begin{pmatrix} 12 \\ -63 \end{pmatrix}$

$Y = -21$

$c = -21$
$m = 0$

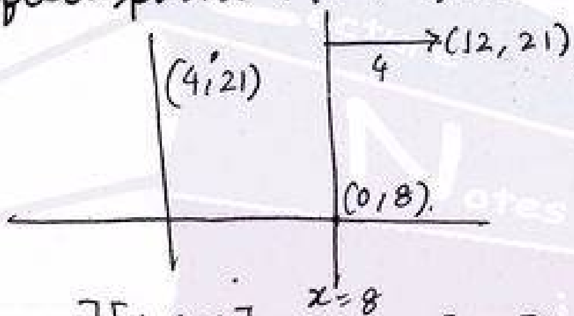
$$T_{comp} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & -42 \\ 0 & 0 & 1 \end{bmatrix}$$

$$P' = T_{comp} * \begin{bmatrix} 12 \\ 21 \\ 1 \end{bmatrix}$$

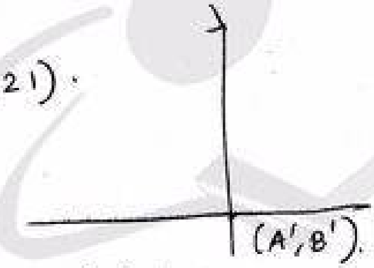
$$\begin{bmatrix} 12 \\ -21-42 \\ 1 \end{bmatrix} = \begin{bmatrix} 12 \\ -63 \\ 1 \end{bmatrix}$$

$$P' = \begin{pmatrix} 12 \\ -63 \end{pmatrix}$$

Q. Reflect point P(12, 21) about X = 8.



$P' = (4, 21)$



$(A', B') = (8, 8)$

$$\begin{bmatrix} 1 & 0 & 8 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -8 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 16 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 8 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -8 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -1 & 0 & 16 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 12 \\ 21 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} -12+16 \\ 21 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 21 \\ 1 \end{bmatrix} \quad P' = \begin{pmatrix} 4 \\ 21 \end{pmatrix}$$

(34) Reflect ΔABC about the line $3x-4y+8=0$ where $A(4,1)$, $B(5,2)$ $C(4,3)$.

$3x-4y+8=0$

$\Rightarrow 4y = 3x+8$

$\Rightarrow y = \frac{3}{4}x + 2$

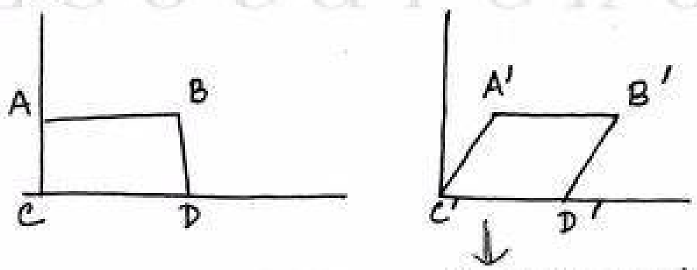
$m = \frac{3}{4}$; $c = 2$

$$T_{comp} = \begin{bmatrix} \dfrac{1-\frac{9}{16}}{\frac{9}{16}+1} & \dfrac{2\times\frac{3}{4}}{\frac{9}{16}+1} & \dfrac{-2\times 2+\frac{3}{4}}{\frac{9}{16}+1} \\ \dfrac{2\times\frac{3}{4}}{\frac{9}{16}+1} & \dfrac{\frac{9}{16}-1}{\frac{9}{16}+1} & \dfrac{2\times 2}{\frac{9}{16}+1} \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_{comp} = \begin{bmatrix} \frac{7}{25} & \frac{24}{25} \\ & \\ & \end{bmatrix}$$

Shearing Transformation

It is a transformation that produce a shape distortion



Shearing along x-axis

$P(x,y) \xrightarrow{\text{shearing}} P'(x',y')$ (Shx, y)

$x' = x +$ shearing factor along y axis

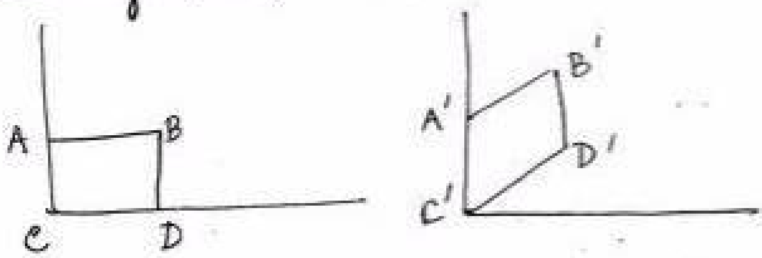
$y' = y$.

$$\boxed{\begin{aligned} x' &= x + Shx\, y \\ y' &= y \end{aligned}}$$

Shearing Matrix along X-axis

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

Shearing along Y axis



$$\boxed{\begin{aligned} x' &= x \\ y' &= y + shy \cdot x \end{aligned}}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ shy & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$ Shearing matrix along Y direction

Computer Graphics

Topic:

Line Clipping And Polygon Clipping

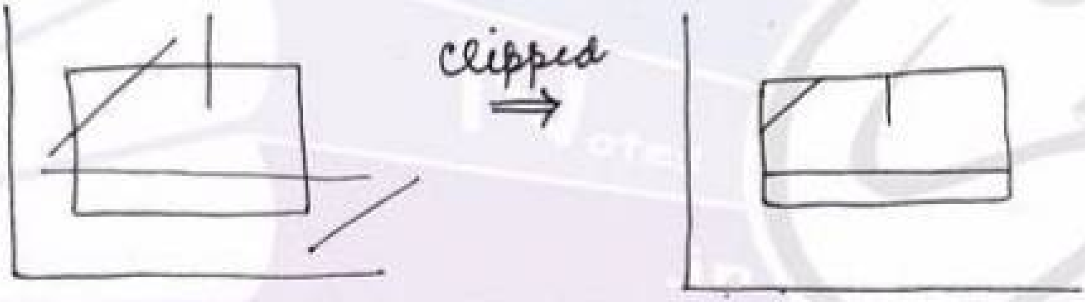
Contributed By:

Verified Writer

Ch-3

Clipping

In Clipping algorithm, a clipping window is given



Defn: any procedure that identifies the visible part of picture which is either inside / outside the clipping window is known as Clipping Procedure / Clipping Algorithm.

Different types of Clipping are:

1. Point Clipping
2. Line Clipping
3. Polygon Clipping

To draw clipping window we require two diagonal points (x_{mean}, y_{mean}) & (x_{max}, y_{max}).

1. Point Clipping

In this algorithm we have to test whether a point is visible inside the clipping window / not if the following condn is satisfied.

$$x_{mian} \le x \le x_{max}$$
$$y_{mian} \le y \le y_{max}$$

(2) Line Clipping Algorithm

Cohen-Sutherland line Clipping Algorithm

R1	R2	R3
R4	R5	R6
R7	R8	R9

amp:

T	B	R	L
1	0	0	1

bit code of R1 Region

Top, Bottom, Right, Left

Similarly find bit code for R2 - R9.

1 0 0 0 – R2.

1 0 1 0 – R3

0 0 0 1 → R4

0 0 0 0 – R5

0 0 1 0 – R6

0 1 0 1 – R7

0 1 0 0 – R8

0 1 1 0 – R9.

Alternate Method to get bitcode for every region

Top : T = 1
 If $y > y_{max}$
 else 0

Bottom : B = 1
 If $y < y_{min}$
 else 0.

Right : R = 1
 If $x > x_{max}$
 else 0.

Left : L = 1
 If $x < x_{min}$.
 else 0

Q

$x_{min} = 20$ P(40, 80).
$y_{min} = 20$ Calculate 4 bit code.
$X_{max} = 60$
$Y_{max} = 40$.
T = 1
B = 0
R = 0
L = 0

1	0	0	0

Apply setep 1, compute the four bit code for the two end points of the line.
After find the four bit code of the two end point of given line, if four bit code is (for both)
0000 ⇒ line is inside the clipping window
So line is trivially accepted and the clipping procedure is not reqd.
(3) If one end end point has bit ~~point~~ code is 0000 and other end point is not 0000, this means some portion of line is visible inside the clipping window & some portion isn't visible. So we'll go for clipping.

(38)

(4) If bit code of both end point of the line is not 0000 then compute logical AND operation of the both bitcode

If the result isn't 0000 then that means the line is totally outside the clipping window. So trivially reject the line

• Else neither trivially accepted / trivially rejected in this case we go for clipping

(5) Calculation of left edge intersection point

Intersectpoint point is (x_{min}, y)

$$\boxed{y = slope(x_{min} - x_1) + y_1}$$

22/7/14

Right edge intersection point

(X_{max}, Y)

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

Top Edge - Intersection Point

(x, y_{max})

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

Bottom Edge - Intersection Point

(x, y_{min})

$$x = x_1 + \frac{1}{m}(y_{min} - y_1)$$

(6) After finding all intersection points compare with point - clipping algo, find out which pair of intersection point satisfy point clipping algorithm

(7) Take that pair of intersection point, draw the line in clipping window which is visible

Ex: Apply Cohen-Sutherland line clipping algorithm. (39)
Clip the line P_1, P_2 with $P_1(10,30)$, $P_2(80,90)$ against the clipping window $A(20,20)$, $B(90,20)$, $C(90,70)$, $D(20,70)$. Find the visible portion inside the clipping window.

$x_{max} = 90$

$y_{max} = 70$

$x_{min} = 20$

$y_{min} = 20$.

find 4 bit code for two end point of the line

$P_1 P_2$ line

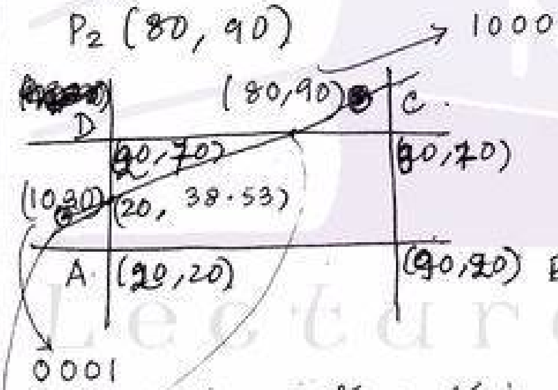
Bit code $P_1 \rightarrow 0001$
$\qquad\qquad P_2 \rightarrow 1000$

logical AND $\quad 0000$

In this case we go for Clipping

$P_1(10,30)$

$P_2(80,90)$



So find top edge intersection point & left edge intersection point

left edge intersection point.

(x_{min}, y) $\quad y = \frac{6}{7}(20-10) + 30$

$m = \frac{6}{7}$ $\qquad = \frac{60}{7} + 30$

$\qquad = 38.57$

$(20, 38.57)$ satisfies point clipping algo as $38.57 < y_{max}$
it selected.

right edge intersection point

$(x_{max}, y) = (90, y)$

$y = \frac{6}{7}(90- \quad) + 30$

$\quad = 98.57 > y_{max}$

So this point is outside clipping window. So this point is rejected.

(4) top edge intersection point

(x, y_{max})

$x = 10 + \frac{7}{6}(70 - 30)$

$= 10 + \frac{\cancel{280}}{6}\ 46.6$

$= 56.6 < x_{max}$

So this point is selected.

~~left~~ bottom edge intersection point

(x, y_{min})

$x = 10 + \frac{7}{6}(20 - 30)$

$= 10 - \frac{70}{6}$

$= -\frac{10}{6}$

No selected.

Q.

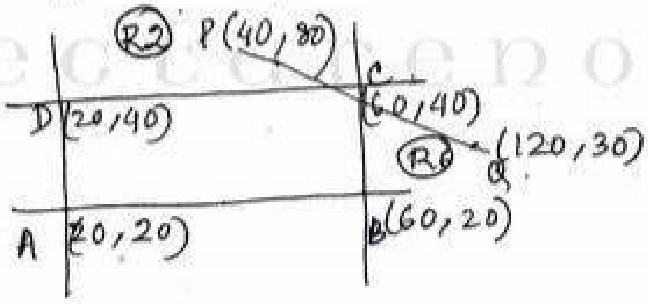
A (20, 20) P (40, 80)

B (60, 20) Q (120, 30)

C (60, 40)

D (20, 40)

Using Cohen-Sutherland Algorithm, find the visible portion of line PQ in clipping window.



$x_{max} = 60$

$y_{max} = 40$

$x_{min} = 20$

$y_{min} = 20$

bit code ~~for~~ 1000 → P

0010 → Q

perform logical and operation

$$\begin{array}{r} 1000 \\ 0010 \\ \hline 0000 \\ \hline \end{array}$$

So go for clipping

$m = \frac{30-80}{120-40} = \frac{-50}{80} = \frac{-5}{8}$

top edge intersection point

$(x, y_{max}) = (x, 40)$

$x = 40 - \frac{5}{8}(40-80)$

$= 40 - \frac{5}{8}(-40)$

$= 40 + 25$

$= 65 > x_{max}$

(65, 40) Reject this point.

right edge intersection point

(x_{max}, y)

$(60, y)$

$y = \frac{-5}{8}(60-40) + 80$

$= \frac{-5}{8} \times 20 + 80$

$= \frac{-100}{8} + 80 \Rightarrow \frac{-25+160}{2}$

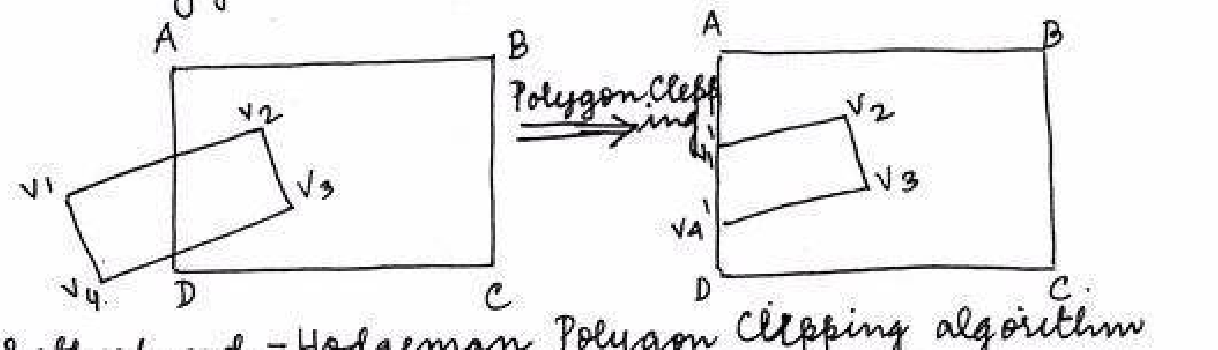
$\Rightarrow \frac{135}{2} = 67.5$

$y > y_{max}$

Reject this point. Similarly other two intersection points are also rejected. So this line lies complety outside clipping window. Trivially Reject this line.

23/7/14

Polygon Clipping



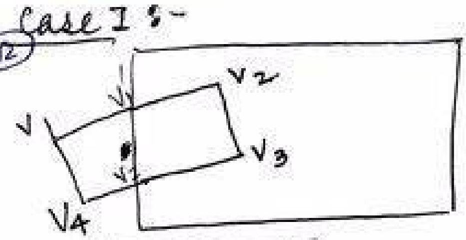
Sutherland - Hodgeman Polygon Clipping algorithm

This algorithm is useful for only clipping of convex type polygon (Convex Polygon → if you draw a line and the line is inside polygon. Concave Polygon - if you draw the line some portion of line is outside the polygon) ← convex ← concave

For clipping a polygon using clipping window consider the following cases:

42

Case I :-

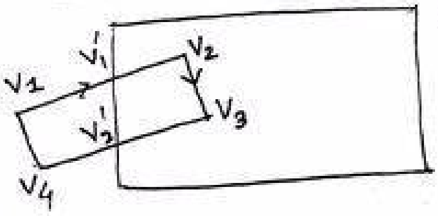


$\boxed{out \rightarrow in}$

Assume that you're scanning the polygon edges in clockwise direction

Set intersection point as V_1' and V_2'

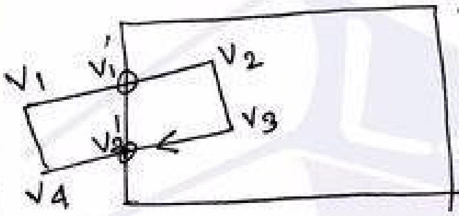
Case II :-



Assume that you're scanning from $\boxed{in \rightarrow in}$

$\boxed{V_3}$ we save the vertex V_3 moving $\boxed{in \rightarrow in}$

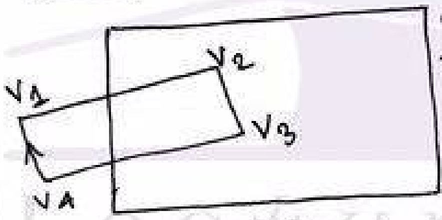
Case III



We are scanning the polygon edges from $\boxed{in \rightarrow out}$

Intersection point $\boxed{V_3'}$ is saved.

Case IV



We are scanning the polygon edges from $\boxed{out \rightarrow out}$

Save none.

After ~~all the~~ saving all the point join all the points to get the desired polygon (visible part of the polygon).

Step ① Read the coordinate of the vertices of the given polygon & the clipping window

Step ② Consider an edge of the clipping polygon & compare the vertices of each edge of the given polygon and record the intersection point.

Step ③ Store the new intersection point and new list of vertices using all the four case described above

Step ④ Repeat Step ② & ③ for other edges of the clipping window

Step ⑤ Using new set of intersection point & new vertices construct the reqd. polygon which is visible inside the clipping window.

This algorithm is not suitable for concave type polygon. There is another algorithm for ~~convex~~ concave type polygon polygon called as Weiler - Autherton Polygon Clipping.

43

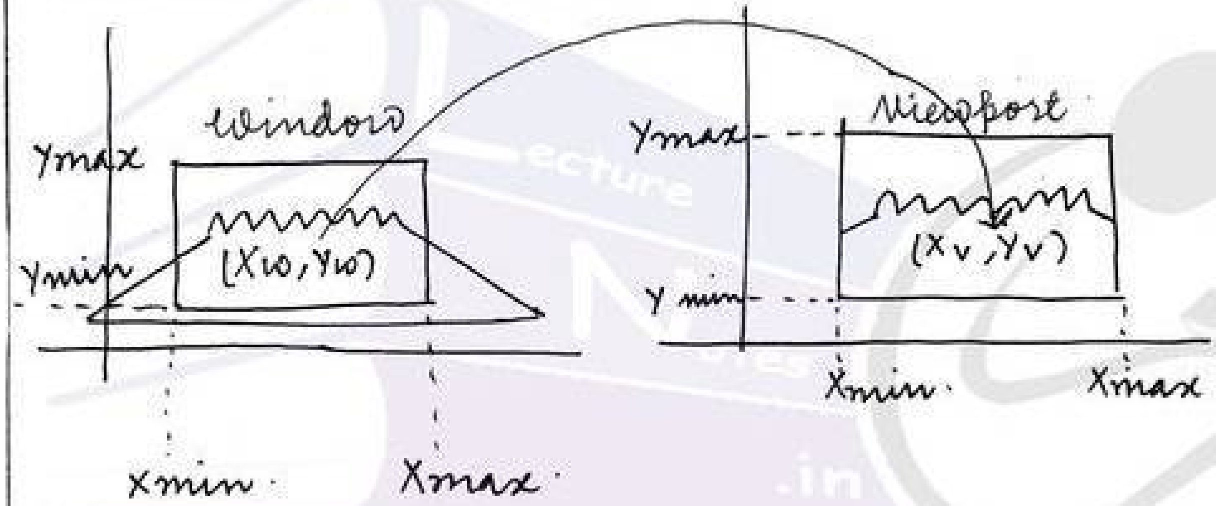
(Module I) →

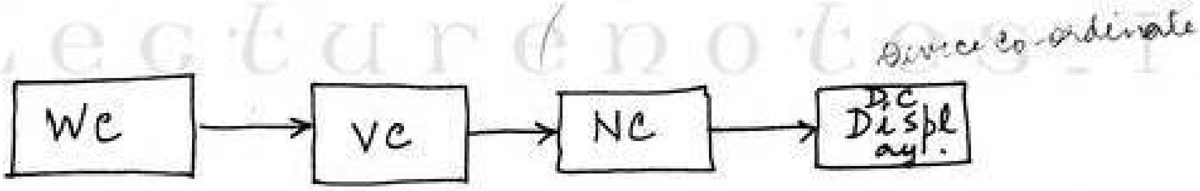
2-D Viewing

Converting

World Co-ordinate System.

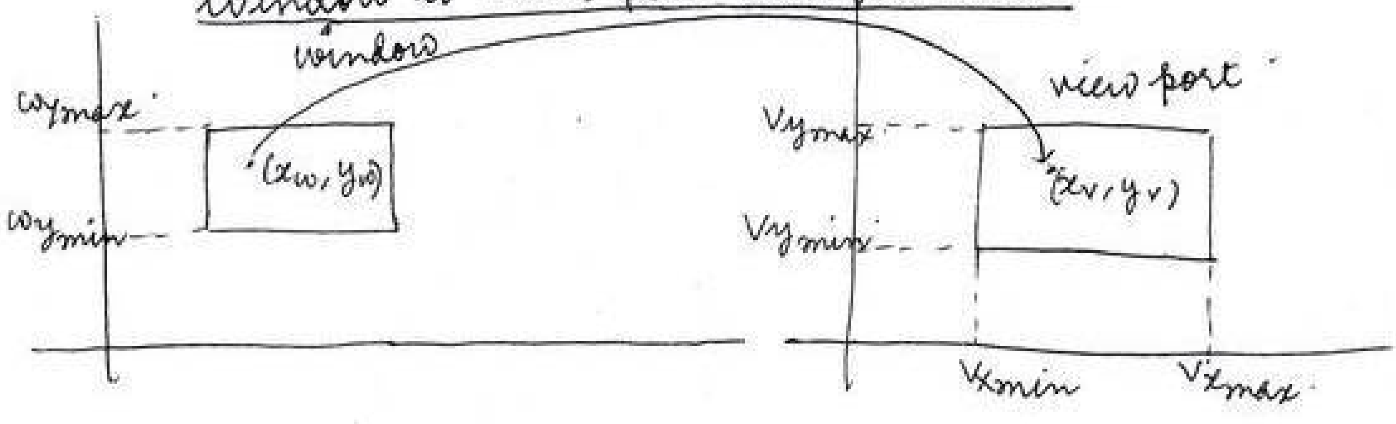
A world co-ordinate is selected for displaying the window
The area on a display device to which a window is mapped
This area is known as ViewPort (rectangular)
In general mapping from window to viewport co-ordinate is known as Window to View Port Transformation.





This pipeline to convert world-coordinate → view co-ordinate → normalized co-ordinate system → Display co-ordinate is called 2-D Viewing Pipeline

Window to view port Transformation.



(44) 28/7/14

Let us take a point (x_w, y_w) inside the window and this point has to map to a co-ordinate point (x_v, y_v) inside the view port.

To maintain the same relative placement in the viewport as in the window, the following equation are varying.

$$\frac{X_v - X_{vmin}}{X_{vmax} - X_{vmin}} = \frac{X_w - X_{wmin}}{X_{wmax} - X_{min}} \quad \text{— (1)}$$

$$\frac{Y_v - Y_{vmin}}{Y_{vmax} - Y_{vmin}} = \frac{Y_w - Y_{wmin}}{Y_{wmax} - Y_{wmin}} \quad \text{— (2)}$$

$$x_v = X_{vmin} + (X_w - X_{wmin}) S_x$$

$$S_x = \frac{X_{vmax} - X_{vmin}}{X_{wmax} - X_{wmin}}$$

$$Y_v = Y_{vmin} + (Y_w - Y_{wmin}) S_y$$

$$S_y = \frac{Y_{vmax} - Y_{vmin}}{Y_{wmax} - Y_{wmin}}$$

Computer Graphics

Topic:

Polygon Filling Seed Fill Scan Line Algorithm

Contributed By:

Verified Writer

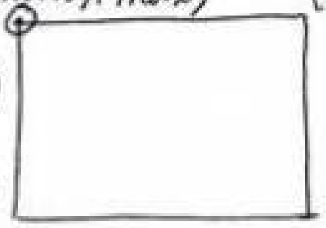
Polygon filling Algorithm

putpixel (x, y, color) - It is a function used to fill particular region with particular color.

Q. Write a code to fill this ~~margin~~ region with a colour say red.

(Xmin, Ymin)
(Xmax, Ymax)

(Xmin, Ymax)



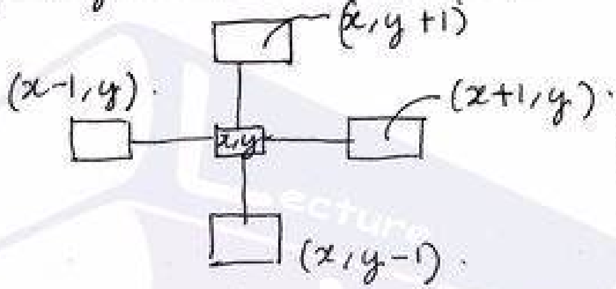
```
for (y = Ymax; y >= Ymin; y--)
{
    for (x = Xmin; x <= Xmax;
                      x++)
    {
        putpixel (x, y, RED)
    }
}

for (y = Ymax ; y >= Ymin ; y --)
{
    for (x = Xmax ; x >= Xmin ; x--)
        putpixel ( x, y, RED)
}
```

```
for (y = Ymin; y ≤ Ymax; y++)
{ for (x = Xmin; x ≤ Xmax; x++)
    putpixel (x, y, RED);
}
for (Y = Ymin ; Y ≤ ~~Xmin~~
```

Seed Filling Algorithm

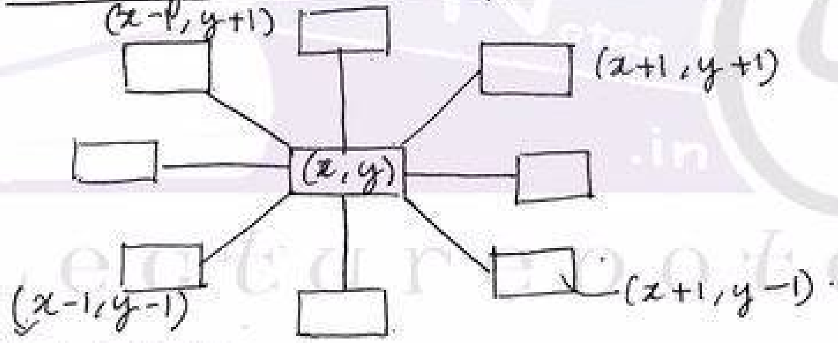
Seed refers to an internal point of the given polygon.
Seed filling algorithm is divided into two parts
(i) Boundary-fill algorithm
(ii) Flood fill algorithm

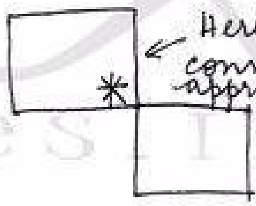
4-way Connected Approach.



One internal point is given and using this internal point, we have to color all four neighbouring points.

8-way Connected approach.





Here 4-way connected approach will not work

(x-1, y) (x+1, y)
(x, y-1) (x, y+1)
(x-1, y-1), (x+1, y+1)
(x-1, y+1), (x+1, y-1).

Boundary-fill algorithm (4-way connected approach)

Step 1:- Start a point inside the region. Assume that the edges of the region are coloured with same colour.

Step 2:- Suppose you start a pixel inside the polygon, then we colour that pixel and all its surrounding pixels or points until we meet a a pixel that is already coloured.

96

Boundary Fill Algorithm (4-way connected approach)

~~Step 1 - Start a point inside the region. Assume that the edges of the region are colourid with same color.~~
~~Step - 2 - Suppose you start a pixel inside the polygon the~~

void - Boundary - fill (int x, int y, int newcolor, int edge color)

{

int current ← current color of the given co-ordinate point

current = getpixel (x, y) // It is a predefined function, will return the color of (x, y).

If (current != edgecolor and current != newcolor)

{

putpixel (x, y, newcolor)
Boundary - fill (x+1, y, newcolor, edgecolor);
Boundary - fill (x-1, y, newcolor, edgecolor);
Boundary - fill (x, y-1, newcolor, edgecolor).

30/7/2014 **Flood Fill Algorithm**

- Suppose we want to color the entire area whose original color is interior color, and replace it with fillcolor
- Then we start with a point in this area, and then color all surrounding points until we see a pixel that is not interior color.
- used when an area defined with multiple color boundaries
- start at a point inside a region
- replace a specified interior color (old color) with fill color
- fill the 4-connected or 8-connected region until all interior points being replaced.

Differentiate b/w Flood-Fill algorithm and boundary-fill algorithm.

Boundary - fill algorithm	Flood Fill algorithm
① The area is defined with a single color boundary	① The area is defined with multicolor boundaries
② The interior points are replaced by a new color	② The interior is coloured with any color
③ The interior points are replaced by a new color	③ The ~~old~~ color is replaced with a new colour.

Flood Fill algorithm (using 4-way connected approach)

void FloodFill4 (int x, int y, int newcolor, int oldcolor)
{ if (getpixel (x, y) == old color)
{ putpixel (x, y, newcolor);
FloodFill4 (x+1, y, newcolor, oldcolor);
FloodFill4 (x-1, y, newcolor, oldcolor);
FloodFill4 (x, y+1, newcolor, oldcolor); ←

30/7/17

Flood Fill Algorithm

(97)

- Suppose we want to colour the entire area whose original colour is interior Color and replace it with fillColor
- Then we start with a point in this area and then colour all surrounding points until we see a pixel that is interiorColor.
- Used when an area defined with multiple color boundaries.
- Start at a point inside a region.
- Replace a specified interior color (old color) will fill color. (new ~~old~~ color)
- Fill the 4-connected or 8-connected region until all interior points being replaced.

Differentiate b/w flood fill & boundary fill algorithm

In case of boundary fill algo, area is defined with a single colour boundary whereas in flood fill it is defined with multi-colour boundary

Boundary fill – interior is not filled with any colour.
flood fill – interior is filled with any color
→ Interior points are ~~filled~~ replaced with new colour.
→ Old colour is replaced by new colour.

Using 4 way connected approach

```
void FloodFill (int x, int y, int newcolor, int oldColor)
if(getpixel(x, y) == oldColor)
{
    putpixel (x, y, newcolor);
    FloodFill (x+1, y, newcolor, oldcolor);
    FloodFill (x-1 , y, newcolor, oldcolor);
    FloodFill (x, y+1, newcolor, oldcolor);
    FloodFill (x, y-1, newcolor, oldcolor);
}
}
```

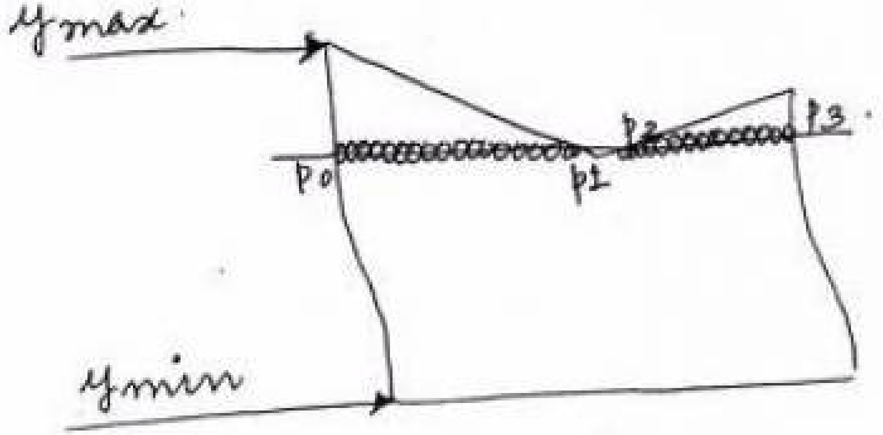
add 4 more points for 8-connected approach
i.e for points (x+1, y+1), (x+1, y-1), (x-1, y-1), (x-1, y+1).

Scanline Fill Algorithm

- Intersect scanline with polygon edges
- Fill b/w pairs of intersections
- Basic algorithms

For $y = y_{min}$ to y_{max}

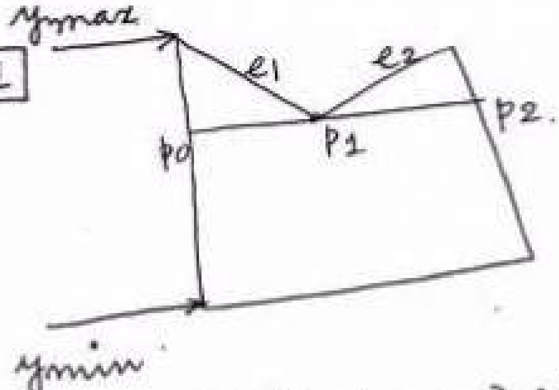
1) intersect scanline y with each edge
2) sort intersections by increasing x [p_0, p_1, p_2, p_3]
3) fill pairwise ($p_0 \rightarrow p_1$, $p_2 \rightarrow p_3$, ...).



Special Cases of Scanline Polygon Fill Algorithm

- Intersection is an edge end point

Case 1

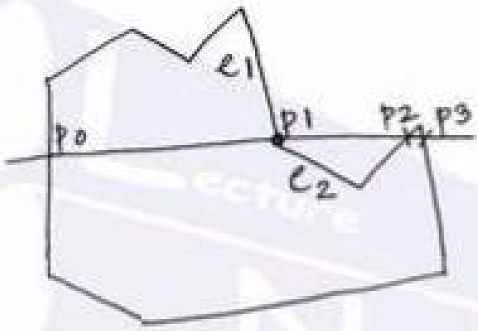


Intersection point (p0, p1, p2) ???

→ (p0, p1, p1, p2) so we can fill pairwise

→ In fact if we compute the intersection of the scanline with edge e_1 and e_2 seperatily, we will get the intersection point p1 twice. Keep both of the p1.

Case 2)



However in this case we dont want to count p1 twice (p0, p1, p1, p2, p3), otherwise we will fill pixels b/w p1 & p2 which is wrong.

Summary : If intersection is ymin of edges intersection point, count it, else dont.

Aliasing & Anti Aliasing

- In CG, term aliasing refer to any unwanted visual artefacts
- The undesirable effects in graphics, images or objects are known as aliasing
- Aliasing is common in CR because screen or file resolution are finite whereas the mathematical model describing an image have infinite resolution.
- Aliasing produces the staircase / jaggies like appearance in image
- Aliasing ~~produces~~ refers to the degradation of a sound, image or other signal during the sampling process due to low resolution sampling.

5) Produces distortion in image when representing a high resolution signal at a lower ~~so~~ resolution

Anti Aliasing

techniques to minimize the aliasing effects.
Anti-aliasing is a method to reduces the staircase or jaggies structure in the image.

Types of anti-aliasing techniques

** Super Sampling
Multi Sampling.

Computer Graphics

Topic:

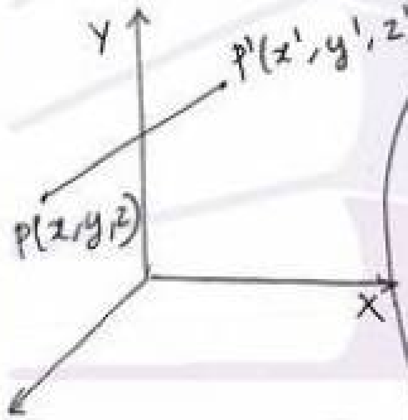
Three Dimensional Geometric And Modeling Transformations

Contributed By:

Verified Writer

3-D Transformation

Translation:



$$\begin{cases} x' = x + \Delta x \\ y' = y + \Delta y \\ z' = z + \Delta z \end{cases}$$

$\Delta x, \Delta y, \Delta z$ - translation factor along x, y, z axes respectively

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} + \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

In homogenous coordinate format

$$\Rightarrow \boxed{\begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}}$$

$\Downarrow$

Translation in homogenous format in 3-D.

Scaling:-

$$\begin{aligned} x' &= x \cdot s_x \\ y' &= y \cdot s_y \\ z' &= z \cdot s_z \end{aligned} \Rightarrow \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

how to represent this in homogenous format?

scaling about origin

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Derieve the transformation matrix for scaling an object about an arbitrary point

$x_f \; y_f \; z_f$

$S_x \; S_y \; S_z$

Cartesian Format:

$x' = x_f + (x - x_f) S_x$

$y' = y_f + (y - y_f) S_y$

$z' = z_f + (z - z_f) S_f$.

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} 1-S_x & 0 & 0 \\ 0 & 1-S_y & 0 \\ 0 & 0 & 1-S_z \end{bmatrix} \begin{bmatrix} x_f \\ y_f \\ z_f \end{bmatrix}$$

Homogenous Format.

Steps

1. Bring a fixed point to the origin 2. at origin scale the object
3. After scaling retranslate the object into original position

$$T_{(cmp)}^{xx} = \begin{bmatrix} 1 & 0 & 0 & x_f \\ 0 & 1 & 0 & y_f \\ 0 & 0 & 1 & z_f \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_f \\ 0 & 1 & 0 & -y_f \\ 0 & 0 & 1 & -z_f \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = T \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotation in 3-D.

Rotation about z-axis.

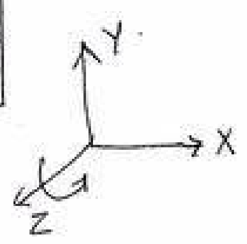
If the rotation is carried out about z axis, then the z co-ordinate remain unchanged because rotation occurs per.pendicular to z axis, while x & y coordinate exactly same as 2-D rotation

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

In matrix ~~form~~ $\longrightarrow$

Cartesian Co-ordinate format

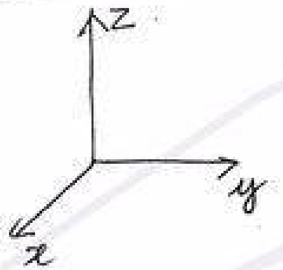
$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$



(52) In homogenous format

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

Rotation about X axis



$x' = x$

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

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

Cartesian

$x' = x$

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

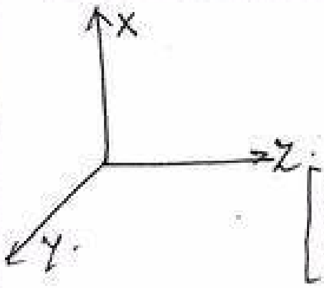
$z' = y\sin\theta + z\cos\theta$

Homogenous format

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow \text{Matrix form}$$

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

Rotation About Y Axis



$y' = y$

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

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

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

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

5/8/14 Find the transformation matrix that is used to perform the following transformation

(a) (i) Translate 4 unit in X direction,
" 2 units in negative of Y direction & 3 unit in z direction

(b) (ii) Scale by 3 units in all dimension

(c) (iii) Rotate 90° around z axis.

(a) Translate 4 units in x direction, 2 units in negative Y direction & 3 units in z direction

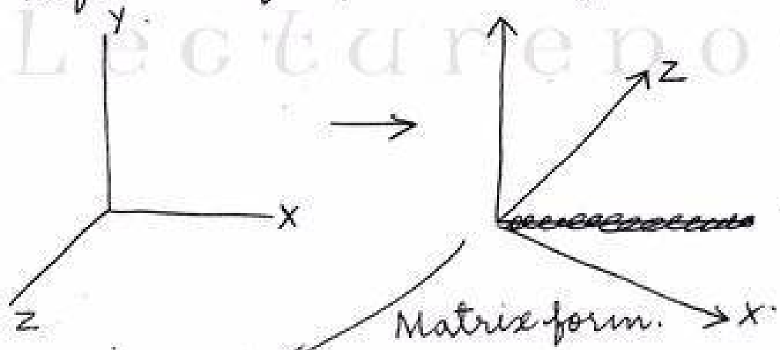
$$\begin{bmatrix} 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(b)
$$\begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(c)
$$\begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3-D Reflection

Reflection of a point $P(x, y, z)$ w.r.t x-y plane



Matrix form.

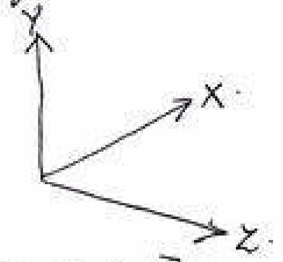
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Reflection about z axis, x, y coordinate are same, z becomes negative

-1 3 -3 1
3 -6 3 0
-3 3 0 0
1 0 0 0

-1 3 -3 1
3 -6 3 0
-3 0 3 0
1 4 1 0

(5) Reflection ~~about a~~ of a point about y-z plane



$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \Rightarrow \text{Transformation matrix}$$

Reflection of a point about x-z plane

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

**

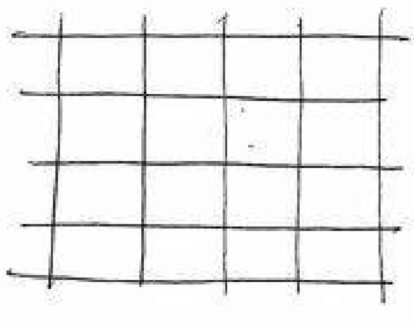
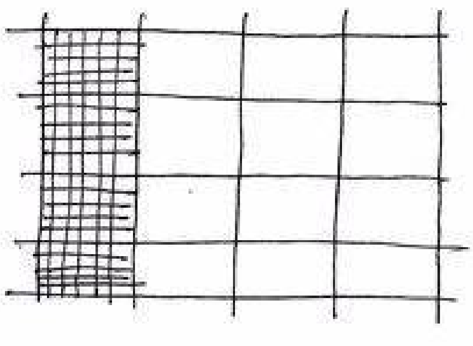
Super Sampling

It is a method of anti-aliasing by taking corner of each pixel and creating the average color, then that pixel is displayed in the screen.

Method:

1. Split single pixel into sub-pixel
2. Sample at the middle of each sub-pixel
3. pixels color is the average of the sub-pixels color.

• Pixel's final color is a mixture of sub-pixels colors. Simple method: sample at the middle of each sub-pixel. Then pixel color is the avg. of sub pixels color.

Multisampling

- takes multiple samples for each pixel
- With multisampling each cell ~~ex~~ has two samples in it
- You can see that the other 3 samples are taken from neighbouring cells.
- So multisampling takes into account the colors around the pixel.
- This gets you a blend of color to achieve the desired result
- It only occurs when a cell is covered by more than one color; otherwise a single color is chosen and value is not calculated.

Sub-pixel Weighting Masks

Instead of considering each pixel to be of equal importance assign a weight to each pixel
Usually consider the center pixel to be most important

1	2	2	1
2	3	3	2
2	3	3	2
1	2	2	1

Example weight for each pixel

Total wt. = 32

Final color of pixel = Sum of each (sub pixel color × sub pixel wt) / Total weight

Computer Graphics

Topic:

Bezier Curves And B-Spline Curves

Contributed By:

Verified Writer

Curve-Design

Divide the curve into several pieces.
Write mathematical formula to derieve each curve piece & then join them

A Spline Curve → is a flexible strip that is used to produce a smooth curve through a set of points. Mathematically this is defined as a piecewise cubic polynomial function whose first & second derievative are continous accross the various point of the curve section. A spline curve is designed by a set of coordinate points known as control points. When the curve passes through all the control point, the resulting curve is known as interspolating spline curve. When curve is passed through some of the control points the resulting curve is known as approximation spline curve. ✓

Techniques to Design the Curve.

(i) Bezier Curve } In syllabus
(ii) B-Spline Curve }
(iii) Hermite Curve

Bezier Curve

Given a set of control points $p_0, p_1, p_2, \dots p_n$ ((n+1) control points). A parametric bezier curve is defined as $P(t) = \sum_{i=0}^{n} P_i B_{in}(t)$

where P_i is a set of control points and $B_{in}(t)$ is known as Blending function of the Bezier curve.

Value of $\boxed{B_{in}(t) = C(n,i)\, t^i \cdot (1-t)^{n-i}}$

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

* Blending function also known as Benstein Function / Benstien Constant

$$\boxed{P(t) = P_0 B_{0,n}(t) P_1 B_{1,n}(t) + \cdots P_n B_n(t)}$$

For cubic polynomial function put $n = 3$.

$$P(t) = P_0 B_{0,3}(t) + P_1 B_{1,3}(t) + P_2 B_{2,3}(t) + P_3 B_{3,3}(t)$$

$$B_{0,3}(t) = (1-t)^3$$

Parametric Continuity cond^n

To ensure a smooth transition from one section of a piecewise parametric spline to the next curve we can impose various continuity cond^n at the connection point. Each section of a spline curve is described with a set of parametric ~~co~~-ordinate function

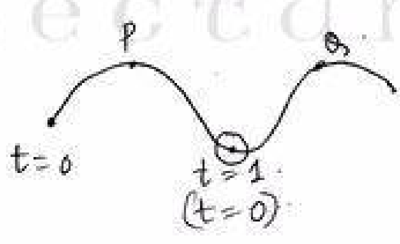
$x = x(u)$

$y = y(u)$ $\quad u_1 \le u \le u_2$.

$z = z(u)$

We define the parametric continuity by matching the parametric derivatives of two successive ~~sectwo~~ curvesection at their common boundary.

Zero Order Parametric Continuity (C^0)



$P(t=1) = Q(t=0)$ ~~Q(t=0)~~ → If cond^n satisfied satisfying it is zero-order parametric continuit

First Order Parametric Continuity (C^1)

$P'(t=1) = Q'(t=0)$

P' & Q' are first order derievative

Second Order Parametric Continuity Cond^n (C^2)

It means both first & second order derievative of two curve section are same at the intersection point $\boxed{P''(t=1) = Q''(t=0)}$

(56)

Geometric Continuity Cond^n.

Zero Order Geometric Continuity (G^0).

It is same as C^0, i.e two successive curve section must have same co-ordinate pos^n at their boundary point $\boxed{P(t=1) = Q(t=0)}$

First Order Geometric Continuity (G^1)

~~Partial~~ Parametric first derivative are proportional to an intersection of two curve section. If P, Q are two piece of curve $P'(t)$ and $Q'(t)$ must have same direction of tangent vector but magnitude is not the same.

$$P'(t) \neq Q'(t) \quad G^1 \not\Rightarrow C^1$$

$$C^1 \Rightarrow G^1$$

Second Order Geometric Continuity Cond^n (G^2)

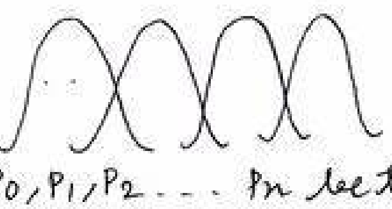
Both first & second derivative are proportional at their boundary point and both tangent and magnitude are same.

$$G^2 \Rightarrow C^2$$

$$C^2 \not\Rightarrow G^2$$

B Spline Curve

↳ Bell shaped



Let $P_0, P_1, P_2 \ldots P_n$ be the set of control points. The B Spline curve is defined as $P(t) = \sum_{i=0}^{n} P_i B_{i,d}(t)$.

$$\boxed{\begin{array}{l} t_{min} \leq t \leq t_{max} \\ 2 \leq d \leq n-1 \end{array}}$$

$B_{i,d}(t)$ are called Blending function of B-Spline Curve.

$$\boxed{B_{i,d}(t) = \left[\frac{t - t_i}{t_{i+d-1} - t_i}\right] B_{i,d-1}(t) + \left[\frac{t_{i+d} - t}{t_{i+d} - t_{i+1}}\right] B_{i+1,d-1}(t)}$$

Cox - deBoor Recursive Formula.

Uniform Cubic B-Spline Curve.

Defined as $\boxed{P(t) = \sum_{i=0}^{3} P_i B_j^3(t)}$

($n=3$, as it is cubic)

$\Downarrow$

$$\boxed{P(t) = P_0 B_0^3(t) + P_1 B_1^3(t) + P_2 B_2^3(t) + P_3 B_3^3(t)}$$

$$B_0^3(t) = \frac{1}{6}(1-t)^3$$

$$B_1^3(t) = \frac{1}{6}(3t^3 - 6t^2 + 4).$$

$$B_2^3(t) = \frac{1}{6}(-3t^3 + 3t^2 + 3t + 1)$$

$$B_3^3(t) = \frac{1}{6}t^3$$

$$P(t) = \frac{1}{6}\begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

$$\begin{matrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & -3 & 0 \\ 1 & 1 & 1 & 0 \end{matrix}$$

~~Fractal Geometry~~ Properties of B-spline Curve:-

The polynomial curve has degree $K-1$ and C^{K-2} continuity over the range of u described by $n+1$ blending function for $n+1$ control points. The sum of the B-spline basis funcⁿ for any parametric value of u is 1.

The basis is parametric for all each basis function is positive or zero for all parametric value i.e $N_{i,K} \geq 0$

for $K=1$ each basis function has 1 maximum value

Maximum order of the curve is equal to no. of defining polygon vertices

Curve generally depends on the defining shape of the polygon & always lies b/w the convex hull of the defining polygon. Any one of the control points can affect the shape of polygon of a curve at most K curve section i.e if you change the position of one control point it changes only that section of the curve.

(180) KNOT VECTORS.

Different kinds of Knot Vectors

↳ are used to construct the blending function of B-Spline Curve.

1. Uniform Knot Vector.

$X = [1, 2, 3, 4, 5, 6, 7]$.

$X = [0, 1, 2, 3, 4, 5]$.

$X = [0, 0.25, 0.5, 0.75, 1]$.

Open Uniform Knot Vectors.

→ are uniform knot vectors with repetition allowed

eg. $X = [0, 0, 0, 1, 2, 2, 2, 2]$.

eg. $X = [0, 0, 1, 2, 3, 3]$

Non Uniform Knot Vector.

May not be equally spaced and repetition of value is allowed

$X = [0, 0, 2, 0, 4, 0, 7, 1, 0]$.

Q. Generate a B spline curve of order 4 with four polygon vertices

$P_1 (1, 1)$

$P_2 (2, 3)$

$P_3 (4, 3)$

$P_4 (6, 2)$.

Computer Graphics

Topic:

Fractal Geometry, Fractal Classification And Fractal Dimension

Contributed By:

Verified Writer

2/8/14

FRACTAL GEOMETRY

All of the modeling techniques covered so far use Euclidean geometry methods.
- Objects were described using eq^{ns}. This is fine for manufactured objects that have irregular/fragmented features.

Natural objects can be described realistically described using fractal geometry methods (mountains, clouds, coral).

Fractal methods use procedures rather than eq^{n} to model objects - procedural modelling.

The major characteristics of any procedural model is that the model is not based on data, but rather on the implementation of a procedure following a particular set of rules.

Characteristics of Fractals.

A fractal is a geometric figure/natural object that combines the following characteristics

a) its parts have same form or structure as a whole except that they are at a different scale and may be slightly deform.

b) its form is extremely irregular/fragmented and remains so, whatever the scale of examination

c) it contains "distinct elements" whose scales are vary ~~are varies~~ and cover a wide range

d) formation by iteration

e) fractional dimension

Generating Fractals.

A fractal object is generated by repeatedly applying a specified transform function to points in a region of space.

If $P_0 = (x_0, y_0, z_0)$ a selected initial position, each iteration of a transformation function F generates successive levels of details with the calculation

$$P_1 = F(P_0), P_2 = F(P_1), P_3 = F(P_2)$$

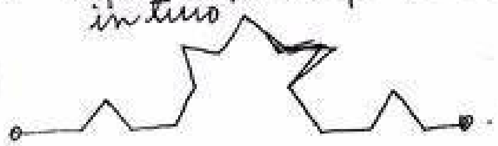
In general transformation is applied to a specified set of points

Generation of Koch Curve

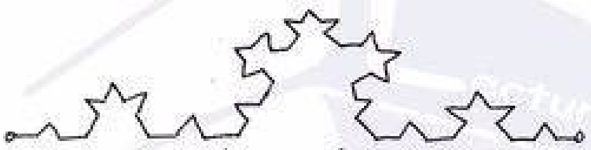
1. Start with a straight line

2. The straight line is divided into 3 equal parts and the middle part is replaced by two linear segments at angles 60° and 120°

3. Repeat the steps 1 & 2 to the four line segments generated in two



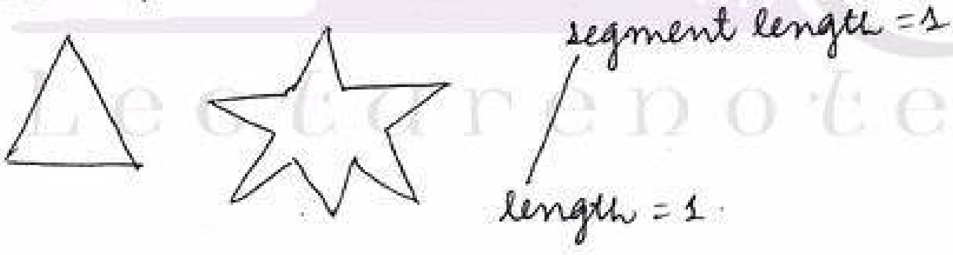
4. Further iterations will generate the following curves

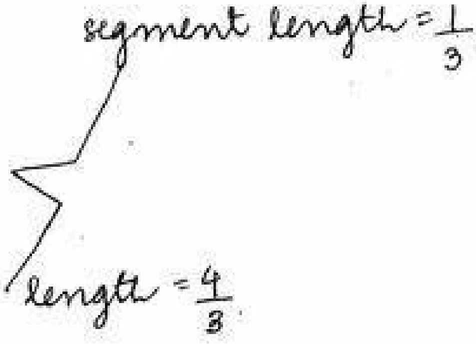


Repeat three times ✓

It can be iterated an infinite number of times by ✓ dividing a straight line segment into three equal parts and substituting the intermediate part with two segments of the same length.

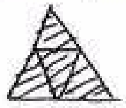
Example: The Koch Snowflake



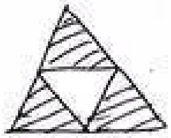


Construct the Sierpinski Triangle.

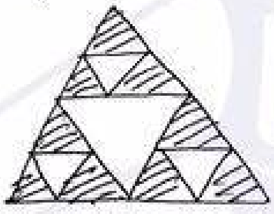
1. Start with the equivlateral triangle △
2. Connect the midpoint of each side of the triangle to form four seperate triangle



3. Cut the reclangle in the centre



4. Repeat the steps 1, 2, 3 on the three black triangles left behind



The centre triangle ~~for~~ of each black triangle at the corner were cut out as well.

5. Further repetition with adequate screen resolution will give the following pattern

It can be generated by infinitely repeating a procedure of connecting the midpoint of each side of Δ to form 4 seperate Δs & cutting out the Δ in the center

Fractal Dimension

The amount of variation in the structure of a fractal object is described as fractal dimension D

—More jagged looking object have larger dimensions

Calculating the fractal dimension for particular fractals

$$D = \ln(n) / \ln(1/s)$$

s = scaling factor

n = no. of subparts in subdivisions

64 Koch's Snowflake

After division have 4 segments

- $n = 4$ (new segments)
- ~~$s = 3$ (old segments)~~
- Fractal Dimension

$D = \ln 4 / \ln 3 = 1.262$

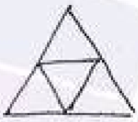
$n = 4$
No of new segments

$s = 1/3$
segments reduced by 1/3

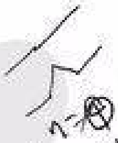
$D = \ln 4 / \ln(1/(1/3))$
$= \ln 4 / \ln 3$
$= \boxed{1.262}$

Sierpinski's Triangle

Divide each side by 2.
- Makes 4 triangles
- We keep 3



- There $n = 3$.
  get 3 new triangles from 1 old triangle.



$S = 1/3$ $\quad S = 1/2$ $\quad \ln 3/\ln 2$ $\quad n = 4$ $\quad s = 1/3$ $\quad \ln 4/\ln 3$

- $S = 1/2$

Fractal Dimension
$= D = \ln(3)/\ln 2 = \boxed{1.585}$

Classification of Fractals

Fractals can be classified into 3 groups:

(1) Self similar fractals
- These have parts that are scaled down versions of the entire objects
- Commonly used for modelling trees, shrubs etc

(2) Self affine fractals
- Have parts that formed with different scaling parameters in each dimensions.
- Typically used for terrain, water and clouds.

Invariant fractal sets

65

- Fractals formed with non linear transformations
Mandelbrot set, Julia set - generally not so useful

Computer Graphics

Topic:

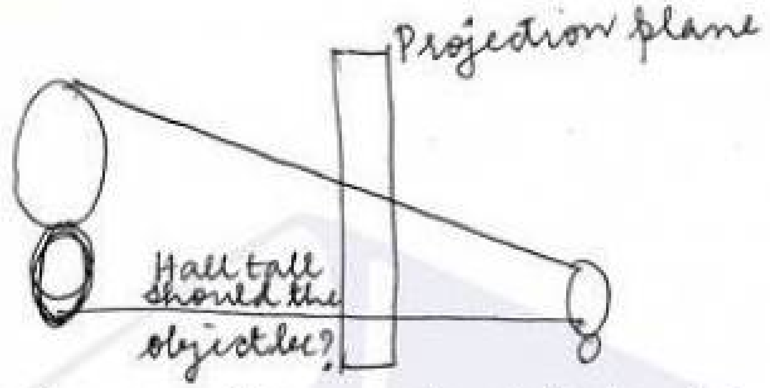
Parallel And Perspective Projections

Contributed By:

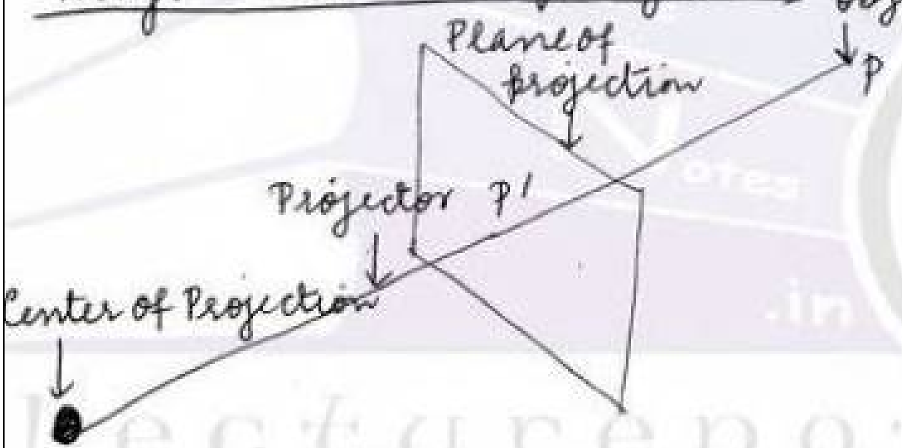
Verified Writer

Projection

3D projection is any method of mapping three dimensional point to a two dimensional plane.



Major Elements of Projection – ~~Projector~~ Geometry.



P (3D) ⟶ P' (2D).

- 3D Projection
  - Perspective
  - Parellel
    - Orthographic
      - Elevation
      - Axonometric
        - Isometric
    - Oblique
      - Cavalier
      - Cabinet

Perspective projection is divided in 3 parts:

- 1 pt.
- 2 pt
- 3 pt.

Gregarious – one who enjoys being in crowd.

1/8/14

Parellel Projection

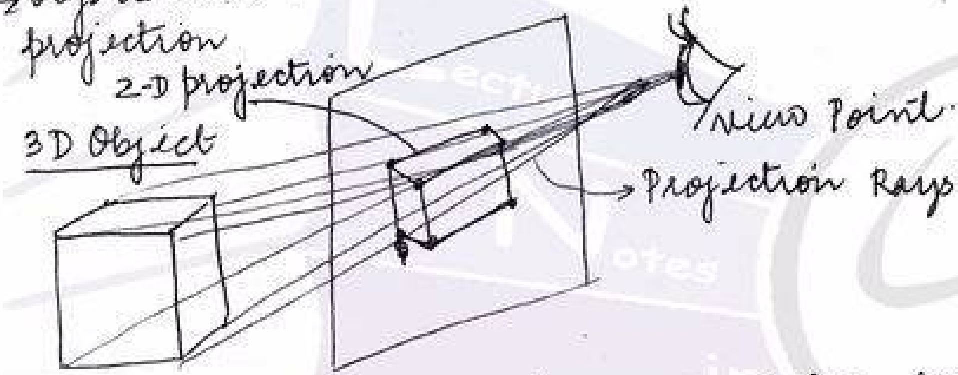
→ center of projection is at infinite distance
→ projectors are parellel to each other

• ~~Perspective Projection~~

→ it is good for exact measurement
→ the object size in case of parellel projection isn't changing
→ in parellel ~~sos~~ projection, objects look less natural

Perspective Projection

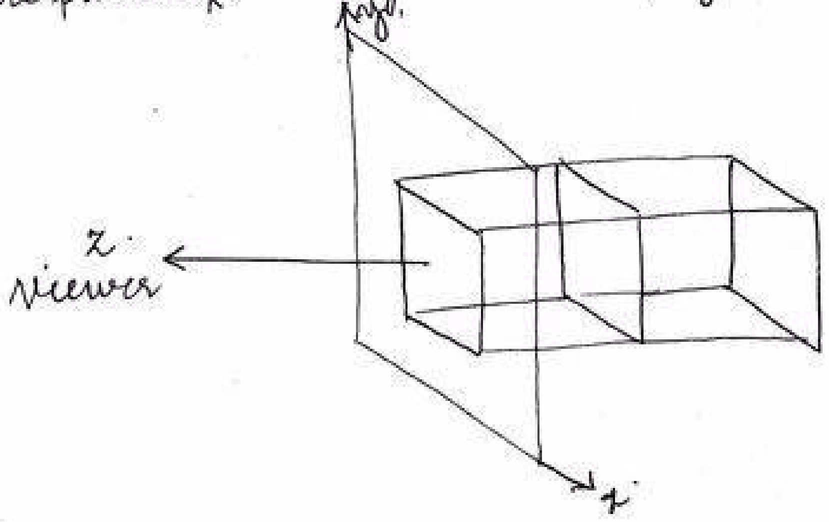
→ center of projection is at finite distance
→ projectors are not parellel to each other
→ projectors intersect at the center of projection
→ size of the object may change in perspective projection
→ objects look more natural as compared to parellel projection



Engg. graphics are obtained by projection from the 3-D object to the viewing surface (the projection plane).

Orthographic Projection

→ One type of parellel projection where the lines of projection are parellel & orthogonal to the projection plane.



...ction along z axis :-

...ansformation $x' = x$
$y' = y$

z-co-ordinate info is lost!

...ographic projection Matrix $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

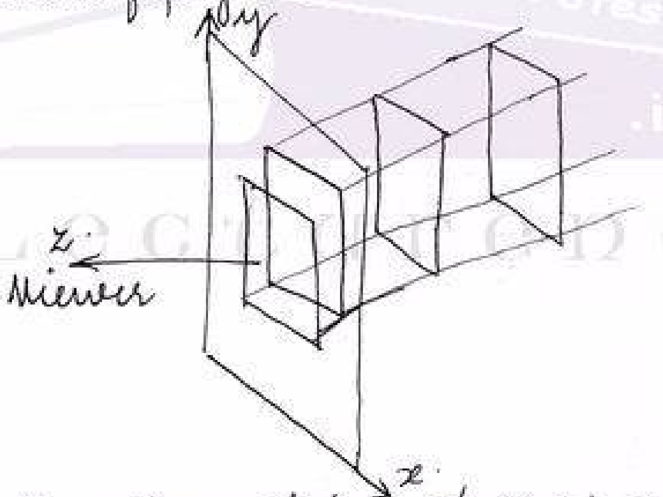
...ographic projection Matrix for production along x axis

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

...ilarly all y co-ordinates become zero for projection ...ong y axis.

...lique Projection

lines of projection are parallel, but not orthogonal to the plane of projections



Transformation: $x' = x + k_1 z$
$y' = y + k_2 z$

The z-co-ordinate value of the object point, leads to a shift of x, y co-ordinates of the projected point, proportion to z

Oblique Projection Matrix (along z axis). $= \begin{bmatrix} 1 & 0 & k_1 & 0 \\ 0 & 1 & k_2 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

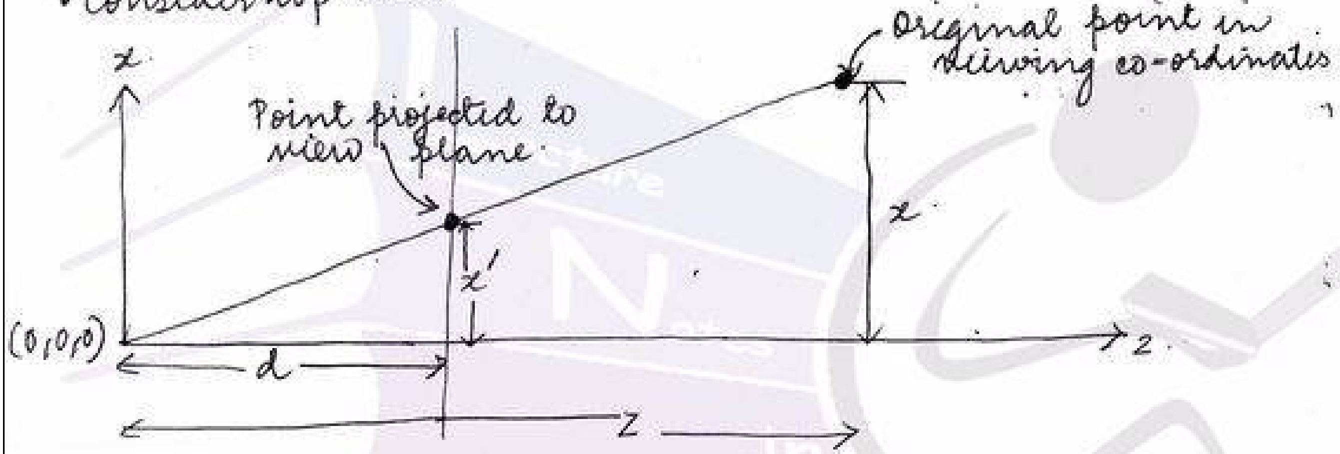
(10) Oblique projection along x & y axis:-

$$y' = y + k_1 x \qquad z' = z + k_2 x \qquad \begin{bmatrix} 0 & 0 & 0 & 0 \\ k_1 & 1 & 0 & 0 \\ k_2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$x' = x + k_1 y \qquad z' = z + k_2 y \qquad \begin{bmatrix} 1 & k_1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & k_2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Perspective Projection Matrix

- Suppose a camera position at origin $(0,0,0)$
- Suppose a view plane is at distance d from origin
- Consider top view



$$\frac{x'}{d} = \frac{x}{z}$$

The projected co-ordinates
$x' = dx/z$
similarly $dy/z = y'$

$$\frac{y'}{d} = \frac{y}{z}$$

$$\boxed{\begin{aligned} z' &= d \\ x' &= \frac{dx}{z} \\ y' &= \frac{dy}{z} \\ z' &= d \end{aligned}} \qquad \begin{bmatrix} \frac{d}{z} & 0 & 0 & 0 \\ 0 & \frac{d}{z} & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Perspective projection Matrix $M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}$

homogenous co-ordinates

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix}.$$

In normal co-ordinates, $(x', y', z') = (dx/z, dy/z, d)$

One Point Perspective projection along z axis

$$[x', y', z', 1] = [x \ y \ z \ 1]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & r \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\left.\begin{array}{l} x' = x \\ y' = y \\ z' = z \\ 1 = rz+1 \end{array}\right\} \rightarrow [x \ \ y \ \ z \ \ rz+1]$$

$rz+1 \Downarrow$ equal to 1
so divide all by $rz+1$.

r = shifting value along z axis

$$\left[\frac{x}{rz+1}, \frac{y}{rz+1}, \frac{z}{rz+1}, 1\right] = [x', y', z', 1].$$

Single point perspective proj. along y axis

$$[x' \ y' \ z' \ py+1] = [x \ y \ z \ 1]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & p & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

$$[x' \ y' \ z' \ 1] = \left[\frac{x}{py+1} \ \frac{y}{py+1} \ \frac{z}{py+1} \ 1\right]$$

Computer Graphics

Topic:

Visible Surface Detection Intensities Shading

Contributed By:

Verified Writer

20/8/2014

Visible Surface Detection Method

(70)

$c_1(u) = (u^2+2u-2, u^2)$

$c_2(u) = (u^2+3u+1, u+1)$.

Prove that $c_1(u)$ & $c_2(u)$ are continous, C^0, G^0 where they are joined $c_1(1)$ & $c_2(0)$.

Do they satisfy C^1 and G^1 continuity

In 3-D computer graphics visible surface detection / hidden surface removal is the process to determine which surface / part of the surface is visible from a certain viewpoint

This can be done by using two approaches:

(i) object spaced approach

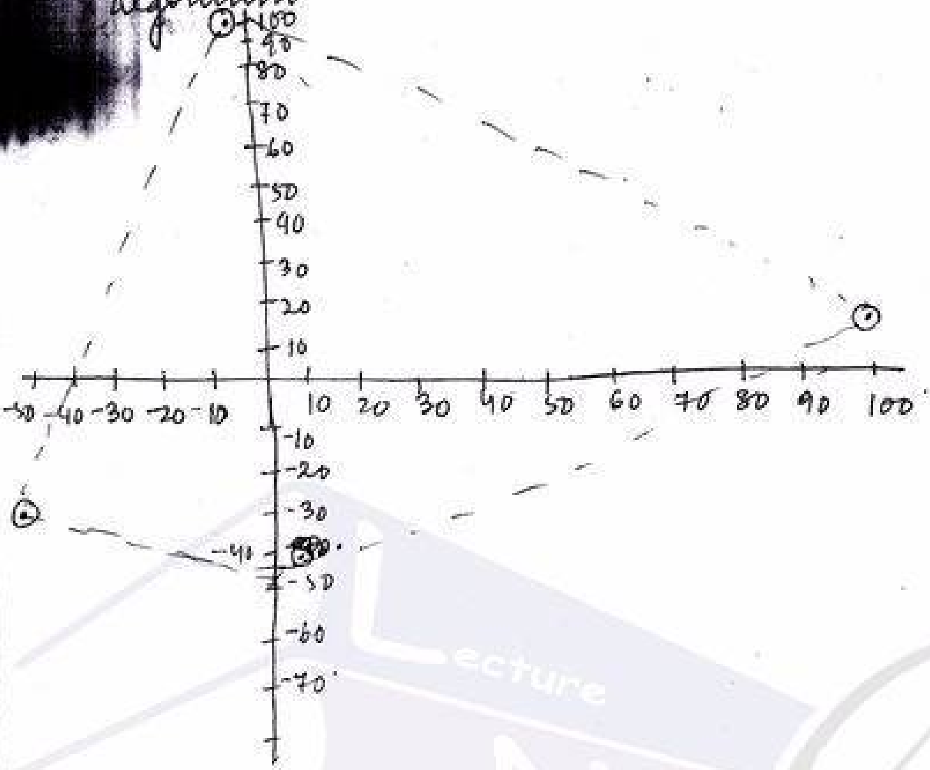
(ii) image space approach

→ In object space approach similar objects are compared to each other to find out the visibility of the surface / object

→ In image space approach we compare pixel by pixel to find out the visibility of an object

x value of point

polygon have 4 points A(-10,100), B(-50,-30), C(5,-40) D(100,10), $Z_A=5$, $Z_B=20$, $Z_C=15$, $Z_D=30$, calculate z value of point P(10,60) Zbuffer algorithm or depth buffer algorithm (71)



This algorithm is not suitable for transpr

72

Intensity Attenuation

As radiant energy from a point light source travels through space, its amplitude is attenuated by the factor $1/d^2$ where d is the distance that the light has travelled

A surface close to the light source (smalled) receives a higher incident intensity from the source than a distante surface (large d)

Problem in using the factor $1/d^2$ to attenuate itensities:
The factor $1/d^2$ produces too much intensity variations when d is small, and it produces very little variation when d is large.

- We can compensate for these problems by using inverse linear or quadratic functions of d to attenuate intensities

Attenuation Function :-

A general inverse quadratic attenuation function:

$$f(d) = \frac{1}{a_0 + a_1 d + a_2 d^2}$$

- The value of the constant a_0 can be adjusted to prevent f(d) from becoming too large when d is too small.

What Gouraud shading misses :-

- Highlights on the surface are sometimes displayed with anamolous shapes
- Can cause bright / dark intensity streaks to appear on the surface (Mach-band effect)

Dividing the surface into a greater number of polygon faces can reduce these effects.

Phong Shading

A more accurate method for rendering a polygon surface
Interpolates normal vectors, and then applies the illumination model to each surface point
Method developed by Phong Bui Tuong called Phong shading or normal-vector interpolation shading.
More realistic highlights
Greatly reduces the mach band effects

...ding the given surface into a number of polygonal ...ces

...alculate the normals of each polygonal faces seperately.

Determine the average uninormal vector at each polygonal vertex.

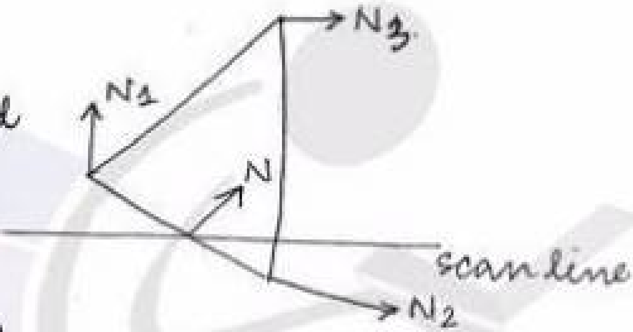
Linearly interpolate the vertex normal over the surface of the polygon.

...pply illumination model along each scan line to calculate projected pixel intensities for the surface points.

** Phong Shading. Step 2

Internal

The normal vector N for the scan line intersection point along the edge between the vertices 1 and 2 can be obtained by vertically interpolating between edge endpoints normals.



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

Increment methods are used to evulate ~~method~~ normals between scan lines and along each individual scan line.

Computer Graphics

Topic:

Computer Animation

Contributed By:

Verified Writer

23/9/2014 Animation:

Animate = "to give life to"
specify, directly or indirectly, how 'thing' moves in time and space.

Animation is the process by which we see still picture MOVE Each picture is shot on film one at a time and is shown at the rate of 24 picture per second making the picture appear to move.

Animation is the process of creating movement on the screen with a series of stile picture with the help of persistance of vision.

Types of animation: keyframe animation
procedural animation

7.4 Keyframe Animation

A simple yet effective way to animate 3D objects.
Each frame represents a key position for the object.
Number of frames are limited so object may appear to "jump" when going from one frame to other.
One solution is to just manually make more keyframe. This takes time and the programmer may not want to do it.
Instead of making them manually, use interpolation to create new frame.
Interpolation is creating a new position between existing positions.
Interbetweening with linear interpolation.
Linear interpolation creates in between frames at equal intervals along straight lines. The ball moves at a constant speed. Ticks indicate the locations of inbetween frames at regular time intervals.

Method for generation of key frames.
Compute first a small no. of key frames.
Interpolate the remaining frames in between these key frames (in-betweening).
Key frames can be computed
- at equal time intervals
- according to some other rules.
- for example when the direction of the path changes rapidly.

In-betweening examples

Given co-ordinates of a 2D point

key frame n: (x_n, y_n)

key frame n+1: (x_{n+1}, y_{n+1})

time interval between the two key frames: 1/second

To get smooth animation, needs at least 30 frames per second.

Solution: insert at least further 2 frames between the given two key frames.

Calculating in between frames using linear interpolation.

$\Delta x = (x_{n+1} - x_n)/3$.

$\Delta y = (y_{n+1} - y_n)/3$.

```
for(i = 1; i < 3; i++)
{
    x_i = x_n + i * Δx
    y_i = y_n + i * Δy
}
```

Calculation of depth (z) value:-

Assume that the polygon is a 3-D plane so equation of the 3-D plane:

$$Ax + By + CZ + D = 0$$

$$Z = -\left(\frac{A}{C}\right)x + \left(\frac{-B}{C}\right)y + \left(\frac{-D}{C}\right)$$

Let us take a point (x_0, y_0).

$$Z|_{(x_0, y_0)} = -\left(\frac{A}{C}\right)x_0 + \left(\frac{-B}{C}\right)y_0 + \left(\frac{-D}{C}\right)$$

$$Z|_{(x_0+1, y_0)} = -\left(\frac{A}{C}\right)x_0 + \left(\frac{-A}{C}\right) + \left(\frac{-B}{C}\right)y_0 + \left(\frac{-D}{C}\right)$$

$$Z|_{(x_0+1, y_0)} = Z|_{(x_0, y_0)} + \left(\frac{-A}{C}\right)$$

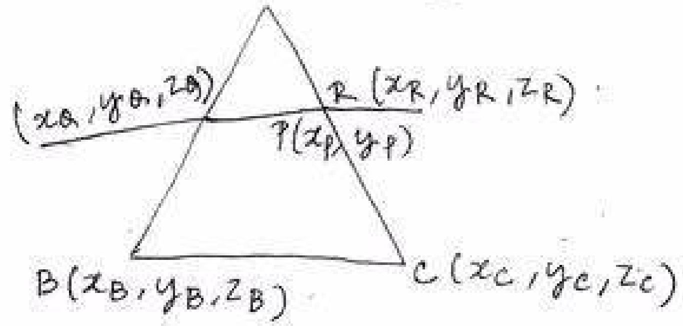
$$= Z_{old} + \left(\frac{-A}{C}\right)$$

The way continue to calculate the Z-value from left to right

$$Z|_{(x_0, y_0+1)} = Z|_{(x_0, y_0)} + \left(\frac{-B}{C}\right)$$

$$= Z_{old} + \left(\frac{-B}{C}\right)$$

Let us assume that the polygon is a triangle



Given an internal point 'P' (x_P, y_P) what is the Z-value of P

$$z_P = ?$$

By applying the linear interpolation technique we have the following equation

$$\frac{y_A - y_Q}{y_A - y_B} = \frac{z_A - z_Q}{z_A - z_B} \quad -(1)$$

$$z_Q = ?$$

$$z_Q = z_A - (z_A - z_B) \times \left(\frac{y_A - y_Q}{y_A - y_B}\right)$$

$$\frac{y_A - y_R}{y_A - y_C} = \frac{z_A - z_R}{z_A - z_C} \quad -(2)$$

$$z_R = z_A - (z_A - z_C)\left(\frac{y_A - y_R}{y_A - y_C}\right)$$

$$z_P = \frac{x_Q - x_P}{x_Q - x_P} = \frac{z_Q - z_P}{z_Q - z_R}$$

$$z_P = ?$$

$$y_P = y_Q = y_R$$

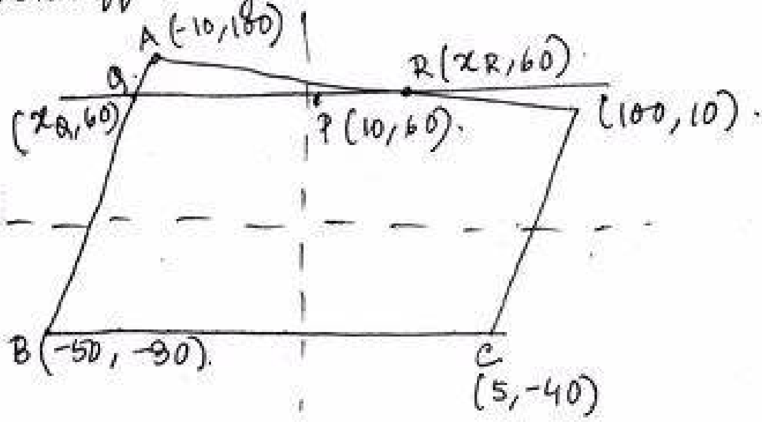
$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

$$\Rightarrow x = x_1 + (x_2 - x_1)\left(\frac{y - y_1}{y_2 - y_1}\right)$$

Q- A polygon having 4 points A, B, C, D. A(-10, 100); B(-50, -30), C(5, -40), D(100, 10).

$z_A = 5$, $z_B = 20$, $z_C = 15$, $z_D = 30$

Calculate the z-value of the point P(10,60) using z-buffer algorithm



$$x_Q = -50 + (-10+50)\left(\frac{60+30}{100+30}\right)$$

$$= -50 + (40) \times \left(\frac{90}{130}\right)$$

$$= -50 + 40 \times \frac{9}{13}$$

$$= -50 + 27.63$$

$$= -22.37$$

$$x_R = -10 + (100+100)\left(\frac{60-100}{10-100}\right)$$

$$= -10 + 110 \times \left(\frac{-40}{-90}\right) = -10 + 110 \times \frac{4}{9}$$

$$= -10 + \frac{440}{9}$$

$$10 + 48.89$$

$$= 38.89$$

$$z_Q = z_A - (z_A - z_B) \times \left(\frac{y_A - y_Q}{y_A - y_B}\right)$$

$$z_Q = 5 - (5 - 20) \times \left(\frac{100-60}{100+30}\right)$$

$$= 5 + 15 \times \frac{40}{130} = 5 + \frac{60}{13}$$

$$= 5 + 4.61$$

$$= 9.61$$

$$z_A - (z_A - z_C) \times \left(\frac{y_A - y_R}{y_A - y_C}\right)$$

$$= 5 - (5 - 15) \times \left(\frac{-10-60}{-10-100}\right) = 5 + 10 \times \left(\frac{-70}{-110}\right)$$

$$= 5 + \frac{70}{11} = 5 + 6.36$$

$$= 11.36$$

$$z_P = \frac{x_Q - x_P}{x_Q - x_R} = \frac{z_Q - z_P}{z_Q - z_R} \qquad z_P = \frac{-22.37 - 10}{-22.37 -}$$

78

Drawback of Buffer Method.

This algorithm is not suitable for Transperant object.

As we are using two buffers in Z-buffer so memory requirement is very high.

A-Buffer Method. (Accumulation buffer algorithm).

This algorithm is suitable for transperant object. It is based on Image space approach.

Accumulation buffer - buffer accomodate multiple pieces of information for each pixel in addition to depth for transperancy or anti-aliasing (high-end movies etc).

A buffer element stores;

- Depth field: Real number
- Surface data field (SDF); stores surface data or pointer.
- when depth >=0;
  - real number is depth of surface at pixel
  - SDF is surface color and pixel coverage percentage.

Depth ≥ 0	Intensity & other info

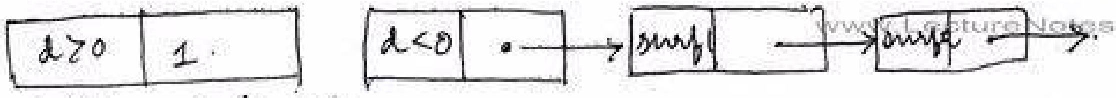
Algorithm (1/2)

Each position in the buffer can reference a linked list of surfaces

- Several intensities can be considered at each pixel position.
- Object edges can be antialiased

Each position in the A-buffer can has two fields

- depth field.
- It stores a positive or negative real number
- Intensity field
- Stores surface-intensity information or a pointer value.



Depth field   intensity field.

(a)   (b).

Organization of an A-buffer pixel position
(a) single surface overlap.
(b) multiple surface overlap.

Algorithm (2|2).

If the depth field is positive
- The number at that position is the depth
- The intensity field stores the RGB.

If the depth field is negative
- multiple surface contributions to the pixel
- the intensity field stores a pointer to a linked list of surfaces.
- data for each surface in the linked list
- RGB intensity components
- opacity parameters (percent of transparency)
- Depth
- percent of area coverage
- surface of identifier
- pointers to next surface.

A - Buffer Algorithm

1. for each pixel [i, j) do.
2. z-buffer [i, j] ← 0
3. Frame buffer [i, j] ← bgcolor.
4. end.
5. for each polygon A do
6. for each pixel in A do.
7. Compute the depth z and intensity at (i, j)
8. If the polygon is opaque polygon, find the depth and intensity of the pixel using z-buffer algorithm and store that surface in the linked list and remove all the polygons which are far away from it
9. If the polygon is transperant polygon, insert that polygon in the list and dont remove the far away polyg

80 10. End.

1/9/2014

Depth Sorting Method (Painter's Algorithm)

This algorithm is based on object space approach and image space approach

- Sorting operations in both image and object space
- The scanconversion of polygon surfaces in image space.

Basic functions:-

- Surfaces are sorted in order of decreasing depth.
- Surfaces are scan-converted in order, starting with the surface of greatest depth.

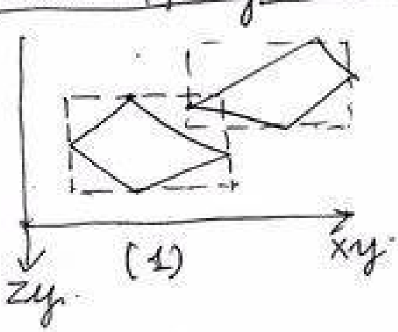
Algorithm:-

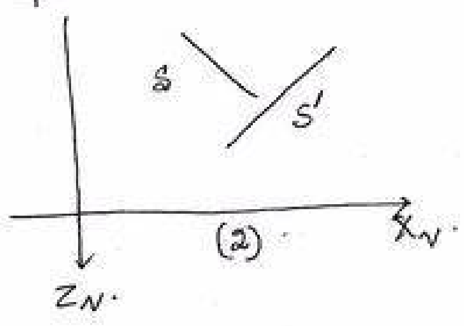
Referred to as painter's algorithm

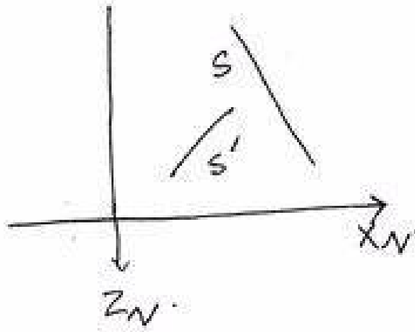
- In creating an oil painting
- first paints the background colour
- the most distant objects are added.
- then the nearer objects and so forths.
- finally the foregrounds are painted overall objects.

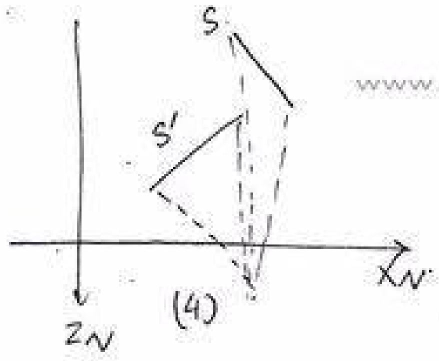
- each layer of paint covers up the previous layer.
- process
  - sort surfaces according to their distance from the viewplace.
- the intensities for the farthest surface are then entered into the refresh buffer.
- Taking each succeeding surface in decreasing depth order.

Overlapping Test Examples.









Overlapping Tests.

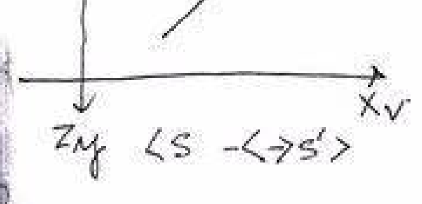
Test for each surface that overlaps with S.

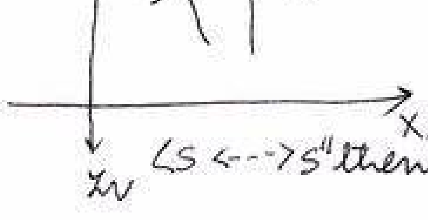
- The bounding rectangle in the xy plane for the two surfaces do not overlap (1).
- Surface 'S' is completed behind the overlapping surface relative to the viewing position (2)
- The overlapping surfaces is completely in front of S relative to the viewing position (3)
- The projection of the two surfaces onto the viewplane do not overlap (4).
- If all the surfaces pass at atleast one of the tests, none of them is behind S.
  - No recording is then necessary and S is scan converted

Surface Reordering

If all four tests fail with S'

- Interchange surface S and S' in the sorted list
- Repeat the tests for each surface that is reordered in the list.





82

Morphing -

→ derived from the word Metamorphosis

→ Metamorphosis means to change shape, appearance, etc

→ defined as:

→ Transition from one object to another

→ Process of transforming one image into another

- An animation technique allows you to blend two still images, creating a sequence of in-between pictures that when played in Quicktime, metamorphoses the first image into second

General Idea.

→ As the metamorphosis proceeds

→ The first image gradually distorted and is faded out

→ the second image starts out totally distorted towards the first and is faded in

Steps Involved:

The morph process consists of

1. Warping two images so that they have the same "shape"
2. Cross dissolving the resulting images

Warping:

A warp is a 2-D geometric transformation and generates a distorted image when it is applied to an image

Warping an image: apply a given deformation to it

Forward Mapping.

- Each pixel in source image is mapped to an appropriate pixel in destination range.
- Some pixels in destination image may not be mapped

Reverse Mapping.

This method goes through each pixel in the destination image and samples an appropriate source image pixel.

all destination image pixels are mapped to some source image pixel.

Cross Dissolving

A cross dissolve is a seq

Morphing Process

Step I : Interpolating the lines
- Interpolate the co-ordinates of the end points of every pair of lines
- Step II : Warping the Images

→ Each of the source images has to be deformed towards the needed frame.

→ The deformation works pixel by pixel is based on the reverse mapping. This algorithm is called Beier-Neely Algorithm

Computer Graphics

Topic:

Virtual Reality

Contributed By:

Verified Writer

Virtual Reality System.

Introduction

What is virtual reality?

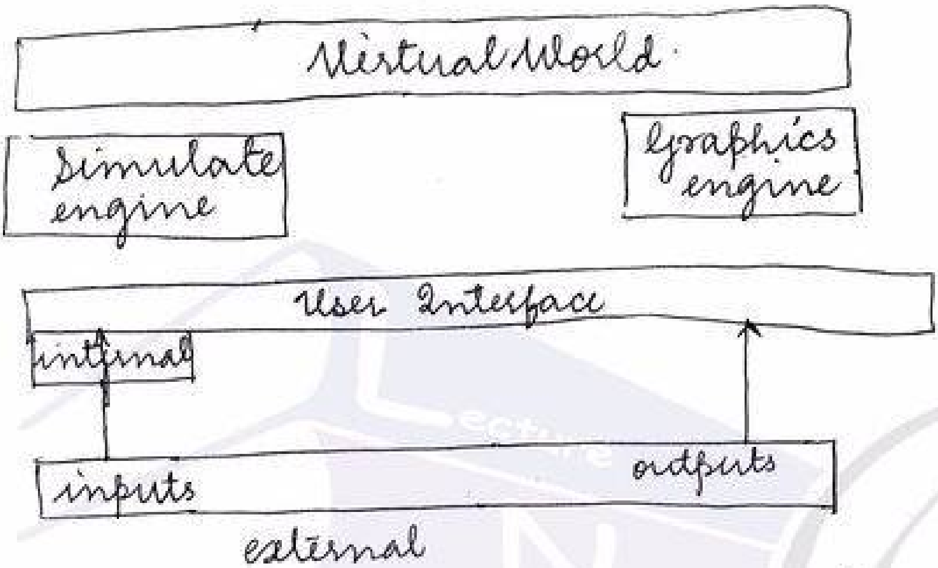
- Virtual reality refers to a high-end user interface involves real time simulation and interactions through multiple sensorial channels.
- VR is technology that allows users to interact with a computer-simulated environment be it a real or imaginary one.

VR is able to immerse you in a computer world of your own making a room, a city, the interior of human body. With VR you can explore any unchartered territory of the human imagination.
VR environments are primary visual experience, displayed either on a computer screen or through special stereoscopic display.

application

It is used in immersive, highly visual, three dimensional 3D environment
Development of

Design of VRS.



A typical VR system consists of ~~six~~ sixcomponents, the virtual world, graphics engine, simulation engine, user interface, ~~or~~ user input and user output

Types of VR System

WOW

Windows on World (WoW):

→ Also called desktop VR.
→ Using a conventional computer monitor to display the 3D virtual world without using a specialized movement-tracking equipment.

• **Immersive VR.**

The goal of VR is to completely immerse a user within a synthetic environment
completely immerse the user's personal viewpoint inside the virtual 3D world.
The user has no visual contact with the physical world

Often equipped with a Head Mounted Display (HMD)

$$z_a = z_1 + (z_4 - z_1)\frac{y_1 - y_s}{y_1 - y_4}$$

$$z_b = z_1 + (z_2 - z_1)\frac{y_1 - y_s}{y_1 - y_2}$$

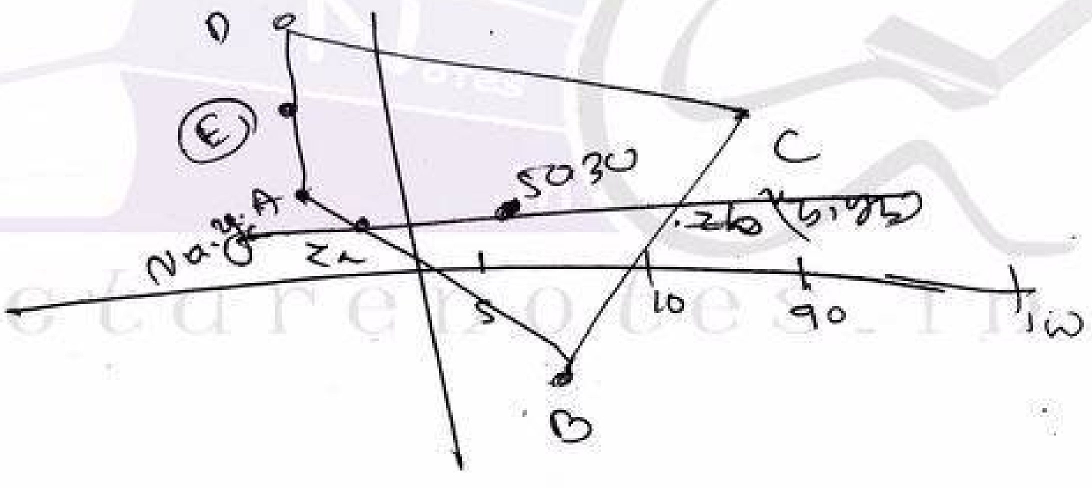
$$z_p = z_a + (z_b - z_a)\frac{x_a - x_p}{x_a - x_b}$$

z1 A = (-10, 100) 5.

z2 B = -50, 030 20

C = 5 40 15

d. 100, 10 30



$$z_a = z_1 + (z_4 - z_1)\frac{y_1 - y_s}{y_1 - y_4}$$

$$z = z_1 + (z_2 - z_1)\frac{y_1 - y_s}{y_1 - y_2} \quad \frac{x_a - x_p}{x_a - x_b}$$

(86)

CRT monitor is use.

can't find the dimension of the random fraction.

inverse scaling - Decrease Sx & Sy less than 1

Inter polation spline & ~~Aproximation~~ Aproximation spline

Hermite
→ Bsc spline

→ CRT

Reflection matrix about a line.

Shadowmasking

3 electrons are bombarded

Transform

Projection

line drawing