Digital Differential Analyzer (DDA) Line Algorithm

- This algorithm works on principal of obtaining the successive pixel values based on the differential equation.

(x_0, y_0) (x_1, y_1)

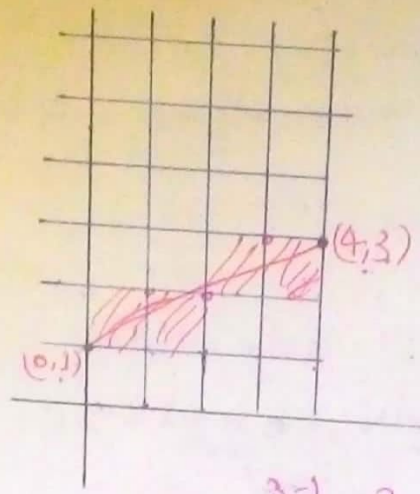
$$m = \frac{y_1 - y_0}{x_1 - x_0}$$

Case 1: $m < 1$

$x_n = x_0 + 1, \quad y_n = y_0 + m$

Plot $(x_n, \text{round}(y_n))$

$x_0 = x_n, \quad y_0 = y_n$



$$\therefore m = \frac{3-1}{4-0} = \frac{2}{4} = 0.5$$

Plot			
x_n	round(y_n)	x_n	y_n
0	1	0	1
1	2	0+1 = 1	1+0.5 = 1.5
2	2	1+1 = 2	1.5+0.5 = 2
3	3	2+1 = 3	2+0.5 = 2.5
4	3	3+1 = 4	2.5+0.5 = 3

Case 2: $m > 1$

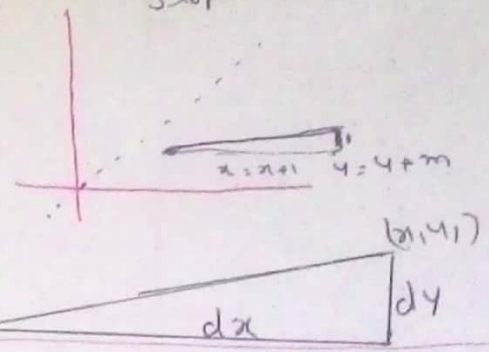
$y_n = y_0 + 1, \quad x_n = x_0 + \frac{1}{m}$

Plot $(x_n, \text{round}(y_n), y_n)$

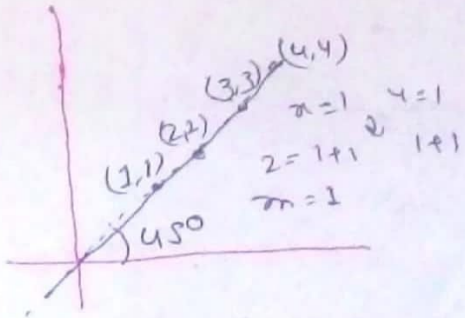
$y_0 = y_n, \quad x_0 = x_n$

Line $y = mx + c$ (m = slope)

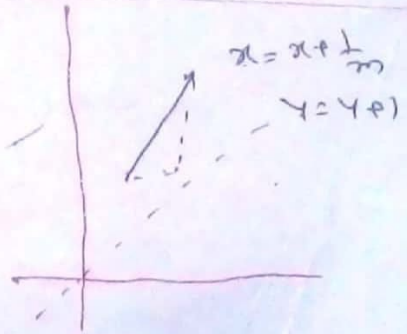
$m < 1$



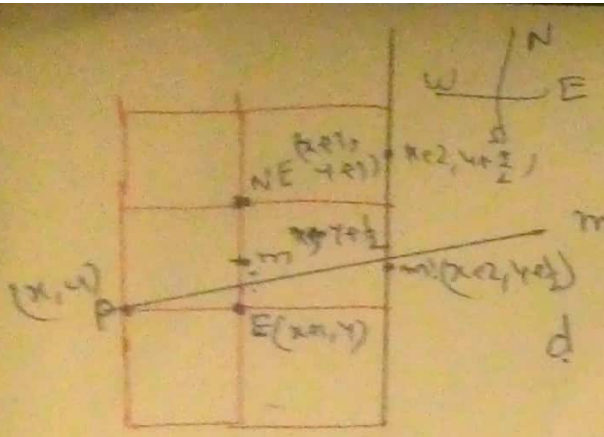
$m = 1$



$m > 1$



Mid Point Line Algorithm (Proof)



$ax + by + c = 0 \quad -(i)$

$m = (x+1, y+\frac{1}{2})$

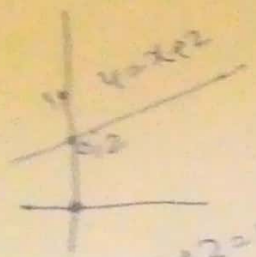
$d = a(x+1) + b(y+\frac{1}{2}) + c$

$= ax + a + by + \frac{b}{2} + c = a + \frac{b}{2}$

$y = mx + c = \frac{dy}{dx}x + c'$

$x \cdot dy - y\,dx + c'dx = 0 \quad -(ii)$

$(a = dy, \; b = -dx, \; c = c'dx)$



$f(x, y) = x - y + 2 = 0$

$f(0,0) = 0 - 0 + 2 > 0$

$f(1,2) = 1 - 2 + 2 = 0$

$f(2,3) = 2 - 3 + 2 = 0$

$-1 < 0$

$f(x, y) > 0$ → above line

- $d < 0 \rightarrow E$
- $d > 0 \rightarrow NE$

→ $E \rightarrow (x+1, y)$

$d_{new} = F(x+2, y+\frac{1}{2}) = 2a + \frac{b}{2}$

$\Delta E = a = dy$

→ $NE \; (x+1, y+1)$

$\Delta NE = dy - dx$

Mid point Line Algorithm

$dx = x_2 - x_1; \quad dy = y_2 - y_1 \Rightarrow$

$d = 2dy - dx, \; \Delta E = 2dy.$

$\Delta NE = 2(dy - dx);$

$x = x_1, \; y = y_1;$

Putpixel (x, y);

While $(x < x_2)$

if $(d \le 0)$

$d = d + \Delta E; \; x = x + 1$

else

$d = d + \Delta NE; \; x = x + 1; \; y = y + 1$

end

Putpixel (x, y);

end while

Mid Point Circle - Algorithm

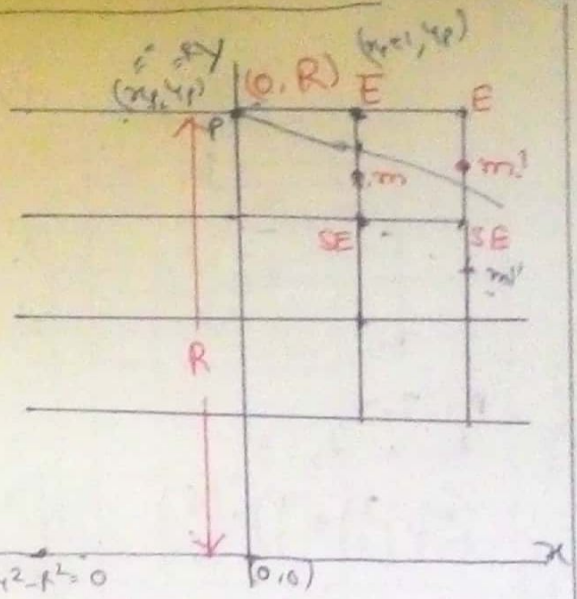
Circle equation

$$F(x,y) = x^2 + y^2 - R^2 = 0$$

- Evaluate F(m)

if $F(m) < 0$ (m inside circle)
Choose E

if $F(m) > 0$ (outside circle)
Choose SE



Decision: if Point (x_p, y_p)

$$d = F(m) = F\left(x_p + 1, y_p - \frac{1}{2}\right)$$

$$d = (x_p + 1)^2 + \left(y_p - \frac{1}{2}\right)^2 - R^2$$

when E ⇒ $d < 0$

$$d_{new} = F(m') = F\left(x_p + 2, y_p - \frac{1}{2}\right)$$

$$d_{new} = (x_p + 2)^2 + \left(y_p - \frac{1}{2}\right)^2 - R^2$$

$$\Delta E = d_{new} - d = 2x_p + 3$$

when SE selected:

$$d'_{new} = F(m'') = F\left(x_p + 2, y_p - \frac{3}{2}\right)$$

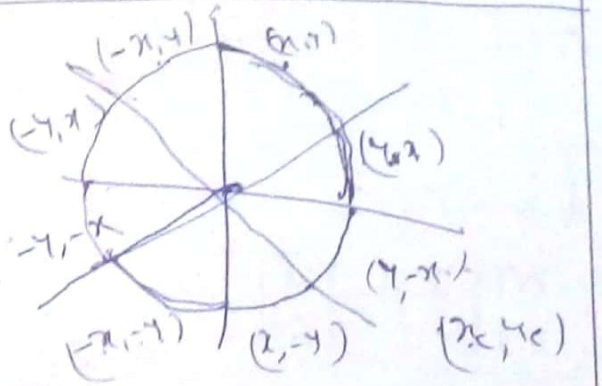
$$\Delta SE = d'_{new} - d = 2x_p - 2y_p + 5$$

For a given point (x', y')

$F(x', y') = 0$ → On Circle

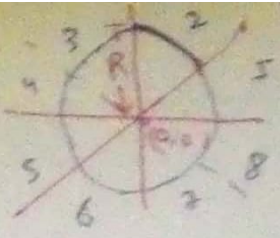
$F(x', y') > 0$ → Outside Circle

$F(m)$ $F(x', y') < 0$ → Inside Circle



Putpixel $(x_c + x, y_c + y)$

Algorithm



$x = 0$; $y = R$; $d = \frac{5}{4} - R$

Putpixel (x, y);

while $(y > x)$ do

If $(d < 0)$

$d = d + \underbrace{2x + 3}_{\Delta E}$; $x = x + 1$;

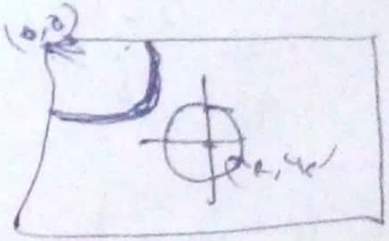
else

$d = d + \underbrace{2x - 2y + 5}_{\Delta SE}$; x^{++}; y^{--};

end

Putpixel (x, y);

endwhile → Putpixel (x, y);



2D Transformation

Transformation

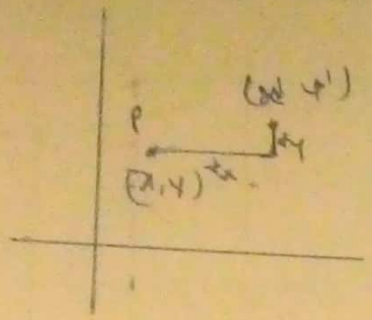
1) Translation

$x' = x + t_x$

$y' = y + t_y$

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

$[x' \; y'] = [x \; y] + [t_x \; t_y]$



$P = 2, 3$

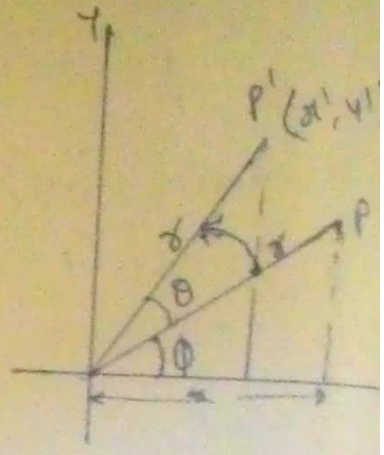
$t_x = 4 \quad t_y = 3$

$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \end{bmatrix} + \begin{bmatrix} 4 \\ 3 \end{bmatrix} = \begin{bmatrix} 6 \\ 6 \end{bmatrix}$

Rotation

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

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



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

Proof-

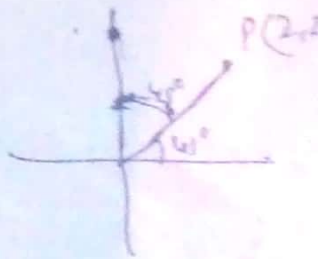
$x' = \gamma\cos(\theta+\phi) = \gamma\cos\phi\cos\theta - \gamma\sin\phi\sin\theta$

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

$x = \gamma\cos\phi \quad , \quad y = \gamma\sin\phi$

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

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



$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos 45° & -\sin 45° \\ \sin 45° & +\cos 45° \end{bmatrix} \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} \frac{2}{\sqrt{2}} - \frac{2}{\sqrt{2}} \\ \frac{2}{\sqrt{2}} + \frac{2}{\sqrt{2}} \end{bmatrix}$

$y' = 2\sqrt{2}$

Scaling

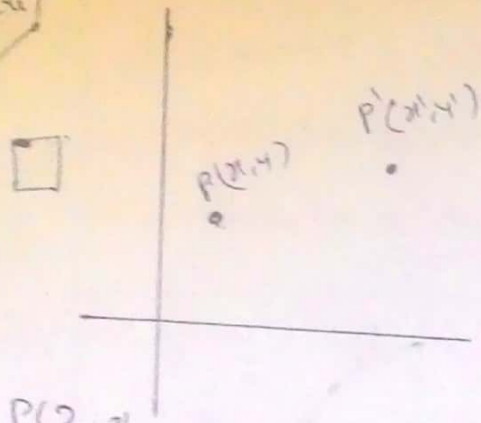
$x' = S_x x$

$y' = S_y y$

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

T_S

tera $T_S \cdot \begin{bmatrix} x_1 & x_2 \\ y_1 & y_2 \end{bmatrix}$ ..



$\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 4 \\ 6 \end{bmatrix}$

Bresenham Circle Algorithm

rcle equation -

$$f(x,y) = x^2 + y^2 - R^2 = 0$$

$$f(x_p, y_p) = x_p^2 + y_p^2 - R^2 = 0$$

$d_E = f(x_p+1, y_p) = (x_p+1)^2 + y_p^2 - R^2$ [+Ve]

$d_{SE} = f(x_p+1, y_p-1) = (x_p+1)^2 + (y_p-1)^2 - R^2$ [-Ve]

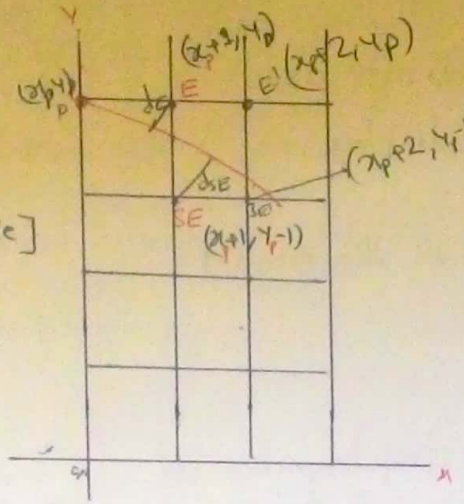
$p = d = d_E + d_{SE}$ // decision Parameter

If (d < 0) - Select E // $d_{SE} > d_E$

$x_E = x_p + 1, \; y_E = y_p$

If (d > 0) → Select SE // $d_E > d_{SE}$

$x_{SE} = x_p + 1; \; y_{SE} = y_p - 1$



F(x,y) > 0 → outside

F(x,y) < 0 - Inside

$d = (x_p+1)^2 + y_p^2 - R^2 + (x_p+1)^2 + (y_p-1)^2 - R^2$

$d = 3 + 4x - 2y$ // $d = 3 - 2R, \; x=0, \; y=R$

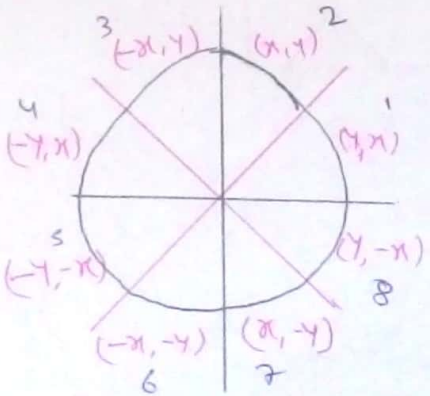
$d^E_{new} = (x_p+2)^2 + y_p^2 - R^2 + (x_p+2)^2 + (y_p-1)^2 - R^2$

$d^E_{new} = 9 + 8x - 2y$

$\Delta E = d^E_{new} - d = 6 + 4x$

$d^E = d + \Delta E$

$\Delta SE \Rightarrow d^{SE} = 4(x-y) + 10$



Algorithm:

```
x = 0; Y = R;   P = 3 - 2R
Putpixel (x, Y);
While (Y > x) do
{  x++;
  If (d<0)
    d = d + 4*x + 6;
  Else
    d = d + 4*(x-Y) + 10; Y--;
end
Putpixel (x, Y);
end while
```

(x_c, y_c) Putpixel $(x_c+x, y_c+y) - 2$

4 $(x_c+y, y_c+x) - 1$

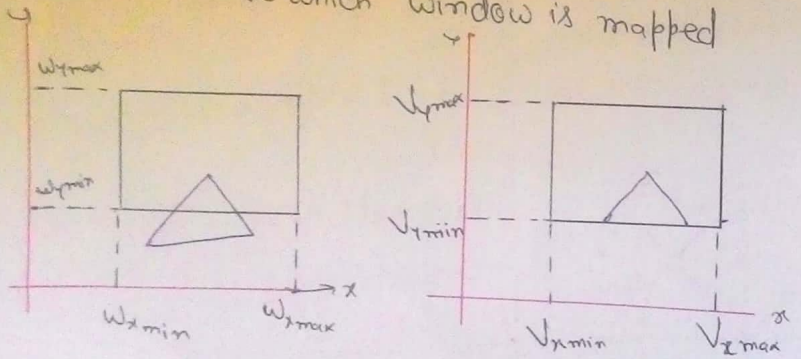
— $(x_c+y, y_c-x) - 8$

Window and View Port

Window Port: World coordinate area selected for display

→ Window is associated with the object rather than with image

View Port: An area on a display device to which window is mapped



window to viewport transformation

$$f_x = \frac{V_{xmax} - V_{xmin}}{W_{xmax} - W_{xmin}}, \quad f_y = \frac{V_{ymax} - V_{ymin}}{W_{ymax} - W_{ymin}}$$

If $(f_x < f_y)$

then $f = f_x$

else $f = f_y$

Let $V_x = V_{xmax}$, $W_x = W_{xmax}$, $V_y = V_{ymax}$, W_y

$$f = \frac{V_x - V_{xmin}}{W_x - W_{xmin}}, \quad f = \frac{V_y - V_{ymin}}{W_y - W_{ymin}}$$

Ex Windowport $(\overset{W_x}{100}, 100, 300, 300)$

Viewport $(50, 50, \overset{V_{xmax}}{150}, \overset{V_{ymax}}{150})$

Convert Windowport coordinate $(200, 200)$ to viewport

↓

$(100, 100)$

$$f = \frac{150 - 50}{300 - 100} = \frac{1}{2} = \frac{V_x - 50}{200 - 100}$$

$$\boxed{V_x = 100}$$

$$\frac{1}{2} = \frac{V_y - 50}{200 - 100} \Rightarrow V_y = 100$$

Each position (x

All transform

Composite tran

unifying repre

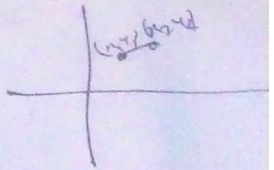
(x, y, w) P

Translation: T_d

$x = (x, y)$

↓

$(x\ y\ 1$



Homogenous Coordinates System

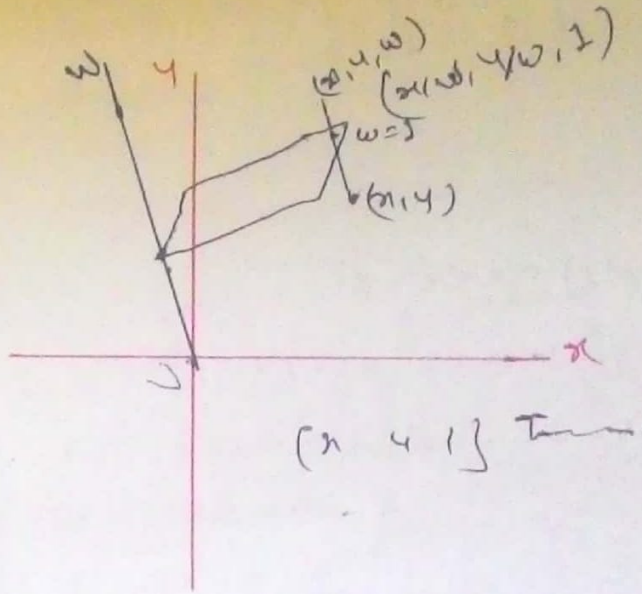
Each position (x, y) is represented as $(x, y, 1)$

All transformation can be represented as matrix multiplication

Composite transformation becomes easier

Unifying representation for transforming

(x, y, w) $\quad P_2 = (4, 2, 2) = (8, 4, 4)$



Translation:

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

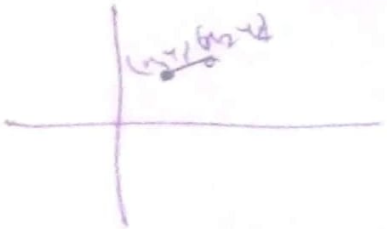
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

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

$(x\ y\ 1) \longrightarrow (x'\ y'\ 1)$

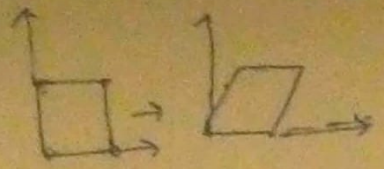
Scaling $= \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$

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



$$= T_t \times T_S \times \begin{bmatrix} x_1 & x_2 \\ y_1 & y_2 \\ 1 & 1 \end{bmatrix}$$

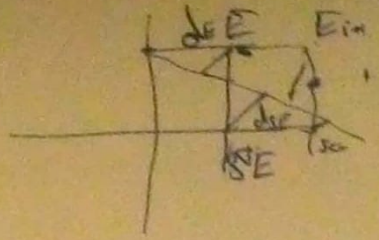
Shearing



$x' = x + R_x y$

$y' = y$

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



$d_E = f(x+1, y) = (x+1)^2 + y^2 - R^2$ (+ve)

$d_{SE} = f(x+1, y-1) = (x+1)^2 + (y-1)^2 - R^2$ (+ve)

if it is inside the circle ; <0 -ve

outside ——— >0 +ve

$d = d_{SE} + d_E = x$

if $d(<0)$ — II E

$d_{new} = (x+x)^2 + y^2 - R^2$

$= x^2 + 1 + 2x + y^2 - R^2$

$= x^2 + 1 + 2x + y^2 + 1 - 2y - R^2$

$= 2(x^2 + y^2 - R^2) + 2 + 4x + 1 - 2y$

$d = 3 + 4x - 2y$ ✓

$d = 3 - 2x$

$d_{new} = d_{E'} + d_{SE}$

$= (x+2)^2 + y^2 - R^2 + (x+2)^2 + (y-1)^2 - R^2$

$= x^2 + 4 + 4x + y^2 - R^2 + x^2 + 4 + 4x + y^2 + 1 - 2y - R^2$

$= 4 + 4x + 4 + 4x + 1 - 2y$

$= 8 + 8x + 1 - 2y$

$d_{new} = 9 + 8x - 2y$ ✓

$\Delta d_E = d_{new} - d = 9 + 8x - 2y - 3 - 4x + 2y$

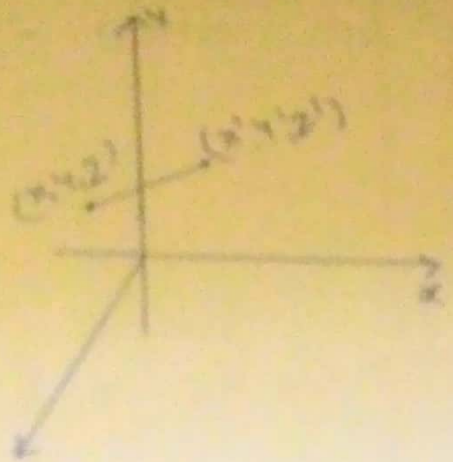
$= 6 + 4x$

3-D Transformation with Homogeneous Coordinates

Translation

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

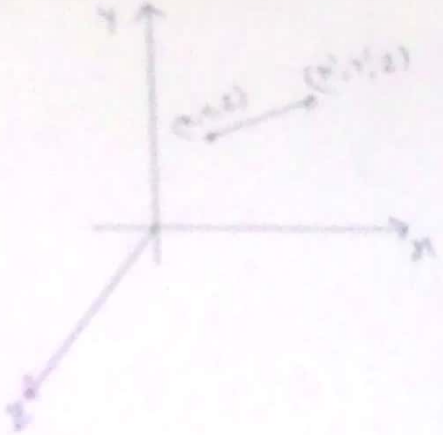
$P' = T \cdot P$



Scaling

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

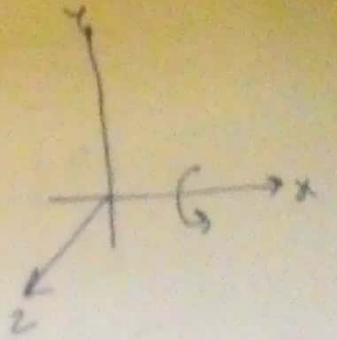
$P' = T_s \cdot P$



Rotation

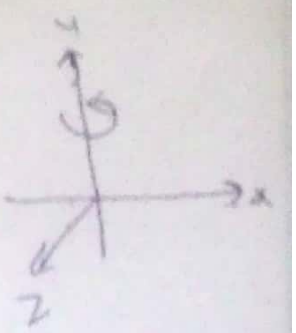
x-axis

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



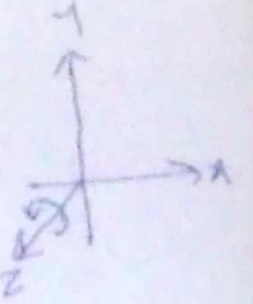
y-axis

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



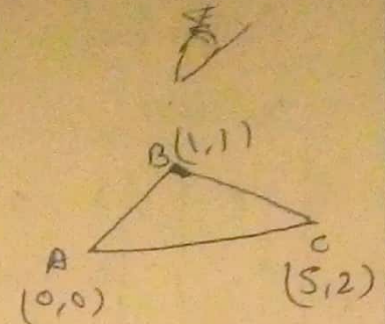
z-axis

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



Q:= Perform a 45° rotation of Triangle A(0,0) B(1,1) and C(5,2)

(i) about origin



$$P = \begin{bmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

$$T_{R\theta} = \begin{bmatrix} \cos 45° & -\sin 45° & 0 \\ \sin 45° & \cos 45° & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

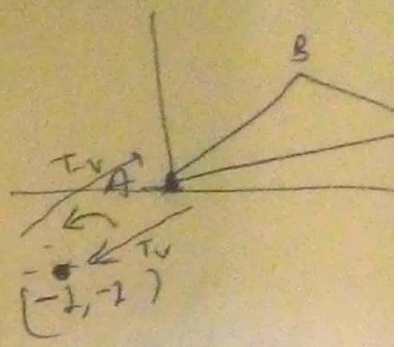
$$P' = T_{R\theta} \cdot P$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

$$P' = \begin{bmatrix} 0 & 0 & \frac{3\sqrt{2}}{2} \\ 0 & \sqrt{2} & \frac{7\sqrt{2}}{2} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x_A' & x_B' & x_C' \\ y_A' & y_B' & y_C' \\ 1 & 1 & 1 \end{bmatrix}$$

$A' = (0,0)$ $B' = (0,\sqrt{2})$ $C' = \left(\frac{3\sqrt{2}}{2}, \frac{7}{\sqrt{2}}\right)$

(ii) about P(-1,-1)



$$P' = T_V \, T_{R_\theta} \, T_{-V} \cdot P$$

$$= \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \gamma_{\sqrt{2}} & -\gamma_{\sqrt{2}} & 0 \\ \gamma_{\sqrt{2}} & \gamma_{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & \\ 0 & 1 & \\ 1 & 1 & \end{bmatrix}$$

$P' =$

Clipping

- Identify portion of a picture that is inside the viewport and removing the portion that is outside the viewport

Cohen Sutherland Line Clipping Algorithm

Algorithm:

1) Assign a 4 bit code to each endpoint C_0, C_1

2) If $C_0|C_1 = 0000$ — completely accepted (inside window)

   else if $C_0 \& C_1 \neq 0000$ → rejected

   else clip

   If line crossed $x_{w\min}$ or $x_{w\max}$

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

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

   where $x = x_{w\min}$ or $x_{w\max}$

   else ($y_{w\min}$ or $y_{w\max}$)

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

   where $y = y_{w\min}$ or $y_{w\max}$

- Verify - $x_{w\min} \le x \le x_{w\max}$? If not satisfy
  $y_{w\min} \le y \le y_{w\max}$] repeat clip

TBRL - bit

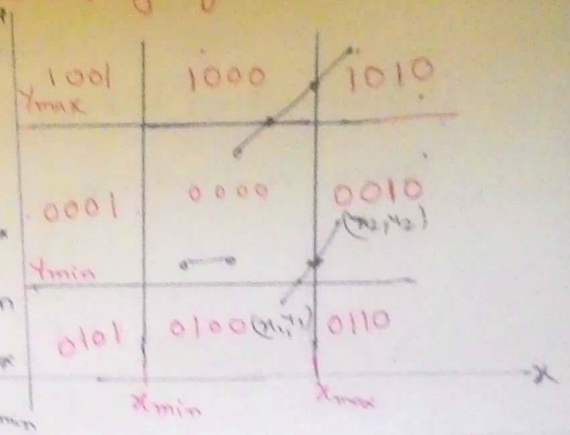
T - Top, B - Bottom, R - Right, L - Left

Top - 1 if $y > y_{max}$

B - 1 if $y < y_{min}$

R - 1 if $x > x_{max}$

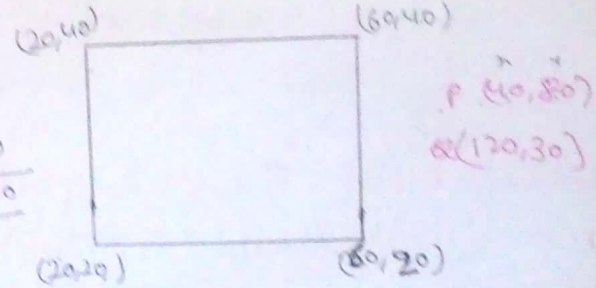
L - 1 if $x < x_{min}$



P - 1000

Q - 0010

AND 0000



P (40, 80)

Q (120, 30)

Bezier Curve

Bezier curve use a constaction curve, In which the interpolating polynomials depend on certain Control points

→ A Bezier curve is a parametric curve that uses Bernstein Polynomial as a basis function/blending function.

Degree 'n' (order n+1) Bezier Curve

$$Q(u) = \sum_{i=0}^{n} P_i B_{i,n}(u), \quad 0 \le u \le 1$$

P_i → Control Point, $B_{i,n}(u) = {}^{n}C_i \, u^i (1-u)^{n-i}$

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

If $n = 3$ (degree) { 4 Control Point }

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

$$R(u) = Q(u) = P_0(1-u)^3 + P_1 3u(1-u)^2 + P_2 3u^2(1-u) + P_3 u^3$$

$$x(u) = x_0(1-u)^3 + x_1 3u(1-u)^2 + x_2 3u^2(1-u) + x_3 u^3$$

{ $0 \le u \le 1$ }

$0 \le u \le 1$

$u = 0.1$

$u = 0.2$

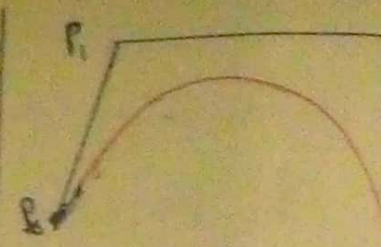
$u = 0.3$

$$B_{0,3}(u) = {}^{3}C_0 \, u^0 (1-u)^3 = (1-u)^3$$

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

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

$$B_{3,3}(u) = u^3$$



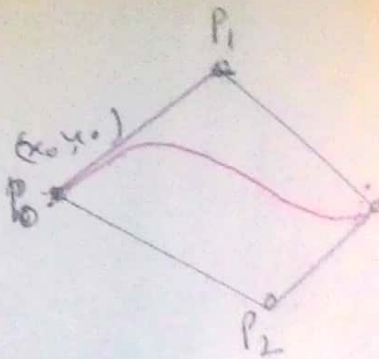
Cubic BezierCu

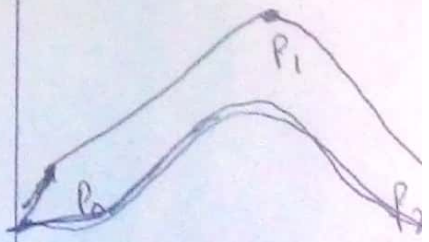
→ 4 Control Point

→ degree 3 { n

↓

Con





Quadratic Bezier Curve

Cohen Sutherland Clipping - Example

Q ⇒ Window is defined as A(10,20), B(20,20), C=(20,10), D(10,10) find visible portion of line P(15,15) and Q(15,5)

x, y	T	B	R	L
Q(15,5) →	0	1	0	0

T ⇒ $y > y_{max}$

B → $y < y_{min}$

R → $x > x_{max}$

L → $x < x_{min}$

P(15,15) → 0 0 0 0

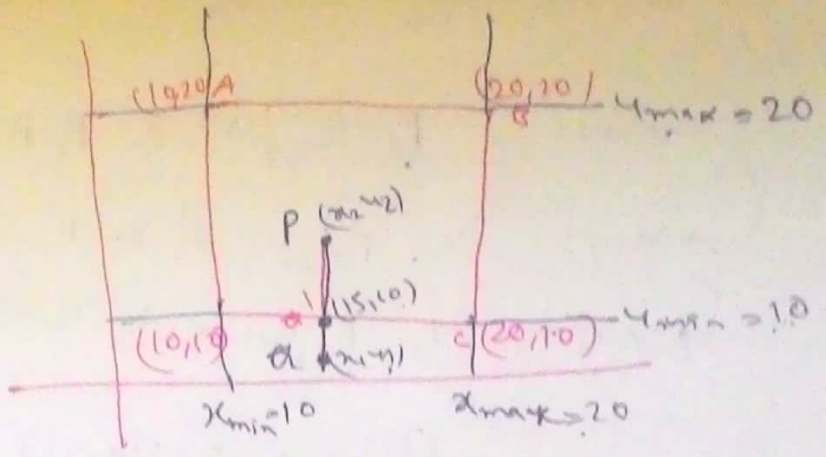
P & Q = 0 →

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

$= 15 + (10 - 5)/\infty$

$x = 15$

(15,10)



$m = \frac{20}{0} = \infty$

Example

Bezier Curve

Ex: Construct Bezier curve for control points $(4,2)$ $(8,8)$ and $(16,4)$
P_0 P_1 P_2

$$Q(u) = \sum_{i=0}^{2} P_i B_{i,2}(u) \quad , 0 \le u \le 1$$

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

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

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

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

$$B_{1,2}(u) = {}^2C_1 u^1 (1-u)$$

$$= \frac{2!}{1!1!} u(1-u)$$

$$= 2u(1-u)$$

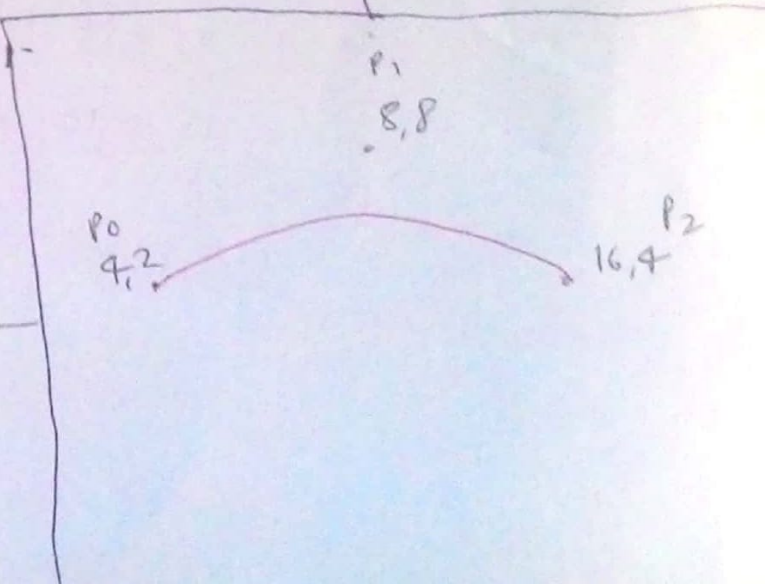
$$B_{2,2} = u^2$$

$$x(u) = x_0 (1-u)^2 + x_1 \cdot 2 \cdot u(1-u) + x_2 u^2 = 4u^2 + 8u + 4$$

$$y(u) = y_0 (1-u)^2 + y_1 \, 2 \cdot u(1-u) + y_2 u^2 = -10u^2 + 12u + 2$$

if

$u =$	$x(u)$	$y(u)$	
$u = 0$	4	2	
$u = 0.2$	5.76	4.0	$-10(0.2)^2 + 12 \times 0.2 + 2 = -0.4 + 2.4 + 2$
$u = 0.4$	7.84	5.20	
$u = 0.6$	10.24	5.6	
$u = 0.8$	12.96	5.2	



B-Splines

→ Each control point is associated with unique basis function

→ Each point affects the shape of the curve over range of parameter values where the basis function is non-zero

$$x(u) = \sum_{i=0}^{i=n} N_{i,k}(u)\, x_i \quad , \quad 0 \le u \le n-k+2$$

→ Control points → $n+1$, and order of curve → k

→ Curve is made up of $n-k+2$ segments

Each segment → Influenced by k points

Basis function:

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

Ex → $n=5$ (6) and $k=3$

t_i $(0 \le i \le 8)$ are knot value

$t_i = \{0,0,0,1,2,3,4,4,4\}$

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

t_i $(0 \le i \le n+k)$ – Knot values

$t_i = 0$ if $i < k$ (e.g. $0 < 3$)

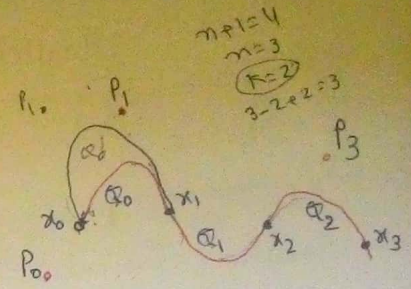
$t_i = i-k+1$ if $k \le i \le n$

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

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

$= 0$ otherwise

B-Spline (Local Control)

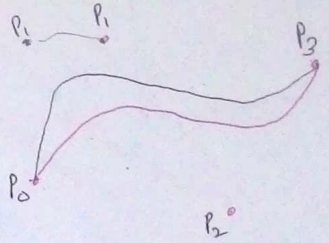


$k=2$

P_0, P_1, P_2, P_3 – Control Points

x_0, x_1, x_2, x_3 – Knot Value

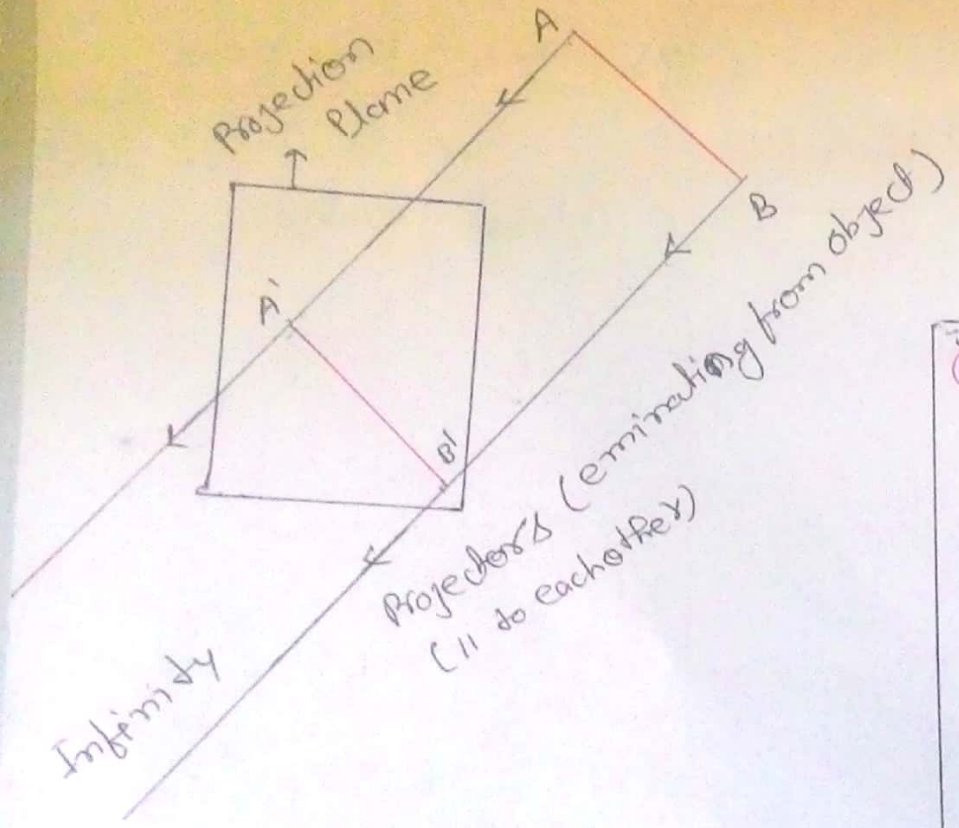
Bezier Curve (Global Control)

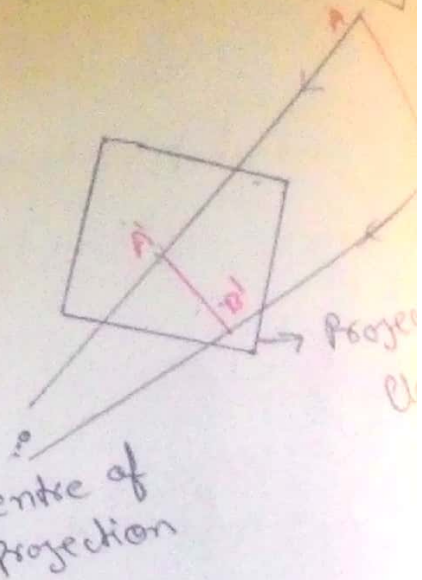


$$x(u) = \sum_{i=0}^{n} B_{i,n}(u)\, x_i \quad , \quad 0 \le u \le 1$$

Projection

- Parallel Projection
  - Oblique
    - Cavalier
    - Cabinet
  - Orthographics
    - Axonometric
      - Isometric
      - Dimetric
      - Trimetric
    - Multiview Orthographic
- Perspective Projection
  - Single Pt
  - 2 Point
  - 3 Point





Orthographics

Multiview (x=0, or y=0 or z=0 planes)

One view is not adequate

On z=0 plane

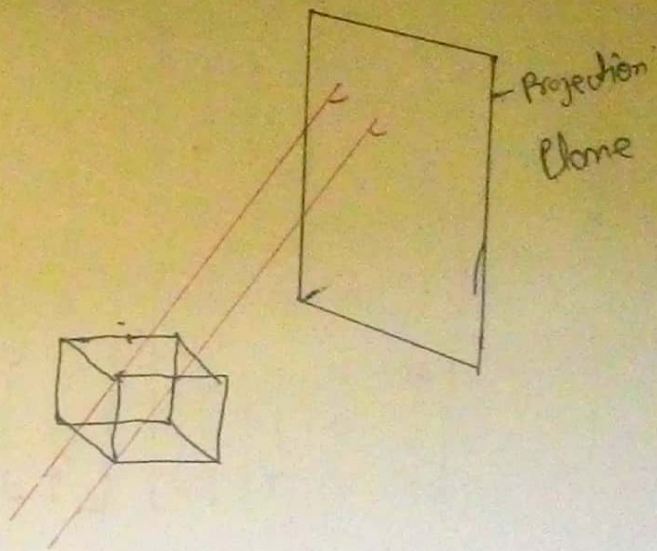
$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Oblique Projection

→ It is method of drawing object in 3 dimension

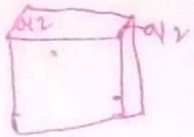
→ Non perpendicular projectors to the plane of projection

→ True shape and size for the faces parallel to projection plane is preserved



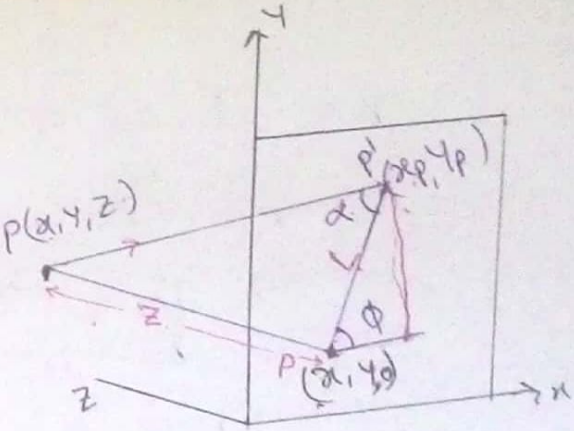
When $\alpha = 45°$ → Cavalier (Lines perpendicular to projection plane are not foreshortened)

When $\cot\alpha = 1/2$ → Cabinet



Lines perpendicular to projection plane are foreshortened by half

ϕ is typically 30° or 45°



$$P = \begin{bmatrix} 1 & 0 & L\cos\phi & 0 \\ 0 & 1 & L\sin\phi & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$x_p = x + L\cos\phi$

$y_p = y + L\sin\phi$

$\tan\alpha = z/L$

$\cot\alpha = 1/z$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p & q & r & 1 \end{bmatrix}$$

$p = -1/x_c$

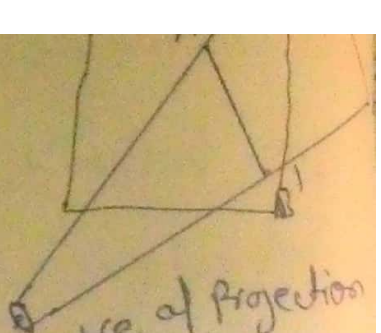
$q = -1/y_c$

$z = -1/z_c$

1) One

2) Two

3) Three



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & r & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ rz+1 \end{bmatrix} \Rightarrow \begin{bmatrix} x/(rz+1) \\ y/(rz+1) \\ z/(rz+1) \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

$$x' = \frac{x}{1 - z/z_c} = \frac{x}{rz+1}$$

$$\Rightarrow \boxed{r = -1/z_c}$$

$$\frac{x'}{x} = \frac{z_c}{z_c - z}$$

$$\boxed{x' = \frac{x}{1 - \frac{z}{z_c}}}$$

$$\Rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/z_c & 1 \end{bmatrix} - \text{matric}$$

$p = -1/x_c$

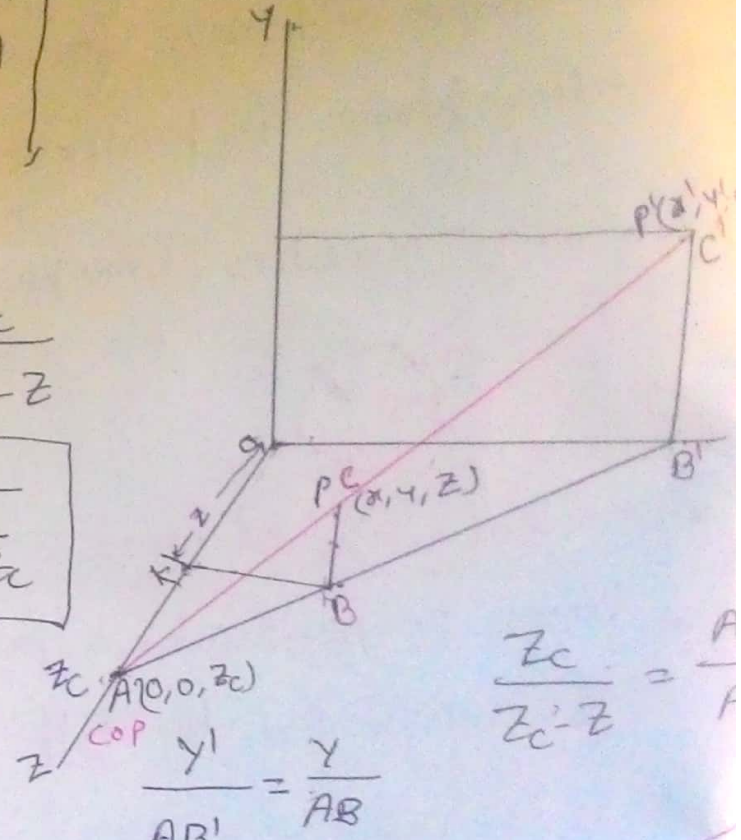
$\Rightarrow q = -1/y_c$

$VP = (0, 0, -z_c)$

$z = 0$ Plan $\Rightarrow$ COP $\Rightarrow (0, 0, z_c)$

$x = 0$ = y $(x_c, 0, 0)$

$y = 0$ —— $(0, y_c, 0)$

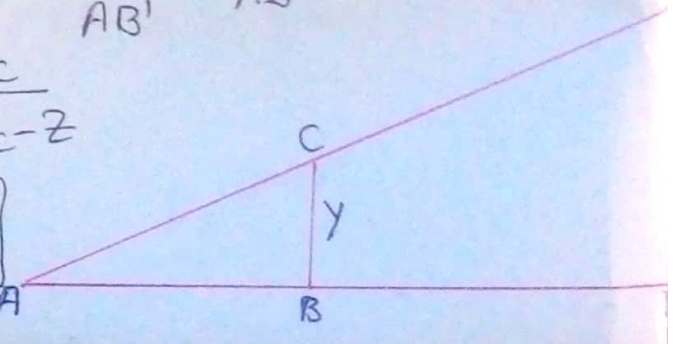


$$\frac{y'}{AB'} = \frac{Y}{AB}$$

$$\frac{z_c}{z_c - z} = \frac{A}{A}$$

$$\frac{y'}{y} = \frac{z_c}{z_c - z}$$

$$\Rightarrow \boxed{y' = \frac{y}{1 - z/z_c}}$$



Perspective Projection Example

Q:- Perform perspective Projection on to $Z=0$ plane of unit cube where center of projection is at $x_c=10$ and $Y_c=10$

Solution:

$$T_{pq} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p & q & 0 & 1 \end{bmatrix} \rightarrow T_{pq} = \begin{bmatrix} 1 & 0 & 0 & p \\ 0 & 1 & 0 & q \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(x, y, z)

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow [x\ y\ z]$$

(column → row)

$p = -Y_{xc} = -Y_{10} = 0.10$

$q = -Y_{yc} = -Y_{10} = 0.1$

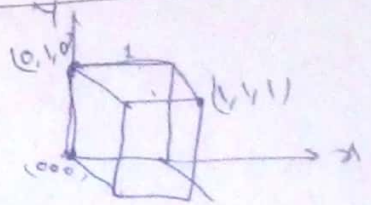
$$T = T_{pq} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -0.1 \\ 0 & 1 & 0 & -0.1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T = \begin{bmatrix} 1 & 0 & 0 & -0.1 \\ 0 & 1 & 0 & -0.1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$T_w = 0 \cdot T$

$$= \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & -0.1 \\ 0 & 1 & 0 & -0.1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0.9 \\ 1 & 1 & 0 & 0.8 \\ 0 & 1 & 0 & 0.9 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0.9 \\ 1 & 1 & 0 & 0.8 \\ 0 & 1 & 0 & 0.9 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1.11 & 0 & 0 & 1 \\ 1.25 & 1.25 & 0 & 1 \\ 0 & 1.11 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1.11 & 0 & 0 & 1 \\ 0.25 & 0.25 & 0 & 1 \\ 0 & 1.11 & 0 & 1 \end{bmatrix}$$



Shading

1) Illumination → How to color single point

Shading → How to color whole object

$$I = I_s K_d \vec{N} \cdot \vec{S}$$

Flat shading (Constant Shading)

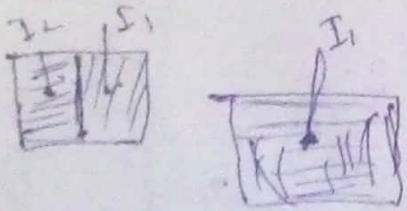
(One light calculation per polygon)

→ Each entire polygon drawn with same color

→ Color is computed once for each polygon

Drawback -

1) Mach Band effect



Gouraud shading

(Color Interpolation shading)

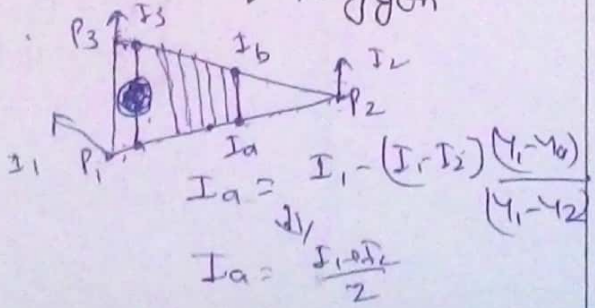
→ Color is computed once per vertex using normal of vertex

→ Colors are interpolated across polygon

Drawback:

Slower than Flat shading

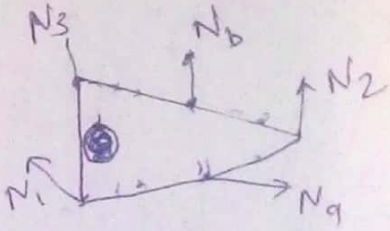
- It can miss highlights that occur in middle of polygon



Phong shading

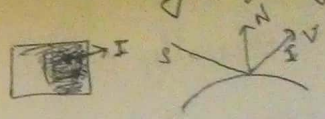
(Normal Interpolation sh

→ Normals are interpol across polygon



Illumination Model

1) Illumination models is used to calculate the intensity of light that is reflected a given point on surface.

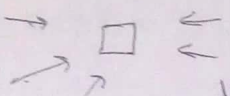


2) Rendering methods use the intensity calculations from the illumination model to determine the light intensity at all pixels in the image, by possibly, Considering light propagation between surfaces in the scene.

Light Source:

1) Point Source
2) Parallel Source — infinite
3) Distributed Source — finite ray

Ambient



$I = I_a K_a$ — diffuse reflecti

Diffuse

→ Rayghe

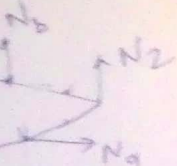
Specular

Phong Shading

nterpolation shad

are interpolate

olygon



Ambient light reflection

- Ambient light is assumed to impinge equally on all surfaces from all direction

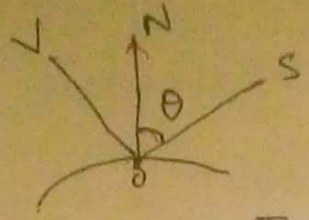
$I = I_a K_a$

I_a → Intensity of ambient light

K_a → ambient reflection coefficient

$(0 \le K_a \le 1)$

Diffuse reflection

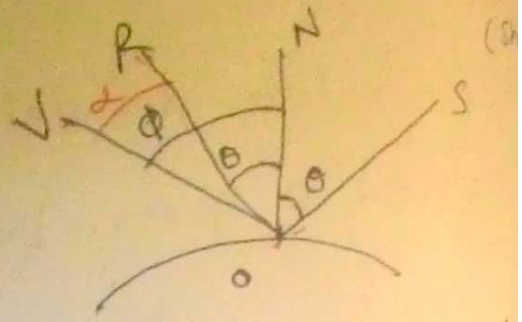


$I = I_s K_d \cos\theta$

$I = I_s K_d \vec{N} \cdot \vec{S}$

For a Lambertion surface (Dull) the amount of light seen by viewer is independent of viewer direction

$0 \le K_d \le 1$ and $0 \le \theta \le 90$



R → reflection vector (all li

$I = I_s K_s \cos^n \alpha$

n → specular intensity

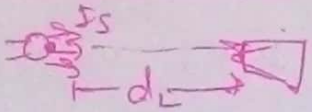
$R = 2N(N \cdot S) - S$

$\cos\alpha = R \cdot V$

more realistic

ambient + diffu.

$I = I_a K_a + I_s K_d \vec{N} \cdot \vec{S}$



$f_{ad1} = \frac{1}{d_L^2}$

$I = f_{ad} I_s K_d \vec{N} \cdot \vec{S}$

Phong Illumination

I = ambient + diffuse + sp