

Digital Differential Analyzer (DDA) Line Algorithm

This algorithm works on principle 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 $(x_n, \text{round}(y_n))$

$$x_0 = x_n, \quad y_0 = y_n$$

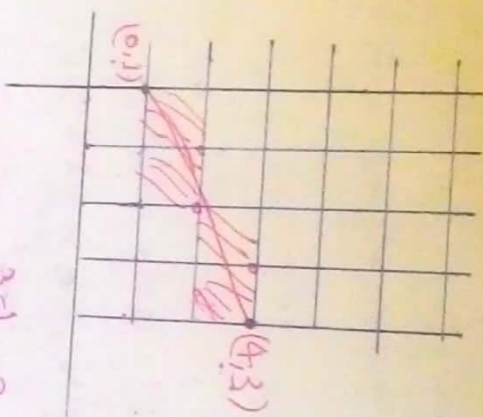
Case 2:

$m > 1$

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

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

$$y_0 = y_n, \quad x_0 = x_n$$



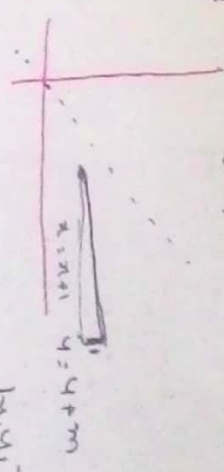
$$m = \frac{3-1}{4-0} = \frac{2}{4} = 0.5$$

Plot

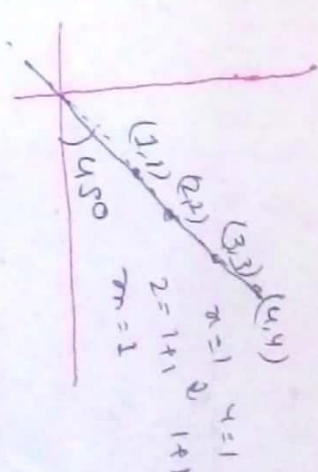
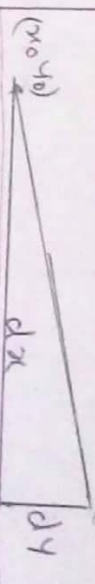
x_n	$\text{round}(y_n)$	x_n	y_n
0	1	0	1
1	2	1	1.5
2	2	2	2
3	3	3	2.5
4	3	4	3

Line

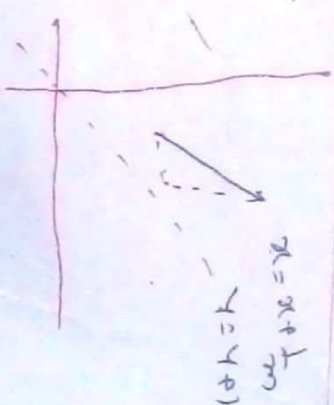
$$y = mx + c$$



$m < 1$



$m > 1$



Mid-Point Line Algorithm (Proof)

(x_1, y_1) (x_2, y_2)

Mid point Line Algorithm

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

$$m = (x_1 + x_2) / 2$$

$$d = a(x_1 + x_2) + b(y_1 + y_2) + c = a(x_1 + x_2) + b(y_1 + y_2) + c = a + \frac{b}{2}$$

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

$$x \cdot dy - y \cdot dx + c \cdot dx = 0 \quad \text{--- (ii)}$$

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

$$-d < 0 \rightarrow E$$

$$d > 0 \rightarrow NE$$

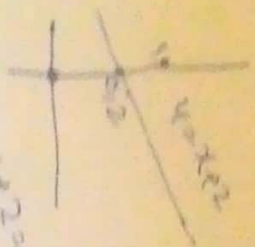
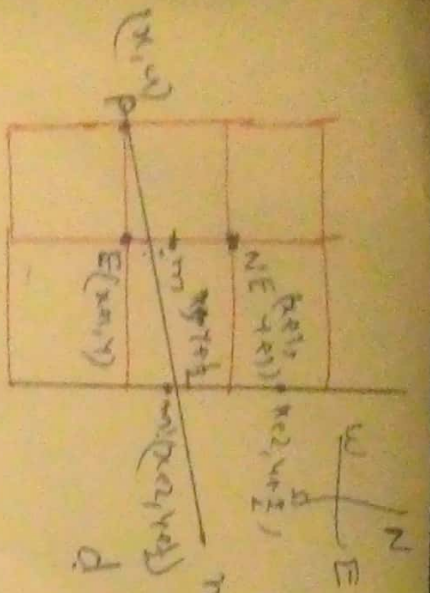
$$\rightarrow E \rightarrow (x+1, y)$$

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

$$\Delta E = a = dy$$

$$\rightarrow NE (x+1, y+1)$$

$$\Delta NE = dy - dx$$



$$F(x, y) = 0 \rightarrow dx = 0$$

$$F(x, y) = 0 \rightarrow dy = 0$$

$$F(x, y) = 0 \rightarrow dx = 0$$

$$F(x, y) = 0 \rightarrow dy = 0$$

end
 Putpixel(x, y);
 end while

d = d + ΔNE; x = x + 1; y = y + 1

else

d = d + ΔE; x = x + 1

if (d ≤ 0)

while (x < x2)

Putpixel(x, y);

x = x1, y = y1;

ΔNE = 2(dy - dx);

d = 2dy - dx, ΔE = 2dy;

dx = x2 - x1; dy = y2 - y1;

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)

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

$\left(y - \frac{1}{2}\right)^2 = y^2 - y + \frac{1}{4}$
 $= y^2 - y$

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

when $E \rightarrow d < 0$

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

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

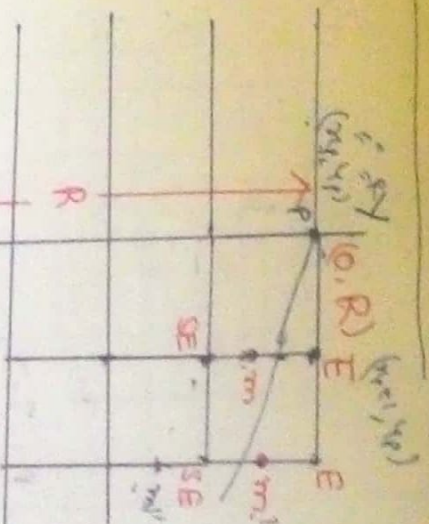
$$\Delta E = d_{new} - d = 2x_p + 3$$

when SE selected:

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

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

Mid Point Circle - Algorithm



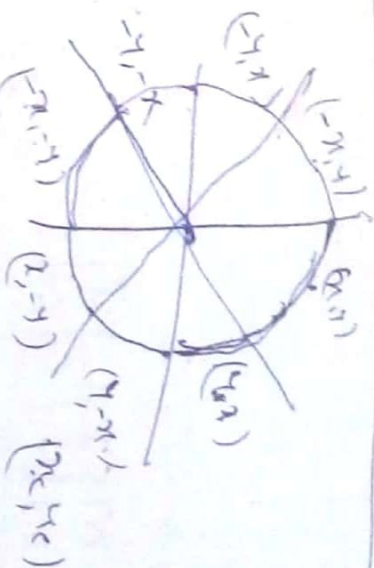
$$x^2 + y^2 - R^2 = 0$$

For a given point (x, y)

$F(x, y) = 0 \rightarrow$ on circle

$F(x, y) > 0 \rightarrow$ Outside circle

$F(x, y) < 0 \rightarrow$ Inside circle



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

Algorithm



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

Subpixel $(x, y);$

while $(y > x)$ do

if $(d < 0)$

$$d = d + 2x + 3; x = x + 1;$$

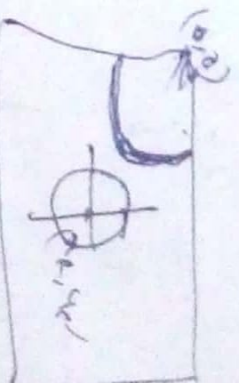
else

$$d = d + 2x - 2y + 5; x = x + 1; y = y - 1;$$

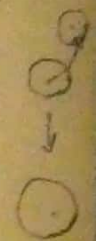
end

Subpixel $(x, y);$

endwhile



2D Transformation



Transformation

1) Translation

$$x' = x + t_x$$

$$y' = y + t_y$$

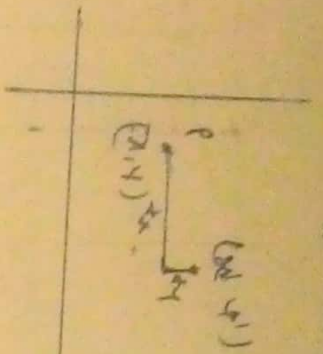
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

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

$$P = 2, 3$$

$$t_x = 4 \quad t_y = 3$$

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

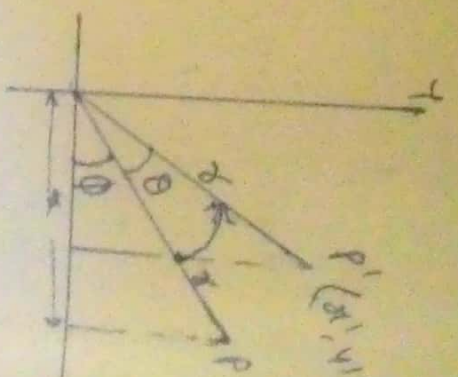


Rotation

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

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

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



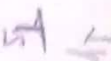
Scaling

$$x' = S_x x$$

$$y' = S_y y$$



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



$$P(2, 3)$$

$$S_x = 2 \quad S_y = 2$$

$$\text{Area } T_5 \begin{bmatrix} x_1 & x_2 \\ y_1 & y_2 \end{bmatrix} \dots$$

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

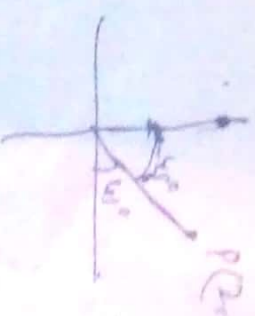
Proof

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

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

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

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



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos 45^\circ & -\sin 45^\circ \\ \sin 45^\circ & \cos 45^\circ \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} \cdot 2 - \frac{\sqrt{2}}{2} \cdot 3 \\ \frac{\sqrt{2}}{2} \cdot 2 + \frac{\sqrt{2}}{2} \cdot 3 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} \cdot 1 \\ \frac{\sqrt{2}}{2} \cdot 5 \end{bmatrix}$$

$$y' = 2\sqrt{2}$$

ode equation -

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

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

$$dE = F(x_{p+1}, y_p) = (x_{p+1})^2 + y_p^2 - R^2 \quad [+ve]$$

$$dSE = F(x_{p+1}, y_{p-1}) = (x_{p+1})^2 + (y_{p-1})^2 - R^2 \quad [-ve]$$

$d = d_E + d_{SE}$ // 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 // $d_E > d_{SE}$

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

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

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

$$d_{new}^E = (x_p + 2)^2 + y_p^2 - R^2 + (x_p + 2)^2 + (y_p - 1)^2 - R^2$$

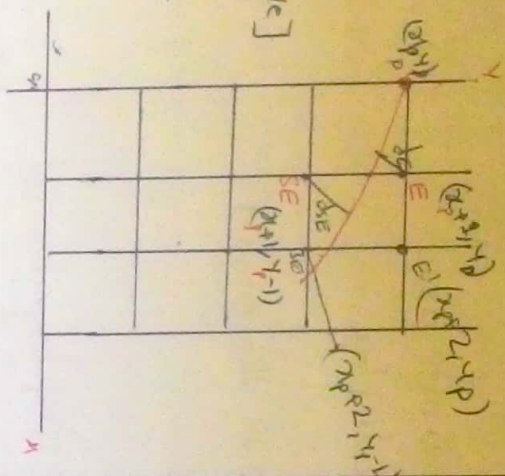
$$d_{new}^E = 9 + 8x - 2y$$

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

$$d^E = d + \Delta E$$

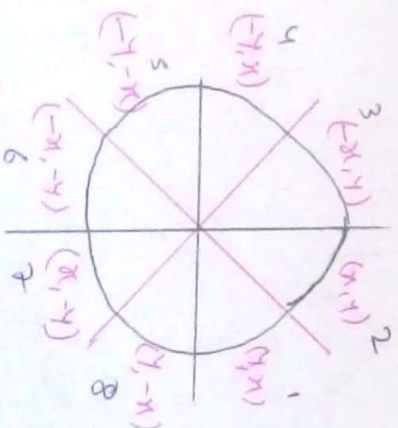
$$\Delta SE \Rightarrow d^{SE} = 4(x - y) + 10$$

Bresenham Circle Algorithm



$F(x, y) > 0$ - Outside

$F(x, y) < 0$ - Inside



Algorithm:

$x = 0, y = R; \quad P = 3 - 2R$

putPixel(x, y);

while ($y > x$) do

if $x \neq y$;

if ($d < 0$)

$d = d + 4x + 6$;

else

$d = d + 4(x - y) + 10; y--$;

end

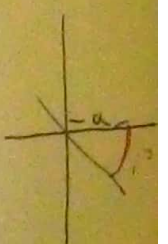
putPixel(x, y);

end while

(x, y) putPixel($x_c + x, y_c + y$);

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

$—$ ($x_c + y, y_c - x$);



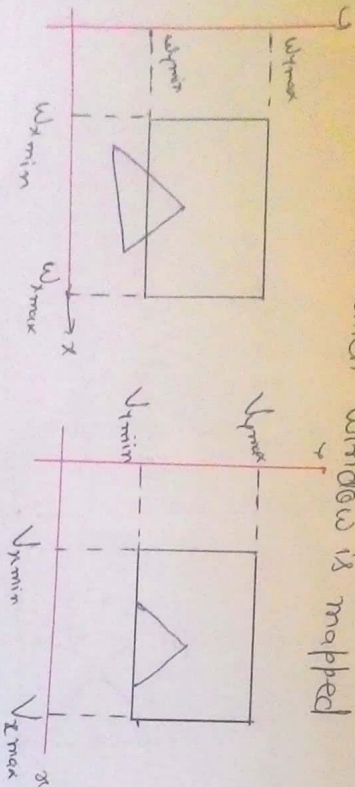
Window and View Port

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 View Port transformation

$$f_x = \frac{V_{x_{max}} - V_{x_{min}}}{W_{x_{max}} - W_{x_{min}}}, \quad f_y = \frac{V_{y_{max}} - V_{y_{min}}}{W_{y_{max}} - W_{y_{min}}}$$

$$if (f_x < f_y)$$

$$then f = f_x$$

$$else f = f_y$$

Let $V_x = V_{x_{max}}, W_x = W_{x_{max}}, V_y = V_{y_{max}}, W_y = W_{y_{max}}$

$$f = \frac{V_x - V_{x_{min}}}{W_x - W_{x_{min}}}, \quad f = \frac{V_y - V_{y_{min}}}{W_y - W_{y_{min}}}$$

Ex Window port (100, 100, 300, 300)

View port (50, 50, 150, 150)

Convert Window port coordinate (200, 200)

to View port

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

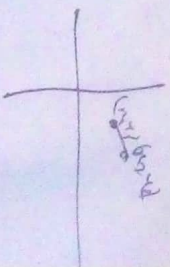
$$\boxed{V_x = 100}$$

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

Each position (x, y, w)
All transform
Composite transform
unitizing reference

Translation: T_x

$$x = (x, y) \rightarrow (x+1, y)$$



Homogeneous 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

$$(x, y, w) \quad R_2(y, 2, 2) = (8, 4, 4)$$

Translation:

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

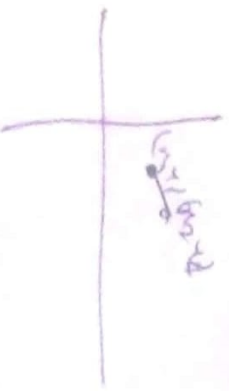
$$x = (x, y) \rightarrow x' = (x', y')$$

$$(x, y, 1) \rightarrow (x', y', 1)$$

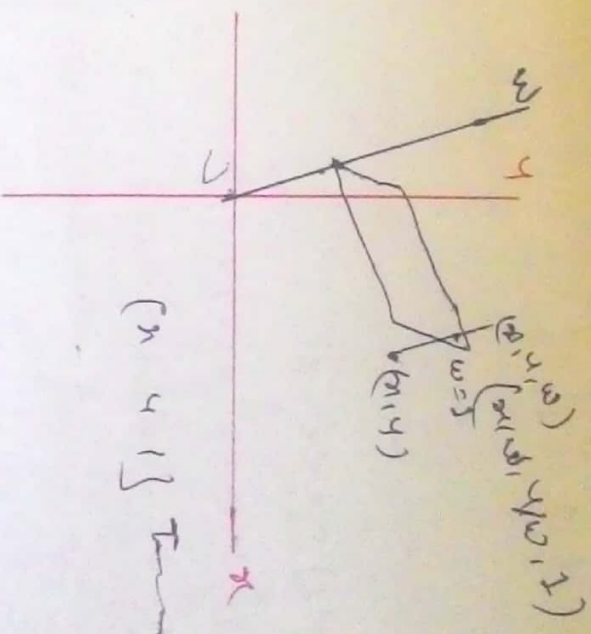
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Scaling:

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



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



$$(x, y, 1) \quad T_{\text{translation}}$$

Rotation:

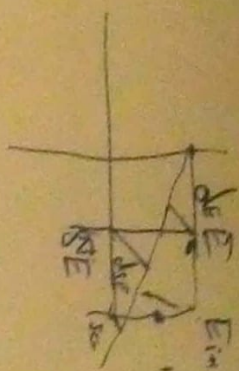
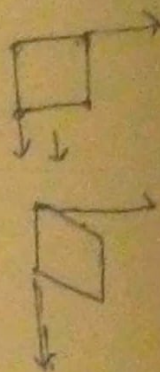
$$\begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Shearing

$$x' = x + k_y y$$

$$y' = y$$

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



$$d_{new} = f(x+1, y) = (x+1)^2 + y^2 - R^2 \quad (\text{ave})$$

$$d_{old} = f(x, y) = x^2 + y^2 - R^2 \quad (\text{ve})$$

If it is inside the circle, < 0 -ve =

Outside > 0 +ve

$$d = d_{SE} \circ d_{TE} = x$$

$$d(x, y) = 11E$$

$$d_{new} = (x+1)^2 + y^2 - R^2$$

0

$$d_{new} = d_{TE} + d_{SE}$$

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

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

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

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

$$d = 9 + 8x - 2y$$

$$\Delta dE = d_{new} - d_{old}$$

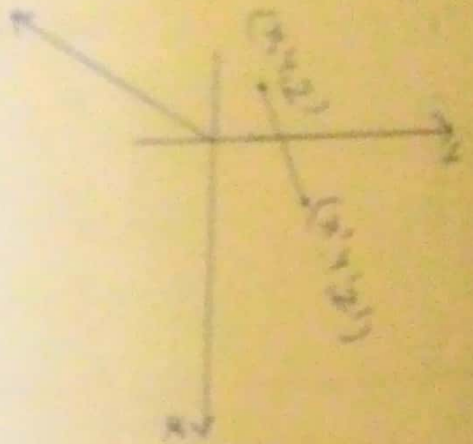
$$= 9 + 8x - 2y - 3 - 4x - 2y = 6 + 4x$$

3-D Transformation with Homogeneous Coordinates

Translation

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

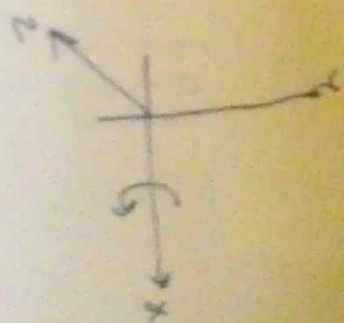
$P' = T \cdot P$



Rotation

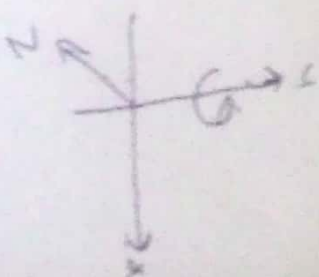
Y-axis

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



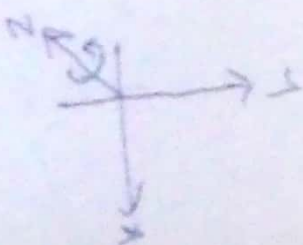
Y-axis

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



Z-axis

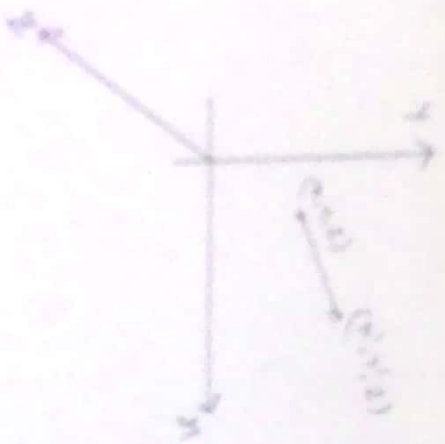
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Scale

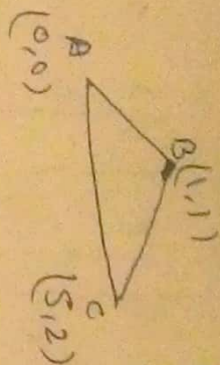
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$P' = S \cdot P$



Q2: Perform a 45° rotation of Triangle A(0,0) B(1,1) and C(5,2)

(i) about origin



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

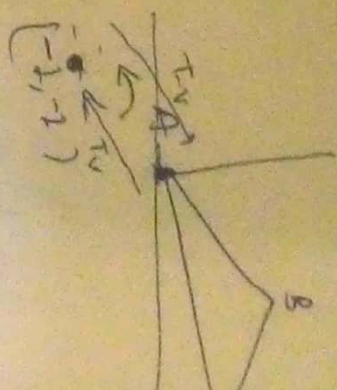
$$T_{R_0} = \begin{bmatrix} \cos 45^\circ & -\sin 45^\circ & 0 \\ \sin 45^\circ & \cos 45^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$P' = T_{R_0} \cdot P = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

$$P' = \begin{bmatrix} 0 & 0 & \frac{3\sqrt{2}}{2} \\ 0 & \sqrt{2} & \frac{3\sqrt{2}}{2} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x'_A & x'_B & x'_C \\ y'_A & y'_B & y'_C \\ 1 & 1 & 1 \end{bmatrix}$$

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

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



$$P' = T_N T_{R_0} T_N^{-1} \cdot P$$

$$= \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{52} & -x_{52} & 0 \\ x_{52} & x_{52} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

$$P' =$$

Clipping

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

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 $C_0 C_1 \neq 0000 \rightarrow$ rejected

else Clip

if line crossed x_{\min} or x_{\max}

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

$$\text{Here } x = x_{\min} \text{ or } x_{\max}$$

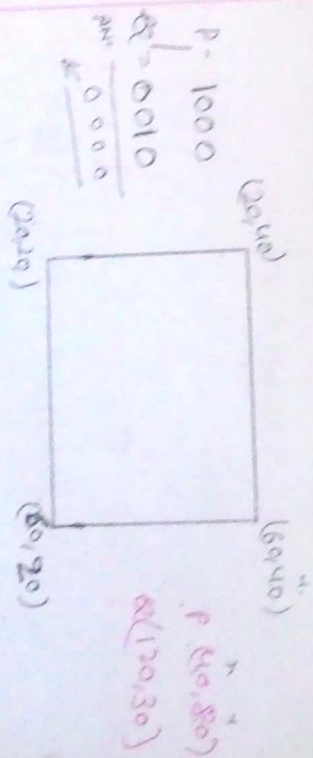
else (y_{\min} or y_{\max})

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

Free $y = y_{\min}$ or y_{\max}

Verify $x_{\min} \leq x \leq x_{\max}$? if not adjust $y_{\min} \leq y \leq y_{\max}$ & repeat clip

Cohen Sutherland Line Clipping Algorithm



Bezier Curve

Bezier curve use a construction 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

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

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

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

Let $n=3$ { 4 Control Point }

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

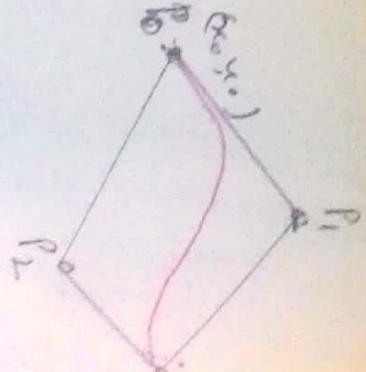
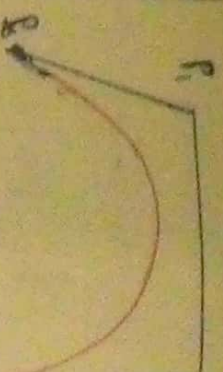
$$B(u) = P_0(1-u)^3 + P_1 3u(1-u)^2 + P_2 3u^2(1-u) + P_3 u^3$$

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

$$x(u) =$$

$$\left. \begin{array}{l} 0 \leq u \leq 1 \\ u=0.1 \\ u=0.2 \\ u=0.3 \end{array} \right\} \text{for } u \leq 1$$

$$\begin{aligned} B_{0,3}(u) &= \frac{3}{C_0} u^0 (1-u)^3 \\ &= (1-u)^3 \\ B_{1,3}(u) &= 3u(1-u)^2 \\ B_{2,3}(u) &= 3u^2(1-u) \\ B_{3,3}(u) &= u^3 \end{aligned}$$



Cohen Sutherland Clipping - Example

$\alpha \Rightarrow$ 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 $Q(15,5)$

$$Q(15,5) \rightarrow \begin{matrix} T & B & R & L \\ 0 & 1 & 0 & 0 \end{matrix}$$

$$\begin{aligned} T &\rightarrow y > y_{max} \\ B &\rightarrow y < y_{min} \\ R &\rightarrow x > x_{max} \\ L &\rightarrow x < x_{min} \end{aligned}$$

$$P(15,15) \rightarrow 0000$$

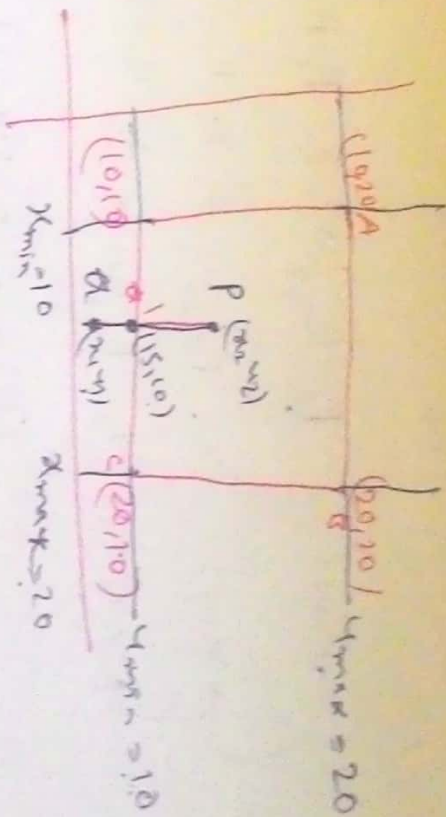
$$P \& \alpha = 0 \rightarrow$$

$$x = x_1 + (y_1 - y_2) / m$$

$$= 15 + (10 - 5) / 0$$

$$\boxed{x = 15}$$

$$(15, 10)$$



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

Example

Bezier Curve

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

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

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

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

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

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

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

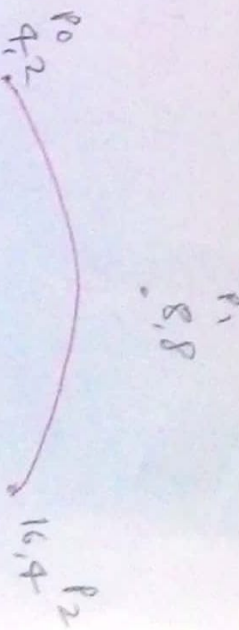
$$B_{0,2}(u) = (1-u)^2$$

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

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

if

u	x(u)	y(u)
u=0	4	2
u=0.2	5.76	4.0
u=0.4	7.84	5.20
u=0.6	10.24	5.6
u=0.8	12.96	5.2



$$P_2 = (16, 4)$$

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

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

$$= \frac{1}{1} u^0 (1-u)$$

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

$$= u^2$$

$$P_2 = (16, 4)$$

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

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

Control points $\rightarrow n+1$, and order of curve $\rightarrow K$

Curve is made up of $n-K+2$ segments

Each segment
 Influenced by K points

Basis function:

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

Ex $\rightarrow n=5$ and $K=3$

t_i ($0 \leq i \leq 8$) 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 ($0 \leq i \leq n+K$) - Knot values

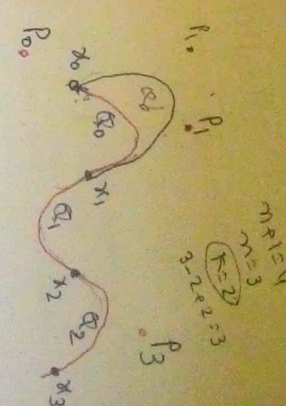
$t_i = 0$ if $i < K$

$t_i = i-K+1$ if $K \leq i \leq n$

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

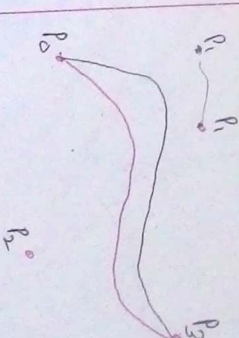
$$N_{i,K}(u) = 1 \text{ if } t_i \leq u < t_{i+1} \\ = 0 \text{ otherwise}$$

B-spline (Local Control)



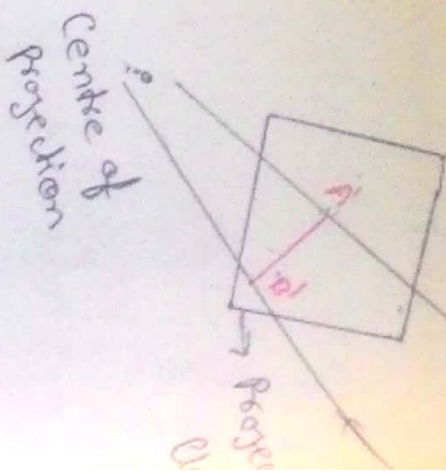
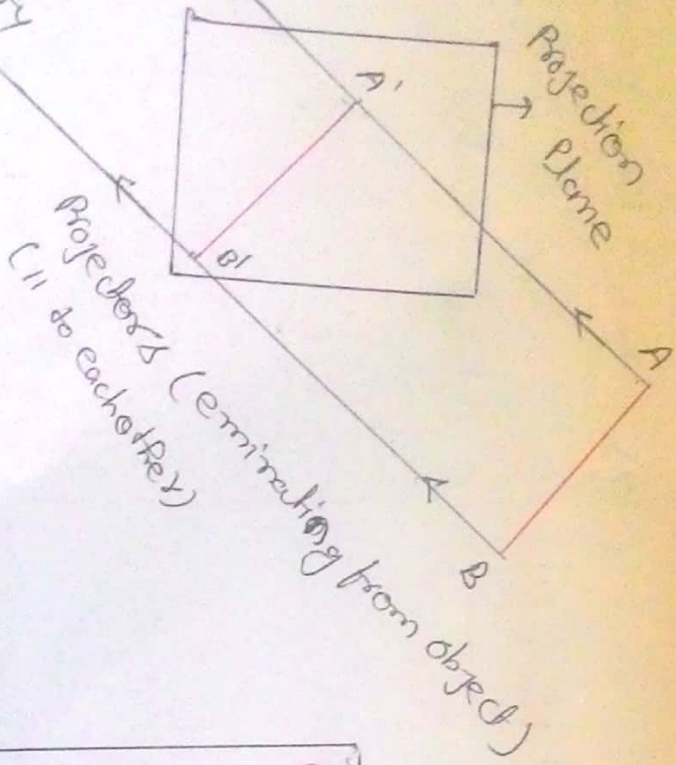
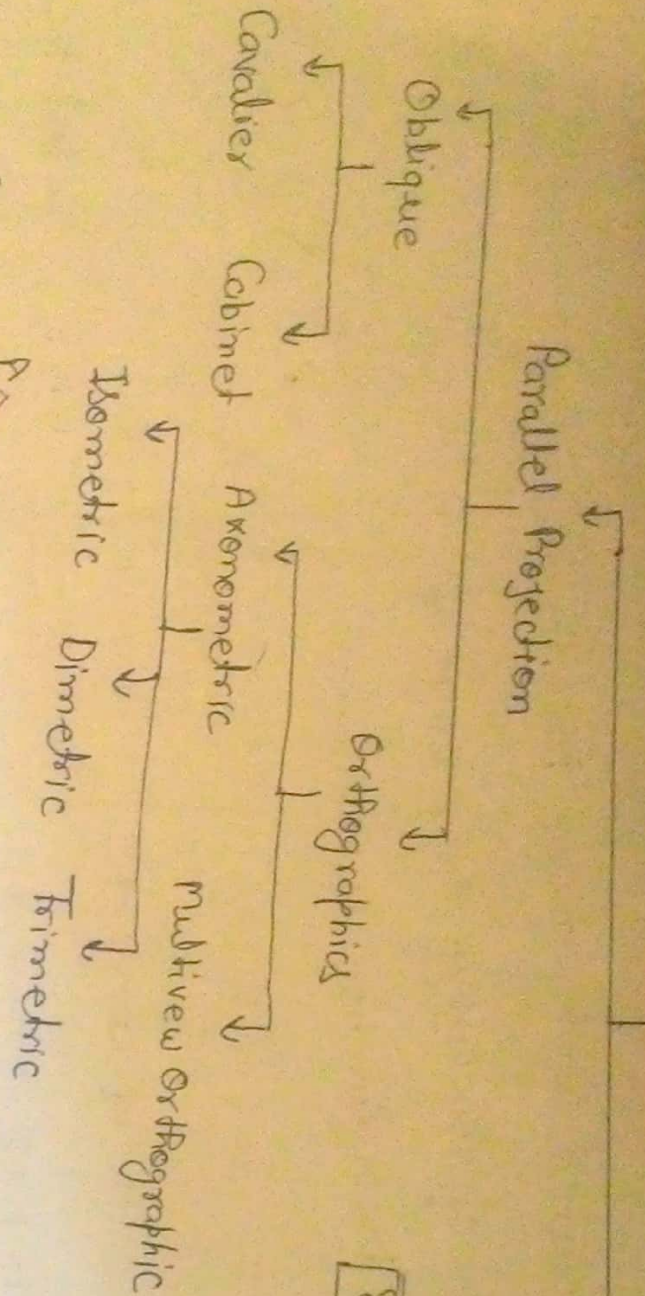
$K=2$
 P_0, P_1, P_2, P_3 - Control Points
 x_0, x_1, x_2, x_3 - Knot Value

Bezier Curve (Global Control)



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

Projection



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 $\alpha = 45^\circ \rightarrow$ Cavalier (lines perpendicular to projection plane are not foreshortened)

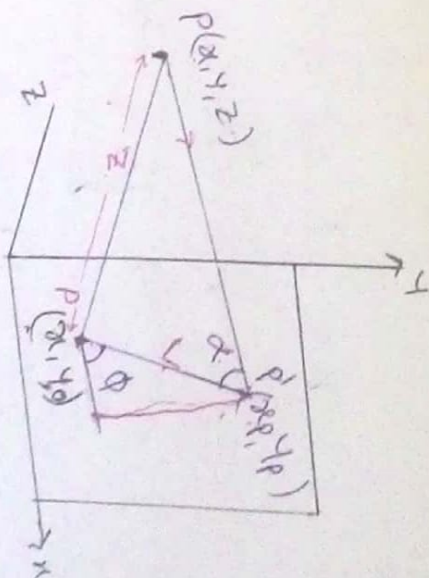
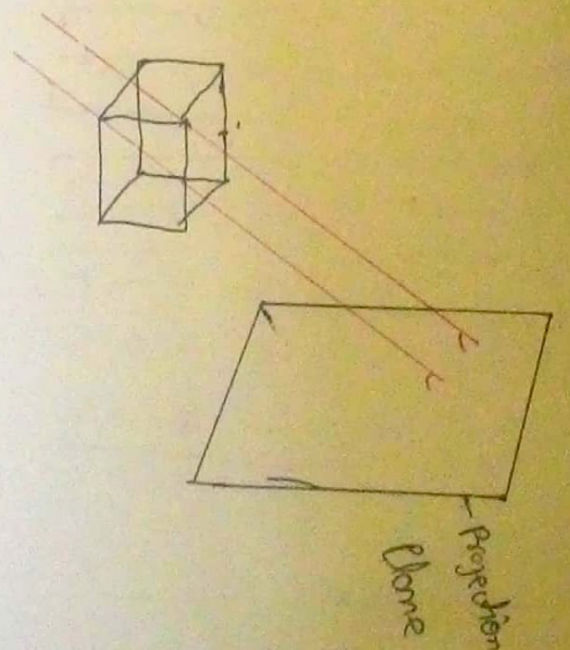


When $\cot \alpha = 1/2 \rightarrow$ Cabinet

Lines perpendicular to projection plane are foreshortened by half

ϕ is typically 30° or 45°

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



$$x_p = x + L \cos \phi$$

$$y_p = y + L \sin \phi$$

$$\tan \alpha = z/L$$

$$\cot \alpha = L/z$$

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

$$p = -\gamma/z_c$$

$$q = -\gamma/z_c$$

$$r = -\gamma/z_c$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \gamma z + 1 \end{bmatrix} \Rightarrow$$

$$\begin{bmatrix} x/\gamma z + 1 \\ y/\gamma z + 1 \\ z/\gamma z + 1 \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

- 1) One
- 2) Two
- 3) Three

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

$$\Rightarrow \begin{bmatrix} x' = -\gamma/z_c \end{bmatrix}$$

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

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

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

matrix

$$p = -\gamma/z_c$$

$$q = -\gamma/z_c$$

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

$$\Rightarrow COP \Rightarrow (0, 0, z_c)$$

$$(x_c, 0, 0)$$

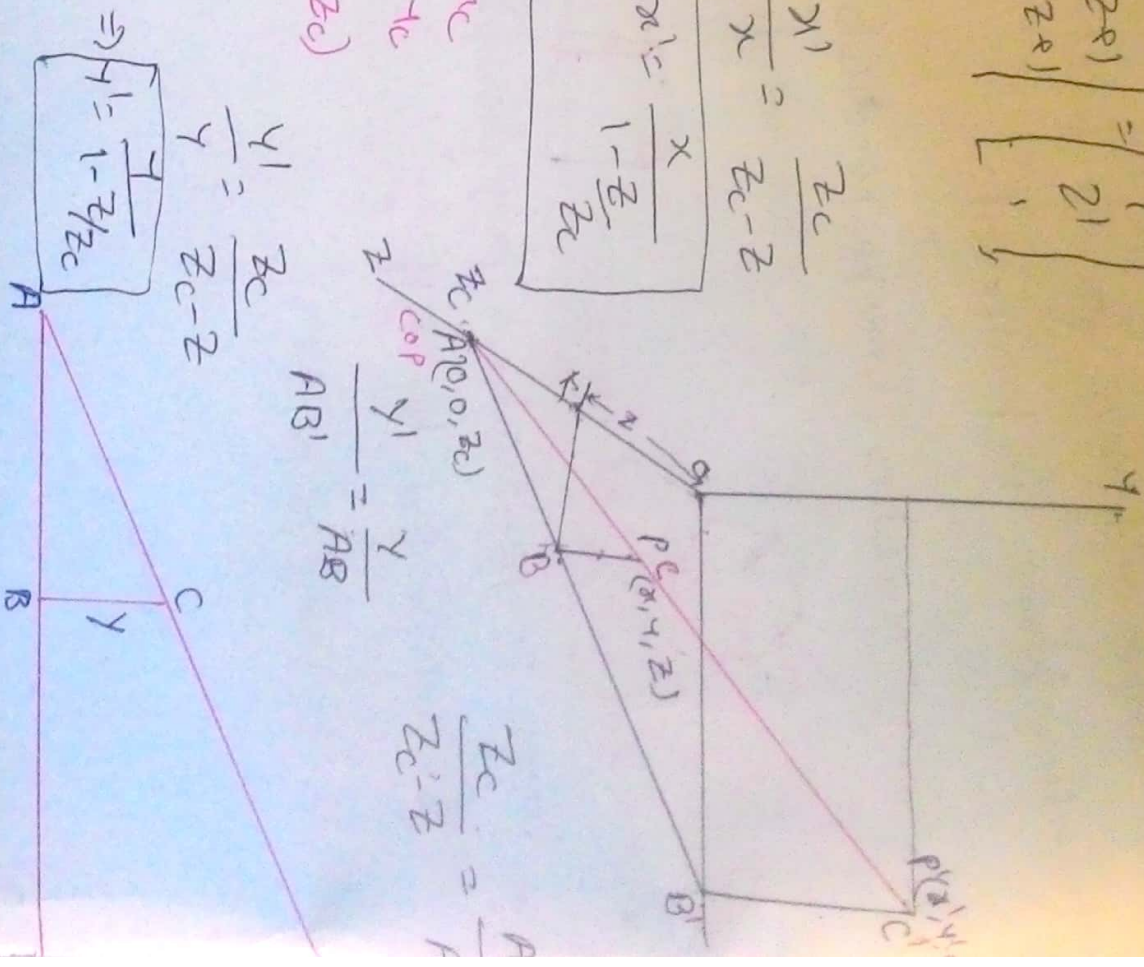
$$(0, y_c, 0)$$

$$z = 0 \text{ Plane}$$

$$x = 0$$

$$y = 0$$

Centre of Projection



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

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

Perspective Projection Example

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:

$$T_{p1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p & q & 0 & 1 \end{bmatrix}$$

→

$$T_{p2} = \begin{bmatrix} 1 & 0 & 0 & p \\ 0 & 1 & 0 & q \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T = T_{p2} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

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

$$p = -x_c = -10 = -0.10$$

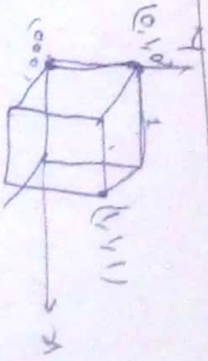
$$q = -y_c = -10 = -0.1$$

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

~~Ans~~

$$T_{w1} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \times$$

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



$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0.9 \\ 0 & 1 & 0 & 0.8 \\ 0 & 0 & 1 & 0.9 \end{bmatrix}$$

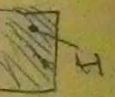
$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1.11 & 0 & 0 & 1 \\ 1.25 & 1.25 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0.25 & 0.25 & 0 & 1 \\ 0 & 1.11 & 0 & 1 \end{bmatrix}$$

Shading

1) Illumination → How to color single point

Shading → How to color whole object

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



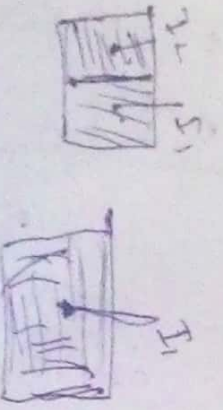
Flat Shading (Constant Shading)

(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

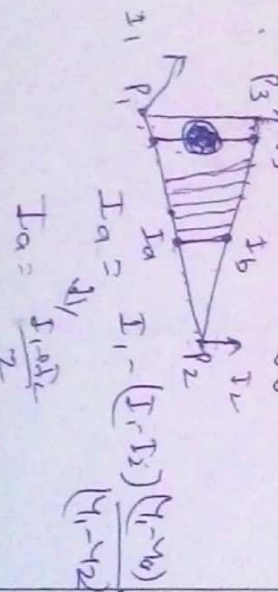
(Color Interpolation shading)

- Color is computed once per vertex using normal of vertex
- Colors are interpolated across polygon

Drawback:

Slower than Flat shading

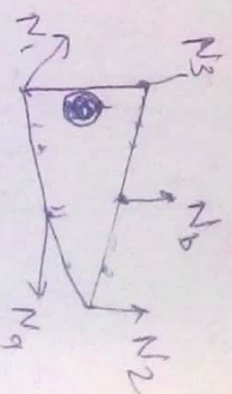
→ As can miss highlights that occur in middle of polygon



Phong Shading

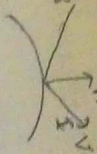
(Normal Interpolation shading)

- Normals are interpolated across polygon



Illumination Model




diffuse
specular
micro



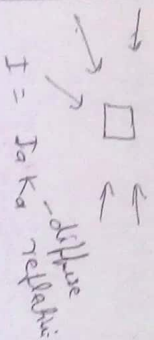
1) Illumination models is used to calculate the intensity of light that is reflected a given point on surface.

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.

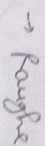
Light Source:

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

Ambient



Diffuse



Specular

Ambient light reflection

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

$$I = I_a K_a$$

$I_a \rightarrow$ Intensity of ambient light

$K_a \rightarrow$ ambient reflection coefficient
($0 \leq K_a \leq 1$)

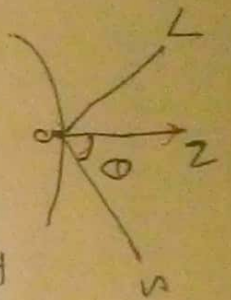
more realistic

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



ambient & diff.

Diffuse reflection



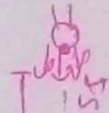
$$I = I_s K_d \cos \theta$$

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

For a lambertian surface

(Dull) the amount of light seen by viewer is independent of viewer direction

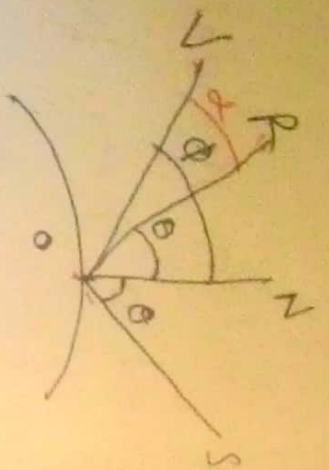
$$0 \leq K_d \leq 1 \text{ and } 0 \leq \theta \leq 90$$



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

$$f_{ad1} = \frac{1}{d_1^2}$$

Specular reflection



$R \rightarrow$ reflection vector (all)

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

$n \rightarrow$ specular intensity

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

$$\cos \alpha = R \cdot V$$

Phong Illumination

$$I = \text{Ambient} + \text{diffuse} + \text{specular}$$