Computer Graphics

- Computer graphics is an art of drawing pictures on computer screen with the help of programming. It involves computation, creation & manipulaⁿ of ~~data~~ images.

  Computer graphics is a rendering tool for the generation & manipulaⁿ of image.

- Applications : ① Computer GUI
  ② Cartography (drawing maps)
  ③ Satellite imaging
  ④ Engineering drawing
  ⑤ Educaⁿ & Entertainment
  ⑥ Training (flight simulaⁿ, CAD)
  ⑦ Simulation & modelling

LINE GENERAⁿ ALGORITHM

- DDA algorithm. (Digital differential analyzer)

```
void DDA (int x0, int y0, int x1, int y1) {
    int dx = x1 - x0 ;
    int dy = y1 - y0 ;
    int steps = abs(dx) > abs(dy) ? abs(dx) : abs(dy);
    float xInc = dx /(float)steps;
    float yInc = dy /(float) steps;
    float x = x0, y = y0 ;
    for (int i=0 ; i <=steps ; i++) {
        putPixel ( x, y, RED); // pass Round(x) & Round(y)
        x += xInc;
        y += yInc;
    }
}
```

Iteraⁿ	x	y	Plotted pts (x/y)

Adv : • Faster than direct use of line eqⁿ.
• multiplicaⁿ is not involved
• Easy as each step involves just 2 addiⁿ's.

Disadv : • It involves costly roundoffs & floating point operaⁿ.
• suitable for line generaⁿ using s/w but not h/w.
• Due to round of error is added in it.

- Bresenham line drawing algorithm :-

① Find Δx, Δy,

② Calculate $D_0 = 2\Delta y - \Delta x$

$\Delta x = |x_2 - x_1|$, $\Delta y = |y_2 - y_1|$

③ Start at iteran $K = 0$,

at each x_K, do the following :-

↳ if $D_K < 0$, plot (x_{K+1}, y_K)

$D_{K+1} = D_K + 2\Delta y$

↳ else, plot (x_{K+1}, y_{K+1})

$D_{K+1} = D_K + 2(\Delta y - \Delta x)$

④ Repeat step 3, Δx no. of time

↳ (0 to $\Delta x - 1$)

table $D_0 = 2\Delta y - \Delta x$

Iteran	x	y	Decision
1	.	`	$D_1 = D_0 + \ldots$

Adv : • Involves only integer arithmetic
• avoids genera^n of duplicate points.
• It can be implemented using h/w as it does not use mul & div.
• It is faster than DDA (no floating pt. calc.)

Disadv : This algo is meant for basic line drawing only. So to draw smooth lines consider other algorithms.

DDA Algorithm	Bresenham's line drawing algo.
• uses floating point arithmetic	• uses integer arithmetic
• opera^n used are * & ÷	• opera^n used are + & −
• Slower	• Faster
• Not accurate & efficient	• more accurate & efficient than DDA
• Can draw circle & curves but not as accurate as Bresenham's line algo.	• Can draw circle & curves with more accuracy than DDA algorithm.

⊙ Midpoint Circle Algorithm :-

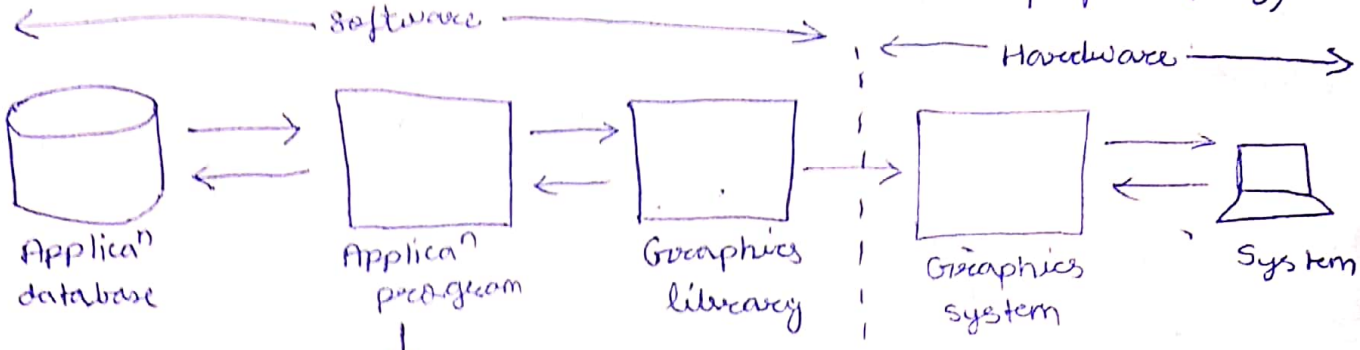
Algorithm :-

Step 1) Put $x = 0$, $y = r$, we have, $p_0 = 1 - r$

Step 2) do while ($x \leq y$) {

```
        plot (x, y)
        if (x p < 0){
            p = p + 2x + 3
        } else{
            p = p + 2(x - y) + 5
            y = y - 1 ;
        }
        x = x + 1
}
```

⊙ Bresenham's circle algorithm : $p = 3 - 2r$ | $p + 4x + 6$; $p + 4(x - y) + 10$



It maps all applica^n objects to images by invoking graphics library.

---

soln^n → ① Prefiltering ② Postfiltering.

- Antialiasing is a technique in computer graphics to remove aliasing effect.

The aliasing effect is appearance of Jagged edges or "jaggies" in rasterized image. It occurs due to distor^n of image when scan conversion is done with sampling at low frequency, also known as undersampling. (results in loss of informa^n of picture

To avoid this loss, our sampling freq is atleast twice than of highest frequency occuring in object.

↳ Min req. freq. is referred as Nyquist sampling freq.

⭐ Liang Barsky –

① Set $u_{min} = 0$, $u_{max} = 1$

② $u_k = \frac{q_k}{p_k}$ (for $k = 0, 1, 2, 3$) , value of u_k lies b/w $[0, 1]$

③ Use foll. inequality to calculate u_k & p_k.

$p_0 = -\Delta x$	$q_0 = x_0 - x_{min}$
$p_1 = \Delta x$	$q_1 = x_{max} - x_0$
$p_2 = -\Delta y$	$q_2 = y_0 - y_{min}$
$p_3 = \Delta y$	$q_3 = y_{max} - y_0$

④ To calc. intersec^ point.

$x = x_0 + u\Delta x$, $x_{min} \le x \le x_{max}$

$y = y_0 + u\Delta y$, $y_{min} \le y \le y_{max}$

⭐ Weiler atherton :–

① make a list of all intersecting points namely $I_1, I_2, \dots, I_n$

② classify those intersec^ points as entering or exiting.

③ Now make 2 list , 1 for clipping polygon & other for clipped poly.

~~④ Start from "to be clipped" polygon list.~~

④ Fill both the list in such a way that intersec^ pt. lie b/w the correct vertices of each of the polygon.

⑤ Start from "to be clipped" polygon list.

⑥ choose the first intersec^ pt which is labelled as entering point. Keep on following list until exit intersec^ pt is found

⑦ Switch the list to the polygon that is clipping list , & find the exit intersec^ point that was previously encountered.

Now keep on following the pts in the list until an entering intersec^ point is found.

⑧ Repeat this clipping procedure until all the entering intersection pts have been visited once.

PROJECTION

PARALLEL

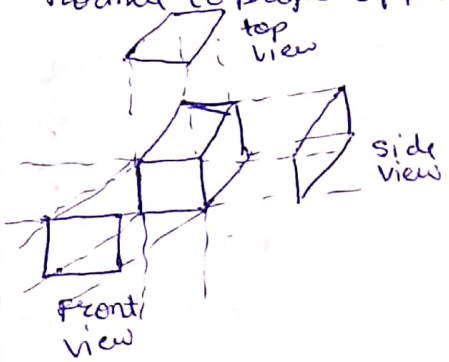
ORTHOGRAPHIC

(top view) – Top projec^n

" – Front projec^n

" – Side projec^n

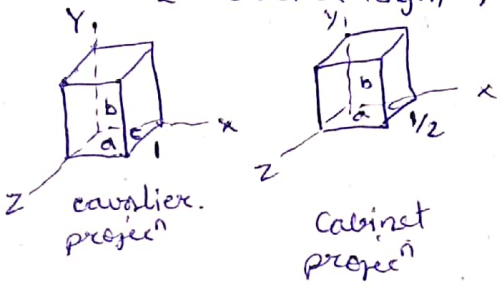
(here, direc^n of projec^n is normal to projec^n of plane)



OBLIQUE (not normal)

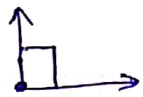
- Cavalier projec^n make 45° angle with projec^n plane.
- projec^n of line ⊥ to view plane has same length.
- cabinet (angle = 63.4°)

½ the actual length,

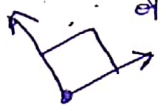


PERSPECTIVE

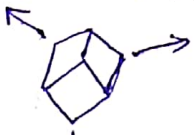
- one point (simple to draw)



- two point (gives both impression of depth)



- three point



(diff. to draw)

→ • Window to viewport transforma^n –

$$\frac{xv - xv_{min}}{xv_{max} - xv_{min}} = \frac{xw - xw_{min}}{xw_{max} - xw_{min}}$$

TBRL

• Sutherland Hodgeman :

vertex	case	o/p

LRBT

3 new vertex

$$\frac{x_v - x_{min}}{nor - x}$$

• 2D transformaⁿ means the change in either posiⁿ or orienta or size or shape of graphics objects.

translaⁿ (2D): $\begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$

shear(2D): $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

(for x direcⁿ shear)

⇒ Area filling – interior defined region or boundary defined region

⊙ Boundary fill → getPixel() / putpixel(x, y, col)

if (curr ! = newcol && curr ! = boundary color) { }

⊙ Scan fill algorithm

① Find the intersecⁿ of the scan line with all edges of polygon
② sort the intersecⁿ of pts by x from left to right.
③ Make pairs of intersecⁿ & fill in color within all the pixels.
(count point twice if direcⁿ changes)

⊙ Shadow types → self shadow & Projected shadow.

⊙ Shading → procedure to color entire surface.

- constant / Flat shading – less computaⁿ needed
- Interpolated shading
- Gouraud Shading – once per vertex
- Phong shading – once per pixel thus heavy computaⁿ's req.

⊙ Cohen Sutherland line clipping algo :- (TBRL)

① Assign the region code for 2 end points.
② If both have region code 0000 then line is completely accepted.
③ else perform logical ang operaⁿ for both region code
if result ≠ 0000, line is outside
else line is partially inside

(choose end pt outside window, find intersecⁿ pt, replace endpt with int. pt. & update region code. repeat)

3D Rotaⁿ's

$$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ tx & ty & tz & 1 \end{bmatrix}$$

translatⁿ matrix

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

scaling matrix

$$\rightarrow \begin{bmatrix} 1 & Sh^y & Sh^z & 0 \\ Sh^x & 1 & Sh^z & 0 \\ Sh^x & Sh^y & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Shear

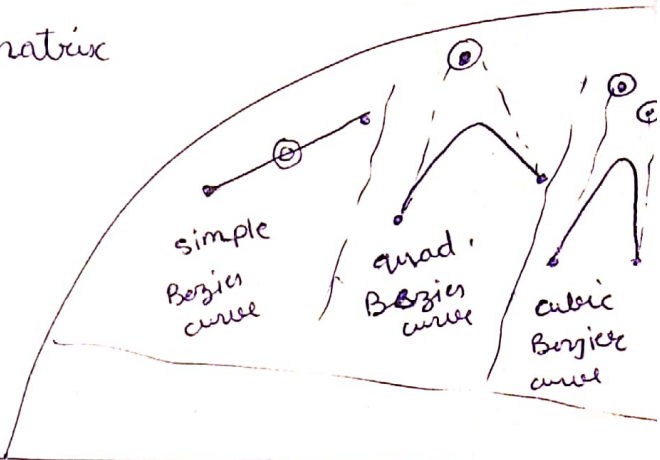
(ACW)

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

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

Rotaⁿ matrix

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



* Types of curves : (curve is ∞ large set of pts. Each pt has 2 neigh. except end pts)

① Implicit curves : procedure is given to test whether pt is on curve or not.
$F(x,y)=0$, given in terms of both dep. & indp. var. $x^2+y^2=100$

② Explicit curve : using $y = f(x)$ curve can be plotted. $y = x^2+3x-8$
(given in terms of independent var.)

③ Parametric curves : used practically as in both ① & ② func^n is needed (know)

* Bezier Curves : these curves are generated under control of other pts. Approximate tangent by using control points

are used to generate circle

Representa^n :- $\sum_{K=0}^{n} P_i B_i^n(t)$ P_i = set of pts

$B_i^n(t)$ Bernstein polynomials.

$$B_i^n(t) = \binom{n}{i}(1-t)^{n-i}t^i$$ | n = poly degree, i = index & t is variable.