Explain Sutherland Hodgman polygon clipping algorithm with suitable example and comment on its shortcoming.

- Sutherland Hodgman is a polygon clipping algorithm, which is an extension of Cohen Sutherland algo., but used for polygon clipping instead of line clipping.
- It uses a pipelined approach involving four clippers viz: left clipper, Right clipper, Bottom clipper & Top clipper respectively.
- Every clipper takes responsibility of eliminating all those vertices which are outside from its perspective and retains those vertices which are inside, in addition the clipper also produces new set of vertices representing the point of intersections.

Input set of vertices representing polygon → Left Clipper → Intermediate set of vertices → Right Clipper → Intermediate set of vertices → Bottom Clipper → Intermediate set of vertices → Top Clipper → Final (output) set of vertices representing clipped polygon

- Every clipper deals with following four cases while processing the set of vertices in clockwise or counterclockwise manner

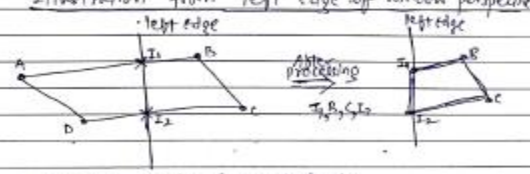
CASE 1: Outside to Inside movement
Action: • record point of intersection
• record the inside vertex

CASE 2: Inside to Inside movement
Action: • record the next inside vertex

CASE 3: Inside to outside movement
Action: • record the point of intersection

CASE 4: Outside to Outside movement
Action: • Do not record any thing

Illustration from left edge of window perspective



case 1 : Moving from A to B
↳ record I_1, B

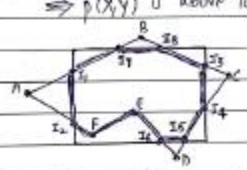
case 2 : Moving from B to C
↳ record C

case 3 : Moving from C to D
↳ record I_2

case 4 : Moving from D to A
↳ record nothing

- for any vertex $p(x,y)$
  if $(x < xw_{min})$
  ⇒ $p(x,y)$ is outside left edge (left of left)
  if $(x > xw_{max})$
  ⇒ $p(x,y)$ is outside right edge (right of right)
  if $(y < yw_{min})$
  ⇒ $p(x,y)$ is below bottom edge
  if $(y > yw_{max})$
  ⇒ $p(x,y)$ is above top edge.

Ex:



ABCDEF → left clipper → $I_1BCDEFI_2$ → Right clipper → $BI_3I_4DEFI_2I_1$ → Bottom clipper → $I_3I_4I_5I_6EFI_2I_1B$ → Top clipper → $I_4I_5I_6EFI_2I_1I_7I_8I_3$ (O/p set of vertices)

↓ limitation :

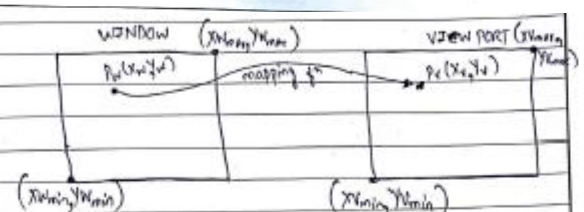
The algorithm can not give guarantee of success for all concave polygons, for some concave polygons it may generate an extraneous line connecting the two expected disjoint polygons.

Q) What is window & viewport ? Derive the window to viewport transformation and also identify the geometric transformations involved.

A mapping from world coordinate picture definition to device coordinate specifications, typically involves concept of window & viewport.

- A window represents the area selected from world coordinate specifications for the purpose of display.
- Window specifies what is to be displayed
- A viewport represents the area on display device where we are interested in mapping the selected window.
- A viewport specifies where it is to be displayed.

- A window to viewport transformation needs to ensure that the relative position of point in window must be maintained in viewport.
- let us consider a window with extents as (XW_{min}, YW_{min}) & (XW_{max}, YW_{max}), a viewport with extents as (XV_{min}, YV_{min}) & (XV_{max}, YV_{max})
- let a point $P_w(X_w, Y_w)$ be the point in window & it needs to be mapped to $P_v(X_v, Y_v)$ in viewport.



Now to retain the relative position of a point, the following equality should hold true :

$$\frac{X_w - XW_{min}}{XW_{max} - XW_{min}} = \frac{X_v - XV_{min}}{XV_{max} - XV_{min}} \quad \text{--- (1)}$$

AND

$$\frac{Y_w - YW_{min}}{YW_{max} - YW_{min}} = \frac{Y_v - YV_{min}}{YV_{max} - YV_{min}} \quad \text{--- (2)}$$

Now from eqn (1) we get

$$X_v = XV_{min} + (X_w - XW_{min})\left(\frac{XV_{max} - XV_{min}}{XW_{max} - XW_{min}}\right)$$

$$\therefore \boxed{X_v = XV_{min} + (X_w - XW_{min}) \cdot S_x}$$

where $S_x = \left(\frac{XV_{max} - XV_{min}}{XW_{max} - XW_{min}}\right)$

Similarly from eqn (2) we get

$$\boxed{Y_v = YV_{min} + (Y_w - YW_{min}) \cdot S_y} \text{ where } S_y = \left(\frac{YV_{max} - YV_{min}}{YW_{max} - YW_{min}}\right)$$

for acheiving this using geometric transformations, the sequence of x^ns would be:

1.) Translate ~~(x_w, y_w) to~~ (xw_{min}, yw_{min}) to origin

$$T(T_x = -xw_{min}, T_y = -yw_{min})$$

2.) Apply Scaling so as to adjust the size of window to viewport

$$S\left(S_x = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}}, S_y = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}}\right)$$

3.) Translate to (xv_{min}, yv_{min})

$$T(T_x = xv_{min}, T_y = yv_{min})$$

$$\begin{bmatrix} x_v \\ y_v \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & xv_{min} \\ 0 & 1 & yv_{min} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}} & 0 & 0 \\ 0 & \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -xw_{min} \\ 0 & 1 & -yw_{min} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ 1 \end{bmatrix}$$

After solving this we get a composite Matrix same as represented by eqn (1) & eqn (2)

Explain what is meant by Bezier curve? state the various properties of Bezier curve

- Bezier curve uses piecewise approximation mechanism, & represents a curve by using set of polynomial, one per coordinate axis.
- Input will be in form of control points used to excercise control on the shape of curve.
- for $(n+1)$ control points $((x_0, y_0), (x_1, y_1), \dots (x_n, y_n))$ specified, the parametric eqn representing a bezier curve is given by:

$$\boxed{P(u) = \sum_{i=0}^{n} P_i \, B_{n,i}(u)} \quad \text{--- (1)}$$

↳ where u is a parameter in the range of 0 to 1

↳ P_i is the i^{th} control point

↳ $B_{n,i}(u)$ is a Bezier Blending fn which is given as:

$$\boxed{B_{n,i}(u) = {}^nC_i \, u^i (1-u)^{n-i}}$$

where ${}^nC_i$ is a binomial coeff. &

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

eqn (1) can be expressed in terms of x & y (& for 3D in terms of x, y & z) resp.

eg:

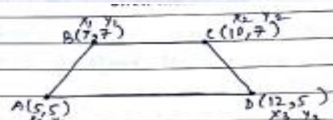
$x(u) = x_0 B_{n,0}(u) + x_1 B_{n,1}(u) + \ldots\ldots + x_n B_{n,n}(u)$

$y(u) = y_0 B_{n,0}(u) + y_1 B_{n,1}(u) + \ldots\ldots + y_n B_{n,n}(u)$

these polynomials can be solved for different values of u in the range of 0 to 1, by deciding the step size.
All generated points $(x(u), y(u))$ then can be connected by using straight line segments.

Properties:
1) degree of polynomial is n for (n+1) control pts.
2) There will be (n+1) Blending functions
3) Doesn't supports local control, but supports global control.
4) The curve always passes through first & last control point.
5) The curve lies inside a convex hull, & therefore supports variation diminishing property
6) Every Blending f^n is nonzero in the range of $u: 0$ to 1 (except the extremes)
7) The line joining I^{st} & II^{nd} control point is always tangent to the curve at I^{st} control pt., Similarly the line joining last & second last control point is tangent at last control pt.
8) If I_{st} and last control pt. coinsides it results into closed curve.

eg:



let A,B,C,D be 4 control pts.

$n+1 = 4 \Rightarrow n = 3$

Polynomials will be

$\therefore\ x(u) = 5 B_{3,0}(u) + 7 B_{3,1}(u) + 10 B_{3,2}(u) + 12 B_{3,3}(u)$

$y(u) = 5 B_{3,0}(u) + 7 B_{3,1}(u) + 7 B_{3,2}(u) + 5 B_{3,3}(u)$

$\because\ B_{n,i}(u) = {}^{n}C_{i}\, u^{i} (1-u)^{n-i}$

$\Rightarrow \boxed{B_{3,0}(u) = \frac{3!}{0!\,3!} = 1}$ $\boxed{B_{3,1}(u) = \frac{3!}{1!\,2!} = 3}$

$\boxed{B_{3,2}(u) = \frac{3!}{2!\,1!} = 3}$ $\boxed{B_{3,3}(u) = \frac{3!}{3!\,0!} = 1}$

$\Rightarrow$ $x(u) = 5(1-u)^3 + 21u(1-u)^2 + 30u^2(1-u) + 12u^3$
&
$y(u) = 5(1-u)^3 + 21u(1-u)^2 + 21u^2(1-u) + 5u^3$

we need to solve these polynomials for different values of u in the range of 0 to 1 with decided step size

eg: $u=0,\ u=0.2,\ u=0.4,\ u=0.6,\ u=0.8,\ u=1$

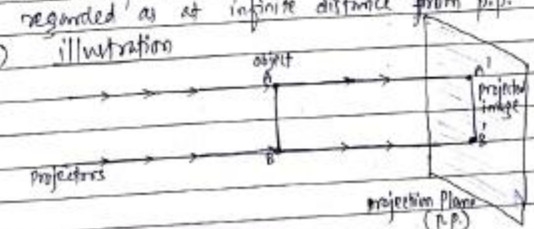
What is meant by parallel and perspective projection?

The purpose of projection is to map specifications available in n-dimensional region of space to m-dimensional region of space.

eg: 3D to 2D

Parallel Projection:

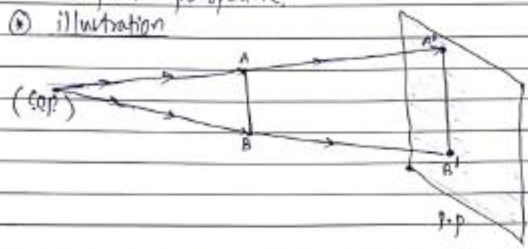
- Projectors runs parallel to each other
  - If they are perpendicular to projection (view) plane then it is orthographic otherwise it is oblique projection.
- Theoritically the c.o.p (center of projection) is regarded as at infinite distance from p.p.
- illustration



- Retains the dimensions of the object (in orthographic projection) after it gets projected.
- Doesn't provides realistic effects as perceived by human visual system.

Perspective Projection :

- Projectors tends to converge at one point called as center of projection (c.o.p.)
- c.o.p. lies at finite distance
- can be classified as 1-point, 2-point or 3-point perspective
- illustration



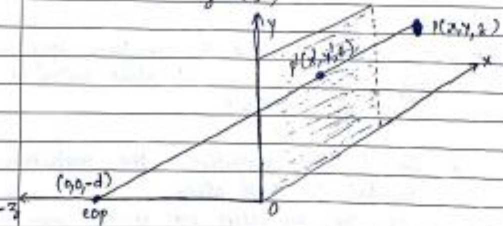
- The size of projected image is inversely proportional to the distance of object from c.o.p. (i.e closer the object more bigger it appears after projection & vice versa)
- Doesn't retains the dimensions of the object.
- provides realistic effects as perceived by human visual system.
- Involves some anomalies like :
  - set of parallel lines not parallel to p.p. tends to converge at one point called as Vanishing point.
  - The object behind gets projected in upside down manner

To Derive Matrix for Perspective projection,

→ let point to be projected is $p(x,y,z)$

→ let projection plane be xy plane

→ let c.o.p is at $(0,0,-d)$ (i.e on $-ve$ z-axis)



using parametric eq^n to represent a line,

$$\boxed{\begin{aligned} x' &= x_1 + u\Delta x \\ y' &= y_1 + u\Delta y \end{aligned} \quad \& \quad z' = z_1 + u\Delta z}$$

Consider line cop → p

Here let $x_1 = 0 \quad y_1 = 0 \quad , z_1 = -d$

$x_2 = x \quad y_2 = y \quad z_2 = z$

$\therefore \quad \Delta x = x \quad , \quad \Delta y = y \quad , \quad \Delta z = z + d$

$$\Rightarrow \quad x' = x_1 + ux \quad = ux \qquad — ①$$

$$y' = y_1 + uy \quad = uy \qquad — ②$$

$$\& \quad z' = z_1 + u\Delta z \; = \; -d + u(z+d) \qquad — ③$$

As $z' = 0$ ($\because$ projection plane is xy plane)

$\Rightarrow$ eq^n ③

$$0 = -d + u(z+d)$$

$$\therefore \quad u = \frac{d}{z+d}$$

$$\Rightarrow \quad x' = \frac{xd}{z+d} \qquad y' = \frac{yd}{z+d} \qquad \& \; z' = 0$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{d}{z+d} & 0 & 0 & 0 \\ 0 & \frac{d}{z+d} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

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

$$\text{Matrix}_{perspective} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & d \end{bmatrix}$$

is applicable when assumptions made holds true

Explain Z buffer organizati' algorithm for hidden surface removal.

→

- Z buffer algorithm is also called as depth buffer algorithm.
- It uses two buffers (a 2D array)
  1) z buffer (depth buffer) used to record z (depth) values
  2) Frame buffer (refresh buffer) used to record intensity values.

Calculations of depth:

A plane is represented by eqn

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

$$\Rightarrow \quad z = -\frac{(Ax + By + D)}{C} \qquad C \neq 0$$

On given scan line y = constant

∴ depth of pixel at $x_1 = x + \Delta x$ along the scan line is

$$z_1 - z = -\frac{(Ax_1 + D)}{C} + \frac{(Ax + D)}{C} = \frac{A(x - x_1)}{C}$$

$$\therefore z_1 = z + \frac{A}{C}\Delta x$$

but $\Delta x = 1$

$$\therefore \boxed{z_1 = z + \frac{A}{C}}$$

Similarly for the next immediate scan line

$$z_1 = z - \frac{(By_1 + D)}{C} + \frac{(By + D)}{C} = \frac{B(y - y_1)}{C} = \frac{B}{C}\Delta y$$

$\because \quad \Delta y = 1 \quad \Rightarrow \quad \boxed{z_1 = z - \frac{B}{C}}$ coherence property

Pseudocode outline of the algorithm:

- initialize the z-buffer to the minimum z value
- initialize the frame-buffer to the background intensity
- for the projection of each polygon or object
    - for each scan line within the polygon's proj^n bounding box
        - for each pixel within the polygon's proj^n bounding box
            - calculate the depth of the pixel $z(x,y)$
            - if $z(x,y) > Zbuffer(x,y)$ then
                - { store intensity value of pixel (x,y) in Frame Buffer
                - store $z(x,y)$ to the z-buffer
                - }
            - end if
        - next pixel
    - next scan line
- next polygon or object
- Display the frame buffer.

(*) It belongs to image space category

↓ uses more memory as it requires 2-Buffers

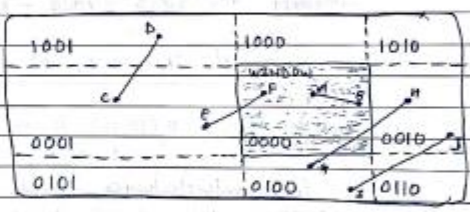
↑ simplicity

Explain the cohen-sutherland line clipping algorithm with suitable example. (10

→ ⊙ The entire picture is regarded as divided into nine regions w.r.t. clip window & the clip window being the central region.

⊙ Every region is identified by a 4-bit code

T	B	R	L
Top	Bottom	Right	Left



⊙ End points of line segment are assigned a region code depending on, in which region the falls.

eg: Region code for G ≡ 0100

" " " H ≡ 0010

⊙ The algorithm uses slope-intercept form of equations to find point of intersections.

Algorithm:

step 1: Accept window extents (xw_{min}, yw_{min}) & (xw_{max}, yw_{max})

step 2: Accept end point coordinates of line segment (x_1, y_1) & (x_2, y_2)

step 3: Assign region codes to both the end points:

```
To assign region code to a point p(x,y)
    set  T=B=R=L=0
if( x < xwmin)
        L=1
else
if( x > xwmax)
        R=1
if ( y < ywmin)
        B=1
else
    if( y > ywmax)
        T=1
```

step 4: If region code for both the end points consists all zeros
$\Rightarrow$ line is totally acceptable (inside)
$\therefore$ display the line & STOP.

step 5: If Bitwise Anding of both the region codes is non-zero
$\Rightarrow$ line is totally rejectable (outside)
$\therefore$ STOP

step 6: Find point of intersection $p'(x', y')$ of line and appropriate window edge by considering region code of a end point which is outside
first calculate "slope" $m = (y_2 - y_1)/(x_2 - x_1)$

if $(L==1)$ i.e w.r.t. left edge

$$\boxed{y' = y_1 + m(xw_{min} - x_1) \quad \& \quad x' = xw_{min}}$$

else

if $(R==1)$ i.e w.r.t. Right edge

$$\boxed{y' = y_1 + m(xw_{max} - x_1) \quad \& \quad x' = xw_{max}}$$

else

if $(B==1)$ i.e w.r.t. Bottom edge

$$\boxed{x' = x_1 + \frac{1}{m}(yw_{min} - y_1) \quad \& \quad y' = yw_{min}}$$

else

if $(T==1)$ i.e w.r.t. Top edge

$$\boxed{x' = x_1 + \frac{1}{m}(yw_{max} - y_1) \quad \& \quad y' = yw_{max}}$$

Step 7: Replace an appropriate end point with the newly calculated point of intersection $p(x', y')$

step 8: For this shortened line segment, repeat from step 3.

step 9: STOP.

Fractals :

- A fractal is a rough or fragmented geometric shape that can be split into parts, each of which is approximately a reduced-size reproduction of the complete shape, based on the property known as self symmetricity or self similarity.

- A mathematical fractal is based on an equation that undergoes iteration, a form of feedback based on recursion.

- Broadly, there are three types of self similarities found in fractals :

  (a) Exact Self-similarity : Fractals defined by iterated function system

  (b) Quasi Self similarity : The fractals appears approximately but not exactly identical when seen at different scales. Fractals defined by recurrence relations are examples of this kind.

  (c) Statistical self similarity : Fractals has numerical or statistical measures which are preserved across scales. Random fractals are the examples of it.

Explain Weiler Artherton polygon clipping algorithm with suitable example.

- Weiler Atherton polygon clipping algorithm is used to overcome a drawback of Sutherland Hodgman algorithm (i.e may not properly clip a concave polygon)

- The polygon to be clipped is called as subject polygon & a window is called as clip polygon

- The boundaries of subject & clip polygon may intersect or may not intersect.
  If they intersect, then these intersections occurs in pairs : One, when subject polygon edge enters the clip polygon & second, when it leaves

- The modification suggested is while traversing the subject polygon edge, ~~~~ for outside to inside movement proceed along the subject polygon edge, however for inside to outside movement proceed along the clip polygon edge from the point of intersection.

- It uses to vertex lists, one for subject polygon & other for clip polygon

- The vertex list includes all the vertices as well as point of intersections.

- To generate polygon after the clipping, we need to traverse the subject polygon list from entering point of intersection till the exiting one & then from such exiting point of intersection we need to traverse the clip polygon list till we encounter the entering point of intersection.

- The above procedure is repeated till the list exhausts.