3D-Transformation

Curve Modeling

Besier Spline Curves

→ Bezéer splines have a number of properties that make them highly useful and convenient for curve.

→ Berier curve can be fitted to any number of control points.

→ The number of control point to be approximated and their relative position determine the degree of Berier polynomial.

→ Berier curve can be specified with boundary conditions.

→ control points using blending f^n, characterizing matrix, or boundary conditions.

∗

* Bezier Curve Equation *

n number of control position + 1.

* Suppose we are giving $n+1$ control-point positions

$P_K = (x_K, y_K, z_K)$ with K varing from 0 to n.

→ $P_0, P_1, P_2 \cdots P_n$ (every point of position P_K, P_K expressed as x_K, y_K, z_K)

* These co-ordinate points can be blended to produce the following position vector $P(u)$, which describes the path of an approximating Bezier polynomial f^n between P_0 and P_n.

* All these points they will be blended or put together, they will be combined together to pr....

* $$P(u) = \sum_{K=0}^{n} P_K \, BEZ_{K,n}(u), \qquad 0<u<1 \quad — (1)$$

$P(u) \Rightarrow$ Describe the path of the polynomial f^n beth then point P_0 and P_n. $P(u)$ express that the summation of K is equal to 0 to n $(K = 0 \cdots n)$

$P(K)$ and then we have $BEZ_{K,n}(u)$, where u is varing between 0 and 1.

BEZ → this is called as the ~~Boinstin~~ Bernstein polynomial, or this is called the blended f^n. The Bezier BEZ blending f^n is also called as the BEZ of K, n of (u), where u, is varing from (0,- 1)

The Bezier blending f^n $BEZ_{k,n}(u)$ are the Bernstein Polynomials / $b_{i,n}(t) = \boxed{n_{c_i} t^i (1-t)^{n-i}}$

$$BEZ_{K,n}(u) = C_{(n,K)} u^K (1-u)^{n-K} \quad \text{— (2)}$$

(K=0 The value of (0-1)
n = number of the control point)

where, parameters C(n,K) are the binomial co-efficients

$$\text{(*)} \quad n_{c_i} \;/\; C_{(n,K)} = \frac{n!}{K!(n-K)!} \quad \text{— (3)} \;/\; \frac{n!}{i!(n-i)!}$$

Equation 1 represents a set of three parametric equations for the three individual curve co-ordinates —

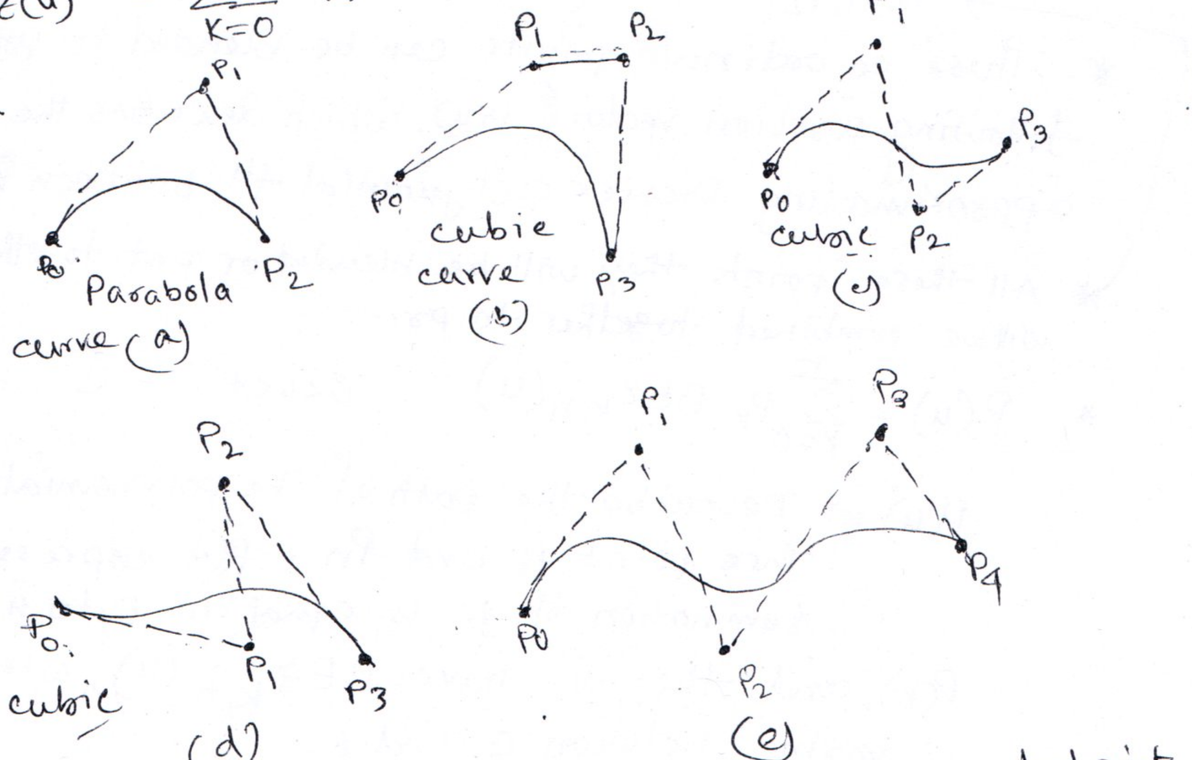
$$x(u) = \sum_{K=0}^{n} x_K \, BEZ_{K,n}(u)$$

$$y(u) = \sum_{K=0}^{n} y_K \, BEZ_{K,n}(u)$$

$$z(u) = \sum_{K=0}^{n} z_K \, BEZ_{K,n}(u)$$

P(u) is nothing but Bezier polynomial f^h which is express the term of — (4)

Spline representation



* Two dimensional Bezier curve generated with three, 4, 5 control points.
— dashed lines connect the control point *

* Figure appearance of some Bezier curves for various selection of control points in the xy plane (z=0)

* Suppose we have control point at same co-ordinate ~~points~~ position, these set of control-point produce a Bezier curve, that is only a single point.

a) P_1 it is pulling the curve up ~~tows~~ towards direction. P_0 is starting point P_2 is the ending point. P_1 is another point which is lying to P_1 and P_2. that is why curve is oriented towards to P_1.

b) How P_0 has control point? P_3 is a last control point in between P_0 and P_3 and P_1, P_2. P_1 and P_2 their controlling the curve direction. so the curve has been drag towards P_1 and P_2. It is more towards because the P_1 is closer here. P_2 also having control . . .

c) It has four control points but we can see that after control point P_0 there having P_1 and having P_2 and P_3. $P_0 \rightarrow$ towards to P_1 and towards P_2 : - - P_3.

d) P_0, P_1, P_2, P_3 (curve is similarly oriented)

e)

lets take 4 control point, where $n = 3$

P_0, P_1, P_2, P_3

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

$\boxed{0 < t < 1}$

$$* \quad b_{0,3}(t) = {}^3c_0 t^0 (1-t)^3$$

$$= (1-t)^3 \quad , \quad {}^3c_0 = \frac{3!}{0!(3-0)!}$$

<u>same way</u>

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

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

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

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

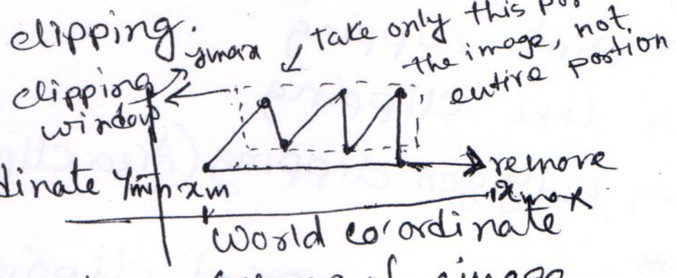
$$y_0(t) = y_0(1-t)^3 + y_1 3t(1-t)^2 + y_2 3t^2(1-t) + y_3 t^3$$

Clipping

* what we understand clipping?

→ we have clipping the part, means we are cutting the part, so, whatever the unnecessary part it is that we have to removed. that we call it as a clipping.

→ let us take, a image.



→ window is given with a co-ordinate X minimum, Y- minimum. X - maximum, Y- maximum is called clipping window.

→ One window is given with co-ordinates (x_{min}, y_{min}) & (x_{max}, y_{max}) is called clipping window.

→ Only this portion of image has to be displayed on.

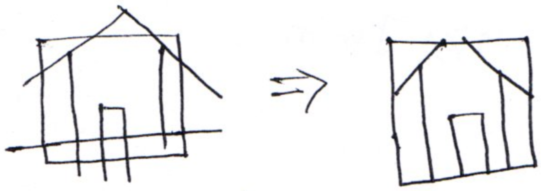
→ so, we want to remove, portion is called clipping window.

→ why maximum, and minimum?

→ Means the portion of the clipping window we are assigning with x minimum and ~~x~~ x-maximum values. and the ymin and y-max values.

→ we have to display the object which is coming inside clipping window and destroy the part that is outside the window has to be clipped.

Applications of clipping



→ clipping will extract part what we ~~design~~ desire.

* It will extract part we ~~design~~ desire

* To indentify the visible and invisible area in 3D object

* For creating object using solid modeling
* For drawing operations.
* For deleting, copying, moving part of an objects.

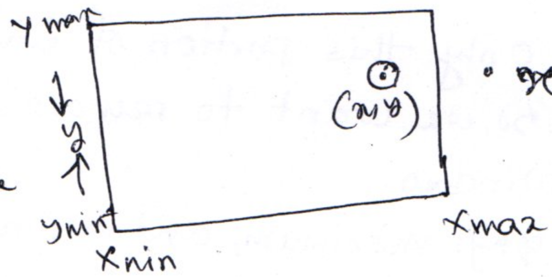
Types of clipping

→ Point clipping
→ Line clipping
→ Polygon clipping (Area clipping)

* Point clipping

→ It is used to determining whether the point is inside the window or not.

→ If point is coming inside the clipping window, we have display it; otherwise no need to display it



• ~~select~~ select it)

* Entire the window, having a point which is (x, y)
* We need to check whether the point is inside the window or not we don't know.
* We need to check condition => wheter its inside the clipping window ore not

1. ~~x should~~ $x \le x_{max}$ (x should be less than or eqaul to xmax)

2. $x \geq x_{min}$

3. $y \le y_{max}$

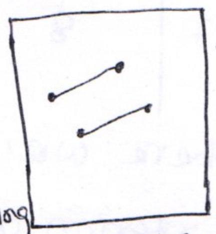
4. $y \geq y_{min}$.

Line clipping

* Line clipping is same as point clipping
* The line appear outside the clipping window that line has to be discarded
* Line that is present inside that has to be accepted.
* So, we need a intersecting point.

→ The part of the line inside the window is kept & the part of the line appearing outside is removed - the line clipping.

* if line and both the endpoints or present inside the clipping window
* both points are present inside the clipping window / visible lines

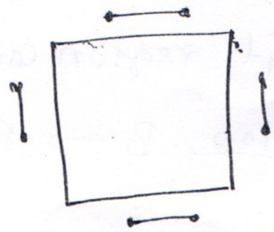


→ Accept the line

→ don't clip.

visible line.

→ If the line / both the endpoints are present outside the window, then we call not visible lines.

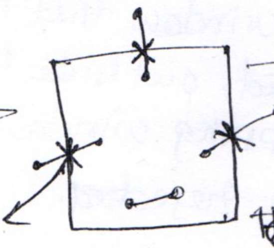


→ not visible lines.

→ don't need of clipping.

→ Reject the line.

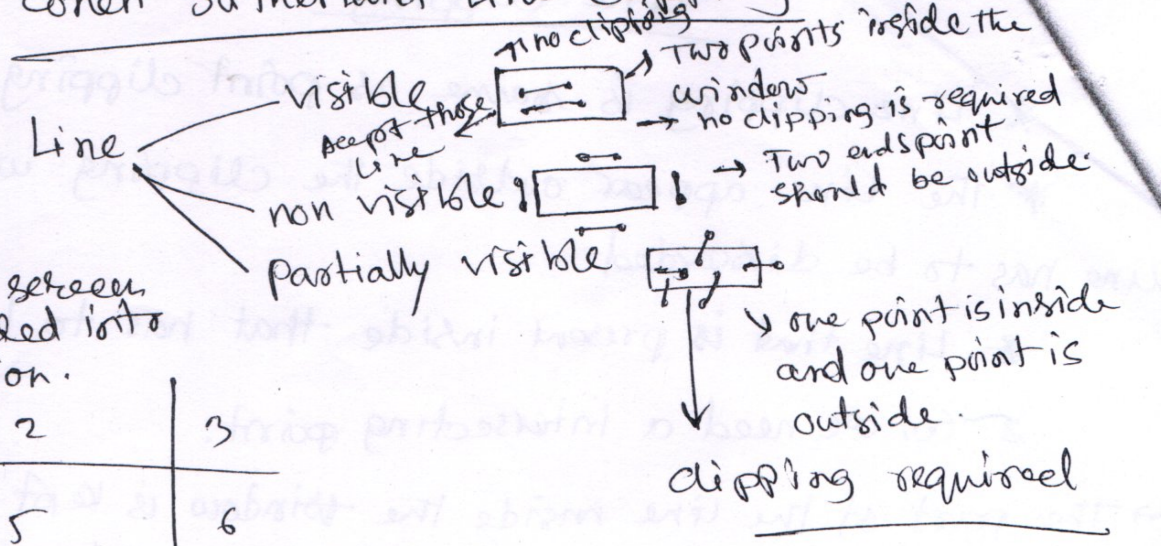
Here, clipping is required



→ Partially visible lines.

intersection point

Cohen Sutherland Line Clipping



→ The complete screen is complete devided into total line region.

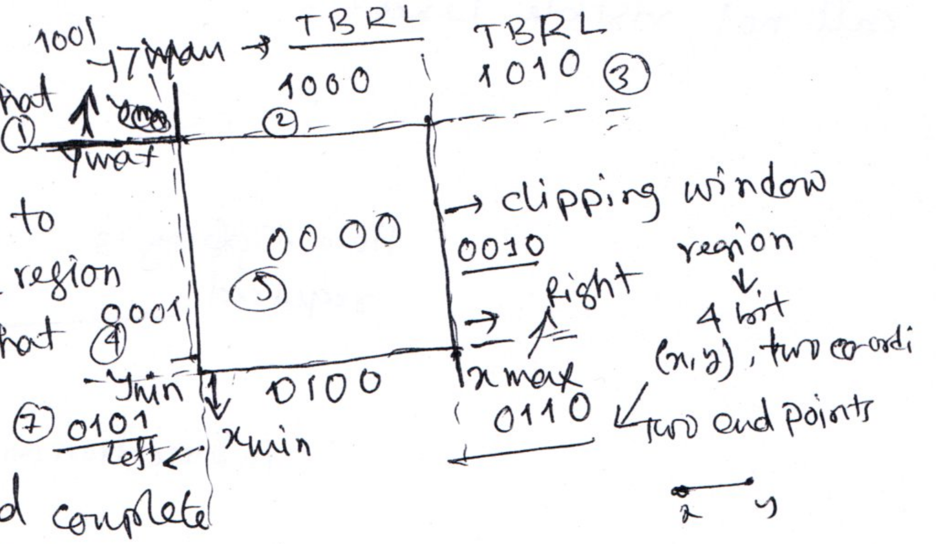
region 1	2	3
4	5	6
7	8	9

→ So, here we've devided the screen into 9 region.

→ Every region we have to devided em into code [code is 4 bit]
0 1

→ 4 bit region code represents – TBRL
T → Top, B → Bottom, R → Right, L – Left.

* whatever the Line that is present inside the clipping window that has to be accepted, outside the region of the clipping window that should be rejected.



→ we already devided complete screen into 9 region, the centre area having the code 0000. [we said each code is a 4 bit]

→ x, should be x_{min} and x_{max}.

→ y should be y_{min} and y_{max}.

Condition — 4 bit form - TBRL

T: $y > y_{max}$ → y is coordinate that has to be present in — between y_{min}, y_{max}.

if y is greater than y_{max} is crossing the window

B: $y < y_{min}$ → below the window

R: $x > x_{max}$ → crossing the window.

L: $x < x_{min}$ → Left the window.

lets take. top ⇒ crossing the outside the window —

Top - 1001

→ Top has to said 1, if above the window.

→ Bottom

→ Right

→ Left [because is the y max in left side]

This vision code in 4 bit.