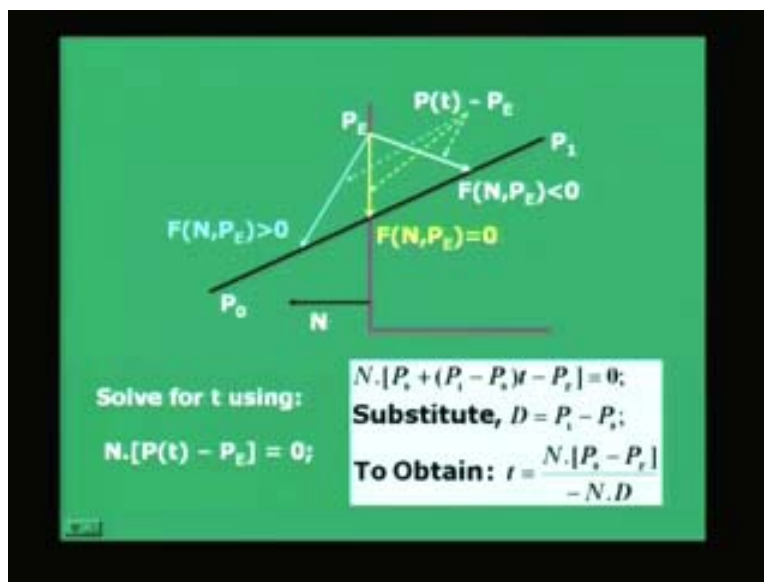


**Computer Graphics**  
**Prof. Sukhendu Das**  
**Dept. of Computer Science and Engineering**  
**Indian Institute of Technology, Madras**  
**Lecture - 21**  
**Clipping: Lines and Polygon**

Hello and welcome everybody again to the lectures on computer graphics. In the last class we started to discuss the Cyrus Beck formulation of line clipping. That was the first of the algorithms of line clipping in two dimensions. That means given a rectangle, a bounding box with coordinates  $x_{\min}$   $x_{\max}$   $y_{\min}$   $y_{\max}$ . We like to find out if part of the line is within the clipping window or it is completely outside or completely inside. So based on that we have to clip the line with respect to rectangle. So we continue the discussion so that you follow the last part of the derivation which was left over in the last class.

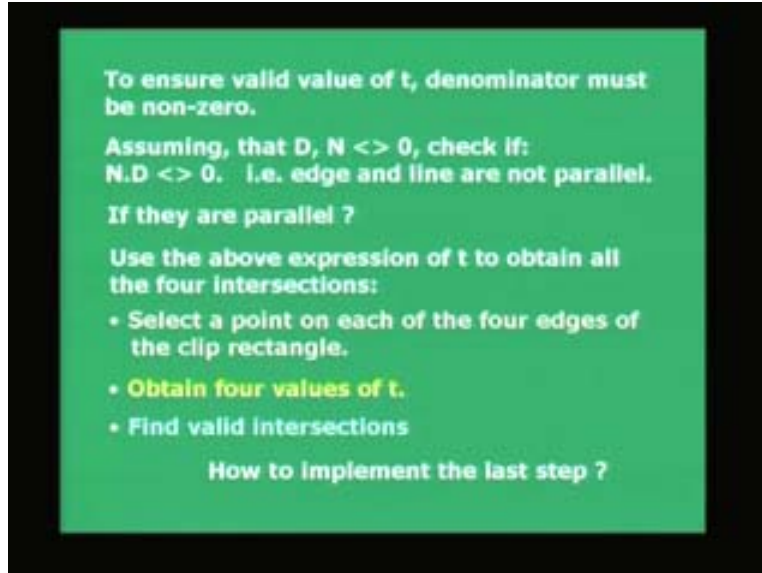
(Refer Slide Time: 01:49 min)



And if you look into the slide we were looking into the formulation of Cyrus Beck which talked about the vector or a functional form, first vector  $P(t)$  minus  $P_E$ . These are the three different vectors given for the three different points  $P$  of  $t$  on the line  $P_0$  to  $P_1$  and  $F$  of  $N$ .  $P_E$  is nothing but the dot product of this vector  $N$  which is normal to the clipping boundary and dot product of this  $N$  with the  $P(t)$  minus  $P_E$ . That is the one which we used for solving for  $t$  and the expression of which can be used is given on the right hand side bottom. This we had seen in the last class.

So the expression of  $t$  is given by the numerator  $N \cdot (P_0 - P_E)$  divided by  $N \cdot D$ . That is what we used to obtain the  $t$ , the intersection of this clipping boundary with the line  $P_0 P_1$ . You have to of course take care of various facts. First of all to ensure valid value of  $t$ , the denominator must be non 0 and assuming that the  $D$  and  $N$  both are not equal to 0.

(Refer Slide Time: 03:58 min)



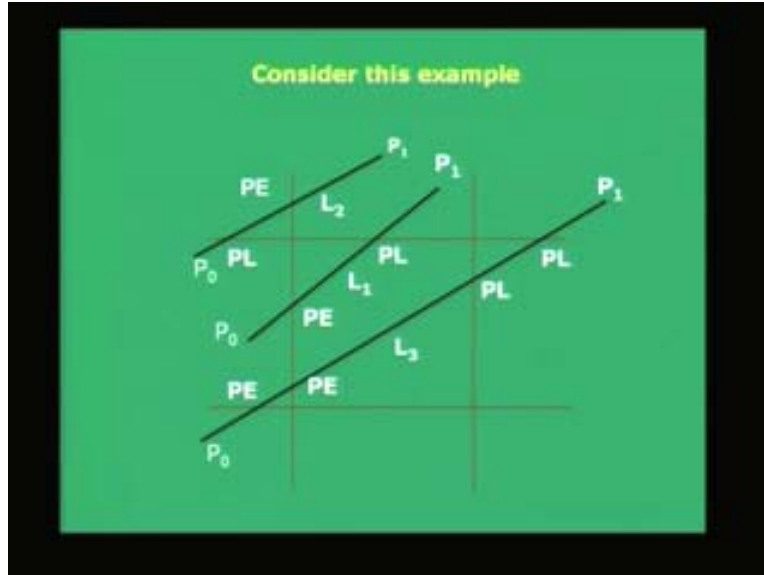
We check first of all the dot product is not equal to 0. That means the edges and lines are not parallel. Basically it means that if this  $N \cdot D$  the dot product is equal to 0, the edge and line are parallel, they will never meet and there is no point trying to find out the value of t where these two lines will intersect. So you do not need to do anything, if they are parallel the dot product is equal to 0 and when it is not equal to 0 we use the above expression which was given in the last slide to evaluate the t and obtain all the four intersections. What are the four intersections? Well, the clipping boundary has four bounding lines. I talked about  $x_{\min}$   $x_{\max}$   $y_{\min}$   $y_{\max}$  with respect to all these four lines I find out the four values of t which the line intersects.

So, after doing that you select and the process consists of selecting a point on each of the four edges of the clip rectangle because in the previous figure PE is the point which was an arbitrary point on an edge. So select that point for each of the four edges, obtain four different values of t. As we look into the slide that is the next step of the algorithm, obtain four values of t from the four different points using the expression given earlier and then find the valid intersections. This is the last one which we have to worry about that how to find out whether this is the valid value of t.

What is the valid value of t?

First of all the t must lie within 0 and 1. Consider this particular example where we had three different lines starting from points  $P_0$  to  $P_1$  and their level  $L_1$   $L_2$   $L_3$  and we will see here that all of them will have in some cases two valid intersections and in some other cases four and the intersections are labeled as PE and PL which we also defined in the last class and we will again go through that very fast.

(Refer Slide Time: 04:05 min)



The steps are to find if any value of  $t$  is outside the range then you reject the line. Else sort with the increasing values of  $t$  and then this solves the  $L_1$ , the line which was in the middle in the previous diagram but it does not solve the lines  $L_2$  and  $L_3$ . So the criteria to choose the intersection points PE or PL are the following that move from point  $P_0$  to  $P_1$ . If you are entering the edges inside the half-plane then that intersection point is marked as PE.

(Refer Slide Time: 05:00 min)

**Steps:**

- If any value of  $t$  is outside the range  $[0 - 1]$  reject it.
- Else, sort with increasing values of  $t$ .

This solves  $L_1$ , but not lines  $L_2$  and  $L_3$ .

Criteria to choose intersection points, PE or PL:

Move from point  $P_0$  to  $P_1$ .

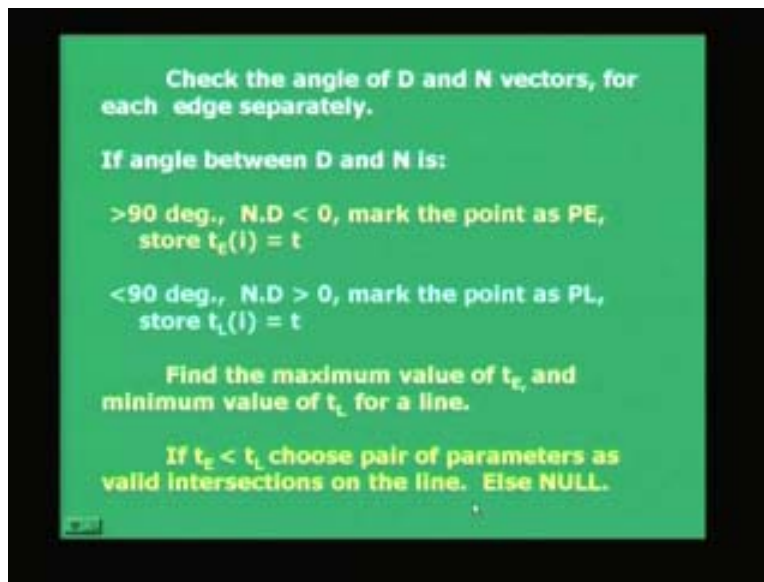
If you are entering edge's inside half-plane, then that intersection point is marked PE;

else, if you are leaving it is marked as PL.

E stands for Entering, L stands for Leaving. So if you are leaving then it is marked as PL. Check the angle of D and N vectors for each edge separately. And if the angle between D

dot N is more than 90 degree or greater than 90 degree then that dot product will be negative, mark that point as PE and store it in an array t of E. The parameter t is subscript entering into the half plane. So I is equal to 1 and you keep on incrementing as much as you get PEs for each particular line. Of course the maximum you will have two values of PE and two values of PL for each particular line because there are at the maximum four impossible intersections of a line with the four edges of a polygon. The last criteria for PL is the second criteria basically which tells what is PL is that this angle is less than 90 degree then the dot product is positive and then you mark that point as PL.

(Refer Slide Time: 06:25 min)

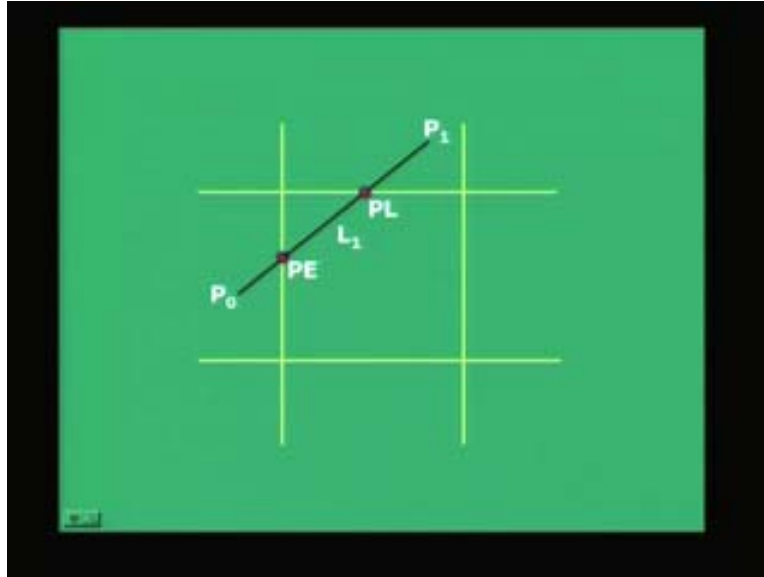


That means it shows that the point line is leaving the half-plane and you store that particular value of t under that array tL of I is equal to t. And after doing this, once it is sorted and once you have done this, find the maximum values of tE and the minimum value of tL for a line. Check if the corresponding maximum value of tE is less than the minimum value of tL. Then you choose the pair of parameters as the valid intersections of a line. Else you do not choose. Let us take this particular example today.

Let us take a line  $L_1$ . This is interesting, the same example which you seen earlier but we will take line by line now. Let us take line  $L_1$  which runs from point  $P_0$  to  $P_1$ . Now you see that actually if you look at the ending points of this particular line in the particular example it has two valid intersections and we have to level them but if you think of the extended line which basically can go to infinity there are four possible intersections with four boundaries of any line. Unless the line of course is parallel to one of the edges it may have in general four intersections. So you have to find out all the four values of t for the four intersections and pick up only those values of t which lie in 0 to 1. That is the first step.

We look into the slide here, you will find basically two intersections of this line  $P_0$  to  $P_1$  labeled as  $L_1$  with the left vertical and the top horizontal edge. So there are two values of  $t$ . The other two values of  $t$  which you get will be lying outside 0 and 1.

(Refer Slide Time: 07:28 min)

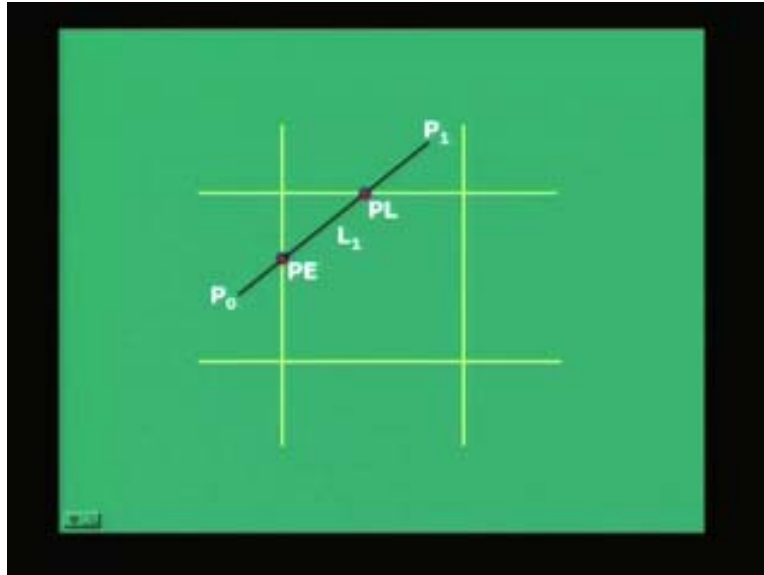


So you do not consider them at any point. So, now you start labeling. The first point is entering the clip boundary half-plane so you mark it as level as PE. You know the condition which will occur that the point will be the dot product and it will be negative and so on. So that is what you do and that is where you level it as PE. You will go for other  $t$ . Also look for the same and you find that the dot product basically is more than 90 degrees and so you mark it as PL.

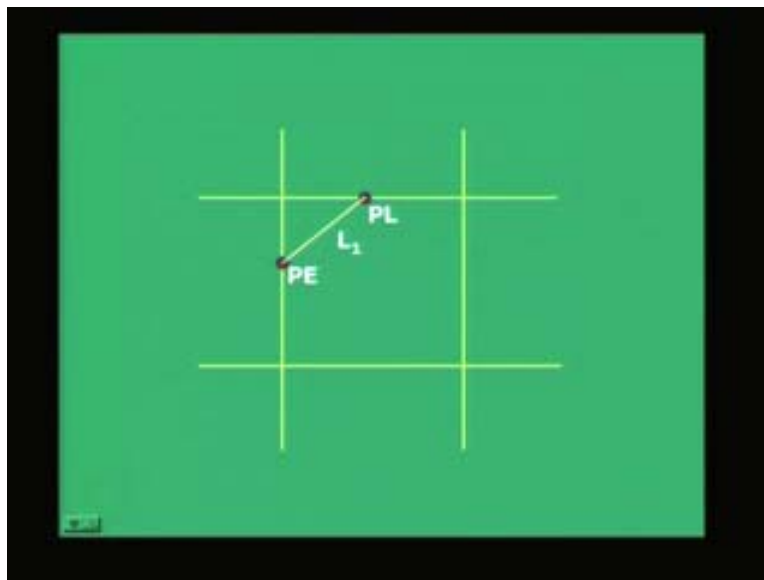
Remember, if you go back to previous slide you see the conditions which I told you, that is what I was talking about, if the angle is more than 90 degrees then dot product is negative and then you mark it as PE entering plane and when it is less than 90 degrees you take the dot product to be positive, mark it as PL. So, these are two criteria which you are using to level PE and PL.

In the next figure, again I repeat the line going from  $P_0$  to  $P_1$ , mark the first one as PE and mark the second one as PL depending upon the sign of the dot product of that  $N \cdot D$ . That is what you do. First of course you evaluate  $t$  and then at that point take the dot product of  $N \cdot D$ . Now you find that there are two values of PE and PL only. So, there is no question of any more sorting and this PE and PL lie amid 0 to 1. The  $t$  value for PL is more than the  $t$  value of PE. So that is the valid part of the line.

(Refer Slide Time: 7:44)

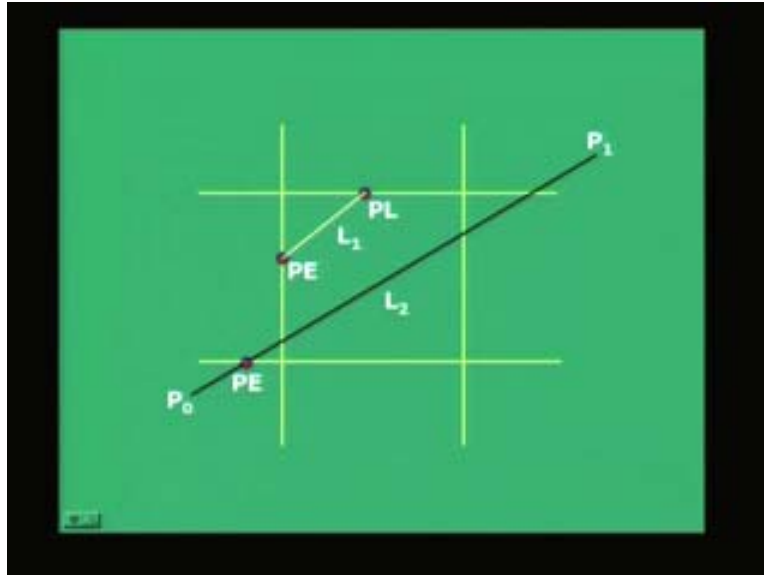


(Refer Slide Time: 08:54)



So you restored  $P_0$  to  $PE$  and  $P_1$  to  $PL$  and say this is the valid part of the line which is within the valid set of intersections or pair of intersections and that is where the portion of line  $L_1$  which lies within the clip rectangle.

(Refer Slide Time: 09: 14 min)



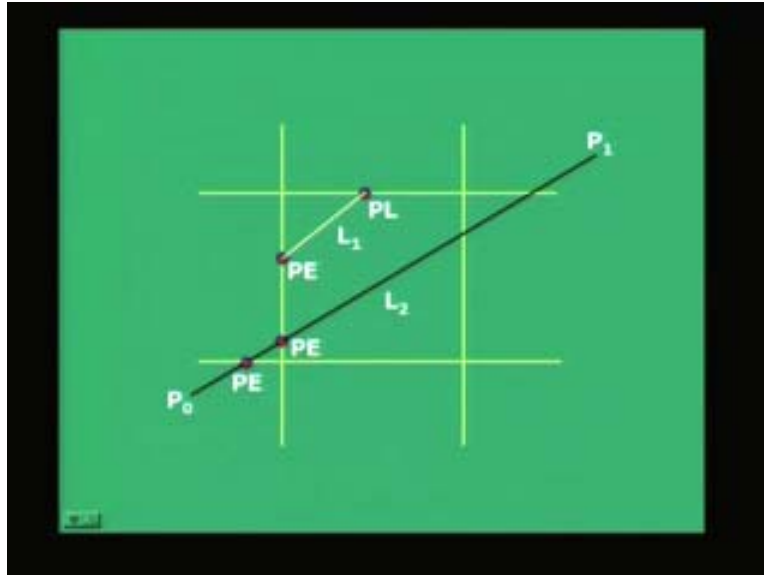
So let us take another line. This is the large line which has four valid intersections. It is the line  $L_2$  again running from some starting point  $P_0$  to finishing point  $P_1$  and this line will be labeled  $L_2$ . So you will have four valid intersections. So you start to move from  $P_0$  to  $P_1$ , that is the algorithm. So, first intersection point which you get is PE because it is entering the half-plane. The corresponding dot product will give the corresponding sign and that is what we are used to.

What is the sign?

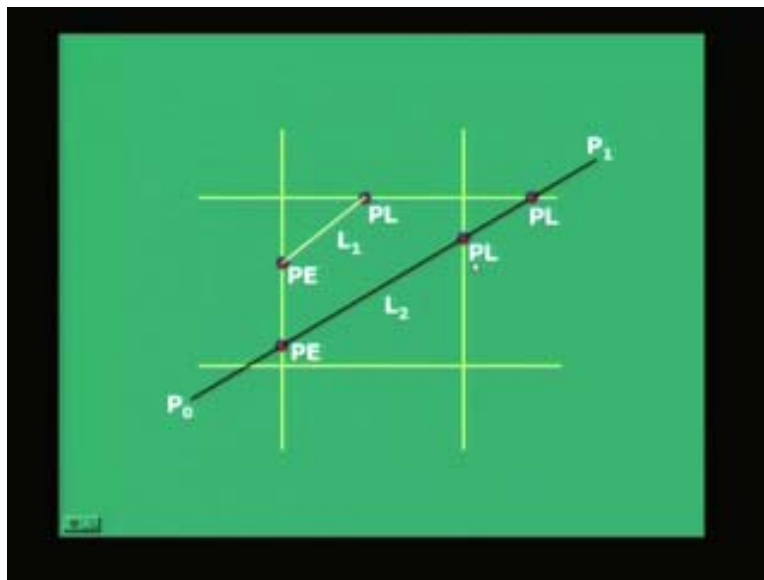
If you remember correctly that is what we use to mark it as PE then the vector will be pointing down and the D vector basically which is  $P_0 P_t$  minus PE will also show an angle which is less than 90 degrees. So that is what you have.

So again rolling it back a little bit here as you say the PE value is more than 90 degrees and the sign is 0. And this is more than 90 degrees, the sign of N. D vector is negative and that is what you mark it as. So the dot product is negative, you mark it as PE, roll it over again, that is what is the dot product negative you marked as PE. You do the same thing for the next intersection point also where you mark it as PE again because it is also entering the half-plane with respect to the left most boundary, left vertical edge of the clip rectangle. So these are the two values of PE.

(Refer Slide Time: 10:30 min)



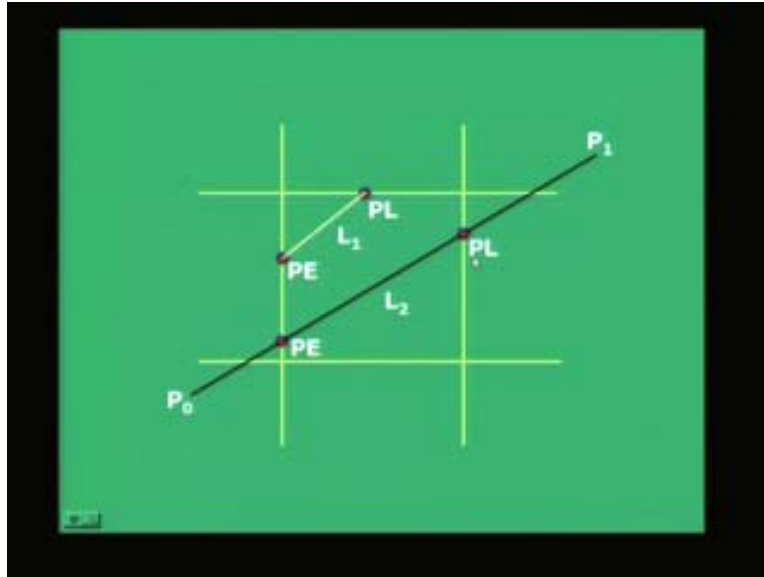
(Refer Slide Time: 11:03)



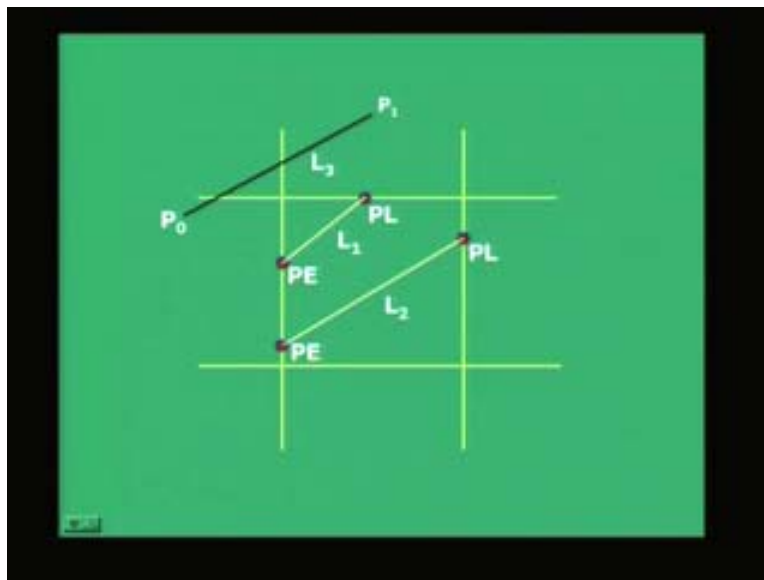
The next intersection will be the right vertical edge and that will be leaving the plane. Before that what we can do is also take the maximum value of this  $tE$  of  $I$ . So you throw out the minimum value of  $PE$  and keep the one which is at the maximum. You find the exiting value of  $PL$  with the right vertical edge and the top horizontal edge you get another  $PL$  and out of them you have to keep the minimum. So you throw this part out which is on the top right, you keep this and these are the two valid intersections the  $t$  value lies between 0 and 1. So this is the part of the line which you will find.



(Refer Slide Time: 11:15 min)



(Refer Slide Time: 11:35)

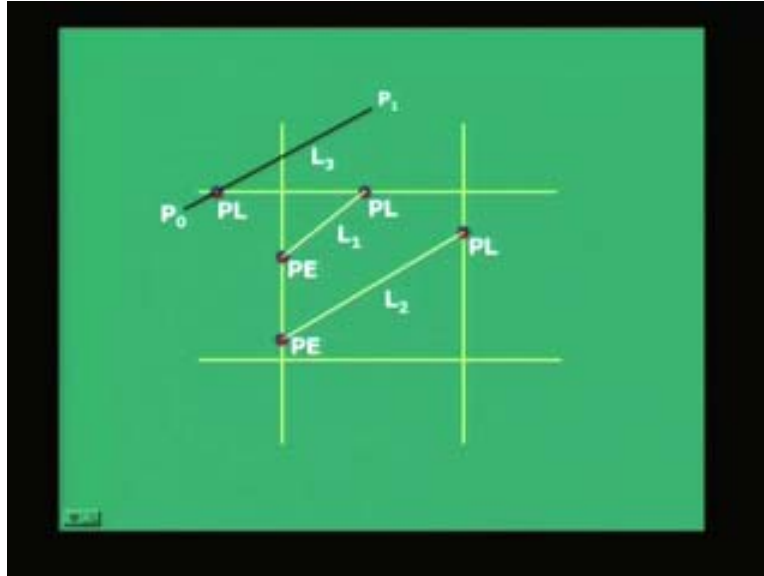


So, this is how you basically find out the portion of the line which is within the clip rectangle. What happens if the line is outside? Let us take another example, the last example  $L_3$  of the line. So this is again running from  $P_0$  to  $P_1$  and there are two valid intersections for the line  $L_3$ .

As we move from  $P_0$  to  $P_1$  you first find a  $PL$  because you are leaving the half-plane. The first intersection is with the top horizontal edge. So you typically get the dot product which is positive. So it will be leaving the plane and then of course as you keep

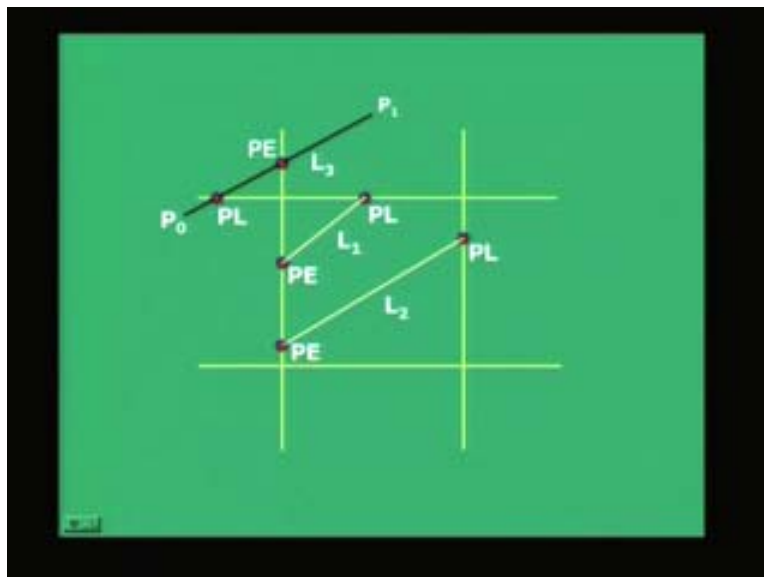
proceeding from  $P_0$  to  $P_1$  the next intersection point which you get is  $P(E)$  because it is entering the half-plane.

(Refer Slide Time: 11:43)



This intersection at that  $PE$  is with respect to the left vertical edge where you get  $PE$ .

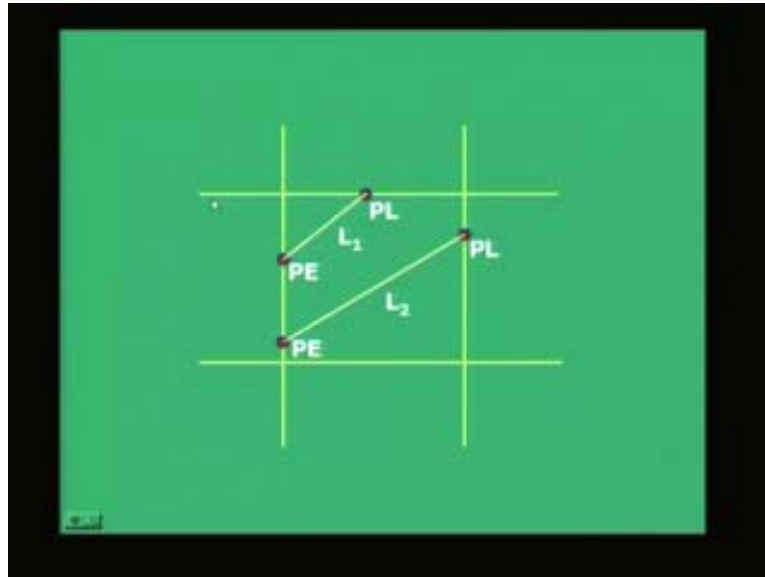
(Refer Slide Time: 11:58)



And now what you find is since there are two valid intersections you do not need to sort anything. But interestingly you find out, if you look into the diagram again that the  $t$  value for  $PE$  is more than  $t$  value of  $PL$ . Since in the previous cases the  $t$  value for  $PE$  was less than the  $t$  value for  $PL$ . But in the reverse case here the  $t$  of  $PE$  is more than that the

parameter for the PL and hence you eliminate this line. So you completely reject that line and out of those three lines, parts of two of these lines will be within the clip rectangle, the one which is outside can be withdrawn.

(Refer Slide Time: 12:36 min)



(Refer Slide Time: 13:17 min)

**Calculations for parametric line Clipping**

Clip Edge	Normal N	$P_E^s$	$P_0 - P_E$	$t = \frac{N \cdot [P_1 - P_0]}{-ND}$
Left: $X = X_{min}$	(-1, 0)	$(X_{min}, Y)$	$(X_0 - X_{min}, Y_0 - Y)$	$\frac{-(X_s - X_{min})}{(X_s - X_e)}$
Right: $X = X_{max}$	(1, 0)	$(X_{max}, Y)$	$(X_0 - X_{max}, Y_0 - Y)$	$\frac{(X_s - X_{max})}{-(X_s - X_e)}$
Bottom: $Y = Y_{min}$	(0, -1)	$(X, Y_{min})$	$(X_0 - X, Y_0 - Y_{min})$	$\frac{-(Y_s - Y_{min})}{(Y_s - Y_e)}$
Top: $Y = Y_{max}$	(0, 1)	$(X, Y_{max})$	$(X_0 - X, Y_0 - Y_{max})$	$\frac{(Y_s - Y_{max})}{-(Y_s - Y_e)}$

§ - Exact coordinates for  $P_E$  is irrelevant.

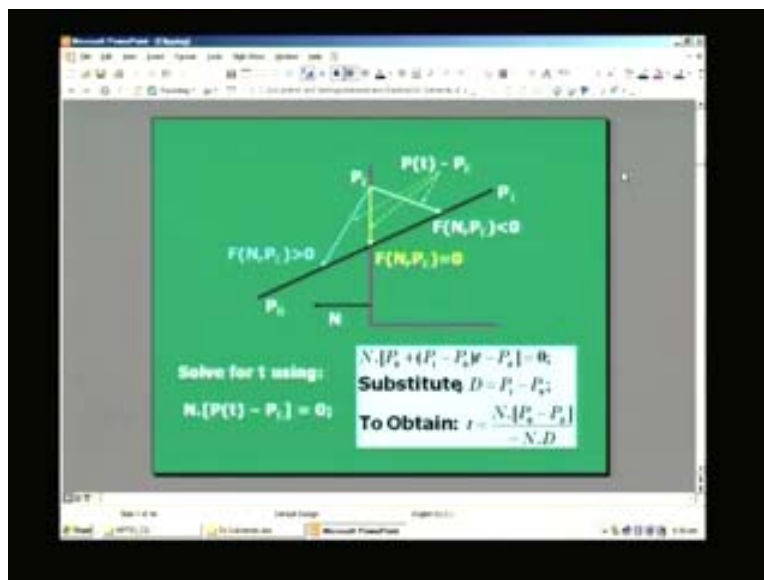
So this is an example which we have seen with this Cyrus Beck formulation. And if you remember that typically we discussed that we have to use the clipping when we talked about clipping with respect to the canonical volume or view port in 2D and that is why if you remember when we were talking about 3D viewing transformations and pipe line we said that the equations of the lines in 2D or surfaces in 3D are very simple, the clipping

formulation also becomes very easy. We have discussed that. And will see that in next table.

As you see here that these are the calculations for the parametric clipping line where the left edge is  $X$  equal to  $X_{\min}$ , right edge is  $X$  equal to  $X_{\max}$ . We already know this. You look into the left vertical column the clip edges are the left, right, bottom and top. Left one is  $X$  equal to  $X_{\min}$ , right is  $X$  equal to  $X_{\max}$ , you know that. Bottom horizontal line is  $Y$  equal to  $Y_{\min}$  and the top horizontal line is  $Y$  equal to  $Y_{\max}$ . And the corresponding normals to this clip edges as given by the normal  $N$  is given in the second column minus 1 0 1 0 and this is all in 2D, this is the normal. And this PE, exact coordinates of PE is irrelevant, so you choose any arbitrary point.

But since this PE must lie on the corresponding edge so that the  $X$  equal to  $X_{\min}$  must have  $X_{\min}$  coordinates  $X_{\max}$  must have  $X_{\max}$  and so on. For the bottom and the top horizontal clipped edges of the rectangle  $Y_{\min}$  and  $Y_{\max}$  must be there. Any arbitrary value of  $X$ , we are talking of the third column for coordinates of PE for the top two  $Y$  could be the arbitrary in the last two values of the third column, the  $X$  could be the arbitrary. PE is irrelevant but of course part, at least one component of PE is relevant not both because PE must lie on it, that constraint must be there. So based on this we can actually find out what is  $P_0$  minus PE, this value which we need it for the calculation PE is given.  $P_0$  is the starting point of the line so you can easily calculate that the fourth column here gives you  $P_0$  minus PE. So substitute that into this expression for evaluating  $t$ . You see the expression becomes very easy now because the numerator you have is the normal vector  $N$  which is given in the second column.

(Refer Slide Time: 15:43 min)



$P_0$  minus  $P_E$  is given in the fourth column and the denominator  $N \cdot D$  basically will be given as, the  $N$  vector given here is normal and what was the definition of the  $D$  vector?  $P_t$  minus  $P_E$  if you roll back and find out..... what was the expression for  $D$ ?

Let us roll back and see the expression for  $D$  which was  $P_1$  minus  $P_0$ . That is the vector for the line that was  $D$ . So, if you roll forward and substitute this into this table which we are viewing right now let us see what is coming in the denominator because  $X_1$  minus  $X_0$  is 0 here since  $N$  does not have any  $Y$  component so you will be left with  $X_1$  and  $X_0$ . These are the simple expressions to calculate.

(Refer Slide Time: 15:23)

**Calculations for parametric line Clipping**

Clip Edge	Normal N	$P_E$	$P_0 - P_E$	$t = \frac{N(P - P_E)}{-ND}$
Left: $X = X_{min}$	(-1, 0)	$(X_{min}, Y)$	$(X_0 - X_{min}, Y_0 - Y)$	$\frac{-(X_s - X_{min})}{(X_s - X_e)}$
Right: $X = X_{max}$	(1, 0)	$(X_{max}, Y)$	$(X_0 - X_{max}, Y_0 - Y)$	$\frac{(X_s - X_{max})}{-(X_s - X_e)}$
Bottom: $Y = Y_{min}$	(0, -1)	$(X, Y_{min})$	$(X_0 - X, Y_0 - Y_{min})$	$\frac{-(Y_s - Y_{min})}{(Y_s - Y_e)}$
Top: $Y = Y_{max}$	(0, 1)	$(X, Y_{max})$	$(X_0 - X, Y_0 - Y_{max})$	$\frac{(Y_s - Y_{max})}{-(Y_s - Y_e)}$

§ - Exact coordinates for  $P_E$  is irrelevant.

You see that the examples are so easy, the calculations are so easy, why?

This is what we discussed earlier before even clipping when we talked about clipping with respect to view ports and canonical volume. Of course canonical views is in 3D and view port is in 2D but you can almost forecast what is going to happen in 3D by looking into the algorithm in two dimension. We will discuss as much of three d later on if not in this lecture definitely in the next lecture. We will see some aspects of 3D as well. But the concept is similar from 2D to 3D is one more instead of clipping with respect to line you clip with respect to surface. That is the difference between clipping in 2D and 3D.

But you can see even the expression of clipping in 2D becomes very simple because the clipping rectangle is very well defined. It is a square rectangle with horizontal and vertical edges and not an arbitrary polygon or an arbitrary rectangle which could be oriented in arbitrary directions and the expressions are known. And in fact in the view port if it is normalized from 0 to 1 the expression becomes much more simple because the  $X_{min}$  and  $X_{max}$   $Y_{min}$   $Y_{max}$  will be 0 minus 1 or plus 1 and the expression becomes much more simpler. Look back into the last column for the expression of  $t$ , this is what you have as the shaded ones.

(Refer Slide Time: 17:31)

**Calculations for parametric line Clipping**

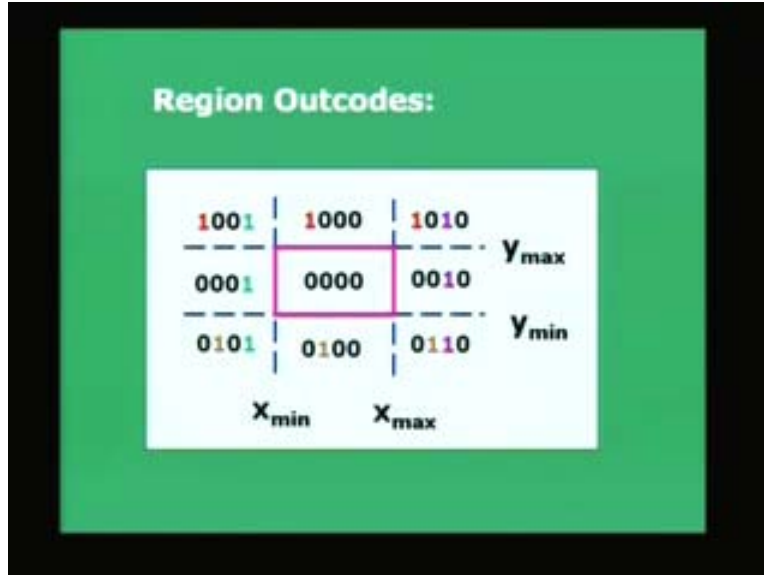
Clip Edge	Normal N	$P_E^s$	$P_0 - P_E$	$t = \frac{N \cdot (P_0 - P_E)}{-ND}$
Left: $X = X_{\min}$	(-1, 0)	$(X_{\min}, Y)$	$(X_0 - X_{\min}, Y_0 - Y)$	$\frac{-(X_0 - X_{\min})}{-(X_1 - X_0)}$ $\frac{(X_1 - X_0)}{-(X_1 - X_0)}$
Right: $X = X_{\max}$	(1, 0)	$(X_{\max}, Y)$	$(X_0 - X_{\max}, Y_0 - Y)$	$\frac{(X_0 - X_{\max})}{-(X_1 - X_0)}$ $\frac{-(X_1 - X_0)}{-(X_1 - X_0)}$
Bottom: $Y = Y_{\min}$	(0, -1)	$(X, Y_{\min})$	$(X_0 - X, Y_0 - Y_{\min})$	$\frac{-(Y_0 - Y_{\min})}{-(Y_1 - Y_0)}$ $\frac{(Y_1 - Y_0)}{-(Y_1 - Y_0)}$
Top: $Y = Y_{\max}$	(0, 1)	$(X, Y_{\max})$	$(X_0 - X, Y_0 - Y_{\max})$	$\frac{(Y_0 - Y_{\max})}{(Y_1 - Y_0)}$ $\frac{-(Y_1 - Y_0)}{-(Y_1 - Y_0)}$

§ - Exact coordinates for  $P_E$  is irrelevant.

So  $X_0 X_1 Y_0 Y_1$  is nothing but the end point coordinates of the line. That is what you need and of course you also need  $X_{\min} X_{\max} Y_{\min} Y_{\max}$  which are nothing but the equations of my clipping edge of the rectangle. So you can see this  $X_{\min} X_{\max} Y_{\min} Y_{\max}$  becomes 0 1 or minus 1 to plus 1, the expressions becomes very easy for evaluating 3D. So now you know the reason why we talked about clipping with respect to normalized view port in 2D or normalized view volume or a canonical view volume in 3D because expressions to evaluate becomes much much simpler and that will help you to compute the clipping parameters in the algorithms that you learn faster because you have many lines to be clipped and for each you have to do about four such edges, four calculations plus a few more steps. Here ends the discussion on Cyrus Beck formulation.

We move on to the next popular algorithm which has been proposed by Cohen and Sutherland. So we look into the Cohen-Sutherland algorithm for line clipping. Well this is not exactly based on the parameter  $t$  to start with. We will see how it comes but will first see what are called as the region outcodes. We look into the slide here. These are the region outcodes. You have to understand this very carefully because this is the key principle of Cohen-Sutherland region outcodes.

(Refer Slid time 18:30)



If you see here the clipping rectangle is given by the rectangle in between and there are four edges of course, this pink colored line or rectangle gives you the clipping rectangle, two horizontal and two vertical of course. And we will say that using this four set of lines or two pairs of lines, we divide the entire 2D space into nine sub regions or sub areas. Sub regions or even regions, we can say regions. Of course, if we take the entire region is the sub region but let us say nine regions we have. Top three, bottom three and middle three. And of course you have  $X_{min}$   $X_{max}$  as usual  $Y_{min}$   $Y_{max}$  too as the parameters which will dictate the equation of the corresponding four lines for the clipping rectangle. I again repeat, two horizontal and two vertical.

If you carefully see that using these two pairs of lines, two horizontal and two vertical or four together it is easy for you to visualize that if I extrapolate this line then the entire two dimensional space is split into nine regions. And for each region I have a code, this is very interesting and must be understood carefully. Each region has a code. The one which is inside, the region which is inside the clip rectangle has four bits which are all 0. Each region is represented by a set of four bits and the bit value can be only 0 or 1, each bit. So, you need four bits to represent the code for each particular region and you have the internal one which is all zeros.

Now you can see these codes form a 3 into 3 array, so look at each row or each column. Let us look at the first row. The bit value for the region outcodes for the first row. What do you think is common? What do you think is common between all the bit codes of the region on the top row? You see the one on the left most bit that can be considered as most significant bit MSB it cannot be considered but you can take any code. But to be consistent in representation you put 1 in the first bit if it is lying in the top row. For any one of the top three regions you have a bit 1 and that is for sure.

Similarly, if you look for the first column of the region outcodes now you can visualize this as a matrix of region outcodes. If you look at the first column that is the left most region you will find that the last bit which is put in a light green color has a bit 1 so that you see as a region 1. You see the region outcodes are marked outside. What about the last row? As in the top row you had a 1 in the first bit and the left most column you had a 1 in the last bit. The last row that is the last three regions outcodes are represented by a 1 in the second bit. This is the light brownish color 1. Each color indicates each row or column of regions out of 3 out of 9. And of course now you are left with the last column. We are mainly bothered about the regions outside the rectangle but that is what we play a very important and interesting role. You see that the third bit which is given by purple color and marked as 1 will be placed as 1 if the region is in the last column. To the right vertical edge of the clip rectangle all these regions will have 1 in the third column. I hope this idea is getting clear because I have used different colors for respective rows or columns for regions outside the clip rectangle. Of course all bits are 0 on the inside as you can see.

I repeat again, first bit is 1 for the first row, last bit is 1 for the first column last, column the second bit is 1 and the third or the last column the third bit is 1. Again I repeat the second bit is 1 for the last row, last row has the second bit 1 which is given in light brown and the purple color indicates the third bit and of course inside the sub regions is all zeros. So I hope this idea is clear. We have a clip rectangle  $X_{\min}$   $X_{\max}$   $Y_{\min}$   $Y_{\max}$  and using these four lines or two pair lines we divide this entire two dimension space into nine regions. One of them is inside and there are eight outside, three on the top and three on the bottom and the two on other sides of the region.

For each such region which is occupying a part of the two dimension space we put a code represented by four bits. Why four bits because you basically have a set of regions, the three regions on the top of the clip rectangle, three regions on the bottom of the clip rectangle, three regions on the left of the clip rectangle and three regions on the right of the clip rectangle. So basically there are four directions. We have the set of three regions on four different directions top, bottom, left and right. So we need four bits which is very natural and that is what used for region outcodes. You look back again and that is what you see that the first bit takes care of top row, second bit takes care of bottom row, third bit takes care of the right most column of the regions and the last bit takes care of the left most column of the regions basically if you visualize this is a matrix.

But you can of course change this 0 1 combination, not arbitrarily in the sense that you can say instead of the first bit we used to represent the regions on the top row or above keep the clip rectangle is the better way to say it then you could change from first to second and something else you put in the first but you must be consistent with the algorithm and you should make such changes which we will see how the algorithm uses this region code, Cohen-Sutherland algorithm uses this concept of region outcodes to clip a line. So whatever you use you must be consistent, you should not switch within the algorithm of region outcodes.



So I hope this idea of region outcodes is clear to all of you because if you do not understand this concept of region outcodes, how each region is labeled with respect to or with the help of this four bit representation then you will not be able to follow the rest. Please draw this and visualize this yourself and then will move over to the algorithm. This is what was given in the table. The bit number for the first which is the MSB has 1. If the region is above the top edge  $Y$  is more than  $Y_{max}$ . If it is below the top edge then it is 0  $Y$  is equal to  $Y_{max}$ . The second bit number is 1 if the region is below the bottom edge  $Y$  is less than  $Y_{min}$ . If it is above the bottom edge then the value is 0 that is  $Y$  is more than  $Y_{min}$ . The third bit number is 1 when it is the right of the right edge that is  $X$  is more than  $X_{max}$ . The third bit is 0 when it is left of the right edge that is  $X$  is less than  $X_{max}$ . For fourth which is LSB the bit is 1 if you are to the left of left edge that is  $X$  is less than  $X_{min}$  and it is 0 if it is right of the left edge where  $X$  is more than  $X_{min}$ . That is, you can tally these set of descriptions given in the table with the region outcodes given here.

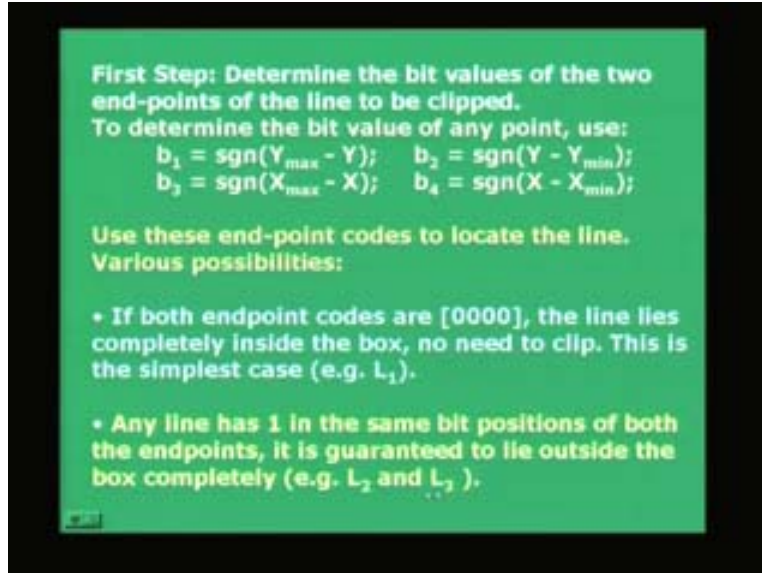
I hope you have copied it already and you will be able to blindly put the region outcodes given a clip rectangle.

(Refer Slide Time: 26:46)

Bit Number	1	0
FIRST (MSB)	Above Top edge $Y > Y_{max}$	Below Top edge $Y < Y_{max}$
SECOND	Below Bottom edge $Y < Y_{min}$	Above Bottom edge $Y > Y_{min}$
THIRD	Right of Right edge $X > X_{max}$	Left of Right edge $X < X_{max}$
FOURTH (LSB)	Left of Left edge $X < X_{min}$	Right of Left edge $X > X_{min}$

This is the table which identifies the corresponding region. Depending upon the XY coordinates of a point you should be able to say where it lies or within which region it is. What is the corresponding outcode of a point or in other words given the XY coordinates of a point you should be able to get the outcode very easily either using this logic from the table or the one which was given earlier in the form of the codes within each regions within the rectangle and eight outsides.

(Refer Slide Time: 30:00)



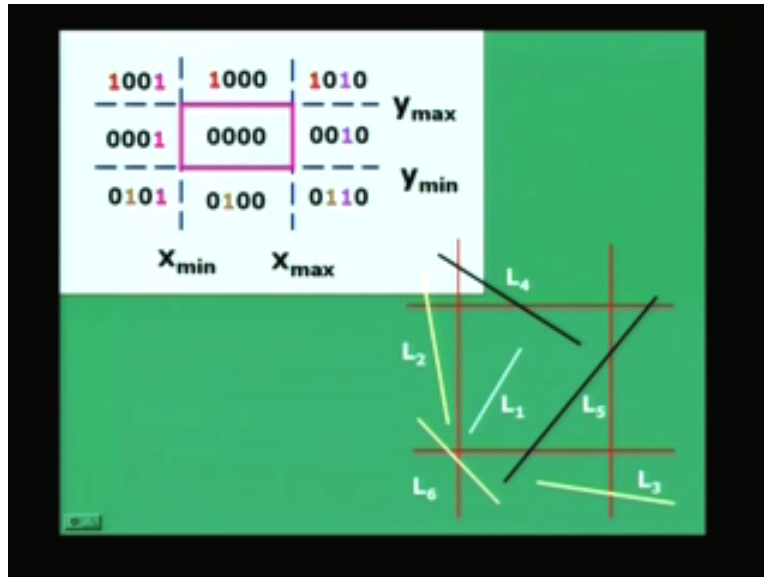
If this concept is clear let us get into the first step of algorithm which says, look into the slide, the first step says determine the bit values of the two end-points of the line to be clipped. That means given  $P_0$   $P_1$  find their corresponding bit values, four bits for each point, the set of four bits for each points. To determine the bit value any point this is the easiest form of the expression because this can be derived from the concept of what we gave in the table before in the previous slide where we say that use the sign of this value either  $Y_{\max}$  minus  $Y$  or  $Y$  minus  $Y_{\min}$  or  $X_{\max}$  minus  $X$  or  $X$  minus  $X_{\min}$  to obtain the corresponding set of four bit values for a corresponding given point  $X$  and  $Y$  are the corresponding coordinates of a point and these are the bit values. Use these end-point codes as given in the slide, in the next line use these end-point codes to locate the line.

Various possibilities exist because as I said before that you can have various combinations of zeros and 1s in the four bit string. The bit values which you have, the bit string to be very precise, if both endpoint codes are four zeros then the line lies completely inside the box, you do not need to clip, your job is over. That is the most simplest and easiest case which you can visualize, if  $P_0$  and  $P_1$  both lie inside the box completely and both lines lies inside the box then both the end point codes will be four zeros and the line lies completely inside the box and you do not need to clip. This is the example in the case for  $L_1$  which you will see in the next slide which will come which is similar to what we used in the Cyrus Beck as well. Almost similar examples of lines we will use.

If the line has 1 in the same bit positions of both the endpoints it is guaranteed to lie outside the box completely. There are two examples of line  $L_2$  and  $L_3$  which we will see in the next class which will have a bit value 1 in the same bit positions of the both the end points. You remember, at the beginning of this slide we said we have to calculate four bit values for each end point separately. Once it is done using the sign value as given in the top, you now check if it is inside the box or then next if it is completely outside the box.

It is guaranteed to lie outside the box completely only when same bit positions of both the end points is 1 each. So we will see the examples of  $L_2$  and  $L_3$  and check their code if it is 1 on the same bit and in case of  $L_1$  of course both the bit values is 0. That is of course straightforward and simple, no need to almost verify that. This is a typical example.

(Refer Slide Time: 30:16 min)



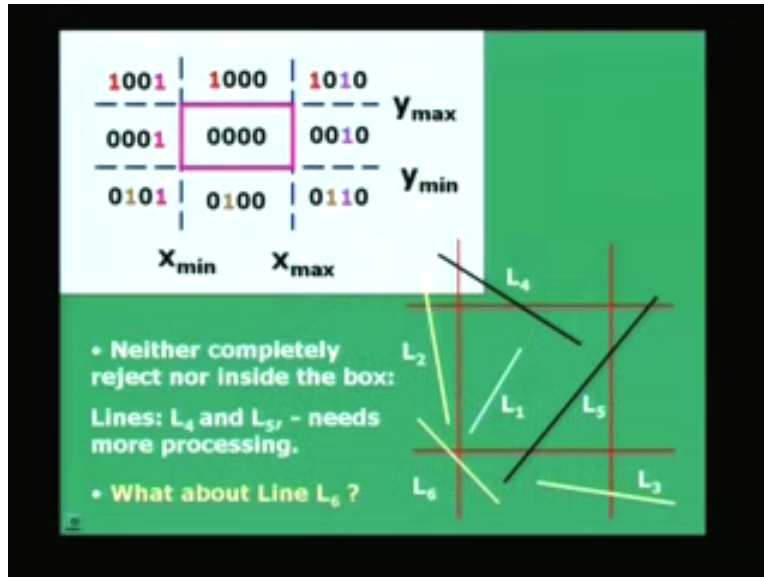
Let us put the region outcodes as well. Although in the algorithm you might use the sign function of  $X - X_{min}$   $X - X_{max}$  whatever the case may be as given in the previous slide to obtain the bit values but will derive it analytically from here. So let us take  $L_1$ , both the end points are lying within the line here which is inside the box so the bit values are four zeros inside the box completely, that is absolutely no problem.

Let us take the line  $L_3$  the left most starting point to the right finishing point, so the starting point bit value is 0 1 0 0 because it is here. The starting point of  $L_3$  is 0 1 0 0. The finishing point is on bottom right so the bit value is 0 1 1 0. As you can see there is a bit value 1 in the second bit position in both the starting and finishing point which is marked by this brown color in the second bit position. And hence since  $P_0$  and  $P_1$  both have 1 for the starting and end point region outcodes this line is guaranteed to lie outside the clip rectangle. No part of the line will be inside and so reject the line so  $L_3$  is completely inside  $L_3$  is rejected.

We will look at  $L_2$  which is on the left side here, starting point from bottom to the top. So 0 0 0 1 is  $P_0$  outcode and  $P_1$  region outcode is 1 0 0 1. I repeat again  $P_0$  is this 0 0 0 1 and 1 0 0 1 is your region outcode for  $P_1$ . If that is so the fourth bit position as you can see has 1 in both the region outcodes for  $P_0$   $P_1$  and since there is 1 in the same bit position for the starting and finishing point region outcodes, remember this long term which I am using, starting point and finishing point region outcodes has same bit value one in the same bit position.

Other values may be 0 and 1 or other combinations but or both zeros we do not bother. But anywhere if you find that the bit value is 1 in both  $P_0$  and  $P_1$  then it is guaranteed that the line is guaranteed to lie outside, completely outside the block and so it is rejected. So  $L_1$  is inside  $L_2$  and  $L_3$  are outside the clip rectangle that is done.

(Refer Slide Time: 34: 10 min)



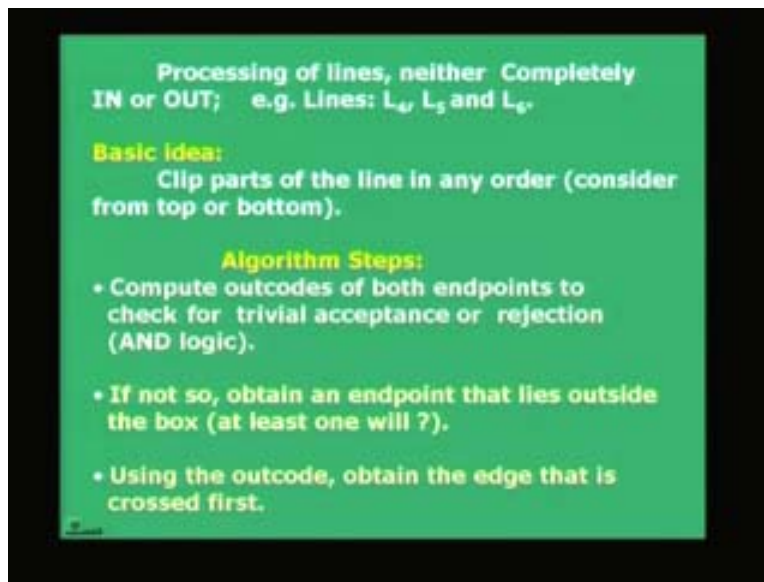
Of course there are three other lines as you can see the within the figure  $L_4$   $L_5$  and  $L_6$  which needs some special treatment. So these three lines are neither completely can be rejected and they are neither inside the box also. Line  $L_4$  and  $L_5$ , part of that is within the box and part of that is outside the box and you will find in the region outcodes of  $L_4$  and  $L_5$  you will see that you neither have the previous two cases which we found for  $L_1$  where the bit values are both zeros, that will not happen in both  $L_4$  and  $L_5$ . Also the cases of  $L_2$  and  $L_3$  also they do not occur because if you take the two end points of  $L_4$  one is all zeros and the other is 1 0 0 1. So you will not have 1 in any identical bit positions.

If you take  $L_5$  which is 0 1 0 0 and 1 0 1 0 here also you will find that there is no 1 in the same bit position and hence  $L_4$  and  $L_5$  are neither completely outside the box nor completely inside the box and these are the lines which require clipping. We have to cut some part of the line which is inside and of course it may happen. These are  $L_4$  and  $L_5$ , what about the line  $L_6$ ? That is still more interesting.  $L_6$  is completely outside the box but it does not satisfy the conditions of  $L_2$  and  $L_3$ . That means let us look at the outcodes for end points of the line  $L_6$  it is 0 1 0 0 and 0 0 0 1 neither is inside nor completely outside. So  $L_6$  also needs a very special treatment. These are the set of lines. So  $L_1$ ,  $L_2$  and  $L_3$  are very straightforward.

Simple steps can tell you whether it is completely inside, accept both the end points, accept the line completely and for  $L_2$  and  $L_3$  throw away the lines and clipping is absolutely not necessary.  $L_4$ ,  $L_5$  and  $L_6$  are the three lines we have seen in the slide now which require clipping or some special treatment to handle to find out when these lines or

neither completely inside or completely outside. So we will go back and look into this slide. This is the picture so we will only consider the cases for  $L_4$ ,  $L_5$  and  $L_6$  lines. So we need to have a special mechanism by which we process lines which are neither completely inside or out with respect to, of course the boundary or the area of the clip rectangle. These are the lines  $L_4$ ,  $L_5$  and  $L_6$ . The basic idea is clip parts of the line in any order and consider top to bottom or bottom to top but in this example we will consider top to bottom top to bottom or bottom to top we will probably consider from bottom to top.

(Refer Slide Time: 37:58 min)

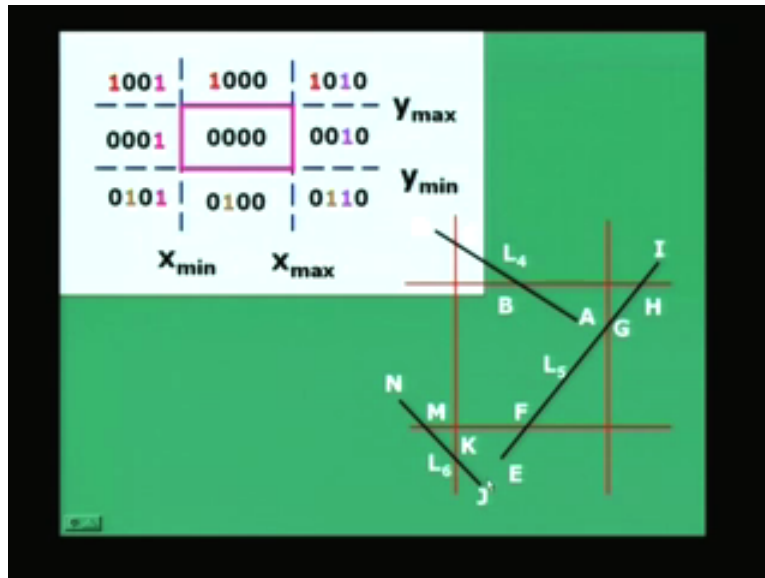


The algorithm steps for these particular cases  $L_4$ ,  $L_5$  and  $L_6$ . Remember, we are only considering the special algorithm steps for those lines which are neither completely inside the clipping rectangle or completely outside because we know within the region outcodes that we can keep the line completely inside or throw away those line completely outside the box. That is one and that is done. We have to look for other lines now it does not satisfy either of these two conditions. The part of that could be inside or may not be also. So it is for only for these cases these algorithms are necessary.

Coming back to this, so we clip parts of the line in any order the algorithm steps are the order following. Compute outcodes of both endpoints to check for trivial acceptance or rejection and like this of course was the step which we do not bother right now for  $L_1$ ,  $L_2$  and  $L_3$  lines. If not so that is from here onwards basically I should admit that we are talking about the lines  $L_4$ ,  $L_5$  and  $L_6$ . Obtain an end point that lies outside the box, at least one will, so it is a question mark I have put. When you reach this point for a particular line, if you are able to reach this algorithm step that means you are unable to completely accept the line or completely reject it because the line is neither completely inside nor completely outside. That means part of the line at least is definitely outside the box and henceforth there is at least one point of such a line which we could not completely accept nor completely reject of those cases at least one of the end points may be both, that is also

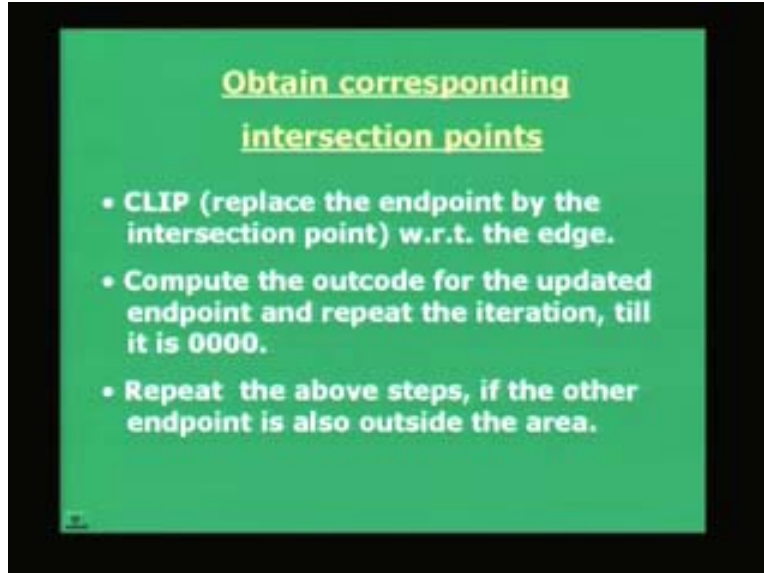
possible but at least one of the end points will lie outside the clipping rectangle of that box. This is guaranteed, if you read this step and that is the step you have reached for a particular line which you could not completely accept because the end points are neither inside because then the end points, the bit values would have been all four zeros. It is the case where it was completely outside where the two end points did not have 1 in the same bit position.

(Refer Slide Time: 38:42)



At least one of them is outside and that is what we used. That is guaranteed, you have at least one point outside the box. So using the outcode obtain the edge that is crossed first. We will see this, how to use the outcode to obtain the edge first. Of course remember, now we have to add basically clip parts of the line in some order and we can move from top to bottom or bottom to top in any order we will start clipping and will see how to clip with the help of outcodes. These are the three lines. Now, of course put some labels of intersections but we still have those lines labeled in black color, colored in black  $L_4$ ,  $L_5$  and  $L_6$ , parts of the line  $L_4$  and  $L_5$  are within the box. We could not accept or reject  $L_6$  but although still it is completely outside the box. So what do you do? These are the region outcodes which you can use to check. So, we look into the algorithm steps again.

(Refer Slide Time: 40: 28 min)



Obtain corresponding intersection points; that is, clip, replace the end points by the intersection section point with respect to the edge that is how you do the clipping. This of course we assume here that you know a parametric form of line clipping and obtain a value of  $t$ . Remember, some expressions even before Cyrus Beck algorithm we talked about obtaining the two values of  $t$  given two parametric lines  $P_0$  to  $P_1$   $P_2$  to  $P_3$  or  $P_0$  to  $P_1$  if you remember, I think I used  $P_0$  prime to  $P_1$  prime, two lines are there with two end points given. You should actually be able to find  $t_1$  and  $t_2$  for both these lines where it intersects unless they are parallel. So use that concept here to clip with respect to the edge to get the end points and of course find it is valid or not. And then compute the outcode for the updated endpoint and repeat the iteration till it is 0 0 0 0.

So what you are trying to do is basically clip and try to push an end point from the out to the in and stop there. We will see that with the help of the algorithm. One example is with the help of these three lines that the line crossing the clip rectangle then what you do is start with one of the end point which is outside the box because it could be the case that one end point is inside the box and another is outside. So you should choose one of the end points in that case which is outside.

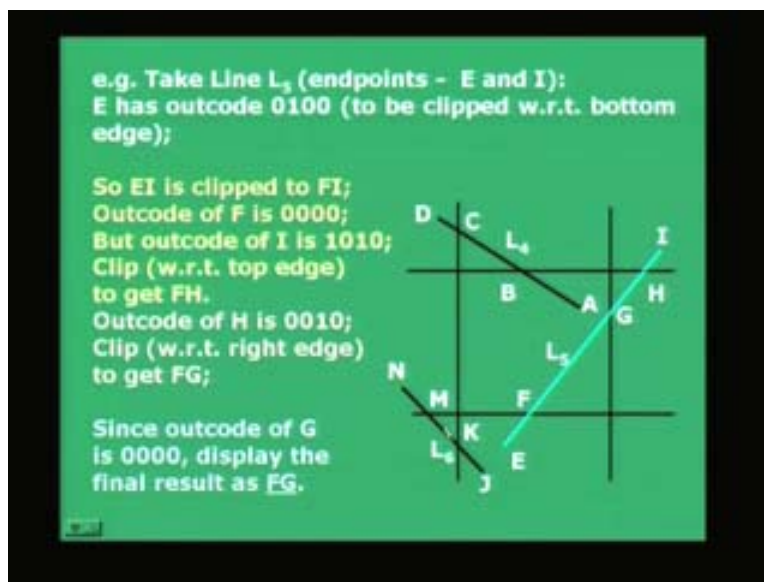
Of course if both the end points are outside you can start from any of the end points but typically we move from bottom to top or top to bottom. It is in one order so choose from end points and start clipping with the first deck and second deck till you get an end point region outcode after clipping which is inside or on the box. That means it has region outcode 0 0 0 0 and that is what the second step is. Repeat the above steps if the other end point is also outside the area. That is fine because you have started clipping from one end and entered into the box and thrown off other parts.

You can start from the other end also and start clipping and stop at a point where you have the point after clipping after intersection. You are picking up those intersection

points when it is inside the box its region outcode is all four zeros and then that is where you stop.

Let us look at this example. We are taking an example of line  $L_5$  with the help of these three steps which you have just seen before, that is what we have done and we are taking this line  $L_5$ , endpoints are E and I of this line and I will take only the example  $L_5$  and I will leave the examples of  $L_4$  and  $L_6$  to work out because if I solve all of this you will not try to solve an example and that is the best process to learn. So please try to solve these examples yourself so that you learn with the help of region outcodes and you clip. I will solve one example and leave two with you.

(Refer Slide Time: 46:09 min)



Of course  $L_4$  as you see here  $L_4$  is much easier than  $L_5$ . So once  $L_5$  is done  $L_4$  is straightforward and  $L_6$  of course I leave it to you, a very interesting example for you to work out  $L_6$ . So let us take  $L_5$ . Endpoints are E and I. Where are the endpoints here?

This is one end point E and the other end point is I. Remember, this line crosses as you see here. It basically crosses 1, 2 and 3, it crosses three boundaries of the clip rectangle or the three edges of the clip rectangle as you may call boundary or edges. So, let us start with one of the end points, top to bottom or bottom to top. Let us start from bottom to top. To start with E, what is the outcode for E? You have the figure with you, it is also written on the screen, it is 0 1 0 0.

So with respect to the 0 1 0 0 if you see a bit position number 2, if you look into the overall chart of that region outcodes the second bit position will tell you that the starting point  $P_0 P_1$  whatever you have chosen is below the bottom horizontal edge, it is in the last set of rows of the three regions. Hence you have to clip this with respect to bottom edge since this point is below the bottom edge.

Remember, at least one bit value will be 1 if the point is outside the clip rectangle because if the point is inside, any point; edge point, arbitrary point, or an end point of a



line, if it is inside the rectangular strip the value will be all four zeros. Any point you take outside the clip rectangle will have at least one or two bit value as 1. So, look at that combination and find out whether it is below the bottom edge or above the top edge or to the left of your left vertical edge or to the right of your vertical edge. You can have conditions, four different conditions will give you four different 1s. It is of course it may happen that you are diagonally across, bottom of the bottom edge and to the left of the left edge or you could be to the top of the top edge and to the right of the right edge and that is all possible. In that case of course you choose any one that does not matter.

As you can see with the line  $L_5$  since the output is 0 1 0 0 it is to be clipped with respect to the bottom edge. Clipping with respect to the bottom edge gives F. So EI will be clipped to FI so we have the new line as FI here. So E will vanish to be replaced by F. F you will find that the point F outcode is 0 0 0 0. So you do not need to clip this line any further from the bottom, you have finished that part of the step. You start to look if the other end point is now outside the clip rectangle. Yes it is. The outcode of the other end point is I and that region outcode is 1 0 1 0. So you might clip this with respect to the vertical edge or horizontal edge. We will clip this with respect to horizontal edge let us say. So clip with respect to the top horizontal edge to replace I by H.

The intersection of the line FI now because E has been reduced to F after clipping from the bottom and the top when the rest of the line F of I FI is clipped with respect to horizontal top edge. You will get the intersection point at H and so the remaining part of the line will be FH. What is H? H has the region outcode 0 0 1 0. It is still not all four zeros so you have to clip it again with respect to which edge? Remember, the line now is from F to H.

I repeat again it is from F to H. The line  $L_5$  has been reduced, E has moved from F, I has moved to H after two clipping the remaining part of the line is from F to H. F region outcode is 0 0 0 0 as given here so you do not clip from the bottom anymore, but from the top still a part is left because the H outcode is 0 0 1 0. If it is 0 0 1 0 that is what you need to clip with respect to right edge to get FG. When you clip, H is replaced by another intersection G and the outcode of G is 0 0 0 0 so you now have two end points of the clipping line after three clippings F and G both having region outcode 0 0 0 0. So you replace the line EI by the final result of the clipping as the line running from point F to point G. Line F to G is the valid part of the line which is within the clip rectangle and that is the part which you display. **I hope the algorithm is very clear.**

The steps are available and the points are leveled in the figure and you see there was one clipping from the bottom, two clippings from the top and the line got reduced to the section of the line which now lies within the clip rectangle. I repeat, line EI will be replaced by FG which will be the clipped line for  $L_5$  and that is the final result which we displayed as FG. **I leave you with an exercise** for you to try the line  $L_4$  which runs from A to D. I can just give you some hints to operate this line as you can see that for the line  $L_4$  out of the two end points the point A anyway has a region outcode 0 0 0 0. So it is within the clip rectangle. You have to start from the other end point D which is at the top. So if you look at D when you clip with respect to vertical edge C that is also not within the box

then the you check the C region outcode, compare it with the top vertical edge, intersect there and get B. So the valid part of the final result of the line to be clipped is A, B. So that is the part to be displayed for  $L_4$  which is very straightforward, I have already solved that.

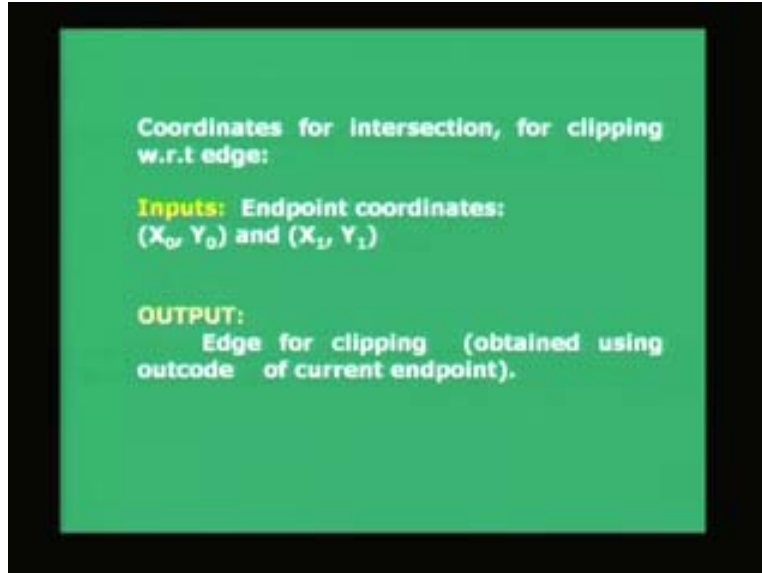
All the steps are not written so write those steps yourself. What about line  $L_6$  which runs from N to J, it is very interesting. You see here that you start from either bottom or top, you can start from say N one of the region outcodes will have a bit value 1. So you can start from N or J bottom or top whatever it does not matter. So if you start from N you will be first clipping it with respect to say, or you can start from the bottom let us say you can clip with respect to the vertical edge or if you start from top N you clip with respect to the vertical edge you find a point M which is also outside the box, start clipping it with respect to the left vertical edge you get point K that is also outside the box and then you move forward and you basically move to the point J.

So as you keep clipping, keep moving the line will be shortened and you will never get any two valid intersections which will somehow come inside the clip rectangle with four bit values as four zeros. This is the special case for line  $L_6$  which you have to deal with where you do the clipping operation but at any point of time you never have any point starting from top to bottom or bottom to top, any point will not be outside and the entire line will be clipped and thrown off.

Instead of this, previous cases of  $L_2$  and  $L_3$  if you remember in the previous figure not this one but a couple of slides back  $L_1$  was within inside the box completely  $L_2$  and  $L_3$  were completely outside the box. So trivial rejection acceptance worked there with bit values all zeros or two end points had a similar bit 1. So use that to throw off the line or accept the line completely.

Of course in the case of  $L_4$  and  $L_5$  we had to clip and find out which two endpoints are now within the clip with respect to end points and you throw off certain sections on line, adjust the end point of the intersection point, keep checking if the intersection point is getting inside the clip rectangle box. You stop at that point and start from other end point if necessary. So you get the valid part as you done for  $L_5$  you can do for  $L_4$ .  $L_6$  is a special case where you can start clipping but you will never come up with an intersection point which has the region outcodes 0 0 0 0. That means the end point will never reach or a clipped intersection point will never lie within the box. That is why  $L_6$  will be thrown out.

(Refer Slide Time: 48:41)



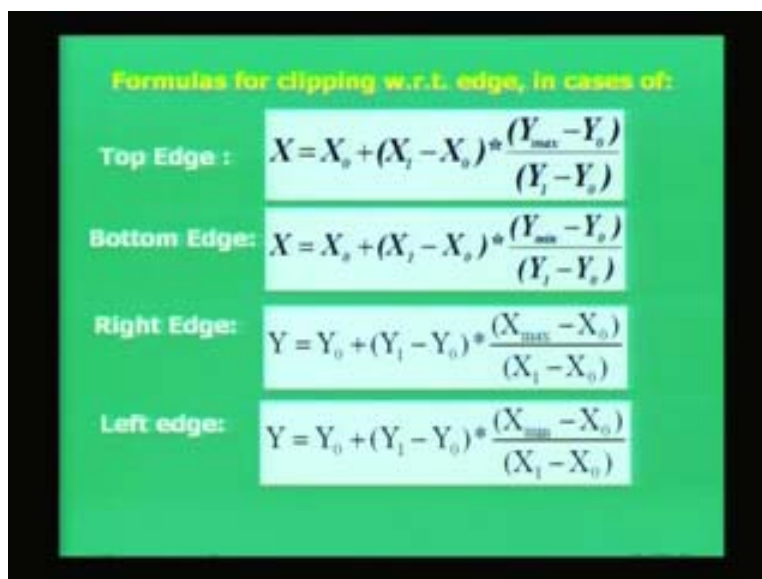
Coordinates for intersection, for clipping w.r.t edge:

**Inputs:** Endpoint coordinates:  $(X_0, Y_0)$  and  $(X_1, Y_1)$

**OUTPUT:** Edge for clipping (obtained using outcode of current endpoint).

So finally the steps of intersection before we wind up the lecture today is that the coordinates of the intersection for clipping with respect to edge can be obtained where the input to the algorithm are the endpoint coordinates  $X_0, Y_0$  and  $X_1, Y_1$  endpoint coordinates that is  $P_0, P_1$  having coordinates  $X_0, Y_0$  and  $X_1, Y_1$  and output is the edge for clipping obtained using the outcode of the current endpoint. That is the edge, output will be the edge for clipping, the clip part of the line which is inside box and it is obtained using outcodes of the current end points and these are straightaway the formulas which you can use for obtaining and the clipping with respect to the edge in cases of top edge, bottom edge, right edge and left edge respectively.

(Refer Slide Time: 50: 48 min)



**Formulas for clipping w.r.t. edge, in cases of:**

Top Edge :	$X = X_0 + (X_1 - X_0) * \frac{(Y_{max} - Y_0)}{(Y_1 - Y_0)}$
Bottom Edge:	$X = X_0 + (X_1 - X_0) * \frac{(Y_{min} - Y_0)}{(Y_1 - Y_0)}$
Right Edge:	$Y = Y_0 + (Y_1 - Y_0) * \frac{(X_{max} - X_0)}{(X_1 - X_0)}$
Left edge:	$Y = Y_0 + (Y_1 - Y_0) * \frac{(X_{min} - X_0)}{(X_1 - X_0)}$

You can see, when you are clipping a line with respect to four edges; two horizontal and two verticals. You are sure of one of the coordinates in any one of these boundary lines why? With the two horizontal edges your value of  $Y$  is fixed  $Y_{\min}$   $Y_{\max}$  you do not have to calculate that for the two vertical edges  $X_{\min}$   $X_{\max}$  was fixed. This was also for the case of the previous Cyrus Beck formulation. Use either a parametric form of  $t$  to obtain this simple formula and that is the main aim of the algorithm to find a simplest formula so that you can avoid all biggest calculations. So  $X$  is fixed here,  $Y$  is fixed here, it is the other part that means when you clip rectangle with respect to these two edges which are the horizontal edges you do not calculate the  $Y$  values of intersection because they are already given to you  $Y_{\min}$   $Y_{\max}$  from the coordinates of the bounding rectangle. You are interested to find out the corresponding  $X$  values for the line to be intercepted with these two edges. For these two vertical edges, for the vertical edges the value of  $X_{\min}$   $X_{\max}$  is given so that is what you do not calculate.

When you want to intercept a line with these two edges you are interested to obtain the  $Y$  value of intersection not the  $X$ . I repeat again, for the horizontal edges you are interested for the  $X$  value and for the vertical edges you are interested in the  $Y$  value. You can see here, top edge and bottom edge which are horizontal, you are interested in  $X$  values or you compute the  $X$  values for the right edge and left edge which are vertical edge you are computing only the  $Y$  values. And these are the expressions which you obtain using the parametric form. I leave it as an exercise for you to obtain these four intersection formulas given  $X_{\min}$   $X_{\max}$  for the right and left vertical edges,  $Y_{\min}$  and  $Y_{\max}$  for the top and bottom horizontal edges.

$X_0$   $Y_0$   $X_1$   $Y_1$  two endpoints of the lines are given. Find out these corresponding values of  $X$  for the top edge and bottom edge intersection  $X$  coordinates and  $Y$  values for the right and left vertical edges. These are the formulas which are given to you. They are of course given in the book Foley Van Dam. You can copy that from the slide which is being displayed.

We will wind up with that slide being displayed here with this lecture. But please try to derive these formulas which is almost similar to the formulas using the parametric form for the Cyrus Beck formula. So I hope the Cohen-Sutherland algorithm is completed. So in two lectures we have finished about two algorithms. We will move to the third algorithm for line clipping in the next class tomorrow which you will find to be the simplest one and also we will have to cover concepts about polygon clipping which will be very interesting.

Please look at the slide and copy these expressions and try to derive them as well and till the next class good bye all of you.