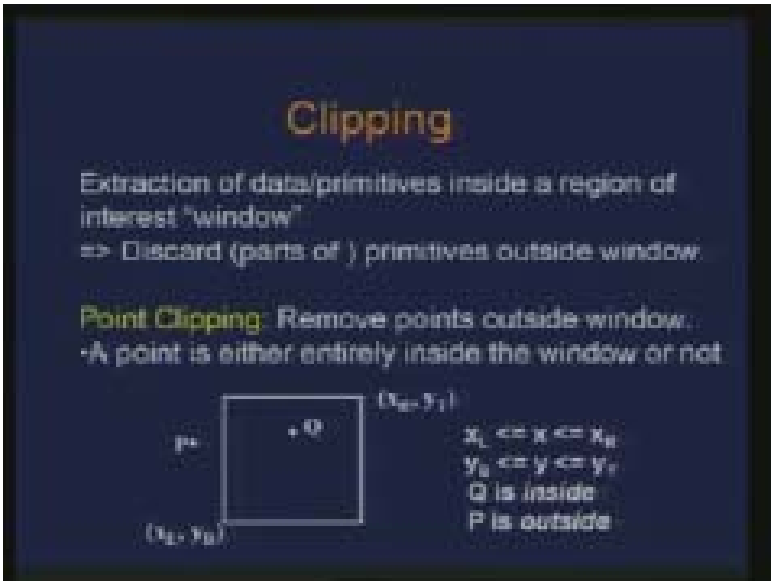
Introduction to Computer Graphics
Dr. Prem Kalra
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi
Lecture - 4
Clipping

We have been talking about clipping. The idea there was that we have a window that is the extent of the scene which we want to display and then we want to make a sort of an answer to a question whether a particular primitive is inside the window or not before we decide to display it.

(Refer Slide Time: 1:25)



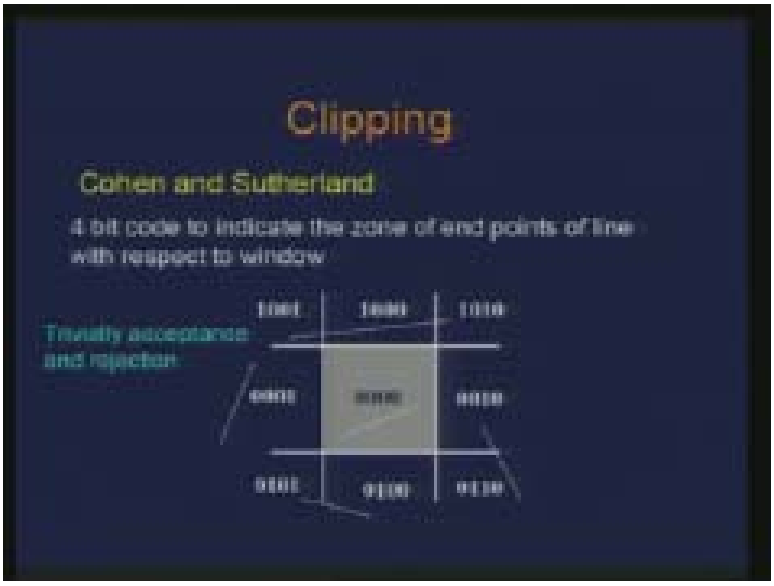
For instance we looked at the point clipping. The point is the simplest primitive so we wanted to answer this question whether the particular point is inside the window or outside the window. So, for the purpose of simplicity if I consider a rectangular window then this question can easily be answered by just looking at the coordinates of the point and the extent of the window to decide whether a particular point is inside or outside. Here P is the point which is outside the window or Q is the point which is inside the window so a simple comparison gives us that result. Then we also looked at the possibility of extending this idea of clipping a point to a line just applying at the end points of the line. And we figured out that it was not always possible that we could determine whether a line is inside or outside just by looking at the two end points.

(Refer Slide Time: 2:37)

In this respect we had a clipping algorithm suggested by Cohen and Sutherland where the idea was you are given a window which is shown here as a grey region and you want to decide things which are inside that window so what you do is you basically give a bit code to this window and the neighboring regions this window has then we actually devise a scheme for some trivial cases. Basically this process is, given one boundary of the window I can decide whether the line is on this side of the window edge or the other side of the window edge.

So, in case if it is on the other side which is actually the side whether the window exists I would declare that it is possibly visible. Therefore just by looking at then the bit code which I get for the end points of the line I can decide whether I need to trivially reject the line or accept the line. If the two end points are inside the window then the bit code for the two end points is going to be 0 0 0 0 and then I can basically follow some logical operation like taking an AND for the two end points which we figured out earlier where if I take the AND for the two end points then if I obtain non 0 bit with only exactly one bit on, then I can say that it is trivially rejected by this window.

(Refer Slide Time: 4:15)



These are the cases which we can see, the cases for trivially acceptance and rejection. Therefore this line is trivially accepted because the two end points have 0 0 0 0 as the bit code and here the two end points are actually on this side, if you look at this edge of the window they are on one side of the window edge and if I take the logical AND of the two end points I find out that it is exactly one therefore I can trivially reject it.

(Refer Slide Time: 5:19)

And however if the case is of this kind where this trivial rejection criteria does not get satisfied then I need to do some more work. So these are the cases like where you have one end point on this side and one end point is actually inside the window and I get a

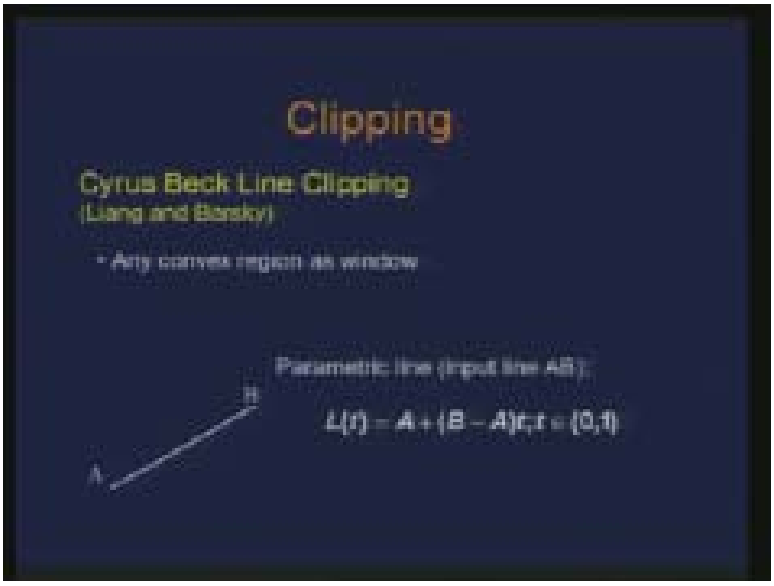
logical AND which is 0 for all the bits then I need to find out the intersection point here with the window edge so that I actually get the segments of the line which is inside and which is outside. So I can subdivide that line again at the intersection point and again check for the trivial rejection and trivial acceptance that is the idea. Here are also the cases which are actually not visible from this window, the two end points are here and here but they are potentially visible because of not satisfying the trivial rejection criteria. So we need to still work on to figure out the segments of the line and therefore check for each of the segments for trivial rejection and acceptance. Therefore this was the Cohen and Sutherland algorithm.

(Refer Slide Time: 6:42)

Properties of the Cohen and Sutherland algorithm: this is basically a very simple algorithm. Just checking for the end points and it is still popular because most of the time when you do clipping it is actually primarily against a rectangular region. It has a very natural choice for a window. So this limitation of rectangular region may be all right for the most of the practical applications, so still it is popular as a method. And the extension to three d as we saw it last time is fairly straight forward. So, instead of having 4-bit opcode we have actually 6-bit opcode and basically we are comparing it with the faces of the parallelepiped structure whether there is a line which can be trivially rejected or accepted.

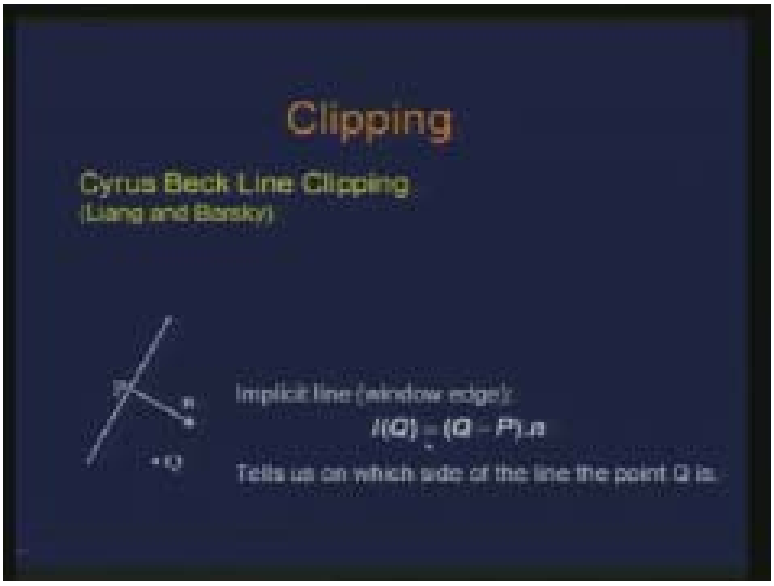
Now let us try to see some other way of doing this clipping where we can resolve certain limitations of this method. One of the limitations is that we are restricting this to a rectangular region. May be still we can design certain bit code for the two end points if we have a convex region. For instance, if I have a triangular region may be I can still do in a very similar fashion the way I did for a rectangular region doing some kind of a logical operation for the two end points but it might become cumbersome even for a general convex polygon. Hence there is another very popular clipping algorithm which is the Cyrus Beck Line Clipping Algorithm.

(Refer Slide Time: 8:39)



And in fact there is a slight variation to it from Liang and Barsky so they are sort of equivalent in terms of the way they work. Therefore this is applied with respect to any convex region as a window. You can take up any convex region and we will see later on that it could also possibly be extended for non convex regions. Therefore what we do here is we consider a line a line in question which is to be clipped as a parametric representation of a line. So if I have a line AB so I represent this line as L(t) is equal to A plus B minus A(t) where (t) is the parameter which runs between 0 and 1. So the range of 0 and 1 makes this line a finite line between the two end points A and B. So I have the representation of the line which is to be clipped as a parametric representation.

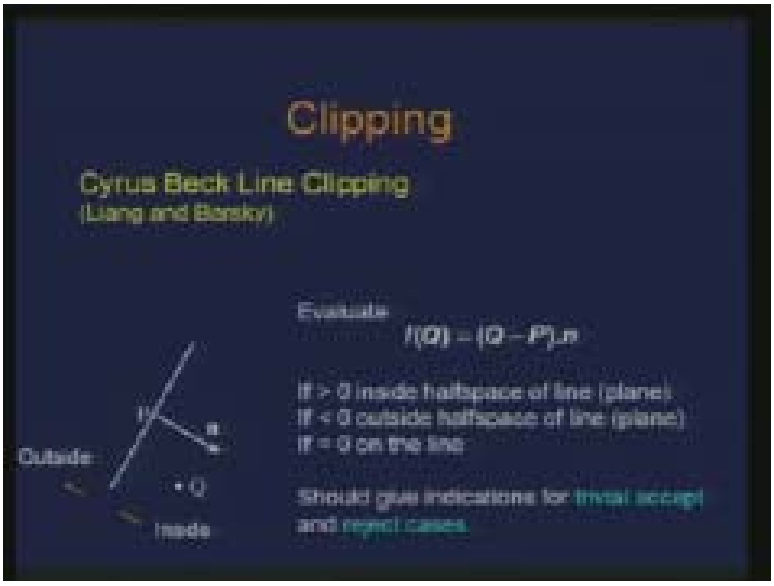
(Refer Slide Time: 10:06)



Now as far as the window edge is concerned, what I am trying to see it as is the window basically consists of several edges so I am trying to answer the question of clipping with respect to the window edges. And I represent a window edge using an implicit representation. Hence implicit representation basically means that I have a line LQ which is given as (Q minus P). n. It is actually giving me the answer to the question; on which side of the line the point Q is, given the point, P is on the line and n is normal to the line which I consider as defining for the inside of the line or a particular side of the line. It is something similar to what we also observed in line drawing algorithm where we were trying to answer the question of the location of the midpoint with respect to the line. So I have this implicit representation which actually enables me to answer the question as to on which side of the line the point Q lies. So this p point is actually any point on the line.

In other words, it is sort of a side distance of the point Q from the line and the sign of that gives me whether it is this side or that side. Now, given the representation which I am going to use for the line to be clipped and for the window edges against which I am going to check for clipping in implicit form let us see how we can go further for designing the clipping algorithms.

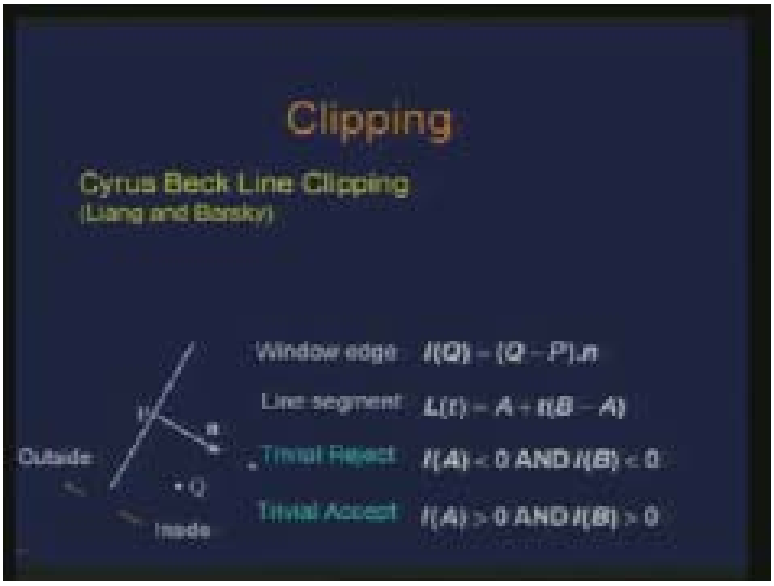
(Refer Slide Time: 12:26)



Here we basically we do an evaluation of a given point Q which is nothing but through the equation of the line itself and if we figure out that this point Q is greater than 0 then it is towards the inside half space of the line. So I have this notion of designing the inside and outside. So inside is this side where I have to find the normal line. This is again a similar kind of representation for half space. just by looking at the sign what I have from this evaluation if it is greater than 0 I have the point towards the inside half space of the line and if it is less than 0 it is on the outside of the half space of the line and if it is equal to 0 then the point is actually on the line itself.

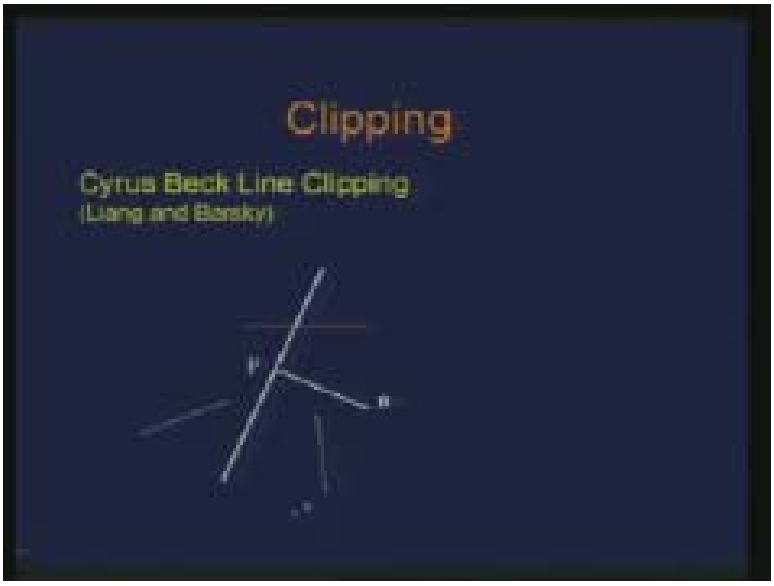
Now, in fact if you recall the way we were trying to design this trivial rejection and trivial acceptance similarly in the case of Cohen and Sutherland we can actually look at certain indications for trivial acceptance and trivial rejections from him. For instance if I figure out that the point Q turns out to be inside with respect to all the edges of the window then the point is going to be inside the window itself. And if it so happens that for the two end points of the line this is the case then the line is inside the window. This is equivalent to the trivial acceptance of the Cohen and Sutherland algorithm and similarly for the rejection because we basically that bit code was just going to do this inside outside operations with respect to the half space defined by the window edge that is precisely what it was doing. So we can do it in this fashion as well. These two are basically equivalent as far as the computation goes. How do you figure out the line is inside the region just while looking at one edge of the window? Basically what we will do is with respect to one line we check. If the two end points happen to be outside that window edge this is going to be rejected.

(Refer Slide Time: 15:38)



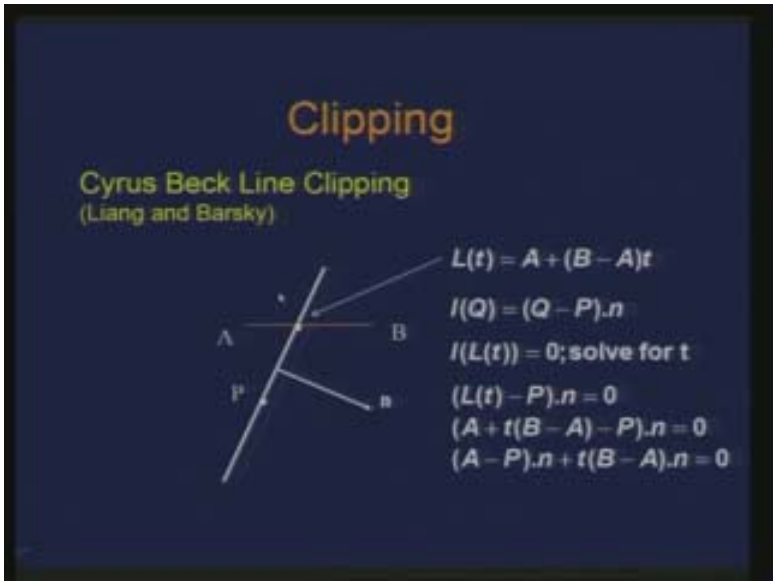
If I have given the window edge in implicit representation L(Q) is equal to (Q minus P).n and the line segment give in parametric representation L(t) is equal to A plus t(B minus A) then what I mean by saying trivial rejection is basically looking at the two end points of the line which are A and B so they should satisfy the criteria of being outside that line and it is negative less than 0. Both of them give me this negative value and similarly for the trivial acceptance. I will not be able to declare the line as such inside unless I have gone through all the window edges. So now we have basically looked at the cases of this nature.

(Refer Slide Time: 7:11)



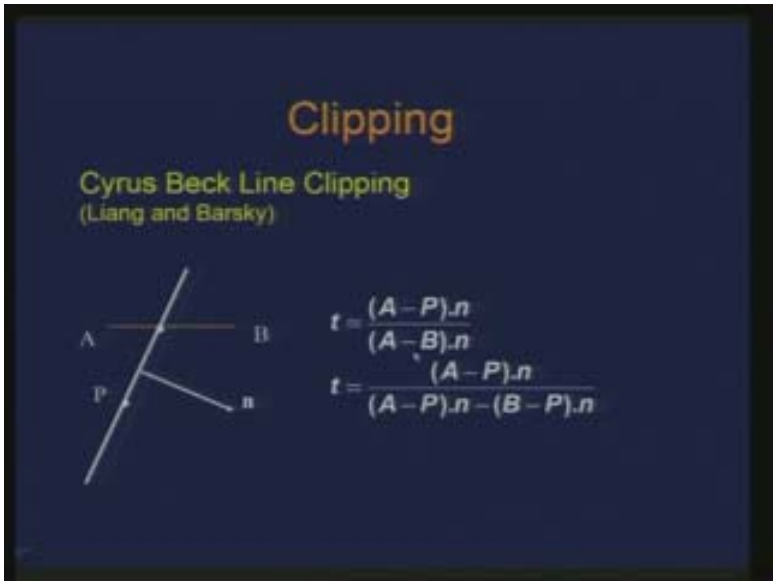
We are basically addressing questions with respect to one window edge. So this inside outside notion is actually associated to an edge and then we can run through with respect to all the edges and that is where we will get the algorithm design. So here this case we have resolved, this case also we have resolved and obviously there will be situations like this and one requires more work to do similarly as what happened in the case of Cohen and Sutherland.

(Refer Slide Time: 17:55)



Now with the given representations we have can we have these things in a sort of concise way. So when I say that I have this line AB in a parametric representation L(t) is equal to 2(A plus B) minus t and the window edge in the implicit form then finding this intersection. Now we are talking about the more work which we need to do when a line intersects the particular window edge which basically requires you to find out this point of intersection and that you can find out by just substituting this L(t) into the implicit equation you have and solve for (t). So it becomes a very simple solution. Here L(t) has been substituted and after just rearranging these terms I can figure out this equation (A minus P).n plus t(B minus A).n is equal to 0. All I need to do is get this (t). And of course t has to be given the range 0 and 1.

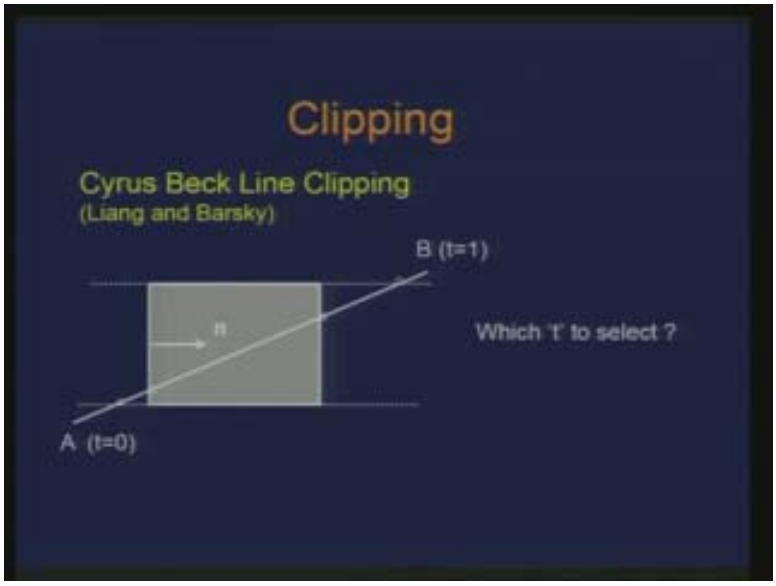
(Refer Slide Time: 19:32)



This (t) is determined as (A minus P).n/(A minus B).n which I can further write like this. So remember that when we were trying to address the question of these L(A) less than 0 L(A) greater than 0 we were basically looking at these (A minus B).n which is nothing but the same computation we did; A lid greater than 0 meant substituting A into my implicit equation of (Q minus P).n. Therefore as far as the computational aspect is concerned it was exactly this. So what I am trying to say is that the computation which I did for deciding my trivial acceptance and trivially rejection cases I just have to reuse them. I do not have to re-compute these I just have to reuse them and that is the advantage I have.

And now once I have done this then it is basically a matter of doing this with respect to all the edges. What will happen is that I can then redefine my As and Bs according to what I get as a point of intersection and give that to the succeeding edge the next edge. All I need to do is redefine these end points and go through the entire circle of my edges I have and at the end of it I am done. Clearly when you look at this, this does not restrict you to the form of the window I have. And the only condition I have is the window needs to be convex, it does not matter what shape I have. So it does not really restrict it to the rectangular shape of the window and I can apply this to any convex shape.

(Refer Slide Time: 22:35)

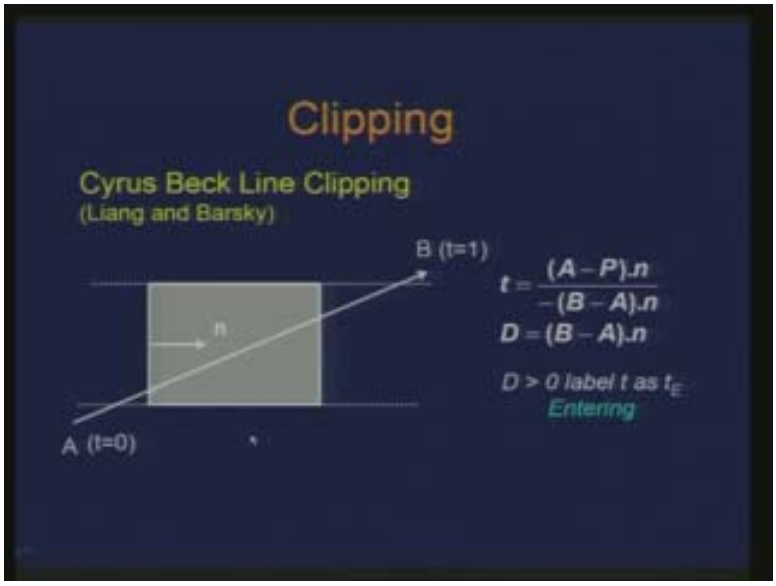


There is a small variation, instead of doing this evaluation of (t) and then updating my end points in A and B and then give it to the next window edge I try to find out the point of intersection with respect to all the window edges. That is where these two are slightly different Cyrus Beck and Liang. Here what I am trying to suggest is that given a line AB I have possibly four intersections with respect to I have a rectangular window with respect to each edge of the window.

Now the question which is being asked is that I find out these point of intersections basically the (t) which gives me the point of intersection then can I figure out anything from those (t) to be able to decide what segment of line is inside if at all the line is inside the window.

Basically it means, here is the line and you have these window edges and what I am trying to do is find out the point of intersection of this line with respect to the window edges which are four in this case and just looking at the values of (t) which I obtain can I suggest anything for the acceptance or rejection of the segment of the line which could possibly be displayed within the window. Basically if I have these four points of intersection then I am trying to answer which of these (t) I need to be able to label that this portion is inside the window.

(Refer Slide Time: 24:52)



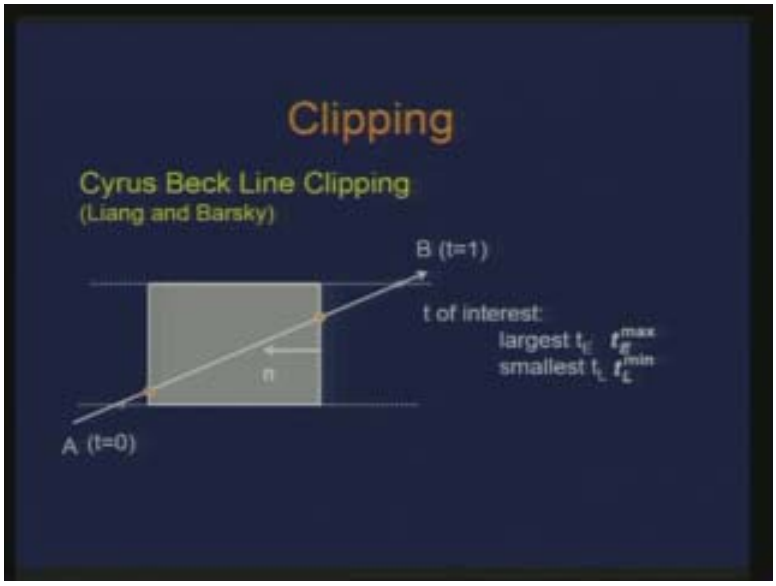
I have these (t)s which are computed as we have seen in this fashion, I have just actually swapped this (A minus P) to (B minus A) and put a negative sign in front, it is exactly the same thing. It just gives you a sort of a notion of direction. If I give for A(t) is equal to 0 and for B (t) is equal to 1 so it is suggesting that I am going from A to B.

Now let us try to evaluate this denominator term which I have here (B minus A).n which I call it as D just look at the sign of this. If the sign of this D is positive means D is greater than 0 then I label the (t) which I have computed as something like a t_E so E here is referring to entering. So the idea here is that, if I have a line like this going from A to B and this is my normal then it is in some sense saying that it is entering that edge or it is entering that space. So I label these (t)s as (t) which I find for D greater than 0. Similarly I have (t) for which the D is negative less than 0 then I label this (t) as t_L and here L referring to leaving. So here is the situation I am trying to illustrate.

Here you have this edge so this is the inside region for this edge and when I find out this point of intersection here so the (t) referring to this I label it as t_L because here when I look at this line it is in some sense leaving that region so I have this notion of entering and leaving which I use for labeling the different (t)s I obtain. Now if I can exploit this notion in order to decide which (t)s to select.

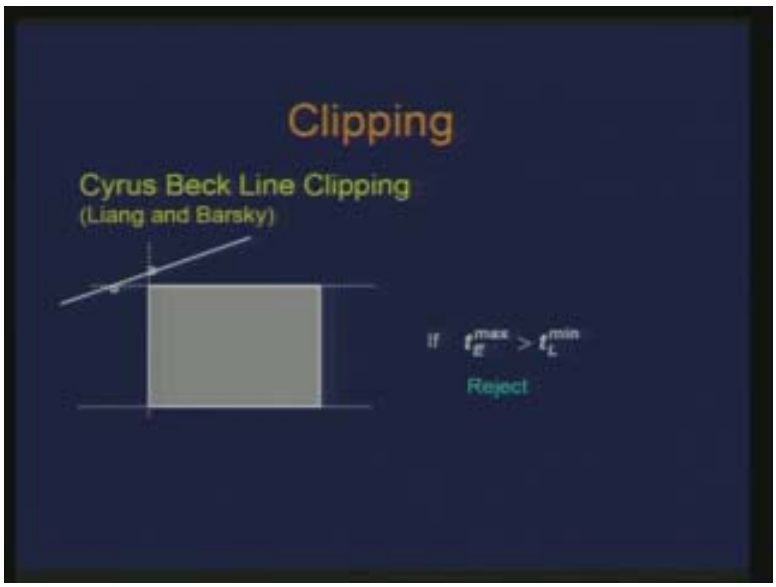
If I consider this particular line, and you know what portion of the line is actually inside the window and what (t)s will have the label E, these (t)s are going to have t_E and these two (t)s are going to have t_L. Therefore if I take that maximum of t_Es and the minimum of the t_Ls then it is fine.

(Refer Slide Time: 29:05)



Therefore this (t) and this (t) both these (t)s are of interest that is to be able to segment this line and which is to be clipped. So the (t) of interest is the largest t_E which I can call it as t_E power max and the smallest t_L which I call as (t) minimum. Now, in fact this can also help me further. Since I am doing just this evaluation of intersection point and then I also need to decide whether the line is actually inside or outside this can even help me doing this where this t_E power max is greater than t_L min.

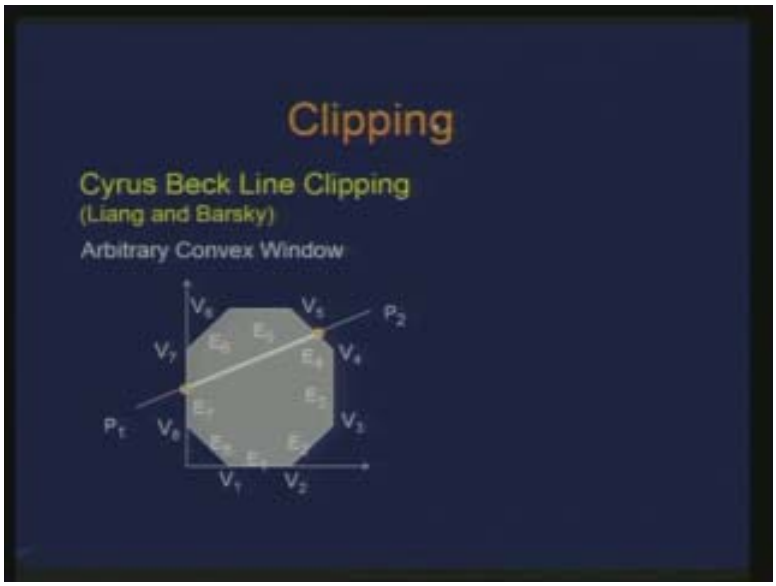
(Refer Slide Time: 30:00)



If this happens then I reject the line and that is what is happening in this case. This is what you will get as (t) max and this is your t_L power min so you will just reject that.

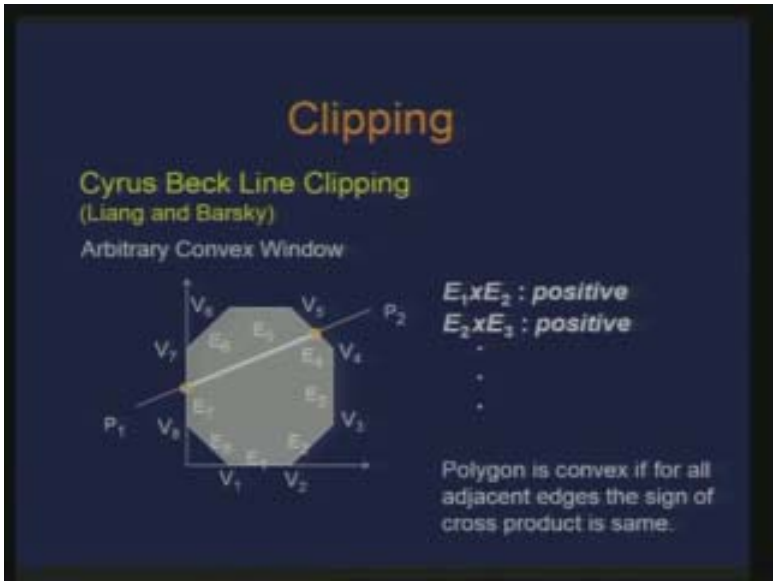
Therefore the evaluation of this t_E power max and t_L power min can also help you for rejecting the line. So this is basically based on finding out the intersection and the trivial rejection part comes after the evaluation of these (t)s.

(Refer Slide Time: 31:29)



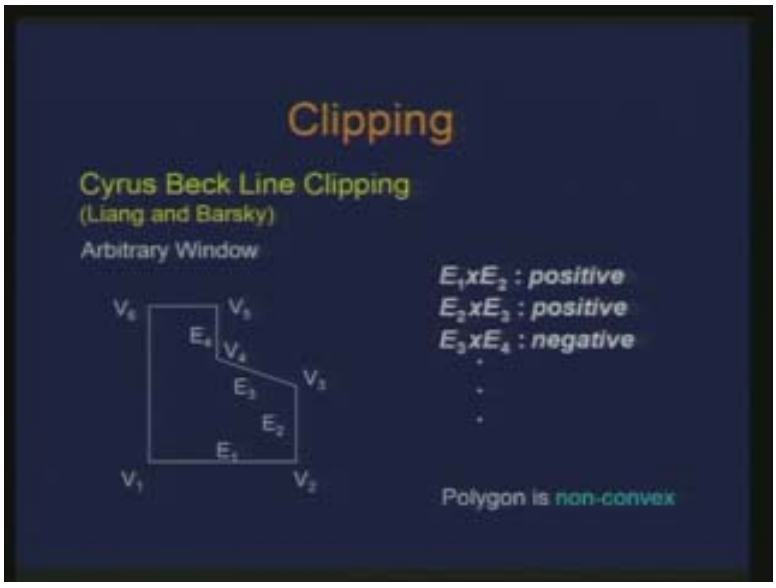
Now clearly as we have said there is no restriction to the shape except that the shape needs to be convex. So here, there is an example where I have taken a convex window which has got eight edges and P_1 P_2 is the line in question and in a similar fashion I did for the rectangular region I can do the clipping of this line and get these points. Now the question is that if I you give you a shape, so first if all you need to answer the question whether I can perform my clipping against that shape whatever I have given here, so you need to answer the question whether the particular shape is convex or not.

(Refer Slide Time: 32:57)



Therefore what you do is, you look at the cross product of the adjacent edges so here in this case when I take a cross product of E_1 and E_2 and my convention is, the thing which is coming out of this plane is positive then I say even cross E_2 is positive and so on. So, if I just take the cross product of all the adjacent pair of edges it turns out that in this case they are all going to be positive or if I had chosen a different convention they could have been all negative. So basically what we are saying is that if I find out that the cross product of the adjacent edges has the same sign then I declare that polygon as convex.

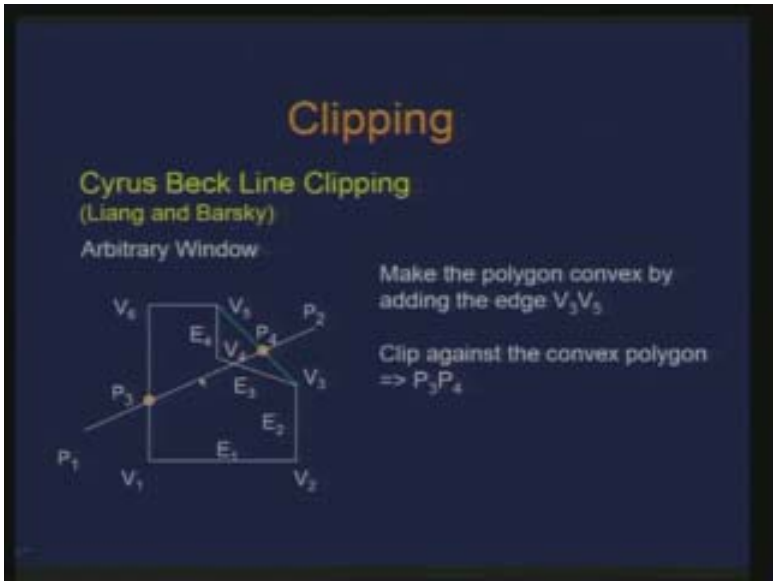
(Refer Slide Time: 34:13)



Let us take a shape which is non convex and see that this gets violated. So here is a non convex shape so clearly that is where you find the non convexity at this point V4. Then what will happen is that you observe that there is a change of sign at some point. All these cross products do not give you the same sign. Then you say that the shape is non convex. Now the question is, can we still do clipping for a non convex polygon or window?

The question which is being basically asked, can I map the problem of solving for non convex to convex by doing some sort of a preprocessing. So the first observation could be that if I generate convex polygons out of it, that is one way to look at and then I actually do this clipping with respect to the entire decomposed convex polygon. But often what happens is that when we are given such a non convex polygon the non convexity arises only at a few points may be 1. So may be what we can do is we can do some corrections to that non convexity and still apply the algorithm which is for convex and arrive at a solution. So with respect to the particular polygon what we can do is something like this, I make this polygon convex.

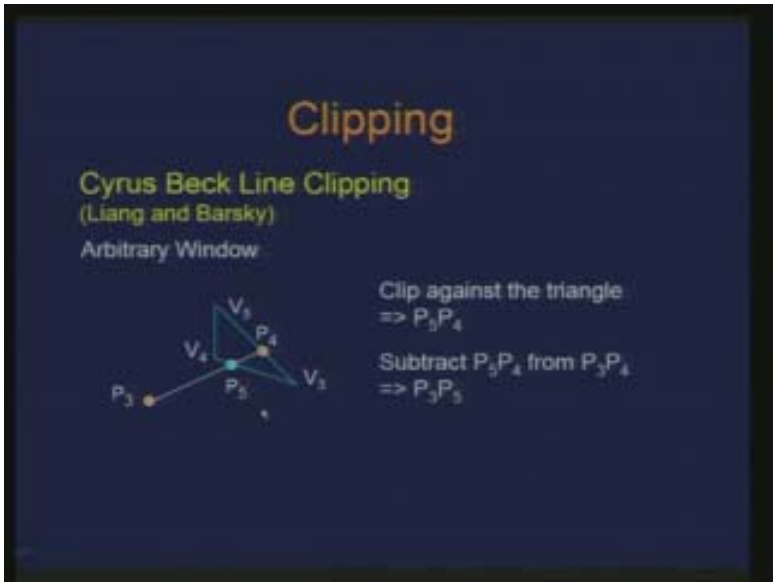
(Refer Slide Time: 36:59)



So, making convex the polygon may actually involve in finding out the convex hull of the points. In this situation may be it is just a matter of finding out where the change of sign occurred and then I take the two end points which were adjacent to it and join by line and make that polygon as convex. But this situation could not be that simple and you actually require finding out the convex hull. At the moment we are not really concerned of the exact algorithm of finding out a convex hull. We have a set of points and these set of points are nothing but nails, nails on a floor. You take a thread start from a nail and just wrap it around then what you see is whatever is given by that thread gives you the convex hull.

In such cases where we are interested in applying this clipping we are able to figure out the convex equivalent without going through an elaborate algorithm of convex hull. In this case we figure out just by looking at the adjacent points to the point which had given the difference sign and I join this line and I get a convex polygon. Then what I am trying to do here is, for the line which was given to me as P_1 P_2 and that is the line in question which was to be clipped I first find out the line which is clipped against this polygon the new polygon including this blue line. Therefore I basically get the points P_3 and P_4.

(Refer Slide Time: 39:53)



Then I consider this polygon which was sort of the additional polygon for making the other polygon convex. Then I do a clipping against this polygon for the resulting line I obtained which was from P_3 P_4 and I get the point P_5. So I just do a clipping of the line with respect to this polygon and then I subtract this from what I had obtained earlier. So at the end I obtain the line P_3 P_5 which is the resulting line to my interest. This is basically applying the same convex region algorithm and doing some preprocessing and post processing to get the line clipped.

There is the another interesting aspect to it, when I say that this is the portion I had got or the way we have seen the clipping algorithm which actually gives you the line which is clipped against the polygon which is inside the window I may call it as something like a interior clipping. In fact when I did interior clipping to find out P_4 P_5 I can do the exterior clipping that means the line which is not inside but outside that region to give me P_3 P_4. Therefore I can just have interior clipping and exterior clipping.

If you have seen exterior clipping being applied somewhere, what happens to your window management system? You have a window management system so one window overlays other window so on and so forth. And what you see is actually the exterior clipped display from the window which is in front. One can think that in terms of clipping. So this basically covers line clipping with respect to a polygon which is

generally a convex but even if it is non-convex for simple non convexity we can still apply a modified version of the same algorithm.

We started with points then we went to line and now I want to do polygon. I have a polygon as an input primitive, I want to do a clipping of this polygon against the window. So the question which is here is, can I extend the line algorithm to do polygon clipping? One thing we have not talked about is that this polygon could also possibly be filled, they may have shade, they may have color, and then there might be some problem.

Polygon clipping: this is the portion to be discussed next. So, given a polygon as an input primitive how do we clip that against a window? Once again when we talked about the Cyrus Beck and Barsky Line Algorithm just the way we had seen the extension to 3D in the case of Cohen and Sutherland. Here also it is straight forward. Here we have considered window edges as the lines against which we are trying to find out the clipping. There we will have window planes. So we will have a volume given by these convex regions and each of the faces of that volume is going to be a plane. But the test we are going to have will have exactly the same computation because we are basically answering the question of inside outside or this side or that side. So instead of saying that with respect to a line we will say that with respect to a plane. Therefore again it is a matter of checking the sign of that computation. So that can be applied even for 3D. Therefore 3D clipping is a just a straight forward extension of what we have seen as 2D. Non convex regions in 3D could slightly be more difficult making them convex.