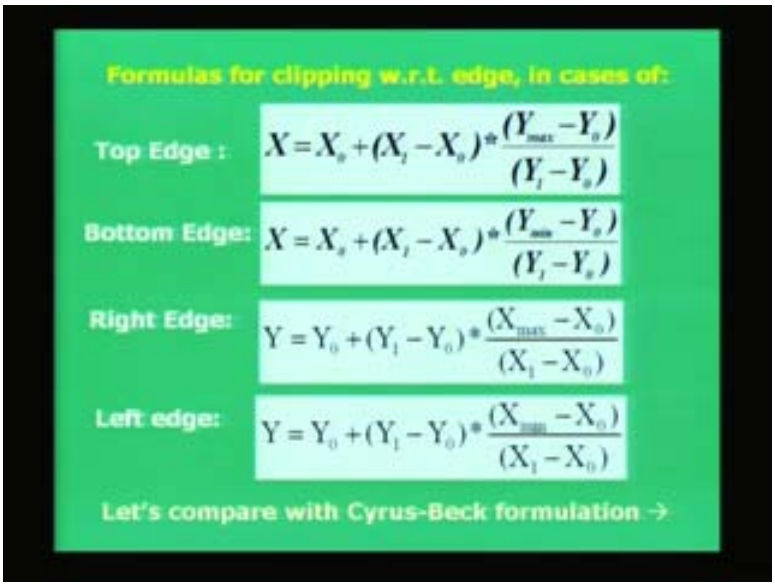
Computer Graphics
Prof. Sukhendu Das
Dept. of Computer Science and Engineering
Indian Institute of Technology, Madras
Lecture - 22
Clipping Lines

Hello and welcome back to the lectures in computer graphics. We have been discussing in the last two classes about clipping lines, algorithms on the method of clipping, straight lines in two dimensions. And we have gone through two formulations the Cyrus Beck and Cohen-Sutherlands algorithms for clipping. Both use the parametric form of the expression of a line for clipping with respect to a rectangle. The rectangle is defined by two vertical edges with X_{min} X_{max} and two horizontal edges Y_{min} Y_{max} coordinates. And given a particular line in parametric form two end points P_0 and P_1 we now know with the help of these two algorithms how to clip. One uses the region outcodes which we have covered in the last class using Cohen-Sutherland and the other method by Cyrus Beck formulation talks about using the functional form of a dot product between the normal to the edge which is being used to clip the line and the vector which dictates the line.

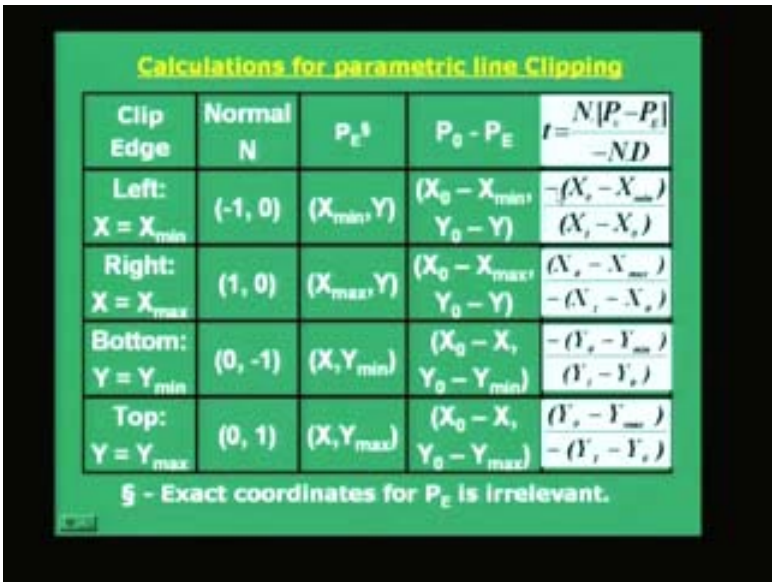
So, we a have functional form on one side and the region codes on the other side. But if you look back into the last slide which we talked about the table, if you look back into the slide we discussed about the formulas of clipping with respect to the edge. In case given the line with point P_1 to P_0 these are the formulas to obtain the intersections X and Y intersections we showed them last class and I hope you could work out them very easily as to how to obtain the intersection X and Y coordinates with respect to the four edges of the rectangle with being used to clip a particular line P_1 P_0 that means X_0 Y_0 X_1 Y_1 are given to you and the edges have corresponding X_{min} X_{max} and Y_{min} Y_{max} as well.

(Refer Slide Time: 2:55)



If you compare this with the Cyrus Beck formulation, in the first class we also showed a table which talked about the Cyrus Beck formulation, you see here, I hope you remember this parametric line clipping calculations which uses the concept of functional form of N. P_0 minus P_E divided by the denominator N. D. If you look at the extreme right column the expressions given here, the numerator and the denominator terms almost exactly match with the corresponding one given here by the Cyrus Beck. Why this is so? This is so because both use the parametric representation of a line to obtain intersections. In both cases the simpler fact being that you are using two horizontal lines and two vertical edges and the expression become simpler.

(Refer Slide Time: 04:05 min)



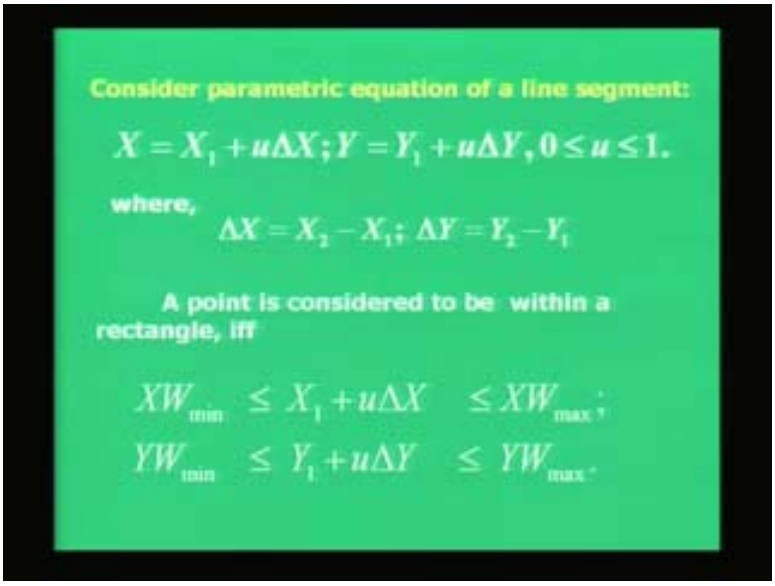
But you should be able to derive as I said before, earlier in one of the classes that given two lines in parametric form you should be able to obtain intersection point using parametric form of a line. So if you look back into this particular case for polygon clipping. Let us look at the expression for top edge, if you look in to the right hand side numerator divided by denominator term here the ratio and look at the top edge of the bottom of the screen now for the Cyrus Beck formulation you see they are same here Y_0 minus Y_{max} the bottom of the screen, the right column Y_0 minus Y_{max} by Y_0 minus Y_1 is the same as those given here.

Similarly, for bottom, right and left edges also you will find that the expressions exactly match. So you are basically using the same expression in the Cohen-Sutherland as well as in the Cyrus Beck formulation for clipping the line. The methodology or the algorithm is little different, the 1 uses the region outcodes we know, four bits MSB LSB, one bit for each particular type of region towards the left top or bottom three or top bottom right whereas in the case of Cyrus Beck we use the functional form of dot product of two vectors.

Look at the angles for the sign and then either eliminate the edge or clip it. So these were the two differences of the two different algorithms. The formulas are same in terms of clipping, the algorithms is little different, one uses the region outcodes, one does not and the other uses the functional form and I hope it is clear. We will move on to the third formulation for line clipping. You see on the slide, the name is given by the proposals it is called the Liang-Barsky Line Clipping.

Now this line clipping algorithm is probably the simplest out of the three which we will study in fact. We have studied two already and the third one today and we know for line clipping we will look and will see in fact the formulations are much much simpler.

(Refer Slide Time: 09:15 min)



Again as you look back into the slide we talk of considering parametric equation of a line segment. This is similar to the parametric expression taken earlier in the previous two algorithms except that in the previous two we use the expression P_0 in the vector form equal to P_1 equal to P_0 plus sometimes some t. In this case instead of the parameter t you can see the reuse of the parameter u which lies between the range 0 to 1 here and instead of a vector equation we split it up into two linear equations of the form of X coordinates X_1 plus u delta X. Delta X is the difference in X coordinates of the starting and finishing point of the line, delta Y the difference in the Y coordinates of the starting and finishing point of the line respectively.

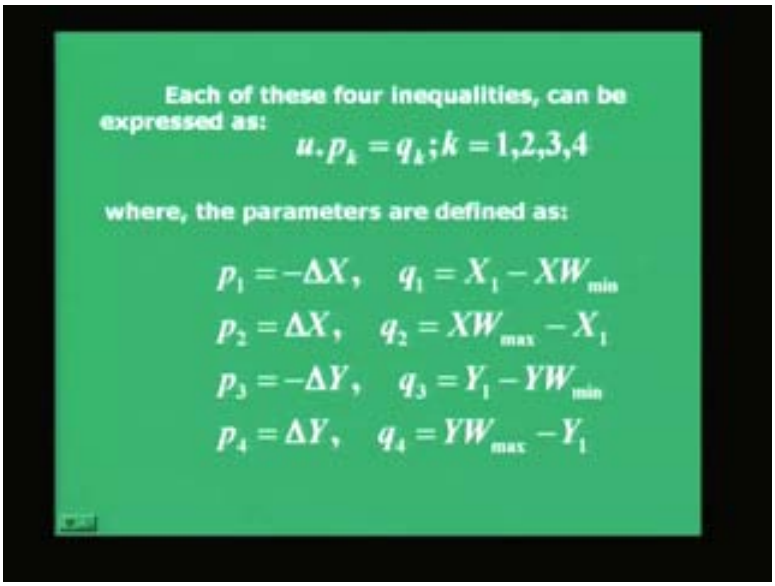
So delta X and delta Y are obtained from the two end points of the line and you substitute it back and for different values of u basically when u is equal to 0. In fact you are at Y_1 and when u is equal to 1 you can substitute and you can see that you at are at X_2 Y_2. So point is that instead of P_0 to P_1 we say it starts at X_1 Y_1 and terminates at point X_2 Y_2 and by varying u between 0 and 1 you are moving from the coordinates X_1 Y_1 or one end of the point on the line to the other end of the point so that is the parametric expression.

If you look ahead the point is considered to be within a rectangle of course given the X_{min} Y_{min} X_{max} Y_{max} for the edges of the rectangle are given to you then we will say that if this inequality will hold good in fact there are four of them, you put together in two equations then if it is satisfied for any particular point X_1 Y_1 X_1 plus delta u X of the middle Y_1 plus

u times delta Y at the middle. And so if this inequality holds good for any point X_1 Y_1 lying less than XW_{max} YW_{max} and also greater than XW_{min} YW_{min} where I must admit here that XW and YW are the X and Y coordinates of the window or the rectangle used to clip the line instead of X_{max} Y_{max} X_{min} and Y_{min} which we have been using earlier.

Similarly, we are using different notations here XW_{min} XW_{max} YW_{min} YW_{max}. So we are talking of two horizontal edges top and bottom, left and right vertical edges as like the previous case but the variables which are used are different. So the point here is considered to be within the rectangle if and only if this inequality holds good and each of these four different inequalities can be expressed as four different equations u. p_q is equal to q_k where k runs from 1, 2, 3, 4 for four different equations. These four equations are nothing but these four equations at the bottom.

(Refer Slide Time 09: 23 min)



It says that I am expressing four inequalities together in two equations here and these are the four equalities in short form where the parameters p_k and q_k are defined as these. You can easily work out this if you have written the previous four inequalities in the last slide given in two expressions then you can easily write it in to four different equations separately with the parametric form p_k and q_k where p_1 and q_1 to p_4 to q_4 are given in the particular form. So that is very easy for you to visualize. I repeat, the expression once again I am talking of these two, at the bottom you see these two expressions, four inequalities given in two expressions and they are expressed in this particular form.
So the p_{is} and q_{is} can easily be calculated.

Why? p_{is} and q_{is} depend on the horizontal and vertical edges of the clipping window or rectangle which is used to clip a line and the line has correspondingly coordinates X_1 Y_1

to X_2 Y_2 so delta X and delta Y can be computed from the end points of the line and the XW_{max} XW_{min} similarly YW_{max} YW_{min} can be computed from the rectangle of the window which is use to clip the line. So if you look back, the parameters are known to you. So the p_{is} and q_{is} are known to you and you can form these two inequalities in terms of the parameter.

I hope this point is clear.

So based on these four inequalities we can find the following conditions of line clipping which says that if you read the slide here if p_q is equals 0 that means if the parameter p_k you know the parameters p_k and q_k now. So p_k is equal to 0 the line is parallel to the corresponding clipping boundary that much you should be able to see. If you have copied the expression p_k in the previous slide you see the expression of p_k they are nothing but the delta X and delta Y. So p_k is equal to 0 basically means the corresponding the delta X and delta Y is equal to 0.

I hope that this point is very clear and delta X basically means that the line is correspondingly parallel to one of the vertical edges in fact to be very precise. And for k equal to 1 we are talking of the left vertical edge k equal to 2, we are talking with respect to the right vertical edge k equal to 3, we are talking of the bottom horizontal edge k equal to 4, we are talking of the top vertical edge respectively.

So for correspondingly k equal to 0 if you look back into this expression in the left hand side the p_k terms they are nothing but the delta X and delta Y respectively. That means you are talking of the line being parallel to either left, right, bottom or top edges depending upon the value of k that is what the expression is. Next we see, if for any k for which the line is parallel to anyone of the respective edges that means p_k is equal to 0. If you find that the other constant q_k for any k is less than 0 then the line is completely outside the boundary.

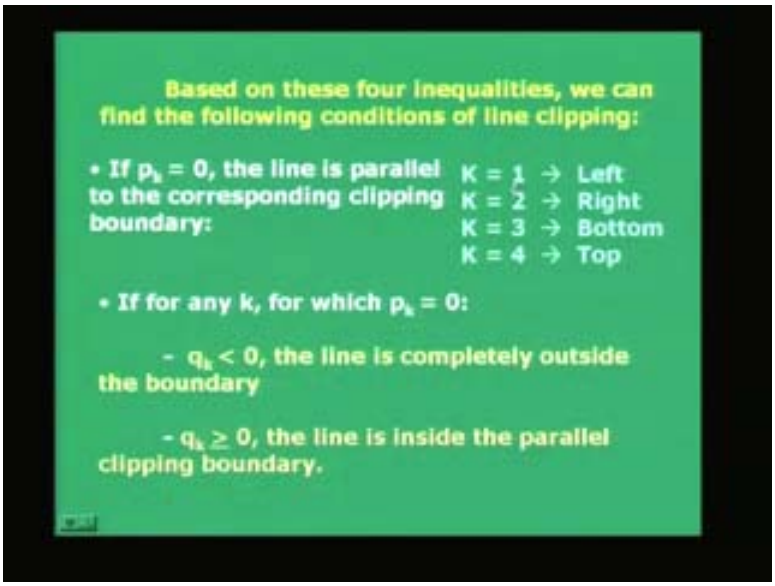
What is q_k? Let us look back.

q_k is this particular X_1 minus this. So looking at the positive or negative value of q_k as you look on the right hand side obviously you can visualize in very straightforward manner that if q_k is negative the line is completely outside the boundary and if q_k is positive or equal to 0 the line is on the boundary or is inside the parallel clipping boundary.

I leave this as an exercise for you to check it out by taking examples that when correspondingly p_k for any k is equal to 0 and depending upon the sign of the q_k I will just look at the sign of the q_k variable and find out whether the line is inside the clipping boundary or outside, completely outside or inside the parallel clipping boundary. This can be easily seen from the expression itself. If you look into the expression and compare yourself the terms X_1 Y_1 and so on with XW_{max} and so on because for any k values between 1 to 4 signifies one particular edge of the clipping boundary.

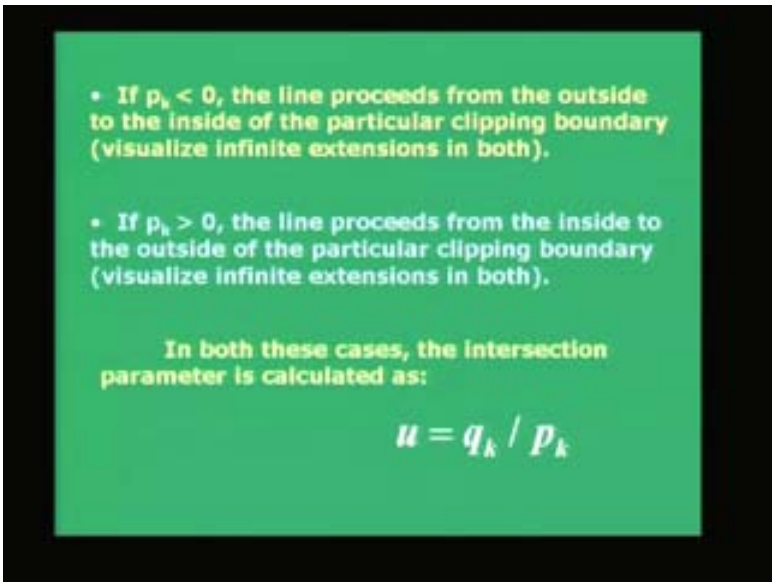
So we are comparing with respect to that particular edge which is signified by the value of k.

(Refer Slide time: 13:41 min)



And as given in the slide I repeat again, k equal to 1 we talk of the left vertical edge, k equal to 2, right vertical edge k equal to 3, the bottom horizontal edge and k equal to 4 the top horizontal edge respectively. So that is always the case for different four values of k. So now we know that when p_k is equal to 0 this is what we do, we look into the sign of q_k and decide whether it is completely inside or outside the boundary. What happens if the value of p_k is negative that is less than 0? In this case the line proceeds from the outside to the inside of the particular clipping boundary and we have to visualize infinite extensions in both directions and say that the line is proceeding form the outside of the clipping boundary to the inside.

(Refer Slide Time: 15:10)



If it is on the other hand, if p_k is positive the line proceeds from the inside to the outside of the particular clipping boundary and visualize infinite extensions from both. Remember, the expression of p_k involves the delta X. So we are talking of X_2 minus X_1 depending upon, you know which end point? Which point you are traveling? Left to right or right to left? You could travel from the outside to the clipping boundary to the inside or from inside to the outside. It depends upon which endpoint ordering you are considering when you are moving form one end point to the other of line.

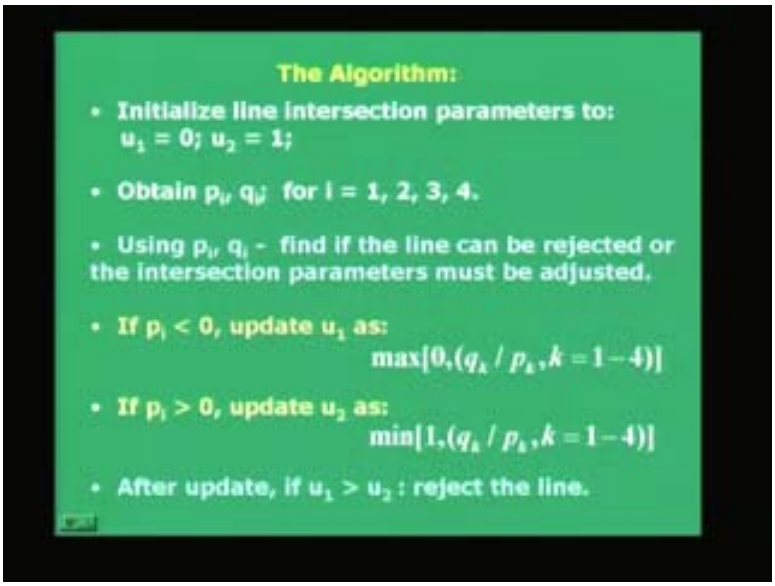
But we know first of all if p_k is of course 0 then we are talking about the horizontal or vertical lines only and then you will look for p_k q_k to find out whether it is inside or outside. But if p_k is not equal to 0 we again look into the sign of p_k and tell whether the line is traveling from outside to the inside of the clipping rectangle. So that is what inside to outside or vice versa based on the sign. So remember these two terms negative it proceeds from outside to inside. If it is positive then we are talking from inside of the clipping boundary to the outside. In both cases the intersection parameter is calculated as following. That is a very simple expression; this is the advantage of the Liang-Barsky formulation.

You see the expression is the simplest form q_k by p_k depending upon the value of k that means what? We did this in the Cohen-Sutherlands and the Cyrus Beck formulation also that we had four values of t and based on that we have to clip lines. So here also we are doing the same, four values of the parameter u instead of t what we have used in the expression and you get four values of u for corresponding to four different values of k 1, 2, 3 and 4. And these are calculated by the expression as given here which is the ratio of q_k by p_k. What does q_k and p_k depend on?

For any particular value of k we have seen that the corresponding value of pk is just delta X and whereas the other value depends on the X_{max} Y_{max} and so on. So based on the coordinates of the clipping boundary and the coordinates of the two end points of the line you can compute q_k and p_k for different values of k and then take the ratio that gives you four different values of u. This is the computation required for Liang-Barsky algorithm and of course it is required only in the case when you find that the sign of the p_k is not equal to 0 when it is negative or positive. If it is 0 we know what to do. So the algorithm, once we know the concept we look at the algorithm for the Liang-Barsky which is the simplest form out of the three. Initialize line intersection parameters to u_1 and u_2 for both 0 and 1.

We use two intersection parameters because we know that either a line intersects a clip boundary at two points. That is why you have t_1 t_2 or u_1 u_2 or it does not intersect at all. That means u_1 u_2 are not lying in the line 0 to 1. Or if they are two parallel lines of course you do not have a finite u in this case previous case t where the line intersects. So this u_1 and u_2 are the parameters which the line intersects with the edges of the rectangle and then for each different values of k which we use an index I in this case obtain the p_{is} and q_{is}, the expression was given to you few slides back. So based on the end point coordinates of the line, based on the bounding rectangle X_{min} X_{max} Y_{min} Y_{max} you obtain p_i and q_i.

(Refer Slide Time: 20:41 min)



Then using the values of p_i and q_i find if the line can be rejected or the intersection parameters must be adjusted. This, we looked into the expression earlier to find out based on the q_i value where you need to check whether the line is completely inside the rectangle or completely outside it or it is traveling from inside to outside. So two conditions we have seen and that is what we use to find out the line. So if p_i is equal to 0 you update u_1 which is set to 0 as the maximum of the set 0, ratio of q_k by p_k where k is equal to 1 to 4. So obtain the ratio q_k by p_k for all the values of k because the p_{is} and q_{is} are known. So now I have switched the index to k but it is the same and so obtain these four ratios and check if any one of them is more than 0 that means it is positive. So obtain definitely between 0 and 1 and obtain the maximum value. That is what you do for u_1. On the other hand, for u_2 if p_i is positive then you update u_2 as minimum of 1 and the ratio again q_k by p_k where k is running from 1 to 4.

So that is what you do, minimum of this, that means expecting here q_k by p_k to be less than 1 then only you pick up one of these k_s and of course if any one of them less than 1 for updating u_2 otherwise you leave u_2 to be 1. On the other hand, for the case of u1 since you have initialized to 0 here, you check if one of these ratios are more than 0 or in fact what you mean by simply taking the maximum of five different values.

So after update you check whether you have got a case where u_1 is more than u_2. If u_1 is more than u_2 then you reject the line. This concept is similar to what you used for the Cyrus Beck formulation also with the PE and PL. If you remember entering the half-plane leaving the half-plane for the Cyrus Beck formulation if you found that typically you expect the PE to be less than PL but if the reverse happens in a case when PE is more than PL with the line, it is the similar case here because we expect u_2 to be more than u1 but if after all the updates if you find u1is more than u2 then you should reject the line. So that is the end of the algorithm. Liang-Barsky is very simple.

Now what I have purposefully done is I have not taken any example to illustrate the values of p_i q_i and also u_1 u_2. But if you remember in both the previous formulations, previous two algorithms both in the case of Cyrus Beck and Cohen-Sutherland I have taken examples of three or more lines to illustrate those two algorithms.

I am leaving this as an exercise now for you otherwise you will not be able to understand. Please take a pencil and a paper and take those examples given in the earlier two classes. That is for which of the two algorithms? Cohen-Sutherland for region codes and the Cyrus Beck formulation for the dot product of two vectors. So, for those two algorithms I have taken some examples of lines from inside the rectangle, some outside the rectangle and some overlapping that means it is coming from outside to inside or going from inside to outside or in a traversing through part of the line be inside the rectangle. So all these different combinations in fact for the Cohen-Sutherland I remember I have taken six different lines.

So for those six different lines I would request you to please try Liang-Barsky formulation. That means you computer the q_k's and p_k's or p_{is} and qis for four different values of I corresponding to four different edges. Check and then apply the algorithm based on the sign of q_k's and p_k's or the q_{is} and p_{is} depending upon whether the line is parallel or not parallel and apply this update formulation and check how the clipping occurs.

Please does that to understand for yourself because I believe if you understand the previous two algorithms, understand the method of algorithms partly at least now without an example you should be able to solve an example yourself and that will definitely give lots of confidence. So please take the previous two examples in the previous two problems which I have solved and please work out. And take this as a home assignment to work out the Liang-Barskys formulation with this simple expression where the expression we found is the simplest form u_i. u is equal to q_k by p_k and you knew the expressions of q_k and p_k given here. That is what we use and there is no necessity to look at the region outcodes and there is no necessity to take dot vector, look into the dot vector whether it is 0 or not then divide all those where done in the previous two formulations.

Liang-Barsky is very simple in terms of the approach and very fast to implement because you have the ratio of two integers, find out the floating value t and that t substitutes back into the expression for the parametric form of a line and you get the X Y coordinates of the line if it is to be clipped. If it is rejected of course you do not need to compute the value of u that is of course the case earlier when we discussed of parameter t in the case of Cyrus Beck or Cohen-Sutherland algorithm.

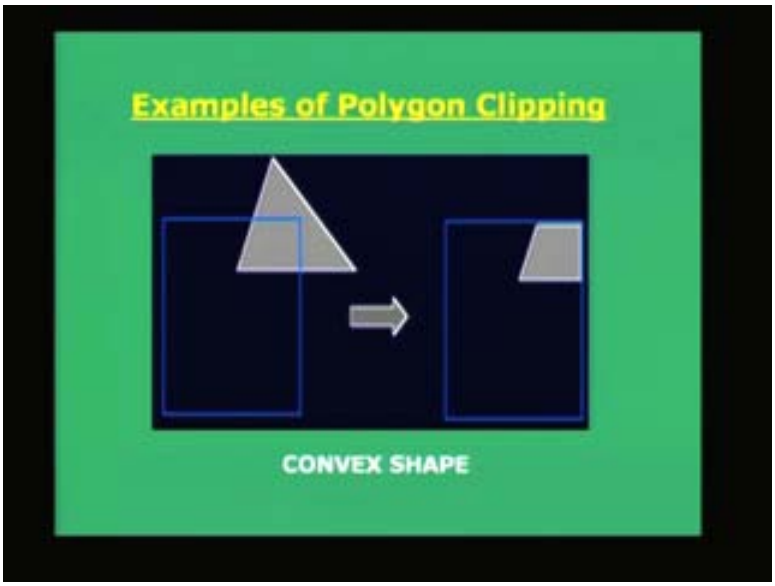
But in the case of Liang-Barsky you compute the u as well when you need to clip the line otherwise you do not do so. So these are the three algorithms which we have covered in this part of the course for clipping the line. Then the next few remaining part of the lecture today we will look into the examples of polygon clipping and find what are the interesting phenomena there with respect to clipping a polygon with the help of the rectangle. So look back into slide we will get into algorithm or a method of polygon clipping.

If you remember polygon filling, polygon is made up of lines, a set of n vertices define a polygon. If there are n vertices there must be n different edges or sides of the polygon. And that polygon may be filled but we have to now clip around filled polygon with the help of the rectangle. And basically since polygon is made up of lines clipping a polygon involves the lines. So we will use one of these algorithms, we know that. We have one of the algorithms at hand.

We have a function at hand where we pass a line and a clipping rectangle, the line could be easily be clipped. So assuming one of those algorithms which we will use for line

clipping if necessary we will see how polygon can be clipped. Let us look at an example of polygon clipping using a convex shape.

(Refer Slide Time: 25: 51 min)



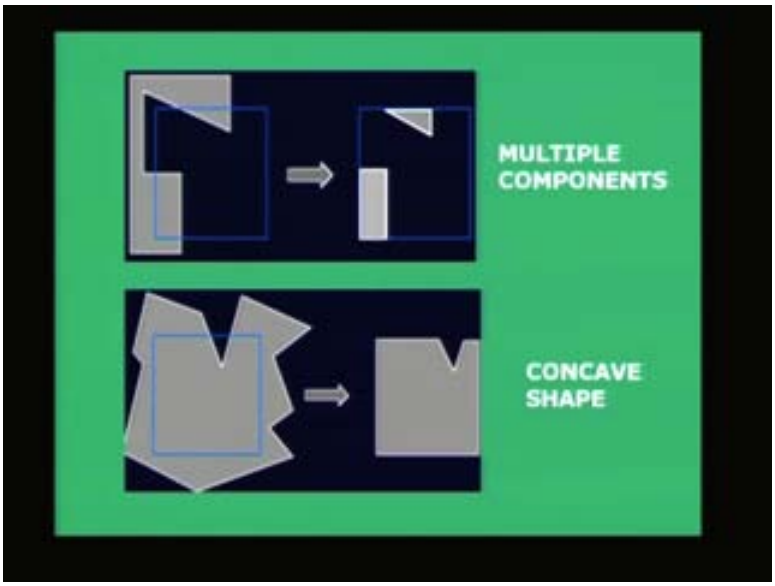
Now, a polygon could be of course in the convex shape or a concave shape depending upon the property of the convexity of the polygon. I hope you understand what the convex polygon is and the difference between the convex and concave. The concept is very simple by chance if you not heard about it, if you take a line completely inside the polygon completely inside a polygon, two endpoints defining a line completely inside the polygon, if these two points are inside the polygon then all those points must also be inside the polygon if the polygon shape is convex.

If it is concave it is not guaranteed, it may be or may not be, that is the difference. I repeat again, take a line where the two endpoints are inside the polygon, arbitrary shaped polygon let us say and I take two endpoints, if all the points of that line are inside the polygon also then the polygons are convex in shape otherwise they are concave in shape that is the difference. We are interested in polygon clipping so that is the example of the convex shape polygon in this case of course a triangle. So the left hand side triangle, the convex polygon will be split by this window then I should obtain this result polygon. I can have this, I can have multiple components clipped out.

This will be a very interesting problem. I can have this in case this is a case of concave polygon as like the case even here we look at this concave shape, both are concave shapes but in one case of course I have one component at the bottom when I clip with respect to this blue rectangle but on the other case on the top I can have multiple

components coming that means when I clip I have two independent polygons now inside the rectangle. I can have two or more that is possible.

(Refer Slide Time 26:27 min)



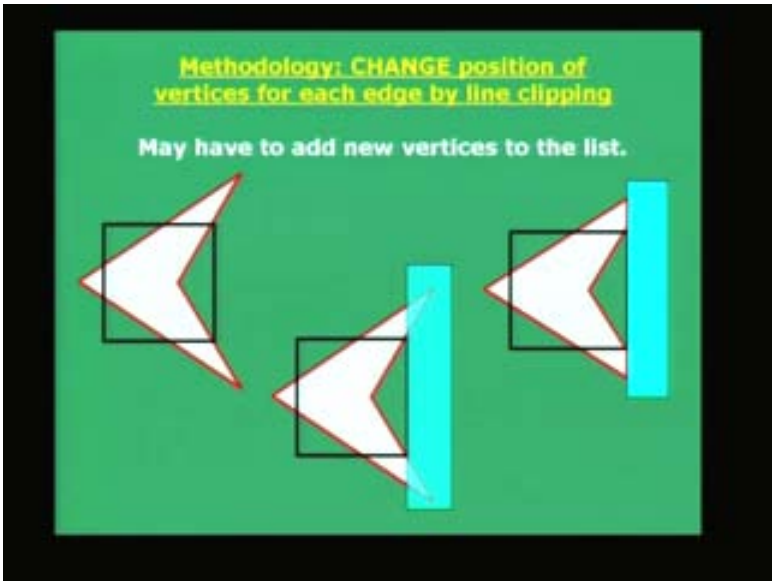
So I must have a very judiciously nice algorithm which can handle either one concrete shape coming out of convex or concave or I can have multiple components two or more which could be clipped out from one complete concave shape only. It will probably not happen in the case of convex but only concave shape where you can have multiple components. I hope this example is very clear. I repeat again, this is the case for a convex shape clipping and this is another example when I have for concave shape I can have one or even multiple components. Let us move ahead to know how do I clip a polygon.

The methodology of clipping, we will illustrate only with figures, you can write the steps of algorithm yourself as we keep moving along. The methodology is, change position of the vertices for each edge by line clipping. Remember a polygon consists of set of n vertices or n edges. So, in effect clipping a polygon with respect to a rectangle edge, the edge of the rectangle which is now used for clipping, it will clip the edge of the polygon which is a line and we know how to clip a line by any of the three algorithms which we have studied so far.

So basically we are adjusting the vertices of the polygon by clipping. That is what the essential methodology is and we will clip with respect to the edges. There are four edges of the rectangle which are used for clipping. So we will use one edge, then the second and third and fourth like that in some order we will go. The order is in fact immaterialize as we will see here. As we see here the concept of the methodology is we need to change

the position of the vertices for each edge of the polygon. Remember each edge of the polygon by the method of line clipping. So you may have to add new vertices to the list. What is the list? Initially this list consists of a set of vertices of the polygon. In this case you would take this polygon marked by the red boundary there are 1, 2, 3 and 4. I repeat there are 1, 2, 3 and 4 vertices in the list for the polygon and let us see how many remain after clipping. How many vertices and edges remain after clipping? Right now the polygon has four edges and four vertices and we will find out at the end how many vertices and edges remain. It would be less, it would be more, let us find out so we may have to add on. So what we do first? Let us try to clip this polygon with respect to the right vertical edge.

(Refer Slide Time: 29: 22 min)



So what it basically means is I am putting this shaded bounding box which will say that I will retain the part of the polygon which is in the left side of the right vertical edge and I will suppress all the amount of the line or the vertices which is behind the shaded area. So I am trying to explain this with one vertical edge which is the right vertical edge and the result of this operation will be something like this. So as you see here, the vertex have been changed in fact I have already added two vertices, number of vertices 1, 2, 3, 4, 5, 6 so totally 6 vertices. There were 4 and 2 vertices which I have already added.

If you look back into the figure once again, if I roll back this is what I am attempting to do. I am trying to suppress the information to the right side of the right vertical edge. That means I put a light blue shaded bounding box and all lines and vertices which are below are under the shaded area.

I will show them out, that means I am basically clipping those lines, how many lines? There are 1, 2, 3 and 4, 4 lines to be clipped and that will result in four new points or four few new points after clipping four lines and those are the four new vertices. And I clip those four lines I will find out which of these vertices basically pass to that particular right hand side from inside to the outside or outside to the inside of the half-planes and then when I suppress that information I will be left with this part of the figure.

So I retain that particular shade and say this is the new polygon which I have clipped only with respect to the right vertical edge. Only one of the edges have been used so far but I keep repeating this process for all the other three edges. What are the other three edges? Top and bottom horizontal edges and of course the left vertical edges, left only? What have we done so far? In this slide as you can see is clip this polygon here with respect to the right vertical edge and this is the resultant figure which I got. The clip rectangle marked by the black boundary then the red boundary with the whitish gray shade talking about the rectangle after clipping and of course the blue rectangle talks about the points on the right hand side of the right edge which have been clipped and chopped off.

Now we clip with respect to the bottom edge and you can see in the figure here, I repeat again, this was the figure on the right hand side which was left after the clipping with respect to the right vertical edge. Now when I clip with respect to the bottom horizontal edge I will add still one more vertex. I repeat, I delete one vertex and add one more vertex so the number vertices are now remaining same, 1, 2, 3, 4, 5, 6. There are 6 vertices but the new shape of the polygon as you can see is given by this. Now two more edges are left, the left vertical and the top horizontal.

I apply the same philosophy, the same logic, the same methodology and the same algorithm and now this is the polygon which we will be left over after if I had chopped with respect to the left vertical edge. So let us count the vertices now, I have two new vertices added one vertices thrown out from the list so I have 1, 2, 3, 4, 5, 6 and 7. So, 7 vertices, out of them from the old list only one vertex at the centre remains, all the vertices which were outside the clipping rectangle have been thrown out. And all these vertices are new, 1 is old and remaining 6 are new vertices. Therefore, three edges have been used for the clipping. Only one more edge left, this is the one. If I suppress the information outside or on the top of the top horizontal edge, now you have to basically add one or two more vertices. So how many vertices are you left with now? There are 1, 2, 3, 4, 5, 6 and 7.

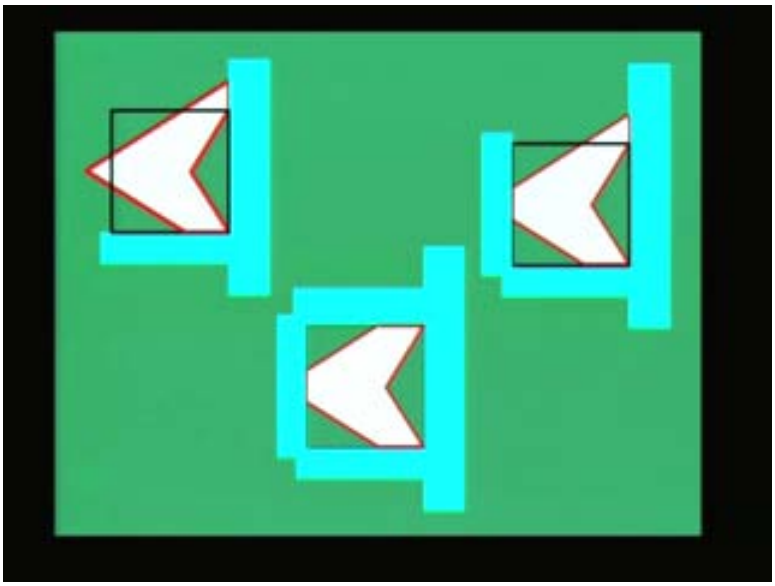
So you have 6 basically 6 new vertices, 1 old vertex in the old list. So I said before, this is the example which shows the methodology that essentially polygon clipping is line clipping. You have to find out exactly how many polygon edges you have to clip, take an edge of the clipping rectangle and pass through all the edges of the polygon find out which one you have to clip respectively and you have to add a new vertex. When you change from one vertex to other you need to of course add a new vertex and throw out an

old vertex if the vertex is outside the half-plane then you throw out and the only add vertices on the boundary and that is interesting. You only have to add vertices which are on the boundary of the clip rectangle, all vertices which are inside the clip rectangle are retained, vertices outside the clip rectangle are thrown out and vertices which are added are on the boundary of the clip rectangle.

Three parts old vertices in the old vertex list there are two parts; one outside which are thrown out, inside which are retained and you add basically vertices on the boundary of the clip rectangle. So let me again roll back this animation for you. This was the original scenario where we had a red bounding polygon with 4 vertices or 4 edges to be clipped with this black bounding box or black window or black rectangle and I first clip with respect to the right vertical edge and this is what remains. Then I clip with the bottom horizontal edge that is what remains after adding two new vertices with respect to the right edge, two vertices with respect to the bottom horizontal edge, again two vertices with respect to the left vertical edge. But in edge case I am throwing out one vertex so far and here again at the end of course again I throw a new vertex out and add a new vertex at the boundary of the clip rectangle.

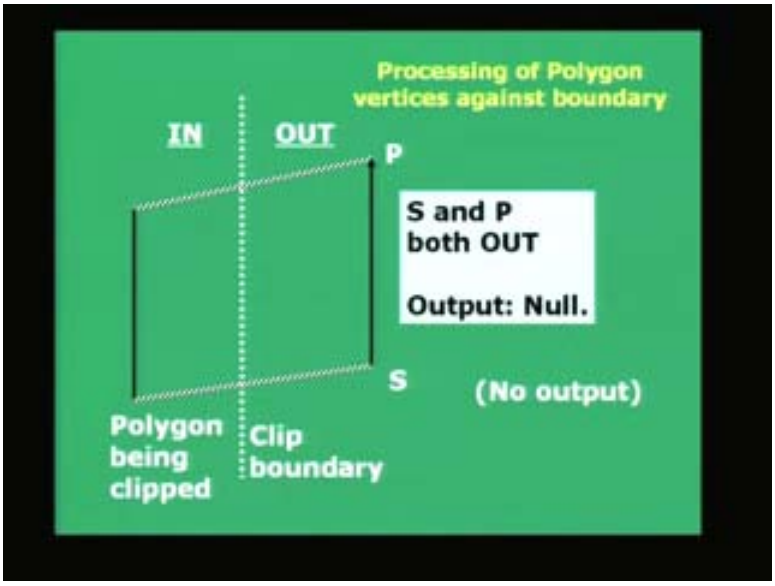
I hope the idea is clear that you have to throw out vertices which are outside in some cases. Not only old vertices are thrown out, new vertices may be added sometimes. Initially it may not be on the boundary so that also many have been thrown out. But at the end you will be only retaining vertices which are completely inside the clip rectangle and vertices which are exactly on the boundary of the clipping rectangle.

(Refer Slide Time: 34: 42 min)



So may be you just need a methodology to know when to retain and when to add a vertex and how the order of the vertices should be changed because a polygon will have the set of vertices P_1 P_2 P_3 P_4 up to P_n and if that is the list of vertices which you have when you are clipping the question comes how do you adjust the vertex and how do you throw it out. We will look in to the algorithm. Now we look into the last part of the process.

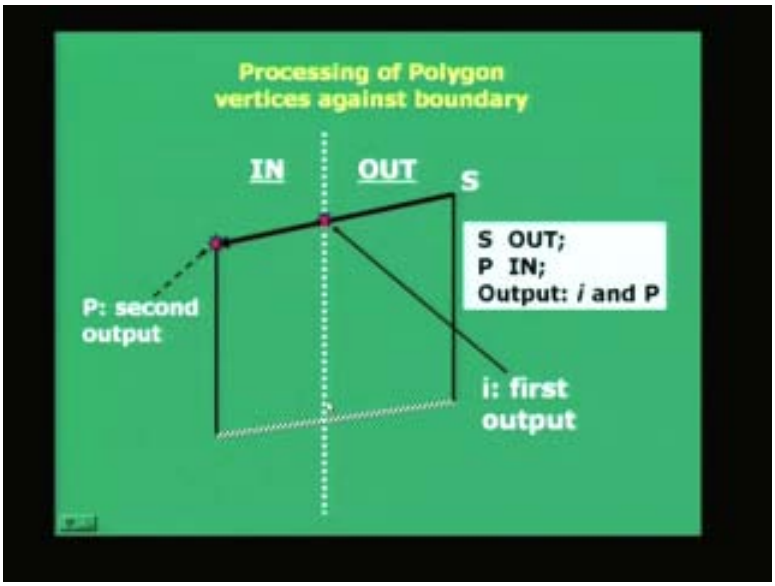
(Refer Slide Time: 36:51)



It is processing of polygon vertices against boundary. Now what this basically means is the following. I must explain this figure. Well the clip boundary as retained here is this dashed white line. Vertical line is the clip boundary, it could be horizontal or vertical but I have taken a vertical line. And I say my clipping rectangle is on the left side of that clipping boundary. That means this clipping boundary is my right vertical edge. If you are looking into the screen this clipping boundary is my right vertical edge of the clip rectangle and hence the inside half-plane is inside which is on the left side of the clip boundary, the outside half plane is on the right of the clip boundary. I hope this concept of half-plane is clear. We discussed about that many times.

Let us say this is a polygon being clipped which is also having four vertices only and we are now interested to know what happens when there are two vertices p and S, I am considering an edge which runs from S to P. So I am considering only two vertices which form an edge of a polygon. Since I said we have to move from one edge to another and the edge consists of two vertices of a polygon. So we will take one edge which consists of two vertices and see different conditions under which we have to adjust the vertices. So in this case this edge S P has two vertices S and P and we say the first condition both S and p are outside are in the outside half-plane or in this case on the right side of the right vertical clipping boundary of the rectangle which is clipping the polygon.

(Refer Slide Time: 40:22)

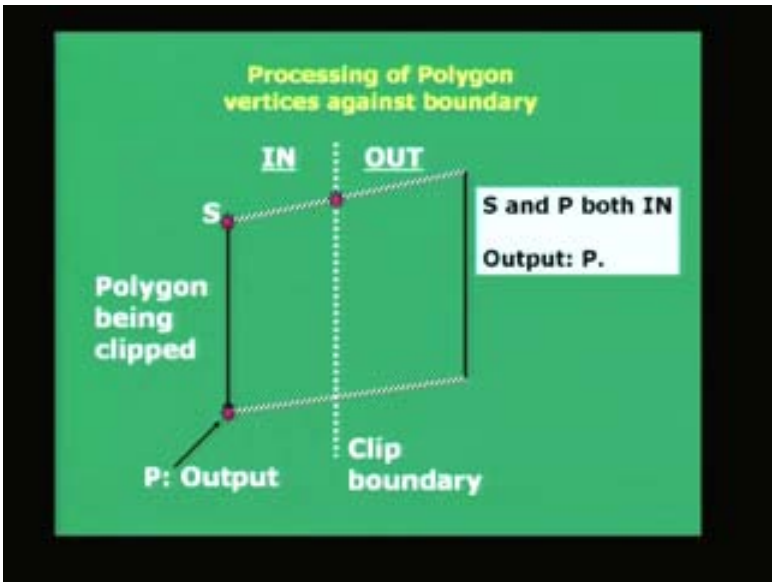


So the polygon being clipped, the left vertical edge is inside but the right vertical edge, there is a vertical edge PS which is completely outside what you do? When you see both S and p are outside just delete them from the list, do not produce any output that is what it says. So you just do not produce an output because it is something like creating a new list after clipping with respect to the boundary because you go through the list of edges and the question is which one do you produce in the output list to create a new output list. That is what we are trying to do.

I repeat again, an old list of vertices of a polygon and with respect to one clipping boundary you are creating a new list which will have some of the old vertices and if necessary a set of new vertices which are lying on the clipping boundary. That is the methodology we are discussing, there will be a four different cases which will arise of a depending upon an edge we are considering with respect to the clipping edge of the rectangle and in this case we are talking of an edge PS.

The edge SP or PS is an edge of the polygon which is marked by this dark black line with an arrow and the clipped boundary is the dashed white vertical line. I say that this is a right vertical line that means my left half plane is inside, right half-plane is on the outside of the clip boundary. And in case when both S and P, remember this S note down if both S and P are out do not produce an output, output is not produced. Let us look at a different condition now.

(Refer Slide Time: 41:09)



I remember, we were talking about processing of polygon vertices with respect to the boundary of the clip rectangle. Now, we talk of an edge of a polygon where S is on the outside half-plane and P is inside. Remember, we are talking of the same boundary which is the right vertical boundary, you are inside the half-plane is in the left hand side and the outside half-plane on the outside but the same concept holds good whatever we were discussing for all the four edges of the clip rectangle.

We apply the same philosophy for all the four edges whatever be the edge you are clipping. And the edge of the polygon you are clipping against an edge of this rectangle, this same philosophy holds good. We have seen in the previous case when both S and P are out we do not provide an output.

In this particular case of processing of polygon vertices against a boundary if you say that if S is on the outside half-plane and P is in inside the half-plane you produce an output where you get it clipped, you clip that SP with respect to IP. That means this I is the intersection of the vertical edge and edge SP. So the output will be I and then P. So the list contains S and then P of the existing polygon.

The output will be produced as I and P, S will not go to the output in this case. I hope it is clear in this case when S is out P is in. I repeat again, output is I and P, what is I? That is the clip point, that is the intersection between S P and the clip vertical line and you already have three formulations of line clipping where we know exactly how to get this point I. That is very easy, you can use any one of the methods or formulations of Liang-Barsky or Cyrus Beck or Cohen-Sutherlands parametric form to obtain the intersection point.

Next is S and P both are inside the half plane. That is the case that is the polygon being clipped, S and P are the vertices which are completely inside. You produce only the P output, this is very interesting, you do not produce S. We can almost guess why it is so because this S has been produced as an output in the previous condition which produced I and then it produced S. So do not produce an S anymore. The order must be preserved in that sequence. In the list of vertices of polygon no polygon vertices repeated twice in general in that sequence. So you do not produce S here, you just produce the output P in this case.

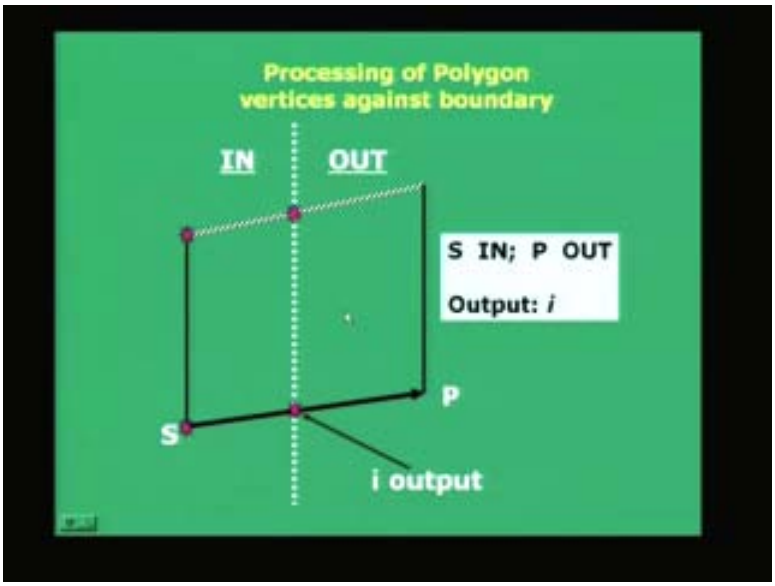
We are talking of both S and P. So we have already considered three cases; first case was S and P out, second was P out S in, S in P out and then last case is S and P both inside. This is still a fourth case left, fourth case left is this when S is in and P is out the output produced is the intersection point only. That is the I, this is very interesting. I roll back all the four cases one after another in the fourth. This is the first case when both S and P are out, do not produce any output.

If S is out P is in produce two outputs in this order, the order is very important in this case. Output produces I intersection that is the first output and second output is P in that order you produce. When both S and P are in only produce the output P and in the last case when S is in P is out produce one output. Not like the previous case when one was in the other was out you produce two output when the line was getting from out to in. In this case the line is goes from in to out you produce one output which is the intersection point only. As you can see with these four cases the polygon has been clipped.

If you see the animation with the four slides not only animation step by step, this was the original polygon, I produce two outputs see here I and P then comes the other point the low left P and then finally this produces out. These four star points are the points of the polygon which we will be retained the other two vertices will be thrown out. So with the help of this demonstration with these four different cases of edges line with respect to completely in completely out or traversing from inside to outside or from outside to inside we have four different outputs of vertices been produced. And in this case you have already seen how a simple polygon can be clipped with respect to one such edge, in this case it is a vertical edge. The same philosophy applies for all other edges that is the other left vertical edge or also the top horizontal or the bottom horizontal.

I hope the concept is getting clear because this if you have followed you have seen that the polygon has been clipped and we get the two old vertices in the list which are inside the polygon in inside the half-plane and we throw out the two vertices which are on the outside half-plane and we have created two new vertices which are on the boundary. If you look back into the figure what was the original polygon? This was the original polygon, four vertices. And after all the clipping in the different four stages we have thrown out the two right hand side vertices which are in the outside half-plane.

(Refer Slide Time: 44:06 min)



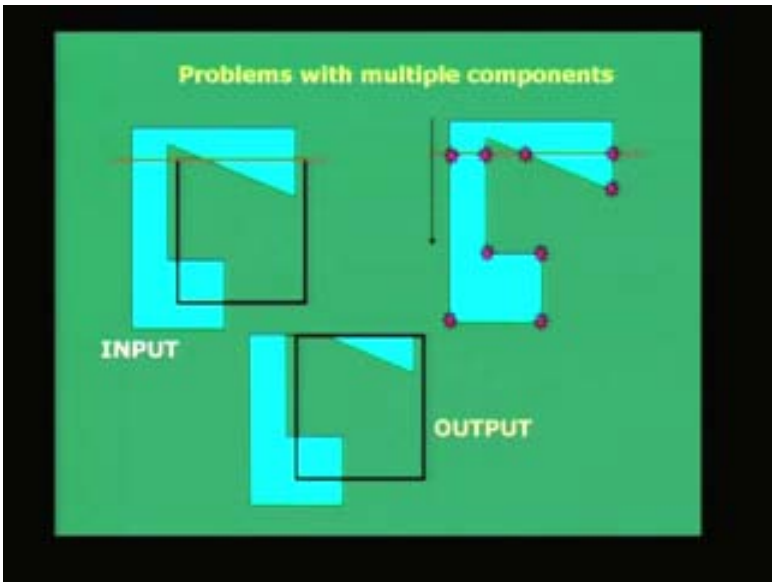
We have retained the two vertices on the left hand side half-plane and we have created two new vertices which are on the clipping boundary the dashed vertical edge. So this four star marked points are the resultant output of the polygon after clipping. I hope this concept is clear of how to clip the polygon edges with respect to one of the edges of the clipping rectangle. Now we have two edges, edges versus edge clip.

We have edges of the polygon and edges of the clip rectangle also or the clipping window and we know how to handle the edges depending upon the condition completely in completely out, inside to outside, outside to inside, how to produce the list in terms of output and creating a new list of vertices basically throw out vertices and keep on adding vertices. That is the concept basically and the basic philosophy is again line intersection line clipping.

Line intersection line clipping will produce as a mid point outputs I on the boundary which are the very basis of polygon clipping which uses the line clipping algorithm. Now there are problems with only this part which are called the multiple components, which are on the multiple components. If you see this particular concave polygon on the figure then you will see that you will land up with two different components of polygon. How to handle such a situation, let us solve this problem in the remaining time and that is the only problem which remains otherwise the methodology is the same.

Use the same methodology to clip the lines. Lines means the edges of a polygon with respect to the four two horizontal and two vertical edges of the clip rectangle and let us start with the top horizontal edge which is labeled by this brownish line.

(Refer Slide Time: 50:03)



Remember, the rectangle is in black and the polygon is in blue shade and the top edge which I am considering is labeled by another line which is about grayish in color. This is the input, this will be the output which you will have after clipping. We consider only that part the top one. If you see here that is what we want to do. We should retain the parts of the polygon which are only below that line and throw out the parts of the polygons or the edges of the polygon which are on the top part of the line. So the resultant part, so when we go inside, if we apply that S and P logic as you are going from outside to inside you remember, you traverse along the boundary of the polygon clock wise, that is what we do. In this case we will be traveling anti clockwise but you can traverse clockwise and anti clockwise you can follow the same logic.

Let us traverse the edges of the polygon in a counter clockwise manner for you and as you are seeing here this is the direction of an arbitrary polygon which will be moving counter clockwise and we will do that for this polygon, apply those logic of S and P both in and out sort of a algorithm which we have studied so far and see what happens. So as we enter from the outside to the inside the output will be two. One will be that the outside vertex will be thrown out and will be producing a new list which will have one vertex on the line at the intersection point another which is at the other end points.

Now the next one, as you go horizontal from this vertex to this both points are inside so we produce the output here. The same thing as we go along, in this case next two edges which we have considered are all inside so you just produce the final output, the same thing here as well. Now when you consider this edge we are going from inside to the outside. When you go from inside to outside the algorithm should produce a point which is just the intersection point. Remember, I repeat again, when you are traveling from

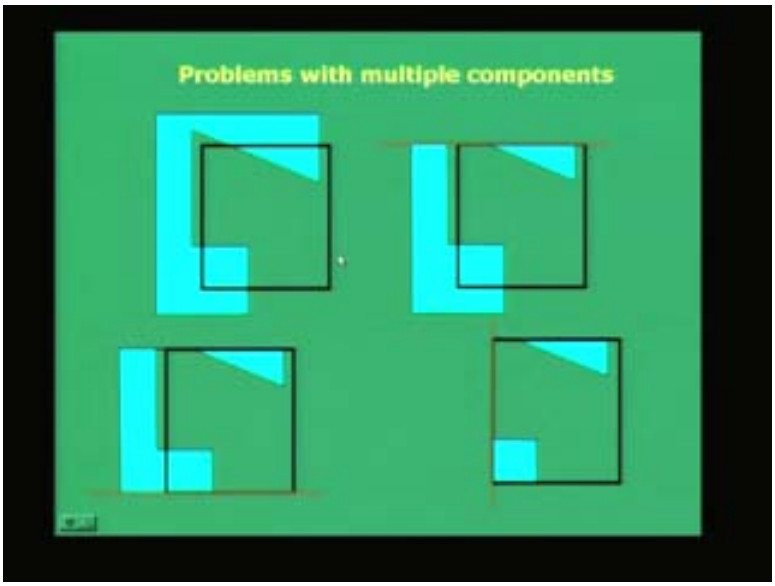
inside to outside there were two outputs, one intersection point another point inside. When of course two vertices are completely outside you throw it out, do not produce any output. When you are going from outside to inside two outputs completely inside the same output goes to the output list and then when you are going out from inside to outside you produce just one intersection point. That is what you have done here when you are traversing from this vertex to the outside here in the middle you are traveling from inside half-plane to the outside half-plane and you produce the intersection point here.

The next vertex will have two of these because it is traveling from outside to inside again from inside to outside you produce the intersection point and that completes the whole and the last one will have the two vertices which are outside so you completely throw it out. So these are the new set of vertices which will be produced by this polygon clipping. How many you have? Remember, if you look into the polygon you had 1, 2, 3, 4, 5, 6, 7, 8 you had 8 vertices. Out of them 1, 2, 3, and 4 were inside and 3 were outside.

I repeat again, I am sorry 1, 2, 3, 4, 5, 6, 7 and 8, I am sorry there are 8 and in them you have 5 inside and 3 outside and you retain all those 5 which were completely inside you throw out the vertices which are outside and produce in fact you produce 4 new vertices. You have produced 4 new vertices. You have retained the 5 old vertices which were inside the half-plane and in fact now you have a polygon with 9 vertices. This is what will be the product output produced by this algorithm 9 vertices which will enclose this.

Now if you have a very sharp philosophy to notice a particular point here, if you look back in to the figure in the output produced here, the output given by the vertices, you join the vertices and try to create the output polygon, you will probably notice a funny phenomena here which I will not talk about but let me completely clip this polygon with respect to all the other edges and show you the output. But please notice that there is funny phenomena, if you have observed it please keep it in mind and we will discuss that but this is a polygon which we will have by clipping with respect to the top vertical edge.

(Refer Slide Time: 51:11)

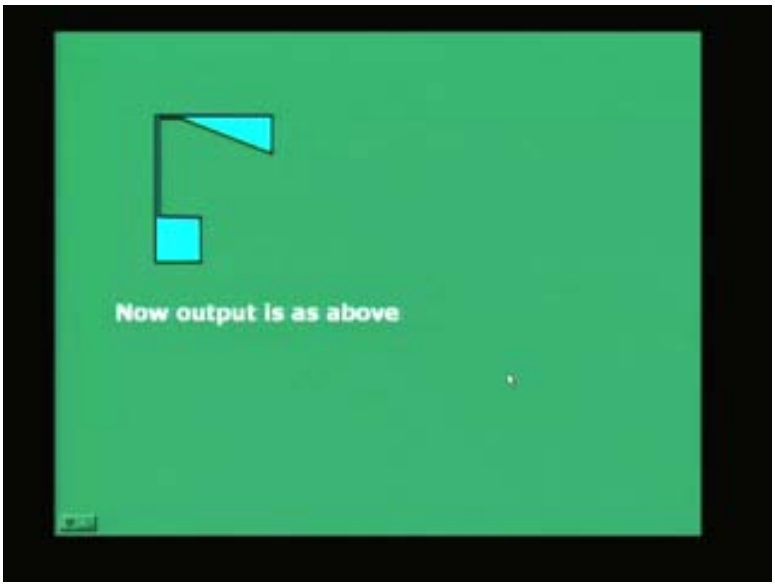


This is what will happen after clipping with respect to top vertical edge. Ten you clip with respect to bottom vertical edge, you throughout the two half vertices outside, these two vertices are outside the bottom vertex. So you completely throw them out you come up with two new vertices again. Then you clip with respect to left vertical edge and you will be left with only this part of the two independent components two multiple components of the polygon.

Now let me take out the rounding box the rectangle and show what is left. Of course I did not talk about the right hand side vertical edge. You can use that but since there is no vertex on the right hand side you don't want to clip any. That is very simple. Now this is the output which you will have. If you look into the figure I will roll it back a little bit. This was the last figure which you have. If I take out the box the rectangle and the vertical edge you will be left with the figure here. This is the output is as above but if you want an output which should be like this, this is your desired output but this is what you have, why do you have this?

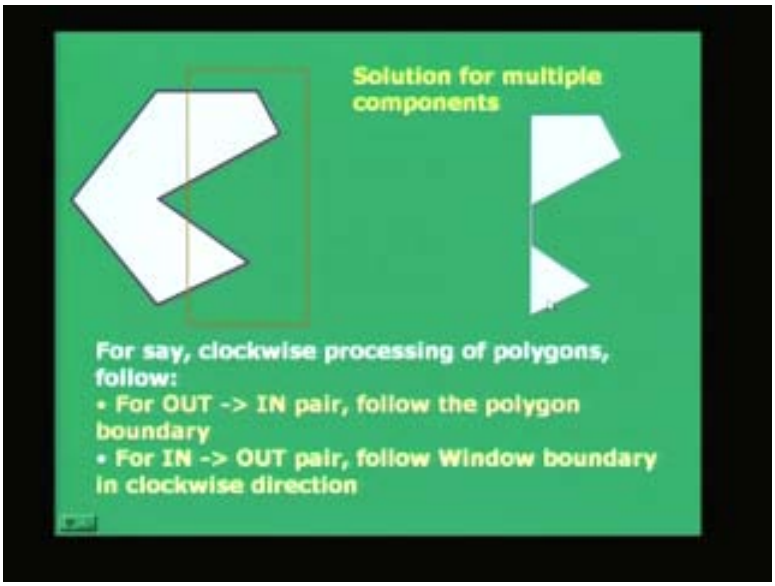
This was the funny phenomena which I discussed about when I was talking here after finishing the operation of clipping with respect to the top horizontal edge. If you try to form a polygon by joining the vertices marked by this pink colored star signs you will find that you need to join the top right and the top left vertices by a line. Hence you will be left with one line at the top which will be created and it will reside here. It will not be thrown out and it keeps remaining, it is there all throughout.

(Refer Slide Time: 52:17 min)



It is there in the top of this figure on the left hand side here. It is also here and when I throw out the boundary this is what you will get. This link between these two multi component region which is basically just one pixel wide. It is a small vertical edge and a small horizontal edge and that is what you need to clip. This is the output, you need to modify the algorithm. Quickly let us go through it. This is the solution for multiple components. For say, you need clockwise processing of polygons you follow this rule. Let us follow this rule because if you clip without this rule you will get this output you will not get multiple components. To get multiple components for out to in pair follow the polygon boundary. When you out to in pair means outside point vertex to an inside point vertex. When you are going from one vertex to another along an edge of the polygon and traveling from outside half-plane to inside half-plane follow the polygon boundary and when you are traveling from inside half-plane to the outside follow the window boundary in the clockwise.

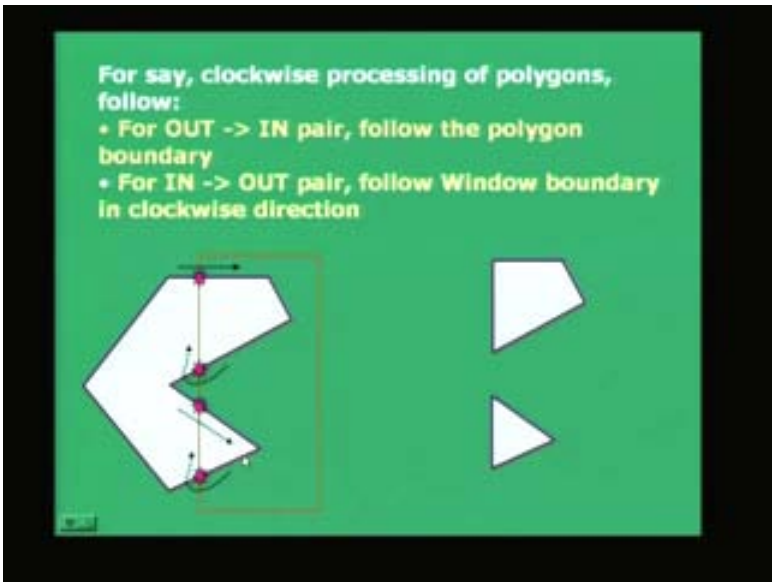
(Refer Slide Time: 52:34)



Now if you following anti clockwise follow anti clockwise. In this case I have followed clockwise in this example. Let us see what it means. I repeat these two steps once again. Read it for say clockwise processing of polygons. If you are going from out to in follow the polygon boundary, for in to out follow window boundary in clockwise direction. What does this mean? When you are going here that is from out to in outside half-plane to inside follow the polygon boundary and so you keep these two vertices. But you are when you are traveling from inside to outside please do not follow the polygon boundary anymore. You should follow the window boundary and then link up these two that this is how you truncate this component and join these two vertices and create a segment here.

I have not completed the entire polygon. I hope you have followed the logic and I am going from inside to outside first. Of course I have just skipped over a few steps here but when I am moving from outside to inside I marked this vertex. I have marked other vertices also. But finally when I am coming out I do not follow the polygon boundary. I follow the window boundary and close these two vertices and create a separate polygon. The same thing applies here.

(Refer Slide Time: 55:01)



Mark this; follow in this case a polygon boundary. But you are now following this, do not follow the polygon boundary follow the window boundary clockwise direction and link these stages. This will help you to create, since these two vertices will now be joined with this clockwise movement along the window boundary rather than the polygon boundary that will help. Initially what was happening was, this vertex was connected to this vertex and then this vertex was connected back again causing a problem for you to have a connected segment. And these two multi components were happening as one segment of a polygon but you want this output. And now this will happen because you will link this one vertex to the other and this vertex to here creating this two set. That winds up the discussion on clipping lines and clipping polygons.

Mainly we discussed three clippings of line clipping algorithms; one is the Cyrus Beck formulation, Cohen-Sutherlands algorithms based on region outcodes and Liang-Barsky the simplest formulation based on four different edges. Then we discussed about the polygon boundaries and we have seen essentially the task of polygon clipping based on line clipping. You basically add vertices, keep certain vertices or throw out certain vertices.

Decision again will be taken based on that into the out movement based on the polygon edge when you are moving from one vertex to another. But introduce a vertex, you are always introducing the vertex at the boundary of the clip rectangle and for that you have to use the formulation of the parametric representation of a line using the line clipping algorithm. That is the basis based on that which we clip lines. Of course one has to take extra precaution about multiple components so that you have isolated multiple components rather than again being linked by a boundary. I hope you have understood

the problem please try to solve this by yourself using these figures. Follow it clockwise, follow it counter clockwise and you take arbitrary shaped polygons and see, apply this logic of in and out strategy of generating a new list of vertices and then multiple components, follow the polygon boundary, follow the window boundary and generate multiple components.

Thank you very much.