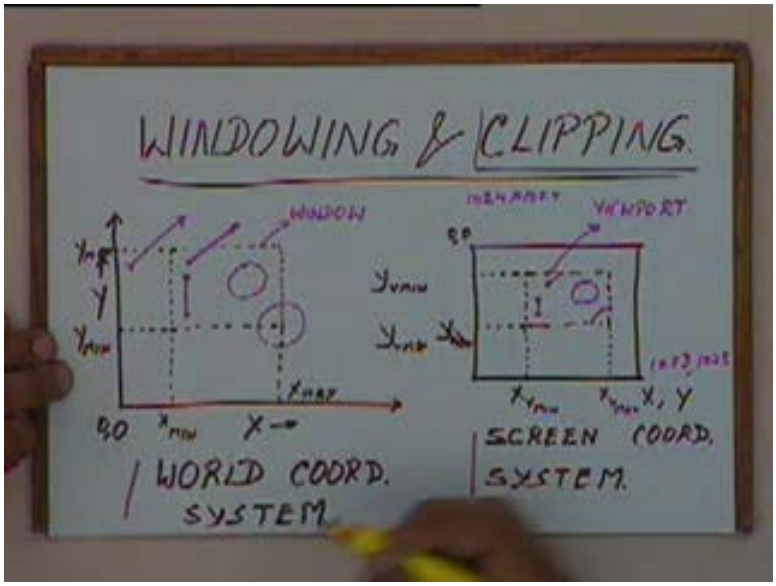
Computer Aided Design
Prof. Dr. Anoop Chawla
Department of Mechanical Engineering
Indian Institute of Technology, Delhi
Lecture No. # 06
Windowing and Clipping

Today we will talk about windowing and clipping.

(Refer Slide Time: 00:01:18 min)



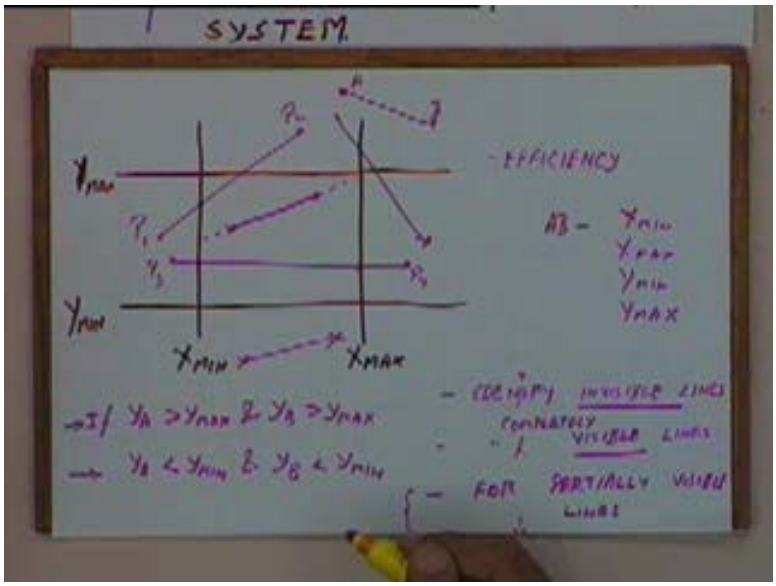
If you are modeling any object and the coordinates can start from minus infinity to plus infinity. In this infinite range in these coordinates which are normally referred to as the world coordinate system. Out of this infinite range we will normally take a finite portion of this range. Let's say starting from this value which we will call as X_{min} and going up to X_{max} and in the y direction from Y_{min} to Y_{max}. If we take this finite portion of this infinite world, we will normally like to map this finite portion on to the screen coordinates. So if this is a size of a screen let's say this being a 0 0 and this being the maximum in terms of x and y. We will normally like this window to be mapped down to a finite portion of this screen maybe we will take the full screen or maybe we can even take a portion of the screen. What we would like to do is this portion of the world, we will to map on to this portion of the screen, so this is referred to as the screen coordinate system.

In this screen coordinate system the x and the y values will again range from let's say if you call this as the x viewing minimum and this is the x viewing maximum. Similarly this is the y viewing minimum sorry this will be the y viewing maximum and this will be the y viewing minimum. Now this finite portion is referred to as the window, this is the window out of the infinite world. A window is thus a finite portion of the infinite world. This portion is referred to as the viewport; this is a part of the complete screen on which this window is to be displayed.

So if we have any objects inside this window, you will like this object to be displayed here like this. Similarly if we have a line here, we like that line to be displayed here. When this line is displayed here, its size as well its coordinates they will be very different from the size and the coordinates over here. These coordinates are what are termed as the world coordinates system, these coordinates will be in the screen coordinate system. Typically let's say if you have a screen has a resolution of 1 0 2 4 plus 1 0 2 4 and these coordinates will be in this range between 0 and 1023. This xy maximum would then be 1 0 2 3, 1 0 2 3. So these coordinates will be in this range but these coordinates can be in minus infinity to plus infinity and if they are within this window, they will be in the range of between Y_{min} and Y_{max} and X_{min} and X_{max}. How do we transform the coordinates of let's say the coordinates of this point on to this point? We have some transformations, we will go into the transformations later on.

Today what we will see is and let's say in this window, if we have a line which is going like this or if you have a line which is outside this window, we need to know that this line is outside the window so that we don't draw it over here. This line should not be drawn on this because this viewport is supposed to draw only those lines or those entities which are inside this window. So if you have a line like this that should not be drawn and if we have a line like this, this line should only be drawn partially like this. The rest of the line should not be drawn. So we will say that this line should be clipped along this edge that means whatever is outside this edge, outside this window that should not be drawn. Is that okay? So that is what we mean by clipping. Whatever is inside that should be drawn completely, whatever is outside the window that should not be drawn, whatever is partially inside and partially outside that should be cut along the edges. If you have a circle like this, only this part of the arc should be drawn. If you have a triangle like this, only this part of the triangle should be drawn. We do not want to draw anything outside the viewport obviously. So we will just see how this clipping can be done.

(Refer Slide Time: 00:08:10 min)



If this is the window, if these are the four boundaries of this window what we have to ensure? An edge like this should get clipped at these two intersection points and edge like this should not be

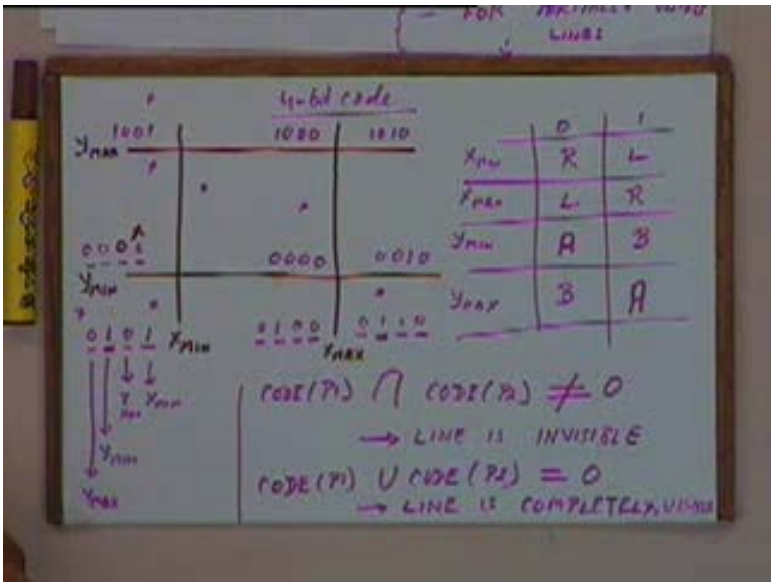
drawn at all and edge like this should be clipped here. Only a portion inside that should be displayed. And how do we do that? Let's say this is the one end point, this is the second end point. How do we clip it along the intersection point? We will need to find the intersection of this line with the line x equal to X_{min}. Similarly we will need to find the intersection of the line P_1 P_2 with the line y equal to Y_{max}. For this line let's say this is P_3 and this is P_4, for the line P_3 P_4 you will have to find out the intersection with the line x equal to X_{min} and x equal to X_{max}. We have to find all these intersections and is very important that all this to be done as efficiently as possible. Efficiency is of prime concern because this clipping is done very frequently and whenever this clipping is done typically you will have a number of entities in the world. So we like to do it as efficiently as possible so that the clipping should be done fast. Speed is of prime concern. One way of increasing the efficiency is that if you can know for sure that some lines are outside, let's say if we have a line like this, if a line is outside this window I should not try to find out the intersection of this line with each of the four boundaries.

Let's say for a line like this AB, if I try to find the intersection AB with either of these lines X_{min}, X_{max}, Y_{min} and so on. If you find any of these intersections, this exercise is going to be wasteful because this line is totally outside and wherever the intersections like, irrespective of that the line is not going to be displayed. So the first thing that we normally do is we try to see if a particular line is totally out of the window. If that line is totally outside the window that is it is either above the Y_{max} line or below the Y_{min} line or to the right of the X_{max} line or to the left of the X_{min} line, at least such lines should be moved all together. I should not try to clip them or find the intersections with the boundaries. So the first thing that we will do is identify invisible lines.

Similarly if a line is completely inside if the line is completely inside even though I need not find out the intersections because the line is inside there is no point in finding out the intersections. I have to draw the complete line anyway. So I will again try to identify completely visible lines and only for lines which are partially visible, lines which are partially inside the window and partially outside we should try to find out the intersection with the edges of the window. And then for partially visible lines we will find out the intersection and so on. For this we will have a different set of steps. How do we identify whether a line is completely visible or completely invisible? I said one way is that lines which are totally above the Y_{max} line will always be totally invisible. How do I check whether a line segment AB is totally above the Y_{max} line? I will have to compare the y coordinates of the points A and B.

So I can say if Y of A is greater than Y_{max} and Y of B is also greater than Y_{max} then the line is invisible. Similarly I can have constraints for each of the four lines. So if Y of A is less than Y_{min} and Y of B is also less than Y_{min} that means the line is somewhat like this, even then the line will be totally invisible. Instead of checking for constraints like this, checking such constraints a number of times can become slightly messy, instead of that we use what is called as the four point code.

(Refer Slide Time: 00:15:09 min)



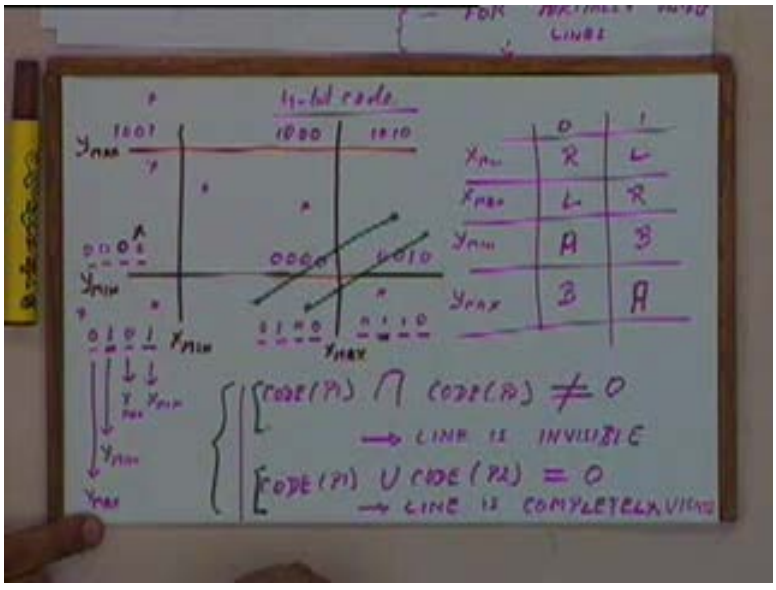
With respect to every line let's say if you take a point here, if we take a point here, now this point is to the left of the line X_{min}, it is above the line Y_{min} but below the line Y_{max}. So with respect to every edge, we will give it a one point code. So for this point if we have 1 2 3 4, since this point is to the left of the X_{min} line we will give it a code of one. If it is above the Y_{min} line, we will give it a code of zero. It is below the Y_{max} line we will give it a code of zero and it is to the left of the X_{max} line we will give it a code of zero. So if any point is to the left of the X_{max} line, mind you the visible portion is also to the left of the X_{max} line. If any point is to the left of this X_{max} line corresponding to that we give it a code of zero. If this point was to the right of X_{min} line, we will again give it a code of zero. So with respect to X_{min}, X_{max}, Y_{min} and Y_{max} with respect to each of them our code will be either 0 or 1. If you are to the left of X_{min}, we will give it a code of one. If you are to the right of X_{min}, we will give it a code of zero. If you are to the left of the X_{max} we will give it a code of zero, if you are to the right we will give it a code of one. If I take Y_{min}, if I am below Y_{min} then I will give it a code of one, if I am above I will give it a code of zero.

Similarly Y_{max}, if I am below Y_{max} I will give it a code of zero, if I am above Y_{max} I will give it a code of, if it is above Y_{max} I will it code of one. As a result if a point is inside this region, it will get a code of 0 0 0 0. If a point is in this region, the point is below Y_{min} so it gets a code of let's say this is a Y_{max}, this is a Y_{min}, X_{max} and X_{min}. So if this is a Y_{max}, if it is below Y_{max} it gets a code of 0, if it is below Y_{min} it gets a code of one. It is to the left of X_{min}, left of X_{min}; X_{min} means one and to the left of X_{max} also, so it will get zero. So this code I am using for the first bit, I am using with respect to X_{min} this I am using with respect to X_{max}, this I am using with respect to Y_{min} and this with respect to Y_{max}. If I then if the point is in this region what code will it get? the first bit with respect to X_{min} would be zero, with respect to X_{max} will also be zero, with respect to Y_{min} again we are on the wrong side so it will be one and this will be zero. If you are here with respect to X_{max} will be one, this will be zero and this should be 0 1 and this would probably be 0 0 1 0. If you come here then with respect to Y_{max}, 1 0 0 1 and this would become…

So in these 9 regions the point will get a different code. If a point is here it becomes 1 0 0 1, if it is here it gets 1 0 0 0 and so on. Now if one point is here and a second point is here, both of them will get the code of 0 0 0 1. If one point is here and the other point is here, this will get a code of 0 1 0 1, this will get a code of 1 0 0 1. Both these points are to the left of X_{min}, so the bit corresponding to X_{min} will be one for both of them. So if I take any line segment and out of the four bits, any one bit is one for both the end points that would mean the line is totally outside the window. If I take a point here and a point here, in that case I get a, with respect to Y_{min} both the points are lying below. So Y_{min}, so this third bit here and a third bit here both will be one. So if I take the code for point one and the code for point two and I take the intersection of these codes. In the intersecting of these codes has a one at any location or the intersection of these codes is not equal to zero, in that case the line segment will be totally out of the window. Is that okay? So if the code for point one intersection in the code for point two is not equal to zero that means line is invisible, completely invisible.

Similarly if I have a point here and a point here then the code for both of them has to be 0 0 0 0. So if I say that the code for point one union with the code for point two is equal to zero then the line is completely visible. If the line is inside then both the endpoints will have a code of 0 therefore the union will be equal to 0. If the line is to one side of either of the 4 edges then the corresponding bit will be one and therefore the intersection will not be equal to 0 and then the line will be totally invisible and these bits can easily be set for each point just by comparing them with the x and the y coordinates of the edges. So by using this fourth and the four bit code, this is a four bit code. By using this four bit code we can very easily check whether the line is completely invisible or whether the line is completely visible.

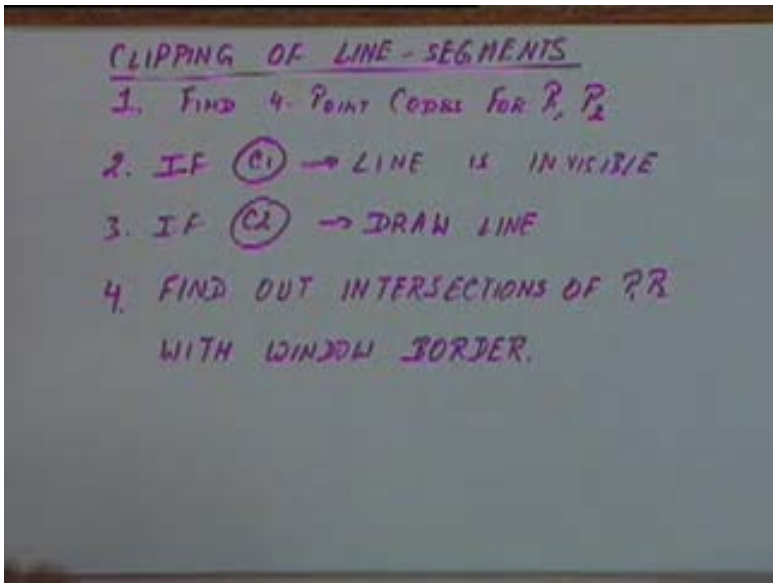
(Refer Slide Time: 26:00)



Of course there will be some cases like this. If we take this 0 1 0 0 and 0 0 1 0, their intersection will be equal to 0 but even then the line is completely invisible. In such cases these checks are not going to result in any decision, we cannot say, we cannot decide on the basis of these checks that the line is invisible. Only when the intersection is not equal to zero will the line be invisible.

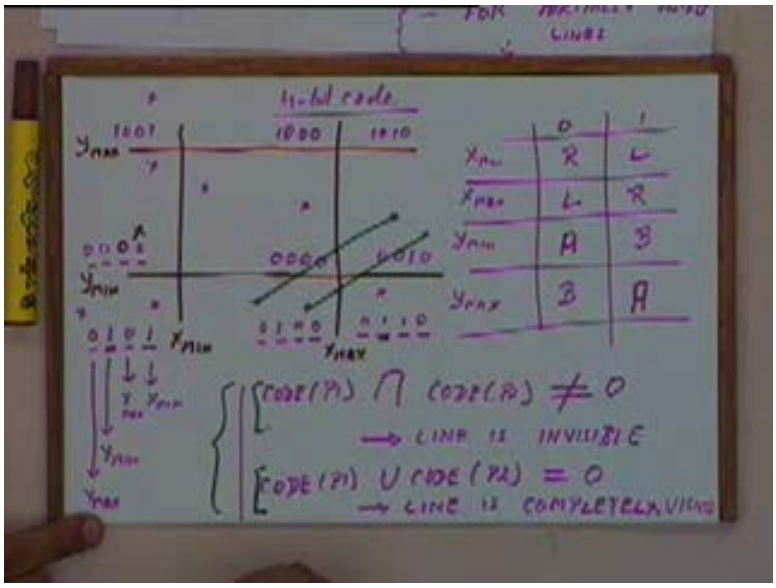
Only when the intersection is not equal to zero, we can say confidently that the line is invisible but if this condition is not satisfied and this condition is also not satisfied then the line can be invisible or the line can be partially visible. Whether the line is like this or the line is like this, we can decide that only by finding out the intersection points. So this four bit code will help us in reading out those lines which satisfy either this criteria or this criteria. Any question up to this point? Then for the other lines for lines like this, for lines like this how do we go about clipping these lines? The first is we check for these lines and then start finding out the intersections.

(Refer Slide Time: 00:27:35 min)



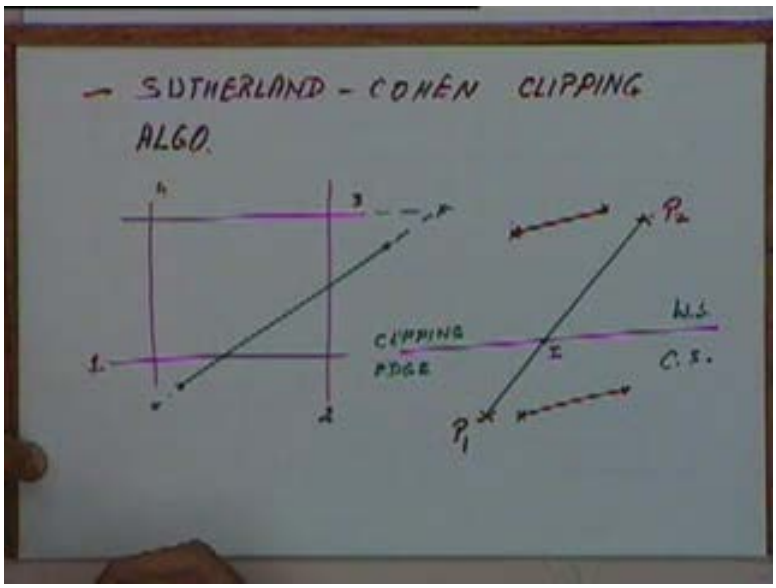
So we write it crudely. The first step is find four point codes for P_1 and P_2. We are trying to clip the line segment P_1 P_2 then find the four point codes for P_1 P_2 then we check for these conditions let's say condition one and condition two. We will check for these conditions condition one and condition two. So I will just right if condition one is satisfied then line is invisible, neglected. If condition two is satisfied then line is visible, draw it. Otherwise what will we do? We find out the intersections of P_1 P_2 with the window border and check which part is inside and which part is outside.

(Refer Slide Time: 00:29:34 min)



We will have to find out the intersection of this edge with all the four edges, all the four edges of the window we will get one intersection here, the other intersection here. We will say that these two intersections are outside this line segment, so we will ignore that. We will find out this intersection and this intersection and then decide that this line is outside, this line is inside and this line is outside and we finally display this edge. So in this fourth step we will have to find out the intersection with each of the four edges of the window and check as to which part of the line segment is inside and which part is outside. This is one clipping algorithm for line segments. Any question about this clipping algorithm? We will just modify this clipping algorithm to increase speed further, how to make it more efficient. Any question about this clipping algorithm?

(Refer Slide Time: 00:31:16 min)



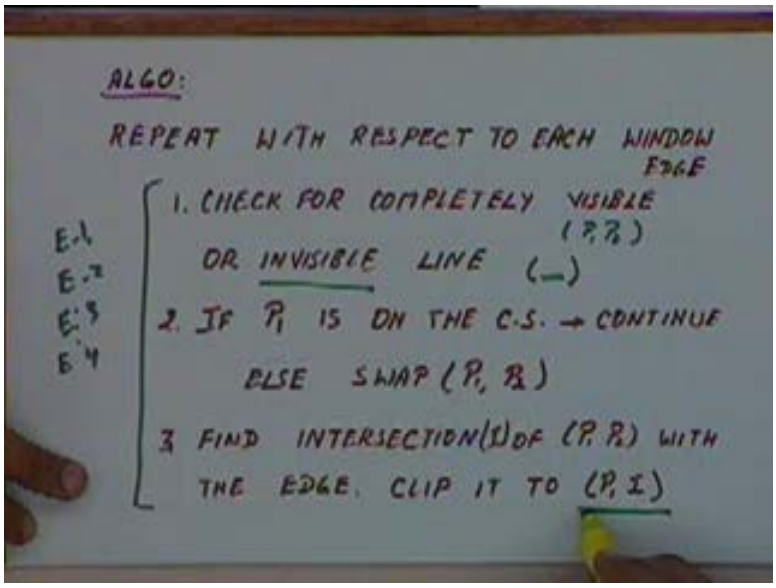
We will now see a clipping algorithm which is attributed to Sutherland and Cohen. Sutherland and Cohen are the name of two scientists who developed this algorithm. Now the problem with previous algorithms that comes is that when we have a line like this, we have to find out the intersection of this line with each of the four edges of the window. We first find out let's say this intersection, will then I also have to check whether this intersection is inside the line segment or not. Since this intersection is outside is really not relevant, only an intersection which is inside this line segment is going to be relevant.

Similarly at this point, this intersection is also outside this line segment so that is also not relevant. In order to take care of all these, what we can do is let's say this is our clipping edge and we have a line to be clipped which is a line like this. I call this as the correct side of the clipping edge, this is my clipping edge and this as the wrong side. [Conversation between Student and Professor – Not audible ((00:33:26 min))] Again please? The correct side is the visible side. The correct side is the visible side and the wrong side is the wrong side. So if I am considering this as my clipping edge, the right hand side of the visible edge or the correct side and left hand side is the wrong side.

Now if I have a line segment whose both the end points are on the correct side then no clipping has to be performed with respect to the clipping edge as straight forward. Similarly if I have a line segment which has both its end points on the wrong side even then no clipping has to be performed with respect to this edge. Only when I have one end point on one side and the other end point on the other side. I have one end point which is P_1 which is on the correct side and P_2 which is on the wrong side. Only in that case do I need to perform the clipping. So in the Sutherland Cohen clipping algorithm, we will perform clipping with respect to each edge separately.

Let's say this is my edge number 1, this is my edge number 2, this is edge my number 3 and this is my edge number 4. I will first perform clipping with respect to edge number 1 and I will check whether both the end points are on the correct side or both of them are on the wrong side. If both of them are in the correct side, no clipping has to be performed with respect to edge number one. If both of them are on the wrong side again no clipping needs to be performed, the line can be ignored. If one point is on one side, the other is on the other side, I find out the intersection and the line from the point P_1 to this intersection point I, this is the line we are intersected in. So I will just write down this algorithm.

(Refer Slide Time: 00:35:47 min)



We will repeat with respective to each window edge. The first thing is check whether the line is completely visible or completely invisible that means both the points are on the correct side or both the points are on the wrong side. If either of the two conditions is satisfied then we can take action accordingly and complete and come out of the algorithm. Otherwise what we will do is CS is for correct side. I will just explain what we are doing. What I said is if P_1 is on the correct side continue else swap P_1 and P_2. In this case P_1 is on the correct side, P_2 is on the wrong side. There is no problem but if my line segment or like this and this was P_1 and this was P_2. P_1 is on the wrong side and P_2 on the correct side. I want to exchange the points P_1 and P_2, I want to call this as P_1 and this as P_2, I want to call the point on the correct side as P_1 so that when I find out the intersection from P_1 to the intersection I, that point I will take as the clip segment.

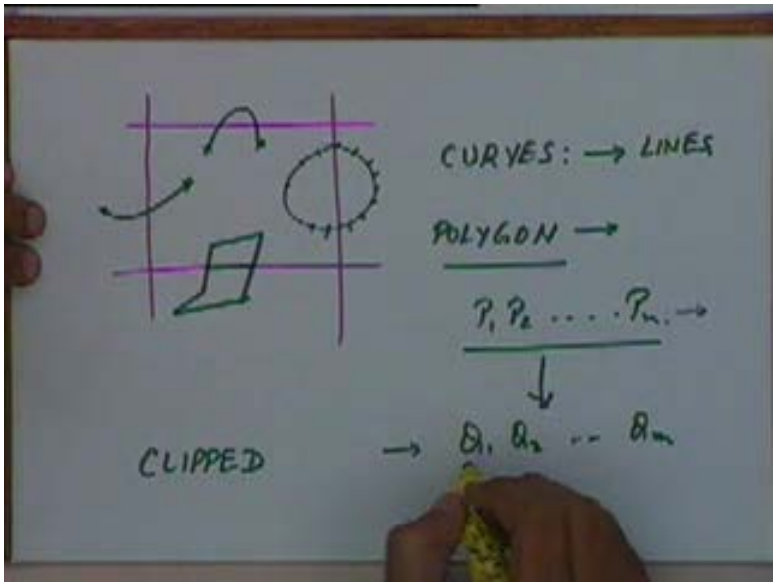
If this is P_1 then from P_2 to I, I had to take P_2 to I. I don't want to do that, I want to take from P_1 to I. So this is P_1 and this is P_2, I will swap P_1 and P_2 and still take the line segment from P_1 to I. So if P_1 is on the correct side continue else swap P_1 and P_2 and then find the intersection of P_1 P_2 with the edge, with the clipping edge and clip the line segment P_1 P_2 to P_1 I. Sir why this swapping necessary? Again please. Why this swapping on the points P_1 and P_2 necessary? If I don't swap but I have to check for is which point is on the correct side and which is on the wrong side. If P_2 is on the correct side then the clipped line will be P_2 I and not P_1 I. Is that okay?

So I will have to keep track of whether P_1 is on the right side or P_2 is on the right side. I don't want to keep extra track of that. I will just check for that and swap them and make sure that P_1 is on the right side. So after the step two I know for sure that P_1 is on the correct side and then in step three I will just find out the intersection of P_1 P_2 with the clipping edge and the line will get clipped to P_1 I and then I will take this line and repeat it and clip it with respect to the remaining three edges. So first I have taken, I will repeat this whole process with respect to edge one then with respect to edge two and so on, I mean edge 3 and edge 4. The first step is check for completely visible or invisible line. If you expand the step out, if the line is completely invisible

then the clipped line itself is P_1 P_2. Then this clipped line P_1 P_2 will be passed on to the next edge for further clipping.

If the line is completely invisible in that case the line to be passed will be on blank line, no clipping will be done after that because the line is totally invisible. So the line is completely invisible where clipping algorithm can end here. If line is completely visible then this clipped edge has to be passed on to the next clipping edge and the process will have to be repeated for each of the edges. So this algorithm is called as Sutherland Cohen algorithm, Sutherland Cohen clipping algorithm and the important thing in this algorithm is first clipping is done with respect to each edge separately and secondly we are making sure that the point P_1 is always on the correct side so that it becomes easier to say that P_1 I will always be the clipped edge, otherwise we will have to decide whether it is P_1 I or P_2 I. Any question about this Sutherland Cohen clipping algorithm? Continuing with clipping further, so far I have been talking of clipping of line segments only.

(Refer Slide Time: 00:43:51 min)

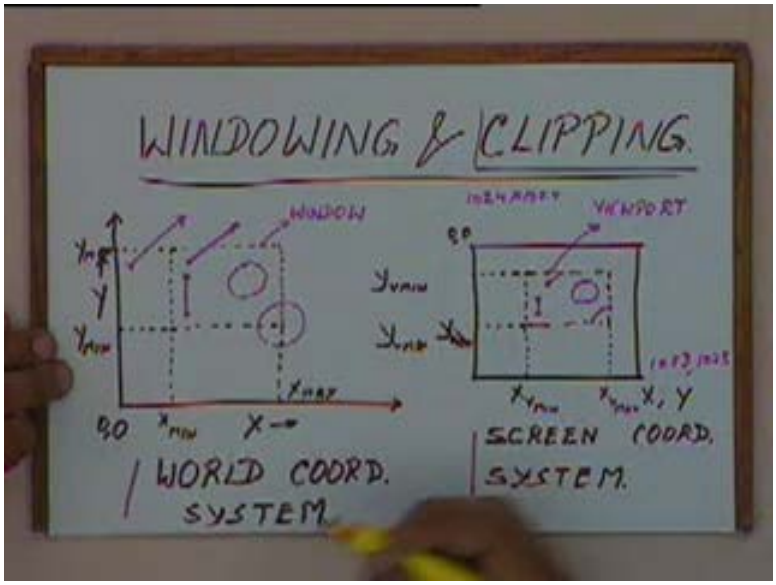


If instead of a line segment, if we have let's say this is the clipping window and if you have curves, if you have a curve like this, in a curve it has to be clipped then clearly the algorithm that you are using so far will not be valid for this curve. Even if I take a simple curve, a simple arc like this even though both the end points of the arc might be inside, a part of my curve can still be outside. Again same algorithm will not work. The second problem is that if we have regions or an area to be clipped, an area can be defined by let's say a polygon. We want this area to be clipped, we like to clip this to this portion. So if you have a polygon given as a list of points, a list of vertices, we like to be able to clip it and find out the clipped polygon. So with respect to curves, the way you look clippings of curves normally will divide this curve into sequence of small line segments. For display of curves normally approximated as the sequence of line segments and then once it is approximated as a sequence of line segments, we will do the clipping of line segments. So the curves they will be reduced to lines and if you have a closed

region bounded by a polygon, if you want to do the clipping for polygons then you will have a special algorithm for that which we will see in the next class.

How do we carry out a clipping of polygon so that we can get a clipped polygon? Polygon is we have seen earlier, they are list of vertices P_1 P_2 till P_n. We want to clip this and get a list of vertices say Q_1 Q_2 till Q_m where this is the clipped polygon and this is the origin, P_1 is original polygon. How do we do this? that we will see in the next class. For curves we will normally approximate a curve by set of lines and then do the clipping. Any questions on clipping that I have covered today. [Conversation between Student and Professor – Not audible ((46:47- 47:18 min))]

(Refer Slide Time: 00:47:28 min)



If you look at this window we say that this window is a finite portion of the infinite world. If you want to do the clipping at the time of displaying the pixel, you decide whether the pixel is inside or outside. Just imagine some curve or some entity which had a very large distance. you also have to display that and then find out the pixel which will correspond to that. The idea of clipping is that whichever portion is outside, you will not try to write down the or you will not try to run a display algorithm for that. If you try to run a display algorithm for that, you will end up doing lot of wasteful computation. This portion might be let's say only 1 % of the complete set of entities you have. If you have a let's say an entity over here, no point find out which pixels will correspond to this circle and for each of those pixels saying that those pixels are outside the window. Then for the each line segment we will have to prove that anyway. So what we are doing now? What we will do is we will divide this in the line segments. Initially it can be very close and find out which line segments are inside and which are outside.

If we find out that the complete curve is trivially outside, we will not bother about it. Then the other thing is for entities like curves I mean for entities of circles, you can easily find out the bounds of the circle. If those bounds are completely outside the clipping window, we can ignore that circle, we need not even try to display that or clip that. Our basic idea is that we will try in

clipping we will try to remove as many entities as possible without displaying them because we know the display is slow. Because in display you are going to compute the detail of each and every pixel and in fact if you remember in some of the display algorithms, we were using space which is proportional to the number of pixels. So if you are talking of an infinite space, we really can't do that. We cannot have infinite number of pixels and decide the intensity of each pixel and so on. We will have to, if we try to compute the pixel corresponding to each of these points depending on the number of entities that we have can also become very expensive.

So first thing we do is we try to see whether this curve has any portion common with the window. If it does not we ignore that, if it does only then we try to do the clipping for that and for that we will approximate it by a set of lines and then do because then we know that some portion is going to be common. Even let's say for a curve like this, if we start finding out the set of pixels that will correspond to the invisible part of the curve, we will have to find out for each of these pixels we will have find out whether the pixels are inside or outside that is going to be wasteful. Instead of that we will divide this into a sequence of line segments.

Normally what we do is we initially start with the very close division like this and only if we feel the need, we will start bringing them closer. If we feel that let's say from here to here, a part of the curve is going to be inside then we will sub divide it further and so on. We will be seeing more of this in detail in the later topics. I think at that point it will become much clearer. Any other questions? So that's all for today.