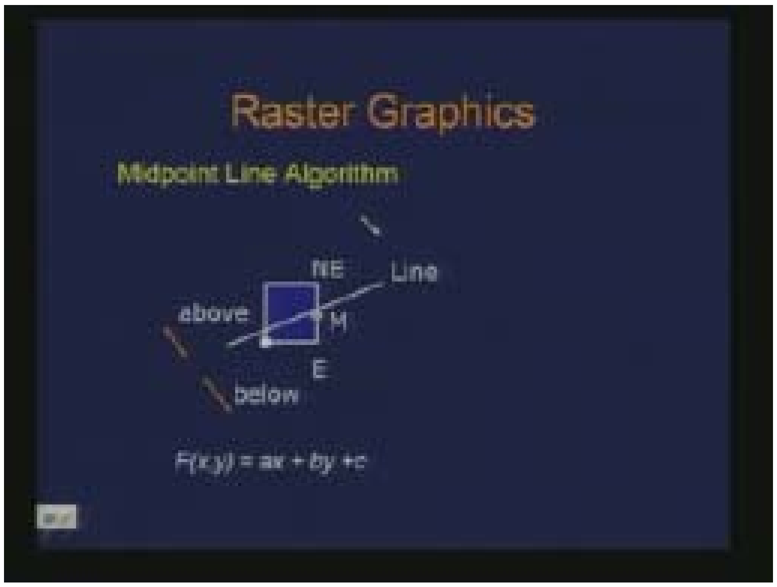
Introduction to Computer Graphics

Dr. Prem Kalra
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi

Lecture - 3
Raster Graphics (Contd….)

Last time we were talking about drawing a line using raster graphics and what we looked at basically was two algorithms one of them was DDA and the other one was midpoint line algorithm.

(Refer Slide Time: 1:18)



Raster Graphics

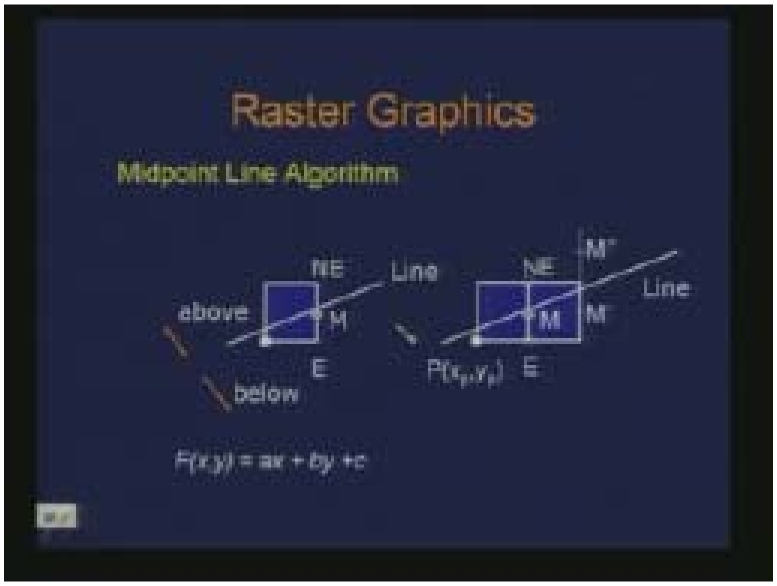
Midpoint Line Algorithm

NE
Line
above
M
E
below

$F(x,y) = ax + by + c$

So the basic idea there was we wanted to simplify the process to decide which point to plot next. and the strategy was that if I have a line given then I define a function form which is f(x, y) which is nothing but the implicit form of the line ax plus by plus c is equal to 0 when I define a line using this representation. Then the question we were trying to ask is that the midpoint between E and NE which are the candidates to be plotted once I have plotted this point how do I decide whether to plot E or NE east or the north east point. And it gets reduced to the evaluation of the midpoint M for the function f(x, y). All one needs to do is check the sign of this f(x, y) substituting for M and decide whether this is below the line or above the line. If it is greater than 0 it is below the line and if it is less than 0 it is above the line and that in turn decides whether I should plot E or NE If it is above the line that means I should plot E because E is closer to the line that is the main idea. Similarly, if it is below the line then NE is closer to the line therefore I should plot NE.

(Refer Slide Time: 3:03)



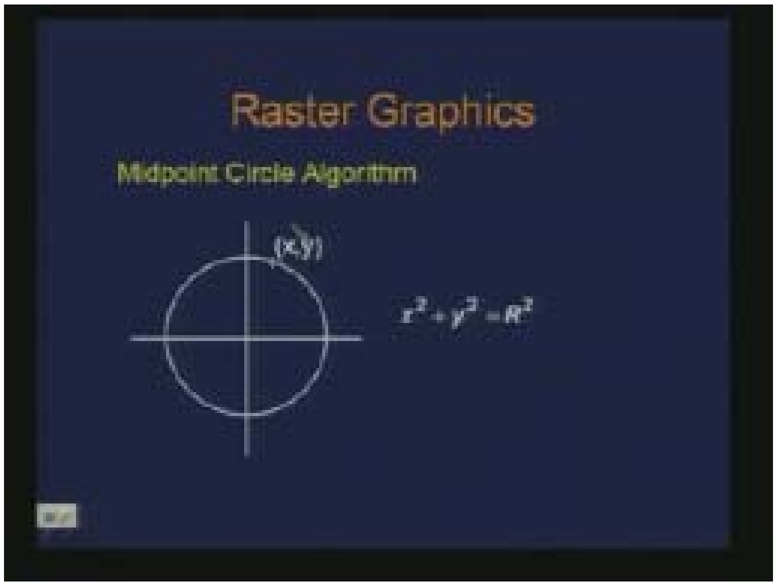
Raster Graphics

Midpoint Line Algorithm

$F(x,y) = ax + by + c$

And then we basically formed an algorithm by which this F(x, y) which is actually acting as a decision variable a decision function how do we update this decision function for the next subsequent points to be plotted? There if I had plotted E then what is the evaluation I need to consider? That is, I would evaluate F(x, y) at M prime. Similarly, if I plot NE then I would do the evaluation for M double prime. So that is the basis of the algorithm and that why we call this as midpoint line algorithm. So this was about drawing or plotting a line. Now we move on to some other primitive.

(Refer Slide Time: 4:07)



Raster Graphics

Midpoint Circle Algorithm

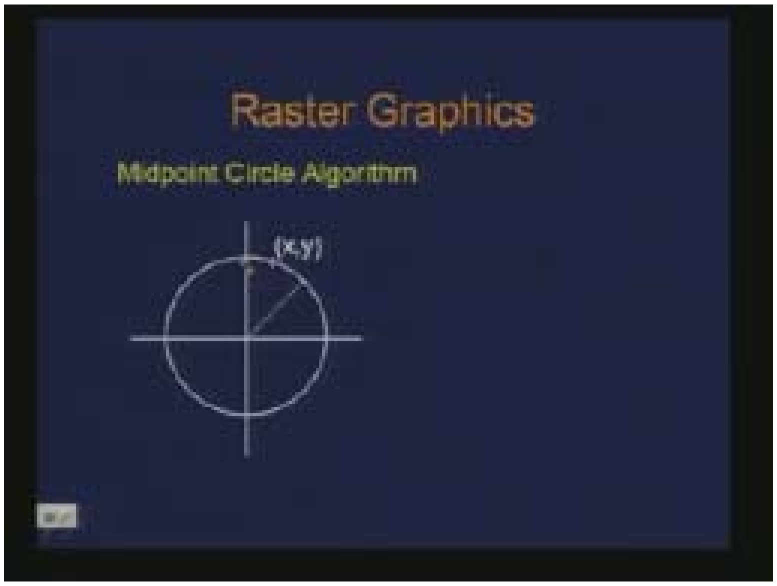
$x^2 + y^2 = R^2$

Now I am plotting this circle, the idea here is that if I am able to address the problem of plotting or drawing line in one octant that we figure out then we can actually do similar process for the other octant. So the question was; what happens if I have to plot a line in the other octant? For instance I can change the role of x and y and keep the algorithm the same and that will take care of if I have to plot a line in another octant then I can do it. So now we move on to drawing or plotting a circle. As we know that the circle is represented in the form like x square plus y square is equal to R square.

Once again one can see the problem of plotting a line as a matter of, I run through x values from 0 to r and obtain the value of y corresponding to the x value I gave from this equation I had plotted. Now what is the implication of that? so the implication of that, for instance if I am considering the first quadrant I start from here somewhere which is the point at 0 arc and these are the subsequent points which get plotted by using just a solution of this equation. Now what you observe here is the sampling which I have in y the points which I plot from this point to this point the gaps keep increasing and that may not be a good strategy then I have to plot these primitives in somewhat uniform sample. Therefore here the sampling is somewhat changing when I go from this point to this point. Now the issue is that, can we device a mechanism by which we can avoid this?

Something similar to what we did in the case of line drawing where we treated the line given the fact that we know the slope of the line by the specification of two end points can we break down this problem into octants?

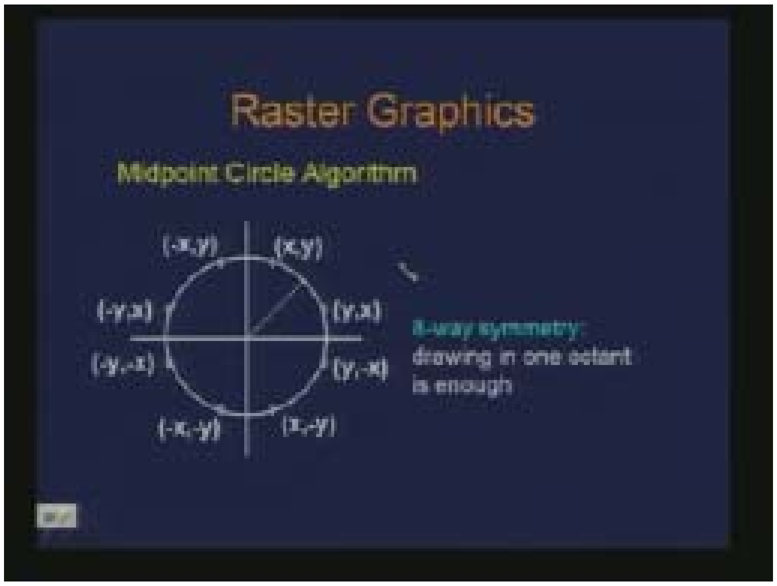
(Refer Slide Time: 7:24)


Raster Graphics

Midpoint Circle Algorithm

If I consider a plot of circle, in one octant in this case there is a point in the circle in second octant it is an interesting property the circle offers that given this point to the value (x, y) how many additional points I can know from this (x, y)? It is 3 because there is a reflection to (x, y) and there are also reflections to this where x is equal to y so there exist what we call as eight way symmetry.

(Refer Slide Time: 8:14)



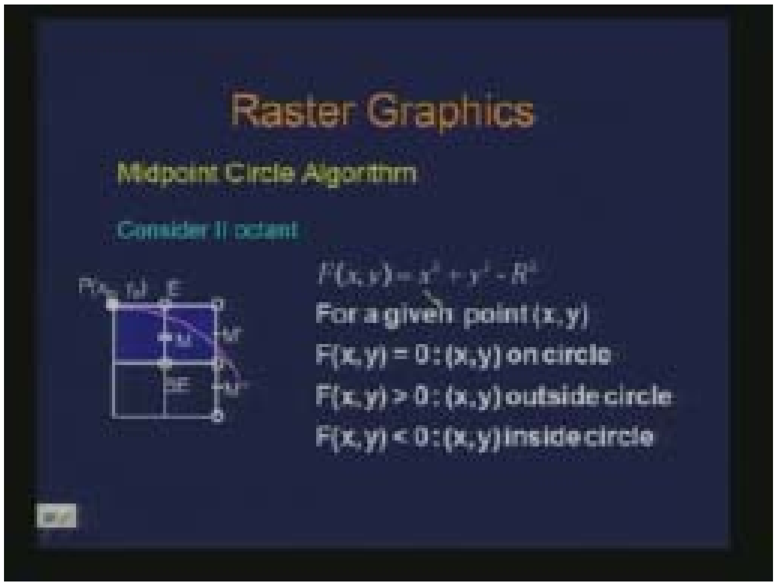
Raster Graphics

Midpoint Circle Algorithm

8-way symmetry: drawing in one octant is enough

In case of given this point (x, y) in this option I can know seven other points just by the value (x, y) which indicates that if I plot the circle in this option I can plot the entire circle. That is what we are going to look at and we still want to use something similar to what we did in line drawing. So we have a representation similar to the implicit representation of the line which was in terms of ax plus by plus c is equal to 0 and here also we have an implicit representation in terms that x to the power 2 plus y to the power 2 is equal to r power 2 which I can say as F(x, y) which is nothing but x to the power 2 plus y power 2 minus r power 2 and I still want to do this in a very simple fashion where I just need to, for instance, check a sign for this function to be able to decide what should I do next.

(Refer Slide Time: 9:27)



Raster Graphics

Midpoint Circle Algorithm

Consider II octant

$F(x, y) = x^2 + y^2 - R^2$

For a given point (x, y)

F(x, y) = 0 : (x, y) on circle

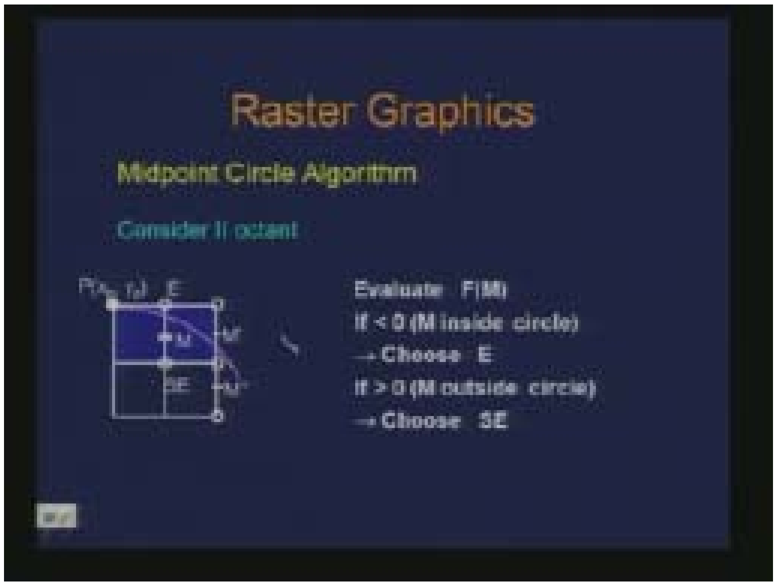
F(x, y) > 0 : (x, y) outside circle

F(x, y) < 0 : (x, y) inside circle

P(xp, yp)
E
M
M'
SE
M''

And this for instance for a given point (x, y) if I have an evaluation of F(x, y) where it equals to 0 that means x, y lies on the circle. If it is greater than 0 I know that (x, y) is outside the circle and if it is less than 0 it is inside the circle. That is what we want to explain. And then once again if I have a point plotted p xp yp then the candidates which need to be looked at the point E and the point SE. Remember we are talking in the second octant which in turn suggests that if I do an evaluation of the point M which is the midpoint between SE and E and try to answer this question on the value of F(x, y) and just evaluate the sign of it then it indicates to me whether I should consider E or SE. Again we are talking in terms of evaluation of the midpoint to this implicit function F(x, y). So it is a straight forward extension to what we have seen in the case of lines.

(Refer Slide Time: 11:36)



Raster Graphics

Midpoint Circle Algorithm

Consider II octant

Evaluate F(M)

If < 0 (M inside circle)

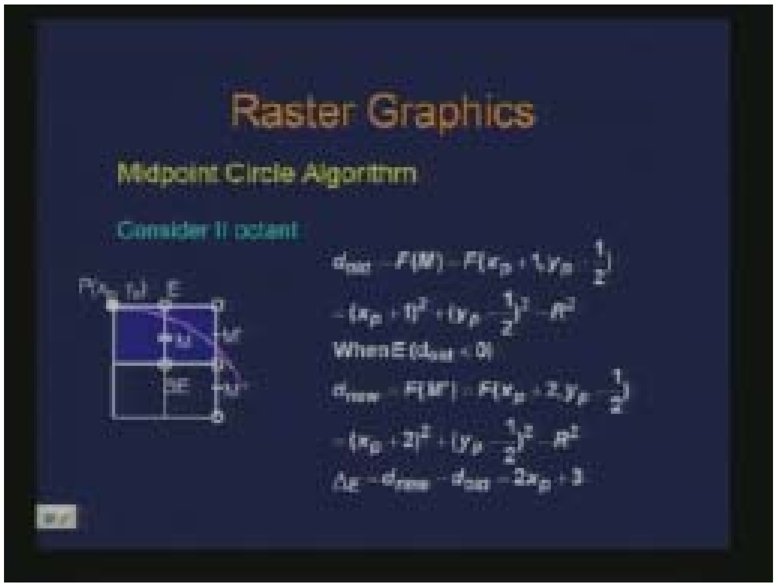
→ Choose E

If > 0 (M outside circle)

→ Choose SE

Therefore we evaluate F(M) and if it is less than 0 and is inside the circle and being inside the circle means that the point E is closer to the circle therefore E should be plotted we choose E and if it is greater than 0 then it is outside the circle then we need to choose SE. So, now we have the algorithm just the way we had line drawing we have circle drawing. The only thing is again we have to worry about how we change these d primes.

(Refer Slide Time: 12:27)



Raster Graphics

Midpoint Circle Algorithm

Consider II octant

$d_{old} = F(M) = F(x_p + 1, y_p - \frac{1}{2})$

$= (x_p + 1)^2 + (y_p - \frac{1}{2})^2 - R^2$

When E ($d_{old} < 0$)

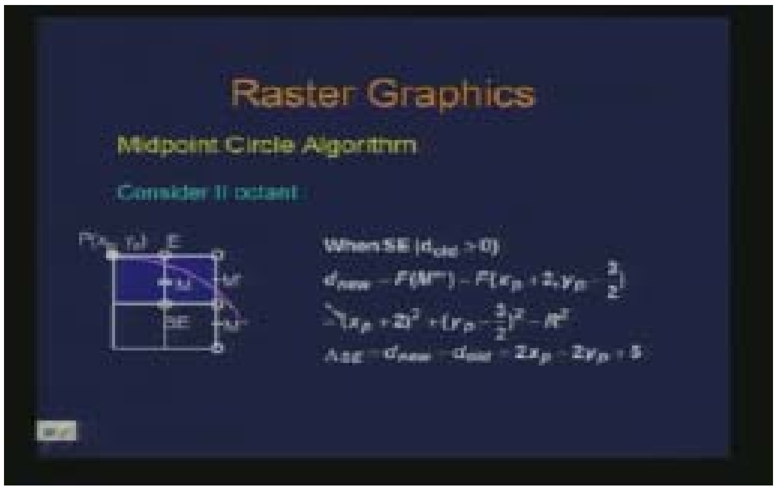
$d_{new} = F(M') = F(x_p + 2, y_p - \frac{1}{2})$

$= (x_p + 2)^2 + (y_p - \frac{1}{2})^2 - R^2$

$\Delta_E = d_{new} - d_{old} = 2x_p + 3$

What is the update of d primes according to whether we have plotted the E point or the south east point. So, once again we have the old d which was the evaluation of F(M) which is nothing but xp plus 1 and yp minus 1/2 so we just substitute that and if it is E which is plotted which is from the evaluation of the fact that d old is less than 0 we evaluate new d. That means what we are saying is that E was plotted. If E was plotted then I need to consider M prime for the subsequent evaluation. I get d new as F(M prime) which I just substitute x as xp plus 2 and y as yp minus ½. This is what I get by substituting. And then I get the change in d which I call it as delta e as d new minus d old which comes out as 2xp plus 3. So similarly I can do this when it is SE.

(Refer Slide Time: 14:24)



Raster Graphics

Midpoint Circle Algorithm

Consider II octant

When SE ($d_{old} > 0$)

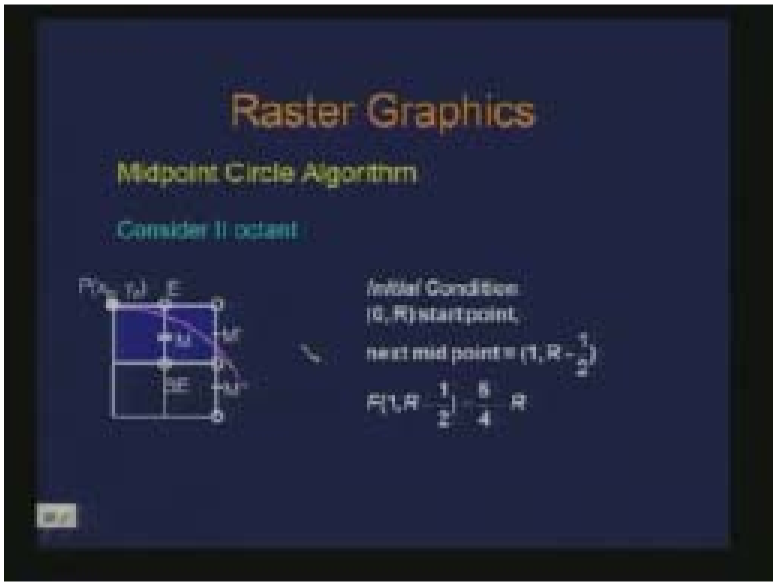
$d_{new} = F(M'') = F(x_p + 2, y_p - \frac{3}{2})$

$= (x_p + 2)^2 + (y_p - \frac{3}{2})^2 - R^2$

$\Delta_{SE} = d_{new} - d_{old} = 2x_p - 2y_p + 5$

If it is SE that means d old is greater than 0 so again the equality you consider in either of the cases and be consistent. And equality now has been considered as SE so when it is SE I do the evaluation of F with respect to M double prime that means I had considered SE for the point to be plotted. Therefore the midpoint which needs to be considered is M double prime so I evaluate d new through M double prime and I substitute the corresponding (x, y) values for M double prime and I obtain this. And then again the change in the value of d which I obtain as d new minus d old comes out as 2xp minus 2yp plus 5. So I have the basic steps which I need for the algorithm except that I still need to worry about what happens on the start the initial value of d. So what happens at the initial value of d?

(Refer Slide Time: 15:54)



Raster Graphics

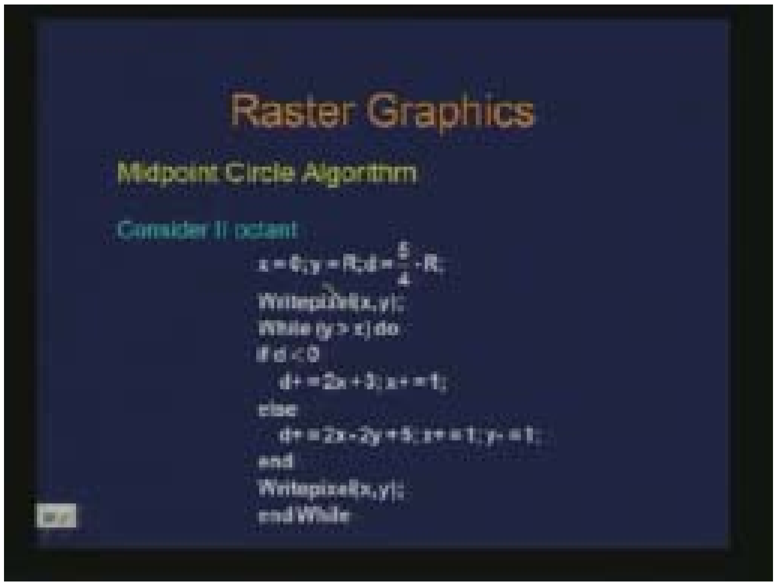
Midpoint Circle Algorithm

Consider II octant

Initial Condition:
(0, R) start point,
next mid point = $(1, R - \frac{1}{2})$
$F(1, R - \frac{1}{2}) = \frac{5}{4} - R$

We are talking in second octant, so I start from the point which is at 0R. So the next midpoint which needs to be considered is 1R minus ½. I just do the evaluation for this midpoint F1 R minus 1/2 which gives me 5/4 minus R. So with this I have all the necessary steps to come out with the algorithm to plot the circle in second octant and using the symmetry of the circle I can plot the entire circle. That is what the algorithm looks like.

(Refer Slide Time: 17:01)



Raster Graphics

Midpoint Circle Algorithm

Consider II octant

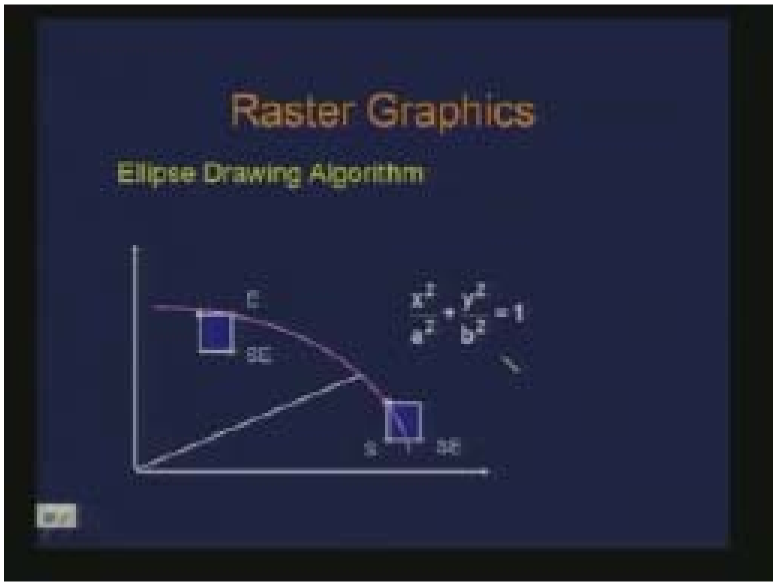
```
x = 0; y = R; d = $\frac{5}{4}$ - R;
Writepixel(x, y);
While (y > x) do
if d < 0
    d+ = 2x + 3; x+ = 1;
else
    d+ = 2x - 2y + 5; x+ = 1; y- = 1;
end
Writepixel(x, y);
end While
```

I start from x is equal to 0 y is equal to R the d start is this and right pixel means I plot the pixel I draw the pixel at that location (x, y) and this while indicates that I am working in the second octant. So if d is less than 0 I consider the delta E for updating d and I just change the value of x and when it is otherwise then I am updating d through delta as E which means I plot the SE point where x is incremented by 1 and y is decremented by 1 and so on. Therefore we have the algorithm. Now once again this evaluation of d indicates that I am doing a real arithmetic so may be I would like to avoid that.

If I have d is equal to 5/4 minus R the question is that what happens if I just consider d is equal to as 1 minus R. In other words I am defining some other d which I call it D which is d minus 1/4 does that affect the algorithm? Comparison would be d less than 0 will imply that D less than minus 1/4 does that change anything? The fact that we are talking in terms of incrementing by 1 every time, it actually does not. So the algorithm remains intact even if when I change this evaluation to 1 minus m.

We will do this comparison in D should be less than minus 1 micron no I am now just saying that that is what it implies if I am saying small d less than 0 but if I just say instead of D less than minus 1/4 to D less than 0 it does not really change anything because of the way I am going to increment. When it is told d less than 0 it implies to doing this comparison in D less than minus 1 micron. But if I just say instead of D less than minus ¼ say D less than 0 it does not really change anything because the way I am doing the increments. We already have line algorithm, circle drawing algorithm, now we can move on to some other primitives for instance an ellipse.

(Refer Slide Time: 20:44)

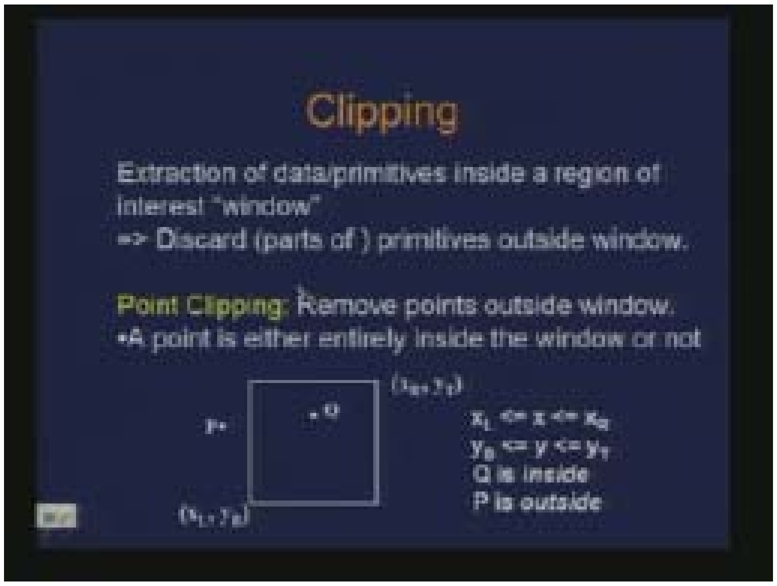

Raster Graphics

Ellipse Drawing Algorithm

In anything if we have some implicit representation we can actually think in a similar way. Here we have the equation of the ellipse given as x square/a square plus y square/b square is equal to 1. The only difference here is that I do not have this nice 8 way symmetric which I had in the case of circle. But I do have some properties by which I can partition the computation. One instance is, if you look at one region here, here if I had a plotted point then the next point which needs to be considered are E and SE whereas in this region if I have a point this is plotted then I need to consider the point S which is the south and SE and that is basically governed by the fact the slope of the curve. If I consider a tangent here, so if this is the break then I have the slope equal to minus 1. Or in other words I can also talk in terms of the gradient which is the sort of easier to compute the fact that I have this representation in an implicit form which is basically f(x, y) and the gradient of f(x, y) can be computed straight forward. So again I can map the problem of plotting ellipse in a similar fashion the way I did for a circle. This is what we have as the midpoint algorithm used in different primitive slots. This is about drawing primitives or rasterization of primitives.

Next thing which we are going to talk about today is the clipping. So we are now moving if you recall the rendering pipeline so we had the last process as rasterization and just before that we need to perform clipping.

(Refer Slide Time: 23:50)



Clipping

Extraction of data/primitives inside a region of interest "window"
=> Discard (parts of) primitives outside window.

Point Clipping: Remove points outside window.
•A point is either entirely inside the window or not

(x_R, y_T)
P
Q
(x_L, y_B)

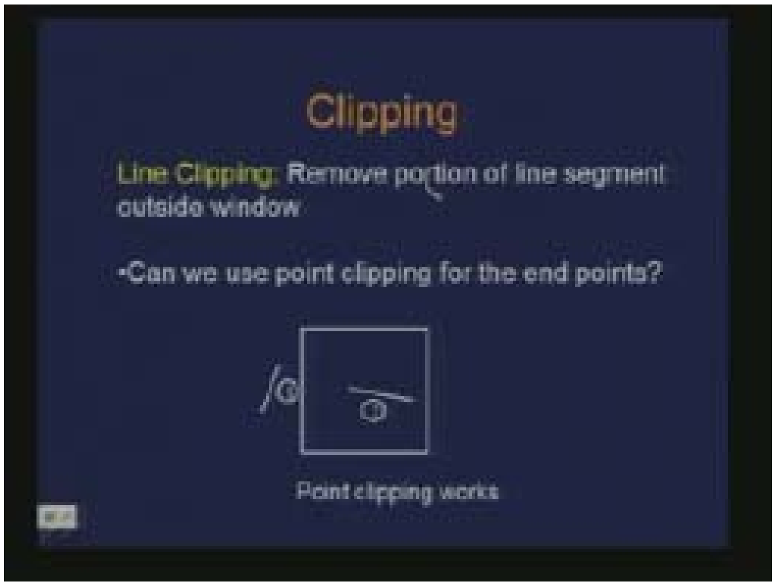
$x_L <= x <= x_R$
$y_B <= y <= y_T$
Q is inside
P is outside

What we mean by clipping? Clipping is basically extraction of the primitives for the data you have inside a region which we generally call it as the window. So whatever comes in that window that is what we are interested at which in turn means that we are looking at a scheme by which we can discard part or fully the primitives which are outside the window. This is what we mean by clipping. Again we can look at in terms of the complexity of the primitives to answer this question whether that primitive is inside or outside that window. That is why we talk about the smallest primitive which is a point.

Therefore we are trying to answer this question whether that point lies within the region of interest of the window or it is outside that window. Now for the purpose of simplicity and convenience let us assume that the window is a rectangular window. There is no reason that window is always rectangular but for the purpose of simplicity we consider that window to be a rectangular window. We are looking at the question of whether a particular point is entirely within that window or it is outside that window. And typically if we have a rectangular window we define the window through these points which are the lower bottom most left most point and the top most and right most point.

Now, through this given that I have the coordinate of point in question as (x, y) the simpler evaluation of (x, y) with respect to the extent of the window which is defined in x and y I can answer this question. Just looking at whether x is greater than or equal to xL and less than equal to xR and y is greater than or equal to yb and less than or equal to YT I am able to answer this question whether the point is inside the window or not. With this we can easily see that the point Q is inside and the point P is outside the window.

(Refer Slide Time: 27:17)



Clipping

Line Clipping: Remove portion of line segment outside window

•Can we use point clipping for the end points?

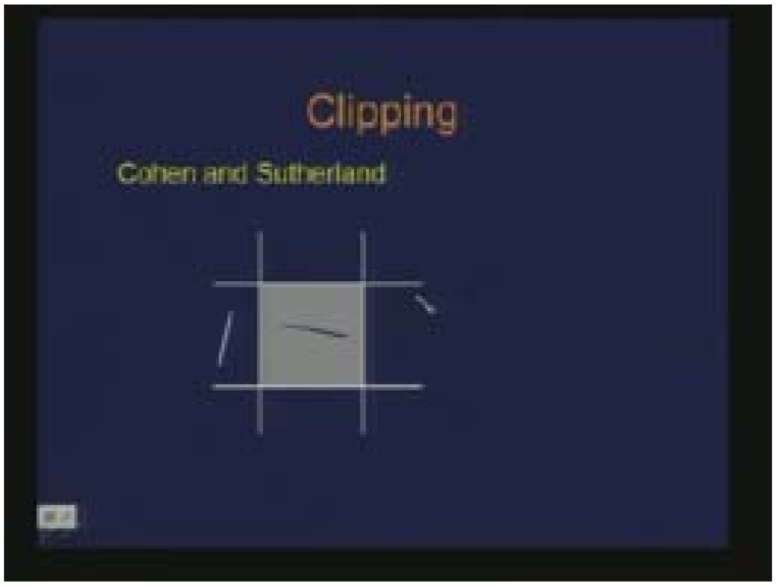
Point clipping works

Line: Is another primitive which is slightly more complex. We would want to perform clipping of line. That means again looking at if there is a portion to be removed or not which is outside the window. Now the question is that since I know how to clip a point that means to answer this question whether a point is outside or inside the window can I extend that idea to answer the question whether a line is inside or outside the window. If the frame is rectangular it is possible to say so. Obviously there are cases where this formulation allows me to do the clipping of line. For instance in this case let us say this point is out, this point is out, this point is out, this point is out and I can perhaps say that this is inside and this is outside.

But what happens to these? Of course there is a point which is inside. So when I am talking about the point clipping to be used in the context of line clipping I am basically considering the end points of that. So I am trying to answer the question with respect to the end points of the line and then see whatever answer I get for the end points whether I can declare that the line is inside or outside. So clearly these cases where one of the point turns out to be inside and one of the point turns out to be outside I have to do something more. I cannot say that the line is rejected or line is accepted, similarly for this.

But here is the case that the two end points turn out to be outside the window if I do the evaluation in the manner which we have seen but the line is not outside the window completely. This suggests that I cannot simply use point clipping to do line clipping I need to do something more about it. But then first of all I may be interested in devising a mechanism through which I can reject or accept certain simple cases and do further evaluation if need be. May be the first pass could be such that I have the mechanism by which I can suggest this is not possible to be inside or this is completely inside. That is what we are trying to look at first.

(Refer Slide Time: 30:41)



Clipping

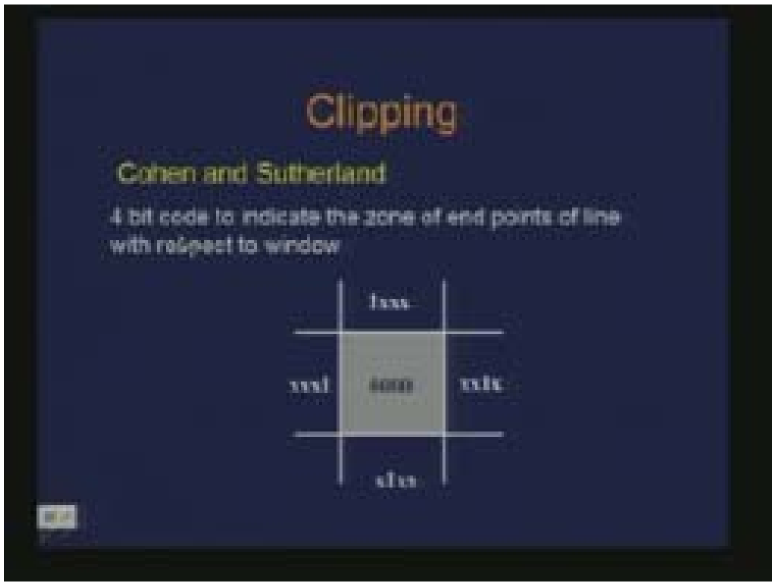
Cohen and Sutherland

In fact Cohen and Sutherland introduced the clipping algorithm. And once again there is the assumption of the window to be rectangular. And if you extend this window and define the adjoining zones to the window and then decide from the fact that where the particular line lies in the zones around that window then may be you can answer this question whether I should consider the line which is inside or outside. For instance, if I have these two points which are in the same zone here and which turns out to be outside the window I have a way to say that the line is outside. And similarly when I have these two end points inside this then I have a way to say yes the line is inside. What are we trying to do is we are trying to assign some sort of a scheme by which we have the notion of these zones around this window and then we do the evaluation of the end points with respect to these zones.

Now, is there a way by which we can assign a certain representation by which we capture these nodes? So here is the suggestion; since we have the extent of that window then we can define these zones in terms of comparing with respect to the extent of the window these zones. What I am also interested in is that some coding scheme in which I assign some code to this window or the zones of the window which actually gives me a very easy way of checking or comparing the end points whether they are outside or inside or which side or which zone. In other words I am trying to assign some code to various zones I have around this window and as well for the window. So what are we trying to say is, there is this line and all I need to answer is that things which are above this line they are out if they are below this line they are in with respect to this line.

Similar question is answered with respect to this line, a similar question is answered with respect to this line and a similar question is answered with respect to this line so all I am saying is a one bit on and off with respect to these four edges of the window. So I can have some kind of a four bit code which would decide about the various zones I have including the window. That is what Cohen and Sutherland exactly did. They came out with this four bit code indicating the zone of the end points with respect to the windows.

(Refer Slide Time: 34:49)



Clipping

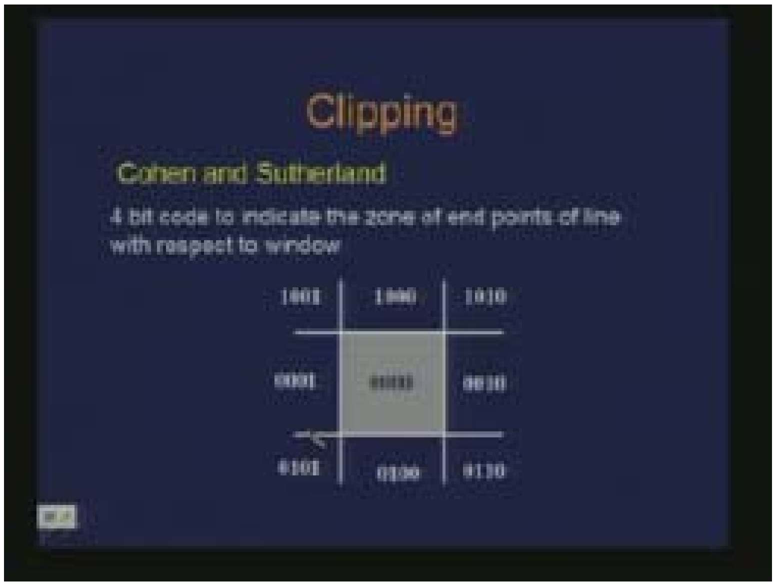
Cohen and Sutherland

4 bit code to indicate the zone of end points of line with respect to window

1xxx
xxx1
0000
xx1x
x1xx

For instance, I may designate 0 0 0 saying that this is my window and this bit is on for the region which is above the window, this bit is on for the region which is right to the window and so on. So now I can have the complete coding scheme for all the zones.

(Refer Slide Time: 35:40)



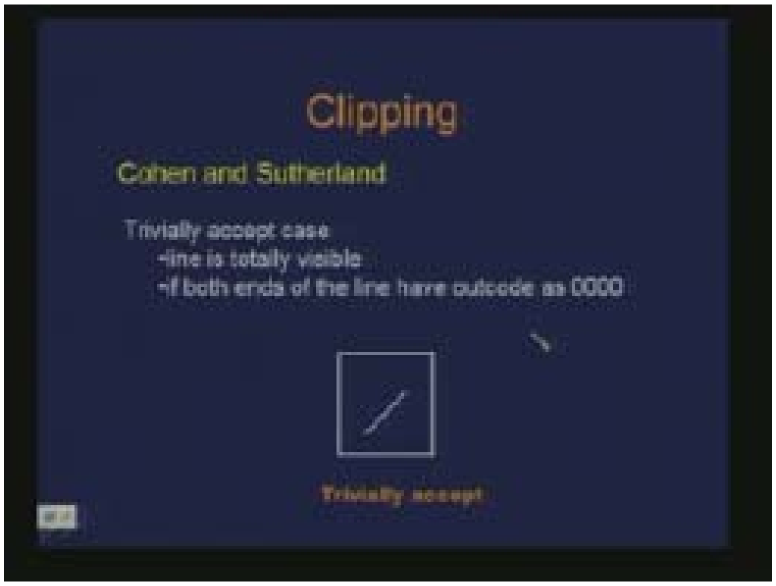
Clipping

Cohen and Sutherland

4 bit code to indicate the zone of end points of line with respect to window

This is basically a combination of the fact that it is left to the window and above the window that is why there are two bits which surround. So now have a coding scheme which we can designate to the end points of the line and by which we locate the end points of the line in various zones. Now the question comes that how do I decide whether I can reject a line easily or I can accept a line easily? Some bit wise operation can actually guide us to this. The simple acceptance is rather easy, the two end points have to have 0 0 0. That is basically an example of trivially accept.

(Refer Slide Time: 37:04)



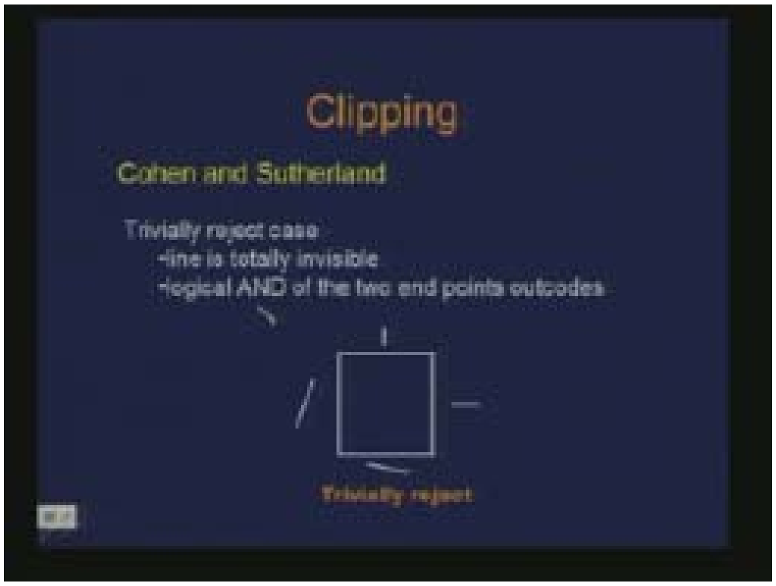
Clipping

Cohen and Sutherland

Trivially accept case
- line is totally visible
- if both ends of the line have outcode as 0000

The two end points have the end code as all bits 0. Now a suggestion is given that some bit wise operation actually indicates to me what should I do for rejecting a line and that is what we are looking at. So we have a mechanism by which we trivially accept a line and similarly we are trying to find a mechanism by which we trivially reject a line bit wise AND.

(Refer Slide Time: 37:55)



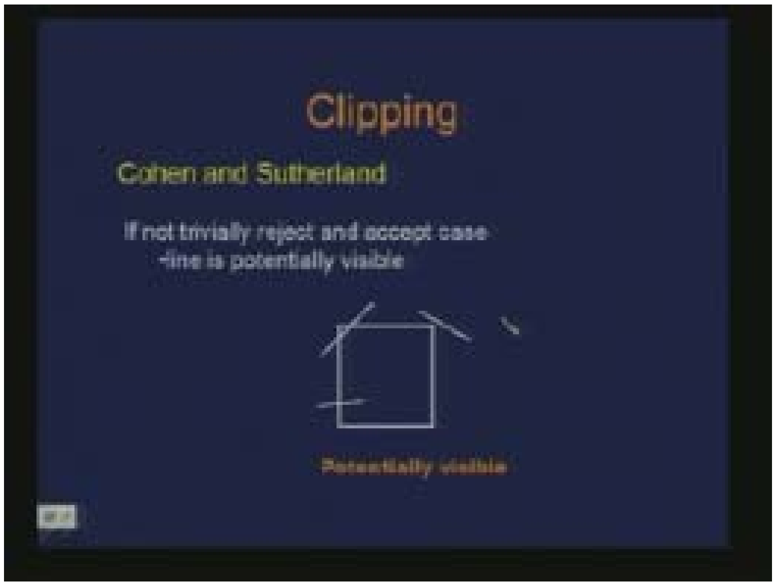
Clipping

Cohen and Sutherland

Trivially reject case
- line is totally invisible
- logical AND of the two end points outcodes

So if I do a logical AND, and if it does not turn out to be all 0s then I have a trivially reject case 0. So these are the cases where I can consider a trivially rejection. Now we have a case known as potentially visible.

(Refer Slide Time: 38:30)



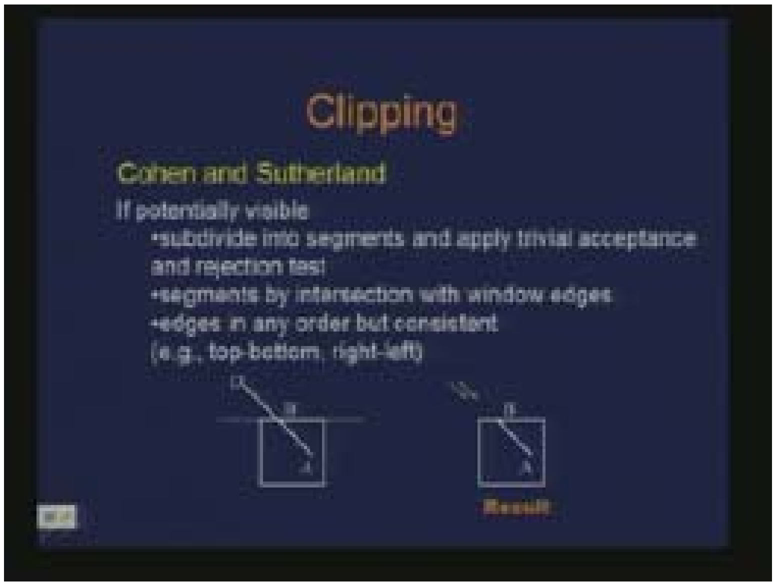
Clipping

Cohen and Sutherland

If not trivially reject and accept case
- line is potentially visible

Potentially visible means this logical AND operation is 0. So if it is potentially visible we have these cases. Now if we have such a case it demands something extra.

(Refer Slide Time: 39:23)



Clipping

Cohen and Sutherland

If potentially visible

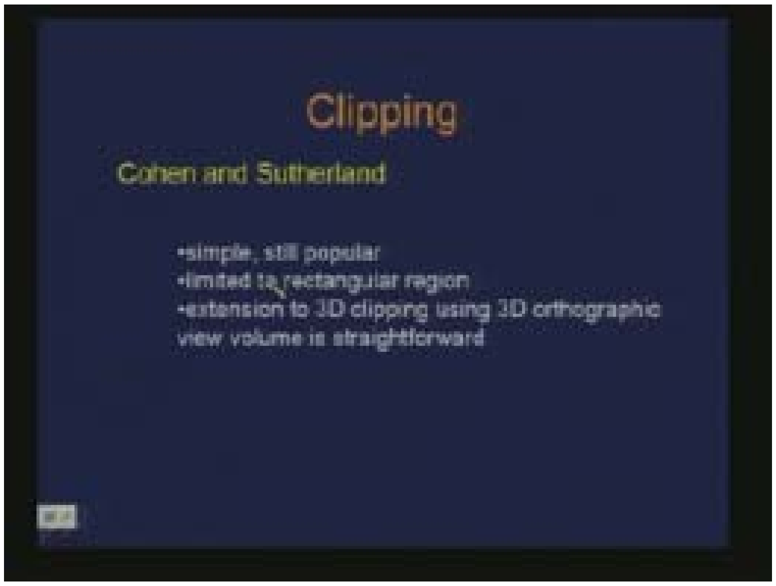
- subdivide into segments and apply trivial acceptance and rejection test
- segments by intersection with window edges
- edges in any order but consistent (e.g., top-bottom, right-left)

D
B
A
B
A
Result

If we now have something extra to do we can actually do a simple strategy where we can divide this into segments the line in question which is potentially visible we subdivided into segment and then apply trivially acceptance of rejection tests to these segments. For instance, in this case if this is the division of the segments I can then find out that BD segment would get trivially rejected whereas AB segment will get trivially accepted. So we need to some subdivision of the line and keep performing this trivially rejections and acceptance operation.

There are various ways in which one can look at it. One may be that you just have a fixed subdividing strategy which could be just divided by two and then keep doing in those segments. What you are having eventually is some sort of a fragmentation of these lines and then each fragment in turn gets tested. And here one needs to be just cautioned that you can choose any order of the edges you have but be consistent to it. Therefore this basically gives you the entire algorithm for Cohen and Sutherland. It is an extremely simple algorithm and is still a popular one because from the fact that this assumption of rectangular window is relevant.

(Refer Slide Time: 41:53)



Clipping

Cohen and Sutherland

- simple, still popular
- limited to rectangular region
- extension to 3D clipping using 3D orthographic view volume is straightforward

In most applications we have the windows are rectangular. That is why it is still popular though with this limitation. Now, if one wants to extend this idea of clipping encoding the different zones you can extend this idea to 3D clip. All you are saying is that there is a 3D volume and each phase of this volume is basically acting as the edge of the window we had in 2D case. So the test which we need to perform is with respect to a plane but it is exactly the same test. And what is the size of the bit code you would get? It is 6 because you are going to answer this question with respect to each phase of this orthographic one. So the size of the bit code you have is 6. Similarly if you have a window which is a triangle in a 2D case all you need is 3 bits. This is very simple but an elegant way of doing clipping.

Next let us try to relax this limitation of rectangular region and come out with an algorithm of clipping where the region of the window is not necessarily rectangular. The constraint which we will still impose is the region is convex. Now the question is how do we find out the point of intersection with respect to edge of the window? The size of the window is defined in terms of the extent of the window. you have this xL, YB, xR, yT that always define these equations of edges which is just a matter of checking the value of (x, y) with respect to this. So you have the equation of edges, you have the equation of line the fact you have the end points and you can find intersection. Here we are addressing the issue of real arithmetic and not integer arithmetic.