

Computer Graphics
Prof. Sukhendu Das
Dept. of Computer Science and Engineering
Indian Institute of Technology, Madras
Lecture – 17
Scan Converting Lines, Circles and Ellipses

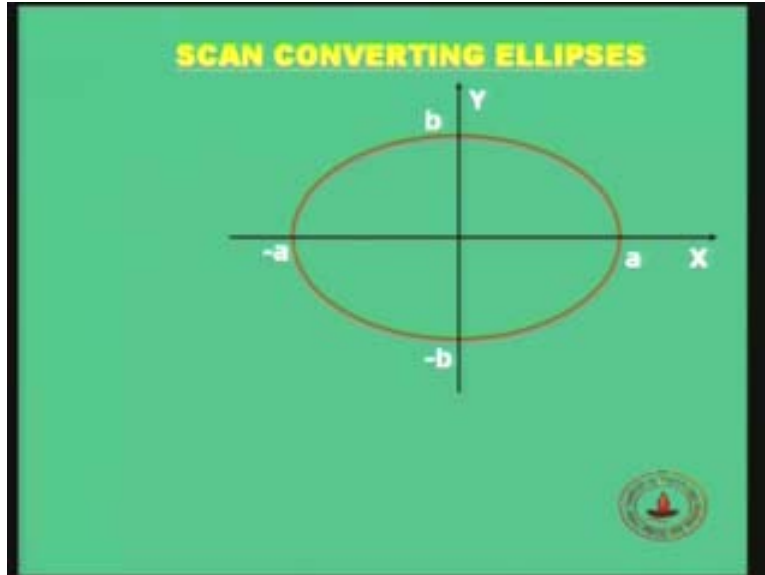
Hello and welcome everybody again, to the lecture series on computer graphics. And in the last few lectures, we have been discussing how to draw lines and circles on the screen, using an integer based algorithm, how to draw it efficiently and quickly and how accurately also as possible. And the criteria for drawing points, pixels for a line or a circle has been the midpoint line criteria or Bresenham's algorithm for drawing lines and circles.

We had seen an example also in the case of a line as well as in the case of a circle, where, we not only used first order differences but also the second order differences to come up with an integer based algorithm where within the loop, you increment the first order differences by integers only and that is how you update the first order differences and get the next value of the function at the next midpoint and so on. And basically at each iteration you have only two choices between the next two pixels to choose from.

So in the case of line if you remember, you had to choose from between east and north east pixels because we were operating in the first quadrant. In the case of a circle you are basically operating in the second octant, to be precise. And once you were getting choosing points between the second octant the choice was basically between the east and south east pixels. So, you know these two different criteria, in fact the same criteria but in two different methods for linear case of a straight line and for a non-linear case in equation for a circle how to evaluate and draw points.

The last of these functions which we will see today is how to draw an ellipse? Now we had discussed this earlier that the circle is a special case of an ellipse. So if you actually have the equation in a method of ellipse, you can actually draw a circle or loop, as a special case we had seen equations of a circle separately and evaluated the midpoint criteria and also seen how to draw the algorithm. So today we will start with ellipse, ellipse drawing algorithm using midpoint criteria, Bresenham's drawing for method of drawing an ellipse. Let us look at the first slide, so the ellipse drawing, methods and conversion of an ellipse is what we have to see.

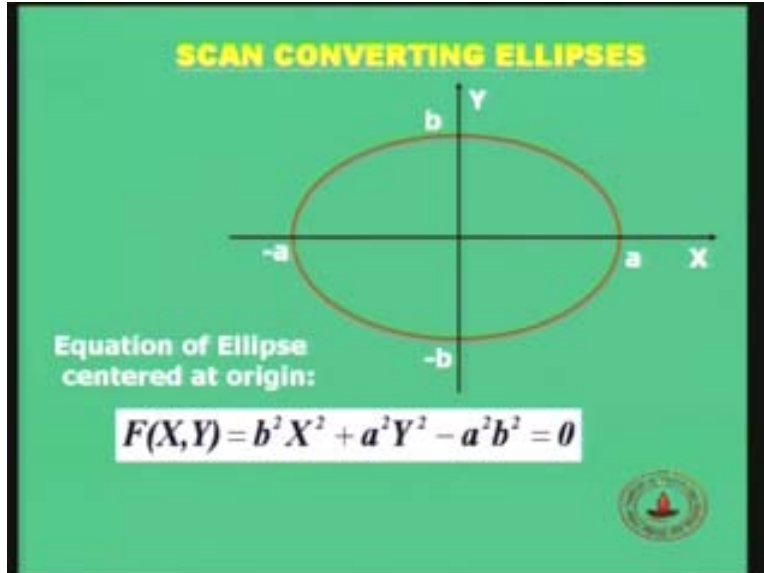
(Refer Slide Time: 00:03:51)



This is the example of an ellipse in two dimensional coordinates system X and Y where you see in this equation, it is centered at the origin, that is what we assumed and we do not bother, in fact, if it is not at the origin or if it is also not aligned with the X and Y coordinates, we know how to do two dimensional transformations and then put it at any point in 2D space. As you can see that the ellipse intersects the X axis at two points in positive and negative direction and also the Y coordinate along in the positive and negative direction.

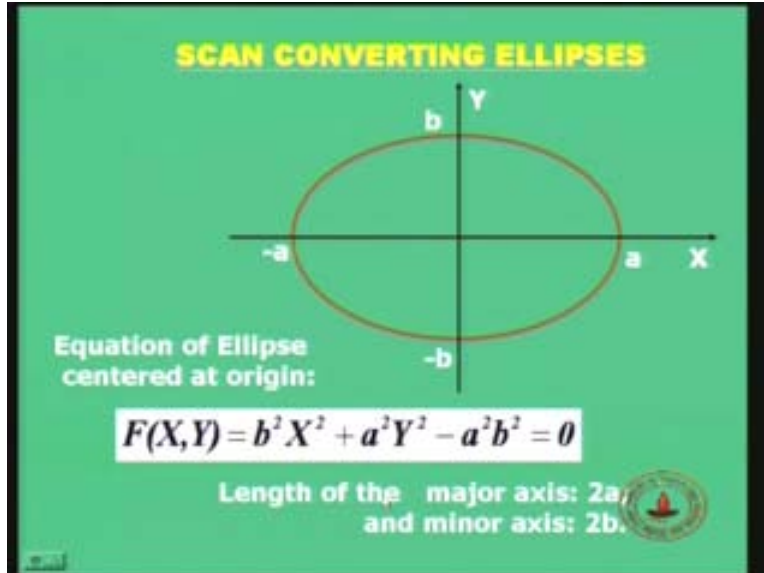
Difference in the case of an ellipse with respect to a circle is its intercept value plus a and minus a along the X axis and the intercepts plus b and minus b along the Y axis would have been same. That is the value a and b would have been the same in the case of a circle in the ellipse which is typically not equal to b. A special case of a equal to b is the case of a circle.

(Refer Slide Time: 00:05:00)



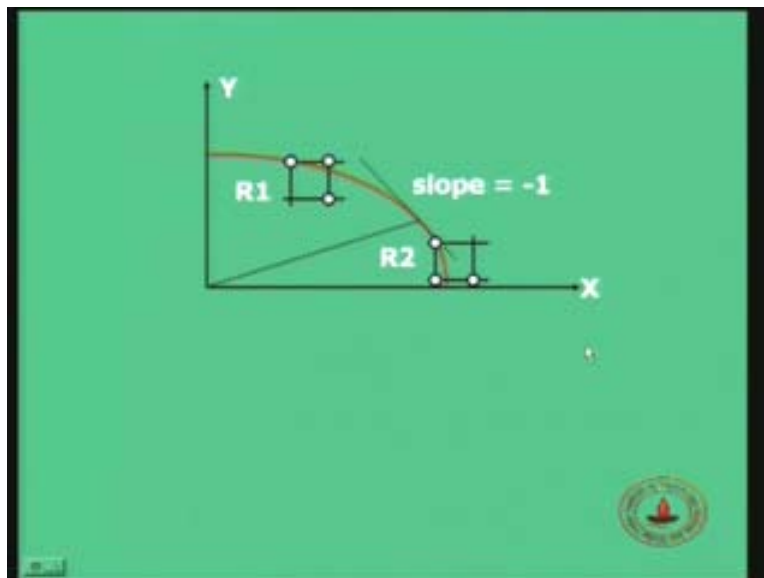
And let us look at the equation of the ellipse centered at the origin where we will say the X axis is considered as the major axis of the ellipse where the intercept is larger and the intercept along the Y axis of distance 2b plus b2 minus b is the minor axis. The equation of an ellipse is b square X square plus a Y square. It is represented in the form of implicit representation, actually it can be written as X square by a square plus Y square by b square is equal to 1. That is the typical expression of an ellipse but we know now in the case of circle as well as for the case of a line we have represented this functional form as F(X, Y) equal to 0. So, that is the equation of an ellipse in the form of F(X, Y) equal to 0, where the length of the major axis that is the X axis basically is 2a and the length of the minor axis is equal to 2b. So 2a and 2b are the intercepts along the major axis and minor axis.

(Refer Slide Time: 00:05:52)



I did say again, that X axis is the major axis in this case. The intercept is running from plus a to minus a and plus b to minus b for the minor axis. If a is equal to b then the special case of an ellipse becomes a circle.

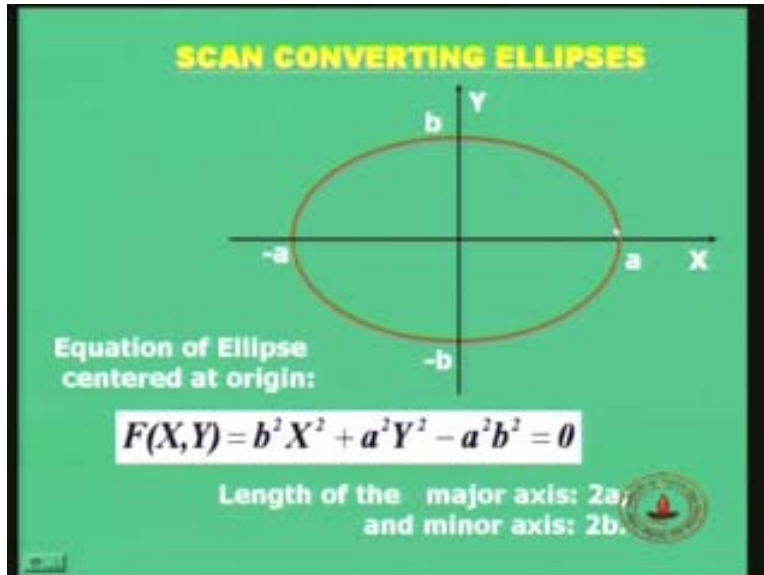
(Refer Slide Time: 00:06:17)



Let us now look at the pixels which lie on the ellipse. If you look into the first quadrant of a part of the ellipse which has to be drawn, if you look at the slide before you can see that if you draw only the points on the first quadrant then by symmetry you can obtain the points in the other three quadrants. This concept was also used in the case of a circle where basically we had an eight way symmetry where we found out the points lying on

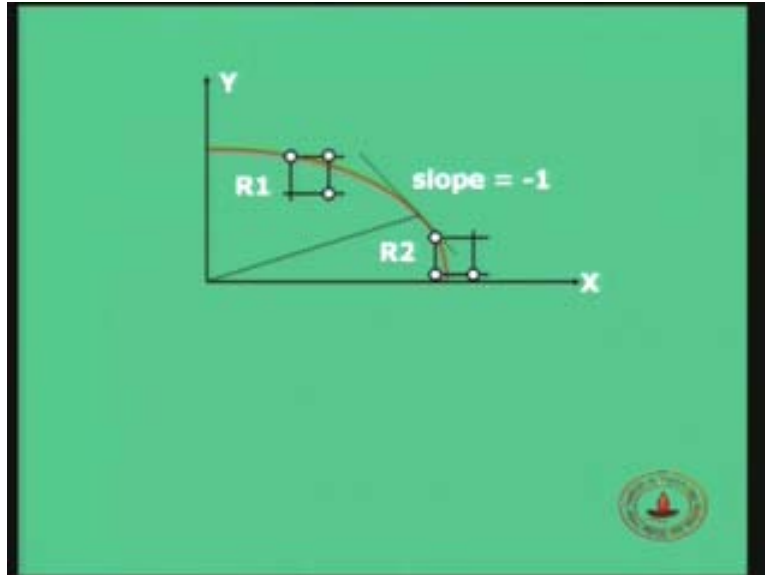
the second octant and all the other points were obtained by symmetry, the other seven points in the seven octants corresponding to a point in the second octant which was evaluated by the algorithm. The other seven points in the other seven octants were automatically obtained by symmetry.

(Refer Slide Time: 00:07:25)



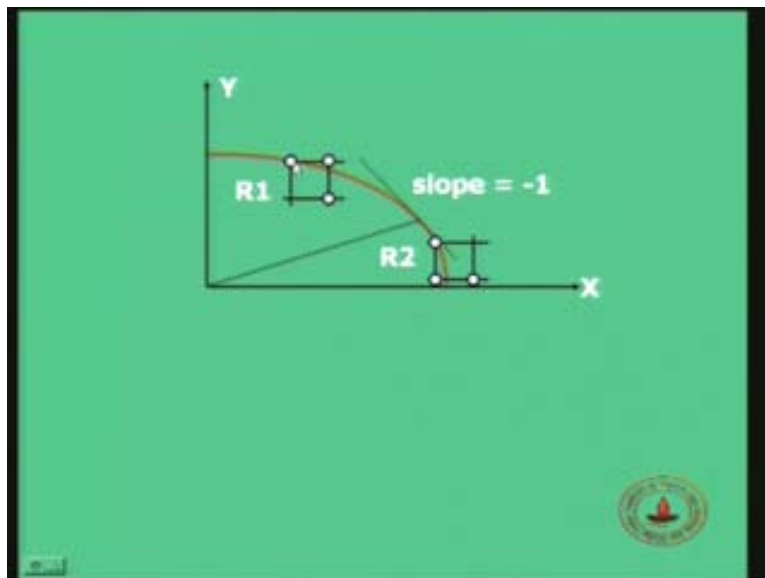
Now, if you look into this circle minus and if we see the points on the first quadrant from the positive intercept plus b on y axis to the section of the curve up to plus a on the x axis. If we somehow get these points by the algorithm, the rest of the points can also be obtained easily by symmetry. So by applying the same logic we basically have to find out 1/4 of the set of points of an ellipse, unlike the case of 1/8 in the case of a circle. So we look at the first quadrant.

(Refer Slide Time: 00:07:53)



And, this is the first quadrant which we have shown again here, and as we see here the reason why we cannot talk of octants but quadrants only in the case of an ellipse is that the choice of the pixels vary from one region within a quadrant to a second region. If you look at region R1 at any stage of iteration this is the current point marked here by the cursor then the next choice of the next pixel will be between east and south east. That is because, the slope of the curve is basically in magnitude less than 1. That is why the choice is between the east and the south east pixel.

(Refer Slide Time: 00:08:31)



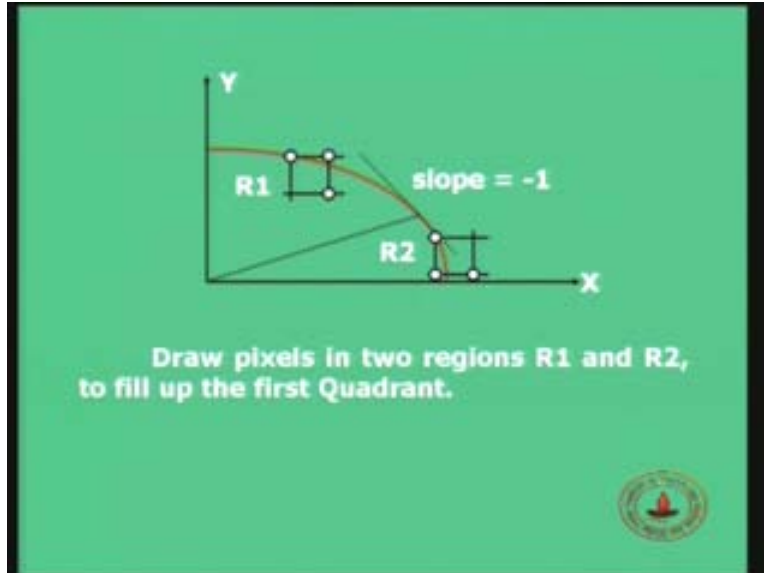
In region R2 the choice is between the south and the south east pixel. This is the case where the portion of the ellipse is nearing the X axis and the slope of the tangent to the curve in the region which we will say as R2, we will split the first quadrant into two regions, now R1 and R2. In the region R1, the slope of the tangent to the curve at any given point is less than equal to 1 and in R2 the slope is more than unity. This is the reason why in region R1 the choice is between east and south east pixel as clearly marked by two other points in the square grid. In region R1 and in region R2 if this is the point at any stage of iteration chosen, the choice in the next iteration will be between south and south east pixels. That is because the slope actually is more here, is more than 1 and hence the choice is between south and south east pixels.

So we have two regions R1 and R2 and we have two different conditions of the choice of the next pixel based on Bresenham's algorithm. The same concept as the case of a circle could be used here but the choice is not like that we have done in the case of the second octant for the case of a circle, whatever we wait for, the second octant is basically valid for region R1. In region R2 the choice has to be now between the south and the south east. And there exist a point on the ellipse where the two regions meet. That is the place where the tangent to the curve has the slope is equal to minus 1.

As you see here the slope is equal to minus 1 at a point which is the demarcating point between region R1 and R2. The points from the Y axis, the points on the curve from the Y axis to this point lies in region R1 and from this point to the X axis lie in region R2. This is the point where the tangent to the curve. I repeat, the tangent to the curve is equal to minus 1 and if you are in region R1, the slope is less than minus 1 in terms of magnitude.

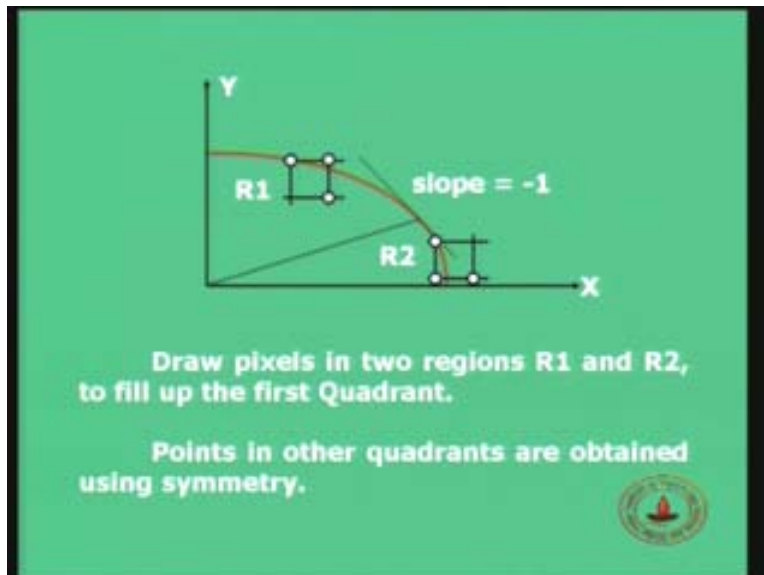
If you take magnitude of this slope and here the magnitude is more than unity. That means, in negative direction it is actually less than minus 1, but in magnitude it is more than 1. So these are the two regions which we have to see now and write two different conditions for the Bresenham's criteria. But in integer based algorithm, look at the first order difference and see what we will get.

(Refer Slide Time: 00:11:36)



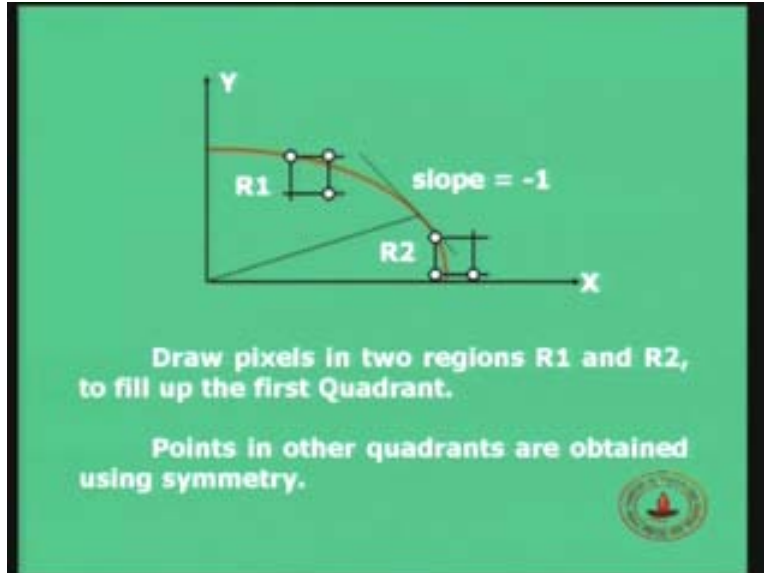
So, the task is to draw pixel in both these regions R1 and R2 and that helps us to fill up the pixels in the first quadrant.

(Refer Slide Time: 00:11:46)



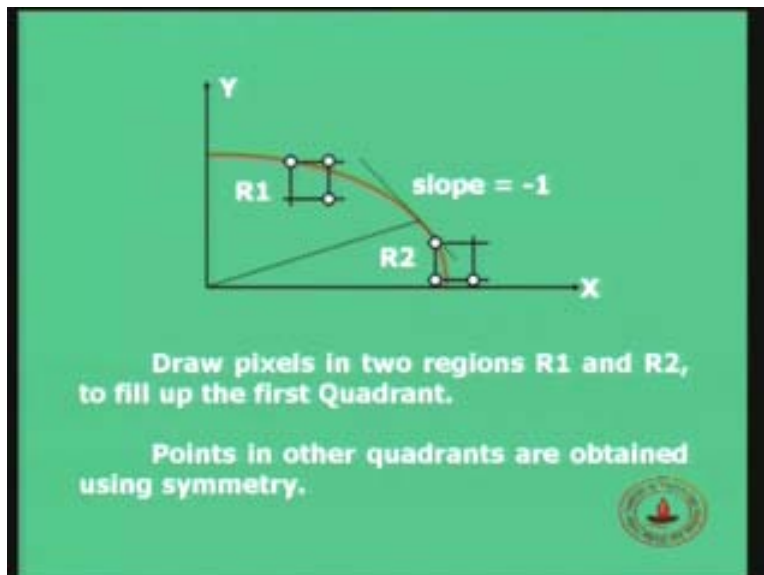
The points in the other quadrants are obtained using symmetry. That means once you have obtained all the points in region R1 and then the remaining points in region R2. This helps you to obtain all the points in the first quadrant. If you find out points in region R1 and then points in region R2, together they form the points in the first quadrant. We know that.

(Refer Slide Time: 00:12:12)



Then, using symmetry you can obtain the rest of the three points in the other three quadrants. For any point you can obtain the third with the three points in the other three quadrants, but if all the points are in a particular octant in this case, as the first quadrant is known to us, all the other points in the curve in the other three quadrants can easily be obtained using symmetry, we know that. So that is another point.

(Refer Slide Time: 00:12:43)

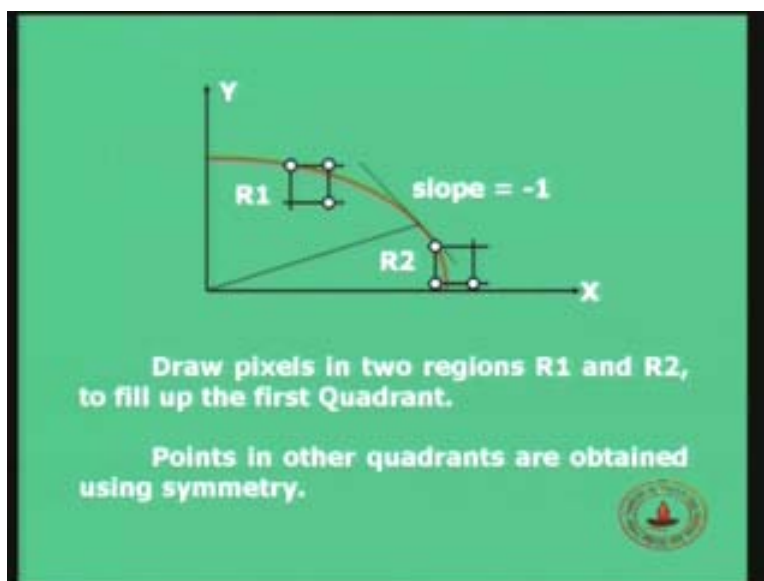


(Refer Slide Time: 00:12:45)



Points in the other quadrants are obtained using symmetry and we had also seen the necessity now to obtain the point and the contour are on the curve on the ellipse where the slope of the curve that is basically the slope of the tangent to the curve is minus 1 because that is the point which will demarcate it in region R1 and R2, isolate the curve into two parts one belonging to region R1 and another belonging to region R2. We have seen that diagram as it is given here.

(Refer Slide Time: 00:13:13)



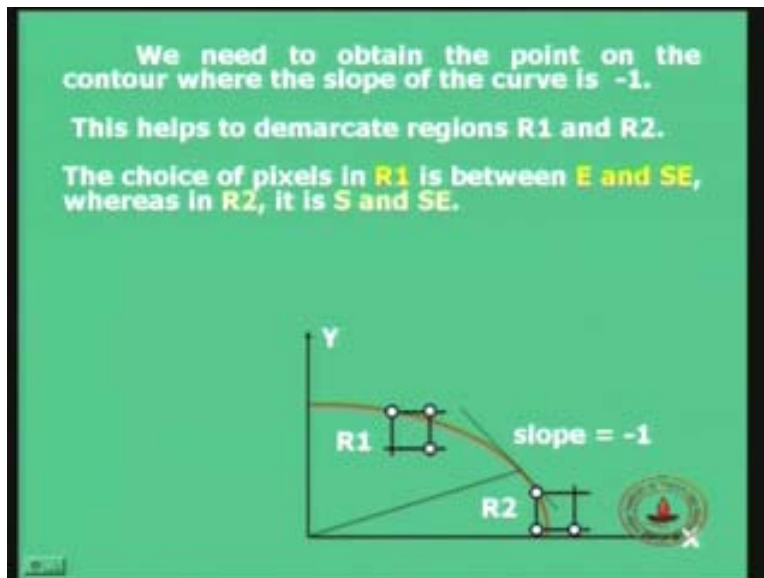
Part of the curve is in region R1 and the rest of the curve is in region R2. And we need to find out this point on the curve where the tangent to the curve has the slope equal to minus 1 so that is the task we need to do first.

(Refer Slide Time: 00:13:29)



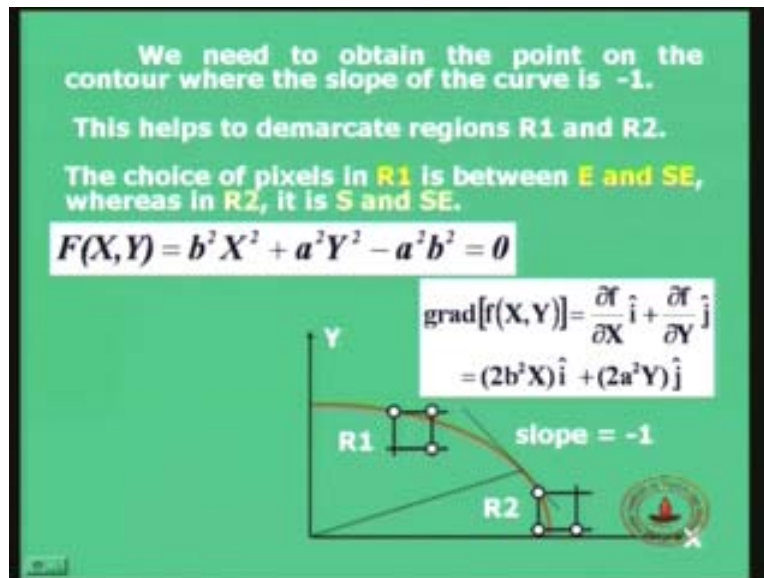
So once we are able to get this point this helps to demarcate regions R1 and R2 that also we know and the choice of the pixels as informed before and as given here in the figure at the bottom.

(Refer Slide Time: 00:13:38)



But the choice of the pixels in region R1 that is from Y axis to this point which splits the curve into two different regions, the choice of pixels in R1 is between east and south east pixels whereas in region R2, it is between south and south east pixels. So if you get this point it is easy for us because you have to apply the choice between east and south east for R1 and south and south east for R2. So, how to get this point? The point can be obtained by trying to find out the slope to the curve. How do you get the slope for the curve in a functional form $F(X, Y)$ equal to 0? The $F(X, Y)$ in the case of an ellipse is given here minus which is $b^2 X^2 + a^2 Y^2 - a^2 b^2 = 0$.

(Refer Slide Time: 00:14:28)



That is the implicit form of the expression for an ellipse. And you need to get the tangent to the curve which is given by the grad function differential. Remember, f is a function of X and Y although it is capital X and small f here but do not worry, it is the same functional form, f here and grad of $f(X, Y)$ is nothing but partial differential of with respect to X \hat{i} plus partial differential of f with respect to Y , \hat{j} at the unit vectors along X and Y respectively.

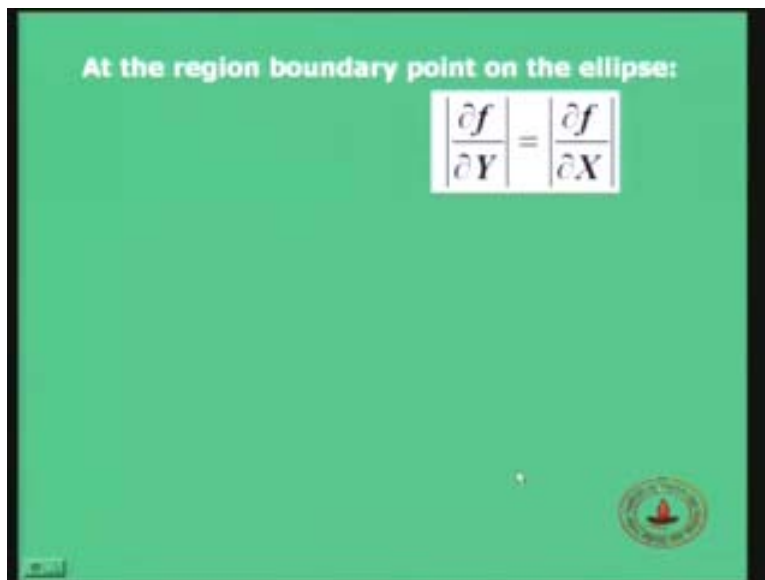
Assume, so this is the vector notation of grad , \hat{i} is the unit vector along X direction which is horizontal to the right and \hat{j} is the unit vector along Y axis which is a vertical vector pointing upwards. So these are two unit vectors and the grad is the partial derivative of F . You can use capital or small f interchangeably. But they are the same in this case and so take that partial derivative with respect to X and Y .

What is f with respect to X ? You see that this is only the one term with respect to X . So that is what we have to differentiate. So you will get it as $2b^2 X$. Partial differential with respect to Y is again one term with respect to Y . So you will get as $2a^2 Y$. So $2a^2 Y$ is here, $\text{d}f/\text{d}Y$ and $\text{d}f/\text{d}X$ we have seen is $2b^2 X$. That is the grad function. The grad function is hence defined as $\text{d}f/\text{d}X$ with unit vector, along X

direction and $\text{del } \text{del} Y$, that is the partial derivative with respect to Y direction with the \hat{j} vector which is unit vector along Y direction. So it is $2b^2 X$ unit vector along X, $2a^2 Y$ unit vector along \hat{j} .

Now if you look at the components of this grad function which gives the slope of the tangent to the curve at any point on the ellipse, you will see now that in region R1 you have the partial derivative along X to be more than the partial derivative along Y and in region R2 the partial derivative along Y will be more than partial derivative along X. If this happens only then the slope of the curve actually is numerically less than 1 in magnitude on region R1 and the slope is going to be more than warning region R2. We are not interested in the actual value of the slope at each point in region R1 and R2. We are going to use the equation of grad for the ellipse to obtain the point on the ellipse where the slope is equal to 1, is equal to minus 1 basically. So at that point what will happen is, minus at the region boundary, what are the region boundaries? The point which is the meeting point of region R1 and R2 on the ellipse, the partial derivatives along the X and Y directions are both same.

(Refer Slide Time: 00:17:17)




This is the property of the grad function which we will use to obtain that point on the ellipse which is as the demarcation point between region R1 and R2 because we will start at some point and we must know at any given point of time whether we are drawing pixels within region R1 or we have actually crossed over to region R2. So, this is the condition which we will use.

(Refer Slide Time: 00:17:52)

At the region boundary point on the ellipse:

$$\left| \frac{\partial f}{\partial Y} \right| = \left| \frac{\partial f}{\partial X} \right|$$

Based on this condition,
we obtain the criteria when the next mid-point
moves from R1 to R2 :

$$b^2(X_p + 1) \geq a^2(Y_p - 1/2)$$


And, if you have a look at this condition, based on this condition we obtain the criteria when the next midpoint moves from region R1 to R2, and the midpoint is if X_p Y_p is the point or pixel chosen at any given stage of iteration, the next point will be X_p plus 1, the next midpoint will be X_p plus 1, Y_p minus $1/2$ because, we were basically going to the right along X direction 1 and minus $1/2$ in that direction because the choice has been between east and south east in region 1. So if that is the next midpoint which we all will be choosing and we find out whether the next midpoint is already here otherwise carry on in region R2, if not, if this condition is satisfied, then we have moved on to region R2. And remember this is $\frac{\partial f}{\partial Y}$ is equal to $\frac{\partial f}{\partial X}$. So $\frac{\partial f}{\partial X}$ is more than $\frac{\partial f}{\partial Y}$.

(Refer Slide Time: 00:18:54)


At the region boundary point on the ellipse:

$$\left| \frac{\partial f}{\partial Y} \right| = \left| \frac{\partial f}{\partial X} \right|$$

Based on this condition, we obtain the criteria when the next mid-point moves from R1 to R2 :

$$b^2(X_p + 1) \geq a^2(Y_p - 1/2)$$

When the above condition occurs, we switch from R1 to R2.



Based on this minus when the above condition occurs; we switch from region R1 and move on to region R2. This is the condition which we will keep in mind which will help us to know that we have moved into region R2 from region R1. These points are obtained from the grad function formulas which we have defined and derived in the previous slide.

(Refer Slide Time: 00:19:22)

At the region boundary point on the ellipse:

$$\left| \frac{\partial f}{\partial Y} \right| = \left| \frac{\partial f}{\partial X} \right|$$


Based on this condition, we obtain the criteria when the next mid-point moves from R1 to R2 :

$$b^2(X_p + 1) \geq a^2(Y_p - 1/2)$$

When the above condition occurs, we switch from R1 to R2.

Analysis in region R1:

Let the current pixel be (X_p, Y_p) ; $d_{old} =$



So analysis in region R1 can continue. Let the current pixel be X_p comma Y_p and I defined a old value of a decision variable d , we are known to these terms, we have used it in the case of line drawing, we have used this in the case of ellipse drawing. So, we use the same term and analogy. I hope you remember all of them, you have worked it out and

we use the same. I do not need to redefine these terms of the decision variable, d start will have to be evaluated, d at the next point whether it is east and south east or south and south east, we can all work out. So let us look at this.

(Refer Slide Time: 00:19:53)


At the region boundary point on the ellipse:

$$\left| \frac{\partial f}{\partial Y} \right| = \left| \frac{\partial f}{\partial X} \right|$$

Based on this condition, we obtain the criteria when the next mid-point moves from R1 to R2 :


$$b^2(X_p + 1) \geq a^2(Y_p - 1/2)$$

When the above condition occurs, we switch from R1 to R2.

Analysis in region R1:
Let the current pixel be (X_p, Y_p) ; $d_{old} =$ 

Slide analysis in region R1 involves that let the current pixel be X_p comma Y_p and the d old is a function of f mid and next midpoint, that one subscript indicates that we are analyzing in region R1. The subscript one indicates that we are in region R1.

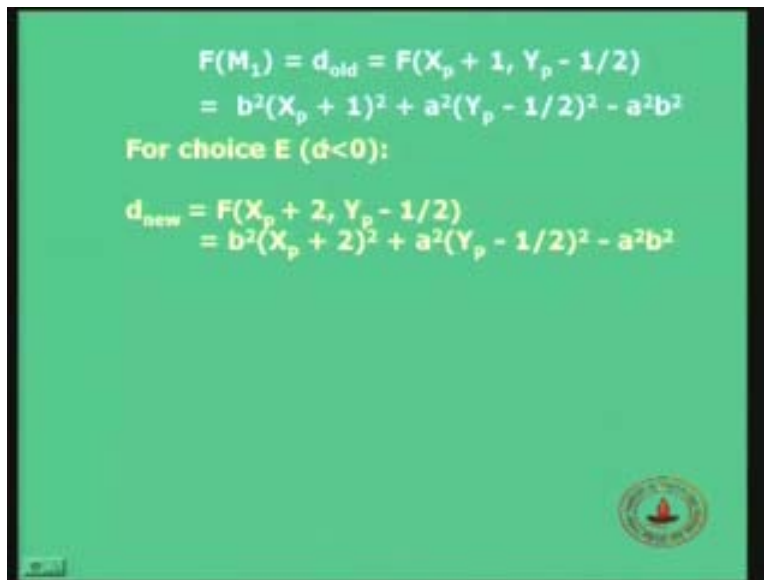
(Refer Slide Time: 00:20:10)

$$F(M_1) = d_{old} = F(X_p + 1, Y_p - 1/2)$$
$$= b^2(X_p + 1)^2 + a^2(Y_p - 1/2)^2 - a^2b^2$$


So $F(M_1)$ is the value of the function F at the next midpoint in region $R1$ which is of d_{old} which is d_{old} . Subscript old indicates the current value or the old value is X_p plus 1 and Y_p minus 1/2, this is because the choice is between east and south east pixel. Since the choice is between east and south east pixel, the next midpoint will be shifted by 1 pixel along X direction and minus 1/2 along Y direction. And if we substitute in the value of F which we all know, you will get this expression for $F(M_1)$ or d_{old} .

So just keep this in mind. This is the d_{old} formula. The old or the current value as you can say for the decision variable d . We were working in region $R1$ and we have two choices. If you have a choice east, that is because that the d is less than 0, the midpoint is basically within the circle. You have to remember the choice is between east and south east pixel and if the curve is passing over the midpoint, the curve is closer to these pixels, so we choose and when the curve is closer to east pixel, the midpoint is below the curve with decision variable we have a negative value. You can easily visualize that because you have done it for a circle, we have done it for the case of a line, we have done it in the previous case also and for ellipse the same criteria for d positive or negative. A Boolean decision is to be taken whether you need to choose east or south east.

(Refer Slide Time: 00:21:45)



$$F(M_1) = d_{old} = F(X_p + 1, Y_p - 1/2)$$

$$= b^2(X_p + 1)^2 + a^2(Y_p - 1/2)^2 - a^2b^2$$

For choice E ($d < 0$):

$$d_{new} = F(X_p + 2, Y_p - 1/2)$$

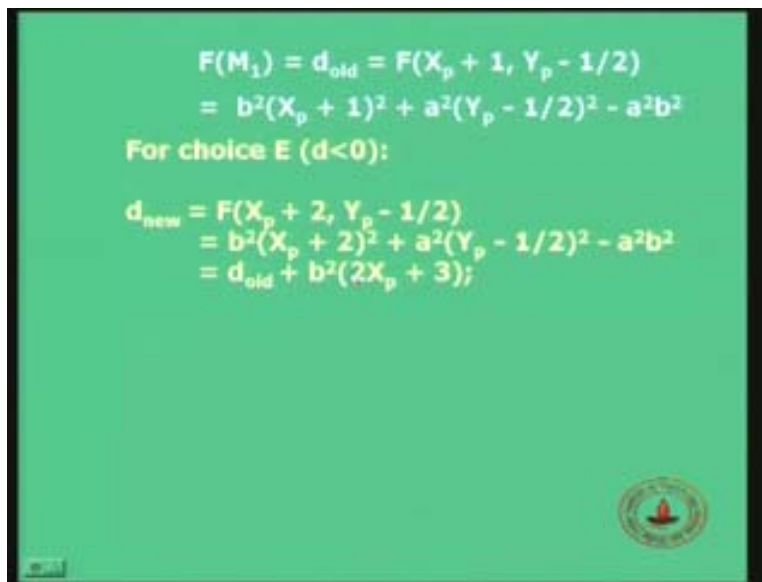
$$= b^2(X_p + 2)^2 + a^2(Y_p - 1/2)^2 - a^2b^2$$

So we know that when d is negative or less than 0 you have to choose east. So for choice east the new value for the decision variable d is given as this, Look at it, it is X_p plus 2, Y_p minus 1/2, because we have chosen east, so the next decision variable will be X_p plus 1 which is already there for next midpoint plus 1 so X_p plus 2.

So as we substitute this X_p plus 2 and Y_p minus 1/2 respective coordinate values into the expression of F , this is the value of d_{new} which you get. You can work it out yourself. Please do not simply copy from the screen which you are seeing. Try to work it out yourself and then I will halt and tell you to find out, if I try to express d_{new} which is given as this expression in terms of d_{old} which is given on the top. As you see the

expressions are almost similar. Lot of terms are matching especially the last terms are exactly same. The difference between these two terms basically comes with the X_p variable. What will be the difference? I can look at d_{new} minus d_{old} , what will be d_{new} minus d_{old} ? If you can evaluate that, then I can write d_{new} as d_{old} plus something else. What will be that something else? If you evaluate you will find out that it will be b^2 multiplied by X_p plus 3. Just find out because you will have $4X_p$ minus 2. So basically you will have $2X_p$ b^2 multiplied by $2X_p$ plus 3. Let us check it out if it is correct or not? That is it.

(Refer Slide Time: 00:23:19)



$$\begin{aligned}
 F(M_1) &= d_{\text{old}} = F(X_p + 1, Y_p - 1/2) \\
 &= b^2(X_p + 1)^2 + a^2(Y_p - 1/2)^2 - a^2b^2
 \end{aligned}$$

For choice E ($d < 0$):

$$\begin{aligned}
 d_{\text{new}} &= F(X_p + 2, Y_p - 1/2) \\
 &= b^2(X_p + 2)^2 + a^2(Y_p - 1/2)^2 - a^2b^2 \\
 &= d_{\text{old}} + b^2(2X_p + 3);
 \end{aligned}$$

So you have d_{new} as d_{old} which is this expression on the top plus this term. This is the additional term which is coming due to X_p plus 2 whole square. So this is the term which will come. See if this is the case I can define my first order difference.


(Refer Slide Time: 00:23:33)

$$\begin{aligned} F(M_1) &= d_{old} = F(X_p + 1, Y_p - 1/2) \\ &= b^2(X_p + 1)^2 + a^2(Y_p - 1/2)^2 - a^2b^2 \end{aligned}$$

For choice E ($d < 0$):

$$\begin{aligned} d_{new} &= F(X_p + 2, Y_p - 1/2) \\ &= b^2(X_p + 2)^2 + a^2(Y_p - 1/2)^2 - a^2b^2 \\ &= d_{old} + b^2(2X_p + 3); \end{aligned}$$

Thus, $(\Delta d)_{E1} = b^2(2X_p + 3);$



If east is the choice in region R1, E and one subscript for the delta d indicates that I am looking at first order difference under the condition that I have chosen east in the first region as b square multiplied by 2X plus 3. Please note down and mark this in rectangular box. This is one important formula which is going to be used in the case of drawing an ellipse. Let us look at the choice.

(Refer Slide Time: 00:23:59)


$$\begin{aligned} F(M_1) &= d_{old} = F(X_p + 1, Y_p - 1/2) \\ &= b^2(X_p + 1)^2 + a^2(Y_p - 1/2)^2 - a^2b^2 \end{aligned}$$

For choice E ($d < 0$):

$$\begin{aligned} d_{new} &= F(X_p + 2, Y_p - 1/2) \\ &= b^2(X_p + 2)^2 + a^2(Y_p - 1/2)^2 - a^2b^2 \\ &= d_{old} + b^2(2X_p + 3); \end{aligned}$$

Thus, $(\Delta d)_{E1} = b^2(2X_p + 3);$

For choice SE ($d \geq 0$):

$$\begin{aligned} d_{new} &= F(X_p + 2, Y_p - 3/2) \\ &= b^2(X_p + 2)^2 + a^2(Y_p - 3/2)^2 - a^2b^2 \end{aligned}$$


If the choice is south east the decision variable is positive or equal to 0. Why? What does this mean? That means between east and south the east midpoint is above the curve. The ellipse is passing in region R1 and the mid point is above the curve and the choice is

between east and south east pixel. So, the midpoint is above the curve, the curve is closer to the south east pixel and hence the south east pixel should be the odd of this choice. And since the point is above the curve, you have the decision variable d at the mid point to be positive greater than or equal to 0.

(Refer Slide Time: 00:24:33)

$$F(M_1) = d_{old} = F(X_p + 1, Y_p - 1/2)$$

$$= b^2(X_p + 1)^2 + a^2(Y_p - 1/2)^2 - a^2b^2$$

For choice E ($d < 0$):

$$d_{new} = F(X_p + 2, Y_p - 1/2)$$

$$= b^2(X_p + 2)^2 + a^2(Y_p - 1/2)^2 - a^2b^2$$

$$= d_{old} + b^2(2X_p + 3);$$

For choice SE ($d \geq 0$): Thus, $(\Delta d)_{E1} = b^2(2X_p + 3);$

$$d_{new} = F(X_p + 2, Y_p - 3/2)$$

$$= b^2(X_p + 2)^2 + a^2(Y_p - 3/2)^2 - a^2b^2$$

Equal to 0, you can choose east or south east, in this case we have chosen it to be south east. And in the case of south east the d new as in the case of east will now be X_p plus 2 that is fine. But since it is south east you have Y_p minus 1/2 minus 1 so you have Y_p minus 3 by 2 that is the expression. Substitute these values into the expression of F . You have this expression b square multiplied by this term, a square multiplied by the Y part, and of course minus a square b square, it is obviously canceling out. Now, I have to again do the same as I have done for east that means I must try to write d new as d old plus some factor.

So subtract d new from d old as you can see now that only one of the term that is a square b square part is the one which will be cancelled out. That term will be having b square multiplied by an X factor and a square also minus Y factor. And what do you get? You get b square multiplied by the same, $2X_p$ plus 3.

(Refer Slide Time: 00:25:40)

$$F(M_1) = d_{old} = F(X_p + 1, Y_p - 1/2)$$

$$= b^2(X_p + 1)^2 + a^2(Y_p - 1/2)^2 - a^2b^2$$

For choice E ($d < 0$):

$$d_{new} = F(X_p + 2, Y_p - 1/2)$$


$$= b^2(X_p + 2)^2 + a^2(Y_p - 1/2)^2 - a^2b^2$$

$$= d_{old} + b^2(2X_p + 3);$$

For choice SE ($d \geq 0$): Thus, $(\Delta d)_{E1} = b^2(2X_p + 3);$

$$d_{new} = F(X_p + 2, Y_p - 3/2)$$

$$= b^2(X_p + 2)^2 + a^2(Y_p - 3/2)^2 - a^2b^2$$

$$= d_{old} + b^2(2X_p + 3) + a^2(-2Y_p + 2);$$


But, a square term will contain minus $2Y_p$ plus 2. Please check it out when you subtract Y square minus $\frac{1}{2}$ whole squared from Y_p minus 3 by 2, whole square. These two terms are important. When you subtract one from the other, the term here is to be subtracted from this term is where you get the coefficient of a square. So I hope these equations you can work it out yourself. Please do it right now and verify and see the first order difference.

(Refer Slide Time: 00:26:11)

$$F(M_1) = d_{old} = F(X_p + 1, Y_p - 1/2)$$

$$= b^2(X_p + 1)^2 + a^2(Y_p - 1/2)^2 - a^2b^2$$

For choice E ($d < 0$):

$$d_{new} = F(X_p + 2, Y_p - 1/2)$$

$$= b^2(X_p + 2)^2 + a^2(Y_p - 1/2)^2 - a^2b^2$$

$$= d_{old} + b^2(2X_p + 3);$$


For choice SE ($d \geq 0$): Thus, $(\Delta d)_{E1} = b^2(2X_p + 3);$

$$d_{new} = F(X_p + 2, Y_p - 3/2)$$

$$= b^2(X_p + 2)^2 + a^2(Y_p - 3/2)^2 - a^2b^2$$

$$= d_{old} + b^2(2X_p + 3) + a^2(-2Y_p + 2);$$

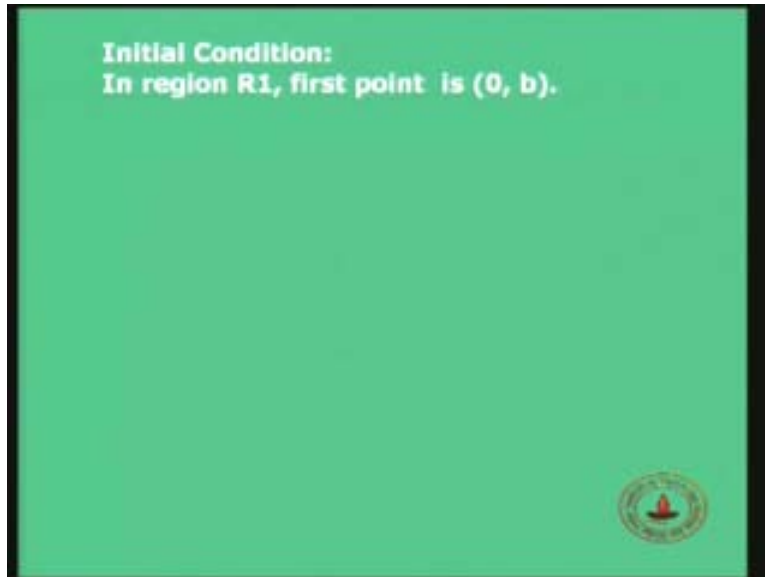
Thus, $(\Delta d)_{SE1} = b^2(2X_p + 3) + a^2(-2Y_p + 2);$



If south east has been the choice and you are in region R1 is given as b square multiplied by $2X_p$ plus 3 and a square. These two formulas are very important. First order difference

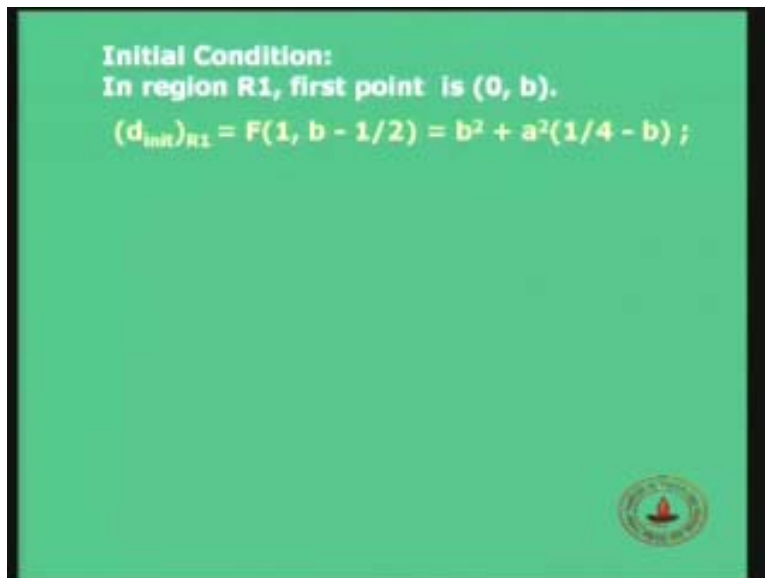
under the condition of east pixel in region R1 and first order difference in the case of south east pixel in region R1. If you remember these two expressions, we can move on to the next slide. I think you will not have any objections, so we move on to the next slide, remembering just these first order differences in region R1.

(Refer Slide Time: 00:26:45)



We need to find an initial condition, so in the region R1, the first point is 0 comma b. So what is d start?

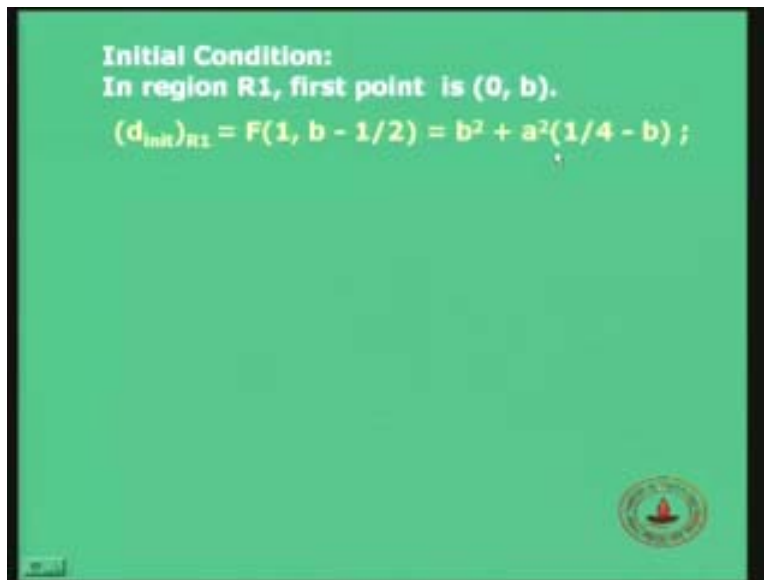
(Refer Slide Time: 00:26:53)



Or in this case we will say as d init is after the first point 0 comma b, the next midpoint will be 1 comma b minus 1/2 and we substitute into d init. This is the value of d init and you get d square plus a square into 1/4.

So, this is the expression of d init and we know the first order differences in region R1 when you choose east and south east. So we can start and go ahead and write the algorithm for region R1. Yes you can do that. But I think we will have to find out the other expressions before writing the whole algorithm for the curve to be drawn on the region R1 as well as in region R2 which will complete the first quadrant. The points should be selected in first quadrant. So we know the condition when you move to the next region. We will see that right again and we will have a new d initial in that region R2.

(Refer Slide Time: 00:27:52)



We have two different choices, so the first order differences also will be the same, but if we look back into the slide minus this is one small point here, that if you are looking for integer based algorithm and the initial value of d itself is a fractional value because we have a 1/4 minus b here. Now only of course if a square is an even quantity it might cancel out otherwise typically you might start with a fractional value here.

Now this is going to be causing a problem. It will slow down when you have floating point arithmetic instead of integer arithmetic. We had this problem in a case of a circle. We had the d start as h minus 1/4 a value of 1 by 4 also came out in the case of a circle. The same thing is coming out in the case of ellipse. What to do? What was the solution we adopted in the case of a circle?

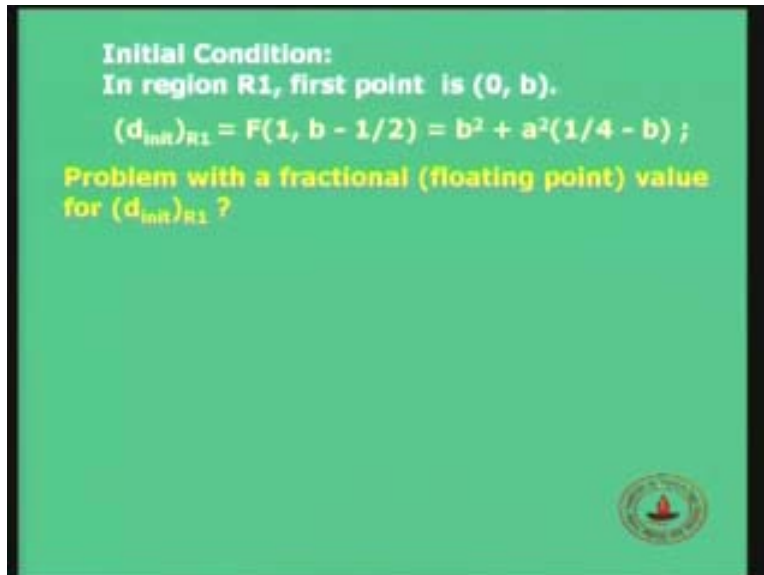
The solution was, we multiplied the entire function F by the existing quantity or changed, pre-minus substituted it, the value of, say, you can use the mechanism which we used in the case of a line where we multiplied the value of function by 2 or 4 as the case may be

or you pre-minus substitute and get a value h is equal to d minus something and get rid of this fraction.

I think in this particular case it would be advisable and easy to look where we multiplied the d init or the function value f by a value 4. If we multiply it by 4 then your d init value will be 4 b square plus a square into 1 minus 4b.

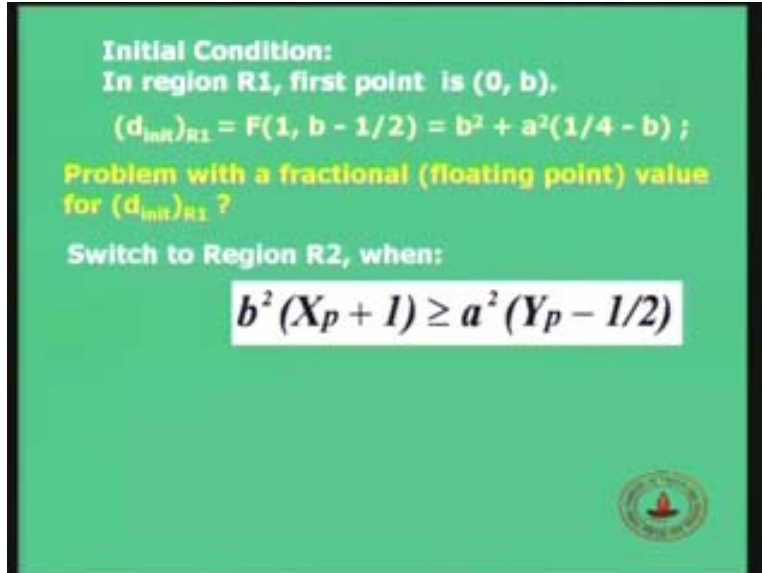
So that becomes, you can start with an initial condition. I will not write that equation but I leave it you as an exercise to workout with the new d init value which is an integer and then also start ahead and find out the first order differences for east and south east with the new value of four times if so. That I leave to you. I am not discussing this, I will go ahead with this set of expressions as it is and also write down the algorithm. The rest of the part, how to make an integer starting from the d init value is left to you. Please make a note of it, that the d init value is a floating point.

(Refer Slide Time: 00:29:45)



So we have a problem with a fractional or a floating point value of d init but by now if you have followed my previous lectures for line drawing and circle where we knew how to draw or where we knew how to start with the integer value in spite of the starting value of the fraction, we know to handle this particular case as well.

(Refer Slide Time: 00:30:12)




Initial Condition:
In region R1, first point is (0, b).

$$(d_{\text{int}})_{R1} = F(1, b - 1/2) = b^2 + a^2(1/4 - b) ;$$

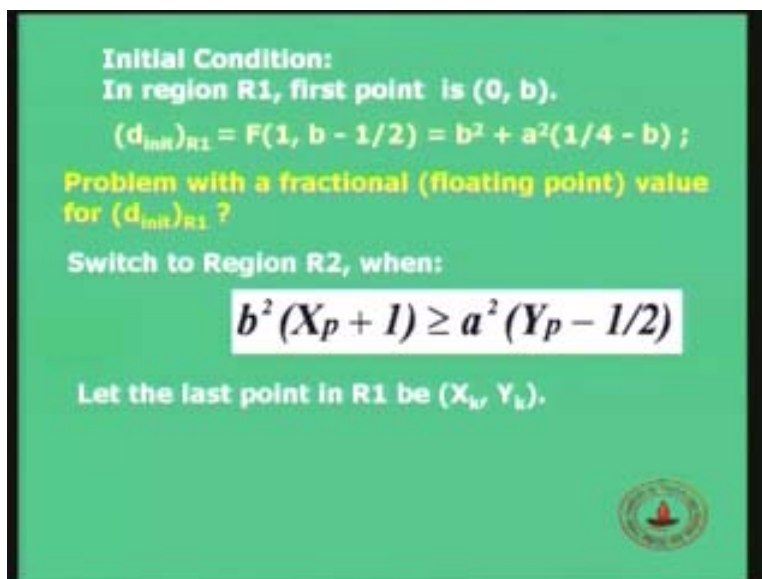
Problem with a fractional (floating point) value for $(d_{\text{int}})_{R1}$?

Switch to Region R2, when:

$$b^2 (X_p + 1) \geq a^2 (Y_p - 1/2)$$


Initial value in region R1 is known. First order differences in region R1 for the choice east and south east pixel are also known. We also know when to switch to region R2. Keep checking this midpoint as long as this condition is not satisfied we are in region R1 and we keep on incrementing, take a decision based on the sign of the value of d and then update using the first order differences for east or south east as the case may be required and then switch only to region R2 when this condition is satisfied. Start from 0 comma b and keep incrementing, keep choosing east and south east, east and south east, one of them, east or south east in fact and then move to region R2 when this condition is satisfied.

(Refer Slide Time: 00:30:58)



Initial Condition:
In region R1, first point is (0, b).


$$(d_{\text{int}})_{R1} = F(1, b - 1/2) = b^2 + a^2(1/4 - b) ;$$

Problem with a fractional (floating point) value for $(d_{\text{int}})_{R1}$?

Switch to Region R2, when:

$$b^2 (X_p + 1) \geq a^2 (Y_p - 1/2)$$

Let the last point in R1 be (X_v, Y_v) .



So let the last point in region R1 it could be an arbitrary point be X_k comma Y_k . It is at this point X_k comma Y_k where we find that this condition of switching to region R2 has occurred and we have now moved to region R2. But the choice, if we remember when we were in region R1 in the first quadrant the choice was between east and south east pixel. The curve was moving almost from a horizontal manner and then it tends to become vertical in the lower part in the region R2 of the first quadrant and the slope is in fact numerically in absolute terms more than 1 and the choice now since the curve is steeping down will be between south and south east pixels.

(Refer Slide Time: 00:31:46)

Initial Condition:
 In region R1, first point is $(0, b)$.

$$(d_{init})_{R1} = F(1, b - 1/2) = b^2 + a^2(1/4 - b) ;$$

Problem with a fractional (floating point) value for $(d_{init})_{R1}$?

Switch to Region R2, when:

$$b^2 (X_p + 1) \geq a^2 (Y_p - 1/2)$$

Let the last point in R1 be (X_k, Y_k) .

$$F(M_2) = F(X_k + 1/2, Y_k - 1)$$

$$= b^2(X_k + 1/2)^2 + a^2(Y_k - 1)^2 - a^2b^2$$

$$= (d_{init})_{R2}$$


So, we need to have a new value of d_{init} in F of M_2 depending upon X_k Y_k . X_k Y_k I must admit here that it is the last point in region R1. So the next point is coming in region R2. But based on these we can actually find out the initial value of d in region R2. And since the choice is between south and south east pixel the Y_k minus 1 factor comes in for the next midpoint or the first midpoint in region R2 and X_k plus 1/2.

Earlier it was the other way round, we had X_k plus 1 Y_k minus 1/2 for region R1. In region R2, it will be Y_k minus 1. We will always decrement it by 1 and X_k plus 1/2, you know all these expressions now, the choice is between the south and south east pixels. So that is the new d_{init} for R2 and in the top we have the d_{init} value for region R1, d_{init} value for region R2 is known, first order difference in region R1 are known as well. The only fact which remains is the first order differences in region R2 in the first quadrant.

(Refer Slide Time: 00:32:57)

$$\begin{aligned} F(M_2) &= d_{old} = F(X_k + 1/2, Y_k - 1) \\ &= b^2(X_k + 1/2)^2 + a^2(Y_k - 1)^2 - a^2b^2 \end{aligned}$$

For choice SE ($d < 0$):

$$\begin{aligned} d_{new} &= F(X_k + 3/2, Y_k - 2) \\ &= b^2(X_k + 3/2)^2 + a^2(Y_k - 2)^2 - a^2b^2 \end{aligned}$$



Now, d_{old} is the $F(M_2)$ which we have derived, which we have seen the expression in the previous slide X_k plus $1/2$ Y_k minus 1 . So it is substitute and break open the expression you get it as this for F X_k plus $1/2$ whole square, a square and Y_k minus 1 comes here so this is the d_{old} .

If the choice is south east again the same condition that means the curve is closer to the south east pixel and d is negative. That means the point is within the curve, the d_{new} value will be X_k incremented by $3/2$ Y_k minus 2 . I think that you can easily find out because if you have chosen south east you would have chosen Y_k minus 1 and X_k plus $1/2$ anyway. So the next X_k plus 1 Y_k minus $1/2$, so the next midpoint will be Y_k minus 2 and X_k plus. So break open that expression and this is what you get as your new d_{new} . Try to express these in terms of d_{old} plus a factor to get the first order difference. Basically to get the first order difference what you have to do is subtract d_{old} from d_{new} , so subtract here, you can see only the constant term a square b square cancel out and you have terms of b square with X_k and a square with Y_k also. Have you worked it out? How much did you get? So it will be d_{old} .

(Refer Slide Time: 00:34:19)

$$\begin{aligned} F(M_2) &= d_{\text{old}} = F(X_k + 1/2, Y_k - 1) \\ &= b^2(X_k + 1/2)^2 + a^2(Y_k - 1)^2 - a^2b^2 \end{aligned}$$

For choice SE ($d < 0$):

$$\begin{aligned} d_{\text{new}} &= F(X_k + 3/2, Y_k - 2) \\ &= b^2(X_k + 3/2)^2 + a^2(Y_k - 2)^2 - a^2b^2 \\ &= d_{\text{old}} + b^2(2X_k + 2) + a^2(-2Y_k + 3); \end{aligned}$$


Just check it out. This is the answer which you should get. It should be d old plus b square multiplied by $2X_k$ plus 2 and a square of minus $2Y_k$ plus 3. This is what is going to be your d new. So if this is your d new which is d old plus this factor it is absolutely simple to write the first order difference.


(Refer Slide Time: 00:34:58)

$$\begin{aligned} F(M_2) &= d_{\text{old}} = F(X_k + 1/2, Y_k - 1) \\ &= b^2(X_k + 1/2)^2 + a^2(Y_k - 1)^2 - a^2b^2 \end{aligned}$$

For choice SE ($d < 0$):

$$\begin{aligned} d_{\text{new}} &= F(X_k + 3/2, Y_k - 2) \\ &= b^2(X_k + 3/2)^2 + a^2(Y_k - 2)^2 - a^2b^2 \\ &= d_{\text{old}} + b^2(2X_k + 2) + a^2(-2Y_k + 3); \end{aligned}$$

Thus, $(\Delta d)_{\text{SE2}} = b^2(2X_k + 2) + a^2(-2Y_k + 3);$



Under the case, you have chosen south east in region R2. Those are the subscript meanings; south east two means region two and south east pixel of the choice. This is your first order difference or delta d under region two. Let us look at the other case.

(Refer Slide Time: 00:34:55)


$$\begin{aligned} F(M_2) &= d_{old} = F(X_k + 1/2, Y_k - 1) \\ &= b^2(X_k + 1/2)^2 + a^2(Y_k - 1)^2 - a^2b^2 \end{aligned}$$

For choice SE ($d < 0$):

$$\begin{aligned} d_{new} &= F(X_k + 3/2, Y_k - 2) \\ &= b^2(X_k + 3/2)^2 + a^2(Y_k - 2)^2 - a^2b^2 \\ &= d_{old} + b^2(2X_k + 2) + a^2(-2Y_k + 3); \end{aligned}$$

Thus, $(\Delta d)_{SE2} = b^2(2X_k + 2) + a^2(-2Y_k + 3);$

For choice S ($d \geq 0$):

$$\begin{aligned} d_{new} &= F(X_k + 1/2, Y_k - 2) \\ &= b^2(X_k + 1/2)^2 + a^2(Y_k - 2)^2 - a^2b^2 \end{aligned}$$


If you are choosing south, the value of d again will be positive or equal to 0 and in that case when you are choosing south the next midpoint will be incremented by $1/2$ along x and minus 2 along Y_k and if I break open the expression for F , this is what I get for my d new. So now what again I have to do? The same thing? Have to express this d new in terms of the d old expression given above plus a factor which will be basically my first order difference under the choice of south in region R2. So subtract and get. You can see that the b square terms completely cancels out here and we are left with the term of d old plus a square minus $2Y_k$ plus 3. So this is the factor difference between d old and d new under the case when you have chosen south and hence minus the first order difference under the case when you have chosen the south pixel in region R2 is given by a square $2Y_k$ plus 3.

(Refer Slide Time: 00:35:46)

$$\begin{aligned} F(M_2) &= d_{old} = F(X_k + 1/2, Y_k - 1) \\ &= b^2(X_k + 1/2)^2 + a^2(Y_k - 1)^2 - a^2b^2 \end{aligned}$$

For choice SE ($d < 0$):


$$\begin{aligned} d_{new} &= F(X_k + 3/2, Y_k - 2) \\ &= b^2(X_k + 3/2)^2 + a^2(Y_k - 2)^2 - a^2b^2 \\ &= d_{old} + b^2(2X_k + 2) + a^2(-2Y_k + 3); \end{aligned}$$

Thus, $(\Delta d)_{SE2} = b^2(2X_k + 2) + a^2(-2Y_k + 3);$

For choice S ($d \geq 0$):

$$\begin{aligned} d_{new} &= F(X_k + 1/2, Y_k - 2) \\ &= b^2(X_k + 1/2)^2 + a^2(Y_k - 2)^2 - a^2b^2 \\ &= d_{old} + a^2(-2Y_k + 3); \end{aligned}$$

Thus, $(\Delta d)_{S2} = a^2(-2Y_k + 3);$



Now it is easy and straight forward for you to visualize that, these are the two first order differences in region R2 and I hope you have noted down and worked out the first order differences when we were in region R2 in the first quadrant. We also had a delta d east 1 and delta d that is the first order difference, south east 1 delta d e 1 and se 1 for region R1 and for region R2 we have delta d south 2 and south east 2 under region two.

So these are the four first order differences which we have for the two different regions. In each region we have a peer of choices. I again repeat region R1 the choice was between east and south east, in region R2 the choice is between south east and east pixel. There were two side pixels with the curve that was basically passing here in east and south east in region R1, in region R2 south and south east the curve was passing through the middle. So you remember that picture with which we started from today.

(Refer Slide Time: 00:37:00)

$$F(M_2) = d_{old} = F(X_k + 1/2, Y_k - 1)$$
$$= b^2(X_k + 1/2)^2 + a^2(Y_k - 1)^2 - a^2b^2$$

For choice SE ($d < 0$):

$$d_{new} = F(X_k + 3/2, Y_k - 2)$$
$$= b^2(X_k + 3/2)^2 + a^2(Y_k - 2)^2 - a^2b^2$$
$$= d_{old} + b^2(2X_k + 2) + a^2(-2Y_k + 3);$$


Thus, $(\Delta d)_{SE2} = b^2(2X_k + 2) + a^2(-2Y_k + 3);$

For choice S ($d \geq 0$):

$$d_{new} = F(X_k + 1/2, Y_k - 2)$$
$$= b^2(X_k + 1/2)^2 + a^2(Y_k - 2)^2 - a^2b^2$$
$$= d_{old} + a^2(-2Y_k + 3);$$

Thus, $(\Delta d)_{S2} = a^2(-2Y_k + 3);$

Stop iteration, when $Y_k = 0;$



So, where do you stop? We started, we had a d init, initial value of d for region R1. We also found out the initial value of d from $X_k Y_k$. After we move from region R1 to R2 we know the initial value of d of region R2. We have to stop the iteration. Stopping the iteration is basically very simple. Then the value of the integer coordinate Y is equal to 0 here, because you keep on decrementing the value of Y in any case in the region R2 all the time and you reach a value when Y is equal to 0 from a positive value.

(Refer Slide Time: 00:37:29)

```
void MidPointEllipse (int a, int b, int value);
```

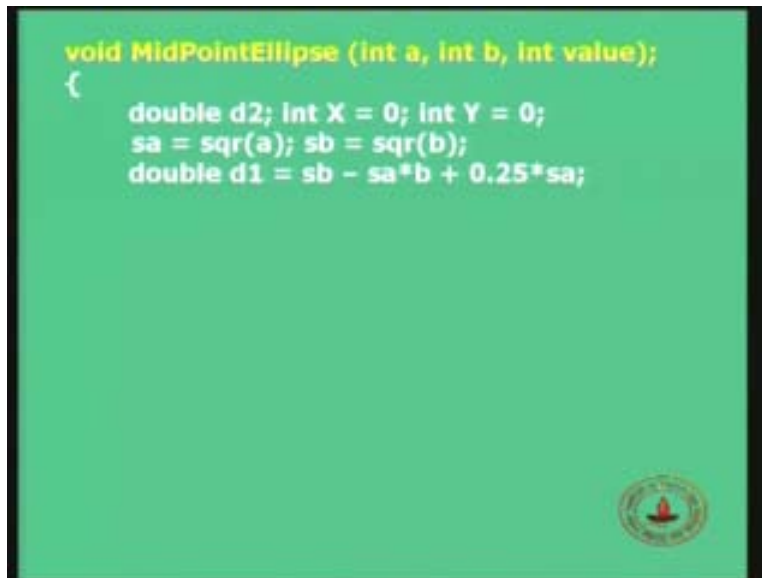


So, let us look at the algorithm in C type of a code for the midpoint ellipse algorithm in terms of when it requires two values a and b , a and b as the major axis parameter and the

minor axis parameter and the value could be the intensity of the beam on the width of the ellipse being drawn or the intensity of the color, it could be the color of the elliptical curve, it could be the width of the curve or it could be the other attributes, drawing attributes like deciding density of a glacier of the color. So here you are passing some value, but the two main parameters for selecting the pixels are nothing but your major axis parameter a and the minor axis parameter b.

(Refer Slide Time: 00:38:17)

```
void MidPointEllipse (int a, int b, int value);  
{  
    double d2; int X = 0; int Y = 0;  
    sa = sqr(a); sb = sqr(b);  
    double d1 = sb - sa*b + 0.25*sa;
```



Therefore, once we know these are the initialization steps necessary before we move on to the iterative method of finding out the points in the two regions. Define a double variable d2, integer value X and Y and you also have a decision variable d2 and d1. d1 is for the region 1, d2 is for the region 2. Now I must admit here that the expressions we got for the first order derivatives and also the initial values are on floating point numbers.


So in this case I am showing you the algorithm with floating point but integer based increments. There are two things which are missing here which I am leaving to you as an exercise. The first thing is change the initial value from the floating point to integer ones. So you know that, we discussed about it just a few minutes back, we have to change the value of a by multiplying it by 4. If there is a factor of 1/4 then that is number 1. **The other part I am leaving to you as an exercise as well.** What we did for a circle? In the case of a circle we saw that the first order differences are not simple constant quantities they are functions of the integer coordinates.

So at each iteration stage now you need to multiply with X and add, so that was becoming costly. So, we moved to a second order difference. So please do that for the ellipse as well when you need it for a circle. I am leaving it, this is as an exercise for you where you find out the second order differences in the case of an ellipse as we did for the case of a circle, separately for region R1 you have to solve and separate kind of differences for region R2 you have to solve. Do not mix them up, solve it. Do whatever

was done in the case of a circle, do that for the ellipse and come up with a new modified version of the ellipse algorithm. What I am going to show now is using the double or the floating point initial value but integer based algorithm based on first order differences.

(Refer Slide Time: 00:40:05)

```
void MidPointEllipse (int a, int b, int value);
{
    double d2; int X = 0; int Y = 0;
    sa = sqr(a); sb = sqr(b);
    double d1 = sb - sa*b + 0.25*sa;
```



So, we have floating point value of d1 and d2 which are the initial values for the decision variable d and d1 is for region R1, d2 is for region R2 which will be evaluated in due course of time, as we move on d1 we start with and we have the other constants sa and sb as well.

(Refer Slide Time: 0:40:20)

```
void MidPointEllipse (int a, int b, int value);
{
    double d2; int X = 0; int Y = 0;
    sa = sqr(a); sb = sqr(b);
    double d1 = sb - sa*b + 0.25*sa;
    EllipsePoints(X, Y, value);
    /* 4-way symmetrical pixel plotting */
```



So, four way symmetrical pixel minus plotting is done by the ellipse points XY value, value could be an attribute for an ellipse, it could be the width of the ellipse, the color or the gray shade, or the intensity of the beam which you want, so that is done. So XY takes care of the other three points, symmetrical pixel plotting and the first condition.

(Refer Slide Time: 00:40:41)

```
void MidPointEllipse (int a, int b, int value);
{
    double d2; int X = 0; int Y = 0;
    sa = sqr(a); sb = sqr(b);
    double d1 = sb - sa*b + 0.25*sa;
    EllipsePoints(X, Y, value);
    /* 4-way symmetrical pixel plotting */
}
```

Remember, as long as this condition holds good we were in region R1, so this sa and sb are defined on the top, a square and b square and Y minus 1/2 and X plus 1. So as long as this is holding good if d1 is less than 0 select east **increment the decision** variable by 2X plus 3 multiplied by sb 2X.

(Refer Slide Time: 00:41:00)

```
void MidPointEllipse (int a, int b, int value);
{
    double d2; int X = 0; int Y = 0;
    sa = sqr(a); sb = sqr(b);
    double d1 = sb - sa*b + 0.25*sa;
    EllipsePoints(X, Y, value);
    /* 4-way symmetrical pixel plotting */

    while ( sa*(Y - 0.5) > sb*(X + 1))
        /*Region R1 */
    {
        if (d1 < 0) /*Select E */
            d1 += sb*((X<<1) + 3);
    }
}
```

This is a **bitwise bit plus** manipulation of X which will make X double. X is an integer anyway. So I can do a bit manipulation. I hope most of you follow the syntax of C otherwise this operator indicates that it is a **bit wise** operation to multiply X by value 2, so that is very simple.

(Refer Slide Time: 00:41:28)

```
void MidPointEllipse (int a, int b, int value);
{
    double d2; int X = 0; int Y = 0;
    sa = sqr(a); sb = sqr(b);
    double d1 = sb - sa*b + 0.25*sa;
    EllipsePoints(X, Y, value);
    /* 4-way symmetrical pixel plotting */

    while ( sa*(Y - 0.5) > sb*(X + 1))
        /*Region R1 */
    {
        if (d1 < 0) /*Select E */
            d1 += sb*((X<<1) + 3);
        else /*Select SE */
            { d1 += sb*((X<<1) + 3) + sa*
              (-(Y<<1) + 2); Y-- }
    }
```

See like notations or select south east if d is positive or equal to 0 and increment d1 by 2X plus 3 multiplied by b square plus a square multiplied by minus 2Y plus 2. So do that and finally decrement Y that you have to do if you are selecting south east and always at the end outside the if then else loop you always increment X because if the choice is between east and south east and you also draw the next ellipse points based on the new XY values.

(Refer Slide Time: 00:41:48)

```
void MidPointEllipse (int a, int b, int value);
{
    double d2; int X = 0; int Y = 0;
    sa = sqrt(a); sb = sqrt(b);
    double d1 = sb - sa*b + 0.25*sa;
    EllipsePoints(X, Y, value);
    /* 4-way symmetrical pixel plotting */

    while ( sa*(Y - 0.5) > sb*(X + 1))
        /*Region R1 */
    {
        if (d1 < 0) /*Select E */
            d1 += sb*((X<<1) + 3);
        else /*Select SE */
            { d1 += sb*((X<<1) + 3) + sa*
              -(Y<<1) + 2); Y--;
              X++; EllipsePoints(X, Y, value);
    }
}
```

So in region one this loop will keep continuing as long as this condition of the while which is put inside that continues then we are in region R1 when it is while, it will come out of this loop and go into region R2.

(Refer Slide Time: 00:42:11)

```
double d2 = sb*sqrt(X + 0.5) +
sa*sqrt(Y - 1) - sa*sb;

while ( Y > 0) /*Region R2 */
```

Thus, we go up to region R2 with the next point X_k comma Y_k . See the expression X_k Y_k which we used earlier and we have the new decision variable for region R2 and that is the function of the last point of the region R1, so from there we know the functional form of the $d_{init 2}$, we have seen that $d_{init 2}$, we have evaluated that point. So use that functional form in the expression form in this algorithm to get the starting value of d in

region2. So that is what is given, d2 is the initial value of the decision variable in region 2, X and Y are the last points in region R1.

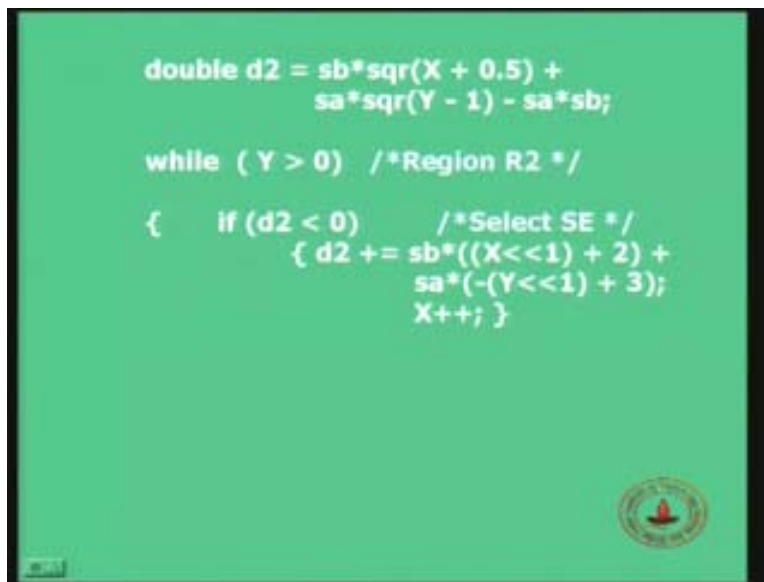
So basically, if I say this is b square multiplied by X plus 1/2 square plus a square plus squared of Y minus 1, of course minus a square b square as usual that will be there. So we put a while loop for region R2 now we have entered and we keep doing this with the condition as long as Y is positive. We have to stop, the terminating condition was Y is equal to 0. So we keep on looping as long as Y is positive.

(Refer Slide Time: 00:43:23)

```
double d2 = sb*sqr(X + 0.5) +
           sa*sqr(Y - 1) - sa*sb;

while ( Y > 0) /*Region R2 */

{   if (d2 < 0)    /*Select SE */
    { d2 += sb*((X<<1) + 2) +
      sa*(-(Y<<1) + 3);
      X++; }
}
```



In this loop we check if the decision variable d2 is less than 0, that is, if it is negative then we are selecting south east and then incrementing the decision variable in region 2 that is d2 by the expression. You have noted down the expression? You can check out the expression given here in a C type of code. It is nothing but b square multiplied by 2X plus 2 and a square multiplied by minus 2Y plus 3 and then of course increment X if you are selecting south east.


(Refer Slide Time: 00:43:59)

```
double d2 = sb*sqr(X + 0.5) +
           sa*sqr(Y - 1) - sa*sb;

while ( Y > 0) /*Region R2 */

{   if (d2 < 0)    /*Select SE */
    { d2 += sb*((X<<1) + 2) +
      sa*(-(Y<<1) + 3);
      X++; }

    else          /*Select S */
      d2 += sa*(-(Y<<1) + 3);
```



In the else condition when you are selecting south minus you do not increment X, but you increment the decision variable d by a square multiplied by minus 2Y plus 3. So it is a square multiplied by, these were just the expressions which you got earlier with whatever was written in this programming in the syntax as given in this code here. We have written this syntax for the case of a line and the circle and we are doing it for the ellipse. Of course we had two versions of the circle. I am giving the version one for the ellipse and I leave it as an exercise for you to get the second order differences and come out with version two ellipse algorithm, so that will be an exercise for you as well.


(Refer Slide Time: 00:44:37)

```
double d2 = sb*sqr(X + 0.5) +
           sa*sqr(Y - 1) - sa*sb;

while ( Y > 0) /*Region R2 */

{   if (d2 < 0)    /*Select SE */
    { d2 += sb*((X<<1) + 2) +
      sa*(-(Y<<1) + 3);
      X++; }

    else          /*Select S */
      d2 += sa*(-(Y<<1) + 3);
```



So, we have the two conditions for selecting south east and east and also incremented the decision variable d and of course out of this if then else. But since we are in region R2, the choice is between south east and east we always have to decrement Y . We always have to decrement Y . Like in the region R1 we were wrongly always incrementing X , in this case we have to always decrement Y when we are working in the second quadrant. And ellipse points will draw the other points based on this. That completes the code.

(Refer Slide Time: 00:45:15)

```

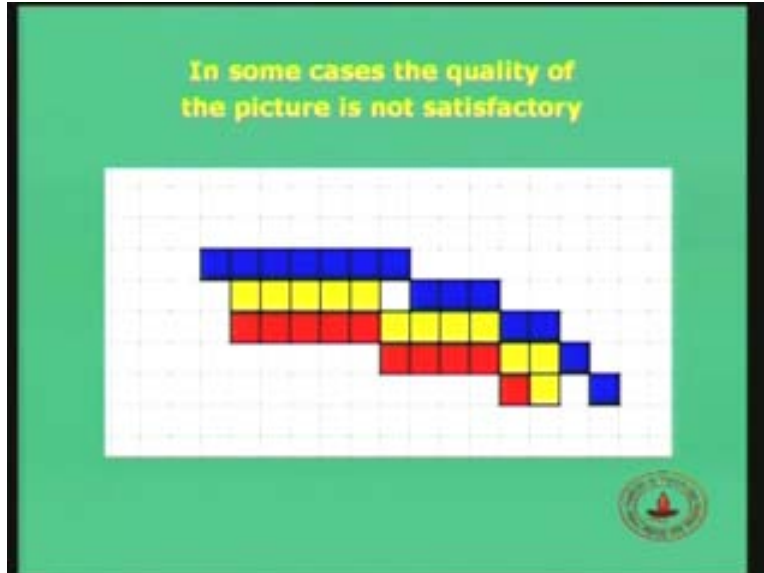
void MidPointEllipse (int a, int b, int value);
{
    double d2; int X = 0; int Y = 0;
    sa = sqr(a); sb = sqr(b);
    double d1 = sb - sa*b + 0.25*sa;
    EllipsePoints(X, Y, value);
    /* 4-way symmetrical pixel plotting */

    while ( sa*(Y - 0.5) > sb*(X + 1))
        /*Region R1 */
    {
        if (d1 < 0) /*Select E */
            d1 += sb*((X<<1) + 3);
        else /*Select SE */
            { d1 += sb*((X<<1) + 3) + sa*
              (-(Y<<1) + 2); Y-- }
            X++; EllipsePoints(X, Y, value);
    }
}

```

I repeat, I will give you the first page of the code which is given here for region R1 along with the initial conditions. If you have anything to note down please note down here and then compare with the expressions which we did together and solve out for the first order differences in the case of east and south east in region R1 and then of course in region R2 we also found out the first order differences for south east and south. So that is what we have all done in terms of the ellipse algorithms. So that completes the method of scan line in ellipse.

(Refer Slide Time: 00:45:47)



We will stop with one small picture in case today where this is an illustration of how things can go wrong in trying to draw an ellipse. Now this factor is related to the problem of the **jaggies** in the case of a straight line as well as in the case of a circle. There are ill effects when you are trying to draw fat lines or fat circles or multiple lines with one after another successive lines with different colors, we had an end point ordering problem if you remember.

So these were the problems which we had in discretised environment. And it can also happen in the case of an ellipse and this is an example which can show you how bad or worse it is. In this case it could be a circle or ellipse it does not matter. What we are trying to do is basically draw a fat circle and this can be done by taking points on an ellipse and expanding not along X and Y, but basically we draw three ellipses or three circles with three different values of a and b or three different values of the various R.

And if we do this the uncertainty which often is that you often may leave certain gaps here. So these are certain things which will be considered in the case when we discuss towards the end of the series of lectures on visual effects and how to avoid the jaggies, aliasing problems, steering effects and discontinuities which will come. We will discuss much towards this in the end but you just keep in mind that in spite of this working with a fantastic algorithm which is going to speed you up, give you very fast results in terms of choosing pixels you will have problems of steering, jaggies, or aliasing, as well as end point ordering and drawing fat lines. So these problems or discontinuities will always be there in discretised environment. This is only an example to show. And in some cases when you draw, you get some results like this, the quality of the picture does not become satisfactory at all.

That brings us to the end of the lectures on scan converting lines, circles and ellipses. And in this section basically what we have done is, we spent the first one or two hours on drawing lines, then of course major on circles where we found out a version one based on first order differences, version two algorithm based on integer differences and then of course we moved to the case of an ellipse where of course I would say we have done only version one algorithm which is based on only first order differences and even the initial value had a floating point problem. And I would request you to take this as an exercise, the first exercise, I will give you a list of exercise problems as take home exercise which you should solve and it also helps you. You should come up with an environment, a programming environment where you can write a small code and see how the points come up and then you plot it in a graph paper or plot it on the graphics screen and see how it comes.

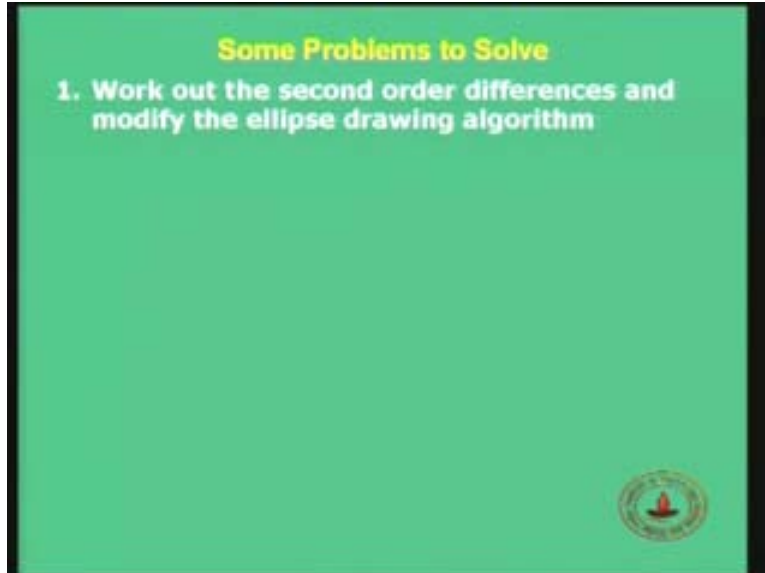
But the first task for now should be, take the ellipse algorithm which we have just discussed today. First you must make two changes, the first is the initial value of d must be made an integer algorithm, an integer value so that the entire algorithm becomes integer based that is number one. Then you should change the version one to a version two by second order differences.

So calculate the second order differences in region R1. So the choice is between east and south east and you say an attempt to see if these second order differences can be used to come up with the completely integer algorithm where within each while loop, you only add integer values but not the function of X and Y coordinates which we have seen just now is the same as the version one circle algorithm is the same as the version one ellipse algorithm where within each loop we are adding the first order differences which are functions of X and Y coordinates.

So we have to do a multiplication addition within each loop which could make the program very slow or the algorithm very slow and over head and within each loop. Either we are talking of region R1 or region R2. Any region, any choice east or south east we have the first order differences to be functions of X and Y coordinates and that is like the version one's midpoint circle algorithm. The second version of the midpoint circle algorithm was based on second order differences which were integer values.

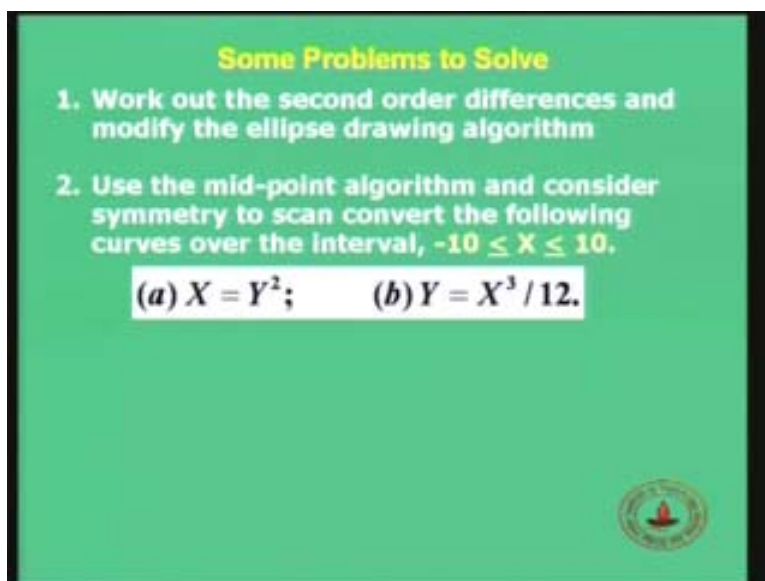
So try to find out if these second order differences in region R1 and R2 are integer values and if they are integer values you can come up with a modified version two algorithm for the case of an ellipse based on the second order differences where within each loop you use second order differences to be which are constants to update the first order differences first and then that is used to update the d loop. Do you remember the version two of the midpoint algorithm for the case of a circle? So attempt that same thing in the case of an ellipse.

(Refer Slide Time: 00:50:21)



I will list you some problems to solve and take home and that the first one which I am talking about is work out the second order differences and modify the ellipse drawing algorithm and come up with a version two. Of course with this the work should also include changing the initial value of the decision variable d to an integer. So do that first and then look at second order differences and see if you can come up with a completely integer based algorithm for ellipse as we have done in the case of a circle.

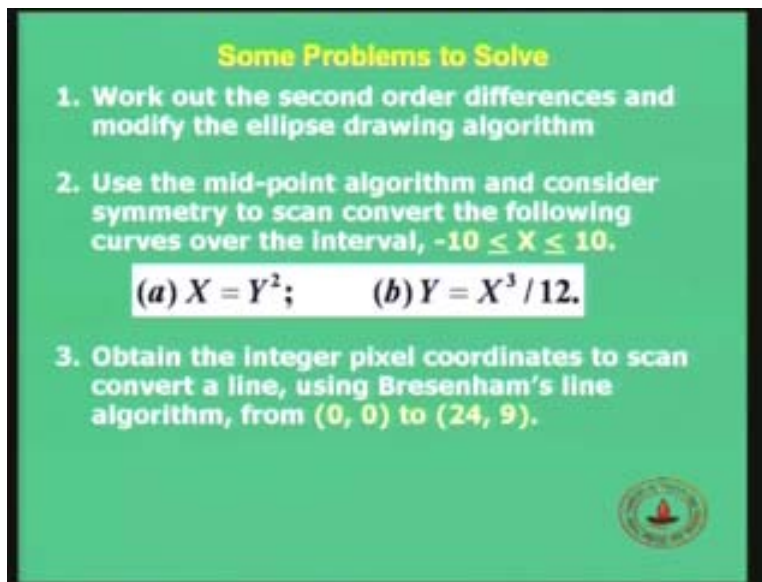
(Refer Slide Time: 00:50:45)



The second task is, use the midminus point algorithm and consider the symmetry to scan convert the following curves in the interval minus 10 to plus 10.

Integer values between plus 10 to minus 10 and two curves are X equal to Y square and Y equals X cube by 12. So these are typical in the case of a parabola or a cubic equation unlike the case of a line or a circle or an ellipse. Here these curves are in fact in the case of a parabola. So try to work it out and see what is the case and you can probably come up with an algorithm to draw a parabola in addition to a circle or ellipse and also draw a cubic expression Y equal X cube by 12. But use the symmetry to find out these curves between minus 10 to plus 10.

(Refer Slide Time: 00:51:29)



Some Problems to Solve

1. Work out the second order differences and modify the ellipse drawing algorithm
2. Use the mid-point algorithm and consider symmetry to scan convert the following curves over the interval, $-10 \leq X \leq 10$.
(a) $X = Y^2$; (b) $Y = X^3 / 12$.
3. Obtain the integer pixel coordinates to scan convert a line, using Bresenham's line algorithm, from $(0, 0)$ to $(24, 9)$.

The third problem is, obtain the integer pixel coordinates to scan convert a line, using Bresenham's line algorithm, from a value 0, 0 to 24 comma 9. If you see the starting point and end point, this line is going to lie in which, it is going to lie in which quadrant? Of course it is going to lie in the first quadrant only 0 0 24 onwards, the slope is more than 1. So it is basically in the second octant in some sense. So you do those manipulations of swapping points and interchanging X and Y values, go back to the algorithm and see what is to be done and find out and implement the Bresenham's algorithm for this particular example.

(Refer Slide Time: 00:52:16)

Some Problems to Solve

1. Work out the second order differences and modify the ellipse drawing algorithm
2. Use the mid-point algorithm and consider symmetry to scan convert the following curves over the interval, $-10 \leq X \leq 10$.
(a) $X = Y^2$; (b) $Y = X^3 / 12$.
3. Obtain the integer pixel coordinates to scan convert a line, using Bresenham's line algorithm, from $(0, 0)$ to $(24, 9)$.
4. Use ellipse drawing algorithm and that derived in problem (1) above, to draw ellipse with parameters: $a = 8, b = 6$.

The last is, minus use an ellipse drawing algorithm, either what we have solved is in first order differences or that which you obtain from the problem one. So use ellipse drawing algorithm and that derived from problem one, that is the algorithm which you derive from problem one. Use both to draw an ellipse with parameters a is equal to given as 8, b is equal to 6. We have taken examples for the case of a line and a circle with certain numerical values and obtain the X Y coordinate points. We did not have time to take an example in the case of an ellipse.

I leave it as an example, as an exercise for you. Please solve it out individually and then just find out the X and Y points. See if they are correct or not, because when you exchange your notes and check up the values then you can tally the results and find out if they are same or not. So a is equal to 8 which is the major axis parameter, minor axis is 6 and then solve and obtain all the points on the ellipse using the algorithm which is based on first order differences, the ellipse drawing algorithm which we have worked out today and also that derived in problem one. So problem one I am saying, obtain the second order differences and get a version two. So use that also to solve and get the points as well as the first order difference in points.

These are the four problems if you are able to work out yourself without much of help. You have understood the lectures very thoroughly and please revise all the equations we have gone through. Revise all the algorithms, try to work out examples given in this lecture slides and talks which I have given and also some others given in the books and also in the exercises. And if you can do that yourself with little bit of practice, you will be quite efficient to solve problems in algorithms for scan line drawing, drawing circles and ellipses. Thank you.