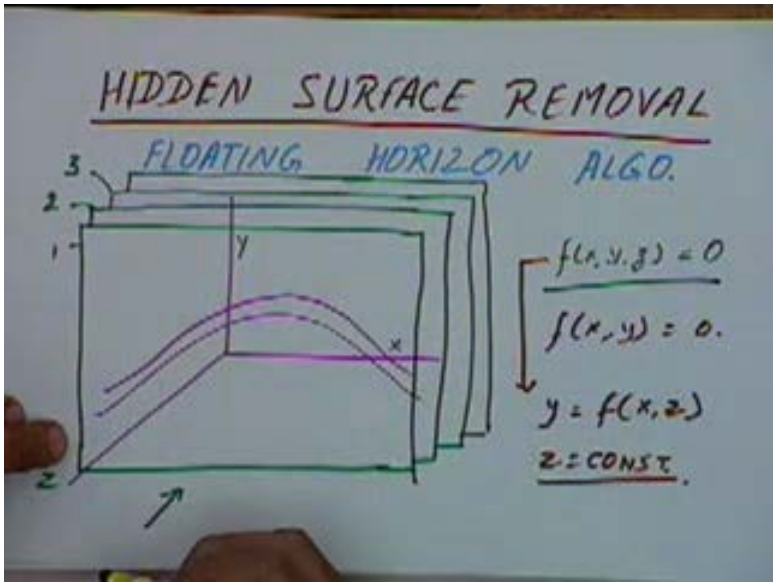
Computer Aided Design
Dr. Anoop Chawla
Department of Mechanical Engineering
Indian Institute of Technology, Delhi
Lecture No. # 14
Hidden Surface Removal (Contd.)

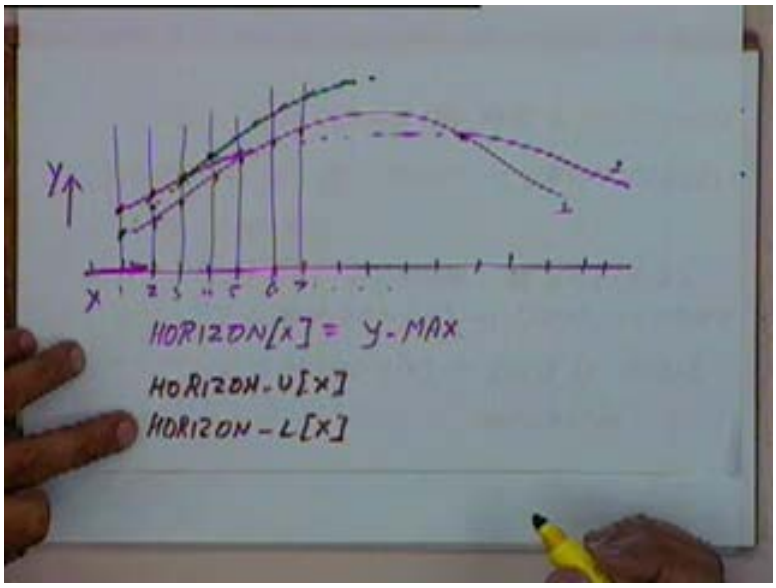
Today we will be talking of the floating horizon algorithm for hidden surface removal and if you remember last time I had mentioned that if you have a curve or if you have a surface of the type f of x, y, z equal to zero then we can use this floating horizon algorithm. If you look at this coordinate system and we are looking at the curve from the z direction let's say. what we will do is we will first take planes parallel to the xy plane and the first plane that we will take will be closest to the eye.

(Refer Slide Time: 00:01:11 min)



If I am looking at from this side, this is the first plane that we will take. The next plane will be just behind that, the one after that will be further behind. If you have a surface given by this, we can get the equation of a curve as that, as we will see it in each successive plane. If I see the intersection of the first plane with this surface I might get a curve of the form, a curve something like this. This is the curve for the intersection with the first plane then you take the intersection with the next plane, the one that is slightly behind. If the next intersection comes something like this, we can draw that again and this way we will keep taking successive planes and their intersections with this surface.

(Refer Slide Time: 00:02:32 min)



Now if you have a situation in which the first curve that we draw is a curve like this and the next curve that you get intersects this curve. That means first it goes like this and intersects the curve like this and like this and goes on further. In this situation if you look at this figure, we are looking at the surface from the front. Since we are looking at the surface from the front, if the curve for the second plane is cutting the curve for the first plane that means the curve for the second plane is at a lower height. So if this curve is at a lower height in that case we will get this curve to be intersecting as we were getting here. And since it is at a lower height that means this portion of the curve is going to be hidden. This way we will keep drawing these curves one after another and the portion which are hidden will come as or will come at a lower y value. This is in the xy plane, x and this is y. In the xy plane the curve will come at a lower height, this is a curve from the first plane, this is the curve from the second plane and this is the basic principle we will use in the floating horizon.

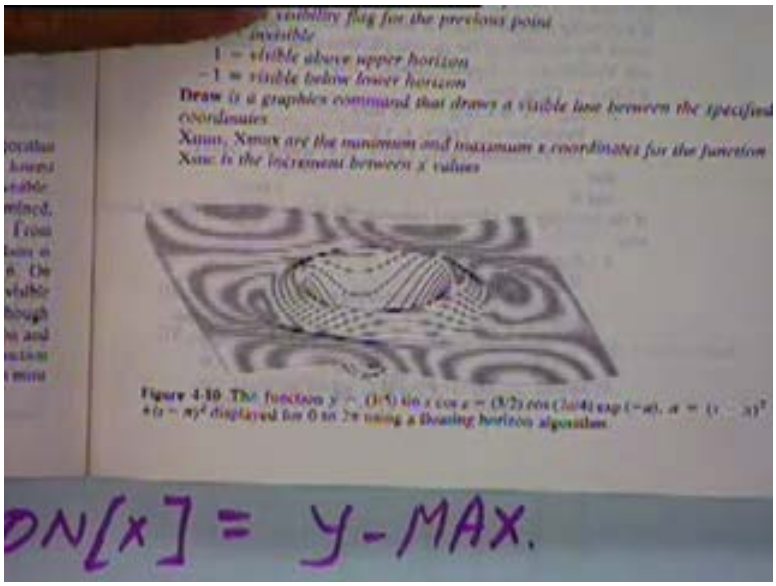
What we will do is along the x axis we will let's say this is my x axis, I will divide this into portions like this. We will let's say each of this mark corresponds to on the next pixel or whatever resolution we choose. So let's say at each x value we can compute the y value that has been displayed, that y value in some sense this is the level of the horizon that we are seeing right now that means as we are going for successive planes, we took the first plane then we took the second plane behind and the third plane behind that and so on. That means we are slowly shifting the horizon backwards and as we are shifting the horizon backwards, the current y value at each point that will tell us the current position of the horizon.

If we get any entity which is above this horizon that will be visible, if any entity is below this horizon it will not be visible. Think of if you are looking at a set of hills, you have a hill in front and if you have another hill behind it, you can see that only if its height is higher than the height of the first hill that is exactly what we are doing here. So we will let's say store a, maintain an array called horizon of x and this will store the current y max value that means let's say if this is and so on.

Let’s say the horizon at one will store the current y max value which is this. Horizon of two will store the y max value which is this. The horizon at five will store the y max value here and when I am drawing the next curve let’s say my next curve is like this. Then the y value of the curve at this point is lower than the horizon at this point, so this curve is hidden at this point. Similarly the y value for the curve here is lower than the horizon of two, so here also this curve is hidden. At this point the two are same, so the horizon will be retained and here the y value is greater than the horizon, so I will update the horizon to this value. In this portion the curve will become visible and so on. So this way we will be maintaining an array called horizon which will store the current highest curve visible or current highest curve that has been displayed.

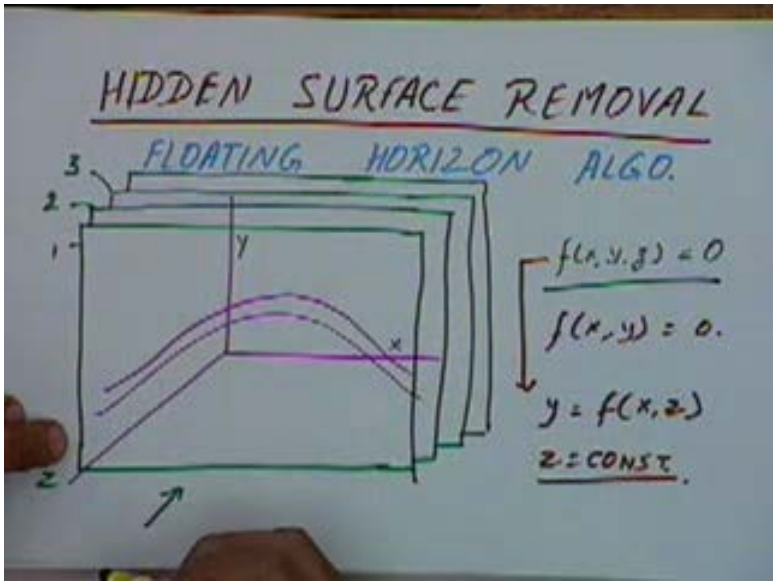
In any curve which is displayed after that will be displayed only if y max or y value is greater than the y max value stored at that point, so let’s write down this algorithm. Again. See if you have this one single surface to be displayed then possibly all these curves will have the same color or intensity. If we have a number of surfaces then you can have them have different colors for that. So in this case right now I am talking of only one single surface and all these curves are obtained by the intersection of the single surface. I have a complicated surface like this and I am taking an intersection with the vertical plane, so I will get in every plane I am getting only one curve right now. What I am saying is that the why we will come to know the item the curve is taking from different planes if we look at just a simple continuous curve. The each curve will look like a continuous curve but we will have a set of curves one curve, the next curve and so on.

(Refer Slide Time: 00:09:07 min)



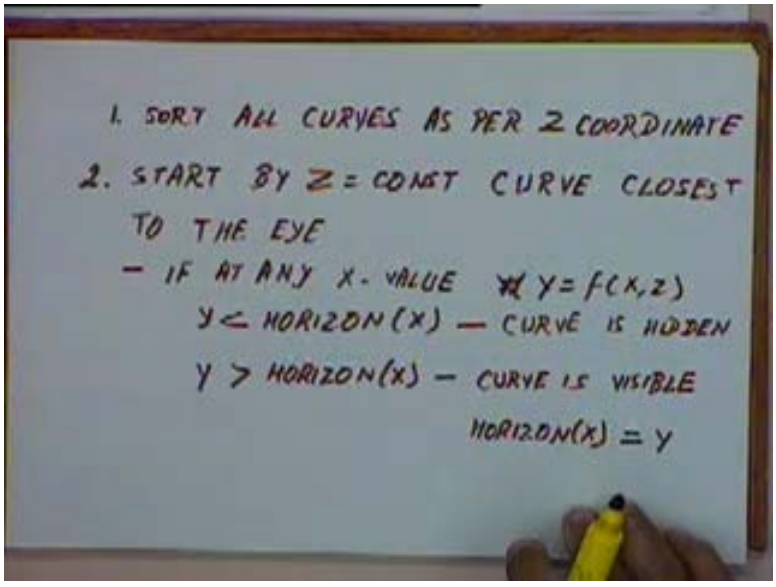
Are you able to see this figure? See if you look at the first curve that has been drawn this curve, this is the intersection with the first plane. The second curve that has been drawn is the intersection with second plane and so on. So this is the kind of effect you will have. Is that okay? Let’s go back to the algorithm and come comeback to the figure later on. The first step is that we have taken a set of planes and for each of these planes you have got a curve, that curve is of the form of this.

(Refer Slide Time: 00:10:07 min)



Essentially what we are doing is we are rewriting this as y is equal to f of xz because we want to compute the y value for the different curves at different x and z locations and then we are taking z equal to constant curves. So we are getting a family of curves and we have to display this family of curves or this set of curves.

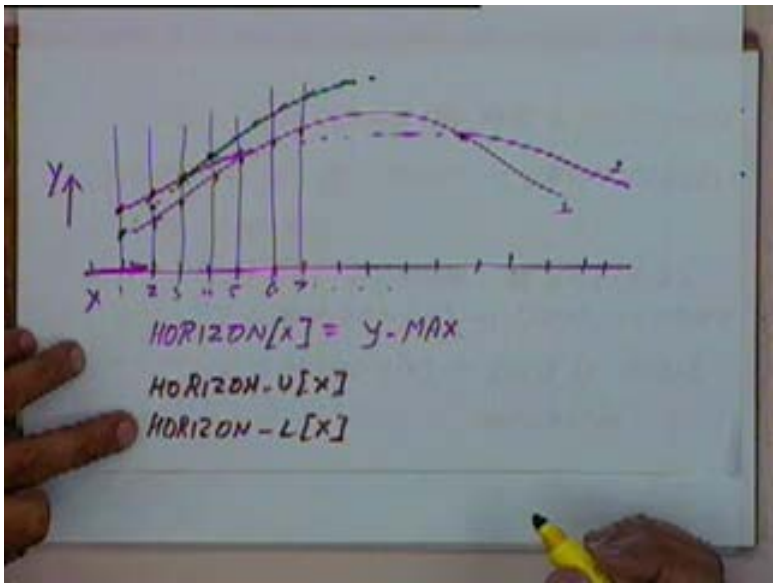
(Refer Slide Time: 00:10:53 min)



So first thing we will do is sort all curves as per the z value or as per z coordinate. The one closest to the eye will be the first curve that we will consider and the one farthest from the eye will be the last curve then excuse me sir. What do you mean by sorting these curves, you don't

have any set of curves do we, I mean this could be. We will arrange them in the decreasing z order in this case, that's all. This is the part of the program that, I am giving the algorithm for that. So first step would be that all the curves that you are getting, you have to treat them in the decreasing z order. The first one closest to the eye then the one after that and so on. Start by the z equal to constant curve closest to the eye and h. Now we have taken the first curve that is the one that is closest to the eye and for this curve, for the first curve we examine the y value at each successive locations and we will compare this y value with the current value of the horizon.

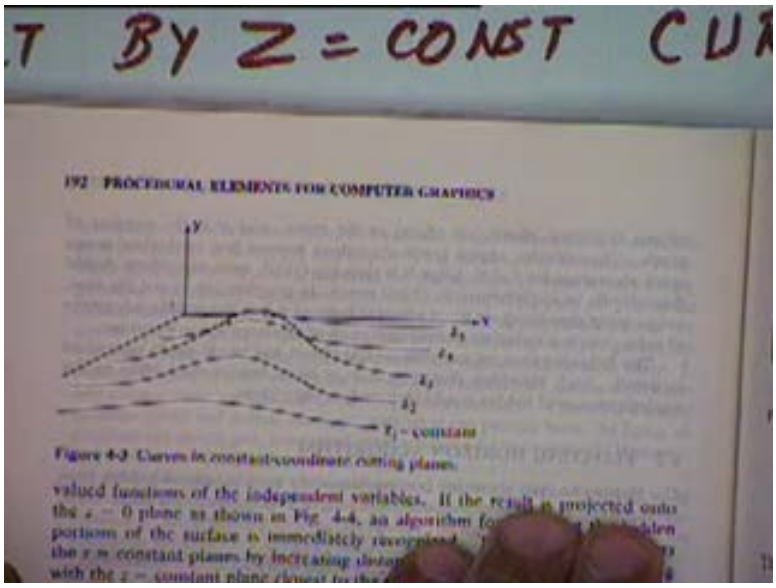
(Refer Slide Time: 00:13:02 min)



We will compute y equal to F of xz that is the y value at the current x location and if this y value is less than the horizon then the curve is hidden and if y value is greater than the horizon of x then we will say that the curve is visible, we will display the curve and we will update the horizon. let's say horizon of x will be equal to y. shouldn't be at every value of x totally highest as well as the lowest point. I am coming to that, I am coming to that. Right now I am just, you think of at the moment you think of this surface, let's say this is my surface you think of it as a solid surface with the complete solid under it. If it is just a planar surface by planar I mean just a sheet kind of surface then what will happen is the surface might be going underneath like this and you might be able to see the surface from under it, for that I will modify it accordingly, I will come to that.

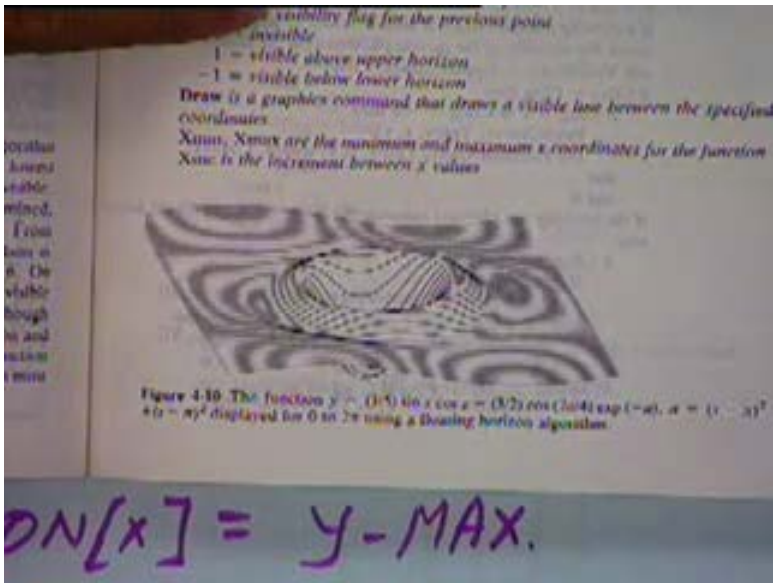
So if at any x value we compute the y value and if the y value is less than the horizon of x then the curve is hidden. If the y value is greater than the horizon of x and the curve is visible and we will update the horizon of x to y. we are doing it for all the x values and we will repeat it for all the curves successively one after another. So by this we will be able to display all these curves one after another and we will get a 3 D effect the way we just saw in the figure. Look at this figure.

(Refer Slide Time: 00:15:45 min)



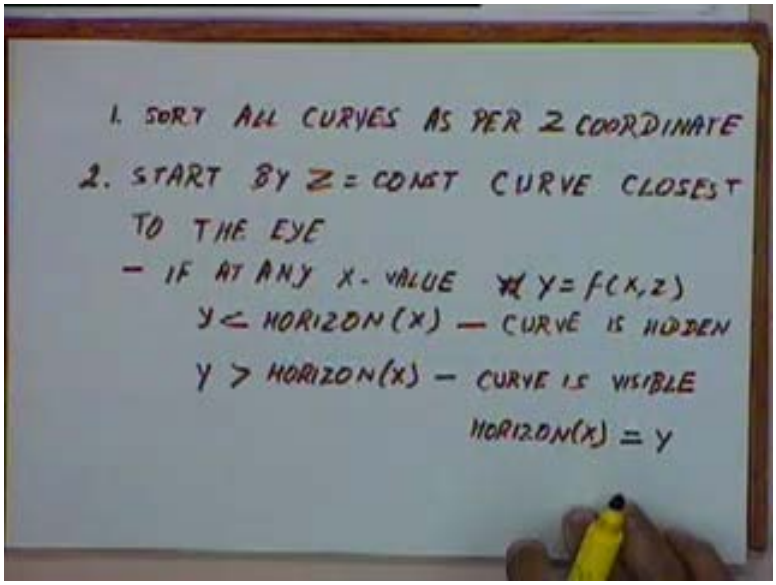
The z_1 equal to constant at the first curve then you have z_2 equal to constant, z_3 equal to constant and then at z_4 equal to constant a part of the curve is going underneath that has been shown as the dotted line that is this part of the curve. And again z_5 equal to constant is again going dotted. So this way by taking successive planes, we are able to get a three dimensional kind of effect by showing the back portion of this hillock as dotted lines and as someone said in this hillock, we are assuming that this is a solid surface that means it is complete solid under it. If it is not, if it is a thin surface in that case this surface may go underneath like if you see it from front, the surface may go underneath like this and from there you will be able to see the lower portion of the surface that we have not been able to incorporate right now. like the previous figure that you have shown. That was the previous figure that had been incorporated.

(Refer Slide Time: 00:17:07 min)



This has been incorporated in this figure. How is that incorporated I will just explain that. Incorporating that what we do, in this right now I am maintaining an array called horizon of x.

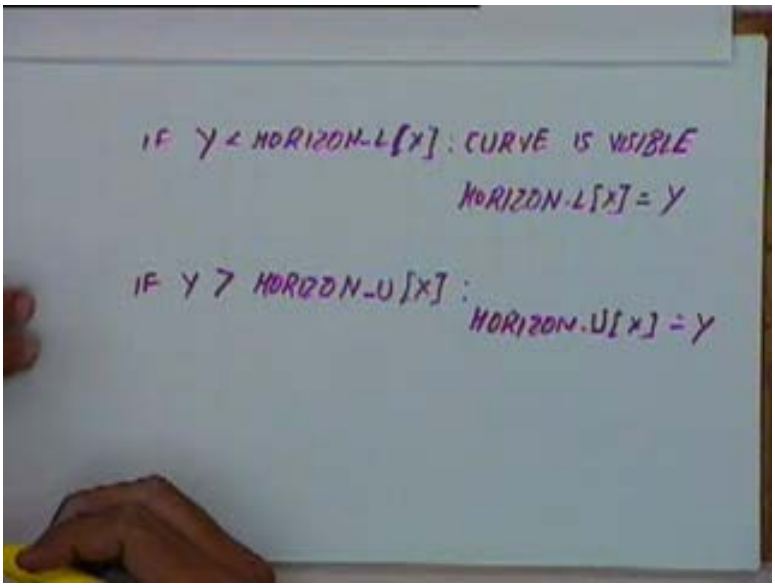
(Refer Slide Time: 00:17:23 min)



Now this horizon is essentially the upper value of y that is visible. We are assuming that everything is solid under it but if it is not then we also need to maintain the lower value of y which is visible. So if I maintain the lower value also that means I maintain two horizons, horizon upper of x as well as horizon lower of x and if my curve happens to go below the horizon lower, let's say if you are looking at a surface from front if any curve happens to go below the lower horizon, this is the my lower horizon right now if this part of the surface goes below that then you will be able to see it from front. So we will maintain a horizon lower as well

as a horizon upper and how do we take care of that? In this algorithm we will in this step and what we will do is let's say if y is less than horizon lower of x.

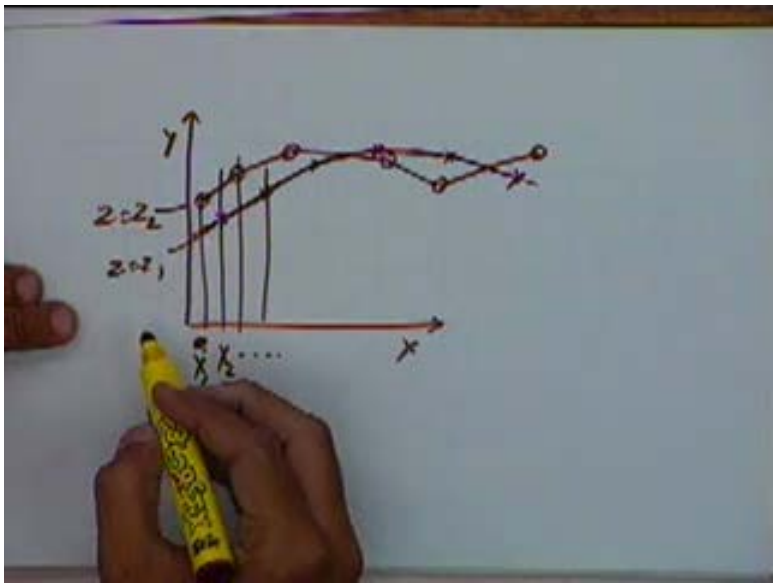
(Refer Slide Time: 00:18:51 min)



Then also the curve is visible and we will update horizon lower and if y is greater than the horizon upper then of course curve is visible and we will say horizon upper of x will be equal to y. So we maintain two arrays, one is horizon lower and the other is horizon upper, by doing that we will be able to display thin surfaces which can also become visible from underneath. Any questions on this algorithm so far? Sir, this is only for rays parallel to z axis. This is only when the viewing direction is at z axis, I am looking at it from the z equal to infinity direction. We can divide from z axis slightly, not exactly. In the figure, from this figure why? This is the z axis in this case.

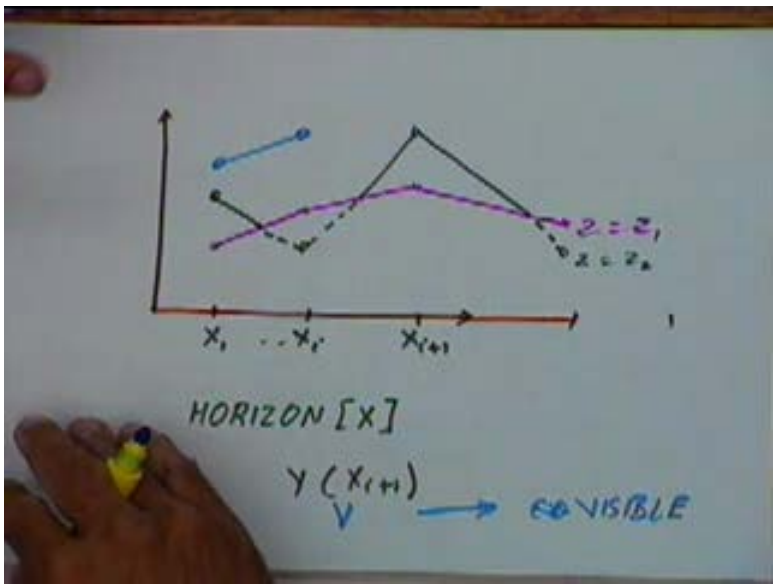
See whatever my viewing direction be, I can always rotate my coordinate planes so that the viewing direction becomes the z axis, you have seen transformations. So whatever my viewing direction be, I can always rotate my coordinate system so that the viewing direction becomes z axis and then I will apply this algorithm. Whenever we are talking of these hidden surfaces algorithm, we normally take the viewing direction to be the z axis. If you are actually looking at the object from some other angle, we will rotate our coordinate system such that the viewing direction will coincide with the z axis. If I am looking at the object from this side then I will change, I will make this my z axis I will rotate my coordinate system accordingly and then redefine the plane. So any questions on this algorithm so far? Now this algorithm we have so far been applying for cases where the surfaces can be represented in this mathematical form that means my surface is a, can be represented by a mathematical equation like this. Often what happens is surfaces are not represented in a mathematical form but they are represented by interpolation.

(Refer Slide Time: 00:22:11 min)



That means if you look at this xy plane then my curve, earlier I was taking my curve to be a smooth curve like this. Instead of this my curve might be a set of points and between these points I might be interpolating. That means this curve or this surface is available as a set of discrete data and not in the form of a mathematical equation. If this curve is not a mathematical equation like this but it is a set of discrete data points then this algorithm might not be directly useful because let's say my one curve is like this, now the next curve is again a set of straight lines which might be something like this. These are my data points for the next curve. So typically what can happen is that at some discrete points, my surface has been disguised that means I would let's say some value x x_1 x_2 so on.

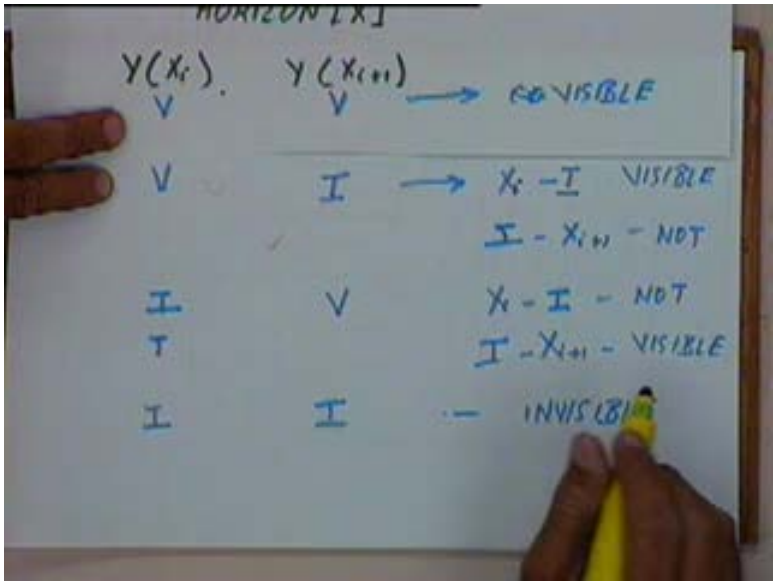
(Refer Slide Time: 00:24:44 min)



And each of these value of xi's, our surface is known that means for the first surface that is this is the z equal to z_1, this is z equal to z_2. For each of these z values, the y values are known as discrete points. For a case like this we can use the same algorithm with a very minor modification that instead of considering the y value at each xi's, we will consider only at the data points. I mean show it in a different figure. The z equal to z_1 curve is given by these set of data points. The z equal to z_2 curve is given by these points in green. In this case we cannot draw a straight line like this or like this. What we will have to do is between these two points, we will interpolate and wherever the intersection lies we will draw a straight line up to that point and then the rest would be dotted. Similarly you continue from here like this and this would be dark.

Similarly in this portion you go like this and this would become. We will carry out an interpolation and while interpolating, we find the intersection of this line and the intersection of this line and up to the intersection point, the curve will be visible and either after that or before that we will decide whether it is invisible or visible. So what we will do is these are the points x_1 let's say this is x_i and this is x_{i+1} and we will maintain in the horizon at each of these xi's. So we will maintain the horizon at x and when we have displayed a curve or a set of curves, the next curve when we are displaying let's say for this point and for the next point we will compute y at x_i and y at x_{i+1}. We will decide whether the curve is visible at y at x_i and at x_{i+1}. If the curve is visible at both the points that means and let's say if we have a data points like this, both these points are visible then the complete curve is visible. So if this is visible, this is also visible then complete curve is visible.

(Refer Slide Time: 00:28:12 min)



If at x_i the curve is visible and at x_{i+1} the curve is invisible. In that case from x_i to the intersection point, the curve will be visible and from the intersection point 2 x_{i+1} the curve will be invisible, will not be visible. On the other hand x_i is invisible and x_{i+1} is visible, it will be the other way around from x_i to I to the intersection point, the curve will not be visible and from I to x_{i+1} the curve will be visible and of course if both of them are invisible in that case the curve is invisible, just one minute. But if our surface is available in the form of a set of data points then

our algorithm will become much more complex not only because I have to compute the intersections but also because let's say after the first intersection is computed, my curve is becoming like this. That means earlier my data points I was taking them at this level and at this level and after this intersection my data point will also have to be taken here because the horizon in this portion can be found out by linear interpolation, between this portion also it can be found out by linear interpolation but I cannot just take the horizon here and the horizon here.

So I have to add this to the data point and then proceed from that point. So if I have, if the surface is available to me in the form of data points then the algorithm becomes slightly complex but it can be taken care of. You were saying something. Normally that would be the case, if it is given as random values then the algorithm will become even more complex. Then you have to take a set of planes and compute the value of curves that at all those x values. Right now we are assuming that it is available at a set of data points. Any other question on this algorithm, on this floating horizon algorithm? in that case I think I will stop here now. After the hidden surface removal is concerned that is all that I am going to cover. From the next lecture onwards we will be talking of finite element method and that basically will be talking of how to compute stresses and displacements for any given object using the finite element approach. So we will start from that in the next class.