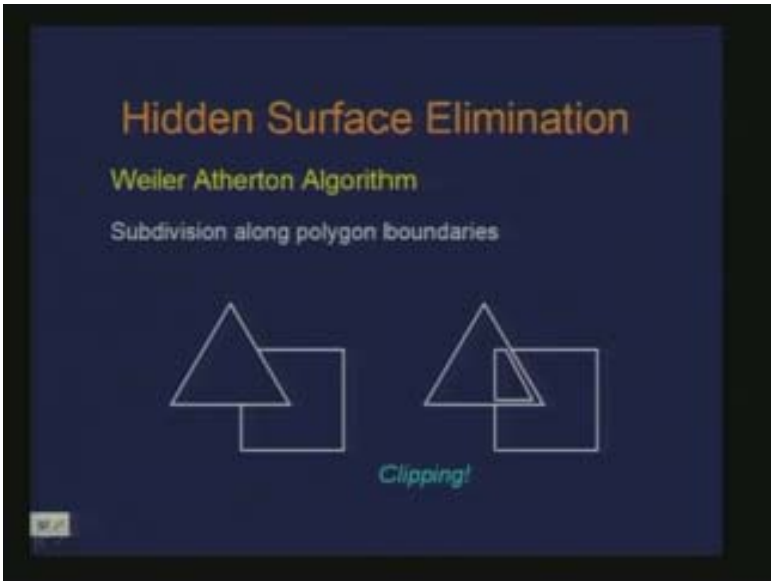
Introduction to Computer Graphics
Dr. Prem Kalra
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi
Lecture - 30
Hidden Surface Elimination (Contd..)

We have been talking about hidden surface elimination methods. Last time we were talking of a method based on area subdivision. The idea was that you can take an area which is to be displayed region of the screen and then you determine the visibility for that area and you try to figure out whether one of the simple cases get satisfied within that area of interest.

We looked at four cases where a polygon could be a surrounding polygon to that area, it could be an intersecting polygon to the area, it could be contained in that area or it could be disjoint in that area. So we try to classify the relationship of the polygon with respect to the area and then decide what to do. And if the case is not a simple case then we subdivide it further. So everything happens in an image space. The area is a displayable area which is in the image space so the computation happens in the image space.

Today we are going to look at another approach which is also of the kind in area subdivision. But instead of subdividing the area which is to be displayed we would subdivided the polygon. The area subdivision where the area was subdivided into four parts was the technique suggested by Warnock, it was the Warnock‘s method.

(Refer Slide Time: 04:28)



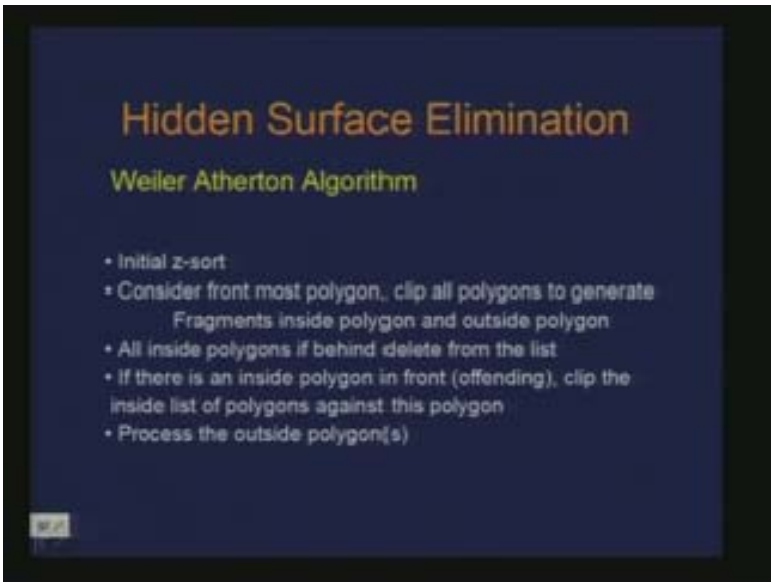
There is another technique by Weiler Atherton where the subdivision takes place along the polygon boundaries. If you have these polygons to be displayed where one of the

polygon obscures part of the other polygon then what we are trying to say here is that the subdivision is in terms of the boundary of the polygon where this polygon obscures this. In some sense it is a clipping method.

We are trying to discard or consider a piece of the polygon, so this is the subdivision taken place for this polygon where you get this part and this part along the boundary and this polygon. After these two fragments or pieces are obtained you figure out whether to display that fragment or not depending on depth of the polygon. So the subdivision or the fragmentation is along the boundary of the polygon. Here the precision where you are doing the subdivision is of object space.

Now what we are saying for the algorithm here is that we first do an initial sort in z or depth and consider the front most polygon clip all polygons to generate pieces or fragments inside and outside polygon. So there is a list for inside polygon and there is a list for outside polygon against the polygon you have considered for clipping and this polygon you have picked on the basis of your initial z sort the basis of which could be anything.

(Refer Slide Time: 06:57)



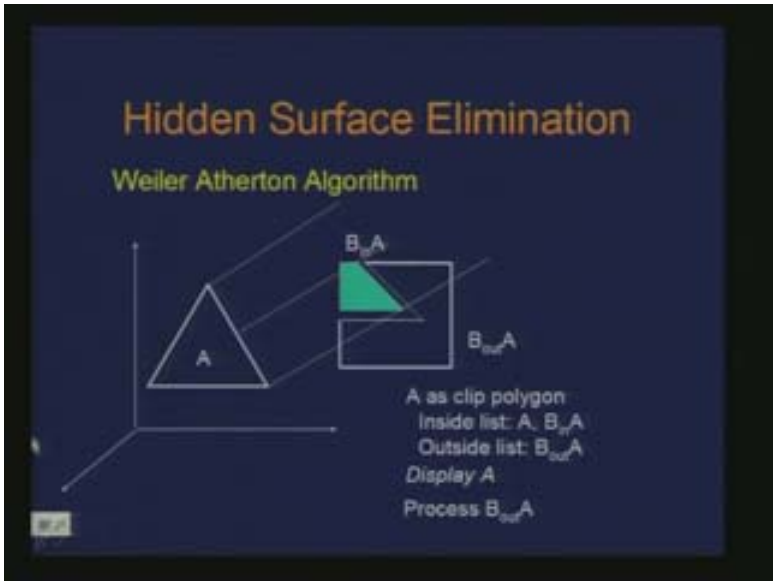
If you find out that all the inside polygons are behind this clipping polygon then you delete those polygons from the list to display because polygon pieces which you have gathered turned out to be behind this polygon against which you are clipping then you need not worry about those polygons so you just discard them.

However, there could be instances where there is an inside polygon in front of this clipping polygon so that is an offending polygon because we had considered an initial z sort on some basis which may not hold true in the real scenario because this could be based on maximum z of the polygon and that may get violated by the sum of the polygon to decide whether it is in the front or behind. So, if you do find offending polygon or

offending polygons then all you have to do is consider that to be the clip polygon and do this process again. So clip the inside list of polygons against this polygon and then you process the outside polygons.

There is a notation of building the inside list and there is a notation of building an outside list. so first you process the inside list just determining the visibility of the polygons within that list and if it turns out that the inside list is all obscured and occluded by this clipping polygon then you just discard them and consider the outside polygon. Just to illustrate through an example; consider a very simple scenario; where you have a polygon A here and this is polygon B and this is the z axis where you are determining the depth of these polygons. So this is (x, y, z) and you are viewing somewhere there from this side. This is in front of that as you see in this situation.

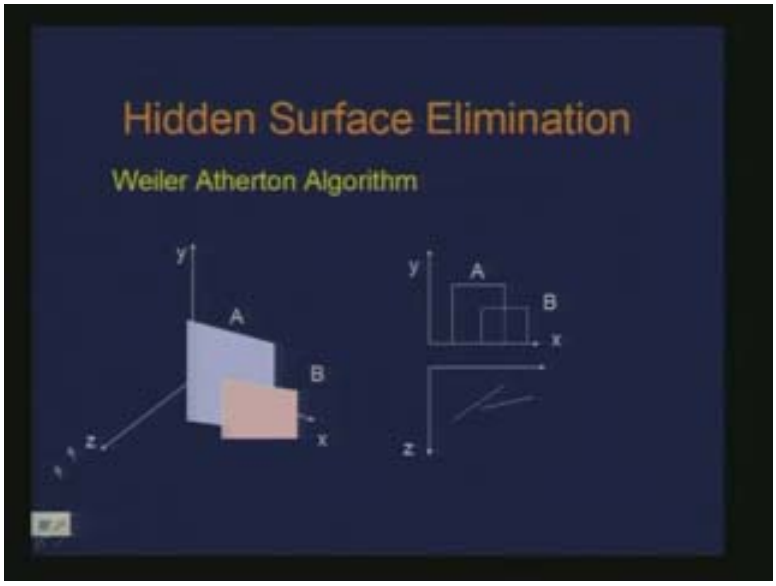
(Refer Slide Time: 10:06)



When the initial sort is taken A turns out to be the first polygon to be picked as the clipping polygon so you consider A as clip polygon so you build the inside list where you have this clip polygon A itself and then there is a portion of B which is inside A so that is the portion which you would find inside A which is in the inside list and then you have the outside list which is this part of the polygon of B which is B out A. Here it is being labeled as B in A the portion of the polygon B which is inside A when you perform clipping and B out A is the portion of the polygon B which is outside A.

When you look at the relative visibility between A and B in A which is the content of the inside list you find out that A is in front of B in A or B in A is behind A therefore you just retain that and just display A. And then you process the outside polygon which is B out A and here it turns out to be a trivial case because this is the only polygon left. Therefore you display B out A. Hence, in some sense the whole method relies on subdividing these polygons into pieces or fragments and establishes the visibility of these fragments with respect to the polygon which you consider in front. You keep doing that.

(Refer Slide Time: 10:54)

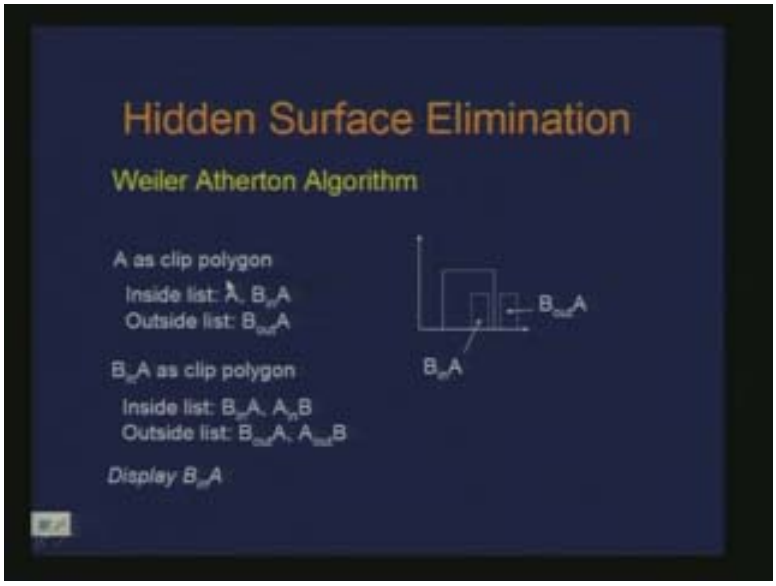


Here is another example where you have the polygon A here, there is another polygon b here so when you look at it in this view this is a, this is b so in z when you see it this will be polygon A and this will be polygon B. Now when you do this listing of inside list and outside list what you will observe is the following. First of all you consider A as the clip polygon because if you look at the z max of A it is in the largest value therefore you consider A to be the clip polygon just on the basis of your initial z sort.

You consider A as the clip polygon so you build the inside list which contains the clip polygon itself first A and then the portion of the polygon B which is inside A which is B in A and then the contents of the outside list just as B out A the remaining portion of the polygon B. Now what happens when you look at the visibility of B in A? With respect to A it turns out that B in A is not behind A but it is actually in front of A so I cannot discard that. Therefore I choose B in A as the clip polygon as opposed to the other scenario where I had B in A behind A and I could easily delete the fragment B in A from the list and display A. But here I cannot because this turns out to be in front of A. Now I pick this as the clip polygon and then the inside list is B in A itself and A in B and that is the portion of A in B.

In fact while I am performing the clipping though strictly speaking I am referring to the clipping against the piece of the fragment B in A but for the purpose of performing clipping of A I could perform the clipping with respect to B itself to determine A and B. So I do not need to perform this clipping against the fragment. This is the inside list.

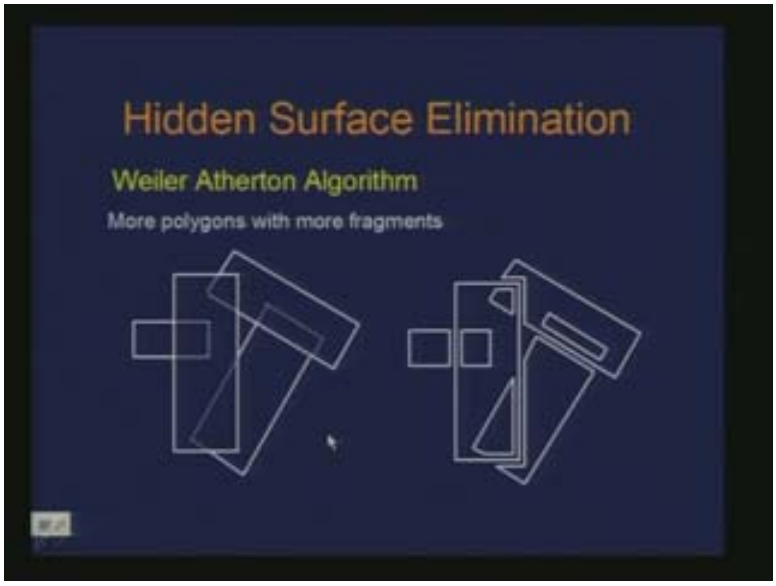
(Refer Slide Time: 14:42)



The outside list is the B out A which is already there and A out B with respect to B in A. Now I figure out that A in B is behind B in A again I establish through the visibility test. Therefore I display B in A then I continue this process. This is how the Weiler Atherton algorithm works. There are more number of polygons therefore there is more fragmentation and more fragments here so all it boils down is to determined these fragments which may look like this and then look at the relative visibility of these fragments. So at the end you will have some display of the fragments. It will be an order of display of these fragments.

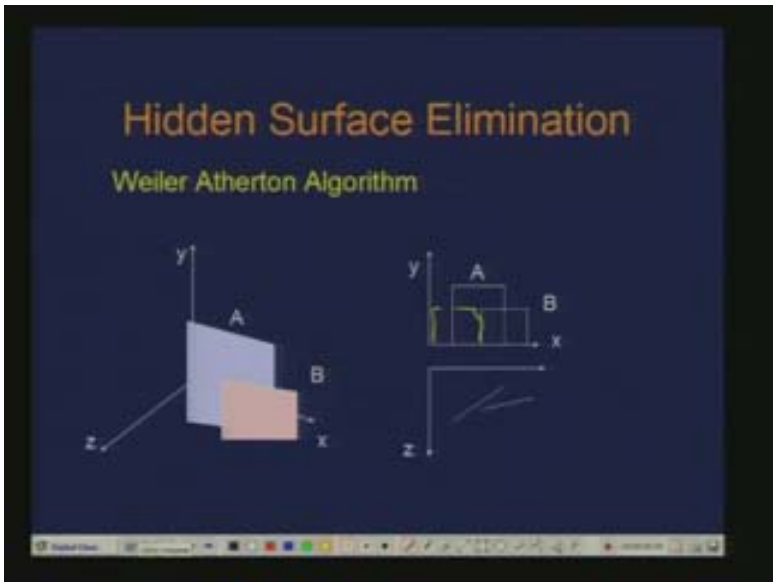
We will move onto another method of hidden surface elimination. It would very much depend on the amount of fragmentation that can happen. The worst case could be that given a polygon here, I am trying to establish this is in a polygon A and all the polygons render or give me the fragments to be tested where I am basically testing with respect to each of those fragments. This can happen for all the polygons so it could be the n square.

(Refer Slide Time: 16:56)



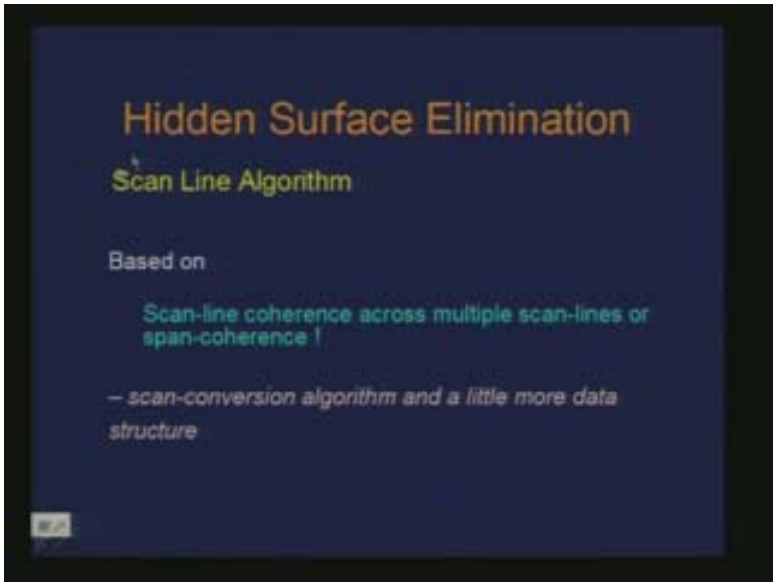
Now there is a situation for instance in this case what we observe is that this polygon is in front of this so there is no interpenetration of polygons, things are little hard when this polygon goes and comes beyond this. Then you may actually do the split of the polygon in 3D and then do this process. If you look at this it will be somewhere and there will be a line here so this line would not appear in the projection. This may actually extend up to this and this line is not there. What you need is to get this line and you would get that through the split of the polygon.

(Refer Slide Time: 19:16)



Here the issue is that what you would get for B. So how would you determine the pieces as B in A and B out A. There is an ambiguity here unless you have this line.

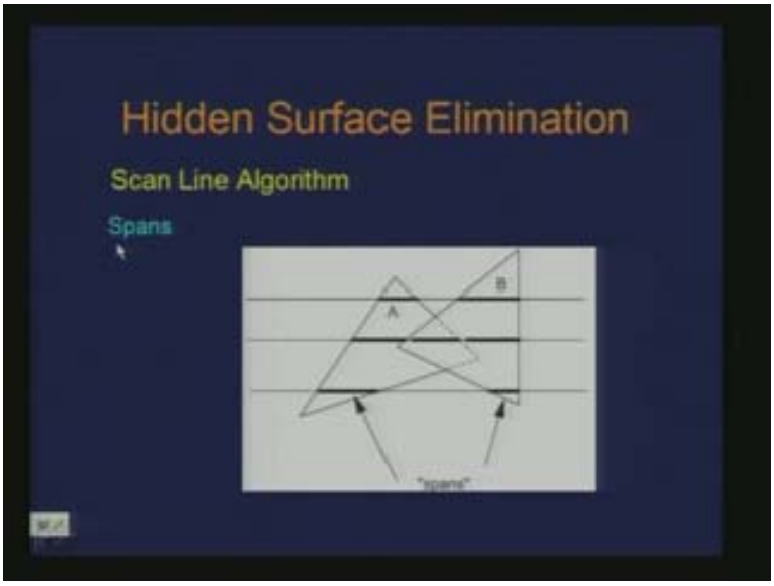
(Refer Slide Time: 20:07)



Here is another method of hidden surface elimination which is actually a combination of z buffer and scan line conversion of polygons. It is a scan line algorithm basically and it is based on scan line coherence across multiple scan lines or span coherence. When you scan convert a polygon then you get span or portion of the scan line within different polygons. For a given scan line you will have a span of this scan line contained in different polygons.

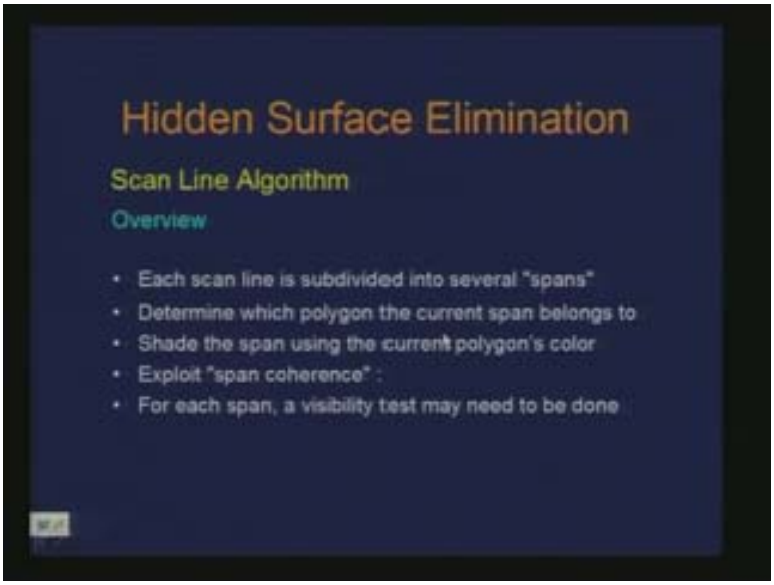
So all you are trying to do is take those spans and figure out which polygons need span around two and if they just belong to one then the visibility is determined. So you just color with the polygon. It is just sort of embedded in the process of scan conversion of the polygon.

(Refer Slide Time: 22:07)



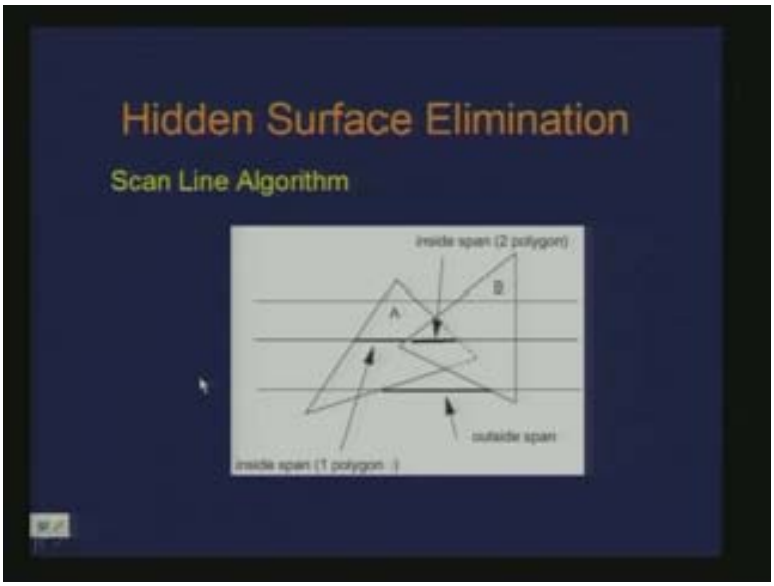
First of all we are referring to these spans. What do we mean by spans? They are nothing but a portion of the scan line. So this is a scan line here, this is a span here, this is also a span but outside the polygons and this is also a span inside another polygon. These pieces of scan line inside or outside the polygons are the spans and these are projections. Again you have span here, span here, span here and so on. So ultimately when it comes to figuring out the visibility then we are looking at the visibility for this span, visibility of this span and visibility of that span and appropriately decide how to render these spans. Overview of this method: Each scan line is subdivided into several spans.

(Refer Slide Time: 23:36)



Determine which polygon the current span belongs to, shade or color the span using the current polygons color or shade. So, in a sense we are actually exploiting the coherence within this span. Once we establish that this span is contained in a particular polygon and this is the polygon which is to be displayed then all the pixels or all the points along that span could directly be rendered.

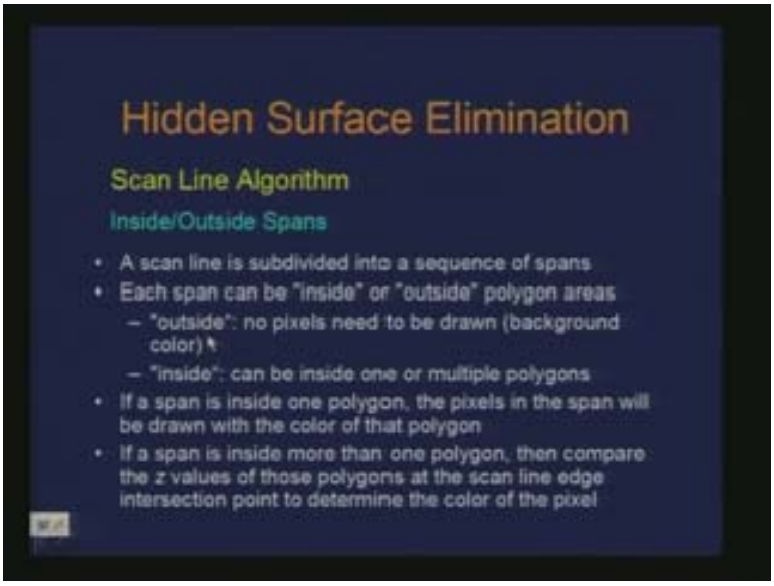
(Refer Slide Time: 24:50)



So in some sense we are exploiting the span coherence and for each span as we observed a visibility test may be required as we observed particularly when there are more than one polygon which are intercepted the span. You may have different kinds of spans. There is a span which is inside only one polygon. This is the polygon A and this is the polygon B and this is the span. For this particular scan line this span is contained in one polygon.

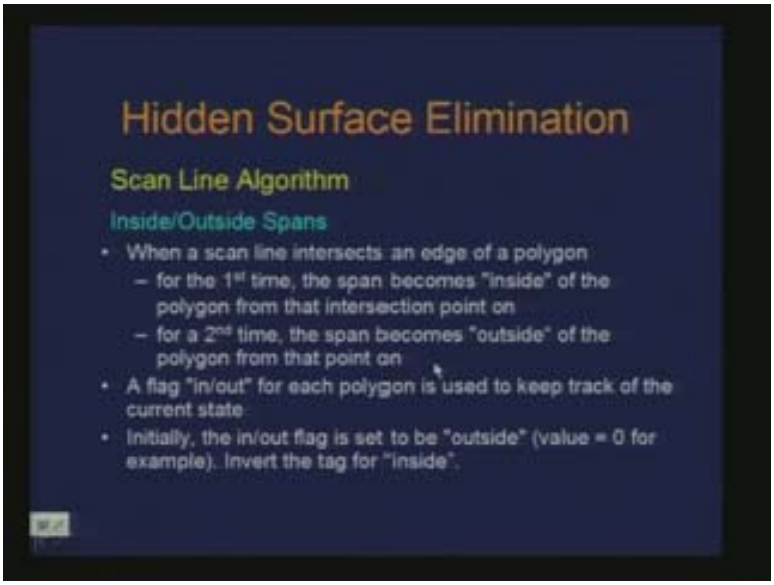
There is this span which is also inside this span. There are spans of the kind; inside span and outside span. Inside span are those which are contained in some polygon and the outside spans are spans which are not inside any of the polygons. So a different strategy may be required depending on whether the inside span is within one polygon or more than one polygon. Therefore how you determine the outside span is again this scan line is divided into sequence which may be an inside span or outside span.

(Refer Slide Time: 27:08)



So the outside span is that no pixel needs to be drawn or you draw the background color because it is not contained in any of the polygons. The inside span can be either inside one polygon or more than one polygon.

(Refer Slide Time: 30:07)

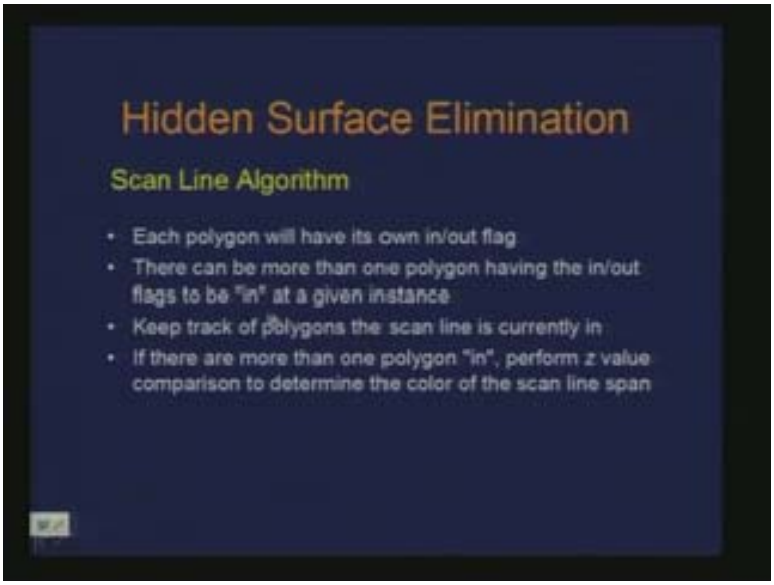


So if a span is inside one polygon the pixels in the span will be drawn with the color of that polygon and if a span is inside more than one polygon then you need to resolve for the visibility which may be through the z value of the polygons. So the question is if you are talking about this labeling of inside outside spans or inside one polygon inside multiple polygons how do we do that?

The visibility test would be required in terms of resolving for z values where we have figured out that this particular span is contained in multiple polygons. So just by labeling inside span and outside span how would you find this out?
Intersection will not give you polygon edges.

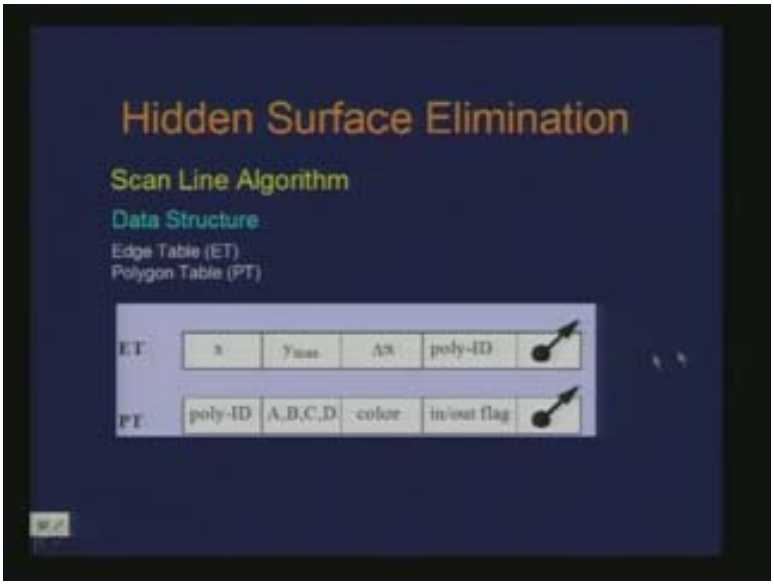
You need to have some sort of a parity check to be able to decide which span is inside which span is outside so it is very much similar to what we have seen in this scan conversion. So, if any scan line intersects the edge of the polygon for the first time the span becomes inside of the polygon from that intersection point onwards, for a second time the span becomes outside of the polygon from that point onwards. This is what parity test means. You just count the number of edges it intersects in the polygon and then decide whether it is again getting in or getting out and then put a flag to mark in or out for each polygon to track the current state. Initially you may label all these polygons to be outside and then do this check.

(Refer Slide Time: 31:05)



We are putting the polygon inside, we are labeling a polygon for this particular scan line as to find out what are the inside polygons. And the track of that would actually determine for a particular span whether this span is contained in this polygon or more than two polygons and so on. Here is an example, what we are saying is that now each polygon has got its own flag which is saying whether it is the in flag or out flag and now there could be more than one polygon having the flags as n. So all that we are saying is that we just have to keep track of the polygons where the scan line is currently in. And if there is more than one polygon in then we need to perform the z value thing resolving the visibility for those polygons.

(Refer Slide Time: 32:55)

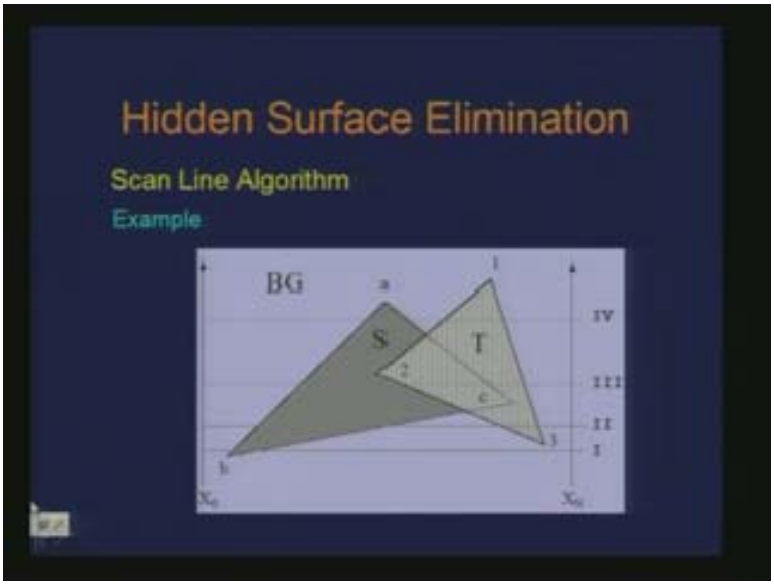


As far as the data structure is concerned which was sort of a sketch of the algorithm we had, so for implementing this you may have a data structure which contains an edge table and a polygon table. The edge table has got the value of x for the lowest y and then you have the y max for the edge, you have the increment of delta x which could be in fact the inverse of the slopes which tells you what increment to give to get to the next point and the idea of the polygon which the edge is contained in so it is an identification of the polygon which give you the fields in the edge table.

For the polygon table you have the idea of the polygon, the definition of the polygon or the plain of which the polygon is made so these are the coefficients of A_x plus B_y plus C_z plus D the definition of the plane of the polygon the color or the shade it has and the flag in and out. Therefore all these will be maintained.

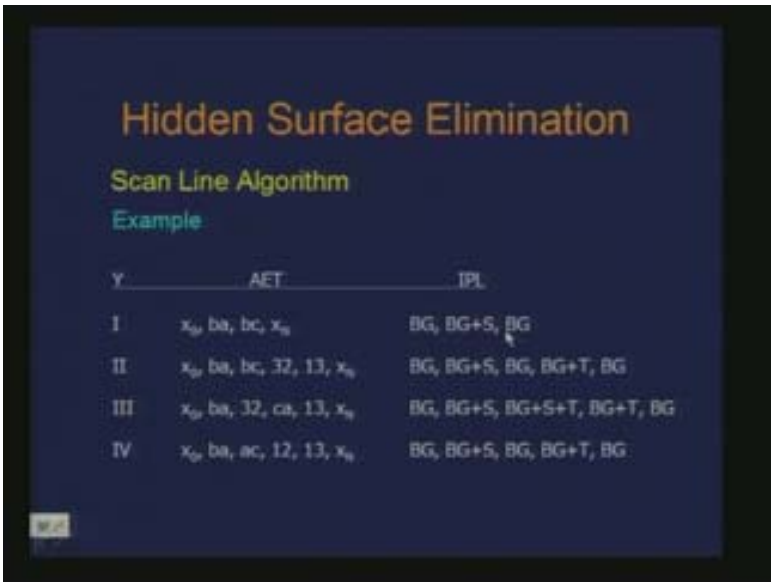
In addition to this we may have some sort of an auxiliary data structure which is in some sense active in polygon list which is just an indication of what are the polygons which are marked in and similarly there is an active edge table. Actually it is either you process that polygon table or you just have an auxiliary data structure which contains this in polygon list. So it may not require an explicit storage it is just an operation. So this active edge table contains the edges of the polygon which are under process for a particular scanner.

(Refer Slide Time: 35:07)



Here is an example; this is the scenario BG stands for background, there is a polygon S and there is another polygon T and you artificially define some extend of the range of the scan line so this could be sort of vertical edges like x_0 and x_n just to give you the finite limited view of the display area. These are the scan lines we are interested in 1 2 3 4. So, if you look at the various scan lines here you have the active edge table for this particular scan line where all it contains are these edges, you have the in polygon list which is giving you the polygons which are intercepted by this scan line.

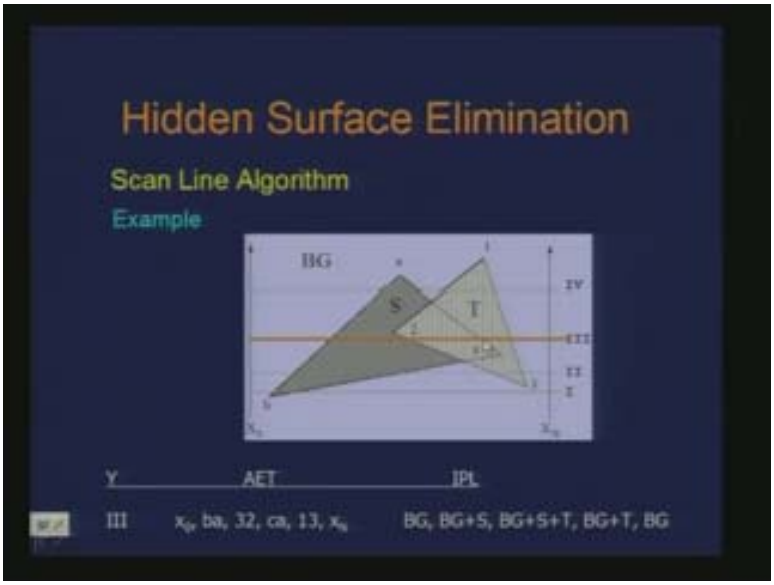
(Refer Slide Time: 35:40)



If I am talking about scan line number 1 so with respect to the active edge table what I have is x_0, this is the vertical edge located at x_0 the edge ba is this edge of the polygon the edge bc is there and x_n is the content of the activation table for this scan line.

Similarly, for in polygon list I have BG then I have BG plus S which is this polygon on the top of the background then again I have the background. Now looking at this it is sort of easy to process what action you need to do. Wherever you find BG active you display BG. Now there is only one polygon number on the top of the BG therefore you display that polygon and then again BG. This is a straight forward way. Similarly, if I go for this scan line I have x_0 ba bc and again this edge 32, 31 and x_n. As far as the polygons are concerned we have BG, BG plus S, BG, BG plus T, BG.

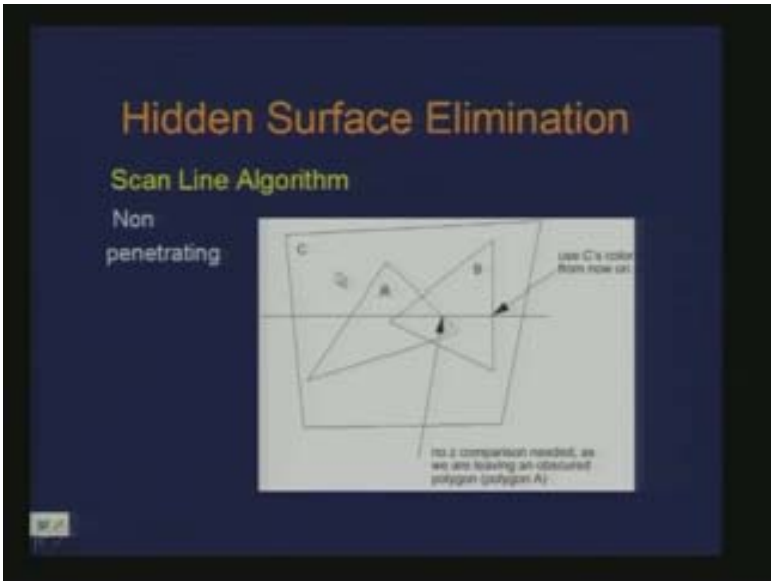
(Refer Slide Time: 40:09)



So this is again a simple case where I have the overlap of the polygons on the top of the background which are disjoint. So as far as the processing of this is concerned it is straight forward. Similarly, if I go to 3 now things are little different. I have x_0, ba, 32, ac, or ca, 13 and x_n and as far as the in polygon list is concerned BG, BG plus S now I have BG plus S plus T there is a span which is contained in more than one polygon. Then again I have BG plus T and so on. This is where I need to do something extra where I have more than one polygon flagged as n. So I would resolve it with respect to z to decide which of the polygon needs to be rendered. Again for the fourth is a simple case.

The two data structures we referred to such as the active edge table and the in polygon list tells you how they basically give you the sequence of rendering units. We are saying that for a given scan line what are the polygons which get flagged as n and here we also have the background and this s comes. Now it intersects this and you get another n polygon. So you have not come out of polygon S.

(Refer Slide Time: 47:21)



Basically you are doing the intersection of the scan line with various edges of the polygon and then doing this parity check. Therefore S would exit here when you intersect from this edge. That is where the parity test gives you the result that you are out. Therefore it is just a matter of finding the intersection of the edges and then labeling the polygons appropriately to be n.

For example, if there is a scenario where there is a big polygon and this could also be considered as the background of the scene where you have polygon A and polygon B and what we consider here is that these polygons are non penetrating that means they are not penetrating with each other. There is a distinct gap between the z values. So in that case there could be some advantages when you are doing rendering with respect to a scan line. If we are talking about a scan line here when I encounter the polygon A in some sense this becomes the obscuring polygon for C so I keep rendering it here and now I encounter another polygon B which is obscuring A and also C. Here what happens is that I exit the obscured polygon A. Here we are only considering between A to B and some background for C to B.

Here I exit the obscured polygon A. Here in fact I do not need to perform the z check. So if I am flagging this s as an obscured polygon then further on I do not have to worry about what happens to the z values. This is just to make this process more efficient. Methods to speed up the process in the case of non penetrating situation:

We need to find out the intersection with the edges otherwise we would not know what spans are there. Try to see what happens if there is a scan line just above it that is here and from that scan line it comes here.

What is the information you can carry? In fact the scan line above this was exactly same in terms of the in polygon list. So there is a sense of depth coherence from one scan line

to another scan line. Unless you hit points like this or point like this where the AET the active edge list changes then there is something else to be done. But if AET is the same then there is coherence between one scan line to another scan line for the depth. Once you have performed the visibility for a scan line for which AET is the same we need not perform the visibility for the next scan line. So there is this depth coherence we have between scan lines.

Now what happens if I have a situation where things are not so simple instead of having a non penetrating situation I have a penetrating situation so what do I do? Can you visualize the penetrating situation? There could be this A coming out of B that is the penetrating situation. There is some sign of false edge to be given which indicates this penetration. So what we are saying is we have a polygon S and this the part comes out of T so that is where it is penetrating so what you need is definition of this edge.

In some sense this is to be a sort of an extra polygon may be it is referring to the same polygon as this for the color information but for treating the other labeling for IPL you need an additional polygon here. Therefore we are adding this false edge and false polygon. So if we look at with respect to this scan line here this is what you have the AIT which is the active edge list where you have this edge ec and this is your polygon list in IPL where you have the polygon as BG plus S plus T plus something. Now with this you can handle the penetrating polygons.

(Refer Slide Time: 49:36)