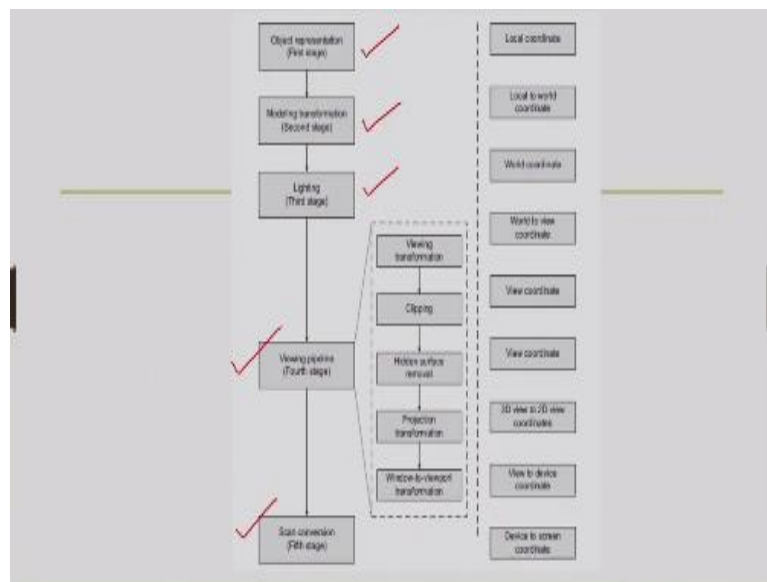


**Computer Graphics**  
**Professor Dr. Samit Bhattacharya**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Guwahati**  
**Lecture - 24**  
**Hidden Surface Removal - 2**

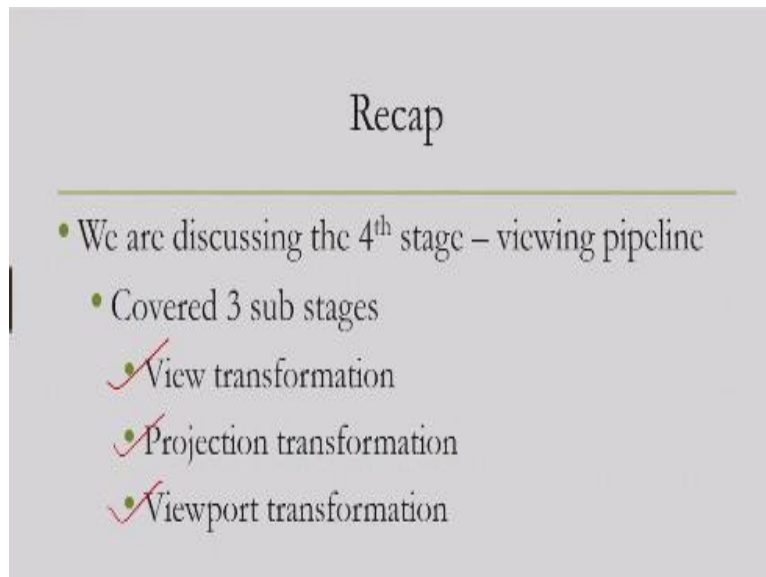
Hello and welcome to lecture number 24 in the course Computer Graphics. We are in the process of learning about the 3D graphics pipeline, which has five stages.

(Refer Slide Time: 00:46)



What are those stages, let us recap. Object representation, modeling transformation, lighting, viewing pipeline, and scan conversion. So we are currently in this fourth stage of discussion that is the viewing pipeline.

(Refer Slide Time: 01:12)



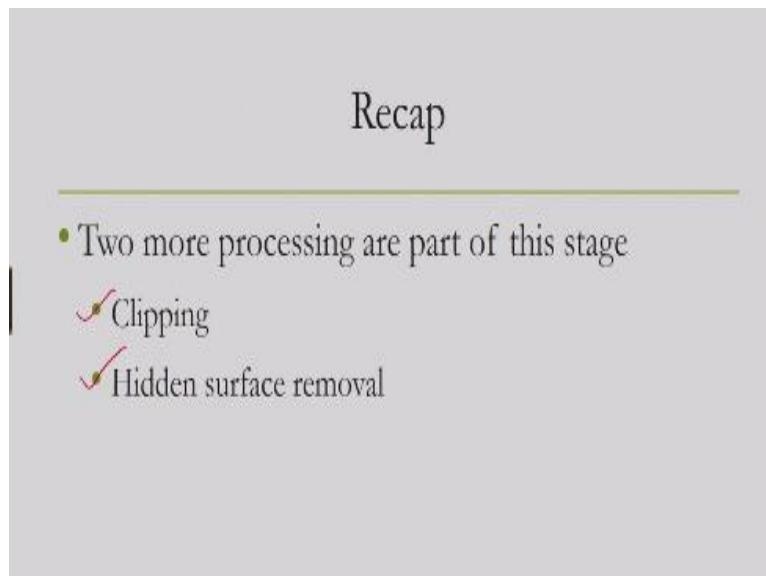
Recap

---

- We are discussing the 4<sup>th</sup> stage – viewing pipeline
  - Covered 3 sub stages
    - ✓ View transformation
    - ✓ Projection transformation
    - ✓ Viewport transformation

There we covered three transformations that are sub stages of the fourth stage namely, view transformation, projection transformation, viewport transformation.

(Refer Slide Time: 01:28)



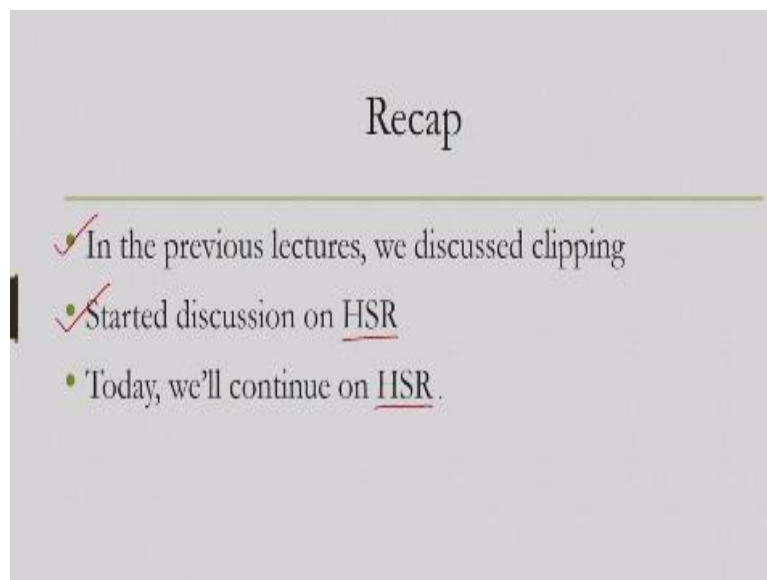
Recap

---

- Two more processing are part of this stage
  - ✓ Clipping
  - ✓ Hidden surface removal

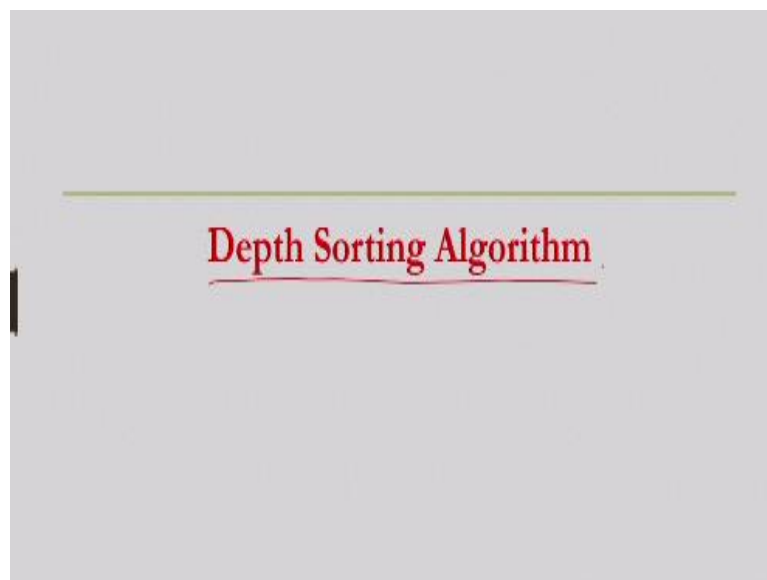
Then there are two more operations. These are also sub stages of this fourth stage, clipping and hidden surface removal.

(Refer Slide Time: 01:40)



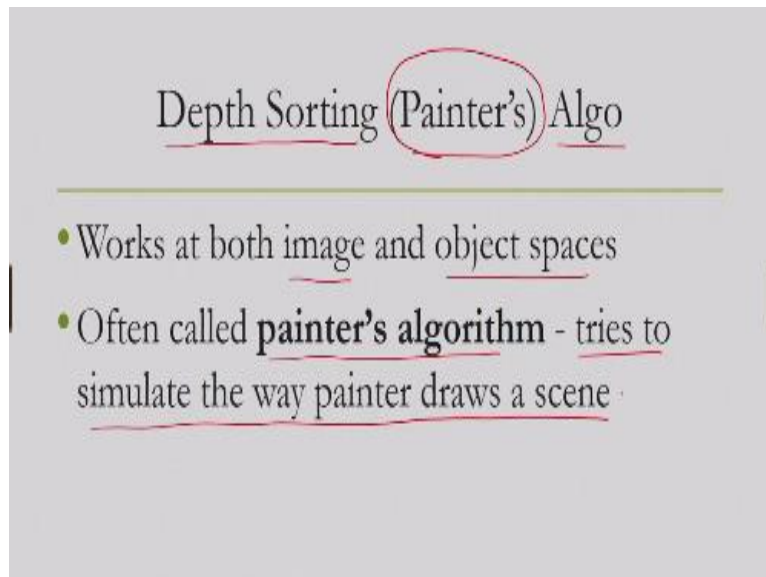
Among them, we have already discussed clipping and we started our discussion on HSR or hidden surface removal. So we will continue our discussion on HSR and conclude that discussion today.

(Refer Slide Time: 01:54)



So in the last lecture, we talked about two hidden surface removal method namely, back face elimination and depth buffer algorithm. Today, we are going to talk about few more hidden surface removal methods. We will start our discussion with another method that is called depth sorting algorithm.

(Refer Slide Time: 02:21)



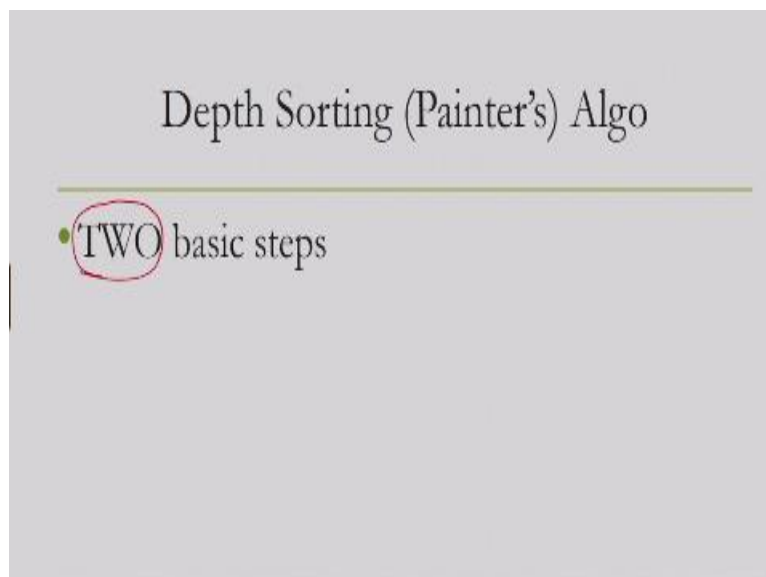
Depth Sorting (Painter's) Algo

---

- Works at both image and object spaces
- Often called painter's algorithm - tries to simulate the way painter draws a scene

Now, this depth sorting algorithm is also known as the painter's algorithm, another popular name and it works at both image and object space. So it works both at the pixel level as well as the surface level. And why it is called painter's algorithm? Because it tries to simulate the way painter draws a scene.

(Refer Slide Time: 02:56)



Depth Sorting (Painter's) Algo

---

- TWO basic steps

Now this algorithm consists of two basic steps.

(Refer Slide Time: 03:06)

### Depth Sorting (Painter's) Algo

---

- First - sort surfaces on their depth (from view plane)
  - Determine max and min depths of each surface
  - Sorted surface list created based on max depth

$S = \{s_1, s_2, \dots, s_n\}$  where  $s_i = i$ th surface and  $\text{depth}(s_i) < \text{depth}(s_{i+1})$

What is the first step? In the first step, it sorts the surfaces based on their depth with respect to the view position. To do that, we need to determine the max and min depth of each surface. That means the maximum and minimum depth of each surface and then we create a sorted surface list based on the maximum depth.

So we can denote this list in terms of notation  $s_i$ , and we can arrange them in ascending order of depth. So in this notation, depth of  $s_i$  is less than depth of  $s_{i+1}$  that is the first stage of the algorithm.

(Refer Slide Time: 04:21)

### Depth Sorting (Painter's) Algo

---

- Next - render surface on the screen one at a time
  - Starting with surface having maximum depth (i.e.,  $s_n$  in  $S$ ) to surface with the lowest depth

In the next stage, we render the surface on the screen one at a time, starting with surface having maximum depth that is the nth surface in the list to surface with the least depth or the lowest depth.

(Refer Slide Time: 04:45)

### Depth Sorting (Painter's) Algo

- During rendering surface  $s_i$ , we compare it with all other surfaces of  $S$  to check **depth overlap**
- Min depth of one surface  $>$  max depth of other surface

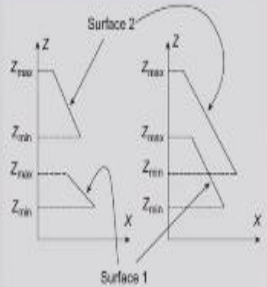
The diagram shows two surfaces, Surface 1 and Surface 2, in a 3D coordinate system with X and Z axes. Surface 1 is a rectangle with its minimum depth at  $Z_{min}$  and maximum depth at  $Z_{max}$ . Surface 2 is another rectangle with its minimum depth at  $Z_{min}$  and maximum depth at  $Z_{max}$ . The diagram illustrates that the minimum depth of Surface 2 is greater than the maximum depth of Surface 1, indicating no depth overlap. A red checkmark is drawn below the diagram.

Now, during rendering, during the second stage, we perform some comparisons. So when we are rendering a surface  $s_i$ , we compare it with all other surfaces in the list  $S$  to check for depth overlap. That means the minimum depth of one surface is greater than maximum depth of another surface as the situation is illustrated here. Because that these two surfaces here, there is no depth overlap because the minimum is greater than maximum. However, here the minimum is not greater than maximum, so there is a depth overlap.

(Refer Slide Time: 05:52)

## Depth Sorting (Painter's) Algo

- No overlap - render surface and remove from S



If there is no overlap then render the surface and remove it from the list S.

(Refer Slide Time: 06:03)

## Depth Sorting (Painter's) Algo

- Otherwise, perform checks
  - ✓ Bounding rectangles of the two surfaces do not overlap
  - ✓ Surface  $i$  is completely behind overlapping surface relative to viewing position
  - ✓ Overlapping surface completely in front of  $i$  relative to viewing position
  - ✓ Boundary edge projections of the two surfaces onto the view plane do not overlap

In case there is overlap, we perform more checks. First check is bounding rectangles of the two surfaces do not overlap, we check for it. Then we check whether surface  $s_i$  is completely behind the overlapping surface relative to the viewing position. That is the second check, this is the first check. In the third check, overlapping surface completely in front of  $s_i$  relative to the viewing position that is the third check.

And finally, the boundary edge projections of the two surfaces on to the view plane do not overlap, this is the fourth check. So there are series of checks that we perform in case there is depth overlap.

(Refer Slide Time: 07:06)

### Depth Sorting (Painter's) Algo

- Bounding rectangles of the two surfaces do not overlap
  - TRUE if there is no overlap in the x and y coordinate extents of the two surfaces
  - If either of these coordinates overlap, condition fails

Let us try to understand these checks. First check is bounding rectangles; do not overlap. Now, how do we check for it? So the rectangles do not overlap if there is no overlap in the x and y coordinate extents of the two surfaces.

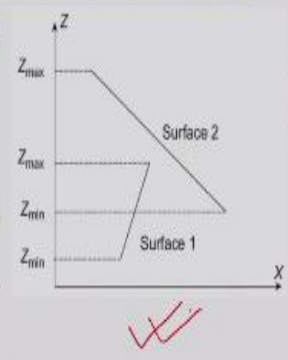
Consider the situation here. This is one surface, this is another surface. So here,  $X_{\min}$  is less than  $X_{\max}$ , so there is overlap. If there is no overlap then  $X_{\min}$  would be higher than  $X_{\max}$  of the other surface. So if either of these coordinates overlap then the condition fails that means bounding rectangles overlap. So we check for X and Y coordinate extents and then decide whether bounding rectangles overlap or not.



(Refer Slide Time: 08:22)

### Depth Sorting (Painter's) Algo

- Surface  $s_i$  completely behind overlapping surface relative to viewing position
  - Determine plane equation of overlapping surface (normal point towards viewer)
  - Next, check all vertices of  $s_i$  with equation
  - If for all vertices of  $s_i$ , equation returns value  $< 0$ ,  $s_i$  is behind overlapping surface
  - Otherwise, condition fails



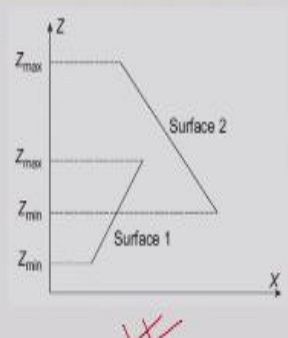
Next check is surface  $s_i$  completely behind overlapping surface relative to the viewing position. Now how do we determine it? We determine the plane equation of the overlapping surface where the normal point towards the viewer.

Next we check all vertices of  $s_i$  with the plane equation of that overlapping surface. If for all vertices of  $s_i$  the plane equation of the overlapping surface returns value less than 0, then  $s_i$  is behind the overlapping surface, otherwise, it is not behind and this condition fails. Situation is depicted in this diagram.

(Refer Slide Time: 09:22)

### Depth Sorting (Painter's) Algo

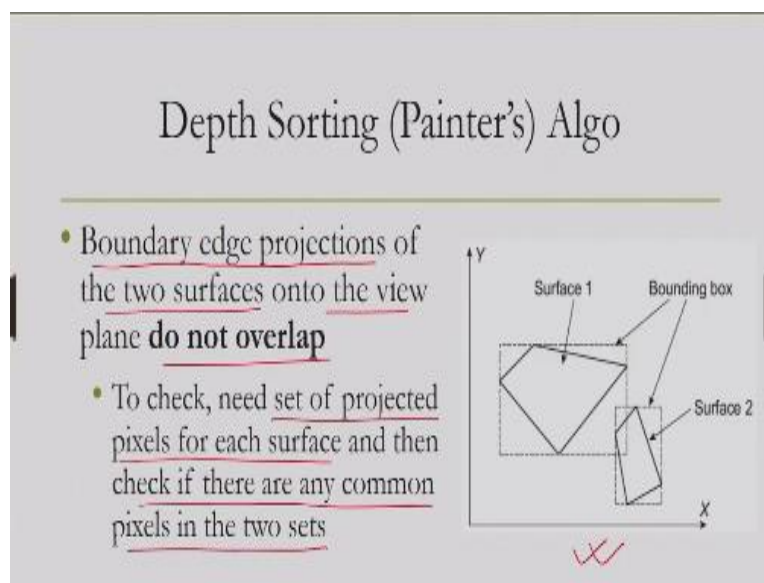
- Overlapping surface completely in front of  $s_i$  relative to viewing position
  - Can be checked similarly with plane equation of  $s_i$  and vertices of overlapping surface (for all vertices, equation should return positive value)



The third condition that we check is whether the overlapping surface is completely in front of the surface of interest  $s_i$ , again relative to the viewing position. Now, this we can check similarly with the plane equations that we have done earlier.

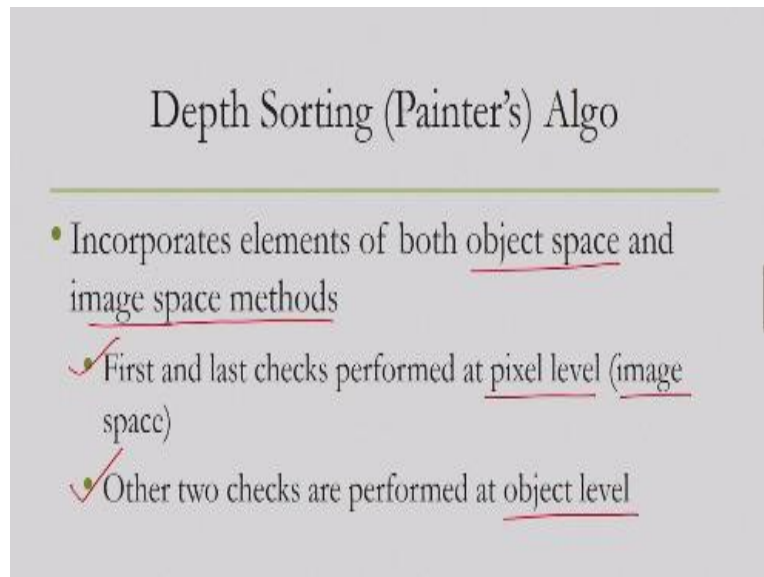
Now, this time we use the plane equation of  $s_i$  rather than this overlapping surface and we use the vertices of the overlapping surface. So we use those vertices in the plane equation and for all vertices, if the equation returns positive value then it is completely in front, otherwise, the condition fails. Situation is shown in this figure.

(Refer Slide Time: 10:19)



And finally, the boundary edge projections of the two surfaces onto the view plane do not the overlap, this is the final check. Now, in order to check for these we need set of projected pixels for each surface and then check if there are any common pixels in the two sets. The idea is illustrated here. If there are common pixels in the two sets then definitely, there is an overlap, otherwise, there is no overlap.

(Refer Slide Time: 11:00)

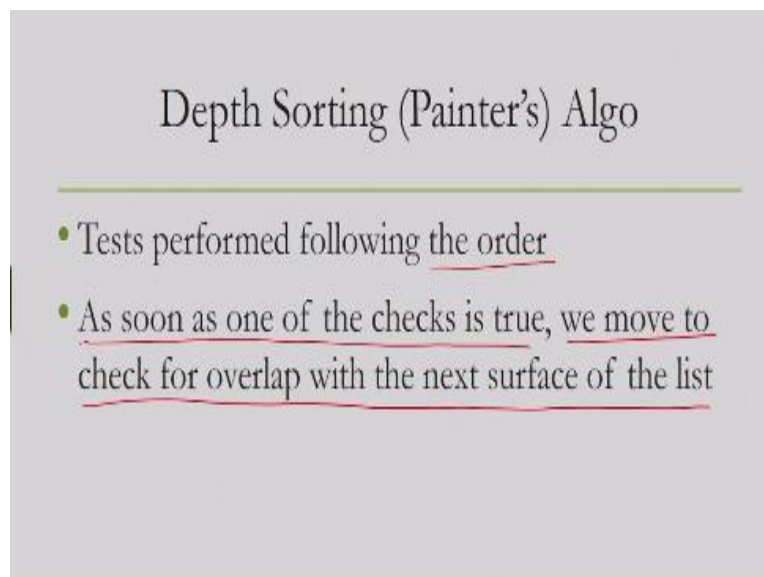


Depth Sorting (Painter's) Algo

- Incorporates elements of both object space and image space methods
- ✓ First and last checks performed at pixel level (image space)
- ✓ Other two checks are performed at object level

Now, as you can see here that this algorithm incorporates elements of both object space and image space methods. The first and the last checks were performed at the pixel level so that is image space, whereas the other two, second and third, were have performed at the object level. So here, the element of object space method is present.

(Refer Slide Time: 11:39)



Depth Sorting (Painter's) Algo

- Tests performed following the order
- As soon as one of the checks is true, we move to check for overlap with the next surface of the list

Now, when we perform the tests, we perform the tests following this ascending order maintained in s and also, the order of the checks that we have mentioned. Now, as soon as any one of the checks is true, we move to the check for overlap with the next surface of the list.

So essentially, what we are doing? Initially, we are checking for Z overlap for one surface, with all other surfaces, if it succeeds then we simply render the surface, otherwise, we perform the checks in that particular order, and during the checks if any check is true then we move to the next step rather than continuing with the other checks.

(Refer Slide Time: 12:43)

### Depth Sorting (Painter's) Algo

---

- If all tests fail, we swap the order of the surfaces in the list (called reordering) and stop
- Then, we restart the whole process again

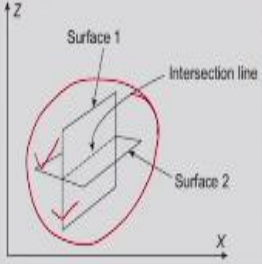
Now if all the test fail, what happens in that case? We swap the order of the surfaces in the list. This is called reordering and then we stop. Then we restart the whole process again from the beginning. So if all checks fail, then we need to reorder the surface list and then we start from the beginning.

(Refer Slide Time: 13:12)

### Depth Sorting (Painter's) Algo

---

- Sometimes, there are surfaces that intersect
- One part of surface 1 at a depth larger than surface 2, other part has lesser depth



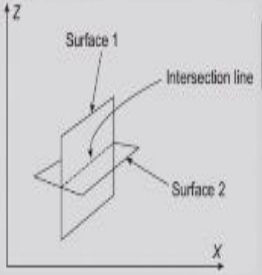
Now, sometimes there are issues. Sometimes we may get surfaces that intersect with each other. For example, see this surfaces. This is one surface, this is another surface, and they intersect each other. So in this example, one part of surface 1 is at a depth which is larger than surface 2, whereas the other part is at a depth which is lesser than surface 2, as you can see in this figure.

(Refer Slide Time: 13:48)

### Depth Sorting (Painter's) Algo

---

- We may initially keep surface 1  
after surface 2 in the sorted list



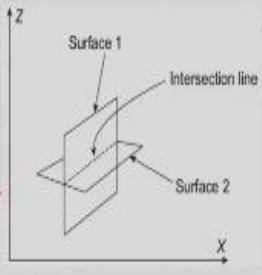
Now, in such situations we may face problem, we may initially keep surface 1 and surface 2 in a particular way that is surface 1 after surface 2 in the sorted list.

(Refer Slide Time: 14:10)

### Depth Sorting (Painter's) Algo

---

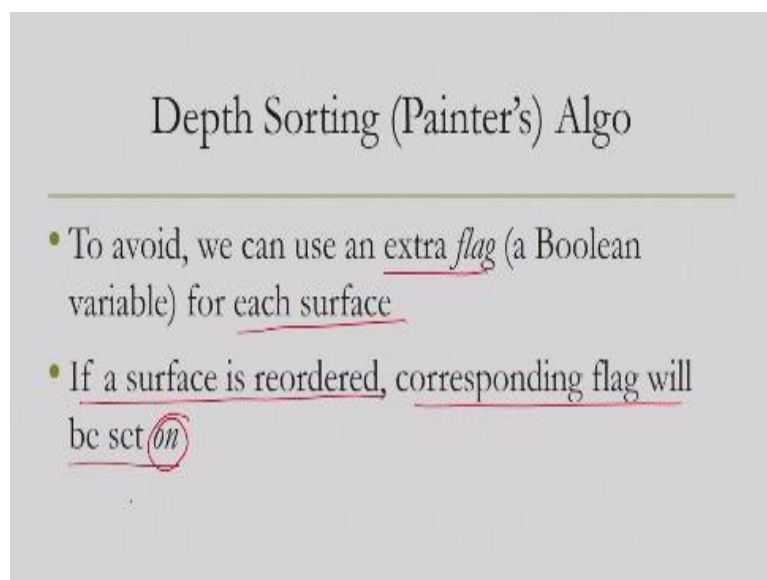
- All conditions will fail - have to  
reorder  $S_1, S_2$   $S_2, S_1$
- Conditions shall fail again - have  
to reorder again  $S_1, S_2$
- ✓ Infinite loop



However, if you perform the algorithm you will see that for these surfaces, all conditions will fail so we have to reorder. But that will not solve our purpose. Even if we reorder, the conditions will fail again and we have to reorder again.

So initially, we have  $S_1$  followed by  $S_2$ , next we will have  $S_2$  followed by  $S_1$ . Then we will have to reorder again  $S_1$  followed by  $S_2$  and this will go on, and we may end up in an indefinite loop because the surfaces intersect and the relative distance between the two are difficult to determine.

(Refer Slide Time: 15:05)



Depth Sorting (Painter's) Algo

- To avoid, we can use an extra flag (a Boolean variable) for each surface
- If a surface is reordered, corresponding flag will be set ON

In order to avoid such situations what we can do is we can use an extra flag, a Boolean flag for each surface. If a surface is reordered then the corresponding flag will be set ON, which indicates that the surface is already reordered once.

(Refer Slide Time: 15:29)

Depth Sorting (Painter's) Algo

- If the surface needs to be reordered next time, we shall do the following
  - Divide surface along intersection line
  - Add two new surfaces in the sorted list, at appropriate positions

Now, if the surface needs to be reordered again next time we shall do the following. We divide the surface along the intersection line and then add two new surfaces in the sorted list at appropriate positions. So when the surface needs to be reordered again, we know that there is an intersection. Then we divide the surface along the intersection lines and then add two new surfaces instead of one in the list in a sorted order.

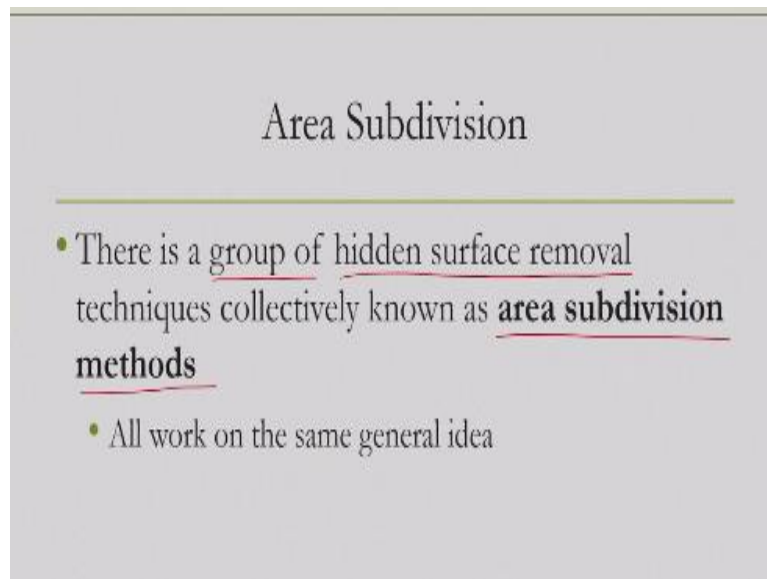
Of course, these two steps are very easy to do and require lots of computations, however, we will not go into the details we just give you some idea rather than the details of how to do that. So that is the basic idea of painter's algorithm.

(Refer Slide Time: 16:15)

Warnock's Algorithm

We will discuss one more algorithm Warnock's algorithm.

(Refer Slide Time: 16:29)



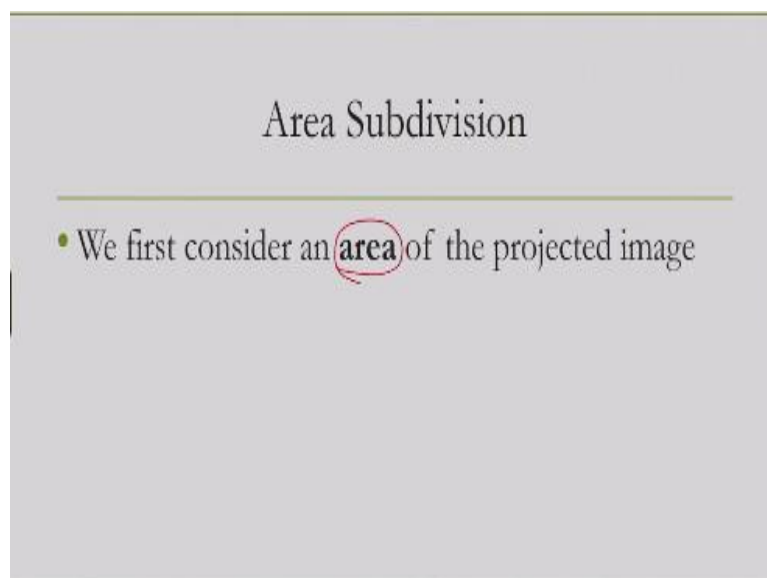
Area Subdivision

---

- There is a group of hidden surface removal techniques collectively known as area subdivision methods
  - All work on the same general idea

This is actually part of a group of methods for hidden surface removal, which are collectively known as area subdivision methods and they work on same general idea. And what is that idea?

(Refer Slide Time: 16:56)



Area Subdivision

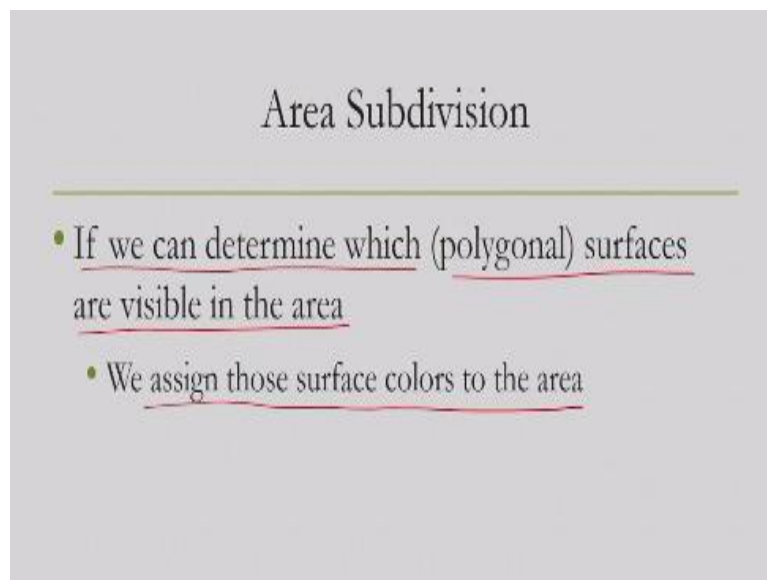
---

- We first consider an area of the projected image

So we first consider an area of the projected image.



(Refer Slide Time: 17:05)



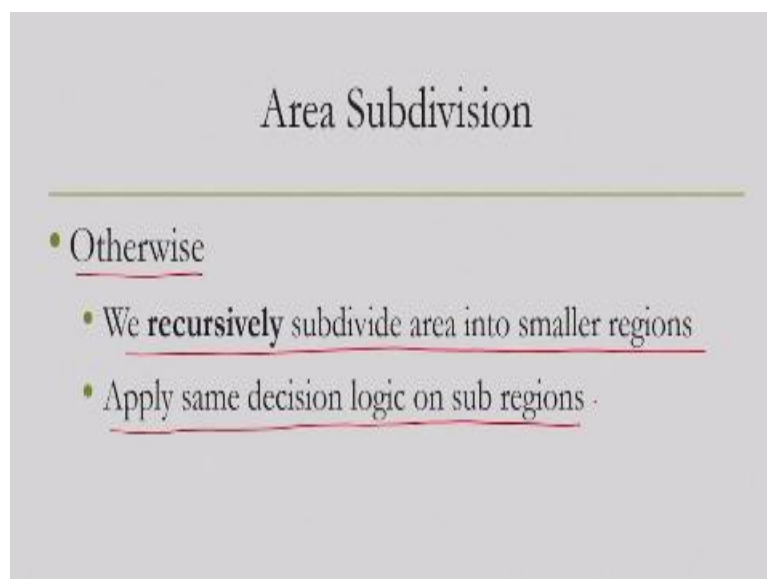
Area Subdivision

---

- If we can determine which (polygonal) surfaces are visible in the area
- We assign those surface colors to the area

Then if we can determine which polygonal surfaces are visible in the area then we assign those surface colors to the area. Of course, if we can determine then our problem is solved. So that determination is the key issue here.

(Refer Slide Time: 17:28)



Area Subdivision

---

- Otherwise
  - We recursively subdivide area into smaller regions
  - Apply same decision logic on sub regions

Now if we cannot determine, we recursively subdivide area into smaller regions and apply the same decision logic on the sub regions. So it is a recursive process.

(Refer Slide Time: 17:44)

## Warnock's Algorithm

---

- One of the earliest among all the subdivision methods

Warnock's algorithm is one of the earliest subdivision method developed.

(Refer Slide Time: 18:02)

## Warnock's Algorithm

---

- First, we subdivide a screen area into FOUR equal squares


Subregion  $P_{311}$  overlapped by the surface

In this algorithm, we subdivide a screen area into four equal squares. As you can see this is the region which we divide into four equal squares  $P_1$ ,  $P_2$ ,  $P_3$  and  $P_4$  then we perform recursion.

(Refer Slide Time: 18:28)

### Warnock's Algorithm

- Then, we check for visibility in each square to determine pixel colors in the (square) region



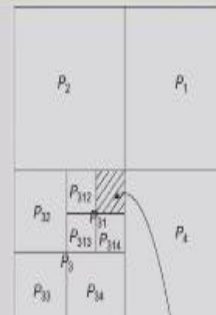
Subregion  $P_{311}$  overlapped by the surface

We check for visibility in each square to determine pixel colors in the square region. So we process each square at a time.

(Refer Slide Time: 18:40)

### Warnock's Algorithm

- THREE Cases to check
  - Case 1 - current square region being checked **does not contain** any surface
  - ✓ We do not subdivide the region any further and assign background color to the pixels contained in it



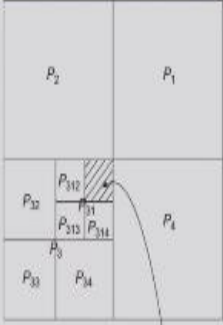
Subregion  $P_{311}$  overlapped by the surface

And in this processing, there are three cases to check. Case 1, is current square region being checked does not contain any surface. In that case, we do not sub divide the region any further because it does not have any surface so there is no point in further checking and we simply assign background color to the pixels contained in this sub region.

(Refer Slide Time: 19:13)

### Warnock's Algorithm

- THREE Cases to check
  - Case 2 - nearest surface completely overlaps the region under consideration
  - Square is not subdivided further
  - We assign the surface color to the region



Subregion  $P_{311}$  overlapped by the surface

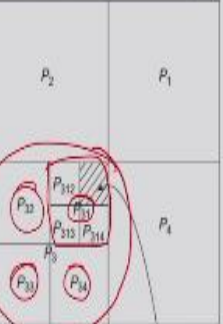
The diagram shows a square region divided into four quadrants labeled  $P_2$  (top-left),  $P_1$  (top-right),  $P_3$  (bottom-left), and  $P_4$  (bottom-right). The bottom-left quadrant  $P_3$  is further subdivided into four smaller quadrants:  $P_{312}$  (top-left),  $P_{311}$  (top-right),  $P_{313}$  (bottom-left), and  $P_{314}$  (bottom-right). A shaded area representing a surface is shown overlapping the subregion  $P_{311}$ . A curved arrow points from the text 'Subregion  $P_{311}$  overlapped by the surface' to the shaded area.

In case 2, the nearest surface completely overlaps the region under consideration. That means it is completely overlapped by the surface that is closest to the viewer. In this case also, we do not further sub divide the square, instead we simply assign the surface color to the region because it is completely covered by the surface. So note that here we need to determine the nearest surface and then determine the extent of this surface after projection so that we can check whether it completely cover that sub region.

(Refer Slide Time: 20:05)

### Warnock's Algorithm

- THREE Cases to check
  - Case 3 - None of these
  - We recursively divide region into four sub regions and repeat the aforementioned checks
  - Recursion stops if either of the cases is met or the region size becomes equal to pixel size



Subregion  $P_{311}$  overlapped by the surface

The diagram is identical to the one in Case 2, showing a square divided into quadrants  $P_1$  through  $P_4$  and sub-quadrants  $P_{311}$  through  $P_{314}$ . A shaded area represents a surface overlapping subregion  $P_{311}$ . In this diagram, the subregions  $P_{312}$ ,  $P_{313}$ , and  $P_{314}$  are circled in red, indicating they are the next level of recursion.

And there is case 3, where none of case 1 and 2 holds. In this case, we perform recursion. We recursively divide the region into four sub regions and then repeat the checks. Recursion

stops if either of the cases is met or the region size becomes equal to pixel size. For example, here, as you can see, we subdivided into four more sub regions  $P_{31}$ ,  $P_{32}$ ,  $P_{33}$ , and  $P_{34}$ . Then  $P_{31}$  we performed another recursion, again divided into sub regions, four sub regions.

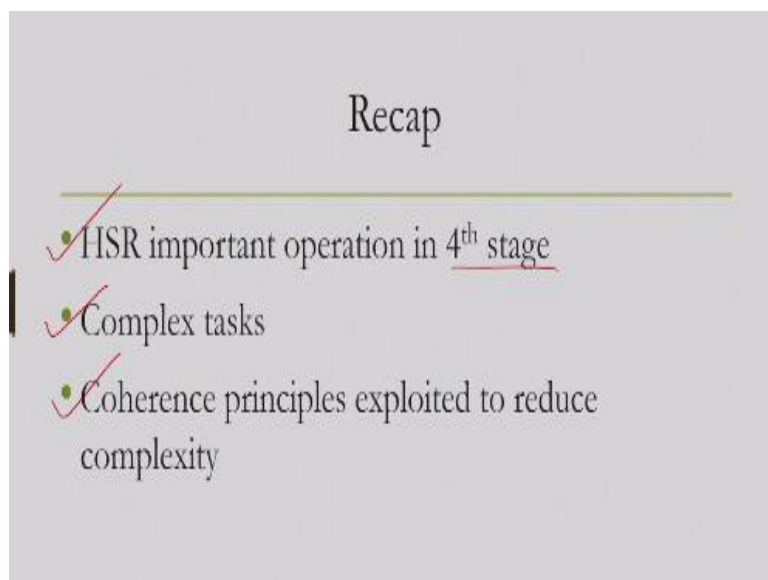
And we continue till either of the conditions 1 or 2 is met or the sub region size becomes equal to pixel size that is the smallest size possible. So this is the idea of the algorithm, where we assume that we are having projected image and then, we divide it into four sub regions at a time and perform recursive steps.

(Refer Slide Time: 21:27)



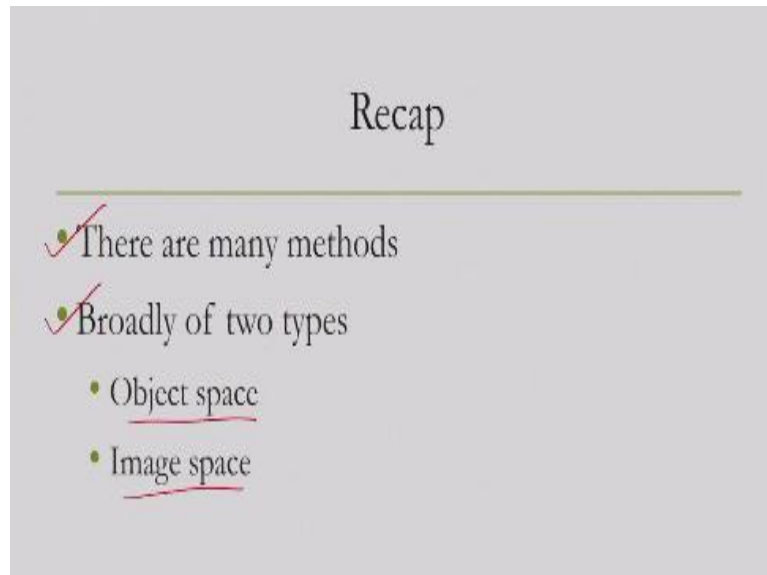
So with that, we have come to the conclusion of our discussion on hidden surface removal.

(Refer Slide Time: 21:37)



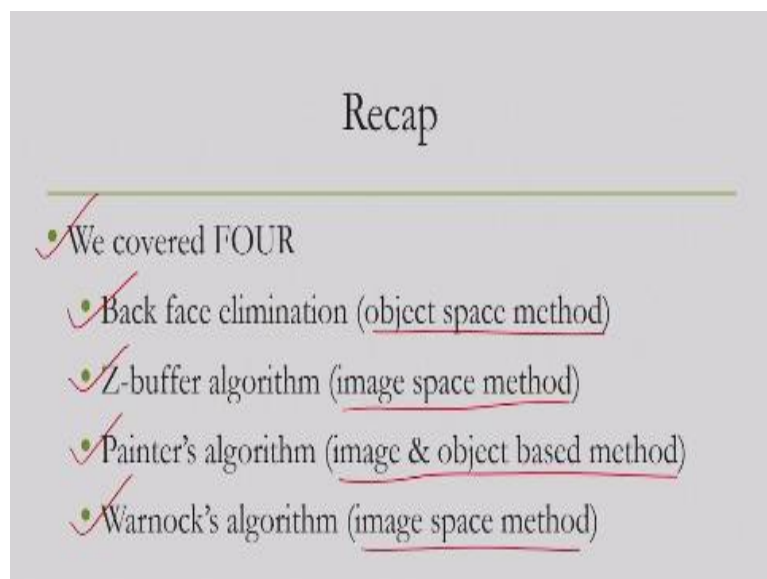
Now, before we conclude, few things to be noted here. The hidden surface removal is an important operation in the fourth stage, but it involves lots of complex operations and we exploit the coherence principles to reduce such complexities. These are the things that we should remember.

(Refer Slide Time: 22:05)



Also, we should remember that there are many methods for hidden surface removal and broadly, they are of two types, object space method and image space method.

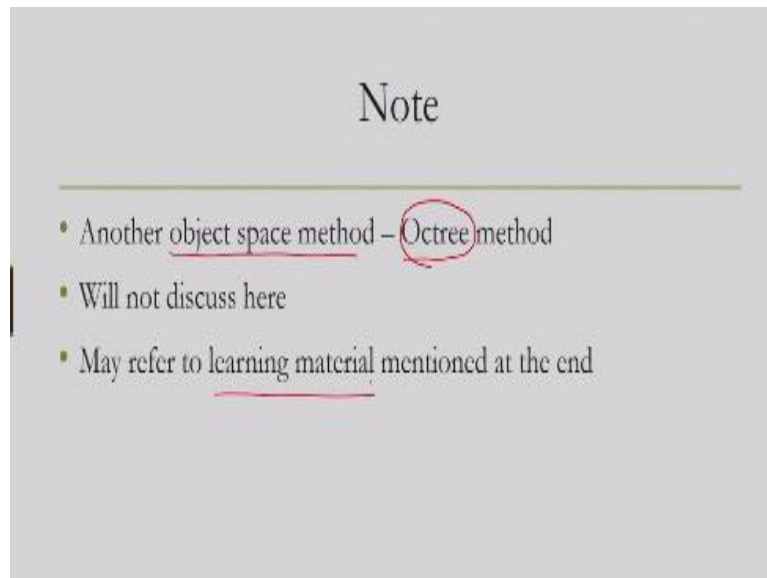
(Refer Slide Time: 22:16)



Among these methods, we covered four such methods that is back face elimination, which is an object space method; Z-buffer algorithm, an image space method; painter's algorithm, a

mix of image and object based method; and Warnock's algorithm, which is image space method. There are other approaches of course.

(Refer Slide Time: 22:42)

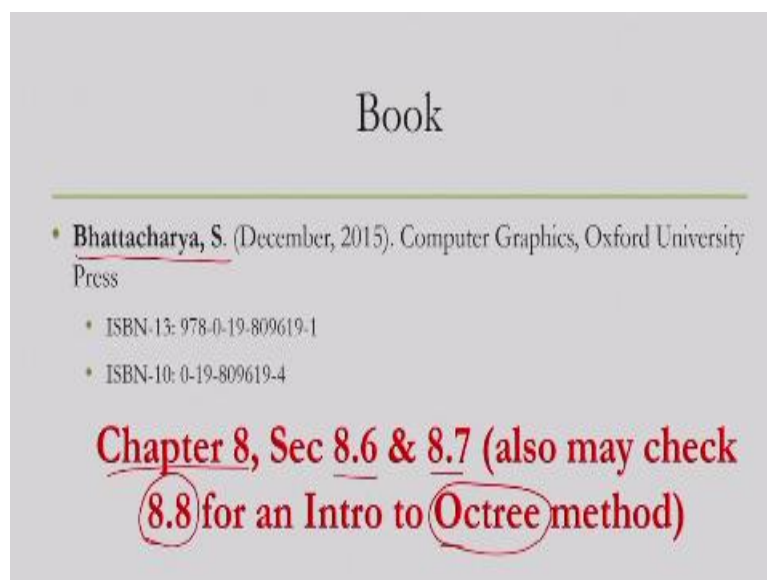


The slide is titled "Note" and contains a list of three points. The word "Octree" in the first point is circled in red. The word "learning" in the third point is underlined in red.

- Another object space method – Octree method
- Will not discuss here
- May refer to learning material mentioned at the end

One popular approach, which is an object space method is an Octree method, which we will not discuss in details, you may refer to the learning material. So we covered fourth stage and all its sub stages namely, the three transformations view transformation, projection transformation, viewport transformation, and also the two operations namely, clipping and hidden surface removal.

(Refer Slide Time: 23:19)



The slide is titled "Book" and contains a list of book information. The word "Octree" in the red text is circled in red.

- Bhattacharya, S. (December, 2015). Computer Graphics, Oxford University Press
  - ISBN-13: 978-0-19-809619-1
  - ISBN-10: 0-19-809619-4

**Chapter 8, Sec 8.6 & 8.7 (also may check 8.8 for an Intro to Octree method)**

Whatever we have discussed so far can be found in this book. You may refer to chapter 8, sections 8.6 and 8.7. Also, if you are interested to learn more about another object space method that is the Octree method you may check section 8.8 as well.

So that is all for today. In the next lecture, we will start our discussion on the next stage of the pipeline that is scan conversion. Till then, thank you and good bye.