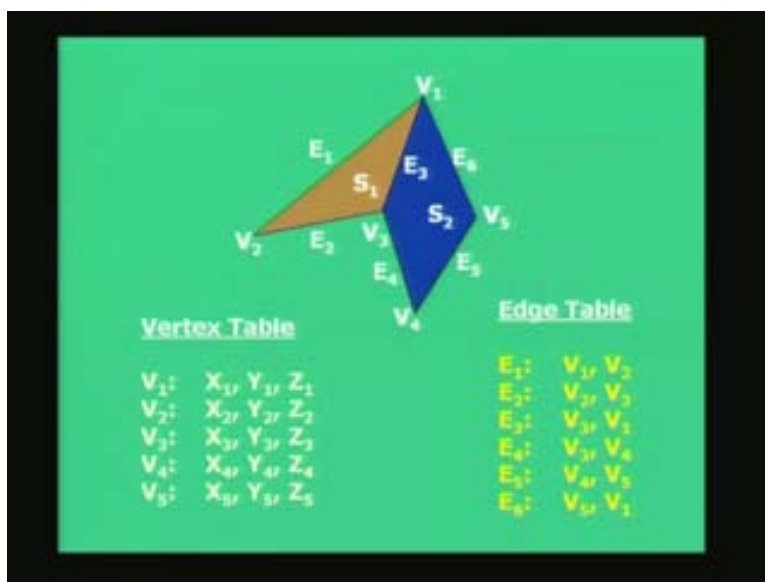


**Computer Graphics**  
**Prof. Sukhendu Das**  
**Dept. of Computer Science and Engineering**  
**Indian Institute of Technology, Madras**  
**Lecture - 25**  
**Solid Modelling**

Hello and welcome everybody to the lecture on computer graphics. We have been discussing in the last two classes the concepts of solid modeling and related to that we had discussed various concepts. The most noticeable among them was the boundary representations and also we had discussed the concepts of sweep representations. If you remember flip right to these two, I did give you an idea of the concept of regularized Boolean set expressions which involves Boolean set operators like union, intersections and difference. And we had seen using those how it could create complex objects from very simple primitives one.

We will extensively see today the use of Boolean set operations and how to create complex objects from that which is based on the principle of union, intersections and difference operators. We will see the area called Constructive Solid Geometry which is a very important part in solid modeling. But we will just go through the last few minutes of where we left off in terms of boundary representations or B-reps as it is called B-reps or boundary representations where a solid is considered to be a polyhedron bounded by several polygonal planes and each of this planes will have a set of edges and so we need to maintain a vertex table or a vertex list, also an edge list and a polygon surface table.

(Refer Slide Time: 02:55)



So we will start from where we left off in the last class. Where if you see in the slide a simple example of just two surfaces out of a solid which are adjacent to one another. One

is a triangle of course the other is a quadrilateral and they share an edge  $E_3$  and the vertex table stores the coordinates of the five vertices which are involved in building up these pairs of surfaces or polygons. And you also maintain an edge table which maintains the list of edges involved in these two surfaces. And the edge table entries contain the pair of vertices which are joined to form the edge. You also maintain a polygon surface table which contains the list of surfaces which are enclosing the polygon.

(Refer Slide Time: 04:51)

The slide contains two tables. The first table, titled 'Polygon Surface Table', lists two surfaces:  $S_1$  with edges  $E_1, E_2, E_3$  and  $S_2$  with edges  $E_3, E_4, E_5, E_6$ . The second table, titled 'Complete (expanded) Edge Table', lists six edges with their corresponding vertices and surfaces:  $E_1$  (vertices  $V_1, V_2$ , surface  $S_1$ ),  $E_2$  (vertices  $V_2, V_3$ , surface  $S_1$ ),  $E_3$  (vertices  $V_3, V_1$ , surfaces  $S_1, S_2$ ),  $E_4$  (vertices  $V_3, V_4$ , surface  $S_2$ ),  $E_5$  (vertices  $V_4, V_5$ , surface  $S_2$ ), and  $E_6$  (vertices  $V_5, V_1$ , surface  $S_2$ ).

Polygon Surface Table		
$S_1$ :	$E_1, E_2, E_3$	
$S_2$ :	$E_3, E_4, E_5, E_6$	

Complete (expanded) Edge Table		
$E_1$ :	$V_1, V_2$	$S_1$
$E_2$ :	$V_2, V_3$	$S_1$
$E_3$ :	$V_3, V_1$	$S_1, S_2$
$E_4$ :	$V_3, V_4$	$S_2$
$E_5$ :	$V_4, V_5$	$S_2$
$E_6$ :	$V_5, V_1$	$S_2$

Since in this case we have just picked two edges and surfaces. The list contains the levels of the two surfaces  $S_1$  and  $S_2$ . The first surface  $S_1$  will have the corresponding edges  $E_1, E_2$  and  $E_3$ . If you look back this is what  $S_1$  contains  $E_1, E_2$  and  $E_3$ . And the second surface  $S_2$  contains four edges  $E_3, E_4, E_5$  and  $E_6$ . Now these three tables the vertex table, edge table and the polygon surface table does not give you an idea of which edge is shared between these two surfaces  $S_1, S_2$  because there are solid body constraints about number of edges to meet to form a vertex and number of surfaces which could share an edge.

Therefore, based on that we build up a complete expanded edge table which will contain, is basically the same edge table with the surface information now appended or attached to that list of entries and we can see that for each edge there is one auto surfaces. The edge  $E_3$  is shared between two surfaces  $S_1, S_2$  and a transform vertex  $V_3$  to  $V_1$ . So that has been entered in the entry for  $E_3$  label and for others we also store the pair of vertices and the corresponding surfaces. Hence, that concludes the discussion of boundary representations. And where do you use these boundary representations? In fact the bigger question is why you have to represent solid object. You have to represent solid objects because you have to give visualization of them in a computer screen in computer graphics literature.

So you have to provide a wire frame diagram then provide shading and then we go on to concepts of shading. Before that of course we have to discuss the concepts later on about

visible surface detection back face scaling we will see. The most important attribute which will be necessary for a polygon when we process them either in terms of ordering those surfaces based on the coloring you need to do we will see that the polygon surface normal is an essential parameter or an important property of the surface which we need to compute.

(Refer Slide Time: 08:42)

**Equations for a normal, N, of a polygon ?**

$$a = \sum_{i=1}^n (Y_i - Y_j)(Z_i + Z_j)$$

$$b = \sum_{i=1}^n (Z_i - Z_j)(X_i + X_j) \text{ where } j = i + 1$$

$$c = \sum_{i=1}^n (X_i - X_j)(Y_i + Y_j)$$

If  $i = n, j = 1$

Assume a triangle (or take any three vertices of a polygon):

$$c = (X_1 - X_2)(Y_1 + Y_2) + (X_2 - X_3)(Y_2 + Y_3) + (X_3 - X_1)(Y_3 + Y_1)$$

$$= X_1(Y_2 - Y_3) + X_2(Y_3 - Y_1) + X_3(Y_1 - Y_2)$$

So let us look at equation of a normal n of an arbitrary polygon. Assuming the polygon has n different vertices or n edges. Remember any polygon, we talked about polygon filling and if you remember that you will definitely remember that the number of vertices is equal to the number of edges. So that is the small n in this sigma operation, the summation operation and there are three parameters a b and c which are listed here. If you carefully see the index is running from I is equal to 1 to n and it is a summation of terms which involve a pair of a product of two terms. Each of the product again is a subtraction of two coordinates. So the coefficient 'a' involves the coordinates Y and J. The coefficient b involves the coordinates Z and X of the polygon vertices and the parameter c involves the coordinates X and Y.

What are these  $X_i$ ,  $Y_i$  and  $Z_i$ ? They are nothing but the coordinates of the vertices of the polygon n of them and the summation goes from I to 1, I is equal to 1 and hence the first of the last one. And you can see generally that j is equal to I plus 1 that means you are subtracting the immediate next successive vertex from the previous one. And of course the only one condition which you need to keep in mind is that when I is equal to n then j is equal to 1. So these three equations which we have just seen now for the parameters a and b and c are in fact the direction cosines of the surface normal n of a polygon. That means if you remember this equation and for a polygon with three or more vertices and if the vertex coordinates that means the  $X_i$ ,  $Y_i$  and  $Z_i$  as in the vertex table are given to you for a particular polygon then you should be able to compute the surface normal n from these set of coordinates.

We will talk about the surface normal  $n$  more when we go to concepts for the various lectures on shading concepts of visible surface detection, back face culling and all that. But for the moment please remember these equations to compute the direction cosines of a surface normal  $n$ .

Now just put a little bit more into those equations. Now we see that cyclically you keep on rotating clockwise and anti-clockwise and keep computing the products of  $Y$ ,  $Z$  and  $X$  coordinates then multiply and sum it up to get a  $b$   $c$ . And let us assume a very simple case of a triangle that means you are talking of a polygon with three vertices. That means  $i$  will be running from 1 to 3,  $n$  is equal to 3, the small  $n$  which you see here in a  $b$   $c$  and if you break open one of these let us say you take the last expression  $c$  is equal to summation of  $X$  and  $Y$  terms the coordinates of the vertices then you can break open this expression and write like this which you can again simplify as the one given on the bottom of your screen. That is the simplest way by which you can write the parameter  $c$  and in fact you can do that and similarly you can write for parameters  $a$  and  $b$ .

Let us go back and see what we did was, I repeat again we just take  $n$  is equal to 3 and you write the three terms for the coefficient  $c$  like 1, 2 and 3 terms and then open up the factors and you can get a few terms cancelled out and you will be able to simplify the expression in this form. Please check it out yourself right now as we are reading these expressions here. Please work it out right now. That is what you get for  $c$  and similarly if you are able to obtain  $c$  you should be able to extrapolate the logic and write expressions for  $a$  and  $b$ . Either do that or use the expressions of  $a$  and  $b$  given in the previous slide and open the sigma term and write these expressions. That is what how you get a  $b$  and  $c$ . These expressions can also be written in matrix form.

(Refer Slide Time: 09.34)

Similarly:  $a = Y_1(Z_2 - Z_3) + Y_2(Z_3 - Z_1) + Y_3(Z_1 - Z_2)$   
 and  
 $b = Z_1(X_2 - X_3) + Z_2(X_3 - X_1) + Z_3(X_1 - X_2)$

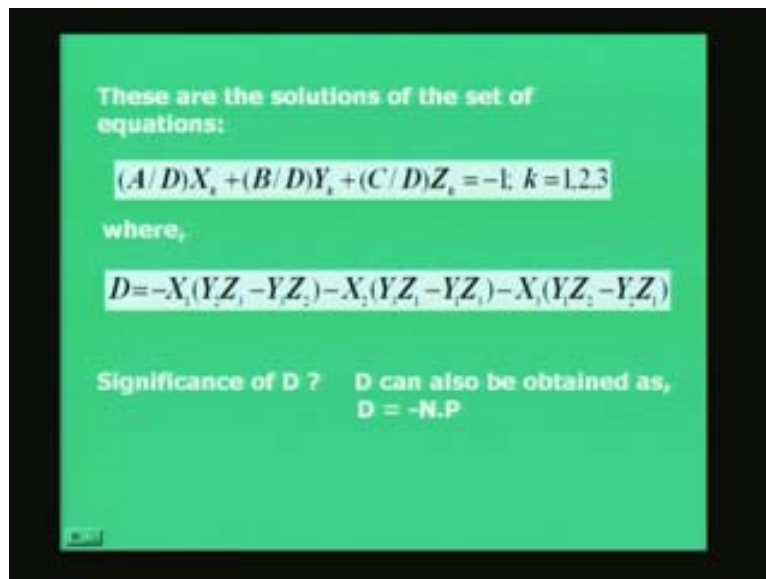
These expressions can be written in matrix form as:

$a =$	$\begin{vmatrix} 1 & Y_1 & Z_1 \\ 1 & Y_2 & Z_2 \\ 1 & Y_3 & Z_3 \end{vmatrix}$	$b =$	$\begin{vmatrix} X_1 & 1 & Z_1 \\ X_2 & 1 & Z_2 \\ X_3 & 1 & Z_3 \end{vmatrix}$	$c =$	$\begin{vmatrix} X_1 & Y_1 & 1 \\ X_2 & Y_2 & 1 \\ X_3 & Y_3 & 1 \end{vmatrix}$
-------	---	-------	---	-------	---

If you see carefully, of course the expression a b is given on the top of your screen. Expression of c was given earlier and if you see for yourself that the determinant basically becomes the determinant of X and Y for c X and Z for b and Y and Z coordinates for the coefficients a, parameter a. So this a b c which are the parameters are of the surface normal. The direction cosines of the surface normal n can also be written very simply in the form of an expression using determinant of a 3 into 3 matrix as given here. If you keep a watch on these a b c and move forward a little bit you will be able to find that these are nothing but the solutions of the set of equations given by this particular expression.

I leave it as an exercise for you to try out that if you are given an expression like this a linear equation where A, B, C and D are the parameters to be computed and there are three of these the  $X_k, Y_k, Z_k$  are nothing but the coordinates of the vertices then for k equal to 1, 2, 3 you can write these three expressions from the desktop equation here and you will be able to compute A by D, B by D and C by D as the one which is given here in the bottom. So what will come out, the denominator of the coefficient will come as an expression like this. Of course the numerator A, B and C will come in the way which is given here.

(Refer Slide Time: 11:09)



So what is this d? Remember this, when you write the set of three equations based on the vertices of the polygon you are basically trying to compute the surface normal n and that is what you get as the solution because any three points form a plane and that should be sufficient for you to define the plane uniquely and obtain the surface normal.

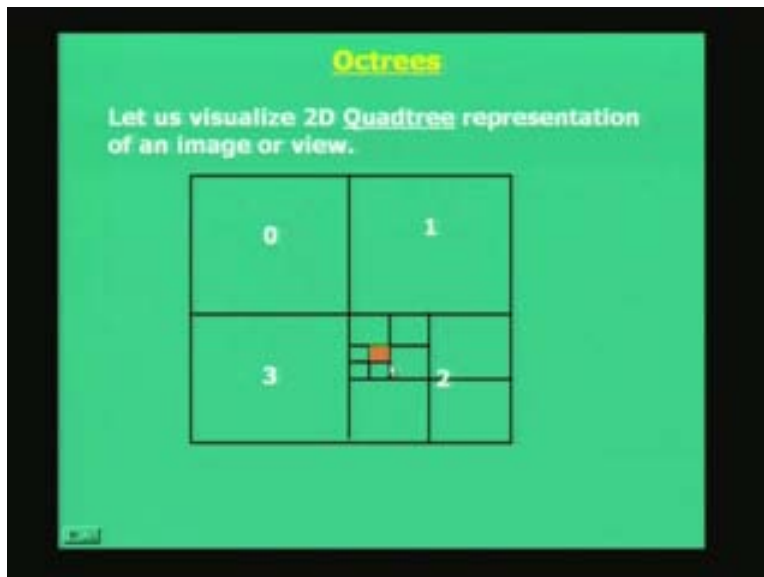
The significance of this denominator term D is very interesting. in fact the D we will see soon that the factor D will basically be 0 if the plane passes through the origin. If the plane passes through the origin the factor D is equal to 0 otherwise D is equal to a dot product of N. P where P is any point on the polygon of the plane and N is the surface

normal. So it is the dot product in the negative signs between the vector joining any point on the plane P with the surface normal. In other words what is D? We can almost visualize that this is going to be the distance of the plane, perpendicular distance or shortest distance of the plane from the origin of the coordinates system. That is what is given by the factor D.

I repeat again, if the plane passes through the origin then the value of D will be 0 and hence that is the case when the plane will be forced to pass through the origin when the value of D is null. This is how you compute the surface normal. Remember these expressions that can be used compute the surface normal. So, that brings us to the end of the discussion of boundary representation.

We will actually start to discuss a new concept of Octrees. Before discussing the concept of Octrees representation of solid objects let us try to visualize a 2D quadtree representation of an image or a view. So what is done in this particular case is the entire image is split into four quadrants. That is what will result in the quadtree representation so you have a 0, 1, 2, 3 you can label it clockwise and anti-clockwise as you may feel like and then depending upon the region of interest we can divide one of these quadrants into four equal parts again and keep doing it so on.

(Refer Slide Time: 13:03)

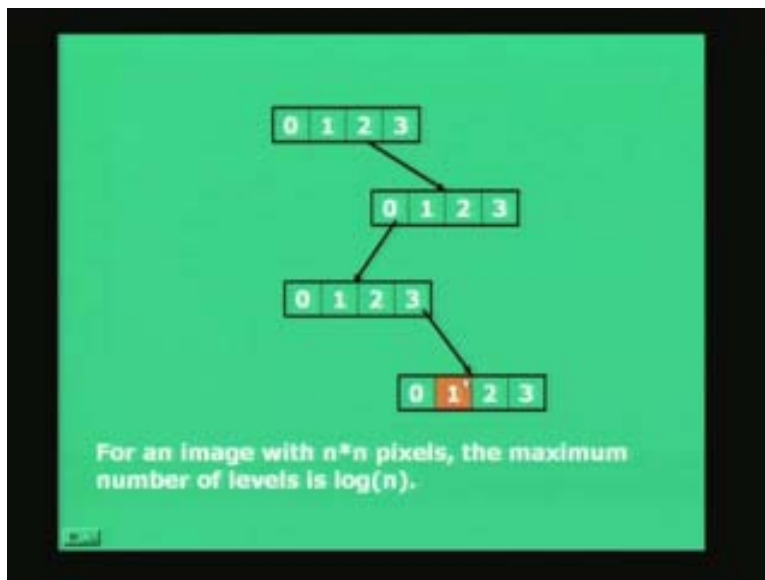


This is a typical representation which is used in concepts of image processing and it is called an image pyramid representation as well. The concepts are closer to this but right now if you look at quarter representation what we mean is the following. That the image consists of four different parts and these four different parts are four children of a tree structure which can be used to represent this image and since each node can have at the maximum four children or four sub nodes as you may call it then hence the name is quadtree.

So, if you look at this representation I am interested to just locate a small sub region with an image which is somewhere here. We will say that this sub region is in the second child of the root node which represents the whole image. And then under the second quadrant it is under the zeroth sub node or child and again within that under the third. So, if you look at the quarter representation of this particular region here is what you will get. The root node which is of course on the top we have to assume that the image has four children basically the root has four and each of those could again have at the maximum four different sub nodes or children from left to right or from 0 to 3.

But since you may not split all the four different sub quadrants or quadrants in fact only the second quadrant is split into four parts.

(Refer Slide Time: 15:01)



So you can see here that the root node which is the child at the top layer is split into four different parts 0, 1, 2, 3. So you can of course draw a root node at the top which represents the image and that will afford children on the top layer or the first level of the decomposition of the image structure. And the second one is again split into four parts. As you can see here this is the second quadrant which splits again into four parts and information is in the zeroth sub quadrant of the second quadrant.

If that is so, you can see that is why the 0 is split into four parts again. I hope you have a visualization of what we had seen in the previous slide or you please draw this yourself on a sheet of paper and try to analyze this and try to visualize this decomposition with respect to the quadrant representation. Quarter representation is given here with respect to the image split done in this particular slide.

So if you go back and forth now in the second level of decomposition the third quadrant at the second level of decomposition is further split into four different parts and you have the first one which is full, which is of area of interest. It basically means that if you are

interested in this particular area or it could be even at the lowest level of pixel because if we keep on decomposing like that into several level, split in to four parts each of them split in to again into four parts and so on, if you on keep doing that at various different levels of decomposition at the very lowest level of the tree or if you are going for maximum decomposition that is possible at the very lowest level you will have a set of pixels for each particular child. The leaf nodes will be a pixel. If you keep on decomposing to the maximum possible extent.

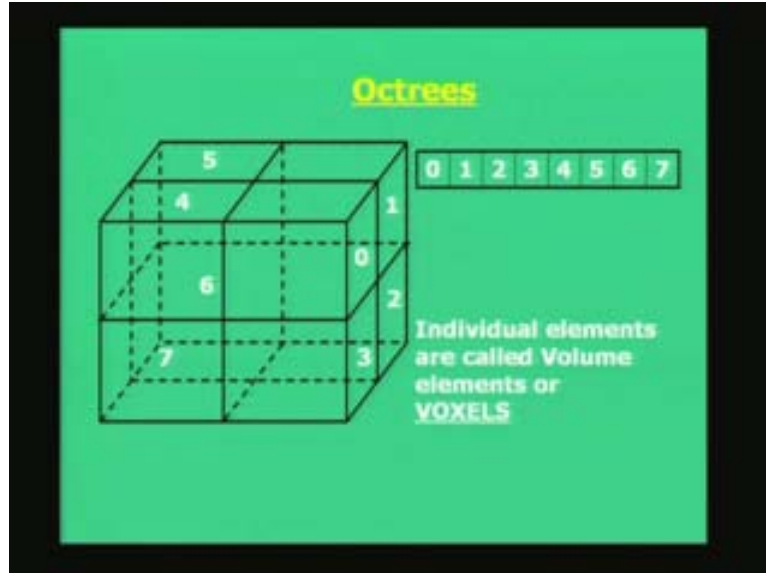
So you know that this could be a pixel or a small sub region as given in the slide and the first level under quadrant 2, then under that quadrant 0, then under quadrant 3. So 2, 0, 3 as you can see here 2 is decomposed into four parts and the 0 is decomposed into four parts and third one is again decomposed into four parts. And this particular region which is levelled as 1 is also labeled in this picture for the certain color indicates, that is my area of interest and that is where my information is or that could be the lowest level of pixel if the image has only about 16 into 16 image or something like that.

So, for the image with  $n$  into  $n$  pixels you can see the maximum number of levels which you can have in a quadtree is a log of  $n$  of course to the base 2. That is what you need to have. That means if you simply have a 2 into 2 image you can decompose and make just one level of quadtree. If you have a 4 into 4 you can decompose into two levels. If you have 8 into 8 image you can decompose into three levels. If you have 16 into 16 image you can decompose in four levels, 32 five levels and so on. That is the figure which you must keep in mind and that is what you do for image in terms of quadtree representation and that is in 2D. Remember it is not a solid, image is not a solid.

Here we are talking of solid modeling. But what we will do now if you understood the concept of quadtrees representing images in 2D we can visualize that we can extend this philosophy or concept to 3D where we can represent objects with the help of Octrees. If you look at this figure that I have taken a very large cube and split it into several parts by planes as you took a line to split a plane an image plane or any two dimensional space into two parts, took another line and split into further parts split into four. What we are doing here in this case is to take a plane and split this volume into two equal parts.



(Refer Slide Time: 20:06)



Let us say if you consider this particular class room in which we are sitting and listening to this lecture. I can take a plane between you and me and split this classroom into two sub volumes as we had done in the case of a two dimensional image space where we took a line and you took a horizontal line or a vertical line it does not matter. If your are taking the line and splitting you are splitting the two dimensional space into two equal sub planes or sub regions. Here, in the case of a three dimensional case the line becomes a plane.

So I take a plane and split this class room into two different parts or sub volumes or sub spaces in 3D and I can take the plane either as a vertical one or a horizontal one and again vertical could be aligned between you and me or it could be orthogonal between my line of side between you and me. So there are three possible planes along the three orthogonal planes, coordinate planes or you can visualize them to be normal to the three orthogonal axis of the X, Y, Z coordinates system. So, if you do that and do the same operation on cubes you can almost visualize, you can see I have taken a vertical plane which is here let us say and then you take a horizontal plane which lies flat and another vertical plane which is orthogonal to both.

Therefore, if you can take three planes and divide this class room, any volume for that matter, any cube, any rectangular parallelepiped, any subspace, any volume in 3D can be split into eight different parts by taking one plane here, one plane there and another plane like that three orthogonal planes is what you need to take. And if you do that then you are splitting this volume let us say what we consider is a very large cube into eight parts, eight equal parts. Of course it is a cube then if you take symmetrically located plane and then you are dividing the space into eight equal sub volumes or parts or regions in three dimensional spaces.

Hence now in the case of an image which we represented by quadtrees after using two lines and splitting into four sub regions we say here that the volume or the 3D space is split into eight equal parts by three different planes. Now this root node which represents the volume of the cube will have eight children. In case of a quadtrees we had four children and now in the case of the Octrees we have eight children. That is what given here and look at the levels which have been done clockwise.

The planes with the volumes right of the vertical plane if you consider the vertical plane as you have a 0, 1, 2, 3 and 4, 5, 6, 7. It is something like that you have four smaller cubical volumes on the right hand side and four equivalently smaller cubes on the left hand side. So totally you have eight sub volumes or volume elements and hence this term called voxels are often used to represent elements of 3D volumes. As in the case of an image you already know the term pixel which represents the word picture element is the meaning of the term pixel you have in 3D equivalent to pixel the term called as voxel. So Octrees is nothing but a voxel based representation. We will see it is basically called a spatial occupancy is equivalent to a 2D region occupancy.

Here we will talk about spatial occupancy but right now let us understand the meaning of the word voxel which calls of volume elements and that means in the first level of decomposition you are coming with eight sub regions which can also be visualize as very large voxel. It could be that each of those sub volumes could be unit element which you are using to represent three dimensional space if you are not decomposing further. But if you are decomposing further like in the case of an image where we took each of those quadrants and split into four equal parts we can do that similar thing in the case of a quadtree representation what do you do.

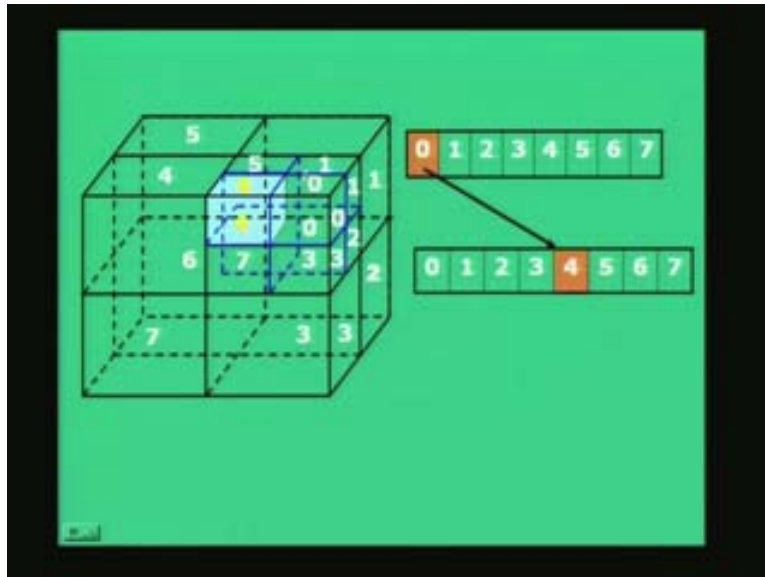
You basically take any one of these or all of these eight sub volumes and use three planes again to split it into eight equal parts again. So, voxels depends upon what level of decomposition you are looking at the smallest element which you will get is called the volume element or voxels. And let us try to look this is the first level of decomposition where the root node will have eight children label from 0 to 7 in sending of course in levels. But you can level them in any order depending upon what sort of sequencing you want for the representation. You can order those on the positive X axis and then on the positive Y axis and so on. And these are the eight different children.

So I hope you are able to visualize this three dimensional figure where the eight sub volumes or eight sub cubes created out of this larger cube by three different planes which will partition this space into eight voxels if you are not decomposing further. So that is the first level of decomposition. But you can split further into eight other parts. This is done in this case where the voxel 0 after the first level of decomposition the voxel number 0 is split again into eight equal parts and again those three similar planes are used.

As you can see here there are two vertical and one horizontal plane. Each of them are orthogonal to one another and you can see within voxel number 0. If you compare this figure with the previous figure this is the unit volume. The zeroth voxel you need not, the

first level of decomposition and the zeroth unit volume is split into eight parts and you still have 0, 1, 2, 3 on the right hand side at the second level of decomposition and say only 4, 5, 6 and 7 and 6 is of course hidden but you can still level that if you want and that is what is on the right hand side.

(Refer Slide Time: 24:43)



Why did I label with the color the volume element 0 at the first level of decomposition and then the number 4. Well we will say here that our interest is in this unit four or this in the octree representation. The node four at the second level of decomposition is of interest because the object lies there.

Let us say the object lies there or the object is a cube which just occupies exactly the volume occupied by the fourth node at the second level of decomposition. And since the fourth node is at the second level of decomposition it falls under the zeroth node of the first level of decomposition here. That is why you have an arrow running from 0 to this sub division and then the four is level. In fact if you have the occupancy or the object occupying some other position as well which falls under the different nodes or the first level of decomposition itself you may need to break and sub divide the other unit volumes at even the first level of decomposition. But let us say for the time being that our object only occupies the unit volume four corresponding to the node four at the second level of decomposition. Hence at the second level of the tree the fourth will be leveled.

Of course if you feel that there is some information, also let us say some information in terms of parts of the solid present in the third node. Let us say third node here at the second level of decomposition you may need to split it that further. So those pointers only will exist for those parts where solid body exists. So what you are talking in a three dimensional space is that you are leveling those nodes only which is occupied by solids. And those nodes which are pointing to null or empty will not have any solid object. It is a

vacuum or space which we are not interested in. This is the way by which Octrees are used to represent solid objects. And this is how you do that in terms of the representation.

Now let us say in this particular room you are interested to know where a particular object is aligned in terms of its quadtree representation. Let us say it occupy some space for which you know exactly X, Y, Z coordinates of a set of polygons which is occupying a small space. What do you need to do for an Octree representation is, if you are interested in this full room which is our region of interest you first split at the first level into eight equal sub volumes? Then you find out within these eight which of the sub volumes is this polygon under the solid line. So take that one out of those eight and then again split into two parts.

Now if that solid is occupying a part of two different sub volumes or even three you need to split all of them into eight equal parts. You keep on doing that till you come to the lowest level of decomposition where you cannot go down below in terms of resolution of defining, defining the minimum unit volume which is in fact call the voxel. So you have to visualize this Octree scenario as if that this entire class room or the room which you are sitting now and watching this show is built up of small unit cubes and those unit cubes are vacant or free if there is no solid object occupying that portion.

If there is a solid object occupying that portion then the corresponding node or leaf corresponding which represents the sub volume are at certain level of decomposition has to be set to on so that will not be a null point or it will not be empty. That is the concept of representations of solids. As you look here, if we see in the slide I repeat again with the simple example of only about two levels of decomposition at the most is that, if a solid occupies a part of four or even the full part of the sub volume four at the second level of decomposition we will say that you will be labeled and it will be set to one. All the other units in the first level of decomposition and the seven of them at the second level of decomposition, if there are no solids occupying that particular space we will set it to null.

And of course why did we break the zeroth unit sub volume at the first level of decomposition into eight equal parts. Well, if you go back to this slide and if this entire zeroth sub volume was occupied by one particular cube then I would not have broken it, I would have said that has been set 1 and the rest would have been 0. But since a part of that zeroth unit volume at the first level of decomposition is occupied partly, I need to break it. So I break it up into eight equal parts and look which of those eight are again occupied.

In this case I have shown that a cube occupies that fourth node or the volume represented by the fourth node and hence that is the one which is set to one and all others are kept back. Now it will be possible that a part of that fourth node is occupied not the full one. So, in that case you might have to again break the fourth node into eight equals of volumes similarly with three planes. Or if you feel that there is some occupancy by solid objects, by planes or whatever it is in the third sub volume at the second level of

decomposition you need to break it further and so on. This is the explanation which shows how to build up Octrees and for representing a solid.

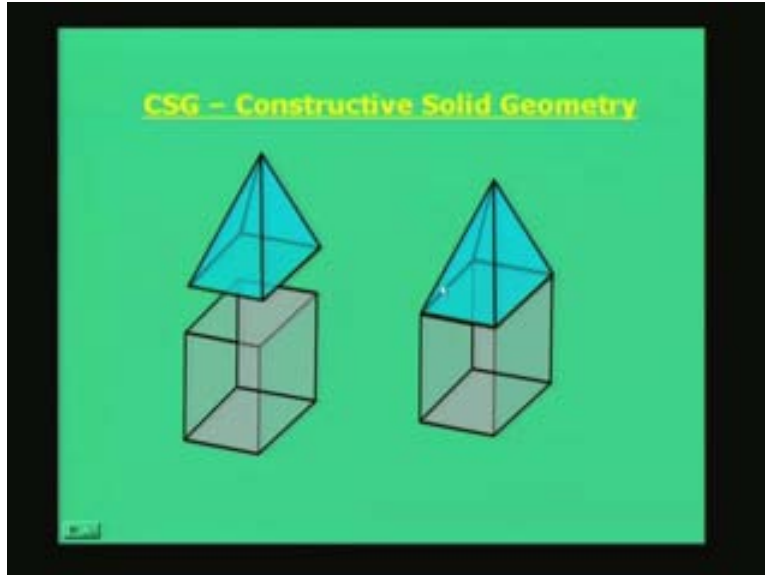
You have to specify the minimum unit volume beyond which you cannot break out. It is equivalent to visualizing the pixel in an image you cannot break the pixel further that is the lowest level in a quadtree. Similarly, in the case of an Octree representation you can have a voxel which is a minimum unit volume. You will say I will not go below one inch cube. Let us say some minimum volume or it could be one centimeter cube, one cc cube of one centimeter length, breadth and height or even one millimeter cube let us say. So you define your volume with respect to the whole dimension and try to fit objects present in a particular environment, it could be a classroom or even a simple scenario where you have just solids, rigid structures like cubes, cylinders, spheres and all that present in a particular room or a space which is there. This is how you build up Octrees.

Now we will move it on to the next representation which is in fact the last one which I will be covering. Then of course a couple of others which I have just named them, of course I have named in couple of classes back. But this is a concept of Constructive Solid Geometry which says that you can combine more than one object and build complex structures. In this case you can look into these two different structures; one is a cube, another is a pyramid with a rectangular base and you need to join these two objects and these two solids.

Well, if you remember in the couple of classes back we did talk about Boolean set expressions and the three operators in the Boolean set expressions were union, intersection and difference. You remember them? They are union, intersection and difference. And we also have seen conditions where we need to probably eliminate a few hanging points, lines and surfaces which result you to this Boolean set operators. And that is a process called regularization where we throw out hanging points, lines and surfaces even because those are not solids they cannot just exist on its own in a particular space. You have to have a solid and that is what you are interested in model solids, not model points, lines and surfaces. Instead of paper should also be bounded by two surfaces; one on the front and one on the back side.

Regularized Boolean set of operations or expressions are used in Constructive Solid Geometry to build up complex structures from primitives. Of course there is another concept of area called primitive instancing which is used for CAD tools.

(Refer Slide Time: 35:19)



You can have primitives at a very higher level such as which could be for an assembly plant, it could be a gear assembly or it could be a spanner or it could be hammer or whatever it is. So you can have different types of assembly structures as primitives. Instantly shape them with the help of cad tools and then combine them and fit them that is all possible. But we are talking of graphics object modeling here at the very low level. We assume the four simple primitives let us say which I kept on giving examples in the last two classes. Even today I am using a cube or a rectangular parallelepiped. A sphere is a one, a cone, cylinder and may be the fifth one could be a pyramid. So assume that these four or five primitives are available to you because you can generate all of these by boundary representations or sweep.

It is very easy to generate all of these in terms of boundary representation generate the X, Y, Z coordinates with the help of what you call as sweep representations. Of course with sweep you can generate curved objects shapes as well. But let us start with this primitive solid. You can visualize the simple 3D geometrical shapes very easily and try to build more complex structures. So we will see one or two building blocks which are used to compute and generate more complex structures out of the low level primitives.

So, if you look back into the slide you have a pyramid with a rectangular base and you have a cube and I want to put the pyramid on top of this cube. So this is the operation which I want. I desire to make a structure which should look like this and I will say that finally this is my structure, this is my goal which will consist of two parts, a union of a cube with a pyramid with a rectangular base. And in such a manner the dimensions of the base of the rectangular pyramid is equivalent to one of the faces on the cube on which it is sitting. This is the simple union operation.

Of course you can do other operations such as the difference and intersection. But if you look at union operations of course assuming that you know how to regularizing it by

throwing hanging points, lines and surfaces. You have to visualize here that I am trying to generate solids which may be interpenetrating or may not be. Typically if you say that there are two solids which I am trying to unite in reality of course one solid cannot enter another solid. So if you want to put those constraints you have to put that saying that I need to position them very correctly such that my faces do not interpenetrate. That is, one surface or one polygon of an object 'a' should not enter another polygon of an object 'b'. That is true in the case of solids. You can put a solid inside one but you cannot collapse them in such a manner that one face of one object enters another one. That is what it is. But of course it depends upon what sort of object hierarchy and what sort of object modeling you want to do.

If you really need to have complex structures sometimes you can have one object, view polygon faces of an object really interpenetrating or getting inside the plane of the other one because we know how to clip polygons with respect to another in 2D and as well as in 3D. So assume that you know that this is what you could use. But in this example what we are saying is we want to create a structure where you must position in such a manner that the planes do not interpenetrate but just touch each other. So you create union and when you create union of course the middle surface is what you need to throw out.

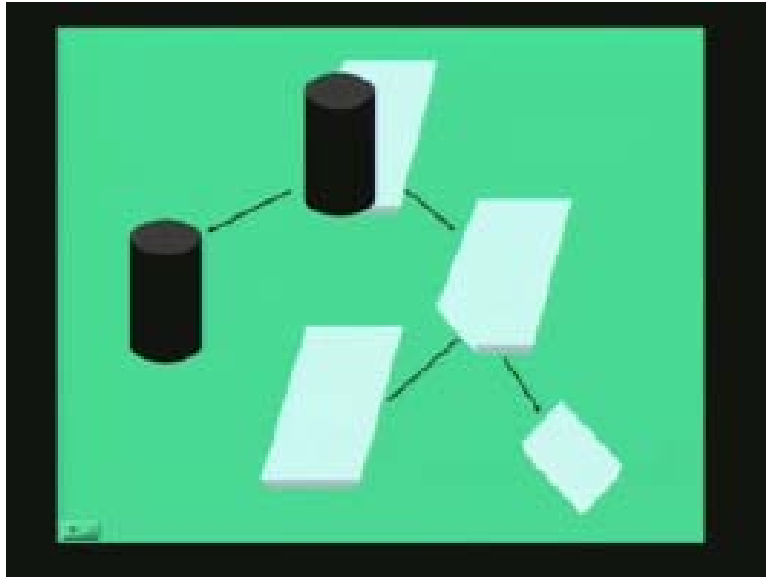
I leave it as an exercise for you to check the Euler's equations with the vertices, edges and faces for this particular structure.

Now, continuing on Constructive Solid Geometry if you have a look here this is what I want to generate. So let us say if I want to generate this particular structure what does this particular structure say? Well I am seeing a cylinder, a black one in front and I am seeing a thin rectangular plate with a small height and they are not only adjacent to one another it is in fact we have to visualize that the plane is coming out of the cylinder.

Of course the cylinder can sit on the plane, it depends upon where you want to position them but you should be able to generate a scenario where the plane and the cylinder are mixed at base that means one object is entering into another. So I will say this object is a union of these two parts. So I will take a cylinder which I have shaded with black for you to give visualization for the sake of visualization otherwise the colors have no significance.

I will like to say that the structure at the root node in Constructive Solid Geometry in this particular example is a union of two different structures; a cylinder and a wedge type of a structure, a rectangular parallelepiped, a very thin one where the left bottom part is truncated or broken as you can see for it. So, we will like to position the cylinder at one point and take the union of these two structures.

(Refer Slide Time: 38:44)



But cylinder is easy to visualize. You can do a boundary representation of cylinders very easily using sweep. How to generate this particular structure for the rectangular parallelepiped? I will say that this is also based on the combination of two different objects. I can say that I can take a rectangular parallelepiped and then subtract a small portion of that with another small object.

So, if you see the large objects on the left hand side and say that is object a and the right side is object b then the object here which is the broken rectangular piece of a rectangular parallelepiped with a broken part can be obtained by taking the operation  $a \text{ minus } b$ . If 'a' is the left hand object here which is the larger rectangular parallelepiped and the b the smaller one and then I can have  $a \text{ minus } b$ . Of course you need to position the b suitably so that you know which part to be taken off or truncated. This is an example of how Constructive Solid Geometry could be use to generate complex structures.

Of course there is no end to create more and more complex structures. If you visualize that you need to add some more object to this. Please go ahead because this is not the final object this is only an example. You can bring any other object may be a sphere, may be a cone or something else and then unite, subtract or intersect. Take a difference of some of the objects and add to this and create more and more complex structures. But the most important part of this is to visualize what you want to create.

If you have visualization and you can model them in mind saying that this is probably union of two or a difference of two or a intersection of two different solids which are again complex in some sense then you can break it into sub parts and so on and construct this tree where at the lowest level of the tree you will have the primitives, the geometrical three dimensional simple primitives which we discussed about. Again I repeat cube, cone, cylinder, sphere and may be a pyramid with a rectangular or even you can visualize hexagonal base or an octagonal base. So you can take four or five of these 3D

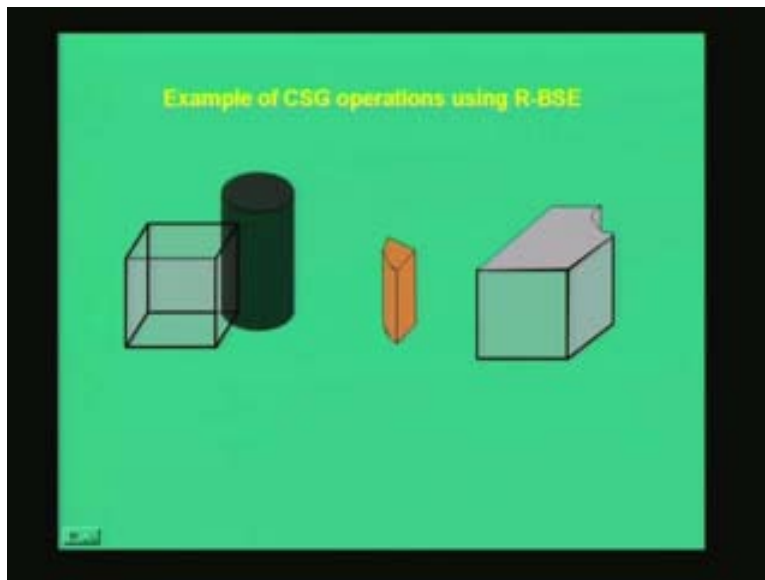


geometrical primitives and then start to do the Boolean set operations. Of course you need to use regularization to throw of these hanging points and lines. But you give these operations, create more complex structures and go up higher, up the ladder. That is what you do. In this case we were just talking about just two different operations; a difference here and then a union here which could create this particular structure.

Now let us look at another example of CSG operation using regularized Boolean set expressions. Let us say I take these two structures. Now I am not giving the final shape and we are giving an example again of a cube and a cylinder. And let us say I position them in such a manner, remember virtual reality, all the solid objects, I keep repeating this again, one object does not enter the other one but in this case the cube has entered. The part of the cube is inside the cylinder and I want to do an operation which will result in this particular structure.

Can you guess what this operation will be between the cube and the cylinder to get this? It should be an intersection operation. So, we have seen union and difference earlier and this is the nice example of the intersection operation.

(Refer Slide Time: 41:55)

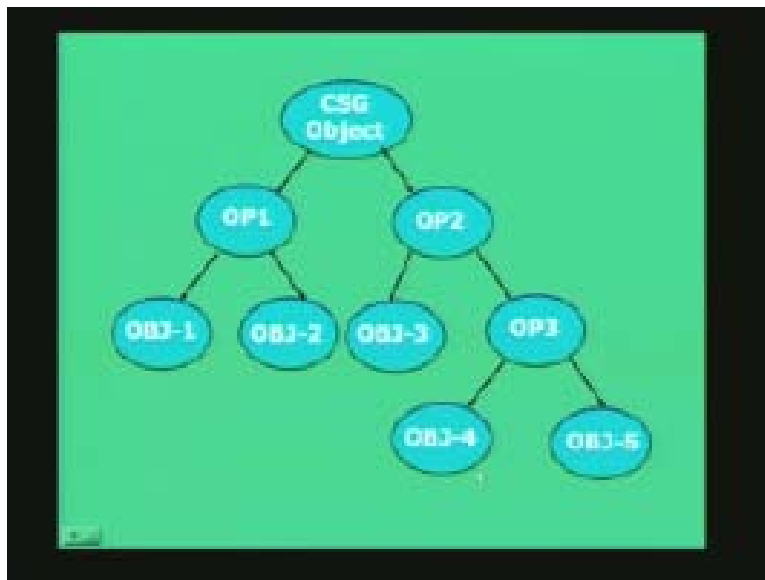


That means intersection of the two solids the cylinder and the cube will result in this particular structure. And this is the common volume between these two; the intersection we will use the common volume. What do you want to do? This is not our main aim. Our main aim is to probably create this structure which is a part of that rectangular parallelepiped chopped off in such a manner that you basically as if this part is missing from that structure. So what do you do? Take the cube again or rectangular parallelepiped and position this wedge at the corner where you want to get this volume taken away. So what you need to do is take this cube put that and then use a difference in operation. So, it is this entire cube minus this resultant structure which will produce this final structure. You can see that. Again it is a two level of decomposition or construction of the CSG. At

the lower level you will have an intersection operation and then a difference operation which will create that final structure.

Of course at the lower level as the leaves of the tree for this CSG you will have the cylinder, you will have two different cubes or one cube, one rectangular parallelepiped and the intersection operation creates the structure in the middle and that is again used by the last operation to create the final structure at the root. This is a typical example of a Constructive Solid Geometry tree where the objective is on the top and which is due to the resultant of two different operations which will unite finally and create this CSG object. And we will say that of course CSG could be at the root which will be a resultant of union of two different objects which will again be the resultant. That means at the lowest level I use object 4 and object 5.

(Refer Slide Time: 43:25)



If you look at the right hand side sub-tree the lowest level at the leaves object four and object five we use an operator three level here to create a structure which is again combined with the object three at the operator two level to create one object. Then you take object 1 and object 2. Of course we assume here that object 1, object 2, object 3, object 4 and object 5 are five different three dimensional geometrical primitives or some of them are of a similar type of simple three dimensional geometrical primitives which are used to create complex structures. So at the final level you have two operators.

You are uniting the left one side creating from object 1 and 3 and right hand side creating object 3, 4 and 5 which is again united or with some operator to create a Constructive Solid Geometry final object. This is just again two or three levels of decomposition. But again as you can see that the CSG object resultant due to this itself could be a primitive one. So you can talk of an object oriented hierarchism, hierarchy in the whole system where I just look at the node and say that is my complex structure and all other

corresponding primitives at the leaf nodes in the sub-tree will have to follow the corresponding operations as given by the tree.

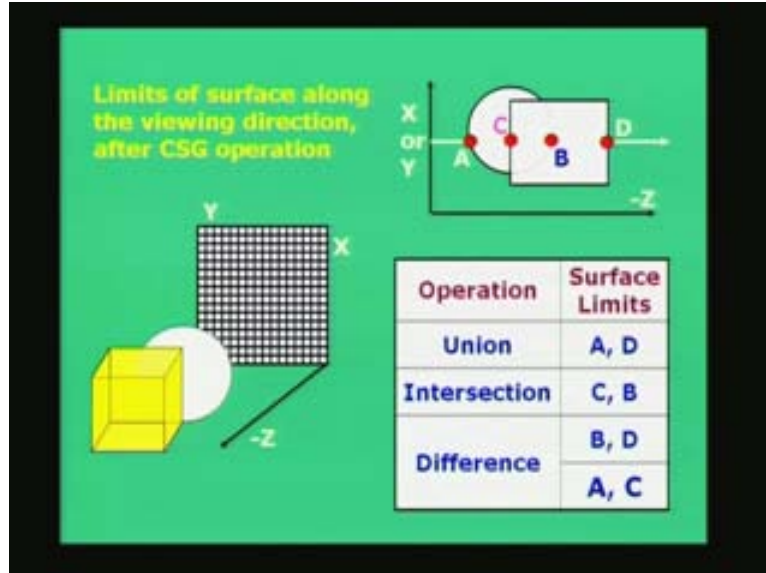
So if I give a transformation to the entire structure at the top at the root node all my corresponding, it will affect the children at the lowest level itself where the primitive objects are defined. They will also go through the similar transformation because that is what we studied in transformation of solids in rigid structures. So you can keep on building up very complex structures and say you just have it pointed to a root node which will give you the polygons representing the particular object. And what happens is that object structure is built with the help of polygons with different children at the leaf nodes which are simple primitives. Simple primitives build slightly complex primitives, complex primitives to simple object structures let us say and then again simple object structures to more complex and complex object structures.

You can now visualize when you talk of complex structures well we are talking of designing of buildings, aircrafts, automobiles, bridges and many shopping malls. Let us say, the inside scenario of a restaurant or a hotel you want to give a visualization of an object. That is Constructive Solid Geometry for you a typical example of a structure.

This is another simple example of limits of surfaces along the viewing direction after CSG operation. Let us say I have two objects a sphere and a cube and I am viewing it through an X, Y plane. If you remember the raster image display and the viewing direction is along the Z direction. We will come back to this again when we discuss ray tracing where you can now visualize that I pass a ray out of one pixel in the X Y plane.

The second diagram shows that I am viewing along on the X. I have taken the projection on the left hand side example on the XZ plane or YZ plane. And I am shooting a ray out along the direction of the view which is minus Z direction out from a certain pixel XY towards the object. Let us say that ray intersects the sphere at two points A and B and it also intersects the solid cube at two points C and D. So as I said before the part of this sphere was inside the cube or vice versa and when we result in a complex structure out of these two primitives and create a complex structure either by union, intersection or difference this is what will result. That means this table tells you what are the surface limits after the regularized Boolean set operations on these two objects and where does the ray intersect that final structure.

(Refer Slide Time: 47:15)



If you look at union of these two structures the sphere and the cube the ray will intersect at the two limiting bounds A and D. If you look at the intersection which is the common bound or the common volume between surfaces, the sphere and the cube then it will intersect in B and C. If you take a difference depending upon whether you are subtracting the sphere from the cube or vice versa you can have two different types of solids which will result and you will either get the limiting B and D or you could get A and C.

I hope this gives an idea of what the Constructive Solid Geometry operations could give you again using a simple example of only two structures and then you can keep on having more and more complex structures by adding more and more primitives. This is an example of the limits of the surfaces along the view direction after certain CSG operations with examples of two solids on the left hand side. The 3D diagram is given where you have the raster frame buffer, set of X Y pixels and viewing direction is along the minus Z direction.

You have two objects not only the sphere is occluding the cube but we have arranged the two objects, solids in such a manner that the part of the sphere is inside the cube. You have to have a union operation to get the resultant structure which could be a non-regular object in terms of not having simple geometrical features like sphere, cone, cylinder or even a pyramid.

So the last few points before we wind up the lecture on solid modeling because for just to wind up before we move on to different criteria, to evaluate solid modeling techniques, we have discussed about CSG using regularized Boolean set expressions, we also discussed boundary representations and we discussed in detail sweep representations and Octrees. I must admit here that this is not exhaustive but these are probably most common one.

Out of them we will see soon find that the regularized Boolean set expressions with CSG and boundary representations are the most commonly used ones. Sweep representations are also used but there are limitations of those. And Octrees are very good to visualize but very complex to handle and represent. In addition to this, I would request you to read concepts about primitive instancing and use it on factor dimensions to define certain other solid objects.

Due to limitations in time and coverage requirement for other concepts in the computer graphics course I am not able to cover all the different aspects. But I have definitely covered the most four different aspects which are used for solid modeling. But there are other operations typically among them are the primitive instancing and fractal dimensions for representing solid objects.

We will talk about use of curves and surfaces later on which also used for representing solids when polygonal modeling is a bad approximation. So coming to the criteria in the last few minutes for comparing different solid modeling techniques the first one is the accuracy. And you read here that Octrees and polygonal boundary representations produce always an approximation of the objects.

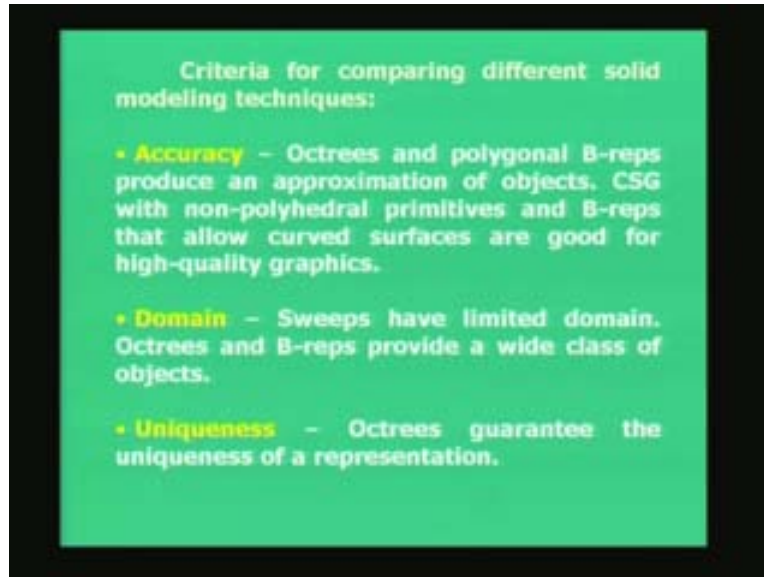
Octrees definitely will produce an approximation because if you look at an octree type of a structure, please go through the book by Foley Van Dam which gives good examples of those, you will find that the surface you will find edges of cubes. It will be a set of an accumulation of cubes put together to form as if a solid. So any surface boundaries will have edges or corrugated step wise bounding surfaces and those edges, vertices will appear very high corrugative in the case of Octrees. If the surface is not smooth, bounding surface is curved typically you will have lots and lots of corrugate. But of course however level of approximation you do by going to using the size of your finite volume the voxel unit for the Octree always you will have a corrugative surface. So, Octrees and polygonal boundary representations work with polygonal spaces or planes or approximation of objects.

CSG Constructive Solid Geometry with non-polyhedral primitives, if you can afford to do that along with boundary representations which also allow for curved surfaces are good for high quality graphics. This is what I was telling you that you need to represent curved surfaces which we will talk of later on but not now because you need to introduce curves and then talk of surfaces. But Constructive Solid Geometry along with B-reps which allows non-polyhedrol primitives and curved surfaces, if you can put all of them together then you are talking of high quality graphics to represent these solids. High regular objects can also be represented.

This domain sweeps have limited domain because it can very beautifully cover the different solids which you can generate a model for a sweep. Octrees and boundary representations provide definitely a large class of objects than sweeps. Well, in terms of uniqueness of the representations Octree guarantees the uniqueness of a representation because if you look at other types of representations then the problem which you will get is that it depends upon the approximation which you are taking of the polygonal vertices.

You can always split a quadrilateral into triangles or you can split triangles into different smaller triangles and all that.

(Refer Slide Time: 52:17)



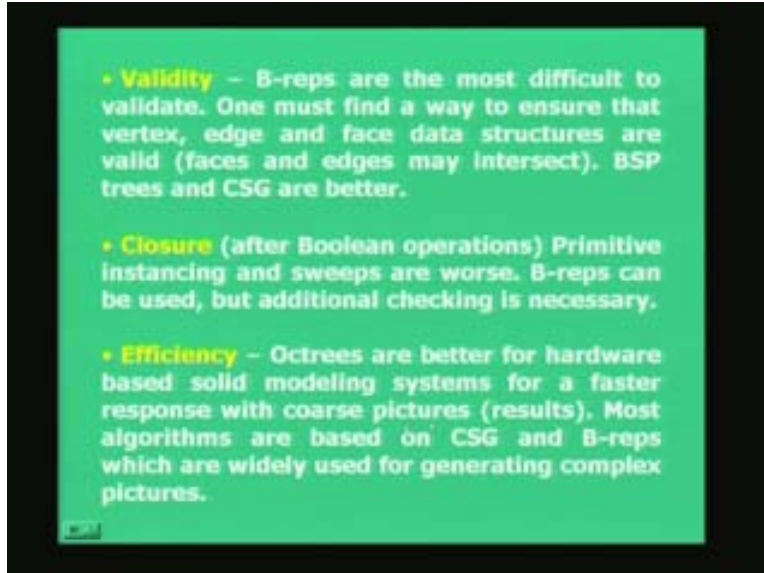
So the number of edges and vertices which will be used in boundary representations finally in your B-reps for representing a solid might change depending upon what level of splitting you need. Of course the same is true for Octrees. But typically the level of decomposition at the initial part when you are looking at sub volumes in different sub spaces one out of eight of those, those do not change much when you do not change the object structure.

Of course if you have changed the object structure in terms of scale, zoom or translate it, the Octree representation also changes. But as long as you are not doing the transformation and you are sticking to some degree of resolution in defining your minimum voxel unit then what will happen is the octree definitely gives you a unique representation. There is no doubt about it. That is not the case.

You see that the Octrees guarantee the uniqueness of a representation which may not be there for polygonal boundary representations or Constructive Solid Geometry because you might approximate curved surfaces with the help of polygons or you might use curved surface. So the uniqueness is not guaranteed, the validity that is the boundary representation are the most difficult to validate. That means you must find the way to ensure that the vertex edge and the face data structures are valid and the faces and edges may intersect in the case of boundary representations. BSP trees and Constructive Solid Geometry are better.

We did not discuss the concept of binary space partition trees to represent the solids. They are basically very good to represent object structures used for visible surface detection.

(Refer Slide Time: 56:53)



When we talk about that next we will see BSP is a very nice way or good data structures to represent objects. But we are talking of validity that means trying to find out if an object structure is valid. We looked at various constraints of the generalized Euler's equations with holes, number of vertices, edges, faces and holes count them. Count the number of holes which pass through the solid put them in an equation and make them valid. We also tried to find out how many surfaces share an edge.

How many edges meet to form a vertex? And also we need to find out whether those are all fine for solid polyhedron type of an object. But if we have a surface getting in to another surface then you may need to bring in some additional constraints to find that whether there are some extra surfaces which are just penetrating into the object and that is the validity which you need to hear about. So I read again boundary reps are most difficult to validate.

One must find a way to ensure that the vertex edge and face data structures are valid which you are using to represent. That means in the vertex table, edge table and surface table need to traverse to that to ensure validity. And then of course the closure, after the Boolean operations, primitive instancing and sweeps are worse under that category of the closure and the boundary representations can be used for if but additional checking may be necessary. The last one is the efficiency with we will wind up. Octrees are better for hardware based solid modeling for a faster response with coarser pictures. But most algorithms generally prefer the Constructive Solid Geometry and boundary representations and hence they are widely used for generating complex structures.

It is a question of trying to balance out whether you want to have a very fast animation or a very crystal clear picture, you cannot have both. You can have both definitely you are talking of very fast memory, very fast processor, multi processor, architecture, very large frame buffers and all that bandwidth crunch problems could be there in terms of loading

the frame buffer in terms of memory. So in terms of efficiency do you want a very crystal clear picture with very little or small amount of animation or do you want very fast animation, things moving very fast in the picture typically in video games then those characters and the solids may be approximated very coarsely.

You may not have very complex representations of those structures. So in terms of efficiency both in terms of speed as well as handling complex structures and let us say you want to provide high quality pictures. You cannot have both speed and that. So if you want to have speed but you are satisfied with smaller quality objects you better go for Octrees because you can have hardware which are suited for this but the picture will be very coarse, they will not be sharp and not many details will be available in the results. But you can make the pictures work very fast whereas for the algorithms based on Constructive Solid Geometry and boundary representations where you need complex pictures but very little animation you can use this to generate highly complex structures with lots of different complex solid objects put together.

This is where we end up the lecture on solid modeling where we will say this mostly towards the Constructive Solid Geometry or B-reps boundary representation which are mostly used.

Although I discussed about solid modeling but mostly I did spend time on these two the Constructive Solid Geometry and the boundary representations which are the most commonly used representations for solid modeling. We move on to the next lectures, we will see based on these solids at the last edge of the viewing pipeline what you need to do algorithmically to paint a particular polygon and give it a shade or a color. Thank you very much.