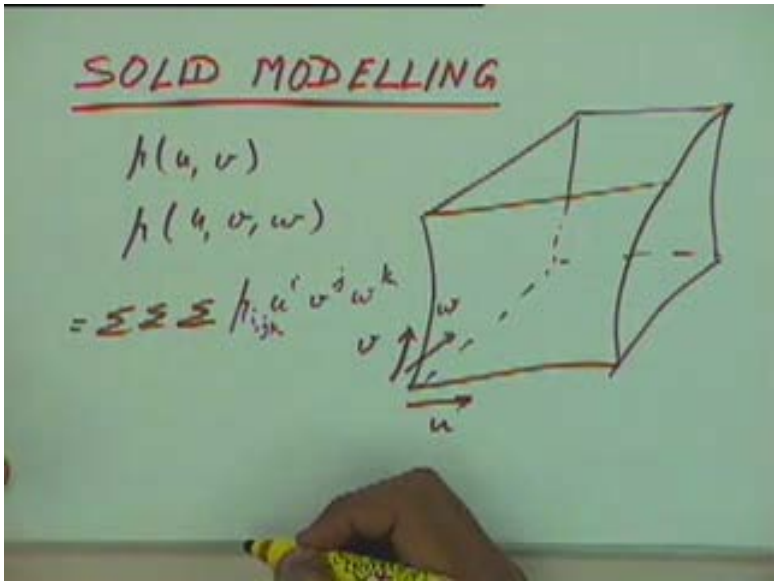
**Computer Aided Design**
**Prof. Dr. Anoop Chawla**
**Department of Mechanical Engineering**
**Indian Institute of Technology, Delhi**
**Lecture No. # 38**
**Solid Modelling**

Today we will be seeing methods for modelling solids. So far we have seen methods for modelling curves and surfaces.

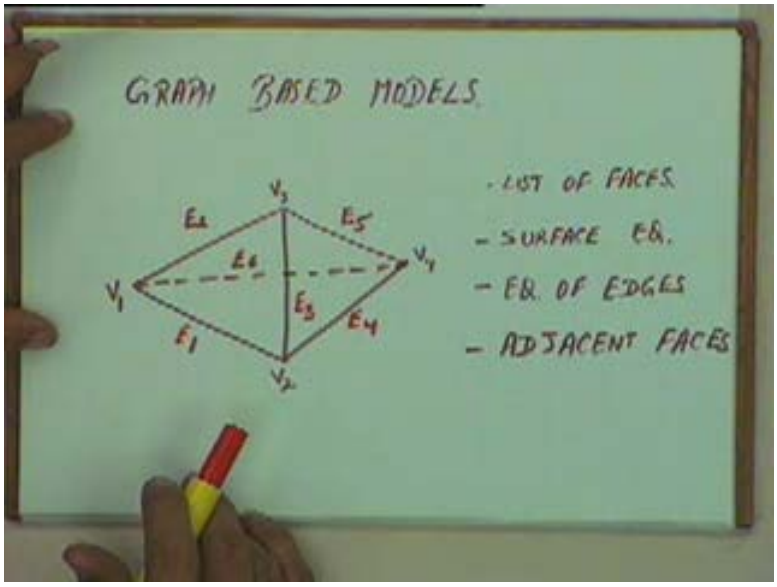
(Refer Slide Time: 00:01:06 min)




For surfaces we said that we can represent a surface in a parametric form by using two parameters whereas a cubic surface or Bezier or B-spline, it can be represented in a parametric form using two parameters. A natural extension of that to solids is to represent solid by having 3 parameters. So we can have solids represented in this manner where U V and W will correspond to let's say the 3 axis in some sense and then let's say U is in this direction, V varies in this direction and W varies in this direction.

So in this way we can define a three dimensional solid using three parameters but this kind of representation is very rarely used because solids, the way we use them in practice will normally not have any such mathematical definition. If for instance, if you write this P of U V W as you know a cubic form as let's say the sigma of $P_{ijk}$ U to the power of i V to power j W to the power k will get a tricubic solid. But such a solid would very rarely be used in any application. So the method we have for, we use for representing solids is a method let's say in the first method that we will see will be a graph based method or we used what is referred to as graph based models.

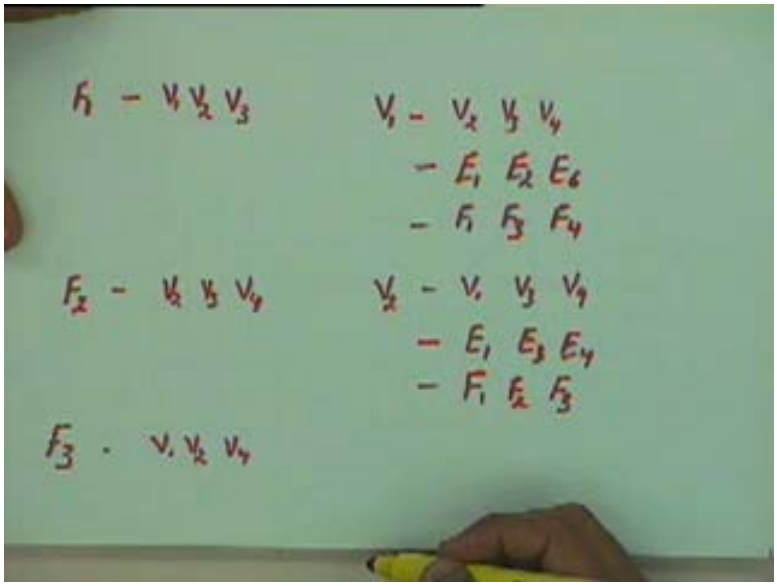
(Refer Slide Time: 00:02:50 min)




So, for in a simple case let's say take a tetrahedron like this and this consists a 4 vertices which are $V_1$, this is let's say $V_2$, $V_3$ and $V_4$. So we will represent solids by a list of faces, list of faces which bound the solid. And for each face, we will have its surface equation, the equation of the surface. We will also have let's say the equation of the edges, the equation and description of the edges. And we will also store that if we have a face let's say $V_1$ $V_2$ $V_3$ which are its adjacent faces, so adjacent faces and so on.

So essentially what we will do, we will take each face, with each face we will have the information on its geometry in terms of its equation and so on and information on adjacency which are the adjacent faces, which are the adjacent edges and so on or which are the defining edges and which are the defining vertices. We will use a representation like that. So let's see that for a simple tetrahedron that I have shown over here. It has a 4 vertices and lets number the edges as let's say this is $E_1$, call this edge as $E_2$, this edge let's say $E_3$ let's say this as $E_4$. This as $E_5$ and the one at the back, the dotted edge is $E_6$. So this tetrahedron has got 4 faces.

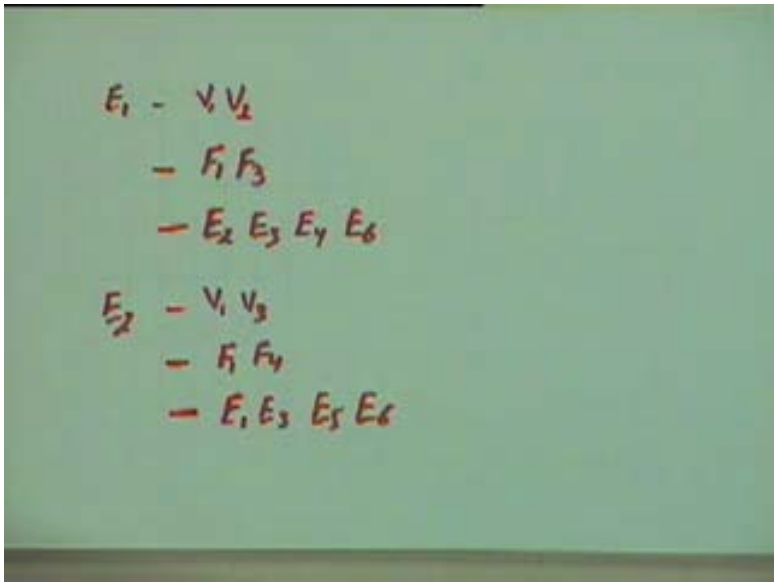
(Refer Slide Time: 00:05:20 min)




The four faces are say the face $F_1$ which is defined by vertices $V_1$ $V_2$ and $V_3$, $V_1$ $V_2$ and $V_3$ the first face over here. Then let's say a face $F_2$ which is $V_2$ $V_3$ and $V_4$. See let's say that is $V_2$ $V_3$ and $V_4$ and the two faces at the back which are let's say faces $F_3$ which is $V_1$ $V_2$ $V_4$ and similarly a face $F_4$. At the moment I am not going into that, so normally the order is important. We show them in an either clock wise or anticlockwise direction by convention but at the movement we won't go in to that. So these are 3 faces or the 4 faces which define the solid will be stored in this manner.

Now the next question is how do we store a vertex? If we consider vertex $V_1$, now this vertex $V_1$ if you look at it is adjacent to 3 vertices $V_2$ $V_3$ and $V_4$. So if you want to store the adjacency information for the vertices, it is adjacent to $V_2$ $V_3$ and $V_4$. Similarly this vertex $V_1$ is a part of 3 edges $E_1$ $E_2$ and $E_6$, so it will have the 3 edges $E_1$ $E_2$ and $E_6$ and this vertex $V_1$ is a part of 3 faces. The face $V_1$ $V_2$ $V_3$, the face $V_1$ $V_4$ $V_2$ and $V_1$ $V_3$ $V_4$. These are faces $F_1$ $F_3$ and $F_4$. Similarly if you take the vertex $V_2$, $V_2$ will be adjacent to $V_1$ $V_3$ and $V_4$. It will be a part of edges $E_1$ $E_3$ and $E_4$ and it will a part of faces $F_1$ $F_2$ and $F_3$. Yeah, I will come to that issue soon, I will come back to that. We are showing a lot of extra information at the movement, I will discuss that.

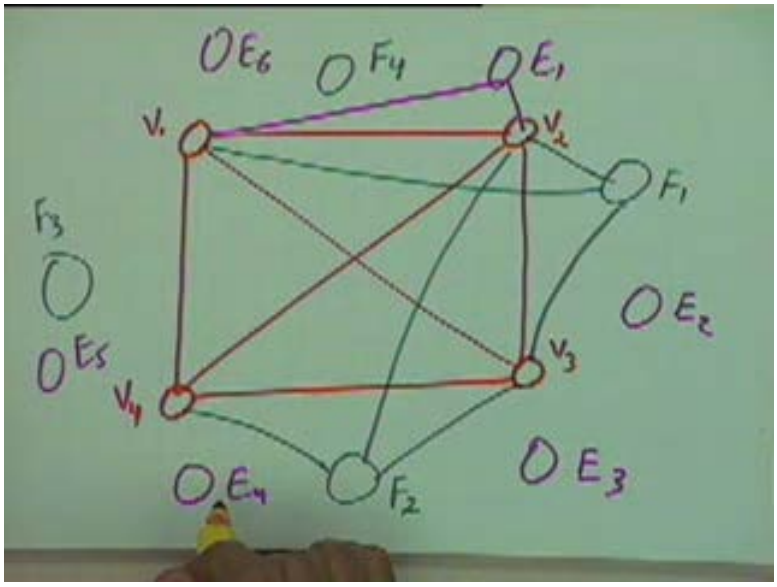
(Refer Slide Time: 00:08:13 min)




So this way we will have a list of vertices and similarly if you take up the list of edges, we will define the edges as let say the edge $E_1$ is defined between vertices $V_1$ and $V_2$. The edge $E_1$ is a part of 2 faces $V_1$ $V_2$ $V_3$ and $V_1$ $V_2$ $V_4$. So those faces are let's say $F_1$ and $F_3$ and the edges which are adjacent to this edge $E_2$ $E_6$ and $E_3$ $E_4$. So the edges adjacent to it will be $E_2$ $E_3$ $E_4$ and $E_6$. So in this, the edges adjacent to $E_1$ will be $E_2$ $E_6$ $E_3$ and $E_4$. So they are the 4 edges which are adjacent to this edge $E_1$.

Similarly if you take up edge $E_2$, we can get this side of vertices defining it $V_1$ $V_3$. The set of faces adjacent to it will be $F_1$ and $F_4$ and the set of edges adjacent to it would be $E_1$ $E_3$ $E_5$ and $E_6$ and so on. So this type of information for the faces, for the vertices and for the edges can define or solid completely. This gives us complete information about the solid. Of course in addition to this we shall be storing the corners of the vertices and so on. We will need the corners of the vertices, we need the equation of the planes and the other geometric details. In addition to the geometric details, this adjacency information is also stored in this type of solid model.

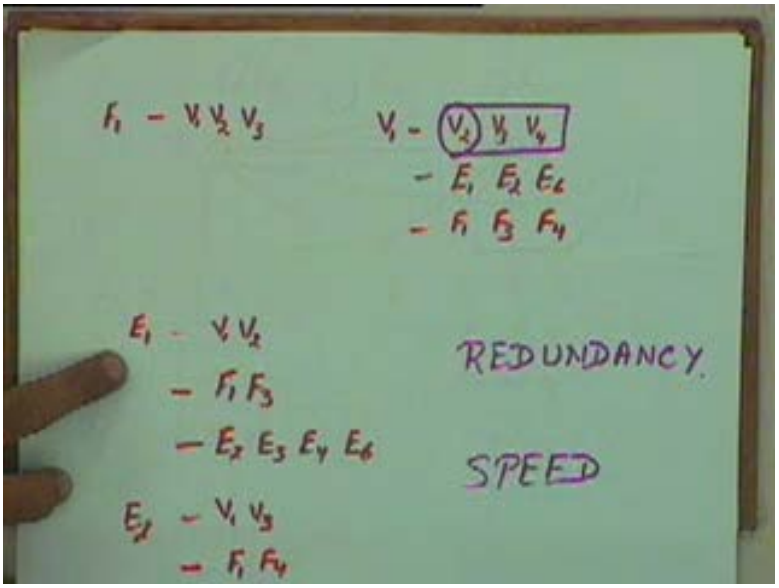
(Refer Slide Time: 00:10:30 min)




If we look at this information in the form of graph, what we are essentially doing that we are building a graph of the vertices edges and the faces. So if we have 4 vertices, we are saying that the 4 vertices are connected to one another. We have a 4 faces, so let's say this is face $F_1$ $F_2$ $F_3$ and this is $F_4$. So if we consider face $F_1$, it is defined between $V_2$ $V_1$ and $V_3$. Face $F_2$ I have taken to be between $V_2$ $V_3$ and $V_4$ so it is defined between $V_3$ $V_4$ and $V_2$.

Similarly on this if I add, I can add the details for $F_3$ and $F_4$ and then I have got 6 edges $E_1$ $E_2$ $E_3$ $E_4$ $E_5$ and $E_6$. So as this edge $E_1$ was defined between $V_1$ and $V_2$, it will have two faces adjacent to it, so there will be arcs connected to the two faces and so on. So this complete graph is what we can store for representing a solid like this. But now as someone just pointed out that if we look at this information, a lot of information is being duplicated. For instance we are saying that this vertex $V_1$ is adjacent to vertices $V_2$ $V_3$ and 4 that is also adjacent to edges $E_1$ $E_2$ and $E_6$ but this edge $E_1$ is defined between vertices $V_1$ and $V_2$.

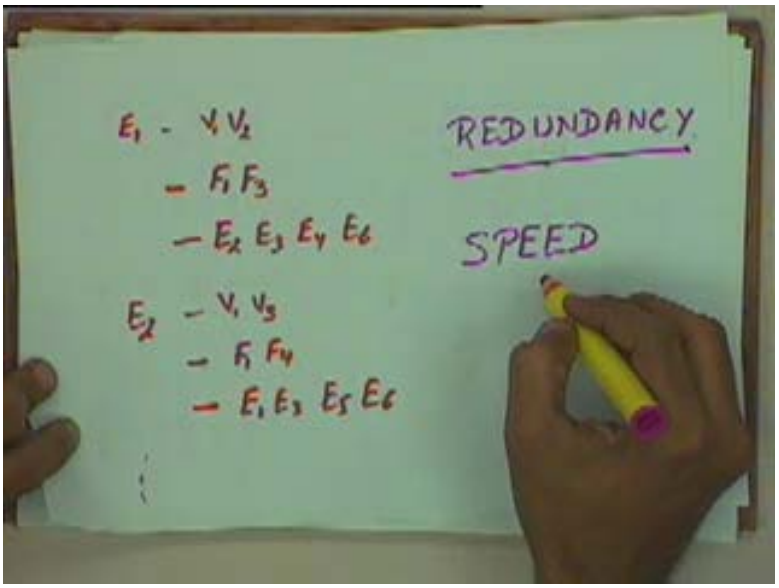
(Refer Slide Time: 00:12:54 min)



So if you look at this $V_1$ is adjacent to $E_1$ and $E_1$ contains $V_1$ and $V_2$ that means $V_1$ is adjacent to $V_.$ So therefore this information is hidden in it. Similarly this complete list is actually hidden in it. Similarly, if we go for this complete model you will find a lot of redundancy that means a lot of data is being represented more than once. If I want to find out the vertices adjacent to $V_1$, I will just look up the edge list and get the list of vertices which are adjacent to $V_1$. So this kind of model at the moment have a lot of redundancy but if we remove this redundancy, the drawback that we have on the other end is that if I find out the list of vertices I have to look at the edges and then look at the description of the edges and then get the list of vertices. So that is slightly time consuming. So the other end of it would be speed, so we have a tradeoff between redundancy and speed.

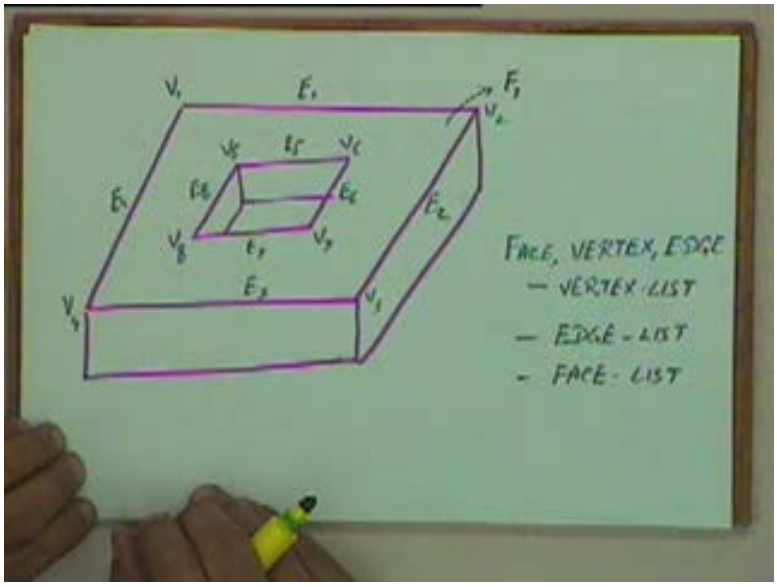
If we remove all the redundancy in this model, we will lose some speed and a time on speed in solid models becomes a very critical factor because lot of these solid modelling operations are quite slow. In fact if you have worked on geographics, you know that the time when you give a particular command and you have to wait for a very longtime. So lot of these solid modelling operations become very slow and at that time if this, if some of this information is directly available, it will help in speed but at the cost of redundancy. One of the biggest disadvantages of redundancy is that if the same information is being stored more than once, we have to ensure that the model is consistent all the time. If by error instead of $V_4$, I put a $V_5$ over here, the model will become inconsistent because this edge list will give us a different set of vertex list and actually vertex list being stored is something else. So that kind of error, that kind of inconsistency you have to be careful about. So in solid models of this type, we have a tradeoff between redundancy and speed.

(Refer Slide Time: 00:15:34 min)



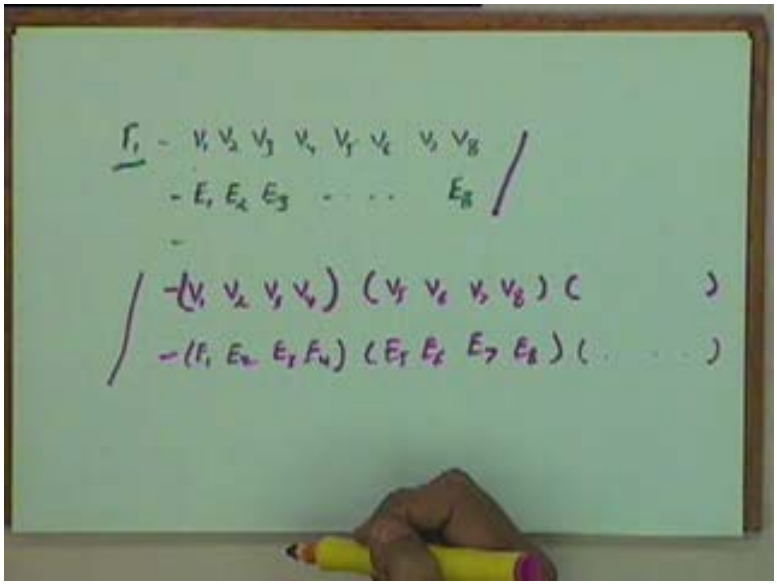

If we remove the redundancy, we will lose some speed and if we want very high speed we have to have a bit of redundancy. So in any actual solid model, how much redundancy is being used that is that will vary from model to model, that will vary from case to case. So models will have a lot of redundancy, some model will have a less redundancy. This tradeoff is a critical issue in the design of solid models or in the design of solid modeling techniques. Does that answer your question?

(Refer Slide Time: 00:16:30 min)




Now in these models let's now look at one more issue. If we have a simple block like this which has, let's say a rectangular slot or a rectangular hole. Now in this, if I want to represent the same model let's say consider this top face, I consider the vertices defining it $V_1$ $V_2$ $V_3$ $V_4$ $V_5$ $V_6$ $V_7$ and $V_8$. And similarly I say consider the edges which are defining it, $E_1$ $E_2$ $E_3$ $E_4$ $E_5$ $E_6$ $E_7$ and $E_8$ and we have a set of faces which are adjacent to this face.

(Refer Slide Time: 00:16:59 min)



If I want to represent this top face, the vertexes defining this face would consist of $V_1$ $V_2$ $V_3$ $V_4$ $V_5$ $V_6$ $V_7$ and $V_8$. This is a list of vertices which define this face, the top face. The 4 vertices of the outer boundary and the 4 vertices of the inner boundary. Similarly, if we consider the list, we get 8 edges and we will get a face list. See so far what we have been doing is that for every face we are storing a vertex list, we are storing an edge list and we are storing a face list.

And similarly for every vertex and for every edge we are storing a vertex list and edge list and a face list. So, if we consider the description of the face $F_1$, we have a vertex list, edge list and similarly we will have a face list. The problem that we have in this is that we are not explicitly storing that out of these 8 vertices, 4 of the vertices are forming an inner hole. So out of, from this vertex list or from this description of the face, if I want to find out the list of edges or if I want to find out which are the holes in it, the list of edges of the outer boundary, the list of edges of the inner boundary and so on, it's not very easy. It's not easy to identify a hole inside this face.

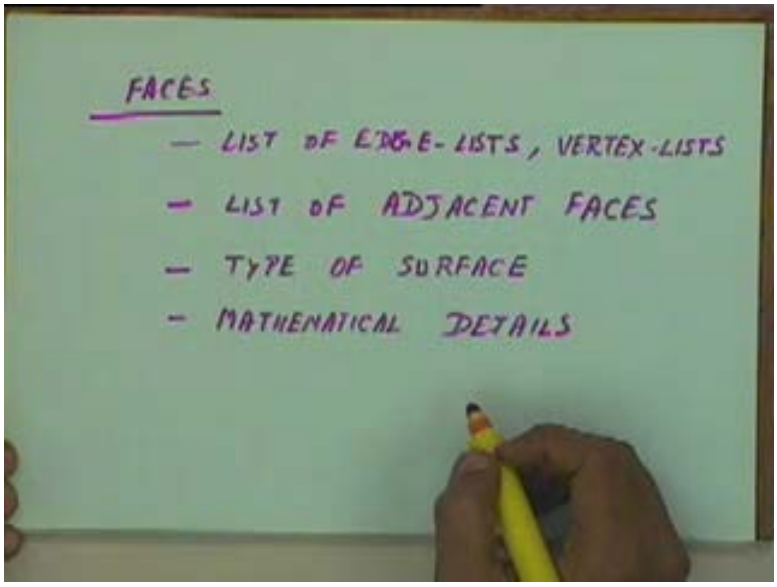
So, therefore the other alternative is that instead of storing it as a list of vertices, we will store it as two separate lists. The two separate list would mean, I will write it here. The first list contains $V_1$ $V_2$ $V_3$ and $V_4$ and the second list contains $V_5$ $V_6$ $V_7$ and $V_8$. So we have the first list is the defining the outer boundary, the second list is defining the inner boundary. The outer boundary is $V_1$ $V_2$ $V_3$ $V_4$ and the inner boundary is $V_5$ $V_6$ $V_7$ $V_8$. So we get an outer boundary and an inner boundary. Similarly, we might have a number of holes, let's say we have another hole over here, so then we will have another list over here. It can't be inside of this. You can't have a, you can't have something inside the hole. So we represent this as a list as vertex lists, list of vertex list, the first would be the outer boundary and all subsequent lists would be inner boundaries.

Similarly we will do for the edge list, we will say $E_1$ $E_2$ $E_3$ $E_4$ this will give us the outer boundary and then $E_5$ $E_6$ $E_7$ $E_8$ this will give us the inner boundary. And similarly if we have other holes, we will have another set of such lists. From the previous model meaning, you mean if we represent the face by a vertex list by this.

No, in this…. (Refer Slide Time: 22:28). Yeah, but if let's say if we want to find out the area of the face, how do you go about doing that? You need to know that there is a hole inside it. So for that we will have to look at the corners of each of the vertex and so on, so that will take time. So instead of spending time there, we prefer to store it as the list of vertices. Instead of this representing it like this, we like to represent a face like this. So that if we want to find out let's say the area of this or if you want the let's say the volume of this solid, we will need a complete description of the face which of course it is implicitly available in this also but it's not directly available. It will take a lot of competition to find out that there is an inner boundary and that kind becomes very slow.

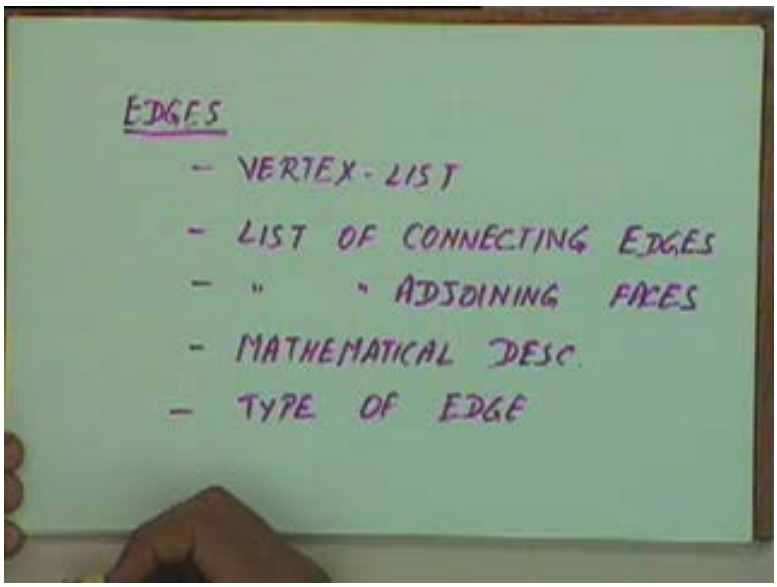
So for every face, every face will be represented as a list of vertex list and the list of edge list and so on. And that is again essentially done with the idea of increasing the speed at the type of competition. With this also theoretically we can always find out the inner edges and so on.

(Refer Slide Time: 00:23:51 min)



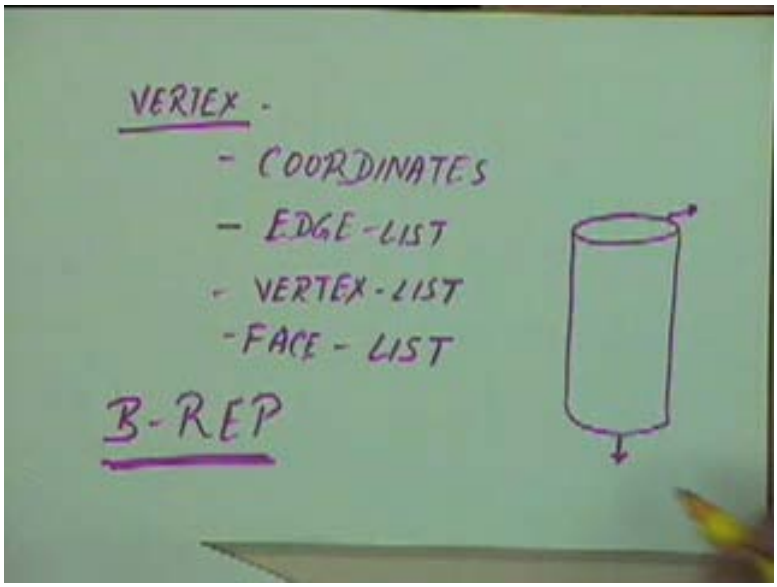

So if we summarize the information that we can store with the faces, the first is let's say a list of edge list and a list of vertex lists then list of adjacent faces. At the moment I am not going into the redundancy issues. Some of this data, we might like to remove in order to decrease the redundancy but this is a kind of information that we can store. Then the type of surface whether it is a planer surface or a cubic surface or a spherical surface, cylindrical surface whatever. And, then the mathematical description or the geometrical description of the surface. Maybe the equation of the plane, if it is a Bezier surface maybe the control points and so on, so mathematical details of the surface. Type of surfaces that can be a let's say cylindrical surface, it can be a Bezier surface, if you are combining it with the surface modelling technique or it can be a simple planar surface. Just a vertex list is not sufficient to describe that, so type of surface can become important. So with the faces we can store this kind of information.

(Refer Slide Time: 00:25:53 min)



Similarly if it top of the edges, we will have vertex list, we will have list of connecting edges, we will have the list of adjoining faces. And again we will talk of a mathematical description of the edge. Mathematical description, I am including everything. Mathematical description and let's say type of edge. So this information we can store with every edge and similarly for the vertex we have a coordinates and we will have an edge list, vertex list and face list.

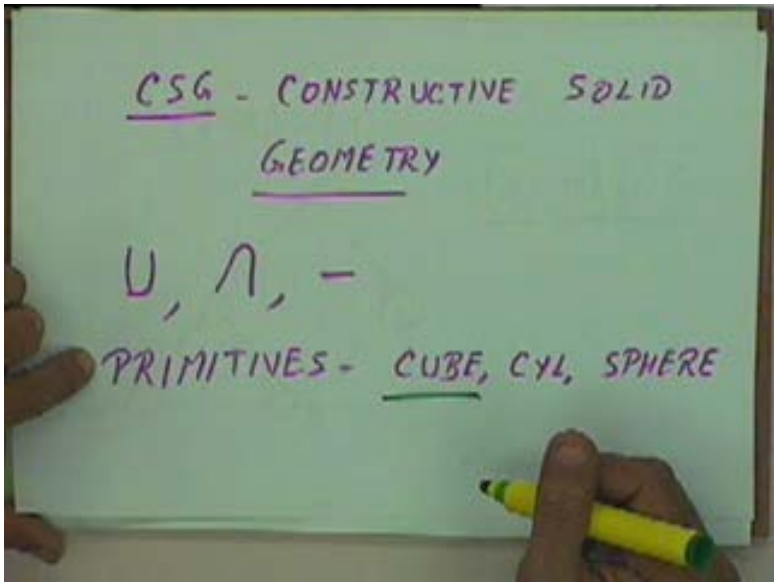
(Refer Slide Time: 00:27:02 min)




This we will be representing for every vertex. In this essentially a graphical representation and one of the standard graphical representation used is a B-REP model. How do you represent a cylinder? Let's say if we have a cylinder, you have one circular face and two adjacent faces to it. Now this circular face in the edge list you will get only two edges, one edge over here and one edge over here. Each of the edges can be circular because in the description of the edge will store the geometry of the edge, mathematical description and type of edge. Adjacent faces will be one face here and one face at the bottom and then we will store the mathematics of this surface. The problem then can come in sphere where you don't have any adjacent faces and any adjacent edges. So all that will remain and we will just get a mathematical description. So this is one method of representing solids.

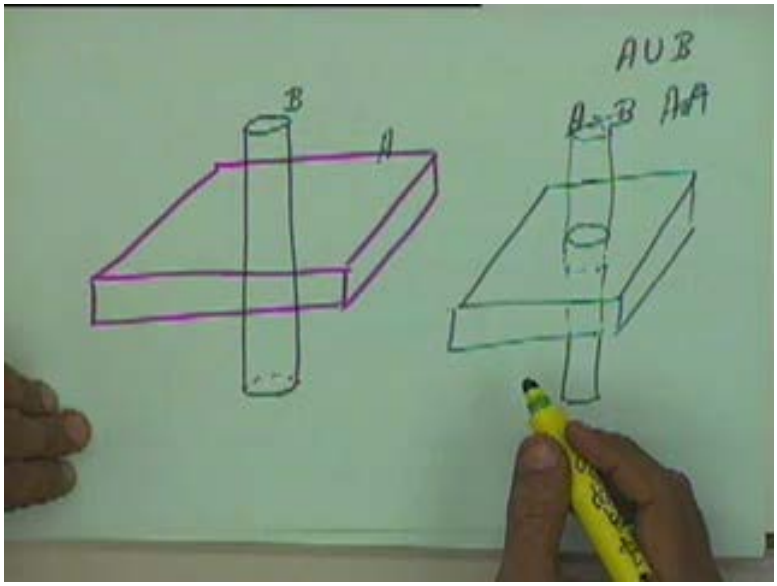
Now the problem that we have in this method is let's say if we want to make this data structure, this is if you want to give this graphical description or this data structure has to build for even a simple object like this, it can become quite complex because this complete graph or even some subset of it will be very difficult for the user to give all this information. And if we have a complicated object, let's say if we take any typical engineering object which will have let's say number of faces and number of edges then this can become a very complex method of describing the solid. Especially if you are talking of an interactive medium, giving the equation of each and every face separately can become quite complex.

(Refer Slide Time: 00:30:00 min)




So in order to have a simple method of handling the solids, we use another method that is referred to as CSG or constructive solid geometry, constructive solid geometry model. Now to ensure a family of geographics, you know the different types of Boolean operation that are available, this let's say union, intersection and the difference. If we consider these three basic Boolean operations and we take a set of primitives, let's say to start with we talk of primitives as maybe a cube and may be a cylinder and a sphere that's a standard primitives right now.
If we take primitives of this type and a set of Boolean operations, we can form a complicated solid also using these primitives and these Boolean operations.

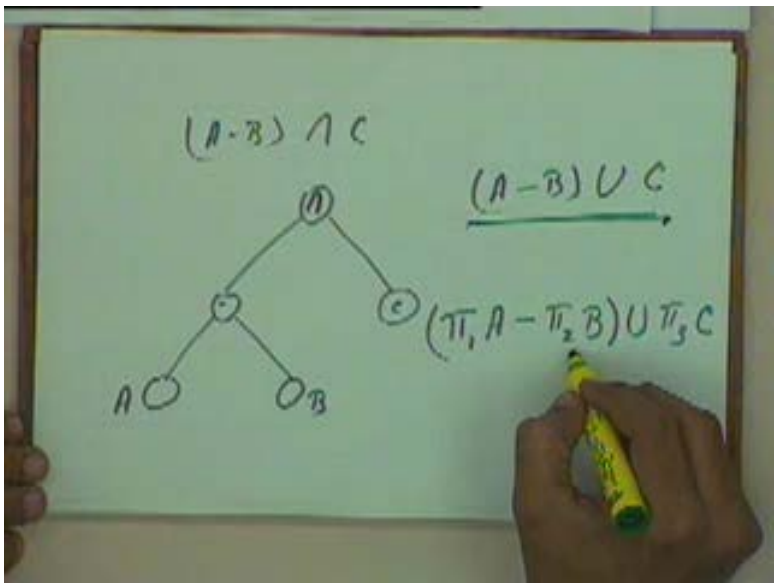
(Refer Slide Time: 00:31:14 min)




Again for instance if we take a simple block like this and we take let's say a vertical cylinder and I take the difference of the two. Let's say this is my solid A and this is my solid B. I will take let say A minus B, I will get block with the hole in it.

Similarly if I take let's say A union B sorry, I will get a block with the projection on either side. Block with let's say a sort of a handle attached to it and so on. So we can take primitives and then carry on Boolean operations on them.

(Refer Slide Time: 00:32:29 min)

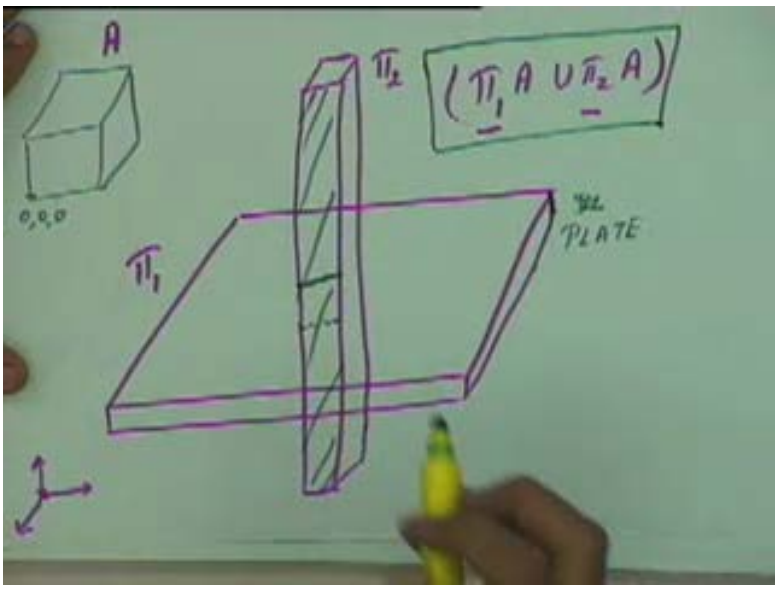



And then we can define let's say A minus B and its intersection with some other solid C and so on. So we can use solids defined in this manner and use them for further Boolean operations. So even a complicated object can be built using a set of Boolean operations and then this Boolean expression that can typically be stored in the form of a binary tree, in the form of a tree that means we will store it in. That means the leaf mode of this tree A and B are the primitives and all intermediate nodes will be the operations. So A minus B and its intersection with C can be stored in a manner like this. So if we want to make a complicated model, we can use a set of primitives and a set of Boolean operations and make that model.
Yeah, but the problem is that identifying such cases of redundancy is not easy. I mean you can vary out with a, by looking at it that operation was redundant but internally system is not very easy to identify that. We will just see that when we actually carry out a Boolean operation it's a very expensive process. Just storing this tree is straight forward.

We can easily store this expression. This expression in a form of this binary tree can be stored but from this binary tree if I want to find out, let's go into that now. From this binary tree if I want to find out the set of surfaces of this body, it can't be done easily because we have one block and we have a second object. For each surface we will have to find out intersection with each surface of the other so that can take a lot of computational time. So if let's say that you are saying we have A minus B and then I am taking a union with C but B is a subset of C. Identifying that B is a subset of C is not trivial. Then typically what will happen is that this kind of redundancy would get removed the moment you try to get a graph based model for this solid but even then this CSG representation would normally remain like this. Simplifying it is normally not attempted. So right now we have just three solids, so we can think of different combinations. Instead of 25 different solids, working of different combination of that is not straight forward.

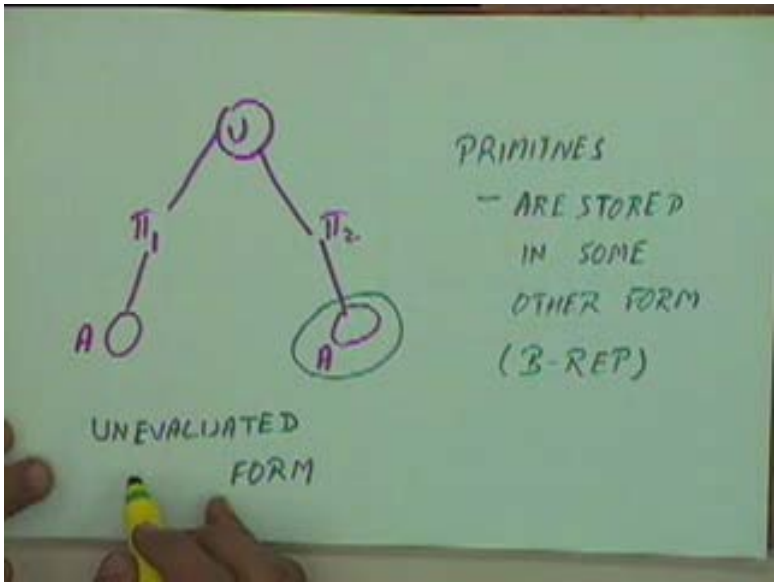
So now let's see a next thing that these primitives that we are using. Right now we talked of simple primitives like cube, cylinder and a sphere, cube cylinder and sphere. We can also have more complicated primitives before that instead of just taking the standard cube as the primitive, we can take a cube and transform it and get let's say a block of any arbitrary shape. Similarly we can take a cylinder and transform it and then get maybe a set of solids from that. So if you are including transformation, my binary or my CSG model instead of just looking like an expression in this manner can become let's say $pi_1$ which is some transformation of A minus that is $pi_2$ and some transformation of that which is B, of the primitive B and its union with let's say $pi_3$ times C. What this means is that we have a primitive A, we transform it by a project by a transformation $pi_1$. We have a primitive B, we transform it by a transformation $pi_2$ and then we take a different primitive.

(Refer Slide Time: 00:37:34 min)




For instance let's say we have only primitive which is a cube, so my primitive is let's say a unit cube. This is a primitive with let's say an origin over here. Now I want to define a plate and I want to take a rectangular rod and I want to find out let's say the union of these two. And I want my origin to be let say some point over here. The first thing that we can do is we can take this cube, transform it into this plate that can be done by some transformation. So let's say this primitive cube is my primitive let say A and the transformation required to transform it into this plate is let's say $pi_1$. And then I take the same primitive A, transform it to get this vertical rod through a transformation let's say $pi_2$ and then maybe I say $pi_1$ of A union with $pi_2$ of A.

(Refer Slide Time: 00:39:20 min)




And if I want to represent it as a binary tree, we will have let's say A, it goes through a transformation $pi_1$. Again we take another instance of A and this goes through a transformation $pi_2$, we take the two together and we take the union. So let's say in geographics when you are taking the primitives and placing them at a particular point and so on, you are actually taking a transformation of a standard primitive. So you can define transformation on primitives and then can define a Boolean operation on top of that.

Now if you look at this solid we have two solids, one is this plate, the other is this rod. And we want to find out the union of these two. Let's say we have for the sake of display, I want to display this combined solid on the spring. For purpose of display, I need to know all the surfaces of the combined solid. If I talk of surfaces of the combined solid, in that my final solid will have all the surfaces of this solid plus it will have the surfaces of this solid which will be split. That means this vertical surface if I consider this surface, this will get split into let's say 3 parts one here, one at let's say the bottom face and one below it. So, we have one surface here, second surface is this surface and third surface is below it.

So what we need to do is we will have to take this surface, the vertical surface, find out its intersection with the surfaces of the solid. Let's say the solid is we will see this block or plate. I will take this vertical surface of the rod and find out its intersection with all the surfaces of the plate. We will find two intersections, we will take this surface and then split the surface into three parts and then decide which surface will be included in the final solid and which will not be included. So this complete process can become quite expensive and then this will have to be repeated for each of the surfaces.

This solid has got 6 surfaces so that we will repeat it for each of the 6, so that can become a very complex process. But just by looking at this, just by looking at this description, it is not possible for us to state as to whether the or it is not possible for us to display the solid. If you want to find out any property of the solid, we cannot do that just by looking at this binary tree. We need to know the equations, the description of the each of the surfaces which defined a combined solid. So this model, this binary tree is normally referred to an unevaluated form of the solid model. It is referred to as an unevaluated form of the solid

model or in this unevaluated form these primitives, they are stored in some other modelling technique but since you are saying that these are primitives, we need some other representation for these primitives. So primitives are stored as or stored in some other form, for instance B-Rep. So if you are using this as a modelling method, the primitives will be stored in some other form and then wherever you want to use this model, we will have to evaluate the model and then use it for our applications. So this tree itself is referred to as an unevaluated form and then from this unevaluated form, once we get the description of each of the faces and make a detailed model that is referred to as the evaluated form. The advantage of the using this form is it gives us flexibility, it is very good for interaction. For a user who is working on a system, it is very easy to form a solid using Boolean operations.

He can visualize the object easily and he can easily define that this is a plate and I want to make a hole in it. So it is good for interaction, it is easy to handle the solid in this manner. Storage and retrieval is easy, I can just store this expression and that's it. I don't have to store the complete graph model. The complete graph model will be quite complex for reasonable solids. So storage and retrieval is easy, interaction is easy, editing is easy and so on. Yes, yes for, let's say for displaying you normally have to convert it into some other form. For findings out its properties, mass, moment of inertia or whatever we will have to convert into some other form and so on. So this method CSG gives us a good method for interaction, good method for defining the solid. Any other question on this? So I will wind up now. In the next class we will go on from this point and see other details of these modelling techniques.