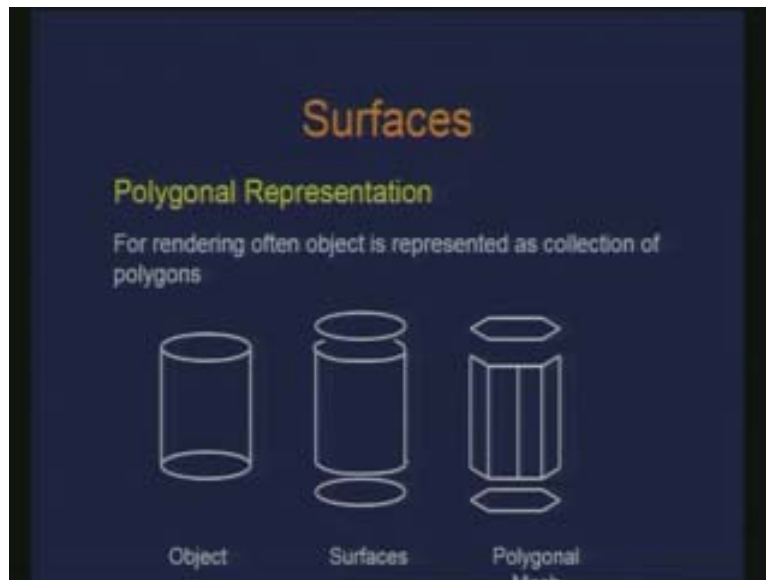


Introduction to Computer Graphics
Dr. Prem Kalra
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi
Lecture # 19
Surfaces (Contd....)

We have been talking about surface generation. Last time we actually looked at methods of displaying surfaces in terms of its representation. So parametric surfaces could possibly be rendered by either breaking them into piecewise linear or co planer elements or you have subdivision of the parametric surfaces. So what we looked at basically was the polygonal representation of objects and the motivation to have polygonal representation could be many. And one of the main motivations is the rendering of polygons in general is hardware supported and also for the acquisition of data from various scanners.

(Refer Slide Time: 02:05)



Therefore you primarily get the point clouds for which you establish the connectivity and form triangles or any polygons. That is the kind of data for which you would like to again use a polygonal representation. So what we basically looked at is that if there is an object of this kind it could actually be made of several surfaces. So in this case there is a surface which is the sides of the object and then there are caps of the cylinders.

And each of these surfaces in turn could be represented as polygons or collection of polygons which we refer to as polygonal mesh. So this is represented using several polygons and this top and bottom are again represented by polygons.

And often for the purpose of simplicity we use triangles as polygons because we can always break down the polygons in to triangles and computations on triangles are again

more efficient. So what we basically looked at is the various ways of representing data structure pertaining to these polygons or the polygonal meshes.

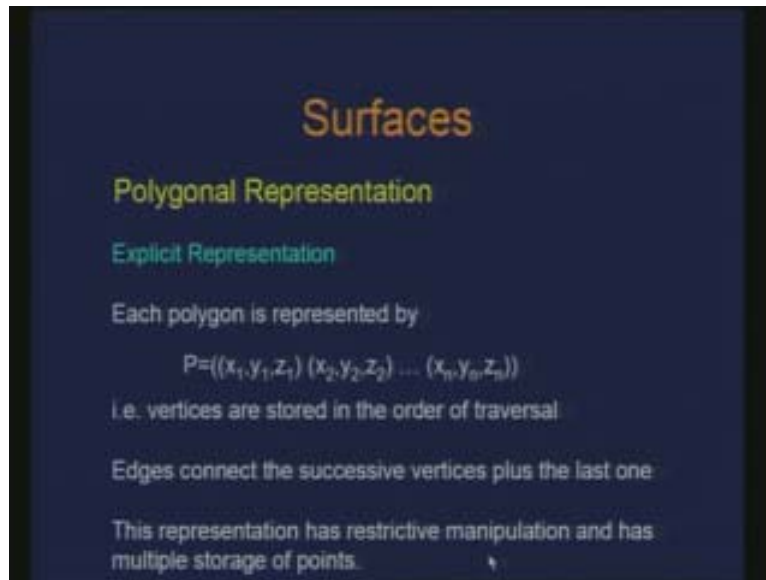
When we are talking about polygonal representation we are looking at representation of the edges which connect the vertices and the polygons which again is a sequence of edges.

(Refer Slide Time: 04:20)



So the explicit representation we looked at is basically a way of representing a polygon as a collection of individual vertices to form that polygon. And you write it in a sequence so that the last edge which will be formed is between the last point and the first point. Therefore the coordinates of each of these vertices is given explicitly.

(Refer Slide Time: 05:05)



Therefore the limitation of such a representation is that the manipulation for such a polygonal mesh is restricted if you want to delete a point or delete an edge you need to get more information so that you can localize your manipulation. In this case it is very difficult. So you will have to again re-form the whole list of vertices to get the polygon. So we looked at incremental enhancement to this representation which is explicit representation to add this facility of manipulation.

The next we looked at was basically a pointer to a vertex list which is very commonly used. So what we have is a table of vertices where all the coordinates which are used for defining the mesh are listed in this manner and it is just a table. Then you have polygon just as an entry reference to that table which could be like V_1, V_2, V_3 these are just the index numbers. In this case if I have P_1 it is just 1, 2, 3 these are the entries in the table.

Now again this representation all though facilitates some manipulations or to the mesh but if I am interested in finding out what are the polygons which are sheared by an edge it is difficult to answer because I actually have to go through the entire list to be able to derive this information.

(Refer Slide Time: 06:17)


Surfaces

Polygonal Representation

Pointer to Vertex List

Each vertex is stored once in a list V
 $V = ((x_1, y_1, z_1) (x_2, y_2, z_2) \dots (x_n, y_n, z_n))$

Each polygon is represented as
 $P = (V_1, V_2, V_3)$
 e.g. $P_1 = (1, 2, 4)$ and $P_2 = (4, 2, 3)$



In this representation it is difficult to find polygons that share an edge.

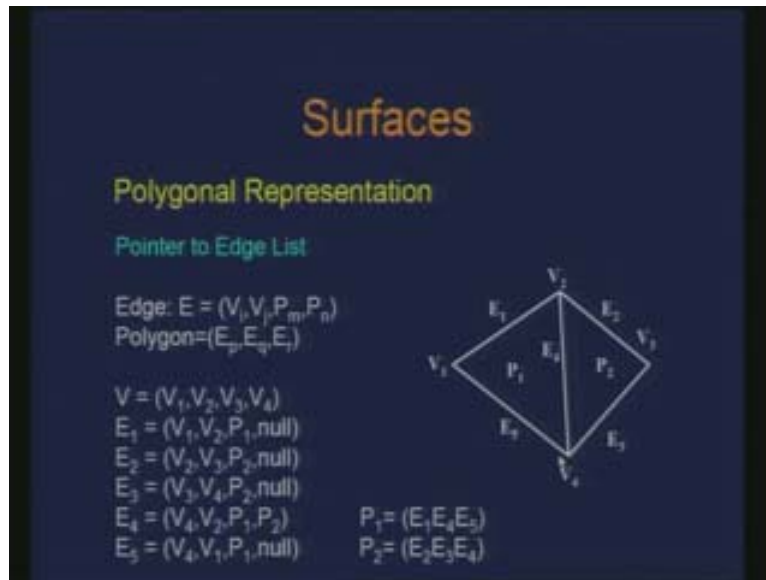
So what we do is we try to modify this data structure where we facilitate this manipulation. And also there is a notion of redundancy. In the explicit representation there were lots of redundancy, points were repeated just to define an individual polygon.

So, that is another parameter or aspect for deciding the data structure you want to use. Now in order that we facilitate the adjacency information to get the information about what are the polygons sheared by an edge we modify the data structure which is sort of edge based and we build an edge list where an edge is defined in terms of the two vertices it joins and the polygons which are sheared by this edge.

Therefore the whole information is available.

In the case of a triangle the polygon is nothing but a collection of three edges which constitute the polygon. So in this case if I am interested in finding out polygon P_1 it is nothing but $E_1 E_4 E_5$. And similarly the individual edge here E_1 is defined through the point $V_1 V_2$ the polygon which is sheared, since there is only one polygon here I just write P_1 and for the other polygon I just say null. In a way it also gives you information if I am interested in finding out a boundary of the polygonal mesh. Through this representation if somebody asks you to find out the boundary of the polygonal mesh, it is very easy. You just scan the edge list and find out what are the edges which are sheared by only one polygon and you have the answer.

(Refer Slide Time: 09:00)



A slight variation could also be done if I am also interested in finding out holes. Hole is also some sort of a boundary. May be you need to have an orientation in which you traverse the list. So, this is in order to distinguish between hole and a boundary. You can always get hole and boundary using this data structure.

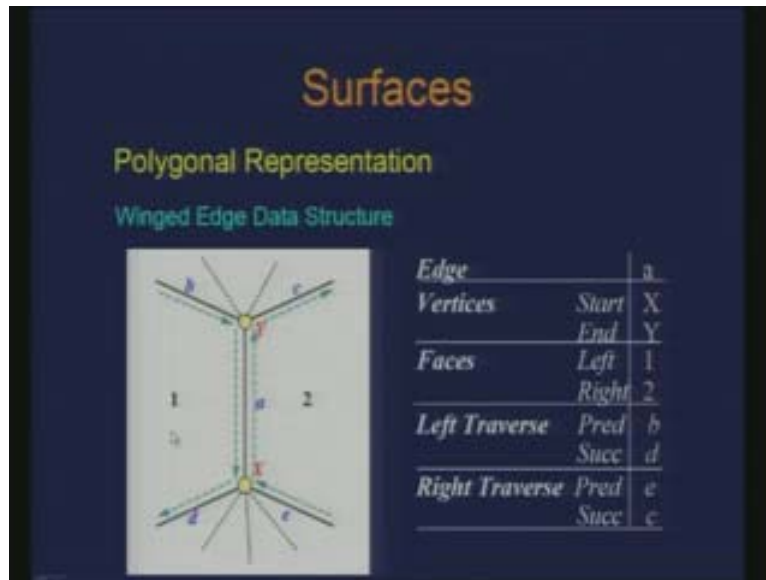
And then again the problem is that if I am interested in finding out more information in terms of what edges are incident on a vertex that is again difficult to find out. I will have to again traverse through the entire data structure. Therefore for that we looked at another data structure which was winged edge data structure where the information was added also about the auxiliary edges to the vertices.

There is an edge for which the data structure is written which is defined between X and Y and you also have additional information about the edges which you get through the traversal of the polygon sheared by this edge a.

For instance, you have information about the predecessor edge for the polygon which is obtained by left traversal and this succeeding edge b. Through this now if you are interested in finding out the edges which are incident on a vertex is relatively easy. You do not have to travel through all the mesh. So again the overhead which you pay is in terms of the information you store so the data structure becomes heavier.

What we are trying to do is we are trying to actually derive some sort of a navigational scheme. So I go from here to here so I have a notion of how I am traversing these edges in a polygon. Thus, if I look from this side I am actually going in a clockwise manner and so is the case here. So my traversal is always clockwise about a polygon. And I am again basing my data structure on the edge.

(Refer Slide Time: 12:19)



This is the connectivity link between points. That is what establishes the connectivity of a mesh. So I just traverse through that and I can derive information about the polygons which would be adjacent to an edge or sheared by that edge. And I also have the explicit information about what vertices the edge constitutes so X to Y I know that.

The additional information for which I am interested at any point of time, for instance getting the edges which would possibly be incident here on this vertex just a traversal scheme would give me the information about the edges here because this edge is given by this traversal, this edge is given by this traversal so from here now I can go to the other traversal that is this polygon and so on. So the number of polygons or edges which I need to traverse is limited. It is not a function of the number of edges in the mesh. So in some sense it is again Constantine. Now we will basically look at the models which can get formed using the combination of surfaces. What are we trying to do is ultimately we are building these models which could possibly be the solids.

(Refer Slide Time: 14:25)

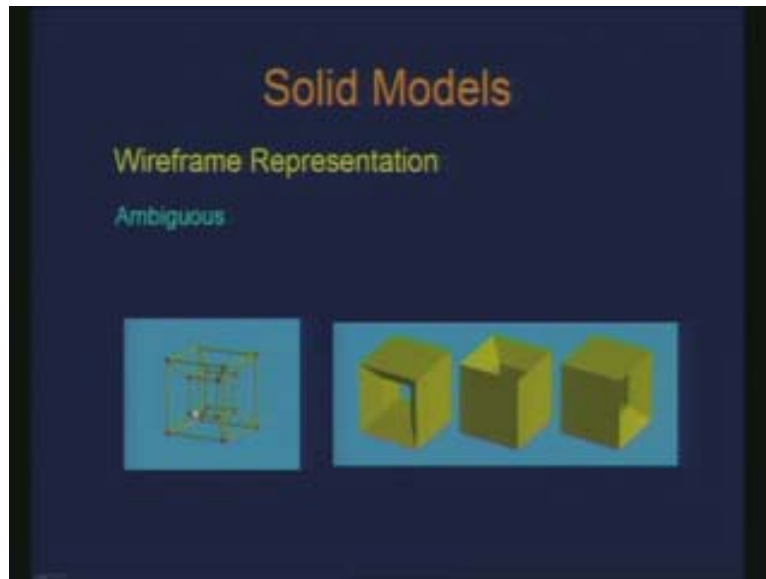


So we have the solids constructed using the surfaces which are the boundaries of that solid. So eventually if we are doing this CAD CAM kind of an application we deal with solids. And surfaces could be the boundary representations of those solids. So if I have a representation for solid models using surfaces as its boundary then I try to make this solid models using wireframe representation. So when I am talking about wireframe representation I am looking at two data structures or two kinds of information which would give me the vertices which are there in the solid and information on how those vertices are connected so it is only the connectivity of points.

In this case if I have an object of this kind a cuboid then all I am having as the information is these vertices 1 2 3 4 5 6 7 8 the red dots and these links. And I render them using only these links. There is a problem with such a representation? This is the way I am going to display it using the links of the vertices. Now the question is, is there any problem with this?

In some sense it is it could be ambiguous. For example, if I have a wireframe representation which is given like this the only thing I have is the information about the links joining to the various points. Then this could be like this, this could be like this and this could be like this so there is lot of ambiguity about what do I see from here. Therefore, just having the information about the vertices which are giving you the boundary of a solid and the respective links is inadequate to be able to know what solid could possibly be there.

(Refer Slide Time: 17:03)



Now what we are looking at is some better boundary representation. In fact the polygonal representation is also a boundary representation. What we are trying to say is that now we will have three elements vertices, edges and faces. In a more generalized sense these edges could be curves and these faces could be surfaces. But we need information about these to resolve the ambiguity of a solid. Now when we are talking about these boundary representations for solids often we actually consider those boundary for the solids which are manifolds.

What do I mean by manifolds? It is basically some sort of a well behaved solid. It is just a qualitative notion. Mathematically what we trying to say there is that it should satisfy some condition so that I have the surface which is a manifold. That condition is basically that for each point if I consider on the surface say point x there exist an open ball open ball means some sort of a sphere with a centre x which is defined using a very small radius some r so that the intersection of this ball and the surface can be continuously deformed to form an open disk.

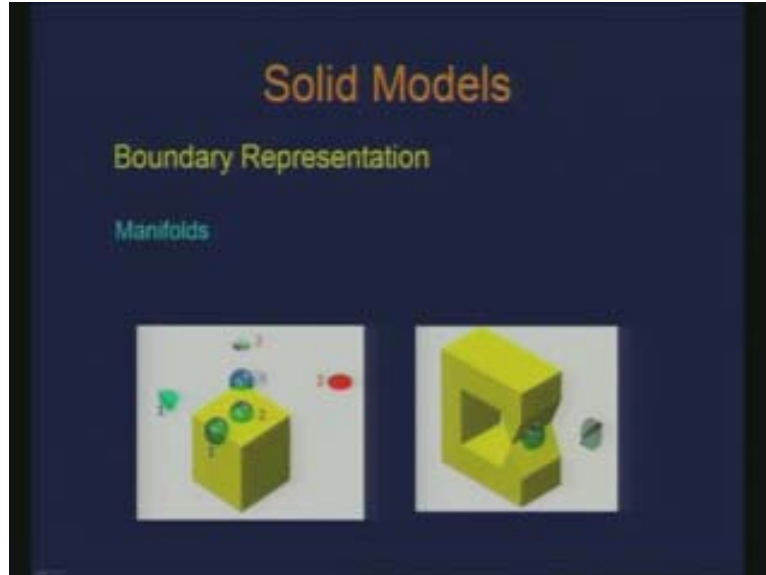
(Refer Slide Time: 20:08)



Then I have the boundary of the surface which is manifold or sometimes also called as two manifold. Here is an example of a cuboid which is two manifold. At location two I define a an open ball a spherical ball the center of which is some point x which is on the surface, now what do I see is what I get as an intersection of this to the surface of this boundary which is a disk as given here with the red, and that is what I mean by open disk. This should happen everywhere. Open disk is something you get after some deformation also. It should be possible to get an open disk even using some deformation.

Now, if I consider here at the edge, I embed this open ball. Now if I look at this location this is what you will see. This is nothing but some sort of a bent disk which I can again twist it right to get the open disk. if we go to this corner again when you see the intersection of this open ball with the surface here at this point you get a three fold kind of a disk and again you can turn it or twist it right to get the open disk.

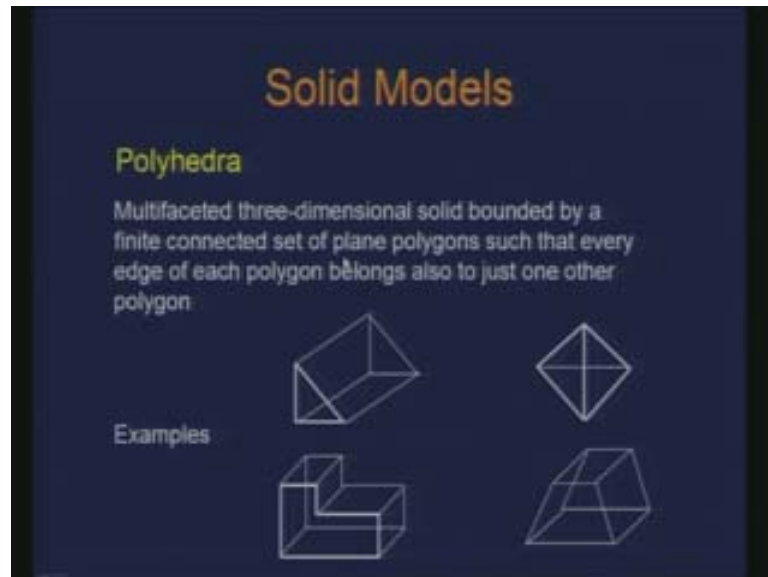
(Refer Slide Time: 20:59)



So all we are saying is that we should be able to get an open disk after some deformation when I embed an open ball. That is what the condition is for two-manifold. When you do not have such a condition we do not call it as two-manifold. Here when we look this is what you find. What you get here is in fact two disks and you can get an open disk only by gluing them together and not by deforming the same disk. Therefore we consider that as a violation of that condition and hence this is not a two-manifold. Many of the models which are generally considered for modeling are manifolds.

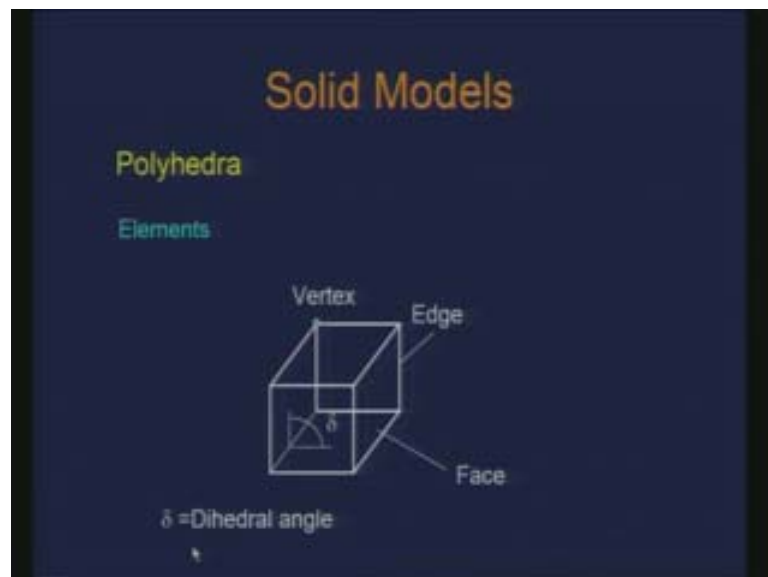
Examples of polyhedra or polyhedral objects: this is where we will be able to have these boundary representations for the solid models. Polyhedra are multifaceted 3D solid bounded by a finite connected set of plane polygons such that every edge of each polygon belongs to just one other polygon. So you have the edge which is shared and this is actually two polygons. Some of the examples are this, we have this. So again you can have both convex and non convex polyhedra. These are the examples of convex polyhedra and this is the example of non convex or concave polyhedra.

(Refer Slide Time: 25:31)



But if you look at each of these edges these are exactly sheared by the polygons. What are the elements which we use for defining polyhedra? We have a vertex, we have the edge and we have the face. This is what we need for boundary representation of a solid. And we also have the angle which is formed at the edge between the two faces of the polygons which we call it as Dihedral angle.

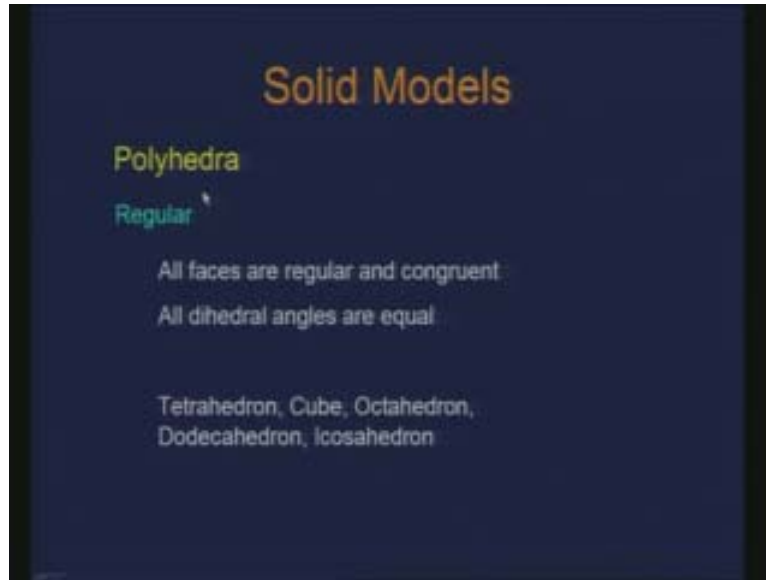
(Refer Slide Time: 26:13)



These are the elements for defining a polyhedra. So you have some special kind of polyhedra which you might be also familiar with. There are regular polyhedra which

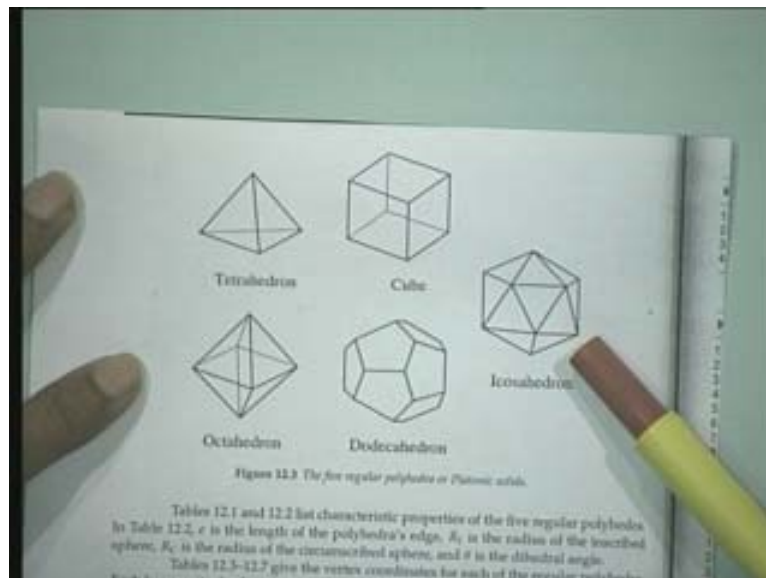
require certain conditions to be imposed that all faces are regular and congruent and the dihedral angles are equal.

(Refer Slide Time: 27:01)



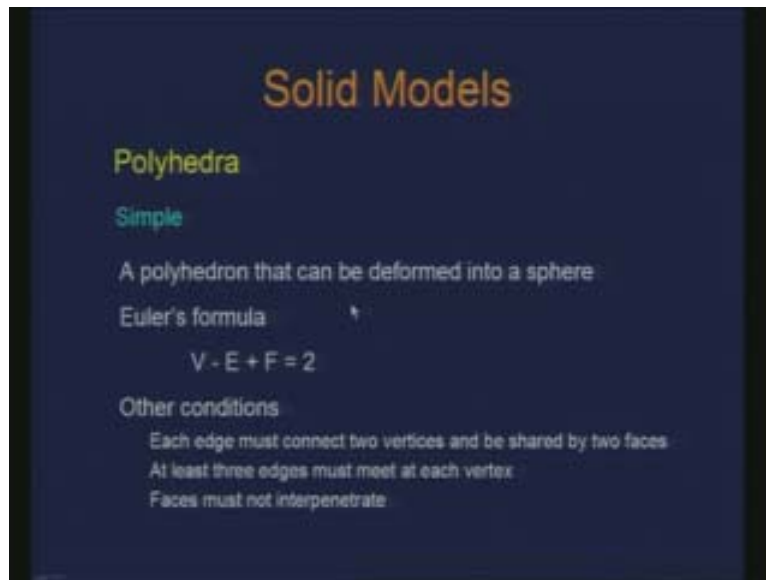
So the examples of these regular polyhedra are tetrahedron, cube, octahedron, dodecahedron, icosahedron so these are also called as the five platonic solids. You have this tetrahedron, the face of a tetrahedron is actually a triangle and there are four faces tetra, total number of faces, this is cube, this is octahedron.

(Refer Slide Time: 27:53)



So here we have the type of the face polygon as a triangle and we have eight faces. So, when you see each of the polygons they are regular and congruent and the dihedral angles are same. Similarly, you have dodecahedron for which you have the total number of faces as twelve and this is icosahedron and the total number of faces is twenty. These are also called as platonic solids. These are some special polyhedra. There are also some polyhedrons as simple polyhedrons that we have. A simple polyhedron can be actually deformed into a sphere or we also call it as homomorphic sphere. That means I can stretch and just consider that solid or polyhedra like a deformable elastic type of a material which you can deform to form a sphere.

(Refer Slide Time: 29:22)



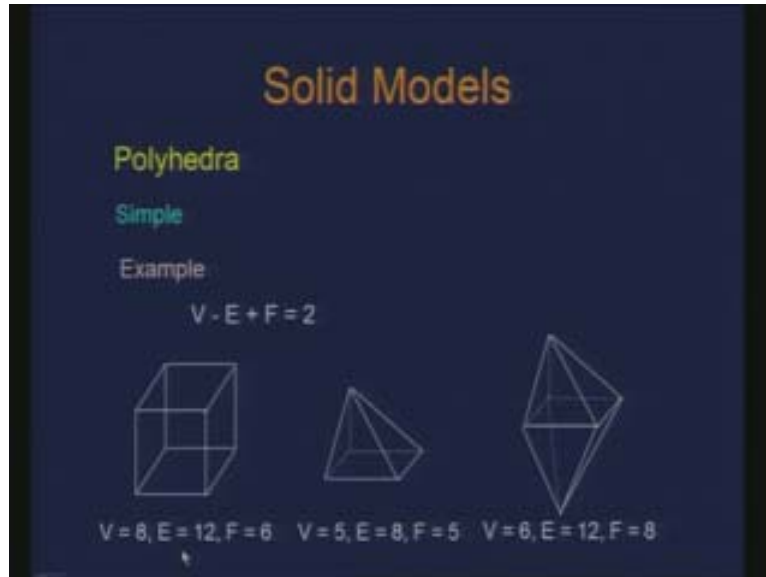
Now there is a certain property which is maintained if you have a simple polyhedra and that is given through what is called as Euler formula. So there is a relationship between the number of vertices, the number of edges and the number of faces which define a simple polyhedra and this relationship is given here. V is the number of vertices, E is the number of edges and F is the number of faces.

So if I take V minus E plus F it turns out to be two for simple polyhedra. So this is actually a necessary condition to have a simple polyhedron defined but there are other conditions which need to be satisfied so that you have the solid defined. For instance, you need to have additional conditions such as each edge must connect two vertices and can be shared by two faces and at least three edges must meet at each vertex and faces must not interpenetrate. These are additional conditions for you to have the definition of a solid.

Now let us see an example where this Euler number the relationship of V , E and F we observe as satisfying. Here is an example so you have a cube again where we have the number of vertices as 8, the number of edges 12 and number of faces as 6. So if you just consider this V minus E plus F you get 2. And we basically can see this as getting

deformed into a sphere. Consider the sphere in a very coarse resolution, it could possibly look like a cuboid.

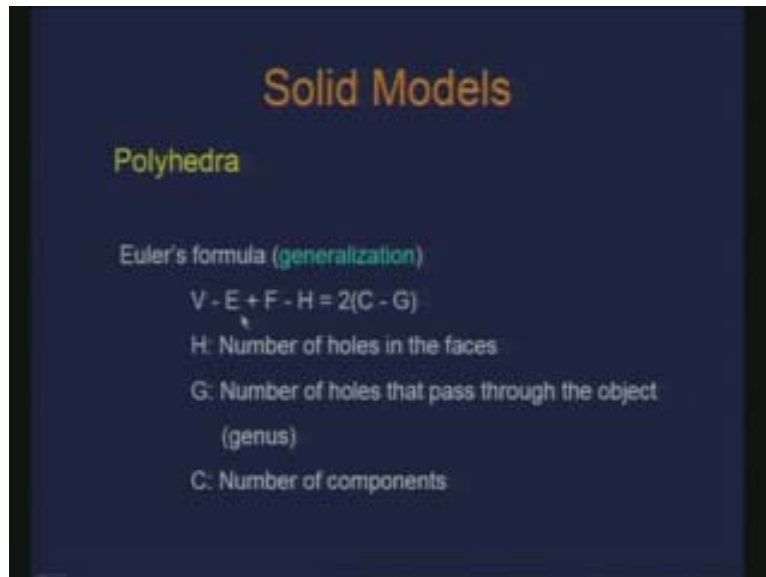
(Refer Slide Time: 31:44)



Similarly, you have for tetrahedron which is also simple polyhedra where the condition of Euler number getting satisfied. The polyhedra which we are considering do not have holes on its faces and there is not a hole through and through in the object, it is a closed surface solid.

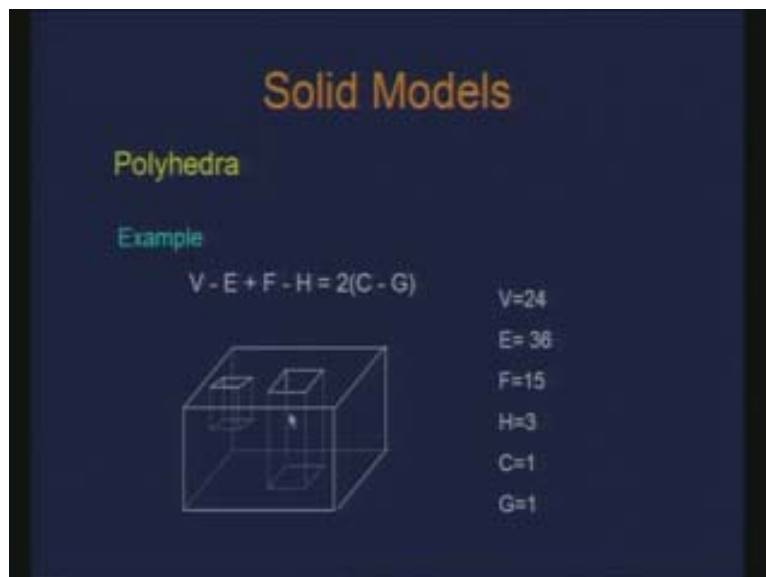
What would happen if we consider them? There is a generalization of the Euler's number where you can also incorporate the holes which are there on the faces. The number of holes that pass through the object often refer to as the genus of the object.

(Refer Slide Time: 33:18)



For instance, sphere has a genus 0, torus has a genus 1 and that is what we mean here and C is the number of connected components. So, if you want to incorporate this information then the Euler number is modified like this. Here is another example, I have a solid like this so this is a through and through hole and this is just a hole at the top and it is not a through and through hole.

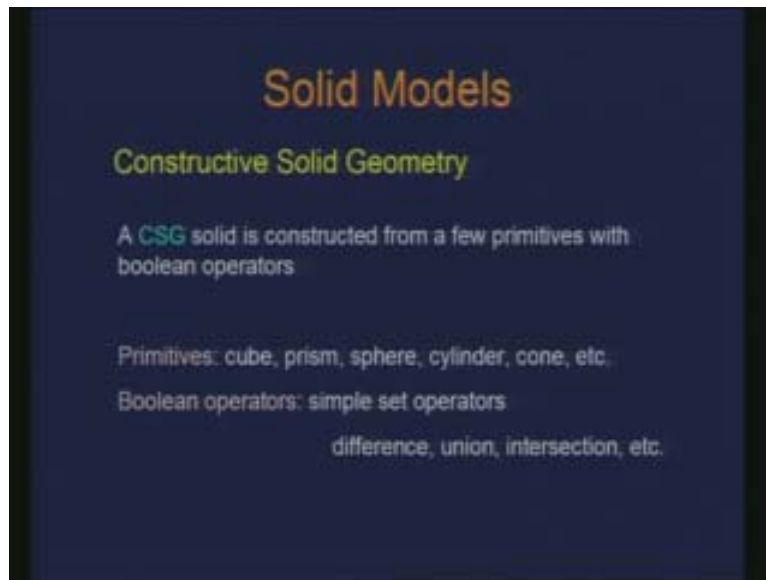
(Refer Slide Time: 34:21)



Now if I count the total number of vertices it comes out as 24 and number of edges as 36 and number of faces as 15. Now the number of holes on the faces is 1 2 3. The connected component is 1 where the whole thing is connected and genus is 1. There is one hole

which is through the object. So if I substitute this information there I should get that the left hand side is equal to the right hand side. Therefore, in this case both the sides turn out to be 0. These are some examples of polyhedral objects with their elements and the property captured using Euler number. Now there is something else which can also be used for representing solids or designing solids or constructing solids. There is something called as constructive solid geometry which is in short called as CSG.

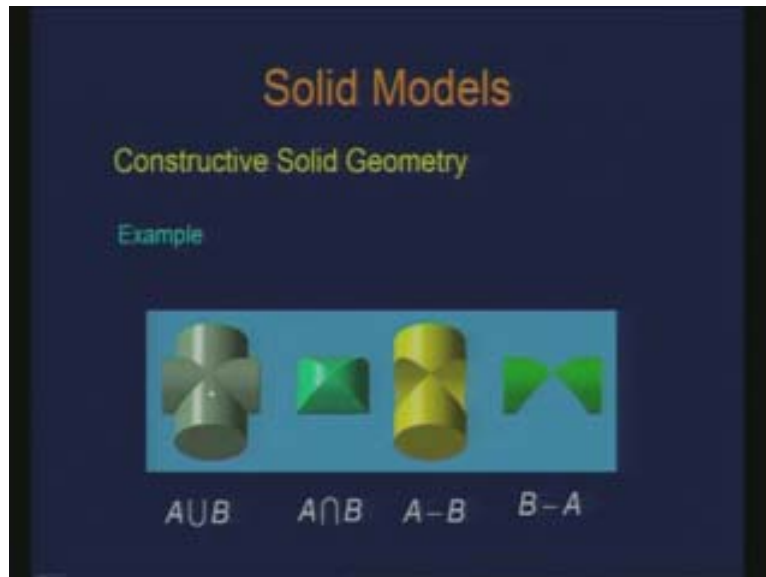
(Refer Slide Time: 35:56)



A CSG solid is basically constructed using a few primitives and some Boolean operators which are performed on those primitives. For instance, you can consider primitives like cube, prism, sphere, cylinder, cone etc. the Boolean operators are nothing but the simple set operations. For example; difference, union, intersection etc. The idea here is that you can construct a fairly complicated solid model using collection of these primitives and some Boolean operations on those primitives. Again it has lots of application in CAD CAM. If I have two primitive of cylinder so A is this and B is this.

There is one horizontal cylinder and there is one vertical cylinder. These are the two primitives I consider. So a simple set theoretic operations for instance union would give me something like this and it is a useful solid. We see such a component in many mechanical parts which is built only by a simple Boolean operation of two primitives. Similarly I can construct A intersection B it could be like this, this is A minus B and this is B minus A. Therefore this is another way of constructing complex solids just by applying Boolean operations to simple primitives.

(Refer Slide Time: 38:40)



In fact when we do ray tracing we will deal with rendering these CSG models because here what you need to also consider is the interior of the solid, what is the closure of the solid, what is the exterior of the solid to be able to decide that this is the part you are going to render. And ray tracing does that by the process of getting the visible points. So ray tracing does this operation inherently. If you are interested in getting this as the N model of the solid there is this block here, there is another block and there is a hole in this block. So we can start with these primitives where you consider this block 1 block 2 and then there is a cylinder here. All it requires is to have a certain transformation of these primitives and perform the Boolean operations.

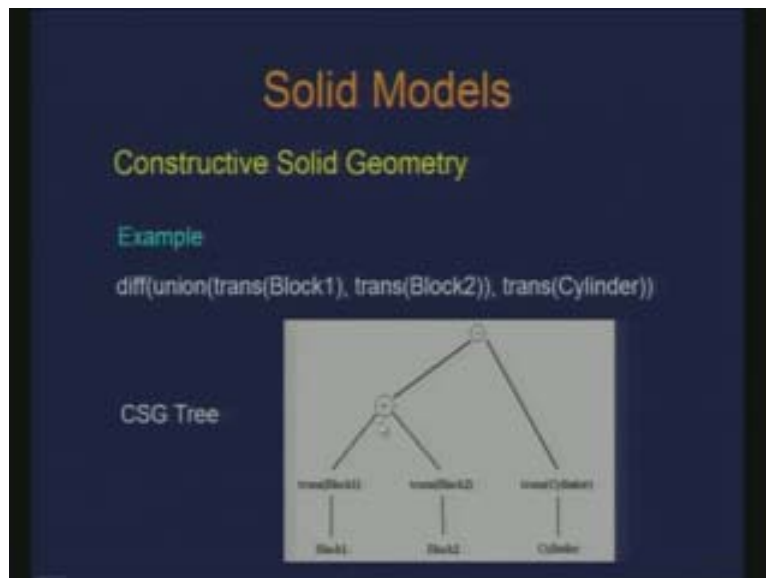
Therefore what you do is you translate these blocks to be able to get in to this configuration, this is block 1, this is block 2 and all it requires is a translation. And then you may also do a translation of this cylinder such that it goes at this location then all you need to do is here you need to perform the difference whereas here you need to perform the union. So the total transformation which you need to be able to get this particular solid is a union of translated block 1 block 2 with the translated cylinder which gives you this and then you take the difference for these two.

(Refer Slide Time: 40:07)



Now, is it unique? This CSG representation is not unique. You can construct this in different ways. Now looking at this as a chain of transformation accompanied with the Boolean operations I can actually represent this as some sort of a tree.

(Refer Slide Time: 42:29)

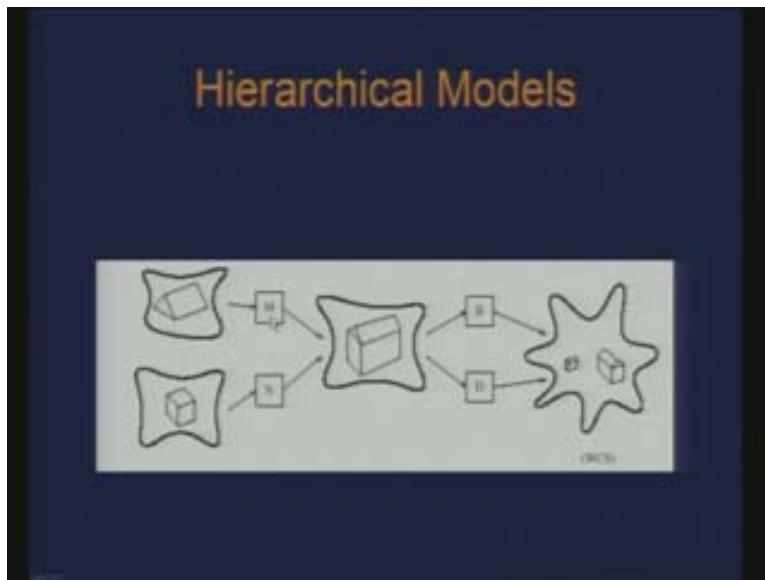


So I can actually write this operation where the leaf here is the primitive, the intermediate nodes are some transformation or Boolean operation. Here you have some transformation like a translation and here you have a Boolean operation. And the root of this whole thing is your solid, the object of interest. So I can actually look at this as a tree structure to construct the solid. So this is basically what we wanted to look at as solid models.

Now there are other representations also. For instance, we can also consider solids to be collection of volumetric elements. In fact there what we are saying is that just like an image is a collection of several dots of pixels similarly a solid is a collection of some volumetric elements or voxels. It is a complete discrete representation of collection of those voxels. So those are useful when we want to do volumetric manipulation, volume rendering and so on. This tree structure which is basically a collection of transformations and Boolean operations also inspires what you have as the assignment. The tree structure is also giving you a notion of hierarchical modeling where certain primitives are coupled with the transformations to define your final model.

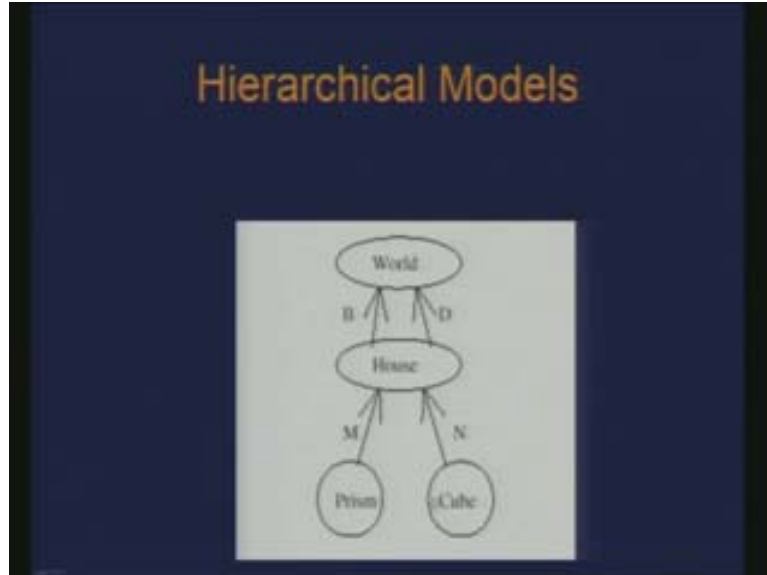
In the rendering pipeline the interest is just only to form a scene or form your world before you perform the viewing transformations and so on. So here the idea is that if you are trying to build a scene which is nothing but collection of houses then you can look at the primitives which would build parts of the model or the model. So here you have some sort of a prism, here you have some sort of a cuboid which will get some modeling transformation to be applied to form this. So an individual house could basically be constructed using a primitive, a simplified model and a transformation. So, once you have got the model of a house then again you can get various instances of this house using the set of transformations again. You can have a bigger house, smaller house, wider house, thinner house and so on and build a whole colony of houses.

(Refer Slide Time: 45:38)



This shows some sort of a hierarchy in the modeling process. So what we are looking at is some sort of a graph a directed acyclic graph bag which will have the leaves as simple primitive objects for the model, these edges are the transformations, again we get some aggregated model and again using some transformations we build our world or scene.

(Refer Slide Time: 47:12)



So a scene can basically be defined in terms of a graph or a tree where the nodes could have a simpler model and that simpler model can again be decomposed into its primitive elements and then you build a whole hierarchy of a model. For instance, when we are dealing with the design of a car there will be several primitives for your model which could be built as a hierarchy. So you can construct one wheel and then build the rest of the wheels and similarly if you want to have symmetric sides then you build one side and take the reflection of that. So eventually what you are doing is you are using combination of model primitives and the transformations to build the entire scene.