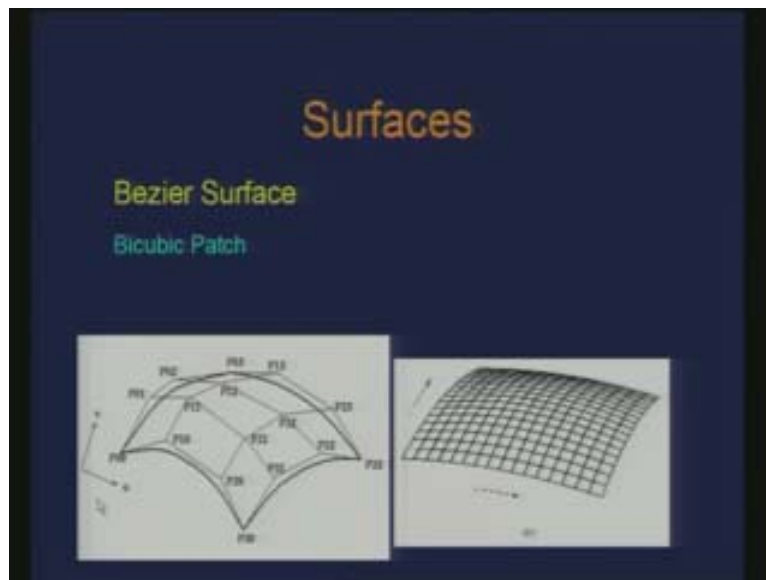


Introduction to Computer Graphics
Dr. Prem Kalra
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi
Lecture # 18
Surfaces (Contd...)

We have been talking about generation of surfaces primarily the parametric surfaces. Last time we talked about the Bezier surfaces. So as an example here what you see is a bicubic patch. Here you have a parametric domain defined through parametric u and v in a rectilinear domain. These are the control points for the Bezier patch and this is the surface you obtain.

(Refer Slide Time: 01:27)

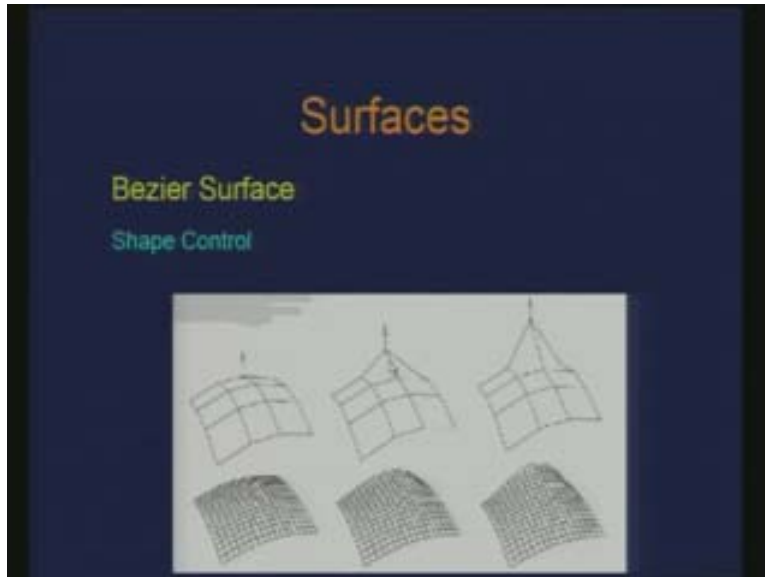


It is a bicubic patch because it is cubic in both u and v . And in fact the construction method we looked at there was one using de Casteljau algorithm where you could do a repetitive bilinear interpolation to obtain the point on the surface or you could also do a tensor product of the Bernstein polynomials in u and v to obtain the surface. So the advantage of the tensor product is that you do not have to deal special cases where the degree in u and degree in v are different, it is there in the formulation itself.

It gives you all kinds of properties similar to curves like affine invariance, convex hull and the shape of the surface which you obtain is basically governed by the location or position of these control points. So it interpolates the corner control points, it also interpolates the boundary curves which means that if I have to construct a Bezier curve using only these four points then the curve which I get is the boundary curve for the surface. So for the change in the shape we have very similar situation as in the case of curves we move control point, we just place a control point and the surface which is

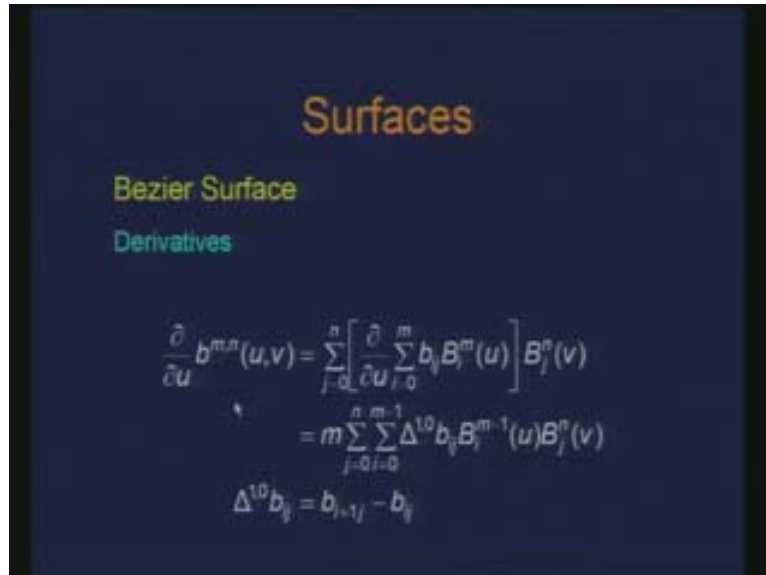
embedded using these control points also move. And once again the Bezier surfaces do not give you local control so the entire surface kind of moves. There is a local control in a pseudo sense depending on the influence of the parameter with respect to the point which is moved. So, that gives you the mechanism of shape control.

(Refer Slide Time: 04:39)



Now we also looked at the property degree elevation. Just the way we could elevate the degree in curves we could also elevate the degree of the surface in any one of the parameters in u or in v or in both. We also looked at the derivatives. The derivatives can actually be computed in a similar way as we do it for curves. So in the tensor product formulation the advantage is that you can always decompose your problem into a univariate case Just the way you will handle the curves then.

(Refer Slide Time: 05:31)



So you can extend those ideas to the bi-variate case which is defined as a tensor product. Here if I am looking at the partial derivative with respect to u all I need to do is take this inside and now this is very similar to as if I was doing the derivative of curves using the parameter u. So, what it gives me is an operator which I call it as delta 1, 0. It is 1 here because I am basically considering the derivative with respect to the parameter u, 0 there is no change in v. And this operator when applied on b_{ij} is nothing but the span of the polygon. This is nothing but span of the polygon b_{i+1j} minus b_{ij} . It is something similar to what we observed in curves.

Similarly, we can talk about partial derivative with respect to v and there the operator is now delta 0, 1 so I will be changing the indices J so b_{iJ+1} minus b_{iJ} .

(Refer Slide Time: 07:29)

Surfaces

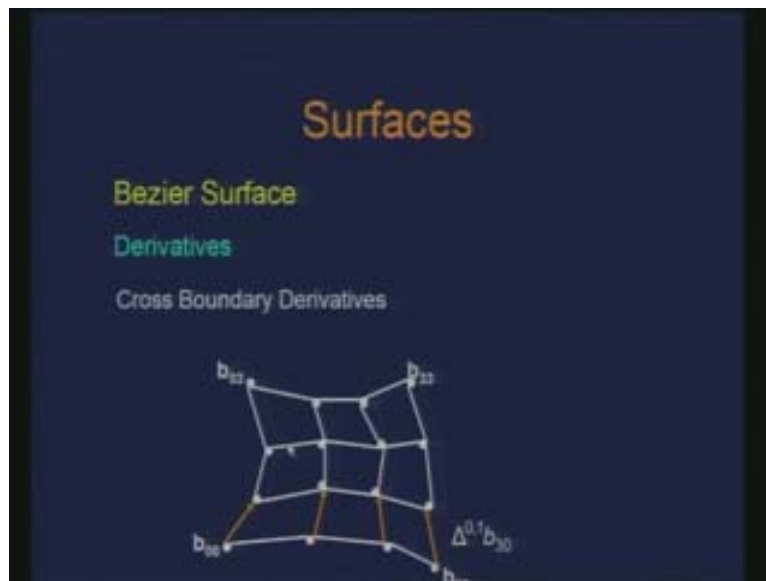
Bezier Surface

Derivatives

$$\begin{aligned}\frac{\partial}{\partial v} b^{m,n}(u,v) &= \sum_{i=0}^m \left[\frac{\partial}{\partial v} \sum_{j=0}^n b_j B_j^n(v) \right] B_i^m(u) \\ &= n \sum_{i=0}^m \sum_{j=0}^{n-1} \Delta^{0,1} b_j B_j^{n-1}(v) B_i^m(u) \\ \Delta^{0,1} b_j &= b_{j+1} - b_j\end{aligned}$$

Then I can see the cross boundary derivatives which are basically defined by the pair in the span of the polygon. For instance, if I am looking for the derivative here all I have to do is take this kind of a strip which is defined by the adjacent pair of the Bezier control points which would give me the derivative.

(Refer Slide Time: 07:33)

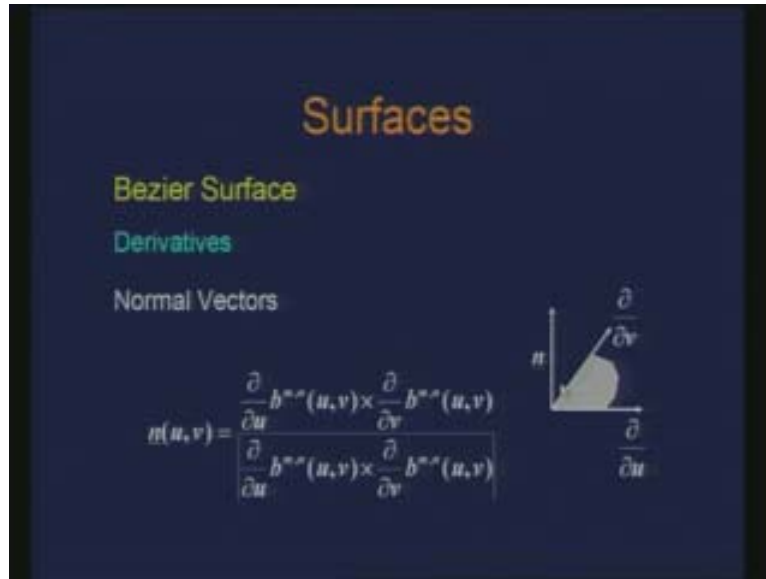


Just the way we had seen the derivative in the case of curves were also Bezier curves of the reduced degree. Similarly, we also observe that the surface derivative is also a Bezier surface. The relevance of derivative is when we want to construct composite patches and we need to worry about the match of the derivatives.

An exercise given in the last class was that, given the partial derivatives at a point how you compute normal at the surface?

The answer is that you just take a cross product because you basically have two tangents which would span a plane there and then you can obtain the normal vector just taking as the cross product of these two tangents.

(Refer Slide Time: 09:11)



So if I am saying that this is the partial with respect to u and this is the partial derivative with respect to v so I am just taking a cross product of these two vectors to give me the normal at this point. You can even get the normalized vector. This is important because eventually you need to render these surfaces. So, while we are talking about rendering how do you propose rendering of a Bezier surface, how would you render a Bezier curve?

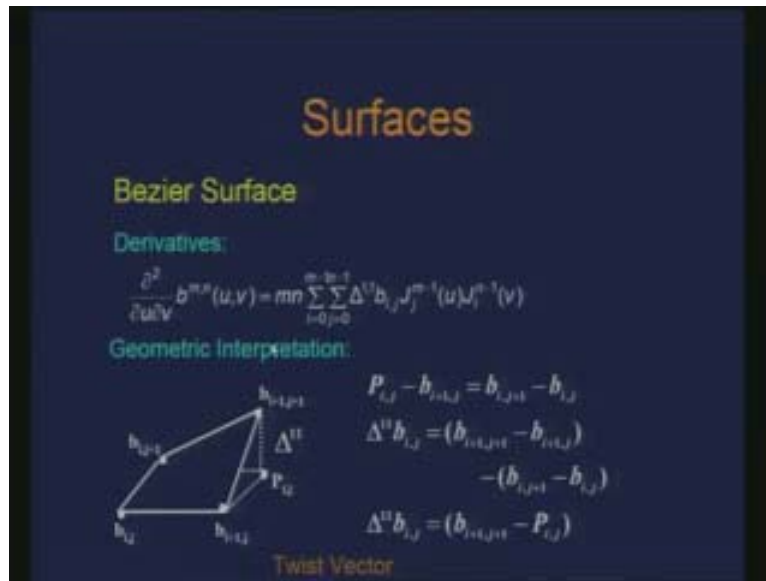
One possibility is that you change the parameter t compute the point on the curve and Join it with the previously computed point with a small line. Eventually it has to be some sort of a continuous curve and that is one way to do it. Therefore it is the computation of the point on the curve by varying the parameter t and Just Join it by some line segment. You are in some sense approximating certain points in between through the line.

We have also talked about subdivision of curve. I can start with the control points or the control polygon of the curve and keep sub dividing. In the limiting case I approach the curve. Therefore I can use that as a mechanism to even block the curve. Starting from the Bezier polygon I can keep sub dividing the polygon under a certain threshold I stop the sub division and that becomes the display of the curve.

Similarly, I can also do for surfaces so both ways. I could either change the parameter u and v then obtain small quads considering those to be polygonal or I could start from the

control net of the Bezier surface and keep doing the sub division. That is a method by which I can display the surface. Again looking at these derivative terms there is an additional derivative which could be also of interest in certain situations. So far we basically took the partial derivative in u or partial derivative in v and there was a physical interpretation of those partials in terms of the tangent vectors of the point.

(Refer Slide Time: 14:15)



We could also get mixed partials partial in u and then partial in v and just by the way we had done earlier there will be an operator Δ^{ij} which would say that there is a difference in u and then there is also difference in v. So, what this mixed partial is doing is, the geometric interpretation of that is, let us say I am talking about these four control points of the Bezier polygon b_{ij} , $b_{i+1,j}$, $b_{i,j+1}$ and $b_{i+1,j+1}$.

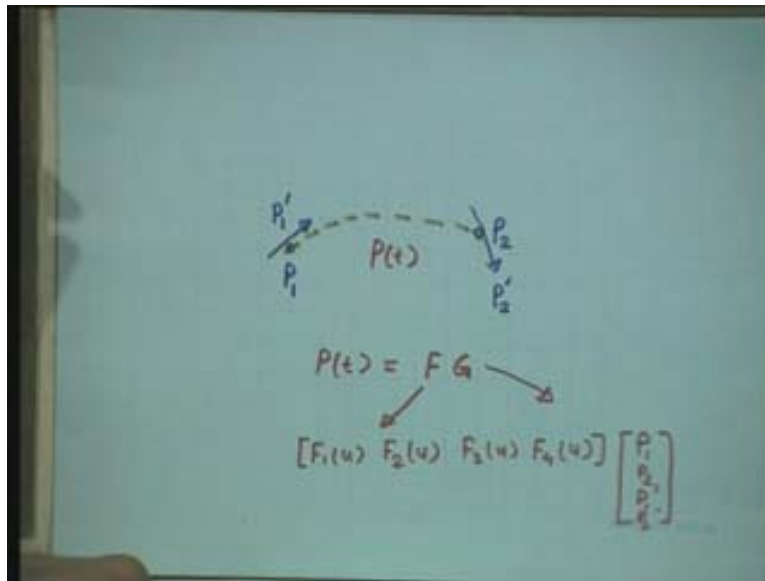
Now if I look at this operator Δ^{ij} what is it doing? It is basically performing the differences in u and v. So this Δ^{ij} is nothing but taking a difference of $b_{i+1,j+1}$ minus $b_{i+1,j}$ and a difference for i; now $b_{i+1,j+1}$ minus $b_{i,j+1}$ and then the difference of these two. That is what the Δ^{ij} would do, it is just an expansion of the differences.

Now I construct a point P_{ij} in the manner that P_{ij} minus $b_{i+1,j}$ which is this is the same as $b_{i,j+1}$ minus $b_{i,j}$ which is this. So I basically construct P_{ij} as the point which would construct a parallelogram using these three points. Then I can basically show that this Δ^{ij} operator is nothing but this offset. Once I have seen the operator Δ^{ij} then this is nothing but $b_{i+1,j+1}$ minus P_{ij} .

You make a substitution from here to this expression and that is what you will get. So what is being interpreted here is that this is sort of a deviation from the plane which is **spanned** between these three points. So had these been 0 this we this would have been a planar. All these four points would have actually given you a planar. So it is a deviation from that plane so sometimes it is called as a twist vector.

If you recall the interpolating cubic splines we had a point P_1 its tangent vector at P_1 point P_2 and a tangent vector at P_2 P_2' . So, given this using cubic splines you obtain a curve like this and the curve which I get is $P(t)$ which is basically expressed in this manner where F is some sort of a blending function which may look like $F_1(u)$ $F_2(u)$ $F_3(u)$ $F_4(u)$ and this G was the geometric information in terms of the position of the point P_1 position of the point P_2 and the tangent vectors P_1' and P_2' so that is what you had here. This was your cubic interpolating spline and this is also called as Hermite spline.

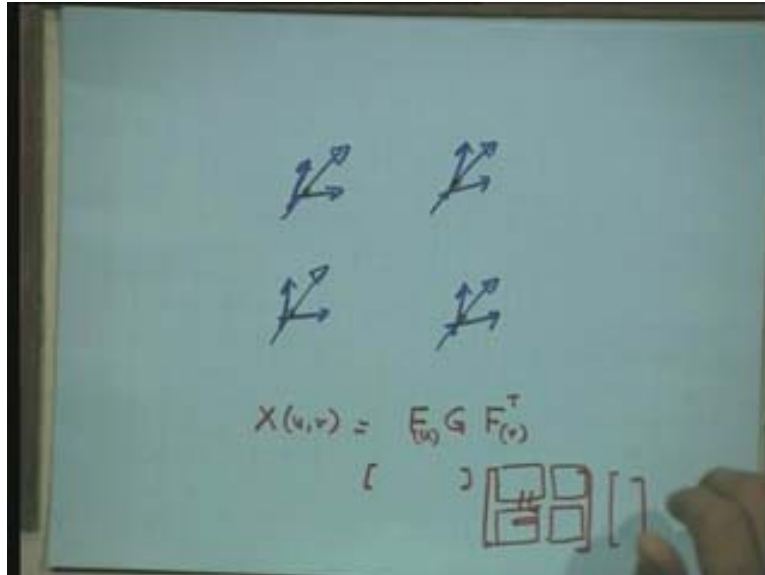
(Refer Slide Time: 20:39)



So the idea is that you have a matrix formulation where you are basically blending geometric information. Now I try to extend similar idea to construct a surface where I would be given some boundary conditions. That is, I am saying that I have information about the position, the interpolating spline was a cubic spline so what I am trying to do is I am basically trying to look at a bicubic surface which would satisfy certain boundary condition. And just by extending the matrix formulation of cubic spline I can see a surface $X(u, v)$ which could be written in some sense these F s and some G and then again FT . So this could be for u and this could be for v and that would match with my matrix formulation or something also like a tensor product. In a Bezier formulation I have the F given as the Bernstein polynomials, Bernstein polynomial in u and Bernstein polynomial in v and the geometric information is the control points so that is how it looks.

Now in this case I have something like this so here I can have some sixteen things in order to satisfy this matrix multiplication formulation so there are some sixteen things. So let us try to see if we can extend the idea of cubic splines the information which could determine these sixteen things so four of which are the positions the four corner points. I can also have the tangent vectors or the derivatives just the way I had in cubic splines so that makes twelve.

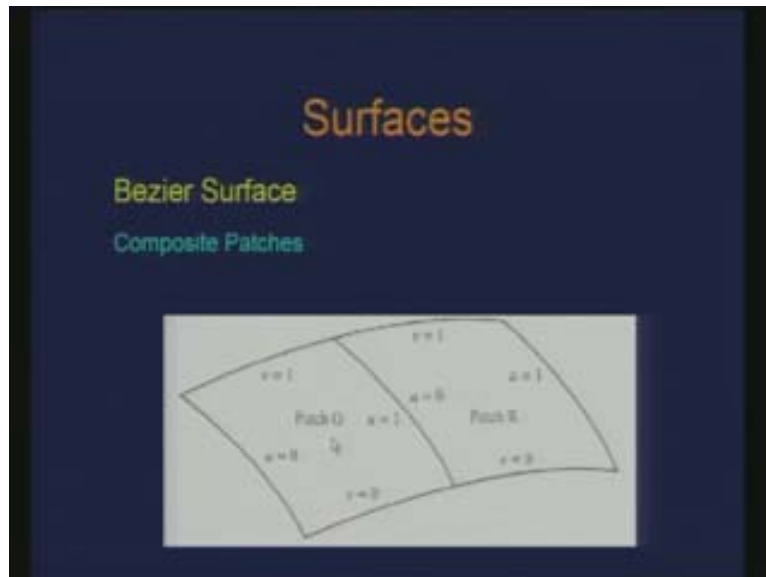
(Refer Slide Time: 25:00)



If I have these mixed partials which are the twist vectors when I have sixteen things then I can construct a hermite bicubic surface where these F primes are very similar to the case in the curves and I may have the information of the position here. Therefore here I could have some matrix of position, here I could have some matrix of tangent vectors in u, here I could have some matrix of tangent vectors in v and here I could have some matrix of partial u and v of the twist vectors.

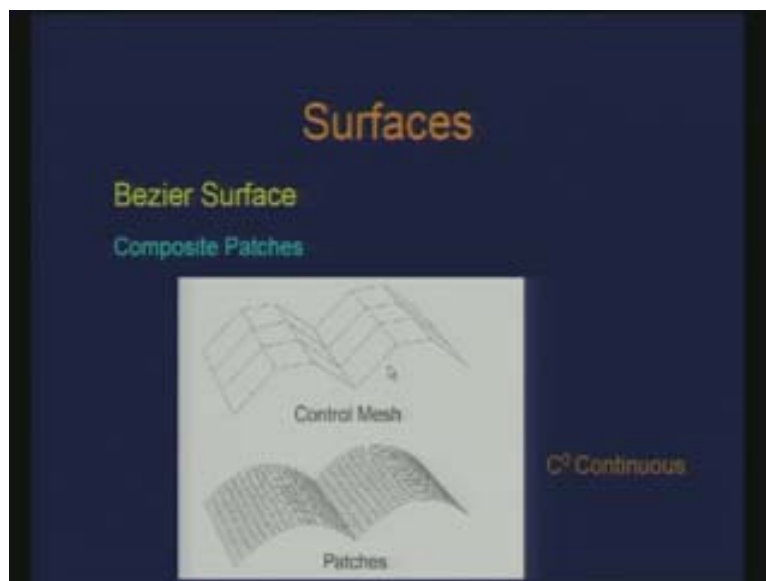
So, given this geometric information I can then construct the bicubic hermite surface. Therefore this is some relevance or application of twist vector. There are other applications also but this is one which you can extend from the curves. One of the motivations of getting these derivatives is to be able to construct composite surfaces just as the way we did composite curves. Here is the situation where I have a patch O and then another patch R all I want is to have a composite patch which joins these two. So once again there could be various ways in which we can look at the joining of the two patches.

(Refer Slide Time: 25:57)



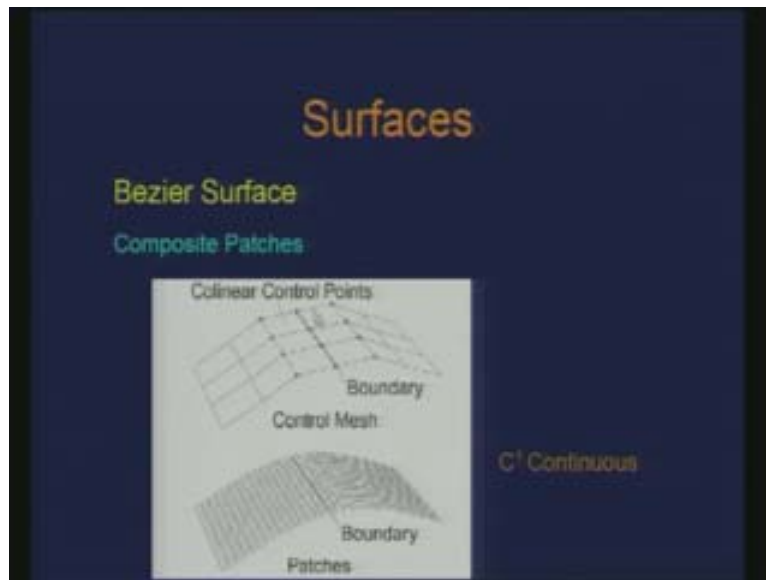
For instance, we can have a composite patch where all we are concerned is that the end points should match. That is, we have a C_0 continuous surface which is given through, if I have these control points for the first patch is same as the first control points for the second patch. As long as they are at the same position I will have the two patches meeting at this curve which is the boundary curve for the two patches just because the boundary control points are the same. That is a C_0 continuous patch. What happens if I am interested in C_1 ? Then the role of derivatives comes into place. The derivatives should be the same.

(Refer Slide Time: 27:02)



If I had taken the case of a curve and another curve and I want them to be C_1 continuous then what is the constraint I need in terms of the position of the control points for the two curves? They should lie in the same line so it is the collinearity. They should be collinear because we are talking about these curves and these spans should also be the same. If I am talking about cubic then b_3 minus b_2 in the first curve it is three times so the other side will also be three times b_1 minus b_0 for the second curve.

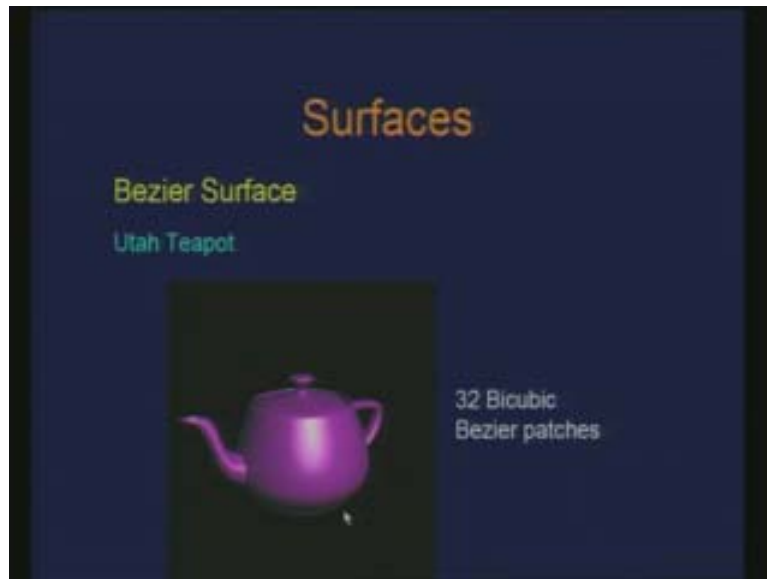
(Refer Slide Time: 29:56)



These spans should be the same as long as I am having the parametric domain definitions between 0 and 1. If I change them then I have to take the ratios. That is what will happen even in the case of surfaces. In order to have a C_1 continuous surface for the two patches here I want this and this to form a line so these are collinear, these are also collinear and so on. Therefore that is the configuration of control points to give me C_1 continuous surface at the boundary.

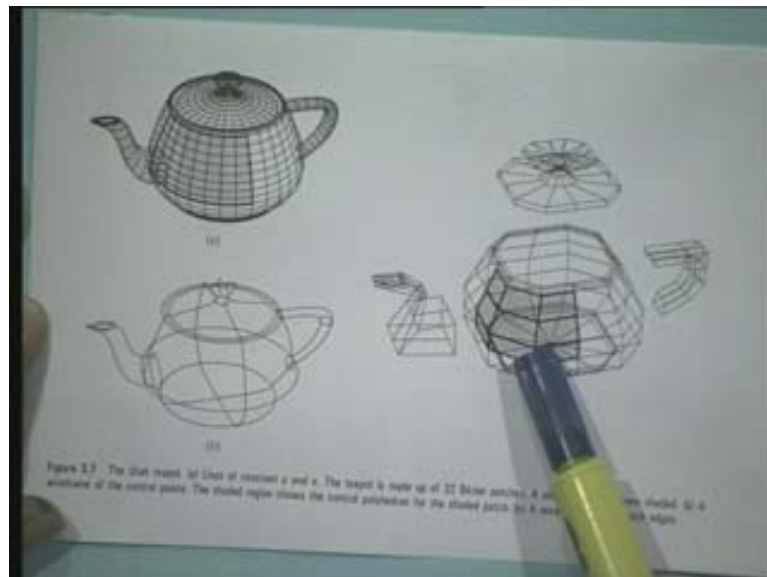
Here is an illustration of the application of these Bezier patches. This is actually a very famous object in the computer graphics community. This is like the image of Lena in image processing. Everybody wants to work with image of Lena and for whatever operations you want to perform in image processing. So this is such an analogy. Whatever you do in rendering, in modeling you try to use this tea pot. This is a very old historical object and model of tea pot which had 32 bicubic Bezier patches.

(Refer Slide Time: 30:31)



Here these are the various patches and the shaded one is one single bicubic patch here corresponding to that you have this surface.

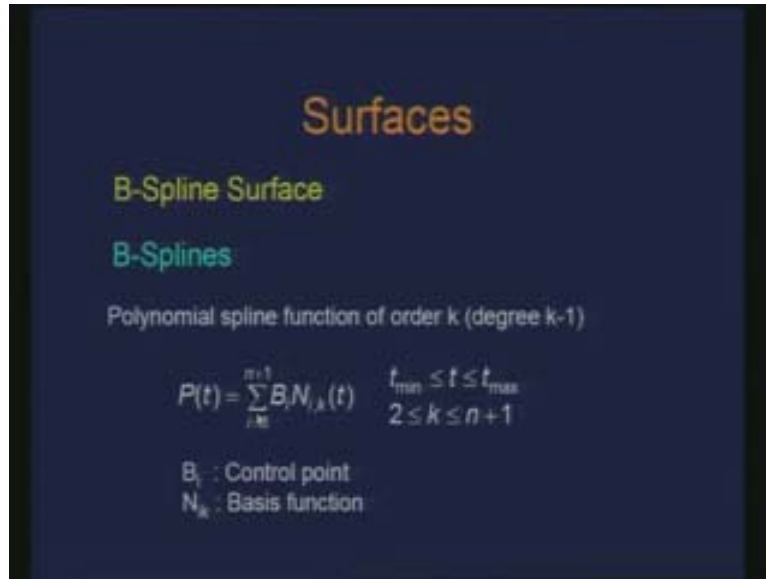
(Refer Slide Time: 32:07)



So you have patches for the handle, patches for the top, patches for this and also there are patches for the base. The nice thing about this model is that you obtain a fairly smooth kind of a surface which you see in reality. The inspiration is coming from a real tea pot. This is sort of an example of the Bezier surfaces and the patches.

Now that we have seen the construction of Bezier surfaces using control points and some sort of a blending of these using appropriate Bernstein polynomials we can actually do construction of other surfaces where the basis could be different than Bernstein polynomials. For instance, I can also construct B-Spline surfaces. That is also the advantage of the tensor product formulation. All I need to do is use the appropriate basis functions in the parameters defined through the parameter domain u and v and just take the tensor product.

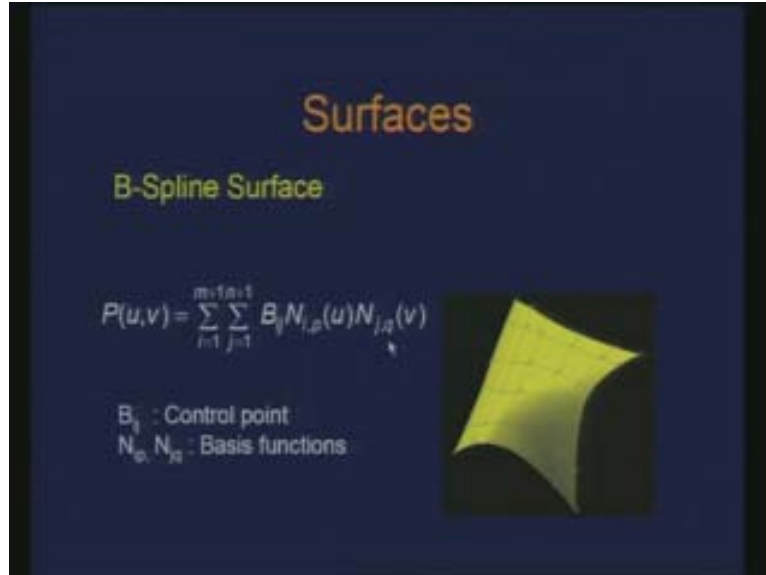
(Refer Slide Time: 34:00)



So if we go back and see how the B-Spline was defined B-Spline was defined basically through these control points and these basis functions. Now I will have these control points in the similar manner as the Bezier surface and use these basis functions and take a tensor product. So you have a basis function N_{ip} u and N_{jq} v . Here this p and q would actually determine the degree of the surface in u and v . Just the way k was determining the degree of the curve which you were trying to construct in B-Spline here p and q would determine the degree of the surface you want to construct.

Once again a surface could be possibly generated using these control points and the basis functions. And again you have the role of knot vectors in terms of defining these basis functions so you could possibly have open knot vectors, uniform knot vectors, non uniform knot vectors and so on.

(Refer Slide Time: 34:53)



It is just the way you were constructing the B-Spline curves where you had various control handles for the shape of the curve which would be carried forward for the surface construction. We also have the properties extended in a similar way.

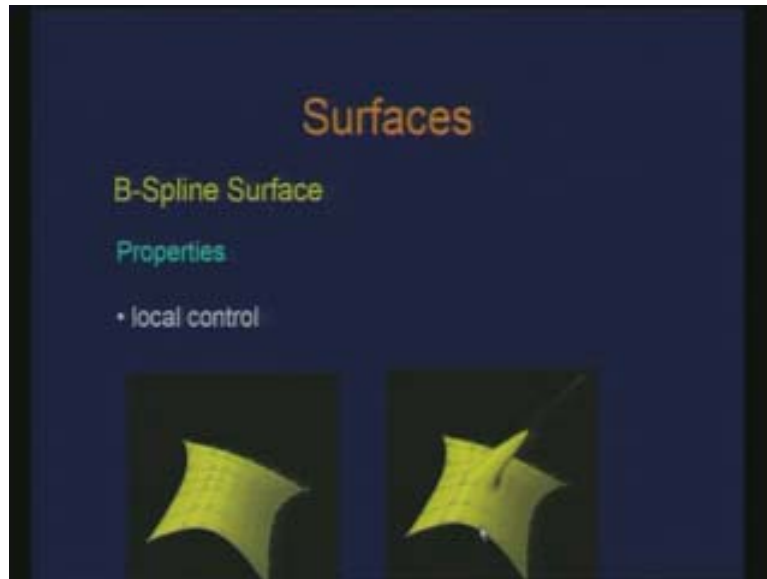
(Refer Slide Time: 36:02)



You have the property of affine invariance, you also have the property of convex hull which is stronger as in the case of curves and there is also a local control just the way we had local control in the case of B-Spline curves so that again gets extended to the surfaces.

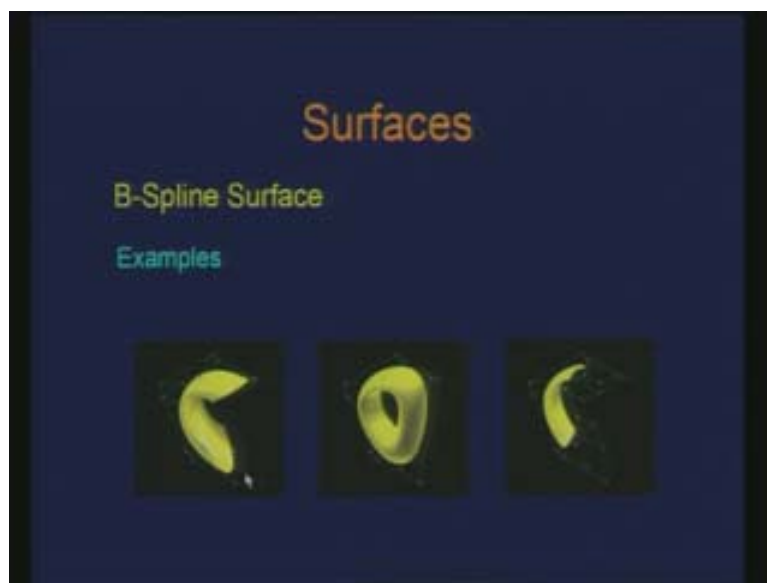
Here is an example; this B-Spline surface and a point here is moved to this location so this is how the surface changes. You can see that only a small neighborhood with respect to the point which is moved the surface has changed. Therefore this local control is there.

(Refer Slide Time: 37:12)



And as I said you have various ways of knot vectors, you may have open uniform knot vector which gives you the property of clamping to the end control points.

(Refer Slide Time: 37:30)



Just the way the B-Spline curve would pass from the end control points if you are using the open uniform knot vector that is the repetition k times of the two ends. Or you could

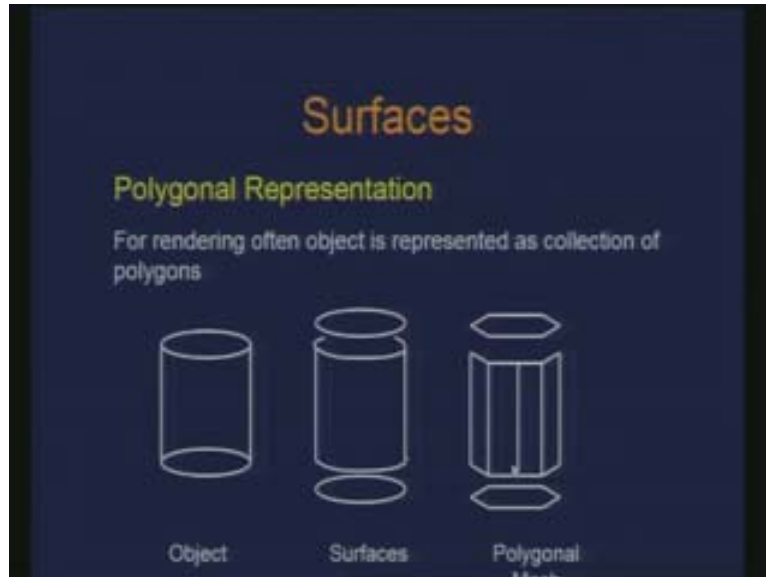
also have a closed surface just by taking the control points in a circular way so that they form a closed B-Spline polygon. Or you could also have the knot vectors which are periodic which would not guarantee that the surface passes through the end control points. Hence, similar properties and ideas which are used in the case of curves can be seen in surfaces.

Now as we were discussing about the rendering or the display of surfaces one of the ways we said that we compute the point on the surface of the curve and approximate by joining a line from the previous point computed. And in the case of surface we could possibly compute quads, quads means four points which could lie in a plane so I just have to display that quad which is a planar primitive.

Therefore, in some sense what I am trying to do is I am trying to approximate the surface which is computed into some discrete elements which are piecewise linear, piecewise linear in the case of curves or in the case of surfaces they are piecewise planar patches. Often it is the situation that we have the representation of surfaces as these parametric surfaces but for the purpose of displaying we come down to these planar patches, reduce the surfaces to collection of piecewise planar patches for various reasons. One of the reasons is that the rendering which is supported in most of the graphics library and the hardware we have supports polygonal shading and polygonal rendering.

Let us try to see how these polygonal representations could be done for the purpose of converting these surfaces or even acquiring the models in terms of polygons. For example, if you have an object which is a cylindrical object so one could conceive this to be made of several surfaces so they could be this surface which is the side surface of the cylinder and there are caps of the cylinder that could be another surface. Now, in order that I display them as polygonal patches or polygonal discrete elements I would convert them into some collection of polygons so this could be one possible decomposition into polygons. Collection of these polygons basically is some sort of a mesh which is generated using these polygons.

(Refer Slide Time: 41:51)



The idea here is that you have representation which could be mathematically continuous but at the end you have this representation which may be desired for several reasons and one of the reasons could be just the rendering. Now let us try to see the data structure which could be possibly be used for representing these polygonal meshes.

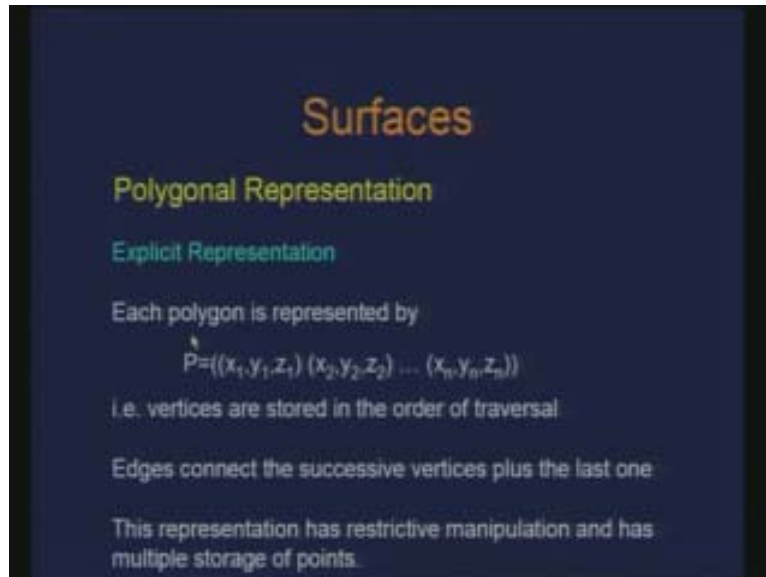
(Refer Slide Time: 43:18)



Therefore this polygonal mesh if you want to look at is nothing but a collection of edges vertices and faces or the polygons. These are the three entities which are involved in the representation of polygonal mesh. We are considering these meshes of the kind where we see that the edge is shared by at most two polygons. So, if we look at the edge it connects

to two vertices and a polygon is nothing but a closed sequence of edges. Therefore as a representation there could possibly be several ways in which we can represent this polygonal mesh.

(Refer Slide Time: 43:53)



One way is that I consider each polygon represented in explicit fashion giving all the points which are defining that polygon. I have these vertices defined through its coordinates $x_1, y_1, z_1, x_2, y_2, z_2, x_n, y_n, z_n$ and so on and also the last one which would connect x_n, y_n, z_n , to x_1, y_1, z_1 just to complete the loop.

Now if you look at this representation, we are saying that each polygon is defined in this fashion. Clearly there is lots of duplication. Many of the vertices are repetitively used. So this may not be a very good way of representing the polygonal mesh plus it is very restrictive in terms of manipulation of the information in the polygonal mesh because it does not capture information about adjacency, it does not capture information about the incidents vertices to an edge and so on and so forth so the manipulation is difficult. Therefore we go one step more and try to see a data structure which reduces this replication and add some features of manipulation.

So what you could have possibly is a pointer to a vertex list. So, instead of defining this polygonal in an explicit manner where you consider each of the vertex coordinates what you can think of is that there is a vertex list or a table of vertices which consist of the coordinates of the points and the polygons are nothing but the index or the entry pointers to this table or vertices. That is a typical representation you might have seen.

(Refer Slide Time: 46:45)


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
 $P = (V_i, V_j, V_k)$
e.g. $P_1 = (1, 2, 4)$ and $P_2 = (4, 2, 3)$

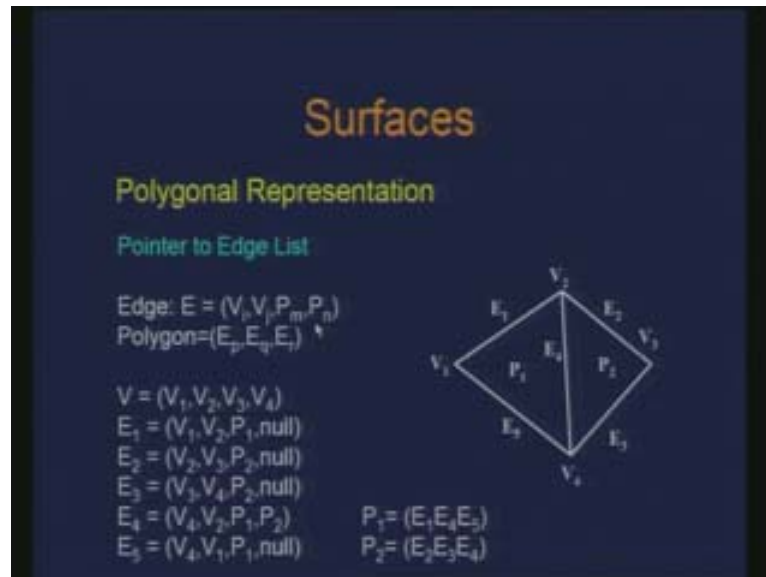


In this representation it is difficult to find polygons that share an edge.

In this example, for instance if I have P_1 defined using $V_1 V_2 V_4$ these are nothing but the indices 1 to 4 in the vertex list. This reduces the redundancy which we had in the earlier representation to a certain extent. Now again from the manipulation point of view if I am interested in finding out polygons that share an edge, so with respect to this edge I want to find out the polygon shared by this edge it is not easy. This data structure does not really facilitate that so easily.

Maybe we have to do something else, maybe we have to bring in information about edges which is sort of missing here. So what we do is we have a pointer to edge list so we build an edge list. So here an edge is nothing but a tuple using the vertices it joins $V_i V_j$ and the polygons which are shared by this edge. And we also have the information of the polygon defined in terms of the edges.

(Refer Slide Time: 48:32)

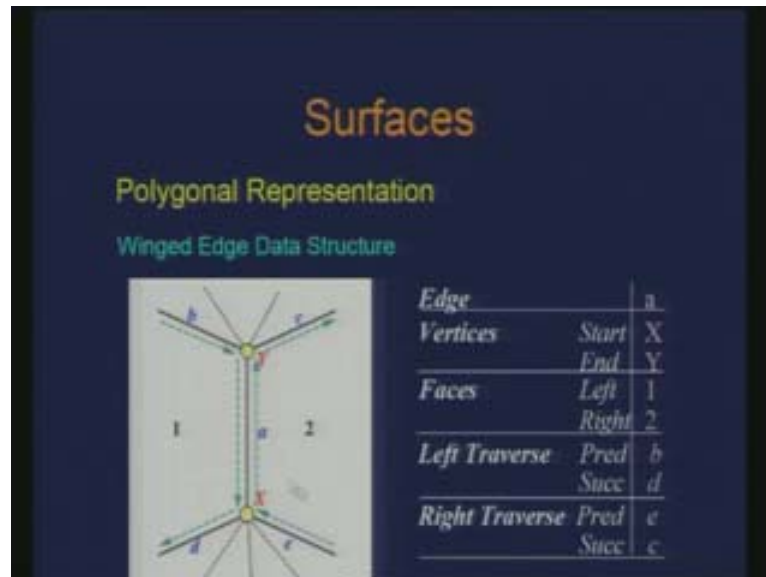


Here again we are considering triangular polygonal mesh and that is why there are only three edges. So in this example for instance I have the vertex list given as $V_1 V_2 V_3 V_4$, these are the points I have then E_1 could be defined in terms of the vertices V_1 and V_2 which is this and the polygons which are sheared for this edge. So you have P_1 and there is no other polygon here so I Just put null here.

Similarly, I can have $E_2 E_3$ and so on. So E_4 you observe that there is a polygon P_2 so you have both these polygons P_1 and P_2 for this edge. So a query for getting the polygon sheared by an edge is straightforward. Now the question is that if I am interested in finding out the edges which are incident on a vertex I may have to actually go through the entire list of edges and that may not be straightforward. Clearly as we demand for both manipulations the data structure would become heavier so you need to store more information.

One of the data structures which is quite popular is the winged edge data structure. In this winged edge data structure what we have is basically the co data structure is considered as edge. There is this information about an edge a , and it gives you the vertices which are there for the edge a , the constitute vertices $X Y$, the faces are sheared by this edge 1 and 2 and there is also this notion that how you are going to traverse the mesh.

(Refer Slide Time: 52:03)



If you look from the left side with respect to the polygon 1 you have this clockwise navigation. And if you look at from the right again you have clockwise navigation. If I am doing the left traversal on this side then what is the predecessor to this edge and what is the successor to this edge? It is b and d. Also in the same way when I am doing a right traversal what is the predecessor and what is the successor? So I have this information in the data structure.

Now if am interested in finding out the incident edges for a point X then I can do that with a limited navigation about a certain number of polygons. This would tell me what are the possible edges I need to look for and what are the polygons shared by those edges so I do not have to traverse the entire edge list.

Therefore if you look at the way it is sort of drawn pictorially you see this edge here which is the principal entity for the data structure a the edge a and there is this face one and on the left side there is the face two and then you have these edges b c d e so they sort of form a **wing above this edge**. So this is the central part of a bird and these are just the wings. That is why it is called winged edge data structure. This is alright as long as we do not consider holes. The moment we consider holes then we may have to do something else.