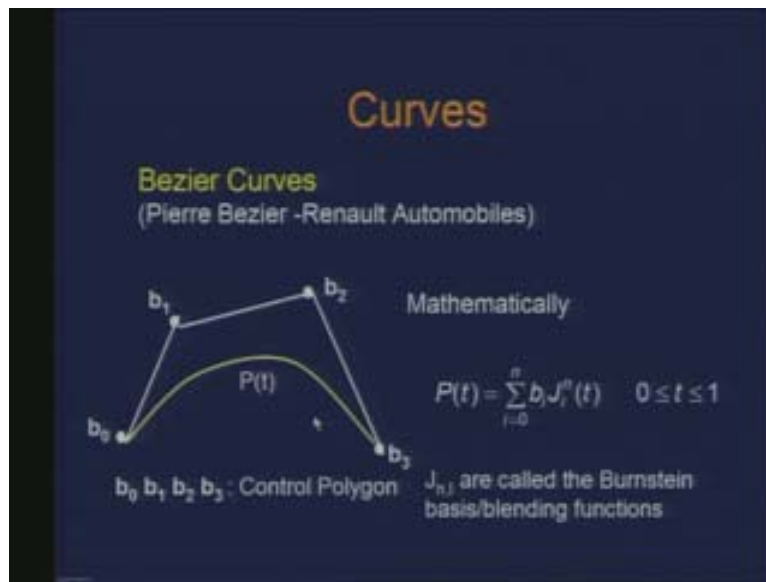


Introduction to Computer Graphics
Dr. Prem Kalra
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi
Lecture # 13
Curves (Contd.....)

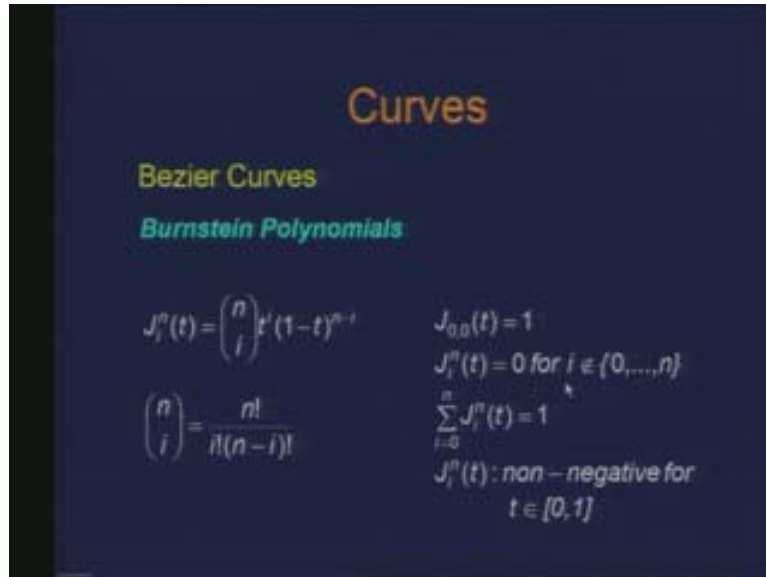
We continue on parametric curves. We have been talking about Bezier curves which are basically the curves which approximate the shape the user wants to design.

(Refer Slide Time: 01:24)



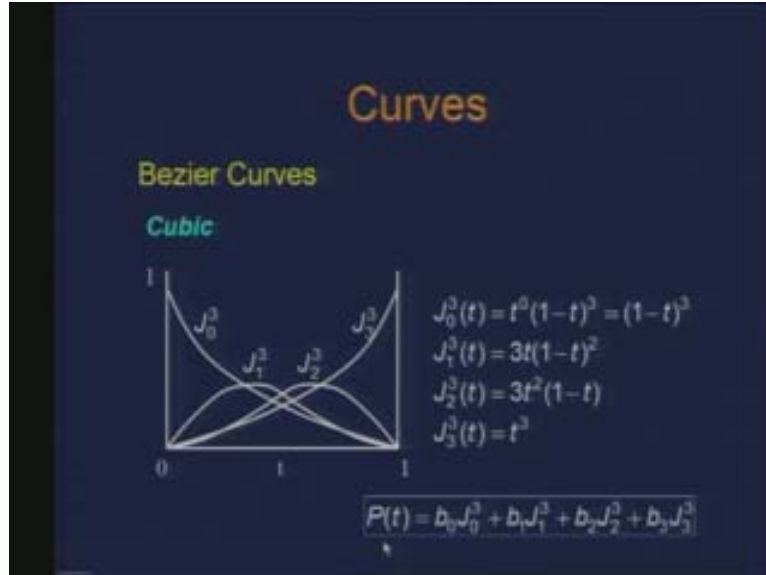
So the way these Bezier curves are defined you are given these control points $b_0 \ b_1 \ b_2 \ b_3$ and all it takes is the blending functions in the form of Bernstein polynomials and when you take a combination of these control points with these blending functions it results into B Spline curves. So, for this input Bezier polygon you get a curve like this. So you can see there is a correlation between the shape of the Bezier polygon and the resulting curve shape. And we also notice that the degree of the resulting curve is directly related to the number of control points we have in the Bezier polygon. In this case it is a cubic Bezier curve where the total number of control points in the Bezier polygon is 4. Sometimes it is also referred as the order of the curve as the same as the number of control points. So the order of a curve is 4 and the degree of the curve is 3.

(Refer Slide Time: 03:15)



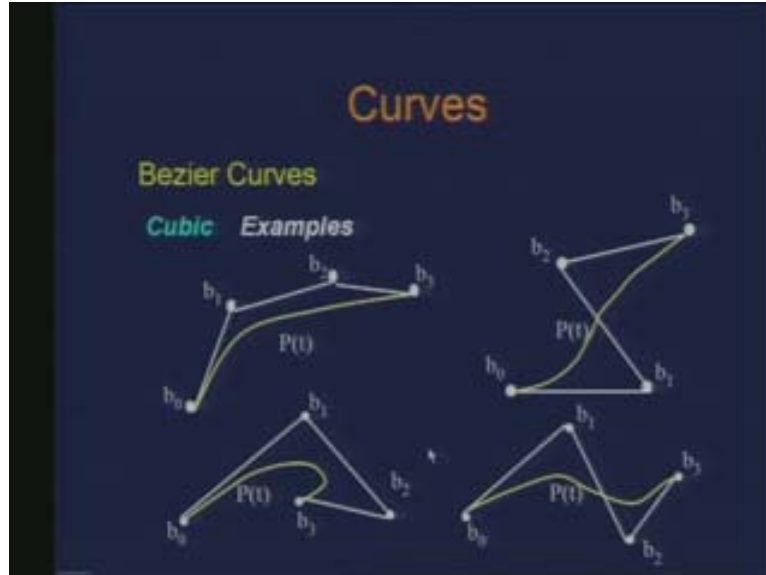
These Bernstein polynomials are basically defined as, these are the polynomials in t^i and $1 - t$ raised to the power $n - i$. And we observe that the definition $J_{0,0}$ defined at t is taken as 1, $J_i^n(t)$ for the rest of the i 's belonging to other than 0 and n is 0 and it sums to 1, all these blending function coefficients sum to 1 between 0 and n and each of these is actually non negative for the t parameters spanning between 0 and 1. These in fact are the properties of the Bernstein polynomials in turn render lots of properties for the Bezier curve. So this partitioning of the unity the convexity of these coefficients etc actually be useful for the properties of Bezier curve. Thus, for a cubic case this is how the Bernstein polynomials look like.

(Refer Slide Time: 04:40)



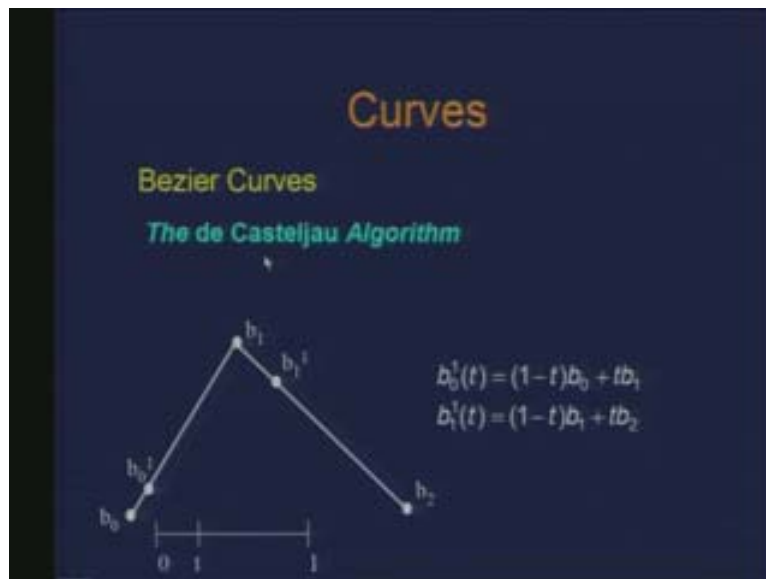
And when you see the curve it is basically a combination of these coefficients with the respective Bezier points here, it is just the combination of that. This definition is somewhat similar to what we also have observed in the cubic splines where we see the resulting curve as a combination or blending of the geometric information given. So here the geometric information is basically given through the control points of the Bezier polygon and that is what you are trying to blend with using Bernstein polynomials. This was basically using the Bernstein polynomials. So these are some of the examples which we saw. So here it all shows that the shape of the Bezier polygon is actually directly influencing the resulting curve.

(Refer Slide Time: 05:46)



Then we actually started looking at a different way of constructing these Bezier curves. One was using Bernstein polynomials just as a basis or blending function and there is another way of constructing these Bezier polygons and that is due to this de Casteljau algorithm.

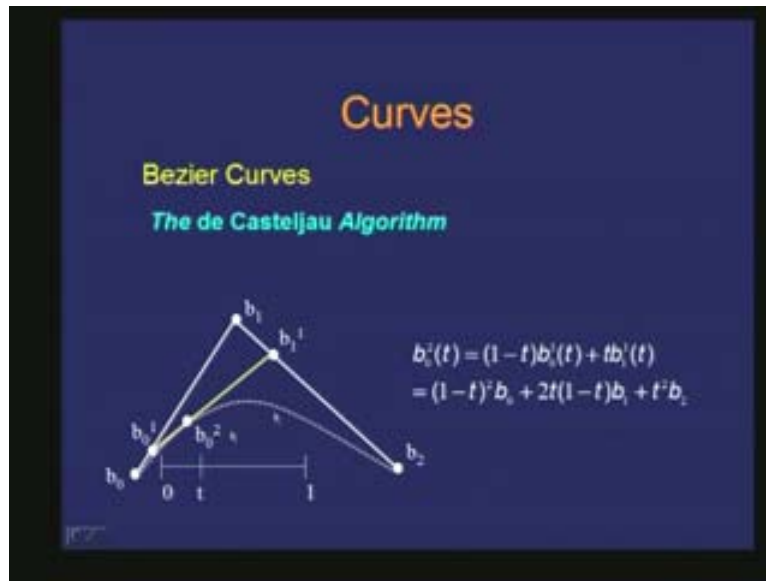
(Refer Slide Time: 06:08)



This de Casteljau algorithm basically works using linear interpolation. So the idea there is that if I have the parameter t is defined between 0 and 1 so for any value of t I can locate a point on the curve by successive application of linear interpolations. That is the underlying idea about this algorithm which basically means that if I apply a linear

interpolation on to this leg or the span of the Bezier polygon, let us say for this leg I apply this linear interpolation to get the point b^1_0 somewhere there, similarly for this span I apply the linear interpolation and obtain the point b^1_1 and then subsequently I get a new span defined between b^1_0 and b^1_1 and I repeat the linear interpolation for the parameter value t along this span and get the point b^2_0 .

(Refer Slide Time: 07:36)



So this b^2_0 is nothing but again a linear interpolation of b^1_0 and b^1_1 . And if I just expand this because b^1_0 was nothing but a consequence of linear interpolation in the Bezier polygon and b^1_1 was also a linear interpolation so if I just expand this I obtain this. And these are nothing but the Bernstein polynomial coefficients of degree two. So all it does is with this application of linear interpolation on this span it locates a point **b20** which is a point on the curve. So for any value of t I can locate the point on the curve by successive application of linear interpolation. That is what de Cateljau algorithm is.

The way I can sort of extend was the kind of example given for a quadratic Bezier curve of degree two but this can be extended to any degree. And all it looks like is some sort of a recursive formulation here. Given the Bezier control points at b_0 to b_n I obtain at some level r th level the points b_{ri} which are nothing but the linear interpolation of the points obtained in the previous level or the previous iteration which is r minus 1 and I keep applying that till this r is equal to i .

(Refer Slide Time: 09:04)

Curves

Bezier Curves

The de Casteljau Algorithm

Given : b_0, b_1, \dots, b_n

$$b_i^r(t) = (1-t)b_i^{r-1}(t) + tb_{i+1}^{r-1}(t)$$

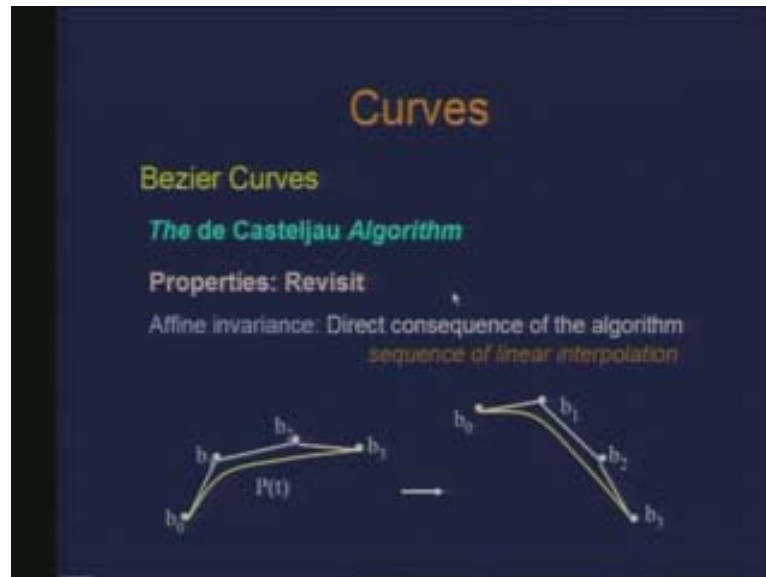
$r = 1, \dots, n$
 $i = 0, \dots, n-r$

$$b_i^0(t) = b_i$$

So that is the algorithm I have for any degree of the curve. And given that the first level where I start these are the points of the Bezier polygon itself. Now, if you look at the way this computation is going on in the case of the example of a cubic curve you observe that you have some sort of a triangular array which is being found. So this and this gives you this and so on. So all you see is basically a triangular array which is being formed. Therefore there should be arrow from here also, there should be arrow from here also and so on if you want to see the computation which is going on. Here all it is being shown is the storage or the array which is formed at different levels.

Here are some properties to look at using Bernstein polynomial constructions. Let us also see how these properties are revealed when we have de Cateljau algorithm construction. We remember that affine invariance was basically a property where it was equivalent if I apply an affine transformation to the curve or if I apply the affine transformation to the Bezier polygon or the control point.

(Refer Slide Time: 12:15)



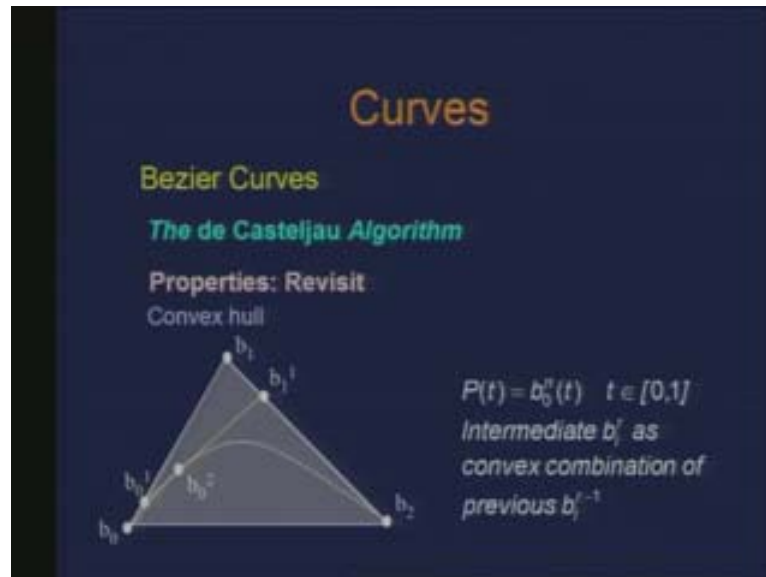
Hence, if I apply affine transformation to these Bezier polygon points then the curve computed is the same as if I had applied that affine transformation to the curve itself. So these are equivalent. So how do we see that in the formulation using de Casteljau algorithm? It is actually from the direct consequence of the algorithm itself because de Casteljau algorithm is employing linear interpolations. And what are linear interpolations? These are just the affine maps. It is actually a sequence of linear interpolation which is what is being used in de Casteljau algorithm this is a direct consequence, affine invariance is a direct consequence. Linear interpolation uses just the ratios and that is what affine maps are.

This property of affine invariance is inherent to the algorithm itself. Now what happens if I am talking about projective invariance. So our Bezier curves are invariant to projective transformation. Here what we have done is we have applied affine transformation and we observed that there is no difference if I had applied to the Bezier polygon or to the curve. These are the two same things.

Now if I apply a projective transformation would I have got the same result? That means, if I apply projective transformation to the Bezier polygon and obtain the curve or I straightaway apply the projective transformation to the curve, **then are these two the same?** The answer is no. So we will see that some of the curves can be made projective transformation invariant at a later stage.

The Bezier curves are not **projectively** invariant. Now the other property which we looked at was the convex hull property. So the convex hull property defines that the curve which gets formed resides inside the convex hull of the control polygon. This was the convex hull property. And due the nature of the Bernstein coefficients they being non negative and summing to 1 this property was easy to observe in that formulation.

(Refer Slide Time: 15:48)

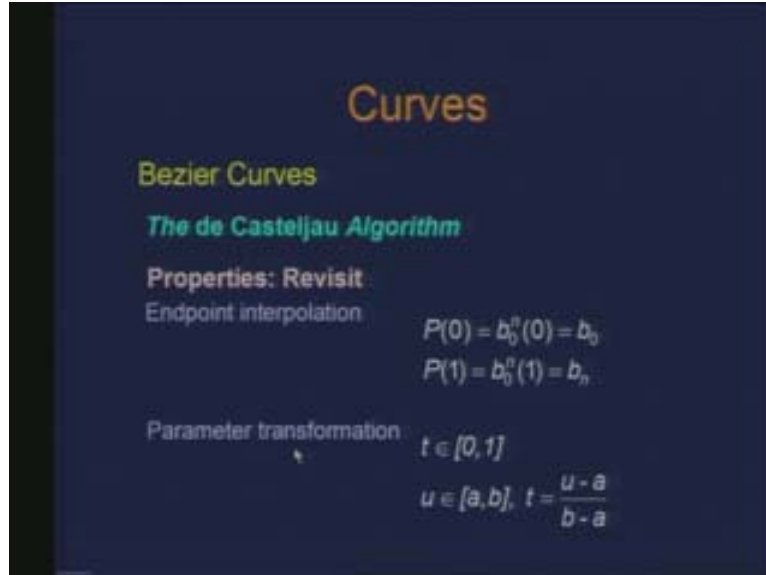


Now let us try to see the same thing in de Cateljau algorithm formulation. So once again if we just look at the construction mechanism which is to use successive application of linear interpolation, so what do I have is, this is the definition of the curve, this bn0t finally gives me the curve $P(t)$.

And if I observe that each of these intermediate points which I obtain after applying the linear interpolation are nothing but convex combination of the previous points of the level r minus 1. It is convex combination because I am just using linear interpolation 1 minus t and t . So, again this successive application of linear interpolation would result all the points which are getting computed to reside in the convex hull. Again this is inherent to the method of using linear interpolation.

The other properties like end point interpolation is also fairly easy to observe because if I just look at the point evaluation of b_0^n at t is equal to 0 it is nothing but b_0 which is the first control point of the Bezier polygon and similarly the evaluation at t is equal to 1 is the last Bezier point which is b_n . So this is also fairly easy to observe.

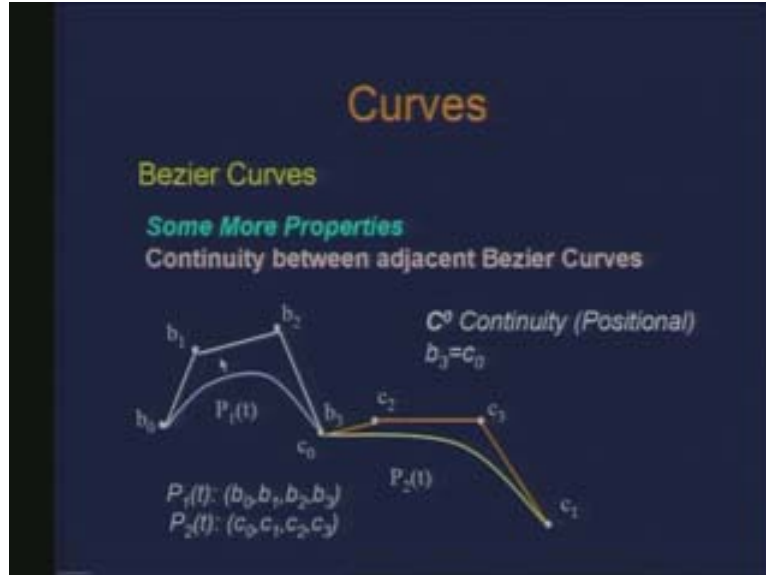
(Refer Slide Time: 18:21)



And if you look at again the parameter transformation that means instead of having the definition t defined in the span 0 and 1 if I have the parameter u defined in interval a and b it does not really change the method or the algorithm. All I need to do is make this affine transformation of the parameter, obtain t as nothing but the ratio of u minus a to b minus a . So most of the properties which we looked at in the case of Bernstein polynomial formulation can also be observed and asserted using de Casteljau algorithm.

Here are some more properties of Bezier curve in general. Here we are considering a scenario where we want to have a definition of one Bezier curve defined as from b_0 b_1 b_2 and b_3 this defines a curve $P_1(t)$ and I have another curve defined using set of control points as c_0 c_1 c_2 c_3 which is $P_2(t)$.

(Refer Slide Time: 20:32)



Now if I am talking about the continuity at the junction point of these two curves the first thing I can look at is the positional continuity where the first point of the second curve should match with the last point of the first curve because that is the minimum requirement of the two curves to be joined. So we are talking about positional continuity which we call as c_0 continuity. This is the first curve.

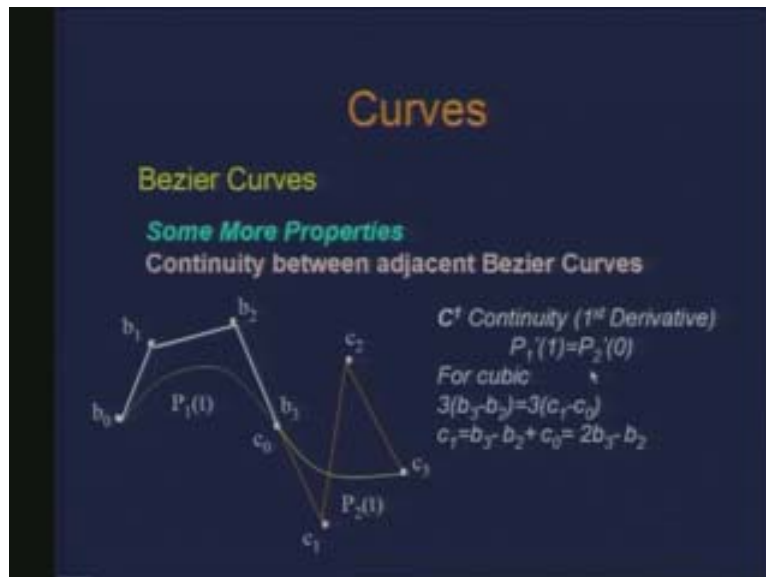
We are not looking at a new construction. We are basically joining two curves. This is a cubic Bezier curve, this is another cubic Bezier curve and if I have to define a joining here how I can define various kinds of joining looking at the desirable continuity I have. So if I have to design the two curves which have just the positional continuity then I can place these four points in anyway I want. But if I have a higher order of continuity then I need to do something else. So this is the minimal configuration of joining at this point which is only the positional continuity. So clearly when I see the effect of this I have this curve and another curve so I see some sort of a sharp change here which does not look smooth. Here again I am talking about smoothness in terms of visual sense.

(Refer Slide Time: 22:49)



So the curve does not look smooth. So, now if I want to make this curve look smooth I need higher order of continuity. Here we considered only the positional continuity then we have to perhaps look at the derivatives, the continuity in derivatives. That is what gives us C^1 continuity where I look at the continuity in terms of the first derivative of the curve.

(Refer Slide Time: 23:41)



Here, again if I have the first curve defined through the Bezier polygon b_0, b_1, b_2, b_3 and the second curve c_0, c_1, c_2, c_3 then here if you look it looks smoother. The transition from one curve to the other curve is smoother. And in order to establish this I would require

the condition being satisfied that the first derivative at the last point for the curve should be the same as the first derivative at the first point of the curve.

And if you go back and look how we found out the derivatives of the curve and do the evaluation at t is equal to 1 for the first curve and at t is equal to 0 for the second curve would in turn give you a relationship between the control points of the two Bezier curves. In fact for the first derivative you remember that it was just the first and the last spans of the polygon, these were the derivatives and that is what is being used here. This is the last span of the first Bezier curve and this is the first span of the second curve.

Therefore, this would give me the desirable location for the control point c_1 . Remember if I am talking about c_1 continuity it already uses c_0 continuity. So I have already established that c_0 is the same as b_3 . Hence for the purpose of designing the c_1 continuous curve the question comes as to where do I locate c_1 and that is what I obtain using this condition. This is how I get a curve which is c_1 continuous. The continuity is being evaluated at the junction point. But actually if you look at the differentiability of this with respect to t it is continuous and so is here.

Now the only constraint which has been imposed is **[orciba]** which necessitates locating the point c_1 . And as long as the c_2 and the c_3 points are concerned you could locate anywhere you want depending on what kind of shape you are looking at. Now I want to have c_2 continuous which would require continuity in second derivative. So again if I do the evaluation of the second derivative for the two curves and do the evaluation at the last point of the first curve and the first point of the second curve I would basically get a relationship of the relevant control points on the two sides.

Now, again given the fact that c_2 continuous curve is going to be c_1 and c_0 continuous I already know certain information about the point c_0 and c_1 , they have already been located satisfying those conditions. Hence using that I can evaluate c_2 .

(Refer Slide Time: 28:23)

Curves

Bezier Curves

Some More Properties

Continuity between adjacent Bezier Curves

C² Continuity (2nd Derivative)

$$b_1 - 2b_2 + b_3 = c_0 - 2c_1 + c_2$$

c_0, c_1 are known from C⁰ and C¹ continuity conditions

$$c_2 = b_1 - 4(b_2 - b_3)$$

Now you see that c_2 is given as b_1 minus $4(b_2$ minus $b_3)$ because c_0 is b_3 and again c_1 minus c_0 is the same as b_3 minus b_2 so that gives you c_1 . Hence you just use that information to get c_2 . So how does it look visually? Now I have the first curve $P_1(t)$ given as this through b_0 b_1 b_2 b_3 and the second curve given as P_2 through c_0 c_1 c_2 c_3 .


(Refer Slide Time: 29:16)

Curves

Bezier Curves

Some More Properties

Continuity between adjacent Bezier Curves



C² Continuity (2nd Derivative)

$$c_0 = b_3$$

$$c_1 - c_0 = b_3 - b_2$$

$$c_2 - b_1 = 4(b_3 - b_2)$$

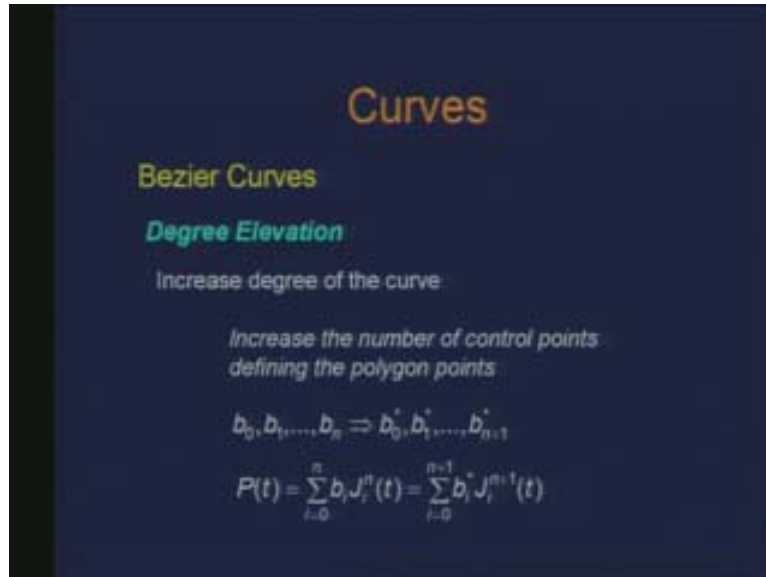
Now if you look at the relationship of the control points in the two curves c_0 is the same as b_3 satisfying c_0 continuity and c_1 minus c_0 is equal to b_3 minus b_2 . Basically all it says is, symmetrically if you want to see that this leg of the polygon in the first case is the same as this leg of the polygon and they are collinear. Whereas if you see here the

evaluation of c_2 can also be written as, c_2 minus b_1 as $4(b_3 \text{ minus } b_2)$ just rewrite what you have obtained earlier for c_2 . Therefore if you just look at this as a vector c_2 minus b_1 is parallel to b_3 minus b_2 . Or all these five points are collinear this, this, this, this, this, that is also a possible scenario in order to have this curve to be c_2 continuous. So what you observe here is that if I join this line here this is parallel to this. So this is how you can obtain the c_2 continuous curve. Of course from the users' perspective when you are designing curves is going to impose constraints where you can locate these points. That takes away the flexibility of positioning these Bezier points. So, for most practical purposes people actually just want to design c_1 continuous curve.

There is another type of continuity which is if I go back to the c_1 continuity here again (Refer Slide Time: 31:59) for a c_1 continuous if you want to look at geometrically this leg is the same as this leg both in direction and size. But if I had the two legs collinear but not necessarily of the same magnitude that means this leg could be longer go here then again I would see that the curve would look smooth as far as the visual sense is concerned. So at times we call that as visual continuity or geometric continuity g_1 . Instead of calling that as c_1 we call that as g_1 . Thus, all we are concerned there is of the direction and not necessarily of the magnitude. It is just a variation of c_1 continuity.

Now going to another feature of these Bezier curves is that even for the purpose of facilitating the design of the curves as we observe that the degree of the curve is basically defined by the number of control points we have. And control points are the control handles for designing the shape of the curve. I can displace those points wherever I want. And larger is the number of control points more is the control I have on to the shape of the curve. Therefore, at times what may be desirable is that I construct a curve of a certain degree let us say cubic which has four control points and suddenly I feel that I should have one more control point because that will help me redesign the curve or change the shape of the curve. So additional control point basically implies changing the degree of the curve because the degree of the curve is directly related to the number of control points I have.

(Refer Slide Time: 34:59)

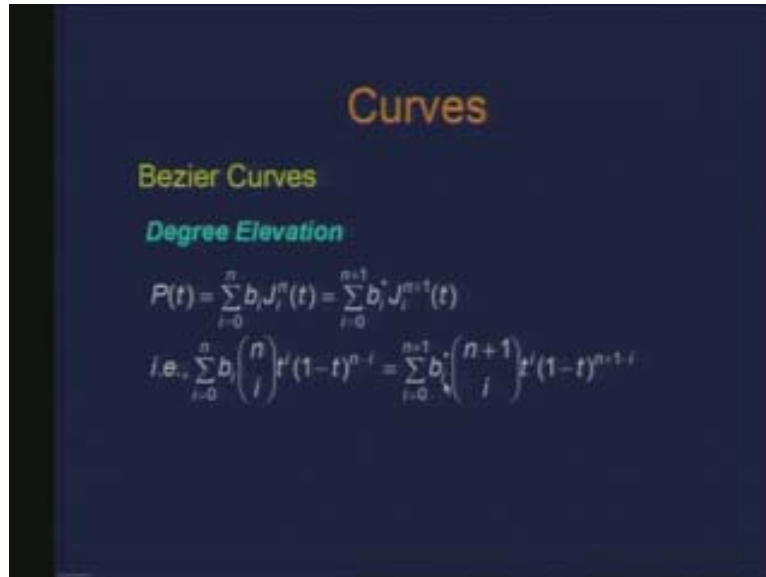


Hence, if I am trying to change the control points, if I want to increase the control points I am basically implying that there is an elevation of the degree of the curve. This could be useful in certain scenarios. Even in the case of joining two curves what I may do is first match the degree of the two curves and in order that I match the degree of the two curves I may have to elevate the degree of one of the curves. So there could be various scenarios there we may use this feature. What are we saying here is that we increase the degree of the curve which basically implies increasing the number of control points of the Bezier polygon.

Hence, if I have a curve defined through the Bezier points b_0 to b_n now I am getting a different set of control points as b_0 star to b_n plus 1 star. First I want to have the same curve defined using a different set of control points with elevated degree. I am not changing the definition or the shape of the curve.

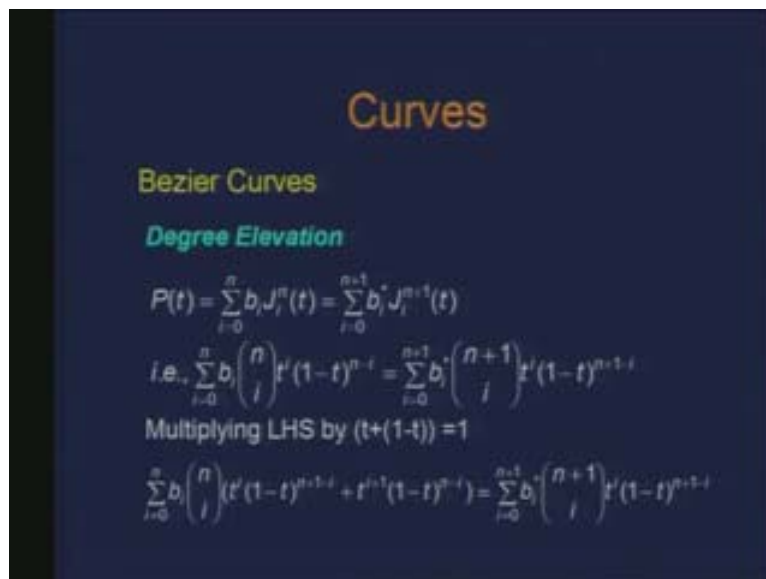
For the given configuration I just want to have the degree elevated which basically means that these two curves the curves obtained using this set of control points and the curve obtained using this set of control points should be the same as far as the curve is concerned. This basically means that I have these curves through these points, these curves through these points and I have these curves as the same. And what I am trying to do here is I already have this left hand side I am trying to locate these b stars which is the set of new control points with an elevated degree for the same curve. So I just basically establish this relation to obtain the b_i stars.

(Refer Slide Time: 38:20)



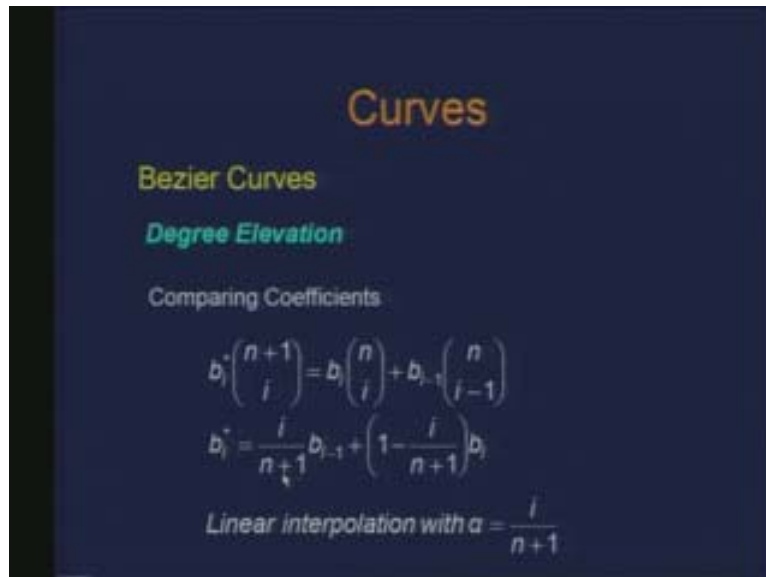
How do we proceed? I equate these and I expand the terms for these polynomials here and then in order to be able to get the coefficients b_i^* I need to have the same degree here. First of all make I need to make this degree high so I have to use some trick. Therefore multiply by unity and that should contain t 's and whatever I need. So I use this identity $t + 1 - t$ which is equal to 1 and multiply the left hand side with this. Now I can equate coefficients of the two sides of say t to the $i+1$ minus t to the $n+1$ minus i .

(Refer Slide Time: 38:55)



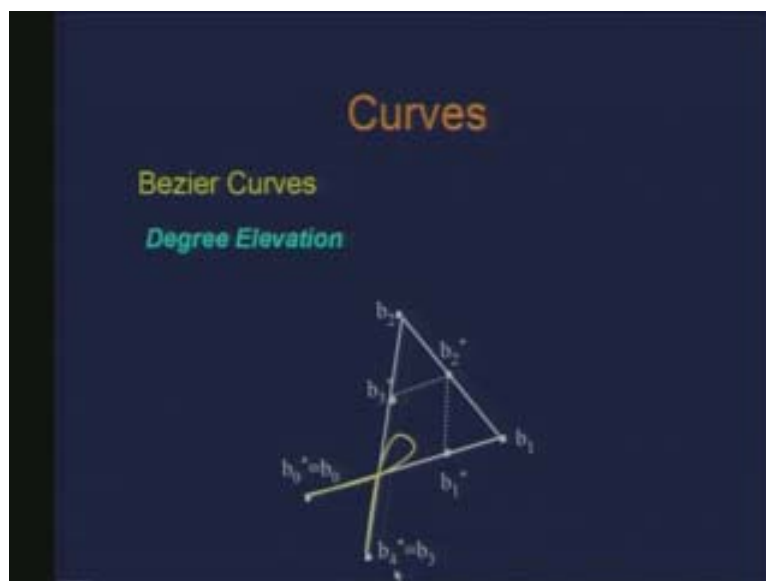
So if I compare the coefficients on the two sides of t to the i and $1 - t$ to the $n+1$ minus i this is what I get.

(Refer Slide Time: 39:34)



Now further simplifying this, this actually gives me this b_i^* as nothing but i by n plus 1 times b_{i-1} plus 1 minus i by n plus 1 times b_i which is nothing but some sort of a linear combination of these Bezier points b_{i-1} and b_i . So if I use the parameter like α equals to i by n plus 1 this is just a linear combination using the parameter α . So just observe an example here. Let us say I have a curve defined through these control points b_0, b_1, b_2, b_3 .

(Refer Slide Time: 40:40)



Again this is a hand drawn curve so you should not see this to be an exact curve. So the exact computed curve may look different. Now if I want to elevate the degree of this

curve using the same result which we had would basically give me these b stars so this point will be some linear interpolation by 1 by 4 using, this is by $\frac{1}{2}$ and this is by 3 by 4 or whatever i by n plus 1 is what we are saying.

This basically does the localization of the b stars for the same curve and obviously the first and the last points have to be the same. Now tell me what happens if I do this degree elevation repeatedly. From here I can go further I have elevated the degree from 3 to 4 and now I do the next elevation and next elevation and next elevation, it will converge to the curve. You can actually prove mathematically also.

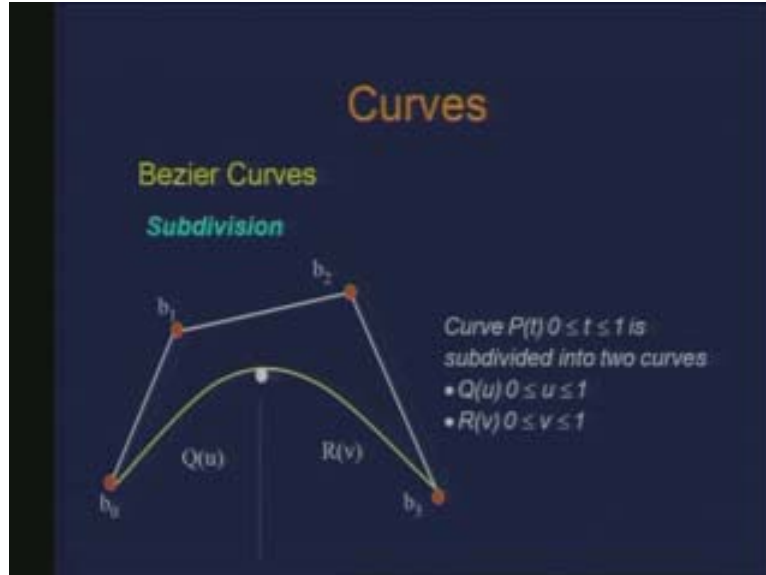
43:15.....There we are trying to do an interpolation of set of points and if I try to have the degree as the number of points then I will have [vigliness] control points that is the number of points which I am trying to interpolate. Often that is what the case you will have in Lagrange interpolation. So you have a very high degree of the curve which you obtain. So this is not the same type, this is not an interpolation here. But of course there are other issues related to elevating a degree.

First of all the moment you increase the degree of the curve computation is an issue so the simplicity is going away. But from the purpose of the design of the shape of the curve you have better control because you just have more number of points to move. And remember that we have the property of something like pseudo control. So the pseudo control gets more influenced by adding more number of points because you are sort of distributing that influence. If I just had a cubic curve if I displace one point it will affect a larger range of the parameter. If I have additional points for the same effect the range is going to be smaller.

Another question I have is we have talked about degree elevation where we were trying to increase the degree of the curve. So can we also think of reducing the degree of the curve? The problem is that, reducing the degree means you are reducing the number of control points and of course retaining the necessary properties of the curve. So it actually turns out that it is not always possible. You might be able to do it only for certain situation. For instance here I can do the reverse construction. From here I go and obtain this, from here I go obtain this but not always.

Subdivision of the curve: This is another feature or property of the Bezier curve. So what do I mean by the subdivision of the curve is let us say I want to divide in two parts.

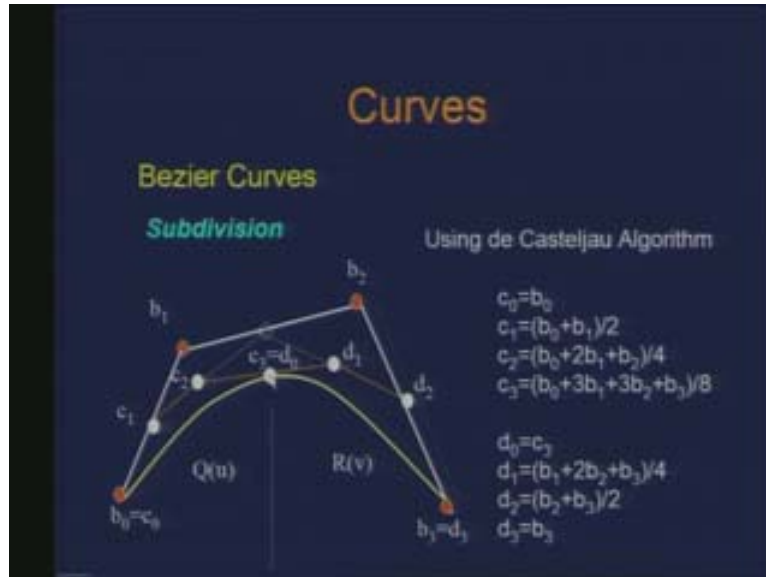
(Refer Slide Time: 47:17)



And when I say two parts I am basically referring to the range of the parameter. This curve $P(t)$ defined between 0 and 1 and I define another set of two curves $Q(u)$ and $R(v)$ which are individually defined in the unit interval for the parameter whereas this actually is referring to a curve up to t is equal to $1/2$ and this $R(v)$ is referring to the other half. So basically here if you look at this curve I am trying to divide this curve into this $Q(u)$ and $R(v)$. If you recall de Casteljau algorithm here is this point. This was the point where the subdivision was to take place.

So, on the left side I needed set of control points which could define this curve and I needed another set of control points which could define this curve. That is what it implied the subdivision of the curve. Now if I consider the evaluation of the curve at t is equal to $1/2$ and if I see the construction from de Casteljau algorithm this is the point which gets located on the curve for the evaluation t is equal to $1/2$. And the other points which I obtain from the previous iterations of de Casteljau algorithm can actually act as the control points. If I do de Casteljau algorithm t is equal to $1/2$ this will get located here, this point gets located there and this point gets located there.

(Refer Slide Time: 48:52)

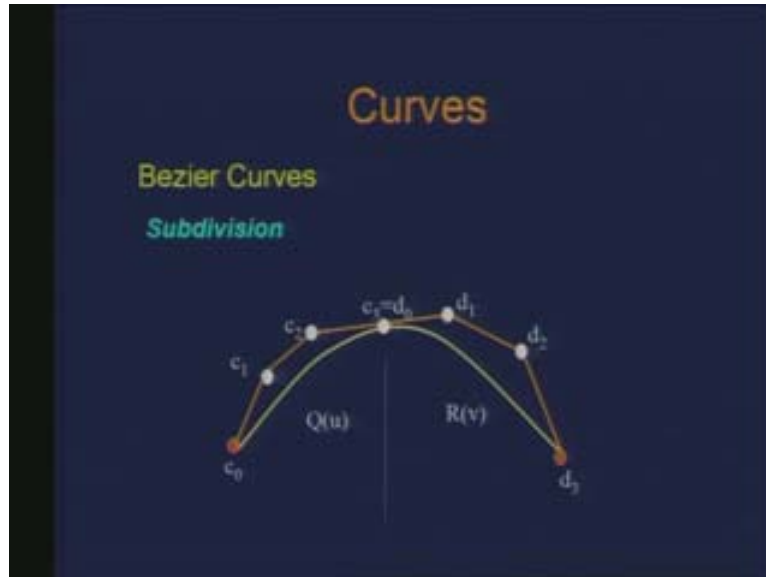


And further application gives me the point here the point here. One more application gives me the point on the curve. And this is the point where this subdivision is taking place and that is where I am subdividing the curve.

Now if I see this point and this point they define a set of control points for this curve. And similarly these two end points define the control points for this part of the curve. So the proof of this is a little involving but you can try.

Now, again if I have to do a repeated subdivision I will converge to the curve. This is like some sort of a corner cutting when you look at the way de Casteljau algorithm is working this corner is cut, this corner is cut. In the subsequent subdivision there will be another corner cut of these and eventually I will get the curve. So it says that if I have a polygon and I just keep chopping these corners whenever they get built ultimately I will get a smooth curve. It sort of intuitively matches with what we see as the corner cutting. This is the sort of result we have for the sub division.

(Refer Slide Time: 52:27)



(Refer Slide Time: 52:40)



If you look at Bezier curves what we figure out is that, as far as the local control is concerned is limited. There is this pseudo local control we have but a change in the position of a point in the Bezier polygon results change in the entire curve which is sort of a global effect. It may not be a very convenient way of designing larger free form curves.

Exact local control is a curve defined in this fashion and if I move this point I observe that the entire curve changes.

What I may just want is only a part of this curve changes rather than the entire curve. So there is a sort of a range of influence with respect to this point. It is there in some form. Actually every point on the curve changes so I cannot do away by not computing those points. As far as the curve is concern I have to do a re-computation of the entire curve. Therefore one thing is that we have no local control or limited local control. And the other thing is that the degree of the curve is fixed by the number of points I have in the control polygon and that may again be an issue for various applications. So it may be nicer to have a de coupling of the control points I used for the gross shape of the curve and the degree of the curve which I designed. These limitations are overcome in B Splines.