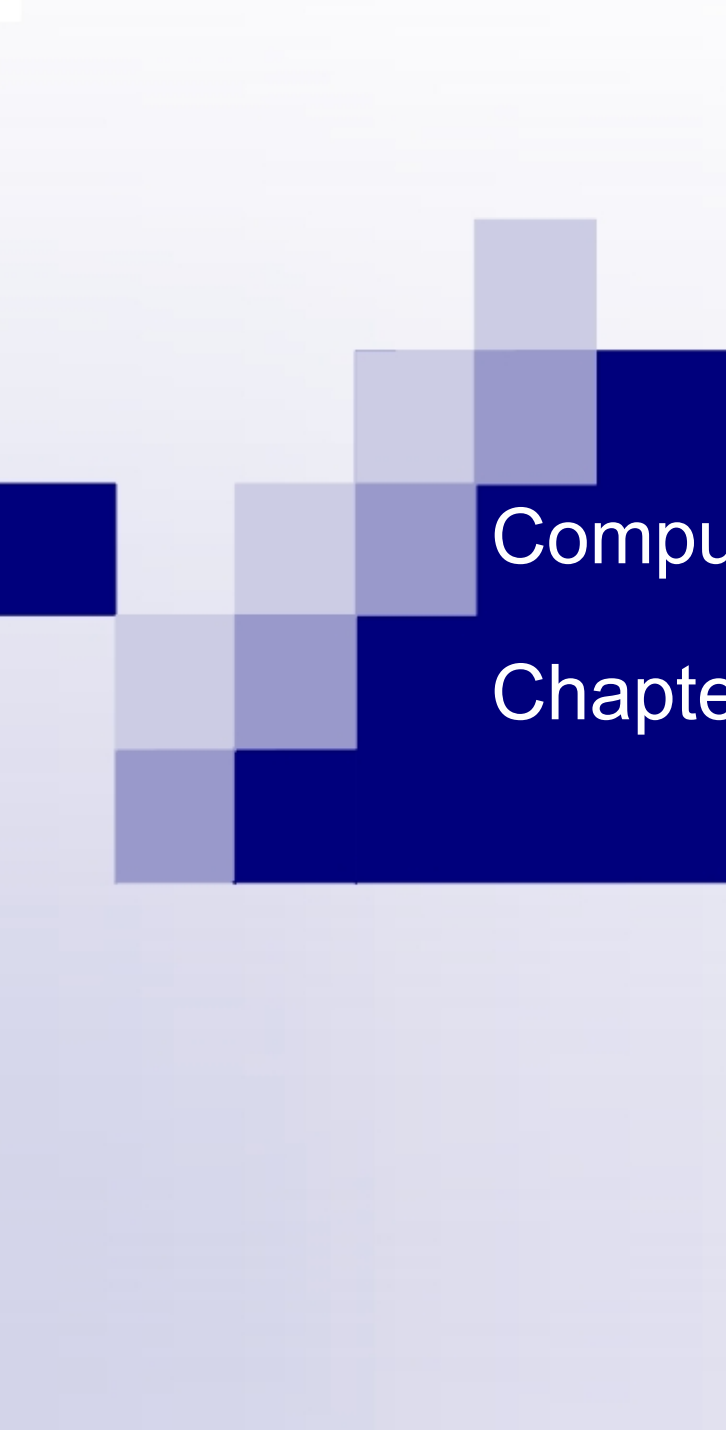




Computer Graphics using OpenGL Hill

Chapter 11: Curve and Surface Design Part II



Properties of Bezier Curves

- ◆ Bezier curves have some important properties that make them well suited for CAGD.
- ◆ These properties are later found to apply to B-splines as well.
- ◆ Exploring these properties and their proofs provides a great deal of insight into Bezier curves.

Endpoint Interpolation.

- ♦ The Bezier curve $P(t)$ based on control points $P_0, P_1, \dots, P_L$ does not generally pass through, or interpolate, all of the control points.
- ♦ But we have seen that it always interpolates P_0 and P_L .
- ♦ This is a very useful property, because a designer who is inputting a sequence of points thereby knows precisely where the Bezier curve will begin and end.

Affine Invariance.

- ♦ It is often necessary to subject a Bezier curve to an affine transformation in order to scale it, orient it, or position it for subsequent use.
- ♦ Suppose we wish to transform point $P(t)$ on the Bezier curve of the equation:
$$P(t) = \sum_{k=0}^L P_k B_k^L(t)$$
- ♦ to the new point $Q(t)$, using the affine transformation T . (T is represented by a 3 by 3 matrix in the 2D case, and by a 4 by 4 matrix in the 3D case.) So $Q(t) = T(P(t))$.

Affine Invariance.

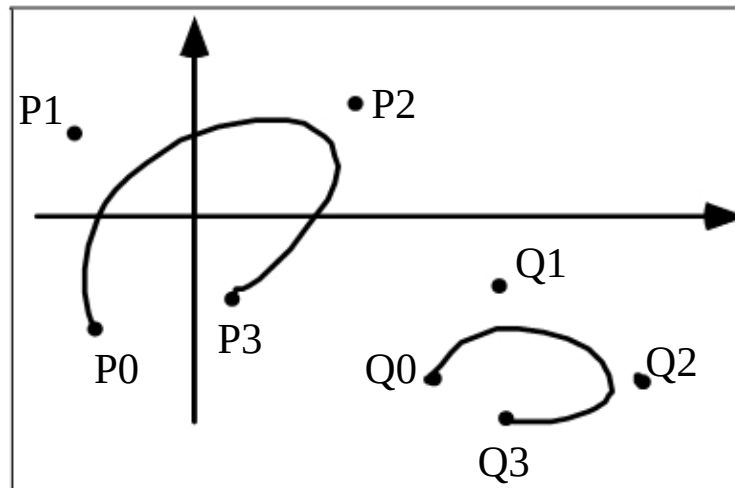
- It appears that to find $Q(t)$ at any given value of t we must first evaluate $P(t)$, and then transform it, effectively starting over fresh for each new t . But this isn't so.
- We need only transform the control points (once), and then use these new control points in same Bernstein form to re-create the transformed Bezier curve at any t !

That is:

$$Q(t) = \sum_{k=0}^L T(P_k) B_k^L(t)$$

Showing affine invariance.

- The figure shows a Bezier curve based on four control points $P_0, \dots, P_3$.
- These points are rotated, scaled, and translated to the new control points Q_k , and the Bezier curve determined by them is drawn.
- This curve is identical point by point to the result of transforming the original Bezier curve.



- 
- ◆ Bezier curves are affine invariant for a very simple reason: they are formed as an affine combination of points, and from Chapter 5 (Section 5.2.8) we know that an affine transformation preserves affine combinations.

Convex Hull Property.

- ♦ Another property that designers often rely on is that a Bezier curve, $P(t)$, never wanders outside its convex hull.
- ♦ The convex hull of a set of points $P_0, P_1, \dots, P_L$ is the set of all *convex combinations* of the points; that is, the set of all points

given by

$$\sum_{k=0}^L \alpha_k P_k$$

where each α_k is non-negative, and they sum to 1.

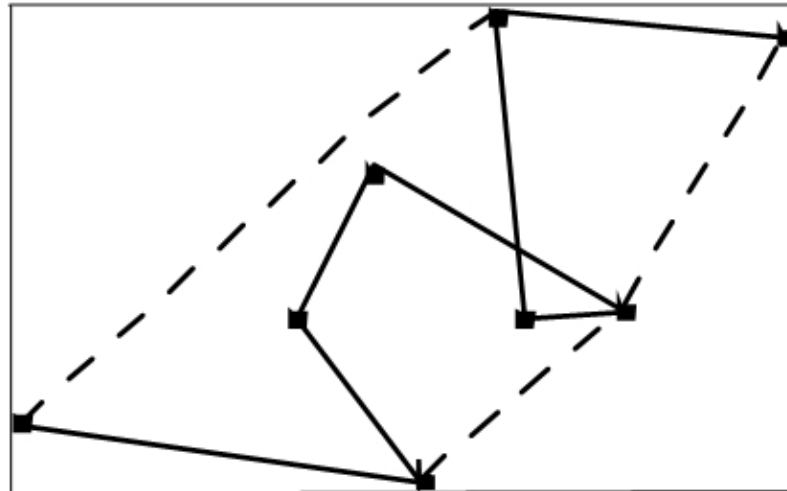


Convex Hull

- ♦ The convex hull property also follows immediately from the fact that each point on the curve is the result of tweening two points that are themselves tweens, and the tweening of two points forms a convex combination of them.

Convex Hull

- ♦ The figure below illustrates how the designer can use the convex hull property.
- ♦ Even though the eight control points form a jagged control polygon, the designer knows the Bezier curve will flow smoothly between the two endpoints, never extending outside the convex hull.



Linear Precision.

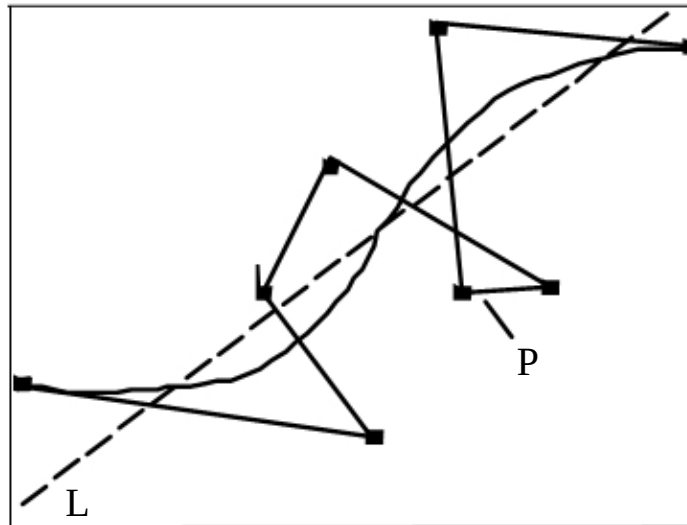
- ♦ Can a Bezier curve be a simple straight line?
- ♦ The convex hull property shows that it can: If all of the control points are placed in a straight line their convex hull collapses to a line.
- ♦ The Bezier curve is therefore “trapped” inside this hull and so must also be a straight line.
- ♦ The property of achieving a straight portion to a curve by properly positioning the control points is called **linear precision**.



Variation Diminishing Property.

- ◆ Roughly speaking, Bezier curves can't "fluctuate" more than their control polygon.
- ◆ More precisely, no straight line (or, in three dimensions, no plane) can have more intersections with a Bezier curve than it has with the curve's control polygon.

- ♦ The figure shows a Bezier curve based on the control polygon P . Line L cuts through P five times but through the Bezier curve only three times.
- ♦ No line can be found that cuts the curve more often than it cuts P . This can be proved in general for Bezier curves.



- 
- ♦ It is a useful property for curve designers as they lay down a control polygon:
 - ♦ They can predict with confidence that the resulting Bezier curve will not undulate wildly or exhibit extra bends and curves.
 - ♦ On the other hand, some alternative curve design techniques, such as certain interpolation schemes, do produce unruly fluctuations.

Derivatives of Bezier Curves.

- A curve can exhibit corners and other abrupt changes when its derivatives with respect to t have discontinuities.

- For a Bezier curve one can show that the first derivative is

$$\mathbf{p}'(t) = L \sum_{k=0}^{L-1} \Delta P_k B_k^{L-1}(t)$$

where $\Delta P_k = P_{k+1} - P_k$

- ♦ So the velocity is another Bezier curve, built on a new set of control vectors ΔP_k .
- ♦ We simply “difference” the original control points in pairs to form the control vectors of the velocity.
- ♦ Note from the form $B_k^{L-1}(t)$ that taking the derivative lowers the order of the curve by 1: For instance, the derivative of a cubic Bezier curve is a quadratic Bezier curve.



- ♦ ***Finding Better Blending Functions.***

- ♦ ***The B-Spline Basis Functions***

- ♦ ***Surface Design***



Finding Better Blending Functions.

- ◆ It might appear that Bezier curves provide the ultimate tool for designing curves.
- ◆ An endless variety of smooth curves can be fashioned by placing control points judiciously in the plane.
- ◆ But we see that Bezier curves by themselves do not provide enough flexibility in curve design.

Finding Better Blending Functions.

- ♦ One problem is that the degree of the Bernstein polynomials used is coupled to the number of control points: a Bezier curve based on $L+1$ control points is a combination of L -degree polynomials.
- ♦ High degree polynomials are expensive to compute, and are vulnerable to numerical round-off errors.
- ♦ We want the designer to be free to use as many control points as desired, even 40 or more.

The Problem of Local Control.

- An even more significant problem is that Bezier curves do not offer enough **local control** of the curve shape.

The Figure 11.19 shows a situation where five control points are used to fashion a Bezier curve (solid), which deviates somewhat from the desired curve (dashed) near $t = 1$.

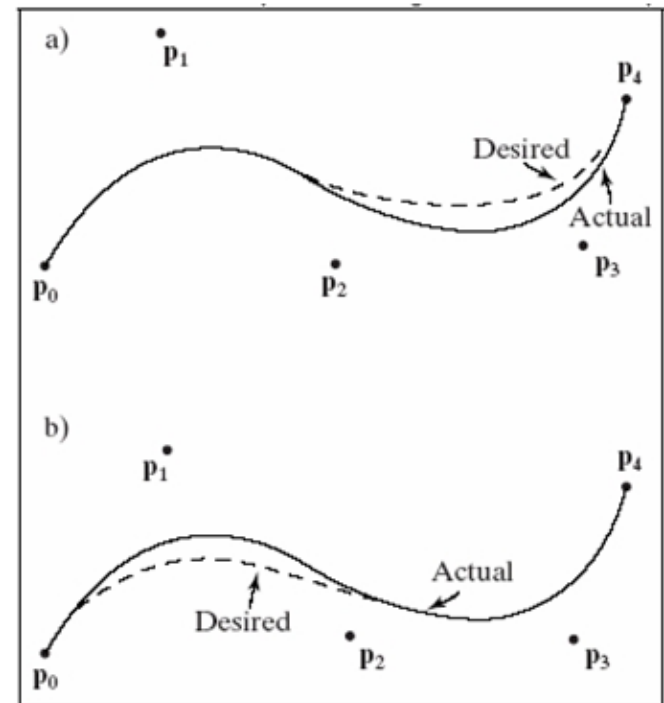


Figure 11.19

- ♦ To correct this the user would move P_2 and P_3 up somewhat to force the Bezier curve closer to the desired curve.
- ♦ But, as shown in Figure 11.19, this also affects the shape of the left half of the curve, forcing it some distance away from the desired version.



Control Point problem

- The problem is that a change to any control point alters the *entire* curve.
- This arises from the nature of Bernstein polynomials: each one is “active” (meaning non zero) over the entire interval $[0, 1]$.
- The interval over which a function is nonzero is often called its **support**.

- 
- ◆ Because every Bernstein polynomial has support over the entire interval $[0, 1]$, and the curve is a blend of these functions, it follows that each control point has an effect on the curve at all t -values between 0 and 1.
 - ◆ Therefore, adjusting any control point affects the shape of the curve everywhere, with no local control.

- Contrast that with the more favorable set of blending functions illustrated in Figure 11.20.

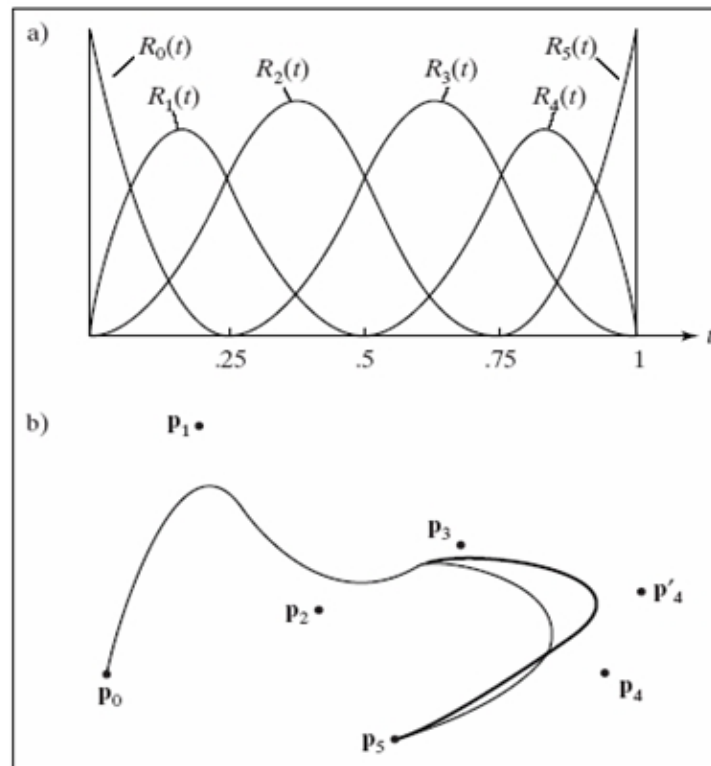


Figure 11.20

Blending functions

- The six blending functions, $R_0(t)$, $R_1(t)$, $\dots$, $R_5(t)$, are shown, each having support that is only a part of the interval $[0, 1]$. For instance, the support of $R_0(t)$ is $[0, .25]$ and that of $R_3(t)$ is $[.25, 1.0]$.
- In fact, at any value of t no more than three of the blending functions are active.

- ♦ Consider using these blending functions to build a curve, $V(t)$, based on six given control points, $P_0, P_1, \dots, P_5$.
- ♦ We use the same kind of parametric form as for Bezier curves:

$$V(t) = \sum_{k=0}^5 P_k R_k(t)$$

- ♦ Figure 11.20b shows the curve for an example set of control points.
- ♦ At each t the position $V(t)$ depends on no more than three of the control points. In particular, for all t in $[0.75, 1.0]$ only the points P_3 , P_4 , and P_5 control the shape of the curve.
- ♦ If the single control point P_4 is moved to P'_4 , only a portion of the curve will change, shown dashed.
- ♦ Thus this set of blending functions does give some local control to the control points.

Wish List for a set of Blending Functions.

- ♦ The blending functions should:
 - ♦ be easy to compute and numerically stable;
 - ♦ sum to one at every t in $[a,b]$;
 - ♦ have small support to offer local control;
 - ♦ interpolate certain control points, chosen by the designer;
 - ♦ be “smooth” enough.
- ♦ More details in Hill pg 620-621

B-spline blending functions

- ♦ We wish to define the B-spline blending functions, $R_k(t)$, in a way that lends some intuition to them and, in addition, leads to a straightforward computer implementation.
- ♦ Although the literature offers many different approaches to formulating B-splines, there is a single formula that defines all the B-spline functions of any order.
- ♦ It is a recursive relation that is easy to implement in a program and is numerically well behaved.



Defining the B-Spline Basis functions

- ◆ Each B-spline function is based on polynomials of a certain order, m .
- ◆ If $m = 3$, the polynomials will be of order 3 and thus of degree 2, and so they will be quadratic B-splines.
- ◆ If the order is $m = 4$, the underlying polynomials will be of degree 3, or cubic.
- ◆ These are the two most important cases, although the formulation allows us to construct B-splines of any order.

Definition of B-spline Functions

- ◆ It is useful to make the order of a B-spline function explicit in the notation, and so instead of saying simply $R_k(t)$, we denote the k -th B-spline blending function of order m by $N_{k,m}(t)$.
- ◆ The detailed shape of a B-spline curve depends on a list of t -values called a “knot vector”: $\mathbf{T} = (t_0, t_1, t_2, \dots)$ (see p. 626) chosen by the designer, and
- ◆ the formula for a B-spline curve is:

$$P(t) = \sum_{k=0}^L P_k N_{k,m}(t)$$

Recursive Definition of the K-th B-spline Basis Function

- ◆ Summarizing the ingredients to this point, we have

- a knot vector $\mathbf{T} = (t_0, t_1, t_2 \dots)$

- $(L+1)$ control points P_k

- the order m of the B-spline functions

- ◆ The fundamental formula for the B-spline function $N_{k,m}(t)$ is

$$N_{k,m}(t) = \left(\frac{t - t_k}{t_{k+m-1} - t_k} \right) N_{k,m-1}(t) + \left(\frac{t_{k+m} - t}{t_{k+m} - t_{k+1}} \right) N_{k+1,m-1}(t)$$

for $k = 0, 1, \dots, L$.

The First Order B-spline basis function

- This is a recursive definition, specifying how to construct the m -th-order function from two B-spline functions of order $(m - 1)$.
- To get things started, the first-order function must be defined. It is simply the constant function 1 within its span:

$$N_{k,1}(t) = \begin{cases} 1 & \text{if } t_k < t \leq t_{k+1} \\ 0 & \text{otherwise} \end{cases}$$

Note that this set of functions automatically sums to one at every t , so it is legitimate to use them in forming combinations of points.

Bezier Curves **Are** B-Spline Curves.

- ◆ Bezier curves were introduced earlier through two approaches: the de Casteljau algorithm and the Bernstein polynomials.
- ◆ We can now state a third approach: Bezier curves are also a special case of B-splines.
- ◆ This is so because the B-spline blending functions defined on the standard knot vector are in fact Bernstein polynomials when $m = L + 1$!

That is, $N_{k,L+1}(t) = B_k^L(t)$,

More Information on B-Splines

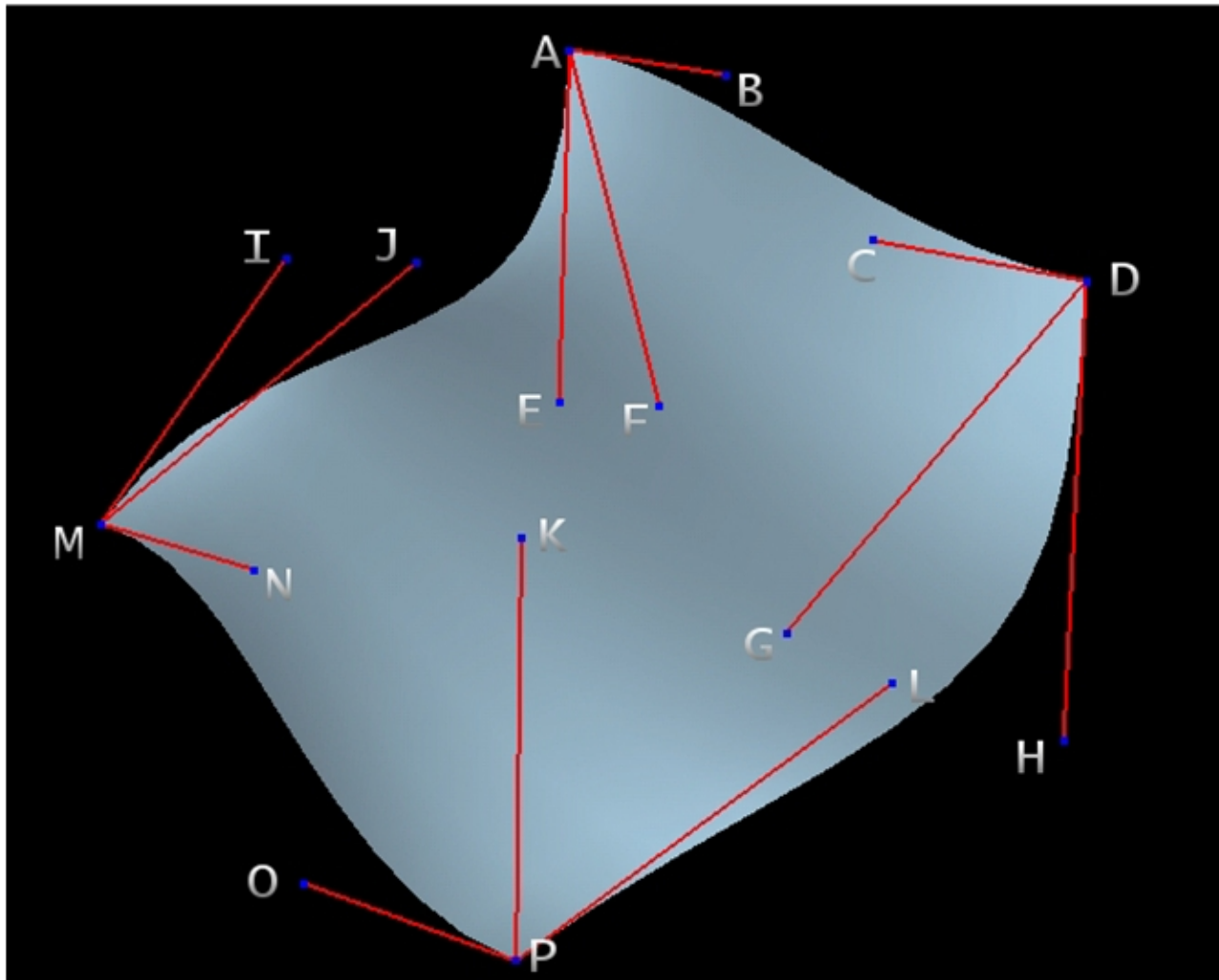
- ♦ More on B-Splines and Curves in Chapter 11.
- ♦ Now: **Surface Design: Modeling Curved Surfaces**



Modeling Curved Surfaces

- ♦ So far we have been examining how to represent and generate 2D and 3D curves.
- ♦ It is straightforward to extend these ideas to the generation of curved surfaces.
- ♦ Now consider how to use Bezier and B-spline curves to design these surfaces, developing powerful tools for creating a rich set of curved surfaces.

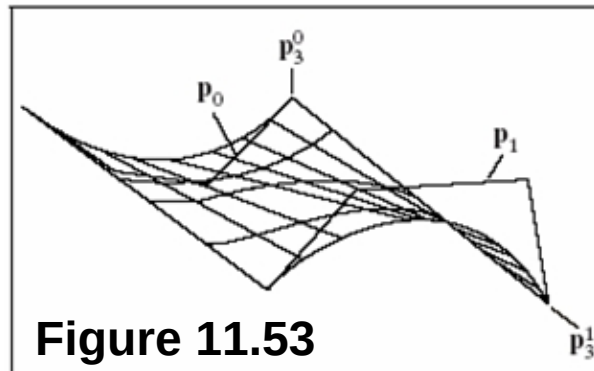
Example Surface



Ruled Surfaces based on B-splines

- ♦ It is particularly easy to work with ruled surfaces.
- ♦ A ruled surface is defined by two “end curves”, one using parameter u and one using parameter v . $P_0(u)$ and $P_1(u)$. These are connected at each value of u by a straight line.
- ♦ Therefore the parametric expression for a ruled surface is just a linear interpolation (or a “tweening”) between corresponding points on the two curves.
- ♦ We have: $P(u, v) = (1 - v) P_0(u) + v P_1(u)$

- ♦ The extension here is to choose $P_0(u)$ and $P_1(u)$ to be B-spline (or Bezier) curves.
- ♦ Figure 11.53 shows a ruled surface where both of the end curves $P_0(u)$ and $P_1(u)$ are cubic Bezier curves. $P_0(u)$ is based on the four control points $P_0^0, P_1^0, P_2^0, P_3^0$, and $P_1(u)$ is based on the four control points $P_0^1, P_1^1, P_2^1, P_3^1$.



- ♦ For a Bezier curve this surface is written as

$$P(u, v) = \sum_{k=0}^3 ((1-v)P_k^0 + vP_k^1)B_k^3(u)$$

- ♦ Visualize how this shape behaves. Its u -contours are straight lines joining corresponding points on the two Bezier curves. Its v -contours are Bezier curves whose control points are “tweens”

$$(1-v)P_k^0 + vP_k^1$$

of the control points of the two Bezier curves.

Surfaces of Revolution based on B-splines

- A surface of revolution is formed when a **profile** $C(v) = (X(v), Z(v))$ is swept about the z -axis.
- The resulting surface has parametric form:

$$P(u, v) = (X(v) \cos(u), X(v) \sin(u), Z(v))$$

- ♦ It is often convenient to express the profile using Bezier or B-spline curves.
- ♦ We do this by selecting $L+1$ control points (X_k, Z_k) and using them to create the curve:

$$(X(v), Z(v)) = \sum_{k=0}^L (X_k, Z_k) N_{k,m}(v)$$

An exquisite surface of revolution: The dome of the Taj Mahal

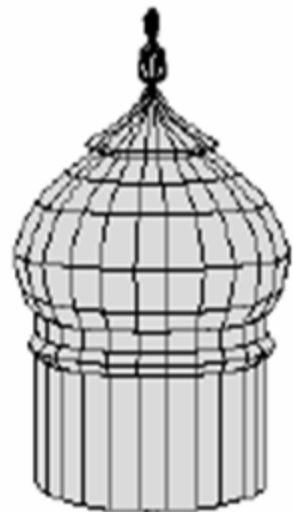
FIGURE 6.69 A surface of revolution - the dome of the Taj Mahal.



a)



b)



c)

Goblet Surface: Control Points

- ♦ Figure 11.54a shows an example profile for a goblet defined by a cubic B-spline curve.
- ♦ The control polygon is also shown.

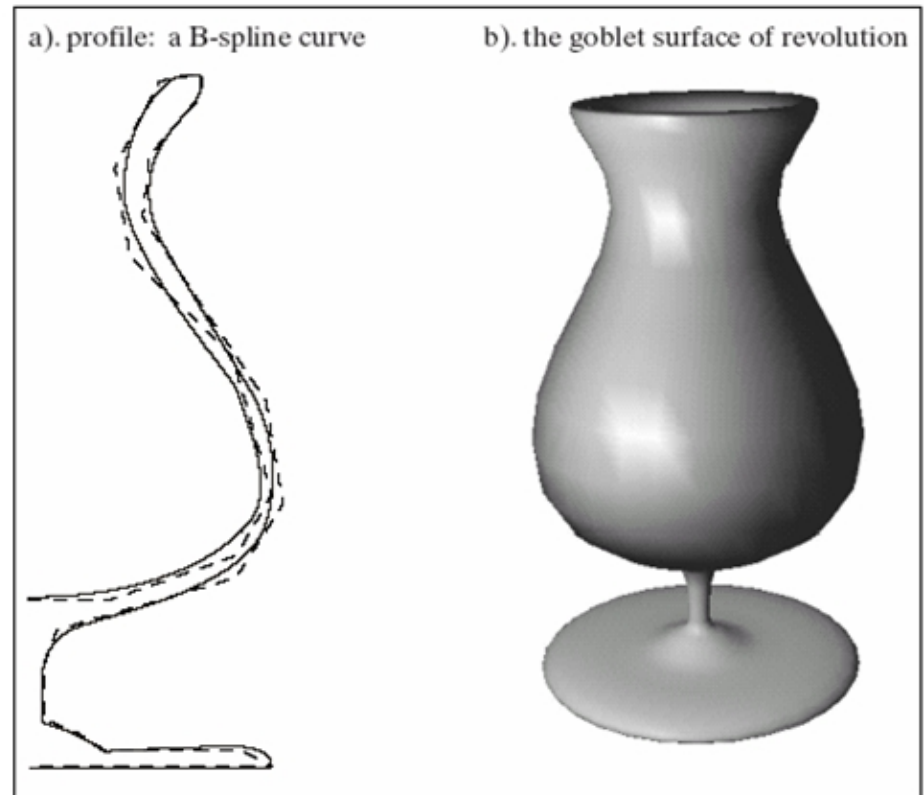


Figure 11.54

Goblet Surface

- ♦ Figure 11.54b shows the resulting surface of revolution.
- ♦ A mesh object for the surface of revolution was created as discussed in Chapter 6, and the mesh was drawn using the `Mesh :: draw()` method.

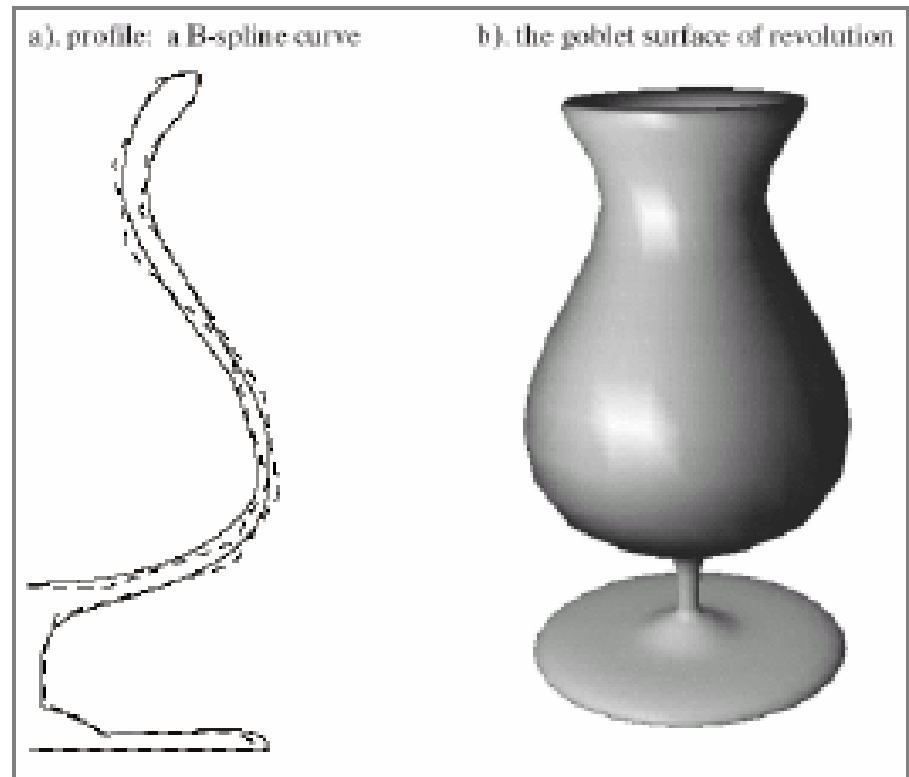


Figure 11.54

- ♦ Bezier curves may also be used for profile design.
- ♦ Figure 11.55a shows the profile of the body of the “teapot” originally designed by Martin Newell [blinn96, crow87].

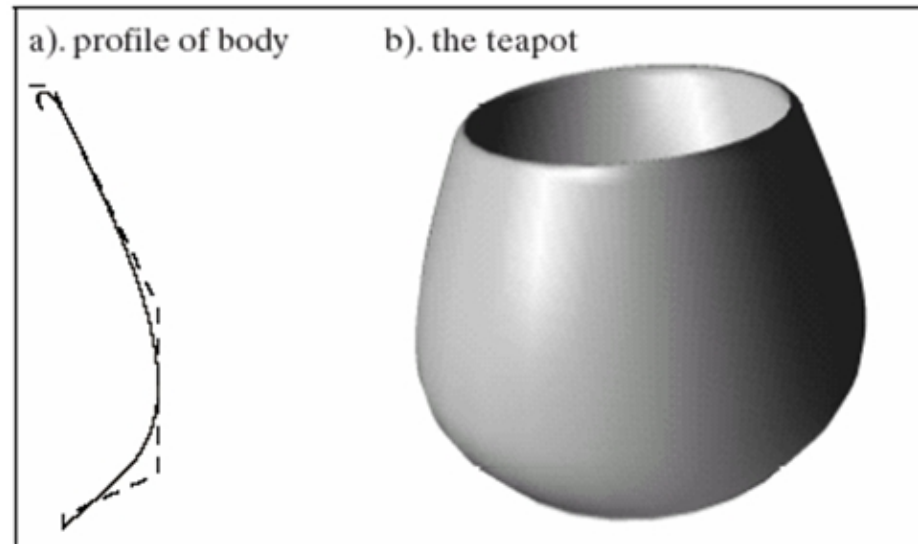


Figure 11.55

The Body defined by 3 Bezier curves

- The body profile consists of three Bezier curves, based on the 10 points displayed in the figure below.

i	x	z
0	1.4	2.25
1	1.3375	2.38125
2	1.4375	2.38125
3	1.5	2.25
4	1.75	1.725
5	2	1.2
6	2	0.75
7	2	0.3
8	1.5	0.075
9	1.5	0

Figure 11.56

- 
- ♦ The first Bezier curve is defined by points 0, 1, 2, and 3, the second defined by points 3, 4, 5, and 6, and the third by points 6, 7, 8, and 9.
 - ♦ Notice that the last segment of each curve is collinear with the first segment of the next curve.
 - ♦ This condition insures that the different Bezier curves blend together with G_1 continuity.

Designing the handle of the Teapot

- We have seen that the body of the teapot is a surface of revolution based on Bezier curves.

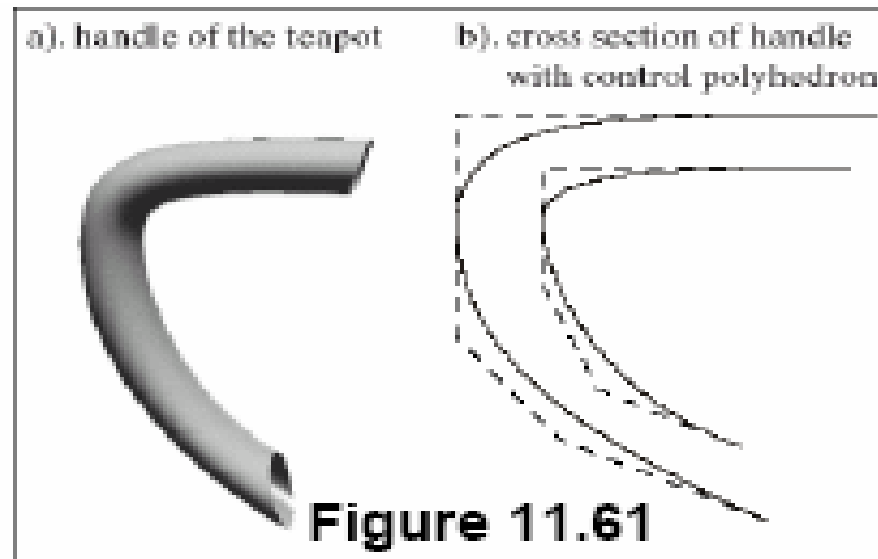
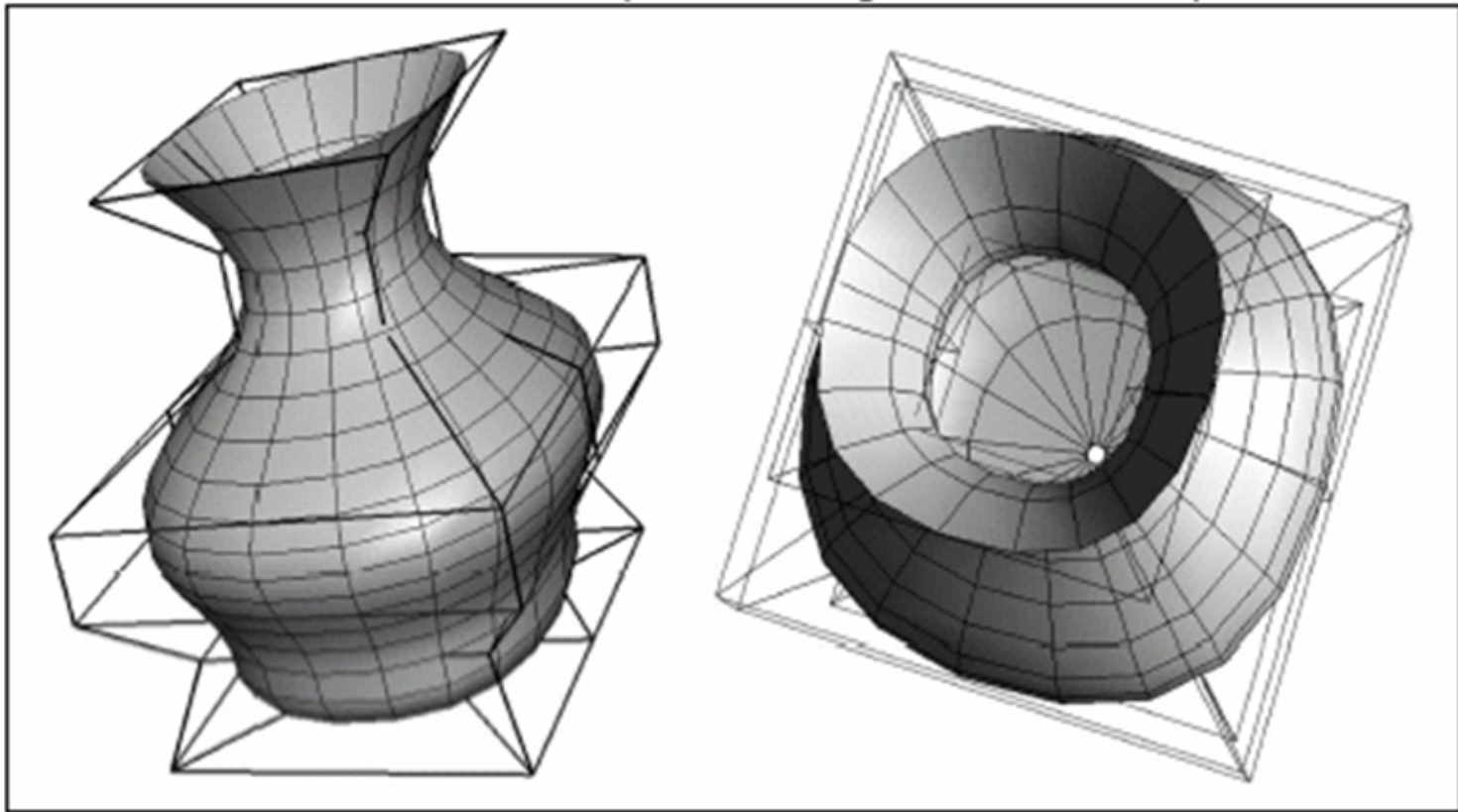


Figure 11.61a shows a 3D view of the handle of the teapot. Part B shows the handle in cross section, along with a cross-sectional view of the control polyhedron for the patches.

The handle of the Teapot

- ♦ The handle's surface is symmetrical about the xz -plane.
- ♦ There is an upper and a lower patch on the positive y -side of the xz -plane, and mirror upper and lower patches on the negative y -side of the xz -plane.
- ♦ More details in Hill pg. 659.

Example of a B-spline surface.



Example of a NURBS surface.

- ♦ See hill, p.639 for definitions and properties.

