



# Modelling and representation 4 – Bezier, B-spline and subdivision surfaces

---

- 5.1 Introduction
- 5.2 Bezier curves
- 5.3 B-spline curves
- 5.4 Rational curves
- 5.5 From curves to surfaces
- 5.6 Modelling or creating patch surfaces
- 5.7 Rendering parametric surfaces
- 5.8 Practical bezier technology for games
- 5.9 Subdivision surfaces
- 5.10 Scalability – polygon meshes, patch meshes and subdivision surfaces

# 5.1 Introduction

---

- This chapter is devoted entirely to a representational form where the primitive element – a **bi-cubic parametric patch** – is a curvilinear quadrilateral .
- Now patch representations are being used more and more as alternatives to the polygon mesh model.
- A net of patches can represent a complex with far fewer elements than a net of planar polygons.



# 5.1 Introduction

---

- Patch representations advantages:
  - It has the potential of 3D shape editing or modelling. Real time shape changing.
  - By using a fast patch to polygon conversion algorithm, it can be used with existing polygon rendering hardware.
  - It facilitates an **automatic** LOD representation.

# 5.1 Introduction

---

- Patch representations advantages:
  - If patch processing algorithms migrate onto hardware then the inherent economy of representation implies a **solution** to the CPU-graphics processor **bus bottleneck** .
  - The lighting resolution for static objects can be made independent of the LOD
  - It is an exact analytical representation.
  - It is a more economical representation.

# Parametric Cubic Curves

---

- Curve and surface approximation:
  - 1st degree approximation: polygon mesh
  - Higher degree approximation:
    - explicit function:  $y=f(x), z=f(x)$
    - implicit equation:  $f(x,y,z)=0$
    - parametric representation:  $x=x(t), y=y(t), z=z(t)$

# Parametric Cubic Polynomials

□  $Q[t]=[x(t) \ y(t) \ z(t)]$

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x$$

$$y(t) = a_y t^3 + b_y t^2 + c_y t + d_y$$

$$z(t) = a_z t^3 + b_z t^2 + c_z t + d_z$$

$$T = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix},$$

$$C = \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix},$$

$$Q(t) = \begin{bmatrix} x(t) & y(t) & z(t) \end{bmatrix} = T \cdot C$$

$$C = MG$$

□ M:

■ 4x4 basis matrix

□ G:

■ geometry vector

■ 4 elements column vector of geometric constraints.

# Three types of curves

---

- Hermit:  $G = [P_1 \ P_4 \ R_1 \ R_4]^T$ 
  - $R_1, R_4$ : the tangent vector constraint
- Bezier:  $G = [P_1 \ P_2 \ P_3 \ P_4]^T$ 
  - $P_2, P_3$  controls  $R_1, R_4$
- splines:  $G = [P_{i-3} \ P_{i-2} \ P_{i-1} \ P_i]^T$ 
  - uniform B-splines, nonuniform B-splines,  $\beta$ -splines





## 5.2 Bezier curves

---

- Definition
- Joining Bezier curve segments (5.2.1)
- Summary of Bezier curve properties (5.2.2)

## 5.2 Bezier curves

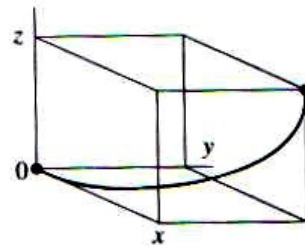
---

- Definition
  - Geometric constraints and blending functions
  - Polynomial form
  - Matrix form
  - Properties
  - Applications

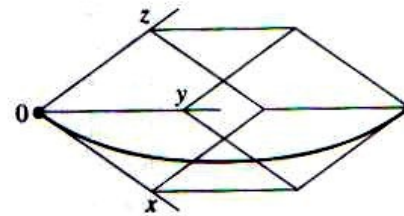
## 5.2 Bezier curves (Geometric constraints )

- Geometric vector  $G_z$  includes :
- 起點 :  $P_0$
  - 終點 :  $P_3$
  - 切線向量控制點 :  $P_1, P_2$

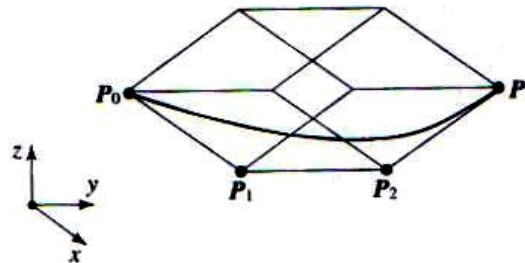
Figure 5.1  
Bézier's concept of curve  
representation.



Curve 'contained' by a cube



Drawing the cube into a  
parallelepiped changes the curve



Vertices used as control points

## 5.2 Bezier curves (blending functions)

- Blending function for each control point

$$B_0(u) = (1 - u)^3$$

$$B_1(u) = 3u(1 - u)^2$$

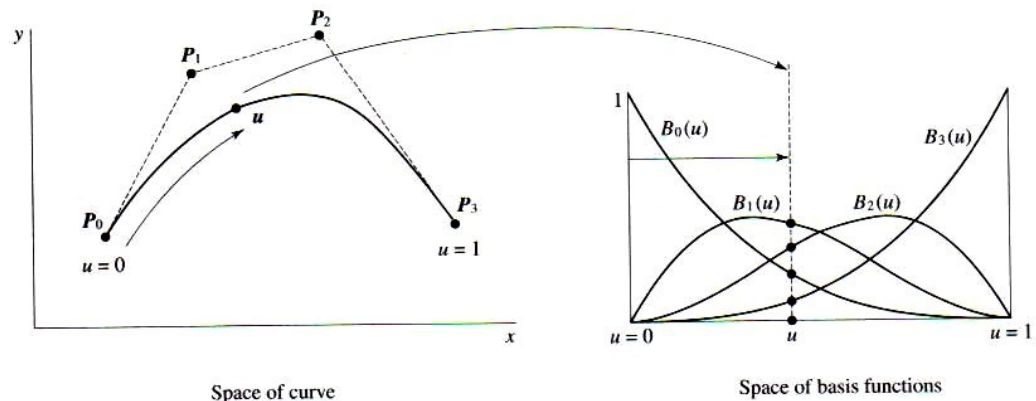
$$B_2(u) = 3u^2(1 - u)$$

$$B_3(u) = u^3$$

- The curve is given by:

$$Q(u) = \sum_{i=0}^3 P_i B_i(u)$$

**Figure 5.2**  
Moving along the curve by increasing  $u$  is equivalent to moving a vertical line through the basis functions. The intercepts of this line with the basis functions give the values of  $B$  for the equivalent point.



## 5.2 Bezier curves

---

### □ Polynomial form

Let  $P_i = (x_i, y_i, z_i)$ ,  $0 \leq i \leq 3$

$$Q(u) = \begin{bmatrix} x(u) & y(u) & z(u) \end{bmatrix} = \sum_{i=0}^3 P_i B_i(u), \quad 0 \leq u \leq 1$$

$$\Rightarrow x(u) = \sum_{i=0}^3 x_i B_i(u), \quad y(u) = \sum_{i=0}^3 y_i B_i(u), \quad z(u) = \sum_{i=0}^3 z_i B_i(u)$$

## 5.2 Bezier curves

---

### □ Matrix form

$$Q(u) = P_0(1 - u)^3 + P_1 3u(1 - u)^2 + P_2 3u^2(1 - u) + P_3 u^3$$

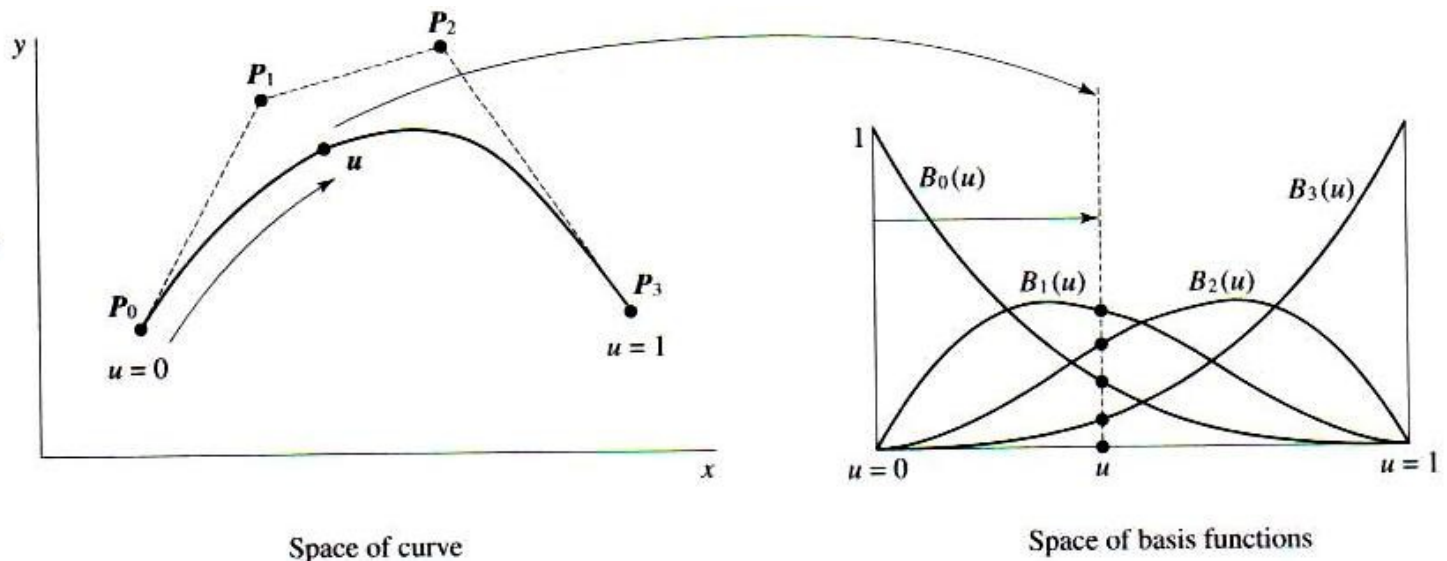
$$Q(u) = UB_z P$$

$$= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

## 5.2 Bezier curves (Properties)

- $Q(0) = P_0$
- $Q(1) = P_3$
- $B_0(u) + B_1(u) + B_2(u) + B_3(u) = 1$

**Figure 5.2**  
Moving along the curve by increasing  $u$  is equivalent to moving a vertical line through the basis functions. The intercepts of this line with the basis functions give the values of  $B$  for the equivalent point.



## 5.2 Bezier curves (Properties)

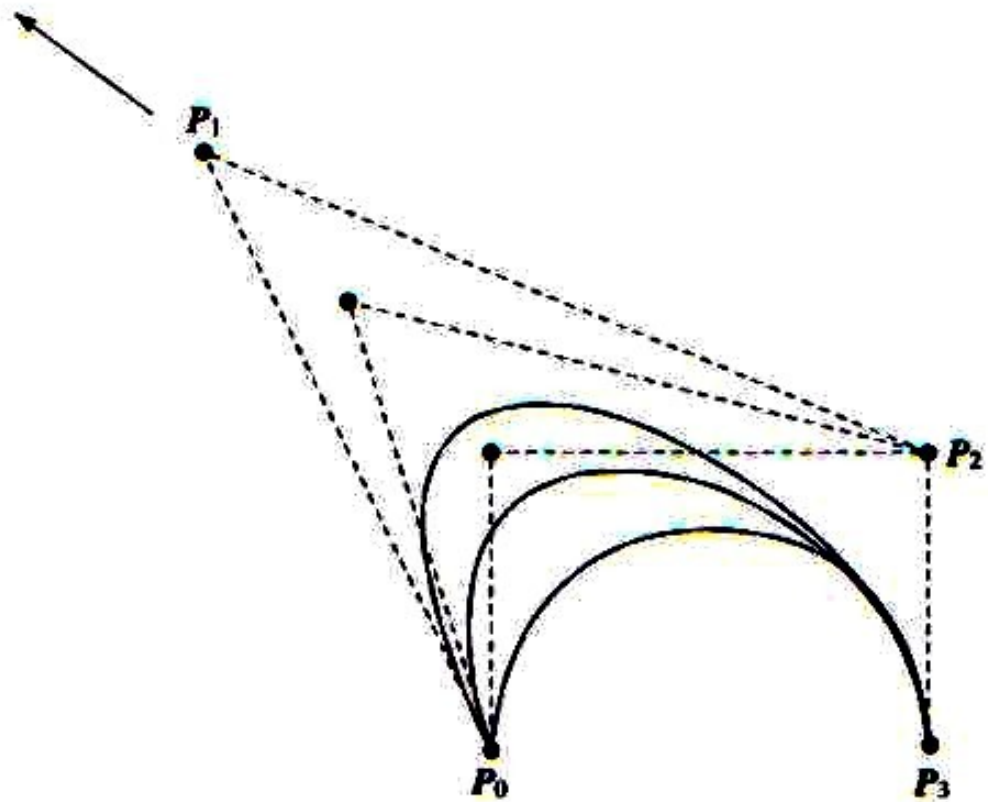
---

- When we move the inner control points  $P_1$  and  $P_2$  we change the **orientation** of the **tangent vectors** to the curves at the end points.
- The positions of  $P_1$  and  $P_2$  also control the **magnitude** of the tangent vectors and it can be shown that:
  - $Q_u(0)=3(P_1-P_0)$
  - $Q_u(1)=3(P_2-P_3)$
- Where  $Q_u$  is the tangent vector to the curve at the end point.



## 5.2 Bezier curves (Properties)

---

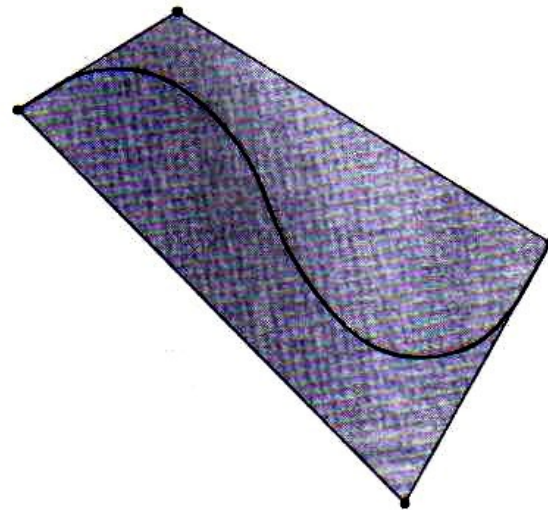


**Figure 5.3**  
Effects of moving control  
point  $P_1$ .

## 5.2 Bezier curves (Properties)

---

- Another way of putting it is to say that the curve mimics the shape of the control polygon.
- An important property from the point of view of the algorithm that deal with curves (and surfaces) is that a curve is always **enclosed** in the **convex hull** formed by control polygon.



**Figure 5.5**  
Convex hull property for  
cubic spline. The curve is  
contained in the shaded  
area formed from the  
control points.

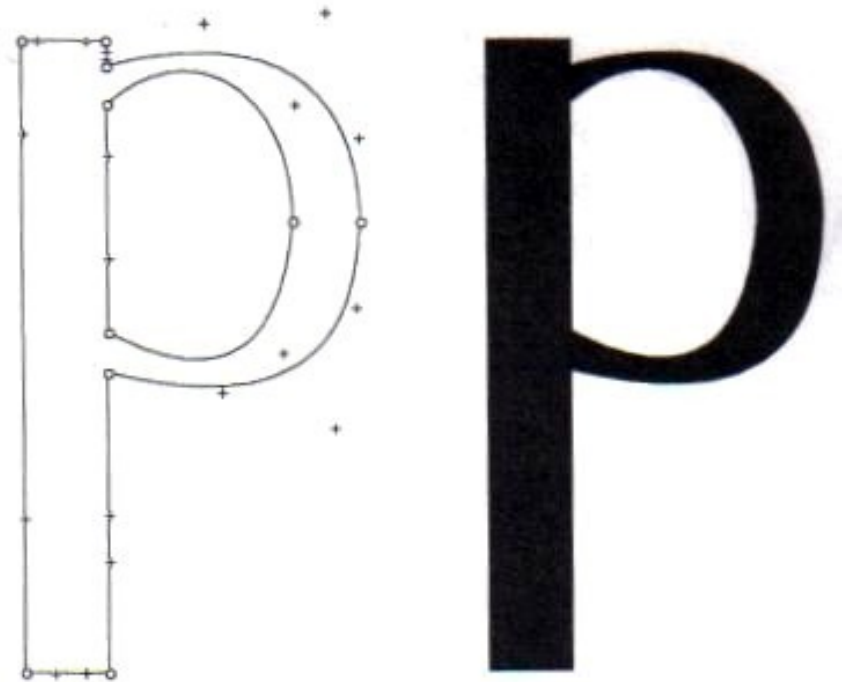
## 5.2 Bezier curves

---

### □ Applications

**Figure 5.4**

Using Bézier curves in font design. Each curve segment control points are symbolised by O + + O.



## 5.2.1 Joining Bezier curve segments

### □ Constraints at the joins

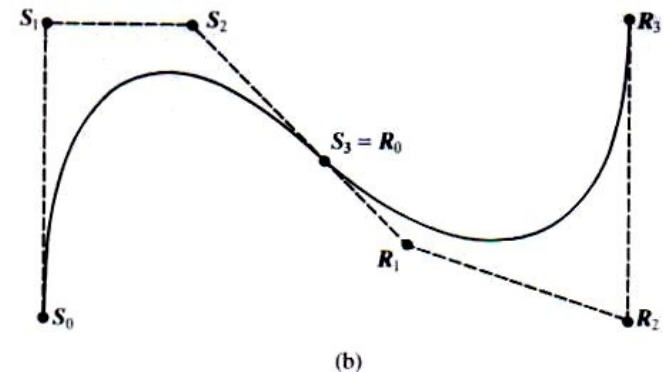
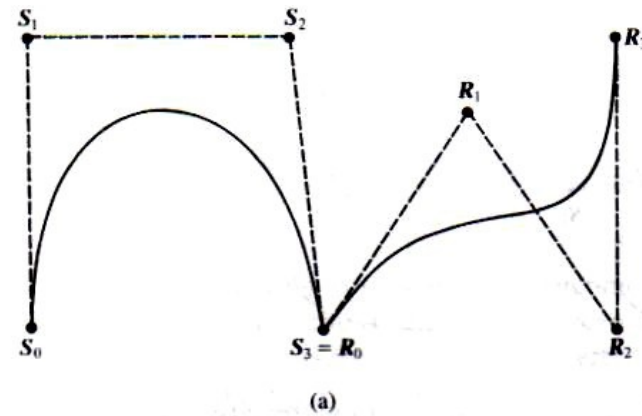
#### ■ positional continuity

□  $S_3 = R_0$

#### ■ first order continuity

□  $(S_3 - S_2) = k(R_1 - R_0)$

**Figure 5.6**  
Continuity between  
Bézier curve segments.  
(a) Positional continuity  
between Bézier points.  
(b) Tangential continuity  
between Bézier points.

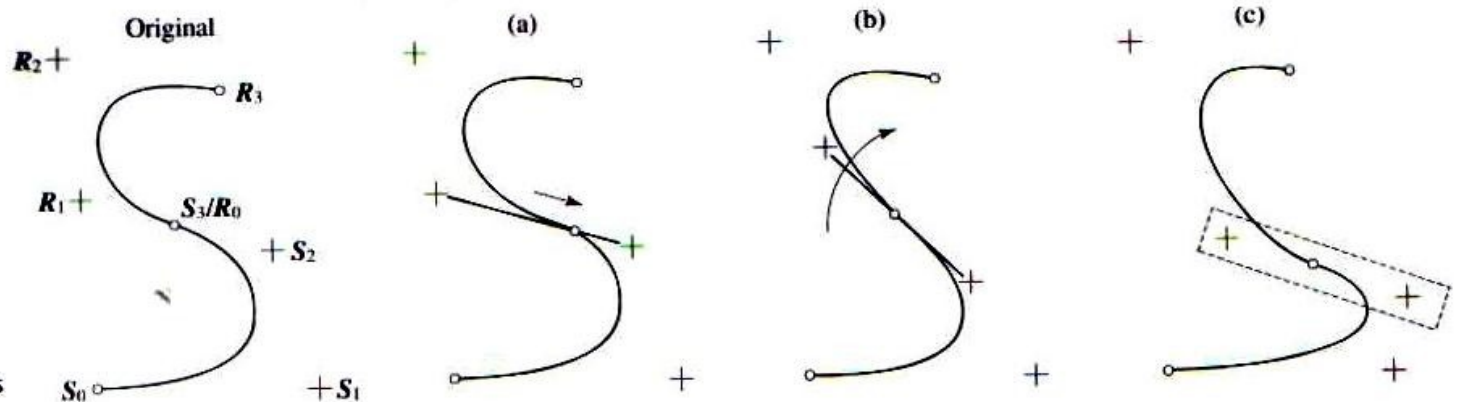


## 5.2.1 Joining Bézier curve segments

- Possible Shape Editing Protocols
  - Maintaining the orientation of the line  $R_1S_2$  and moving the join point up and down.
  - Maintaining the position of the join point and rotating the line  $R_1S_2$  about this point.
  - Moving all three control points as a locked unit.

**Figure 5.7**

Examples of possible shape editing protocols for a two-segment Bézier curve. (a) Maintain the orientation of  $R_1S_2$  and move any of the three control points  $R_1$ ,  $S_3/R_0$ ,  $S_2$  along this line. (b) Rotate the line  $R_1S_2$  about  $S_3/R_0$ . (c) Move the three control points  $R_1$ ,  $S_3/R_0$ ,  $S_2$  as a 'locked' unit.



## 5.2.2 Summary of Bezier curve properties

---

- A Bezier curve is a **polynomial**. The degree of the polynomial is always one less than the number of control points.
- The curve follows the shape of the **control point polygon**.
- The control points do not exert **local** control. Moving any control point affects all of the curve to a greater or lesser extent.
- The **first** and **last** control points are the end points of the curve segment.

## 5.2.2 Summary of Bezier curve properties

---

- The **tangent vectors** to the curve at the end points are coincident with the first and last edges of the **control point polygon**.
- Moving the **control points** alters the **magnitude** and **direction** of the **tangent vectors**.
- The curve does not **oscillate** about any **straight line** more than the **control point polygon** – this is known as the **variation diminishing property**.
- The curve is **transformed** by applying any **affine transformation** to its **control point** representation.



## 5.3 B-spline curves

---

5.3.1 Uniform B-splines

5.3.2 Non-uniform B-splines

5.3.3 Summary of B-spline curve properties



## 5.3 B-spline curves

---

- Composed of a series of  $m-2$  curve segments,  $Q_3, Q_4, \dots, Q_m$
- controlled by  $(m + 1)$  control points  $P_0, P_1, \dots, P_m$
- each curve segment is controlled by 4 control points over a given knot intervals:
  - $P_0, P_1, P_2, P_3 \rightarrow Q_3$  defined on  $[u_3, u_4]$
  - $P_1, P_2, P_3, P_4 \rightarrow Q_4$  defined on  $[u_4, u_5]$
  - ...
  - $P_{m-3}, P_{m-2}, P_{m-1}, P_m \rightarrow Q_m$  defined on  $[u_m, u_{m+1}]$
- each control point influences *four* curve segments. (*local control property*)

## 5.3 B-spline curves

---

- Entire set of curve segments as one B-spline curve in  $u$ :

$$Q(u) = \sum_{i=0}^m P_i B_i(u)$$

- $i=[0,m]$ , the **non-local** control point number
- $u=[3, m+1]$ , **global** parameter

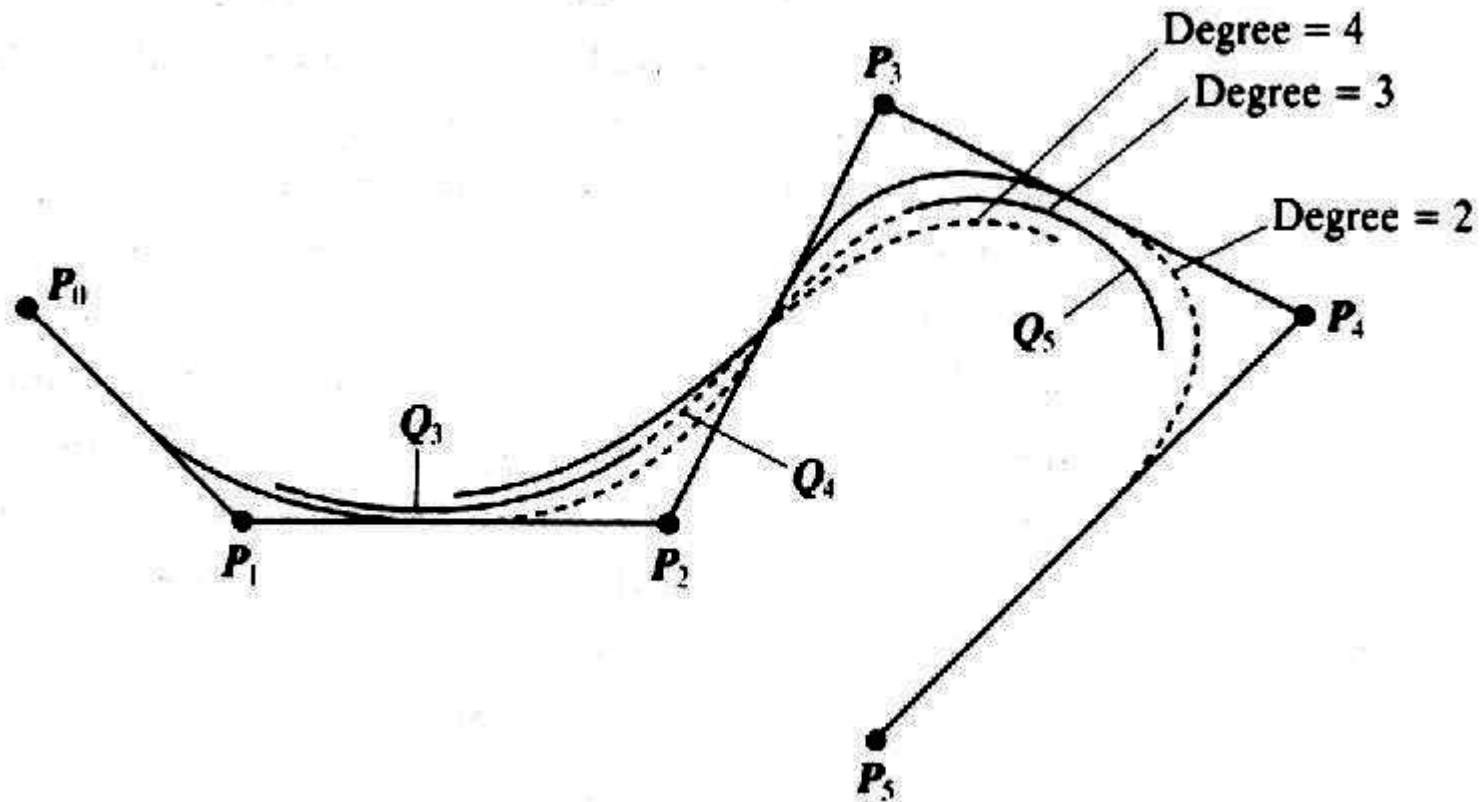
## 5.3.1 Uniform B-splines (Definition)

---

- The joint point on the value of  $u$  between segments is called the **knot value**.
- Uniform B-spline means that knots are spaced at **equal intervals** of the parameter  $u$ .
- Basis functions are defined over 4 successive knot intervals:
  - $B_0(u): [u_0, u_1, u_2, u_3, u_4]$
  - $B_1(u): [u_1, u_2, u_3, u_4, u_5]$
  - ...
  - $B_m(u): [u_{m-3}, u_{m-2}, u_{m-1}, u_m, u_{m+1}]$
- $(m-2)$  curve segments
  - Q3:  $P_0, P_1, P_2, P_3$   $u = [3, 4]$
  - Q4:  $P_1, P_2, P_3, P_4$   $u = [4, 5]$
  - .....
  - $Q_m: P_{m-3}, P_{m-2}, P_{m-1}, P_m$   $u = [m, m+1]$

## 5.3.1 Uniform B-splines

- An example of a B-spline curve



## 5.3.1 Uniform B-splines

### (Basis functions and curve computation)

---

- Basis function for  $Q_i(u)$ : ( $0 \leq u \leq 1$ )

$$B_i = \frac{1}{6}u^3$$

$$B_{i-1} = \frac{1}{6}(-3u^3 + 3u^2 + 3u + 1)$$

$$B_{i-2} = \frac{1}{6}(3u^3 - 6u^2 + 4)$$

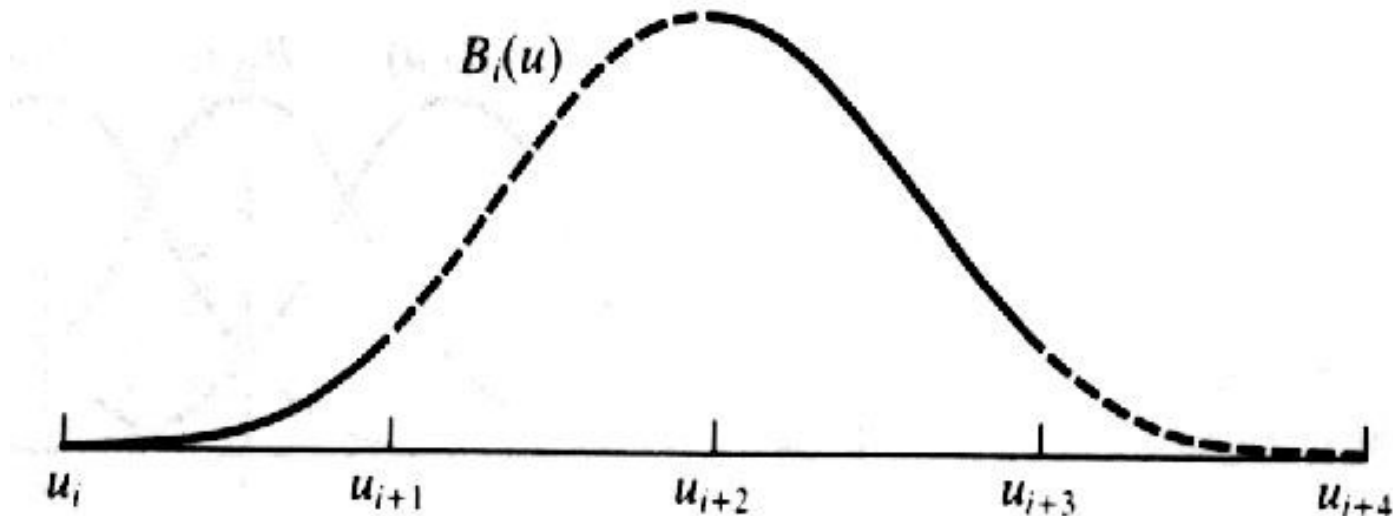
$$B_{i-3} = \frac{1}{6}(1 - u)^3$$

$$Q_i(u) = P_i B_i(u) + P_{i-1} B_{i-1}(u) + P_{i-2} B_{i-2}(u) + P_{i-3} B_{i-3}(u)$$

- It is important to note that this definition gives a single segment from each of the 4 B-spline basis functions over the range  $0 \leq u \leq 1$ .
- It does **not** define a single B-spline basis function which consists of four segments over the range  $0 \leq u \leq 4$ .

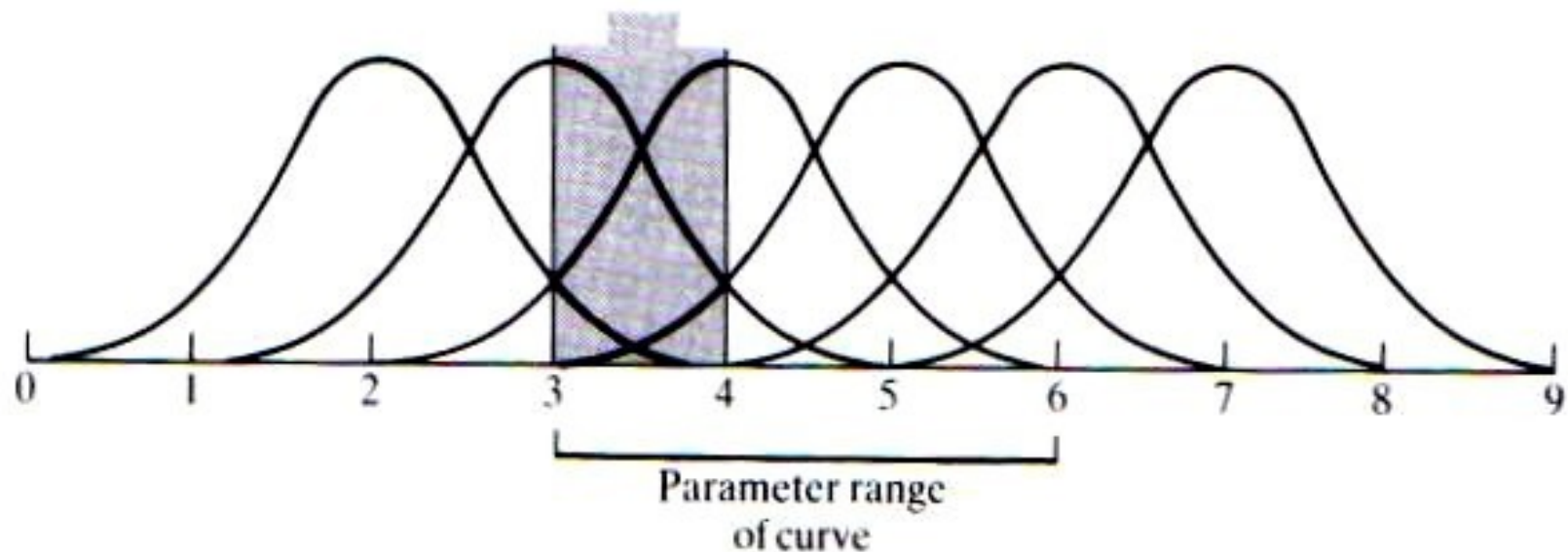
## 5.3.1 Uniform B-splines

- Uniform cubic B-spline blending function
  - Blending function is used to weighing specific control point of a curve. So there is a one-to-one correspondence between the control points and the blending functions.
  - The blending function  $B_i(u)$  is used to weighing  $P_i$ , the  $i$ th control point of the entire curve.



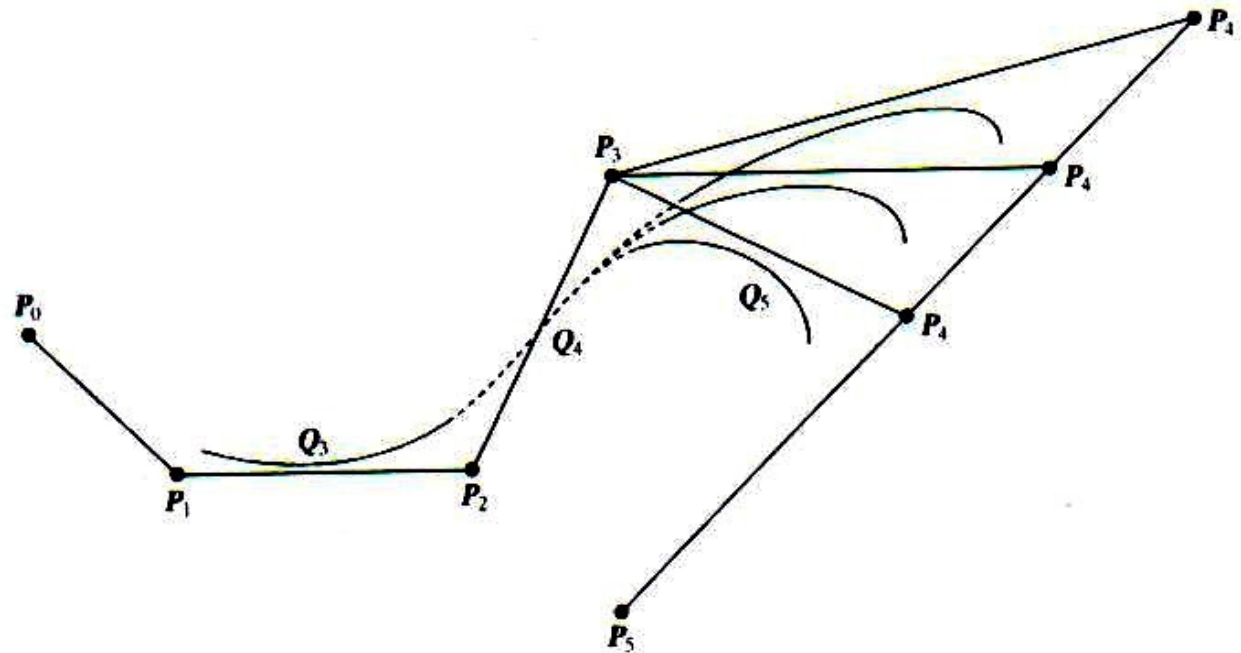
## 5.3.1 Uniform B-splines

- Parameter range and the basis functions for curve segment Q3 (u)
  - B3 的第一個區間, B2 的第二個區間
  - B1 的第三個區間, B0 的第四個區間



## 5.3.1 Uniform B-splines

- The local control property

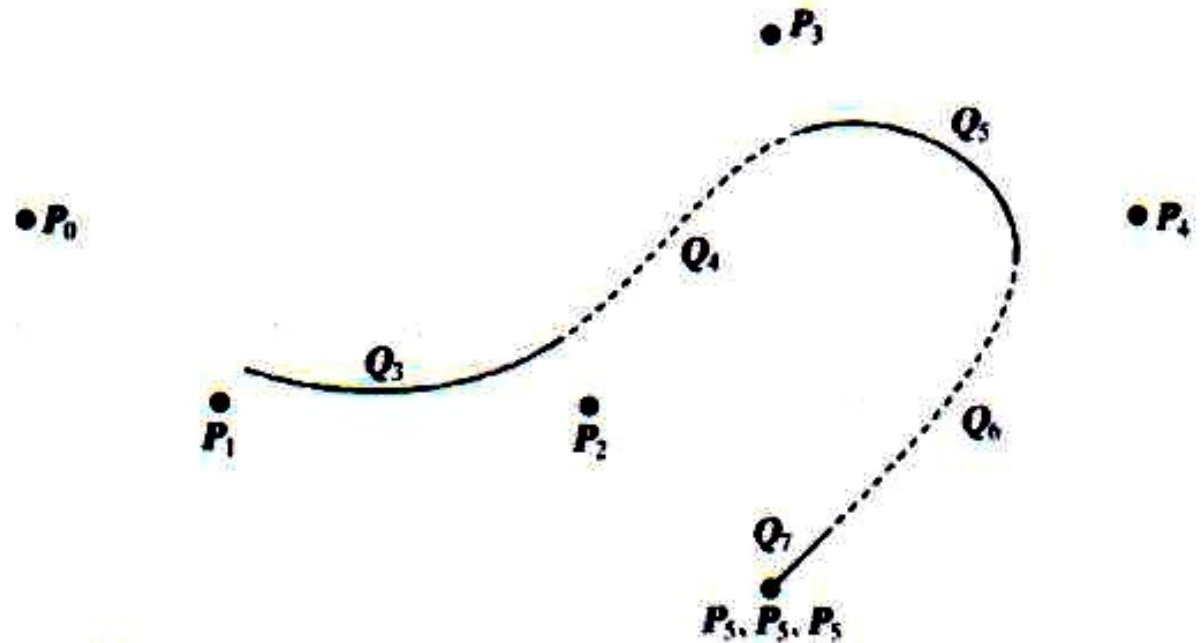


**Figure 5.9**  
Demonstrating the locality  
property of B-spline curves.  
Moving  $P_1$  changes  $Q_5$  and  
 $Q_4$  to a lesser extent.  $Q_3$  is  
unchanged.



## 5.3.1 Uniform B-splines

- The effect of multiple **end** control points

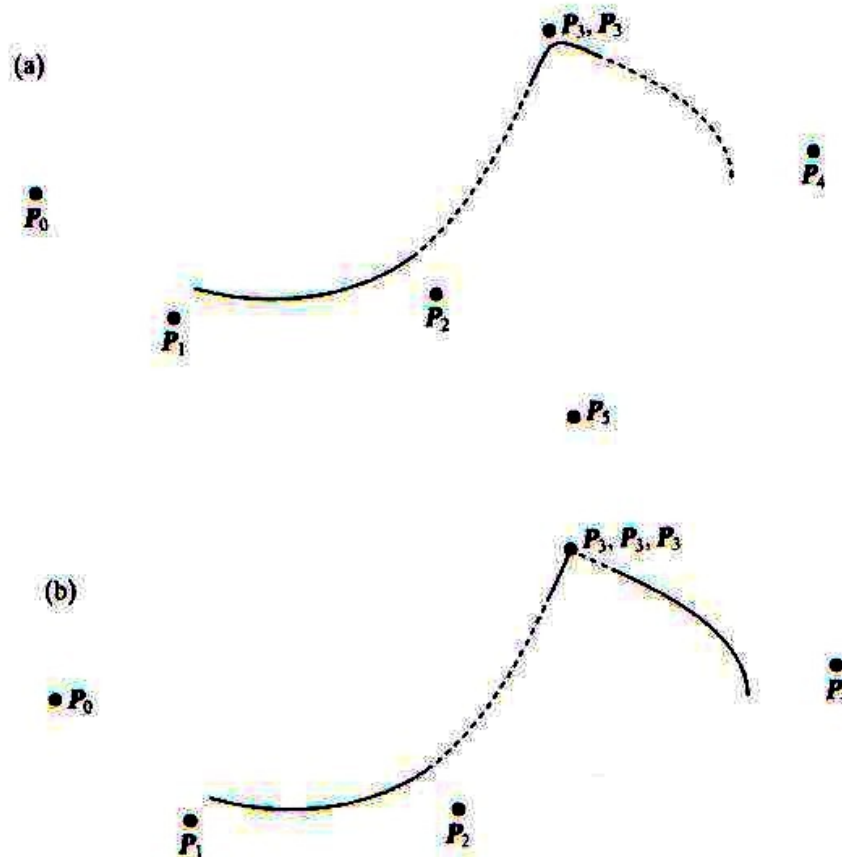


**Figure 5.13**  
Demonstrating the effect of multiple end control points.  $P_3$  is repeated three times, forcing the curve to interpolate it.

## 5.3.1 Uniform B-splines

- The effect of multiple **intermediate** control points

**Figure 5.14**  
Demonstrating the effect  
of multiple intermediate  
control points. (a)  $P_3$  is  
duplicated. (b)  $P_3$  is  
triplicated.



## 5.3.2 Non-uniform B-splines

---

- The **parametric interval** are not necessarily equal.
- The parametric intervals are defined by the **knot vector**.
- Knot vector  $[0,0,0,0,1,2,\dots,n-1,n,n,n,n]$  is often used to offer interpolation at the end points.
- As we seen, successive knot values can be equal and the number of identical values is called the **multiplicity** of the knot.
- 控制 knot 間距是用以調節控制點 blending function 的方法。因此，可以使不同控制點有不同的 blending function. (uniform 的曲線中每一個控制點的 blending function 均相同)

## 5.3.2 Non-uniform B-splines

---

□ Basis function  $B_0(u)$

$$B_0(u) = \begin{cases} b_{-0}(u) = \frac{1}{6} u^3 & 0 \leq u \leq 1 \\ b_{-1}(u) = -\frac{1}{6} (3u^3 - 12u^2 + 12u - 4) & 1 \leq u \leq 2 \\ b_{-2}(u) = \frac{1}{6} (3u^3 - 24u^2 + 60u - 44) & 2 \leq u \leq 3 \\ b_{-3}(u) = -\frac{1}{6} (u^3 - 12u^2 + 48u - 64) & 3 \leq u \leq 4 \end{cases}$$

## 5.3.2 Non-uniform B-splines

- 因為 **knot 間距** 會造成 **blending function** 的變化 . 故需要一個由 **knot 間距** 計算 **blending function** 的方法 . (Cox-De Boor Algorithm)
- Cox-De Boor Algorithm

Degree 1  $B_{i,1}(u) = \begin{cases} 1, & u_i \leq u \leq u_{i+1} \\ 0, & \text{otherwise} \end{cases}$

**order**

Degree 2  $B_{i,2}(u) = \frac{u - u_i}{u_{i+1} - u_i} B_{i,1}(u) + \frac{u_{i+2} - u}{u_{i+2} - u_{i+1}} B_{i+1,1}(u)$

Degree 3  $B_{i,3}(u) = \frac{u - u_i}{u_{i+2} - u_i} B_{i,2}(u) + \frac{u_{i+3} - u}{u_{i+3} - u_{i+1}} B_{i+1,2}(u)$

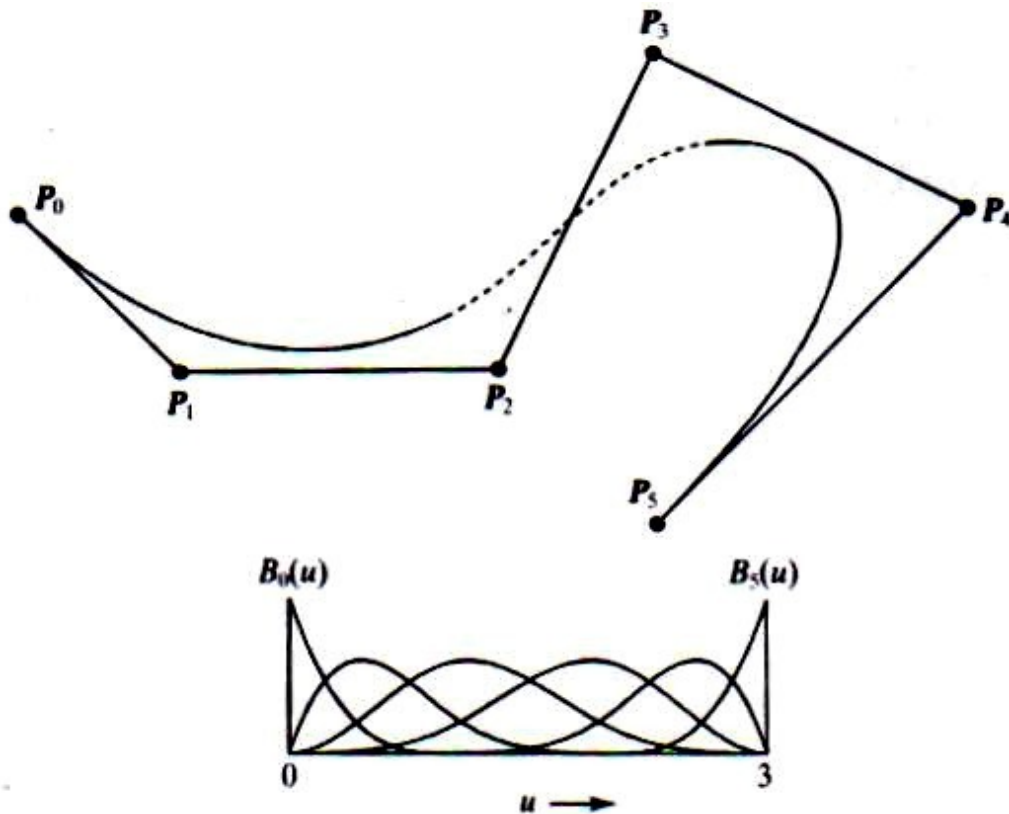
Degree 4  $B_{i,4}(u) = \frac{u - u_i}{u_{i+3} - u_i} B_{i,3}(u) + \frac{u_{i+4} - u}{u_{i+4} - u_{i+1}} B_{i+1,3}(u)$

## 5.3.2 Non-uniform B-splines

- Non-uniform B-splines using a knot vector  $[0,0,0,0,1,2,3,3,3,3]$

**Figure 5.15**

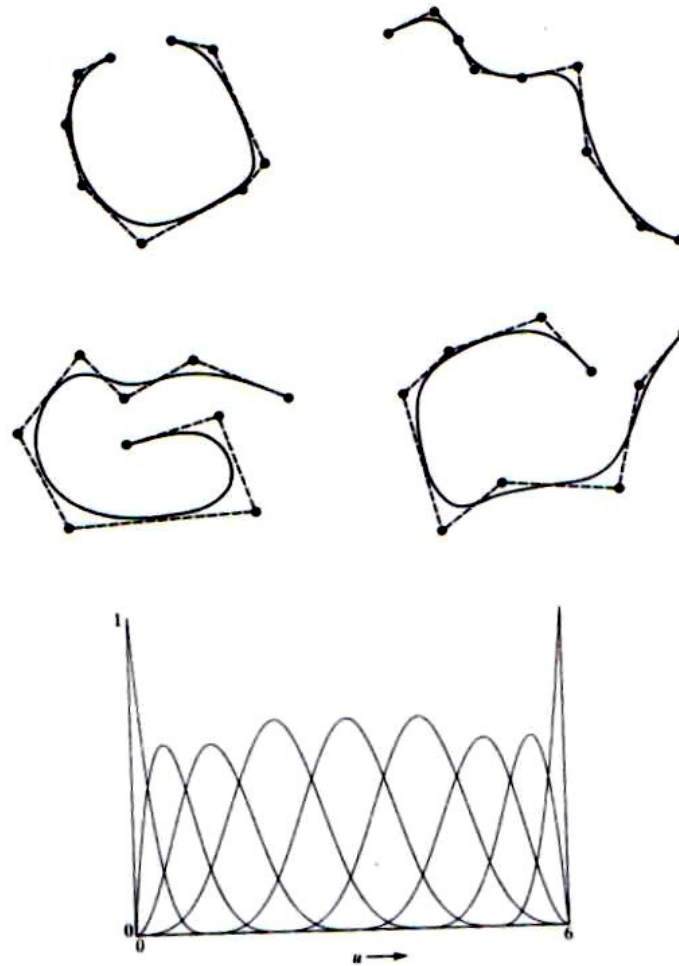
A non-uniform B-spline that interpolates the end points by using a knot vector  $[0,0,0,0,1,2,3,3,3,3]$ .



## 5.3.2 Non-uniform B-splines

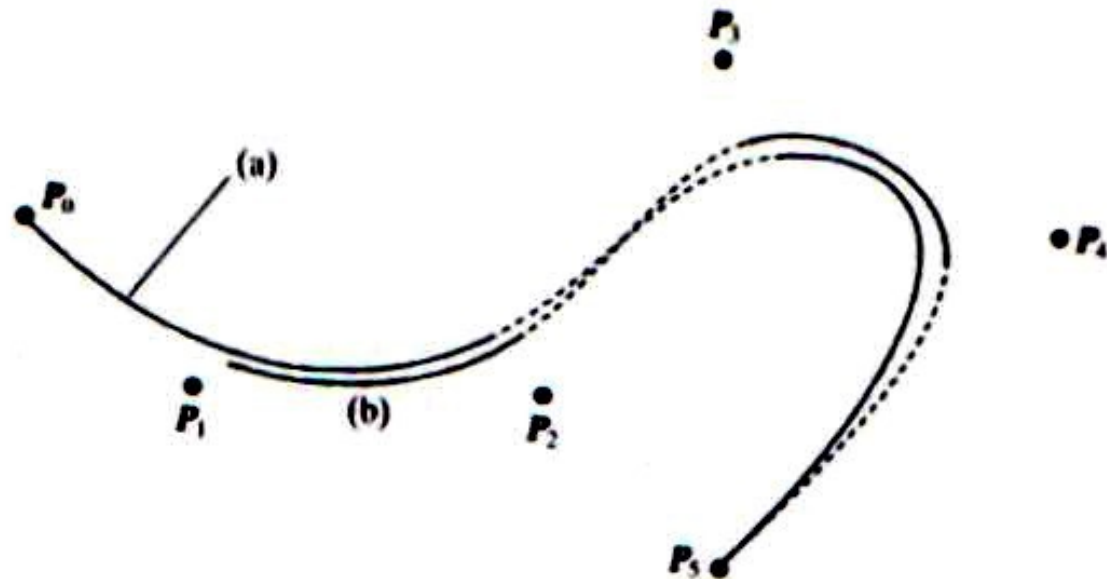
- Flexibility of the B-splines (knot vector is  $[0,0,0,0,1,2,3,4,5,6,6,6,6]$ )

**Figure 5.16**  
Showing the flexibility  
of B-spline curves.  
The knot vector is  
 $[0,0,0,0,1,2,3,4,5,6,6,6,6]$ .



## 5.3.2 Non-uniform B-splines

- Multiple knots vs. Multiple control points

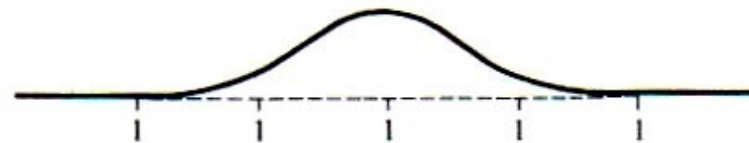


**Figure 5.17**  
Comparing multiple knots  
with multiple control points.  
(a) The curve is generated  
by a knot vector with  
multiplicity 4 on the start  
and end values. (b)  $P_5$  is  
repeated three times.

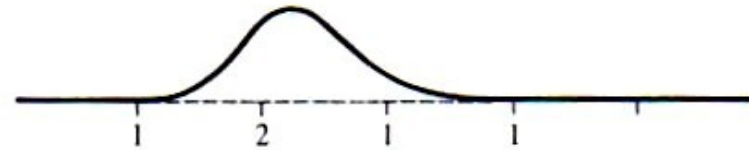


## 5.3.2 Non-uniform B-splines

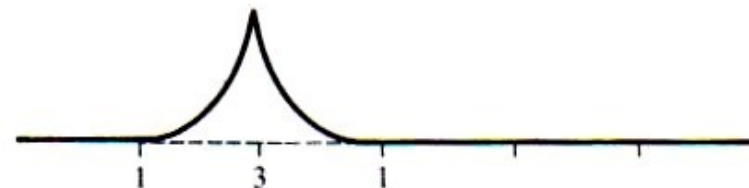
### □ Effect of knot multiplicity



(a)



(b)



(c)



(d)

**Figure 5.18**

The effect of knot multiplicity on a single cubic B-spline basis function.

(a) All knot multiplicities are unity:  $[0, 1, 2, 3, 4]$ .

(b) Second knot has multiplicity 2:  $[0, 1, 1, 2, 3]$ .

(c) Second knot has multiplicity 3:  $[0, 1, 1, 1, 2]$ .

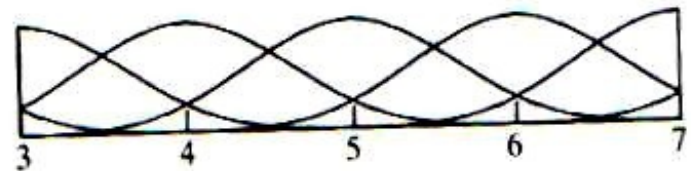
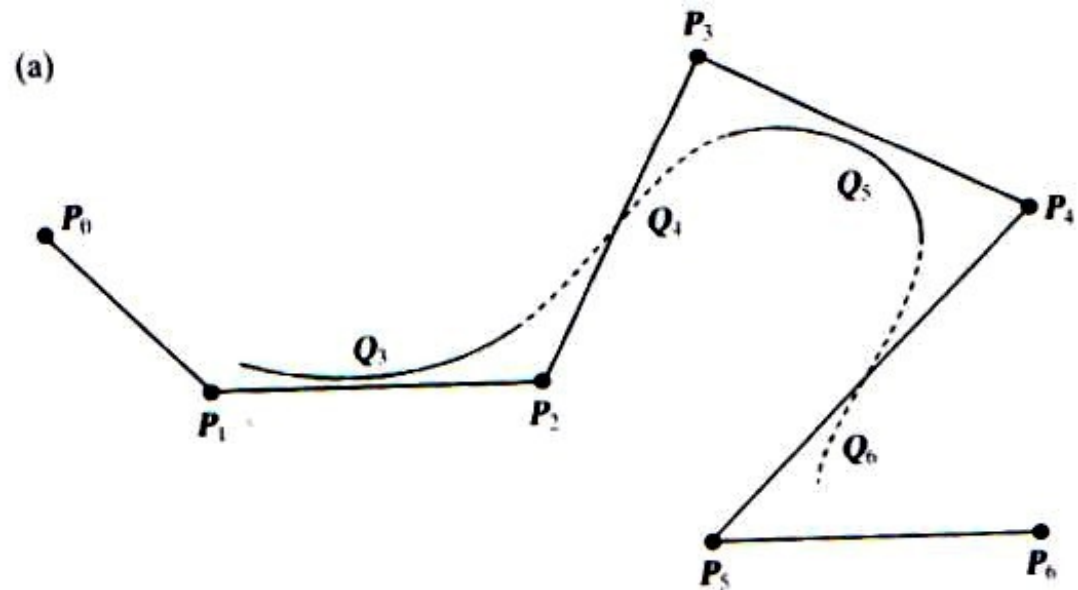
(d) Second knot has multiplicity 4:  $[0, 1, 1, 1, 1]$ .

## 5.3.2 Non-uniform B-splines

- The effect of interior multiplicity (a)

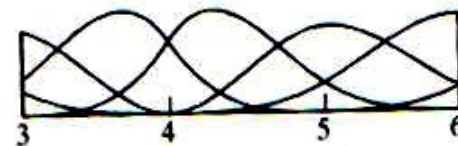
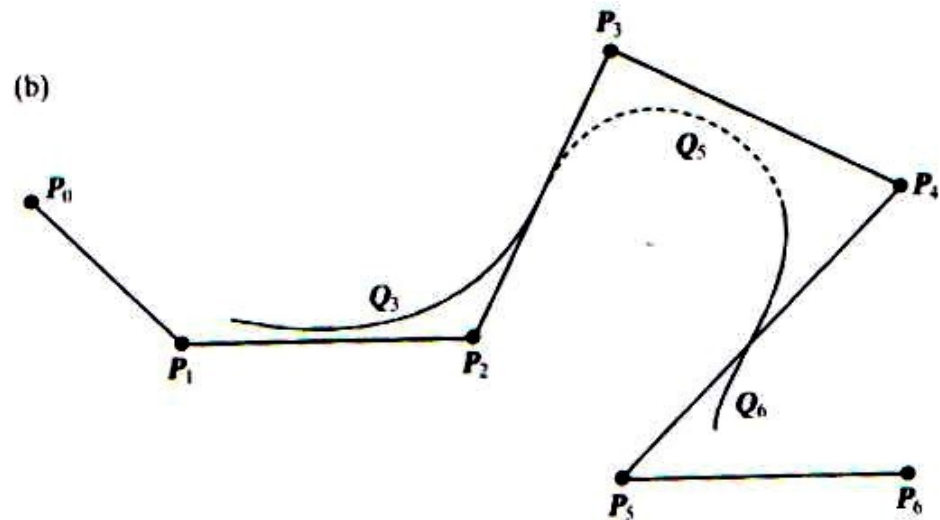
**Figure 5.19**

The effect of interior knot multiplicity on a B-spline curve. (a) A four-segment B-spline curve. The knot vector is  $[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$ . All B-splines are translates of each other.



## 5.3.2 Non-uniform B-splines

- The effect of interior multiplicity (b) , Knot vector =  $[0, 1, 2, 3, 4, 4, 5, 6, 7, 8, 9]$



↑  
Double  
knot

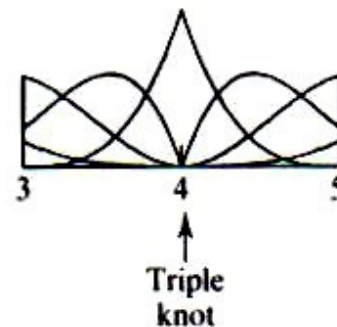
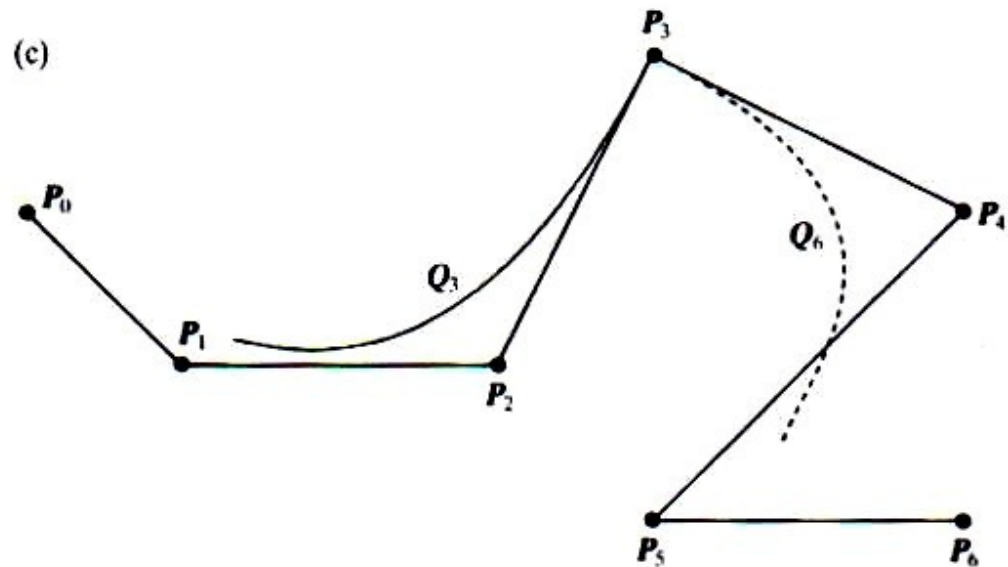
(b) Knot vector is  
 $[0, 1, 2, 3, 4, 4, 5, 6, 7, 8, 9]$ .  $Q_4$   
shrinks to zero.

## 5.3.2 Non-uniform B-splines

- The effect of interior multiplicity (c), Knot vector =  $[0, 1, 2, 3, 4, 4, 4, 5, 6, 7, 8]$

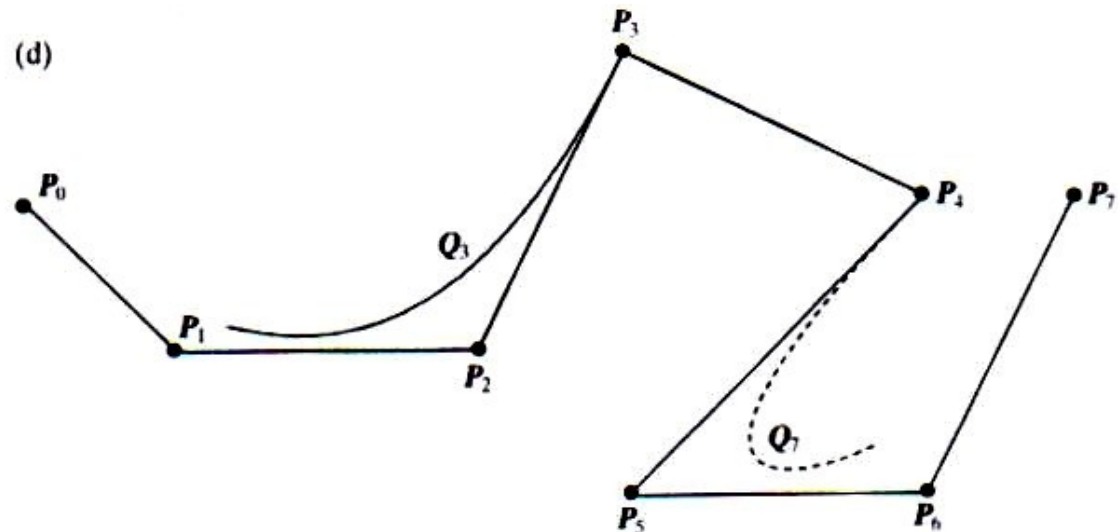
**Figure 5.19 continued**

(c) Knot vector is  $[0, 1, 2, 3, 4, 4, 4, 5, 6, 7, 8]$ .  
 $Q_4$  and  $Q_5$  shrink to zero.  
Continuity between  $Q_3$  and  $Q_6$  is positional.

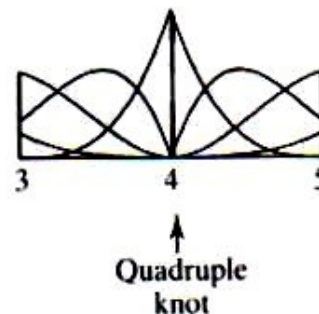


## 5.3.2 Non-uniform B-splines

- The effect of interior multiplicity (d), Knot vector =  $[0, 1, 2, 3, 4, 4, 4, 4, 5, 6, 7, 8]$



(d) Knot vector is  $[0, 1, 2, 3, 4, 4, 4, 4, 5, 6, 7, 8]$ . The curve reduces to a single segment  $Q_3$ . Another control point has been added to show that the curve now 'breaks' between  $P_3$  and  $P_4$ .



## 5.3.3 Summary of B-spline curve properties

---

- The curve follows the shape of the control point polygon and is constrained to lie in the convex hull of the control points
- The curve exhibits the variation diminishing property.
- The curve is transformed by applying any affine transformation to its control point representation.
- A B-spline curve exhibits local control – a control point connected to four segments (in the case of a cubic) and moving a control point can only influence these segments.



## 5.4 Rational curves

---

- 5.4.1 Rational Bezier curves
- 5.4.2 NURBS

## 5.4 Rational curves

---

- A rational curve is a curve defined in **four-dimensional** space – known as **projective space** – which is then projected into **three-dimensional**.



## 5.4.1 Rational Bezier curves

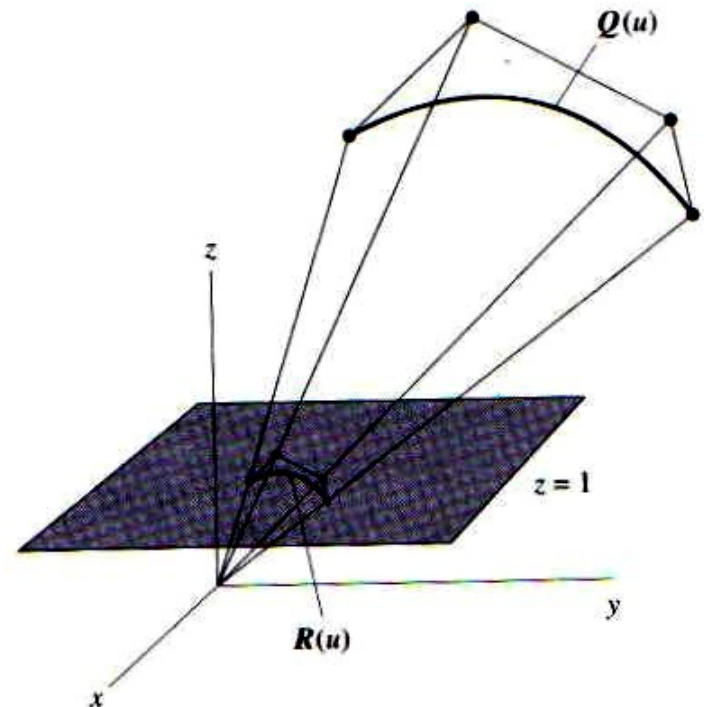
- Consider the projection of a **3D** Bezier curve in to **2D** space: ( $z=1$ )
  - We do this by dividing by  $z(u)$  to define a 2D curve  $R(u)$ :

$$R(u) = \left( \frac{x(u)}{z(u)}, \frac{y(u)}{z(u)} \right)$$

- The 3D curve:

$$Q(u) = \sum_{i=0}^3 P_i B_i(u)$$

$$P_i = (x_i, y_i, z_i)$$



## 5.4.1 Rational Bezier curves (cont.)

---

- Now a special notation is used for writing the 3D control points of a rational curve in 2D space:

$$P_i^w = (w_i x_i, w_i y_i, w_i)$$

- We write our 3D curve as:

$$Q(u) = \sum_{i=0}^3 \begin{bmatrix} w_i x_i \\ w_i y_i \\ w_i \end{bmatrix} B_i(u)$$

- which is projected into 2D space as:

$$R(u) = \left( \frac{\sum w_i x_i B_i(u)}{\sum w_i B_i(u)}, \frac{\sum w_i y_i B_i(u)}{\sum w_i B_i(u)} \right)$$

## 5.4.1 Rational Bezier curves

---

- We project a 4D curve into 3D space
  - Each control point is now:

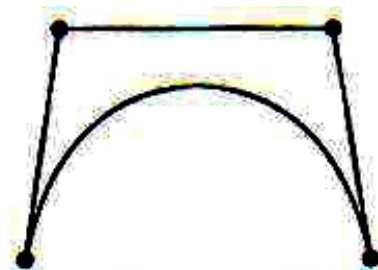
$$P_i^w = (w_i x_i, w_i y_i, w_i z_i, w_i)$$

- And we have:

$$R(u) = \frac{\sum w_i P_i B_i(u)}{\sum w_i B_i(u)}$$

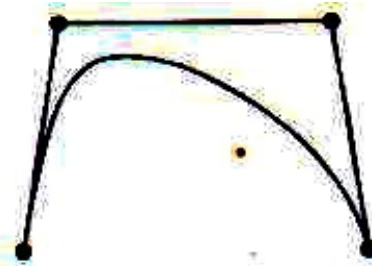
## 5.4.1 Rational Bézier curves

- The effect of changing weight  $w_i$



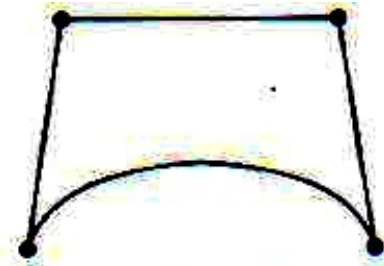
$$w_0 = w_1 = w_2 = w_3 = 1$$

(a)



$$w_0 = 1 \quad w_1 = 3$$

(b)



$$w_0 = 1 \quad w_1 = w_2 = 0.2 \quad w_3 = 1$$

(c)

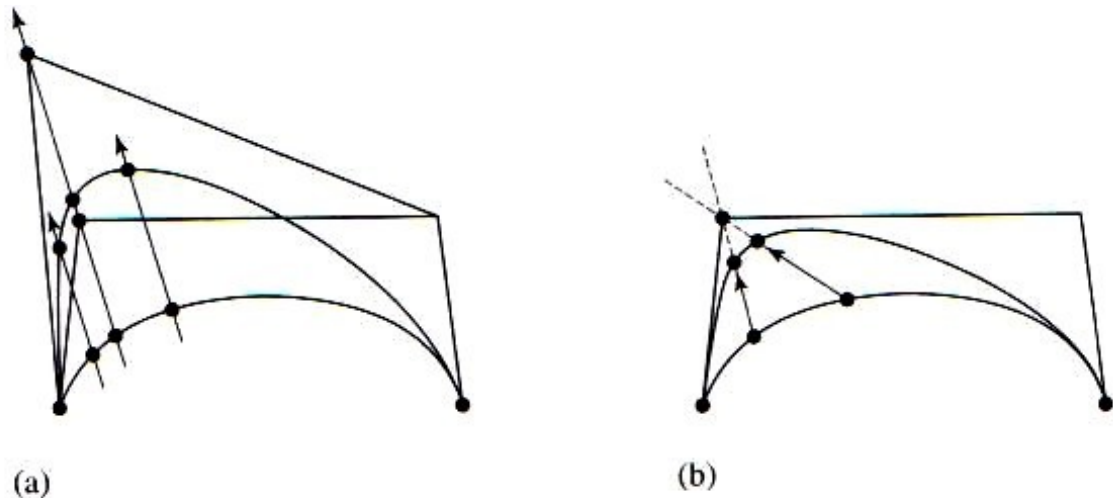
**Figure 5.21**  
Varying the weights of a  
rational Bézier curve.

## 5.4.1 Rational Bezier curves

- The different effect of control point movement and weight adjustment

**Figure 5.22**

Rational Bézier curves: the different effect of control point movement and weight adjustment. (a) Movement of a control point causes any point on the curve to move in a parallel direction. (b) Adjustment of a weight causes any point on the curve to move along a line drawn from the curve point to the control point.



## 5.4.2 NURBS

---

- NURBS stands for **Non-Uniform** Rational B-Splines .
- admits the following possibilities:
  - Interactive placement and movement of **control points**.
  - Interactive placement and movement of **knots**.
  - Interactive control of **control-point weights**.

## 5.4.2 NURBS (Formulation of NURBS)

---

- Let  $P_i = (x_i, y_i, z_i)$  be a 3D point projected from a 4D point

$$P_i^w = (w_i x_i, w_i y_i, w_i z_i, w_i)$$

- The 3D curve of NURBS

$$R(u) = H \left[ \sum_{i=0}^n P_i^w B_{i,k}(u) \right]$$

$$= \frac{\sum_{i=0}^n P_i w_i B_{i,k}(u)}{\sum_{i=0}^n w_i B_{i,k}(u)}$$

- If  $w_i = 1$ , for all  $i$ , then  $R_{i,k}(u) = B_{i,k}(u)$
- The curve is pulled towards a control point  $P_i$  if  $w_i$  increases.

## 5.5 From curves to surfaces

- Cubic Bezier patch surface formulation

$$Q(u, v) = \sum_{i=0}^3 \sum_{j=0}^3 P_{ij} B_i(u) B_j(v)$$

- The **matrix** specification of Equation

$$\mathbf{Q}(u, v) = [u^3 \ u^2 \ u \ 1] [B_z \ \mathbf{P} \ B_z^T] \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

where:

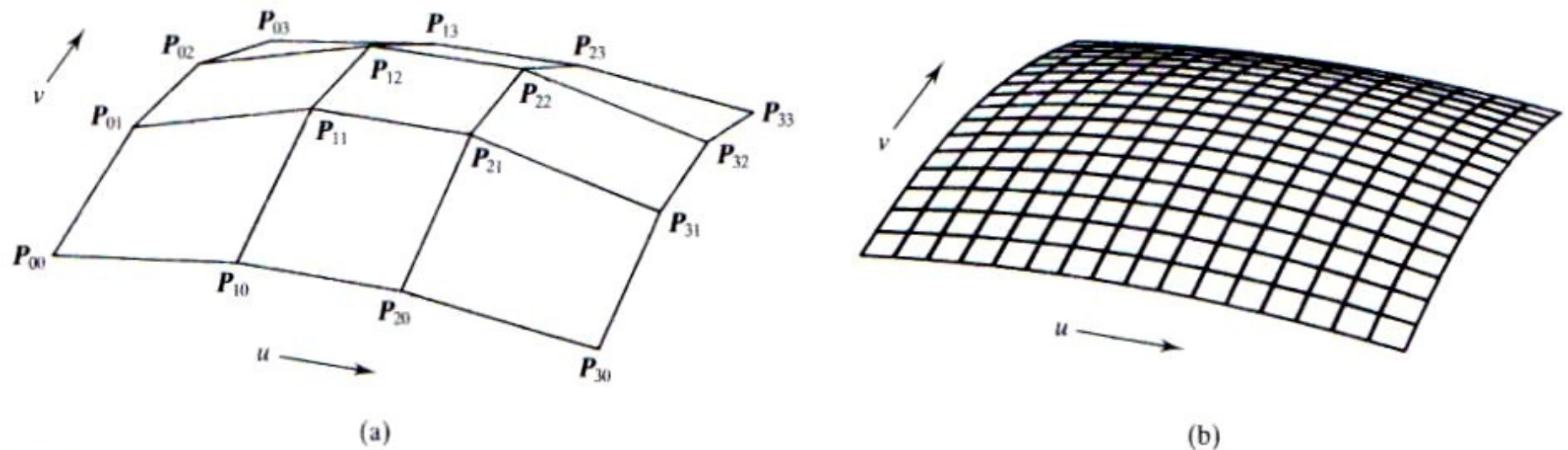
$$B_z = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{P}_{02} & \mathbf{P}_{03} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{13} \\ \mathbf{P}_{20} & \mathbf{P}_{21} & \mathbf{P}_{22} & \mathbf{P}_{23} \\ \mathbf{P}_{30} & \mathbf{P}_{31} & \mathbf{P}_{32} & \mathbf{P}_{33} \end{bmatrix}$$



## 5.5 From curves to surfaces

- Cubic Bezier patch surface



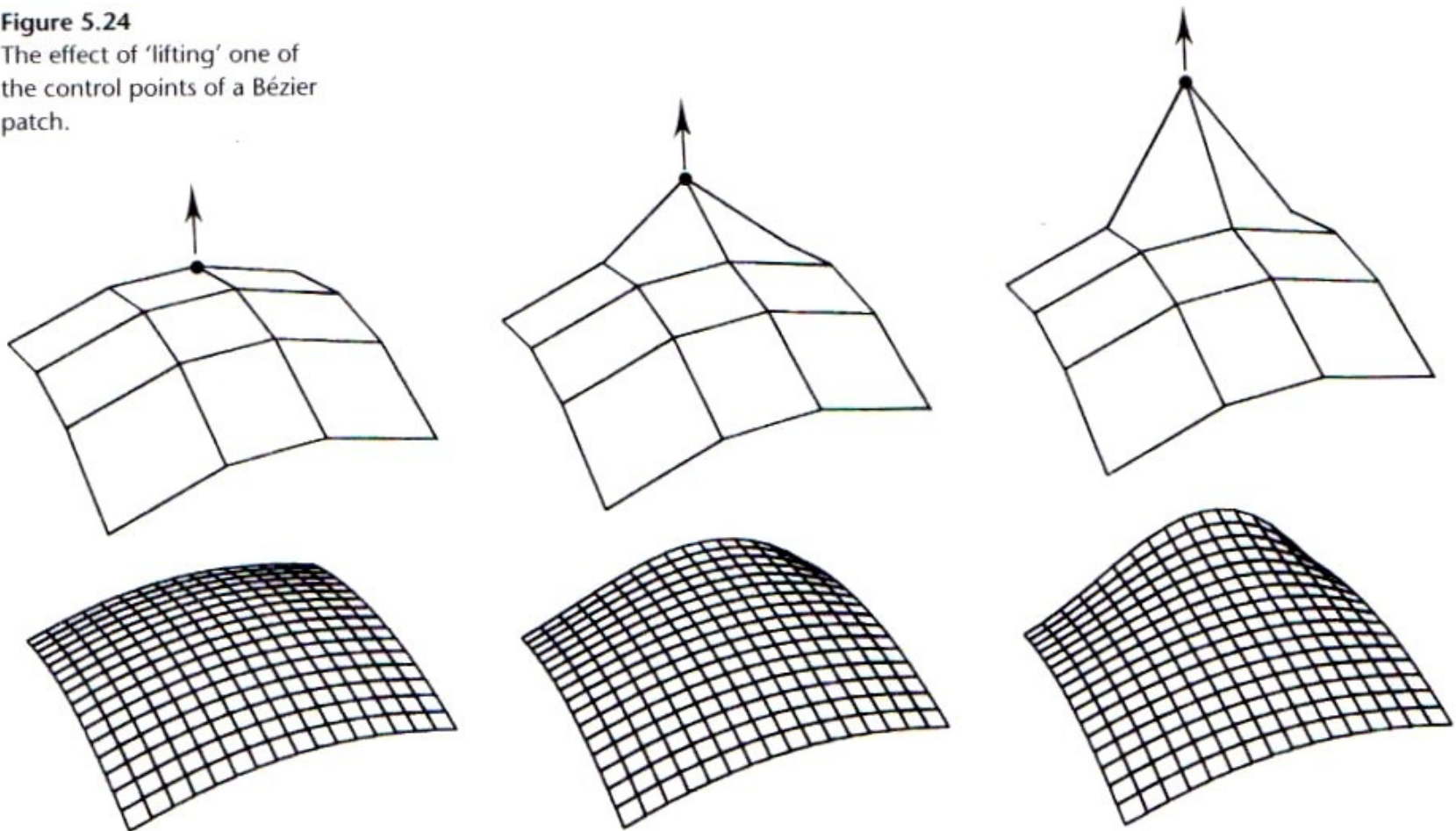
**Figure 5.23**  
(a) A control polyhedron  
and (b) the resulting bi-  
cubic Bézier patch.

## 5.5 From curves to surfaces

- The effect of lifting one of the control points

**Figure 5.24**

The effect of 'lifting' one of the control points of a Bézier patch.



## 5.5 From curves to surfaces

---

□ Vectors at the corner of a patch.

- The tangent vector at  $Q(0,0)$  in the **u** parameter direction.

$$Q_u(0,0) = 3(P_{10} - P_{00})$$

- The tangent vector at  $Q(0,0)$  in the **v** parameter direction.

$$Q_v(0,0) = 3(P_{01} - P_{00})$$

- The cross-derivatives at each end point, sometimes called **twist vectors**, specify the **rate of change of the tangent vectors** with respect to **u** and **v**.

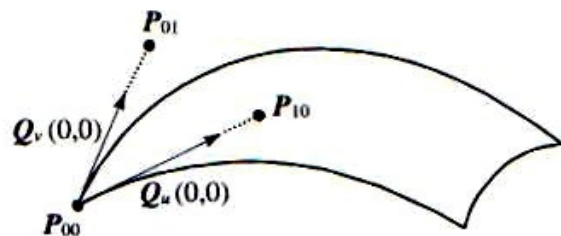
$$Q_{uv}(0,0) = 9(P_{00} - P_{01} - P_{10} + P_{11})$$

# 5.5 From curves to surfaces

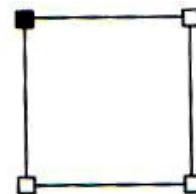
## □ Vectors at $P_{00}$

**Figure 5.25**

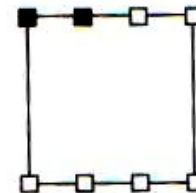
Vectors at  $P_{00}$ . (a) Tangent vectors at  $P_{00}$ . (b) Elements of control point matrix involved in vectors at  $P_{00}$ .



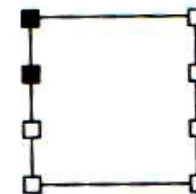
(a)



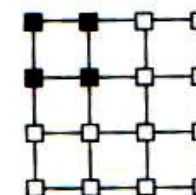
Corner points



Tangent vectors in  $v$  :  
for example,  $Q_v(0,0) = 3(P_{01} - P_{00})$



Tangent vectors in  $u$  :  
for example,  $Q_u(0,0) = 3(P_{10} - P_{00})$



Twist vectors:  
for example,  $Q_{uv}(0,0) = 9(P_{00} - P_{01} - P_{10} - P_{11})$

(b)

## 5.5 From curves to surfaces (zero twist surface)

---

- If we set  $Q_{uv}(i,j) = 0$  then we have a so-called **zero twist surface** or a surface with four zero twist vectors.
- 若已知在邊緣的 12 個點，可求出中間四點：
  - $0 = 9(P_{00} - P_{01} - P_{10} + P_{11}) \Rightarrow P_{11} = P_{01} + P_{10} - P_{00}$
  - $0 = 9(P_{03} - P_{02} - P_{13} + P_{12}) \Rightarrow P_{12} = P_{02} + P_{13} - P_{03}$
  - $0 = 9(P_{30} - P_{31} - P_{20} + P_{21}) \Rightarrow P_{21} = P_{31} + P_{20} - P_{30}$
  - $0 = 9(P_{33} - P_{32} - P_{23} + P_{22}) \Rightarrow P_{22} = P_{32} + P_{23} - P_{33}$

## 5.5 From curves to surfaces (Patch specification)

---

- Each patch is specified by:
  - 4 end points
  - 8 tangent vectors
  - 4 twist vectors

## 5.5 From curves to surfaces (calculate surface normal)

---

### □ Subdivision:

- Subdivide patch until the products of the sub-division are approximately **planar**. The patches can then be treated as **planar polygons**.
- Apply polygon shading algorithm to each polygon.
- vertex normal can be calculated from:

$$a = (P_{01} - P_{00})$$

$$b = (P_{10} - P_{00})$$

$$N = a \times b$$

## 5.5 From curves to surfaces (calculate surface normal)

---

- At each point of the patch:

$$\mathbf{N} = \frac{\partial \mathbf{Q}}{\partial u} \times \frac{\partial \mathbf{Q}}{\partial v}$$

- The surface normal at any point on the patch surface:

$$\mathbf{a} = \mathbf{Q}_u(u, v) = [3u^2 \ 2u \ 1 \ 0] \mathbf{B}_z \mathbf{P} \mathbf{B}_z^T \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

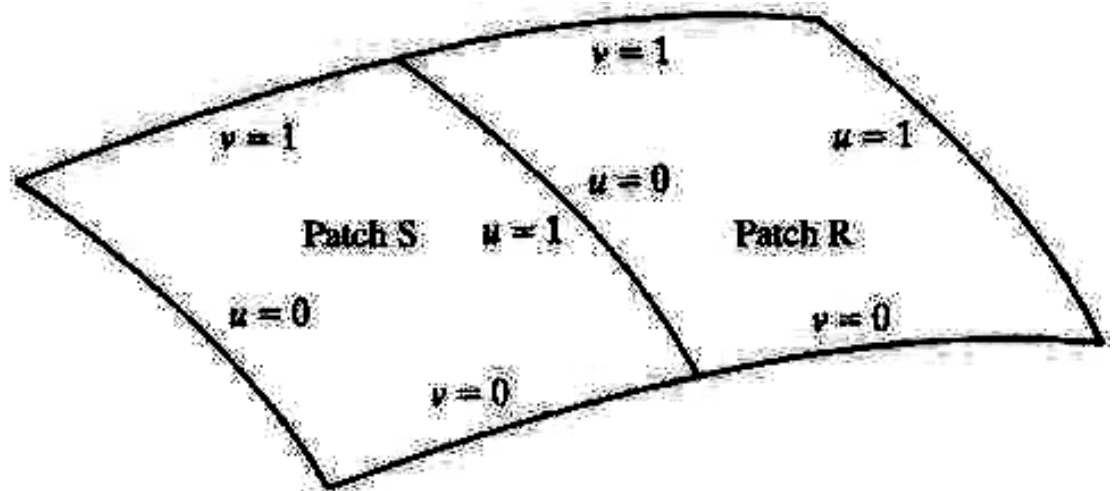
$$\mathbf{b} = \mathbf{Q}_v(u, v) = [u^3 \ u^2 \ u \ 1] \mathbf{B}_z \mathbf{P} \mathbf{B}_z^T \begin{bmatrix} 3v^2 \\ 2v \\ 1 \\ 0 \end{bmatrix}$$

$$\mathbf{N} = \mathbf{a} \times \mathbf{b}$$



## 5.5.1 Continuity and Bezier patches

- Join two patches



**Figure 5.26**  
Joining two patches.

# 5.5.1 Continuity and Bezier patches

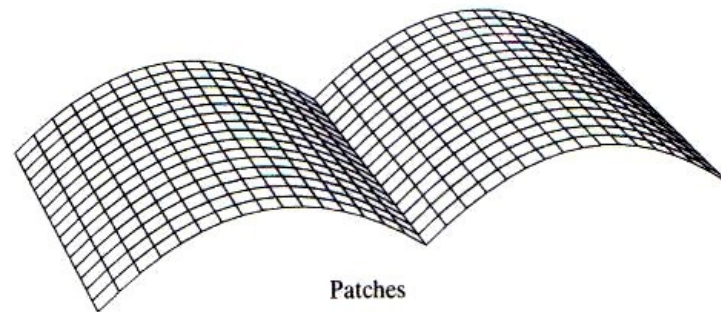
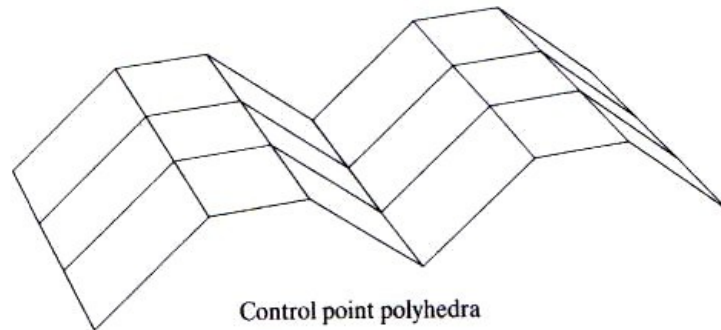
□ Positional or zero-order continuity:

■  $S(1,v) = R(0,v)$  for  $0 < v < 1$

■  $S_{3i} = R_{0i}$   $i = 0, \dots, 3$

Figure 5.27

(a) Positional continuity between bi-cubic Bézier patches, and (b) tangential continuity between bi-cubic Bézier patches.

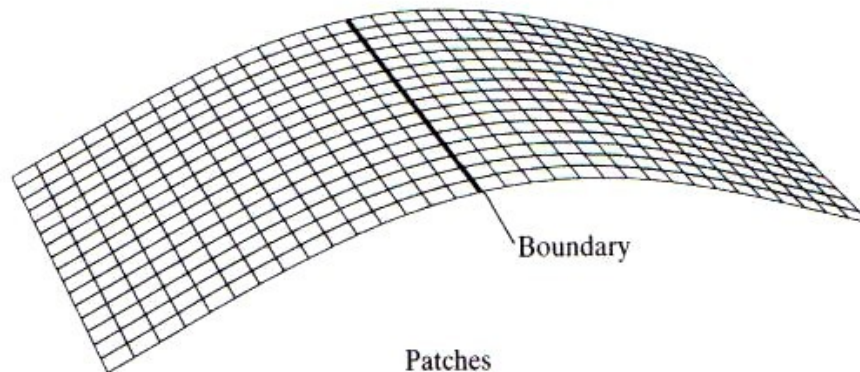
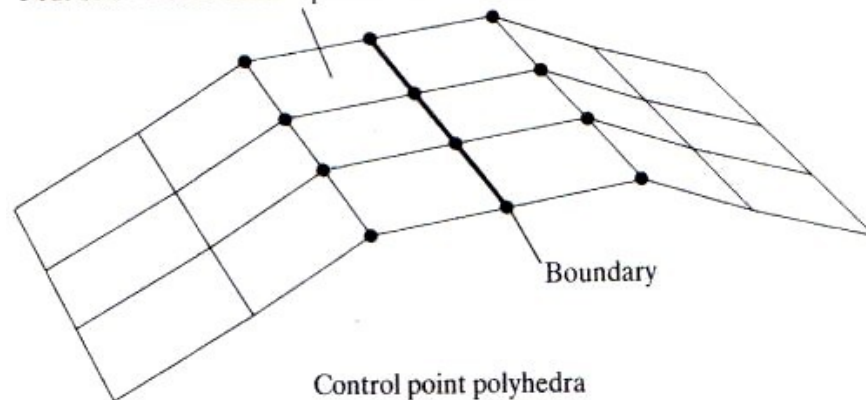


(a)

# 5.5.1 Continuity and Bezier patches

## □ First-order (C1) continuity

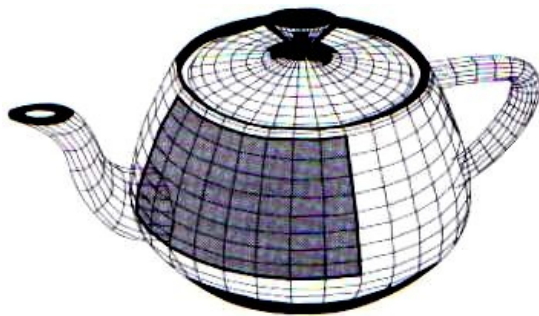
Four sets of three control points must be collinear



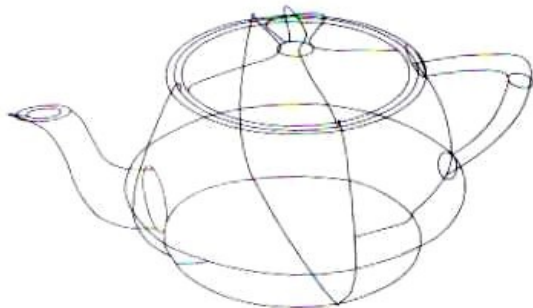
(b)

## 5.5.2 A Bezier patch object – the Utah teapot

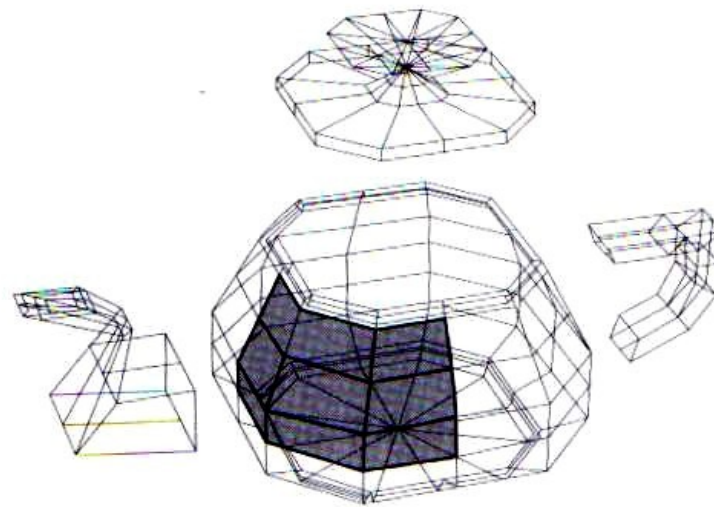
- 32 patches  $\times$  16 control points/patch  
= 288 vertices  
=  $288 \times 3$  real numbers



(a)



(c)



(b)

## 5.5.3 B-spline surface patches

---

- To form a bicubic B-spline surface patch we need to evaluate:

$$Q(u, v) = \sum_{i=0}^n \sum_{j=0}^m P_{ij} B_{i,j}(u, v)$$

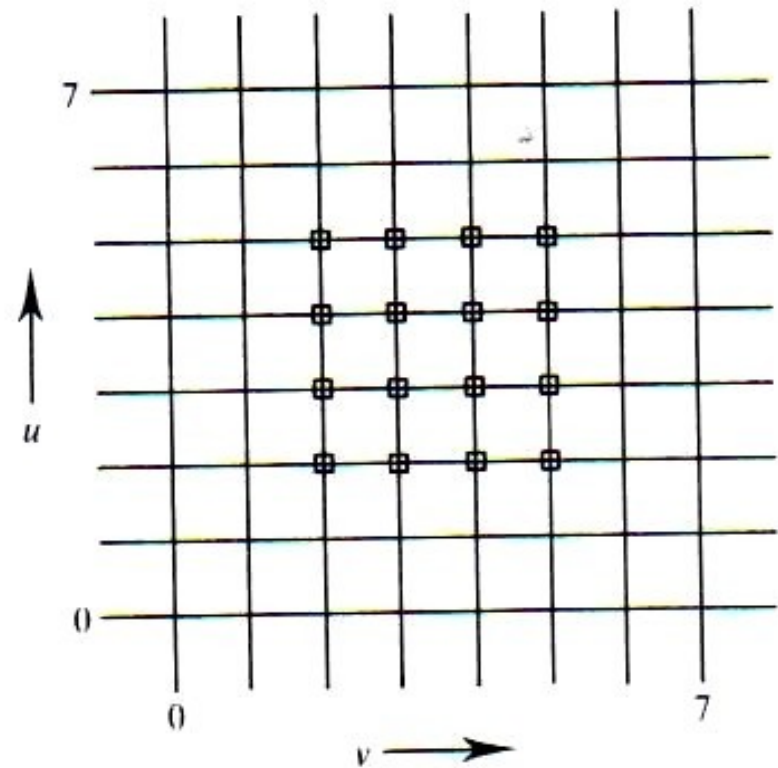
- $P_{ij}$  is an array of control points
- $B_{i,j}$  is a bivariate basis function

$$B_{i,j}(u, v) = B_i(u) B_j(v)$$

- Thus

$$Q(u, v) = \sum_{i=0}^n \sum_{j=0}^m P_{ij} B_i(u) B_j(v)$$

## 5.5.3 B-spline surface patches



**Figure 5.29**

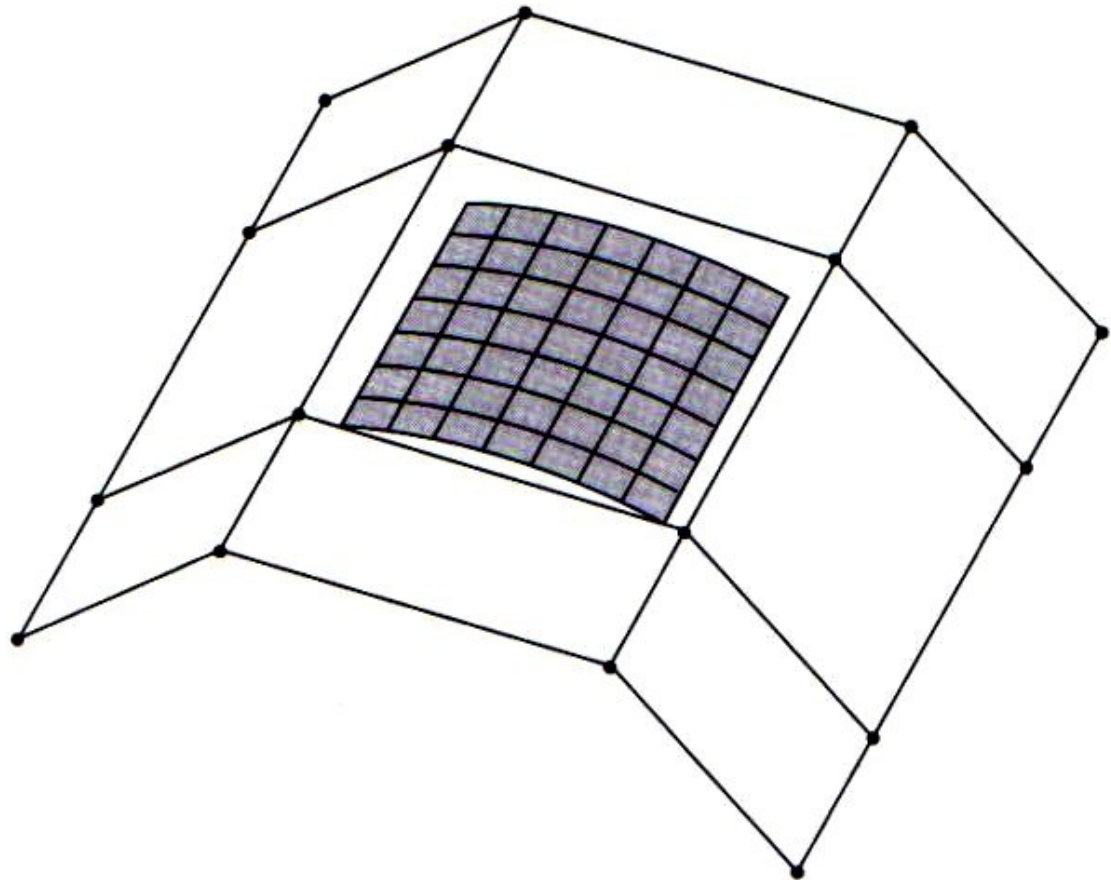
The 16 bivariate B-splines peak at the points shown in parametric space.

## 5.5.3 B-spline surface patches

### □ Single B-spline patch segment

**Figure 5.30**

A single B-spline patch segment.



## 5.6 Modelling or creating patch surface

### S

---

- In this section we will look at the following design or creation methods:
  - **Cross-section design**: here we will look at a simplification of the sweeping techniques described in the previous chapter, restricting the objects to linear axis design.
  - Interactive design by manipulating the control point polyhedron.
  - **Creating a patch net or mesh from a set of 3D points** representing a real object. We could call this surface interpolation, or surface fitting.



## 5.6.1 Cross-section modelling with patches

---

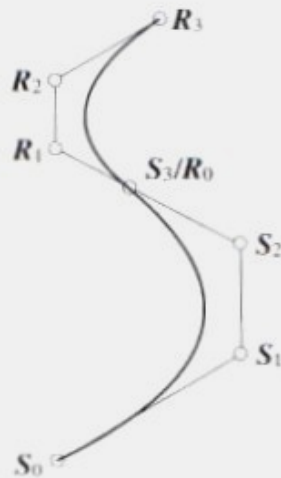
- Linear axis design – scaled circular cross-sections
- Linear axis design – non-circular scaled cross-sections
- Linear axis design - non-circular varying cross-sections

# 5.6.1 Cross-section modelling with patches

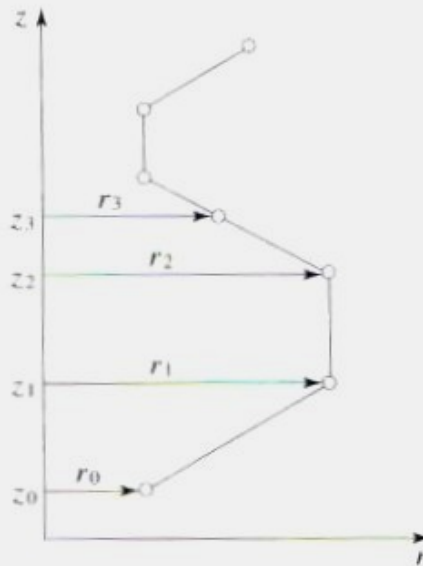
- Linear axis design – scaled circular cross-sections

Figure 5.31

Linear axis design – circular cross-section, object is designed by a single profile curve.



Profile design



$(r, v)$  coordinates of control points

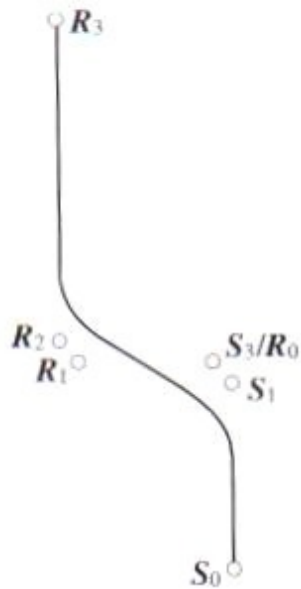


Object

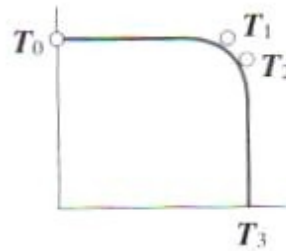
# 5.6.1 Cross-section modelling with patches

- Linear axis design – non-circular scaled cross-sections

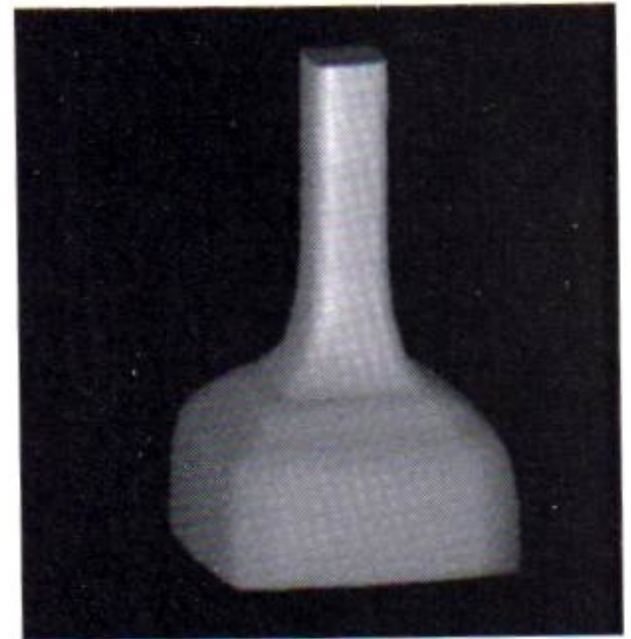
**Figure 5.32**  
Linear axis design – scaled  
(non-circular) cross-section.  
Object is designed by one  
profile curve and one (1/4)  
cross-section curve.



Profile design



Cross-section design



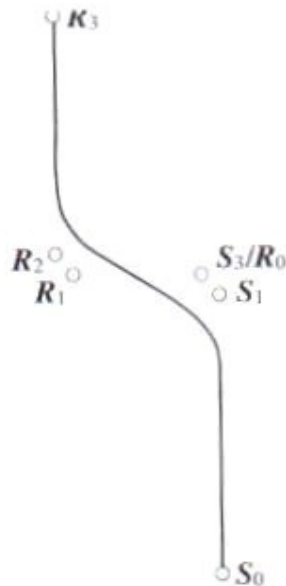
Object

# 5.6.1 Cross-section modelling with patches

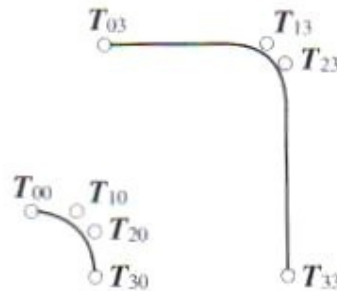
- Linear axis design - non-circular varying cross-sections

**Figure 5.33**

Linear axis design – non-circular varying cross-section. Object is designed by one profile curve and three cross-sections.



Profile design



Cross-section design

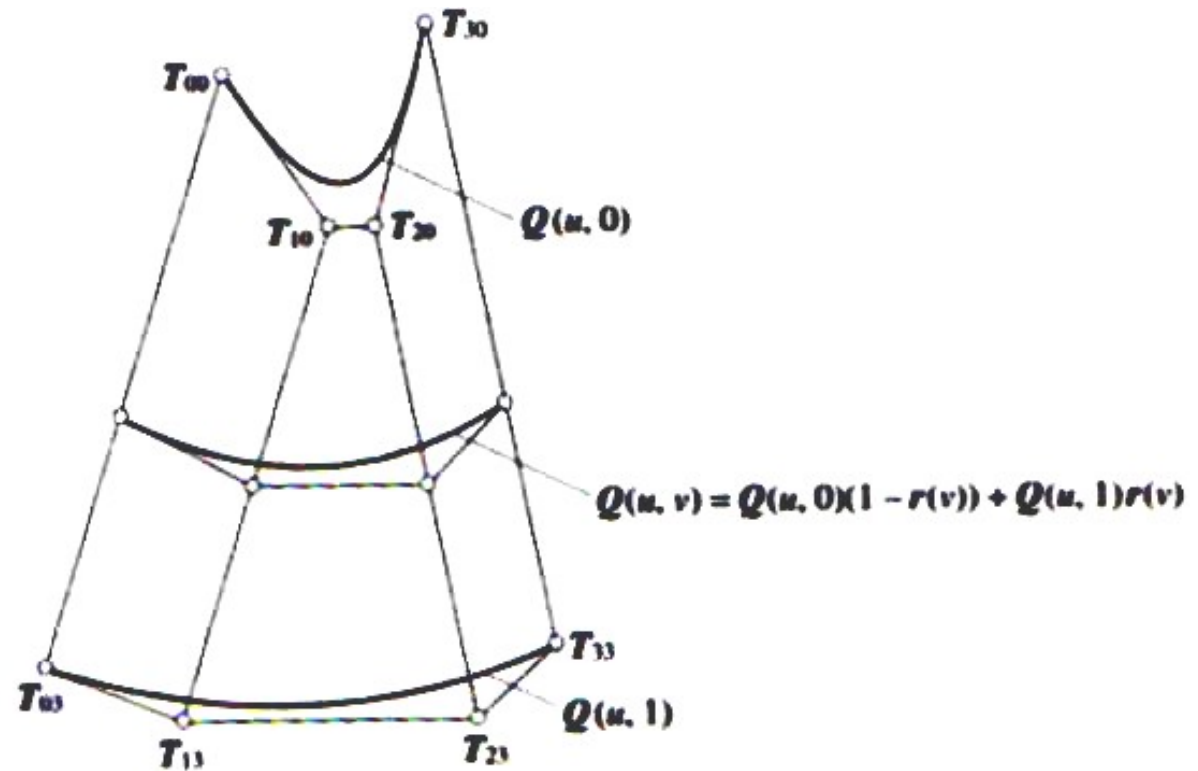


Object

## 5.6.1 Cross-section modelling with patches

- Blending two different cross-sections using the profile curve.

**Figure 5.34**  
Blending two different cross-sections using the profile curve.





## 5.6.2 Editing a patch net – altering an existing net

---

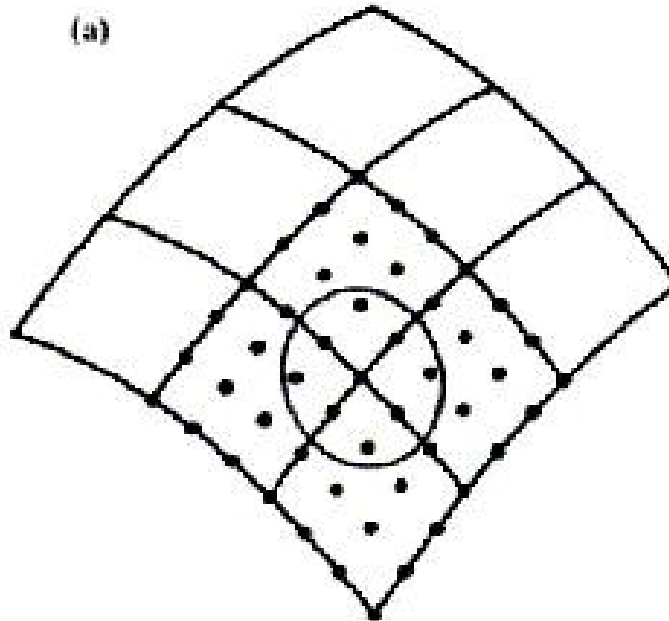
- Problems in editing mesh of patches
- Fine control
- Coarse control

# Problems in editing mesh of patches

- 四個連接在一起的 Bezier patch 及其控制點
- 連續性的限制條件導致中間點的移動必須考慮其週邊八點
- 九個點一起移動則可保有連續性

Figure 5.35

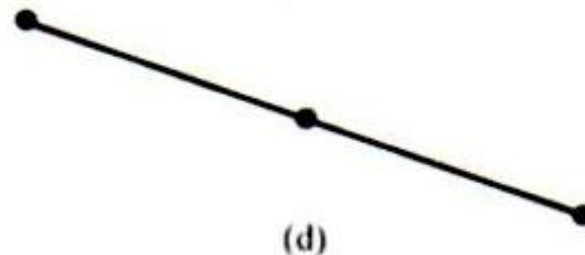
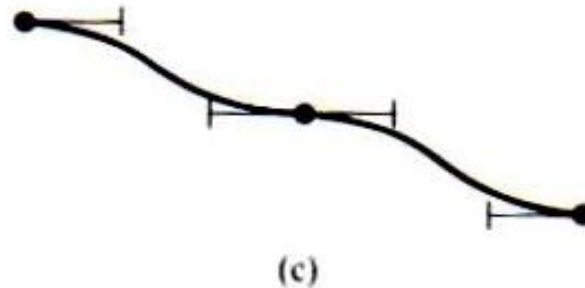
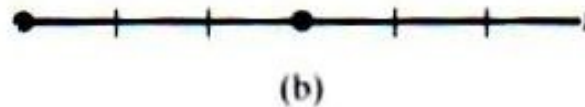
(a) Four adjoining Bézier patches and their control points. Continuity constraints imply that the central points cannot be moved without considering its eight neighbours. All nine points can be moved together and continuity maintained.



# Plateaux effect ( 高原效應 )

- 如果，我們想將 (b) 中的直線變形成 (d) 中的直線．使用一次移動一組控制點的方法無法產生如 (d) 之效果．僅能產生如 (c) 之效果

(b) Undeformed  
two-segment curve. (c)  
Deformed curve by moving  
control points in collinear  
groups. (d) Desired  
deformation.



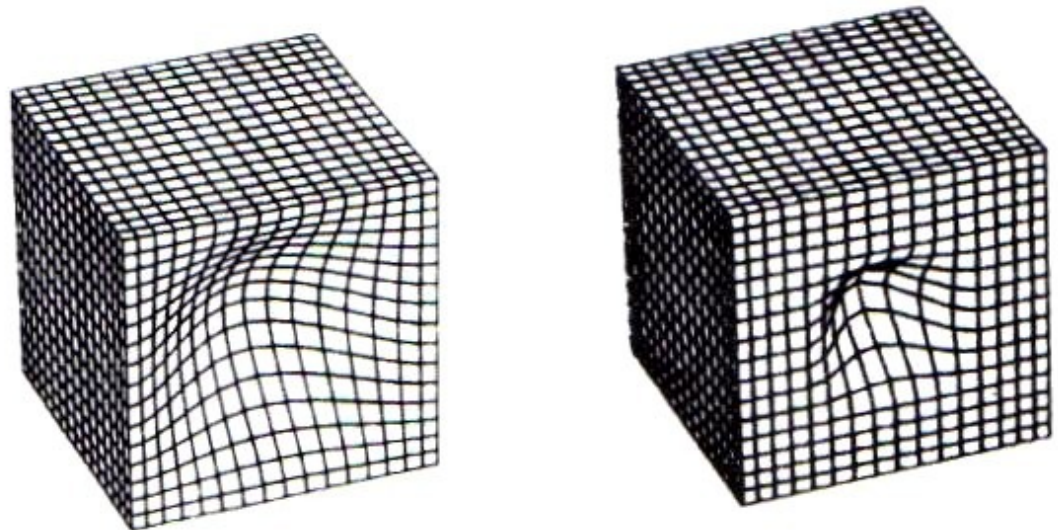


# Fine control

- Local shape deformation
- Locality of control
  - B-spline
- control the **scale** of deformation by locally **subdividing** the patches within the region of the deformation.

**Figure 5.36**

The scale of a deformation as a function of the number of (initially flat) patches used to represent the side of a cube.



# Hierarchical B-spline deformation

---

- In 1988, **Forsey and Bartels** introduced Hierarchical B-Splines
- Definition of a B-spline patch:

$$Q(u, v) = \sum_{i=0}^n \sum_{j=0}^m P_{ij} B_i(u) B_j(v)$$

- Redefined by knot Insertion (Farin, 1990) to a patch:

$$Q(u, v) = \sum_{i=0}^N \sum_{j=0}^M R_{ij} B_i(u) B_j(v)$$

- $N > n$  and  $M > m$

# Hierarchical B-spline deformation

---

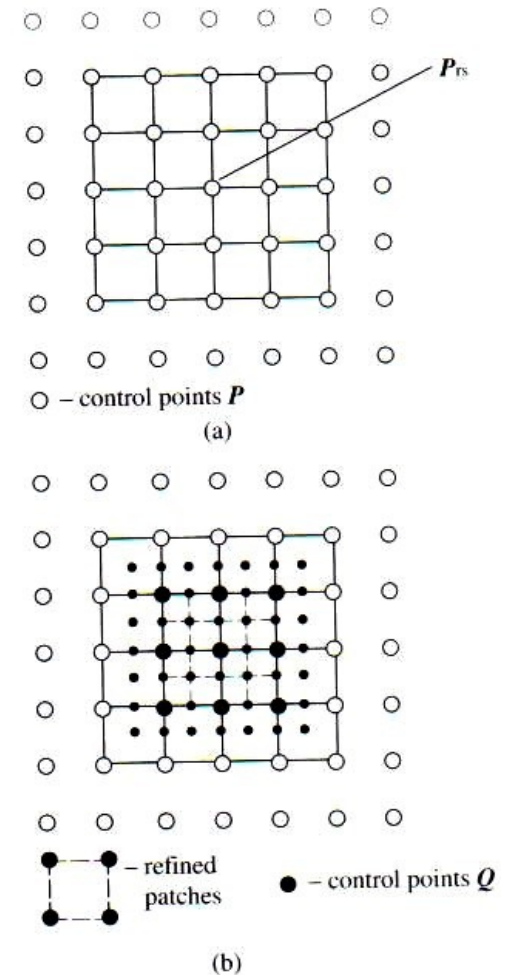
- The new control points are derived as described in Forsey and Bartels. The problem is how to apply this strategy to the region of a surface that interests us. Forsey and Bartels do this by defining a **minimal surface** – the smallest section of the surface to which this refinement of control points can be applied. This minimal surface satisfies **two constraints**:
  - Movement of the new control points produces deformations that are localised to this minimal surface.
  - The derivatives at the boundary of the minimal surface remain unchanged.

# Hierarchical B-spline deformation

- A minimal surface is 16 patches defined by a  $7 \times 7$  control points matrix.
- The control points that are required if the centre four patches are refined to 16.

Figure 5.37

(a) Sixteen patch minimal surface with 49 control points. (b) Central four patches refined to 16 patches (after Forsey and Bartels (1988)).



# Coarse control

---

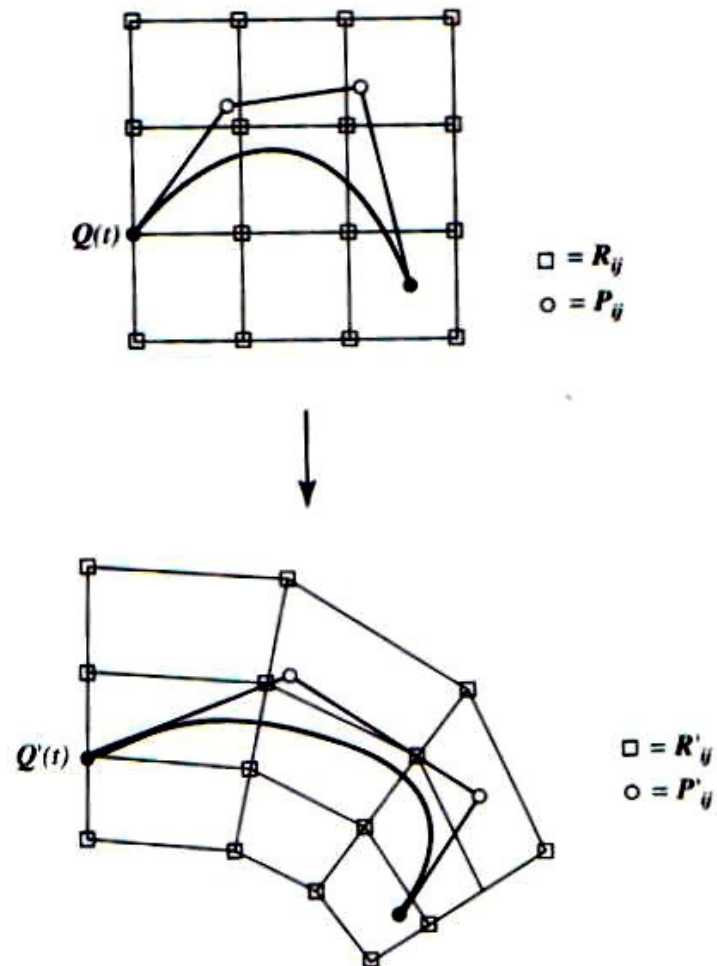
- Global shape changing
  - operate on all control points simultaneously
  - deform  $Q(t)$  defined by control points  $P_i$  to  $Q'(t)$  defined by control points  $P'_i$ :
    - enclose the curve  $Q(t)$  in a unit square and divide up this region into a regular grid of points  $R_{ij}$ ;  $i=0,\dots,3; j=0,\dots,3$ . If we consider the square to be  $uv$  space then we can write:

$$(u, v) = \sum_{i=0}^3 \sum_{j=0}^3 R_{ij} B_i(u) B_j(v)$$

- deform the patch by changing  $R_{ij}$  to  $R'_{ij}$ .

# Coarse control

**Figure 5.38**  
Global distortion of a planar  
curve (after Farin (1990)).



## 5.6.3 Creating patch objects by surface fitting

---

- Interpolating curves using B-splines
- Interpolating surfaces

# Interpolating curves using B-splines

---

- Two applications
  - Modelling
  - Computer animation
- Problem
  - 給定一組資料點，我們要求出一組控制點，使所定義出的曲線通過所有或部份的資料點
  - 方法可參考 Bartels et al. (1987)



# Formal definition for the problem

---

- Consider the data points to be knot values in  $u$  then we have for a cubic:

$$Q(u_p) = \sum_{i=0}^m P_i B_i(u_p) = D_p \quad \text{for all } p = 3, \dots, m+1$$

where :

$D_p$  is a data point

$u_p$  is the knot value corresponding to the data point

- The problem we now have is to **determine  $u_p$**

# Solution for determine $u_p$

---

- Uniform parametrisation
  - Set up to  $p$
  - Completely ignores the geometric relationship between data points and is usually regarded as giving the poorest interpolant in a hierarchy of possibilities described in detail in Farin (1990).
- Chord length parametrisation proportional
  - Set the knot intervals proportional to the distance between the data points.

# Solving control points $P_{xi}$

---

- Consider  $x$  component of  $D_p$ :

$$x(u_p) = \sum_{i=0}^m P_{xi} B_i(u_p) = D_{xp}$$

for all  $p = 3, \dots, m+1$

Where  $D_{xp}$  is the  $x$  component of the data point.

- A system of equations that we solve  $P_{xi}$  is:

$$\begin{bmatrix} B_0(u_3) & \cdots & B_m(u_3) \\ \vdots & \ddots & \vdots \\ B_0(u_{m+1}) & \cdots & B_m(u_{m+1}) \end{bmatrix} \begin{bmatrix} P_{x0} \\ \vdots \\ P_{xm} \end{bmatrix} = \begin{bmatrix} D_{x3} \\ \vdots \\ D_{xm+1} \end{bmatrix}$$



# Interpolating surfaces

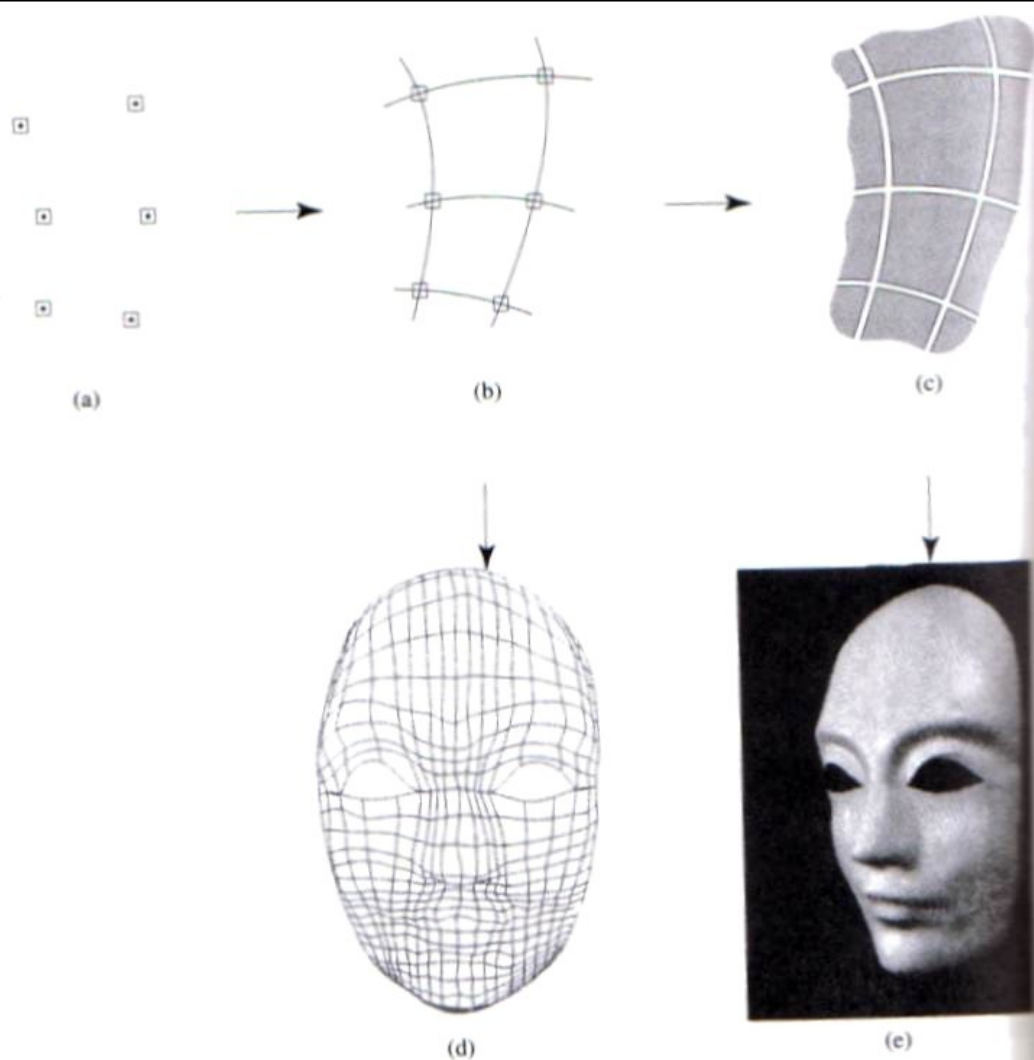
---

- 2 stages
  - find fit B-spline curves
  - B-spline curves are converted to Bezier curves

# A schematic representation of surface fitting

Figure 3.43

A schematic representation of surface fitting. (a) A set of points in three space. (b) Fitting curves through the points in two parametric directions. (c) The grid of curves from the boundaries of the patches. (d) A curve network obtained by interpolation through digitized points. (e) A rendered version of the patch model obtained from (d).



## □ Converting a B-spline curve network to Bézier patches

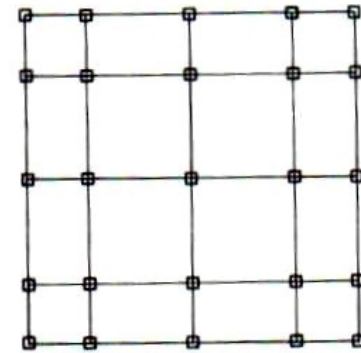
**Figure 5.41**

Converting a B-spline curve network to Bézier patches (after Farin (1990)).

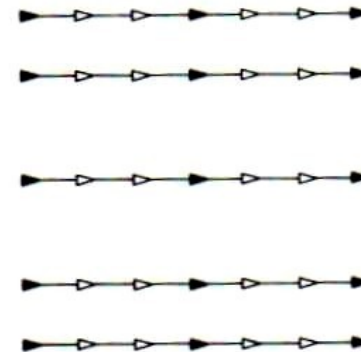
(a)  $5 \times 5$  two-segment B-spline curve network.

(b) Curve network converted row-wise to 5 two-segment Bézier curves.

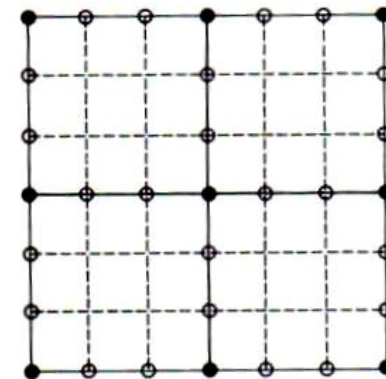
(c) Curve network converted to  $7 \times 7$  two-segment Bézier curves forming the boundaries of four Bézier patches.



(a)



(b)



(c)

## 5.7 Rendering parametric surfaces

---

- The rendering algorithms are divided into two categories:
  - Rendering directly from the parametric description, or the equation describing the **patch**.
  - Rendering by approximating the surfaces with **polygon meshes**.

## 5.7.1 Patch to polygon conversion

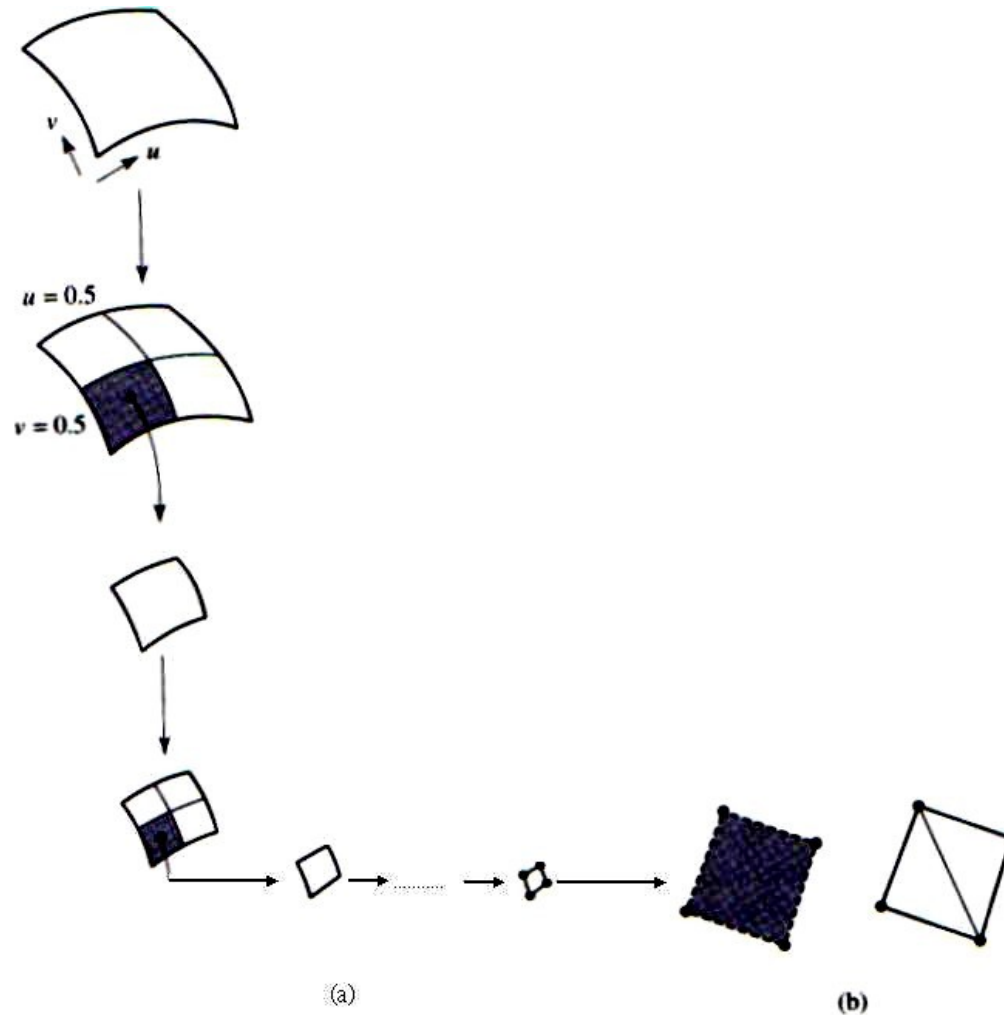
---

- How to derive the polygon mesh approximation
  - Iteratively applies **subdivision** to the patches to sufficient accuracy.
- When to stop the subdivisions
  - Depends on the criterion used to determine the depth of the subdivision and where it is applied on the surface.
  - Types of criteria
    - Screen space
    - Object space



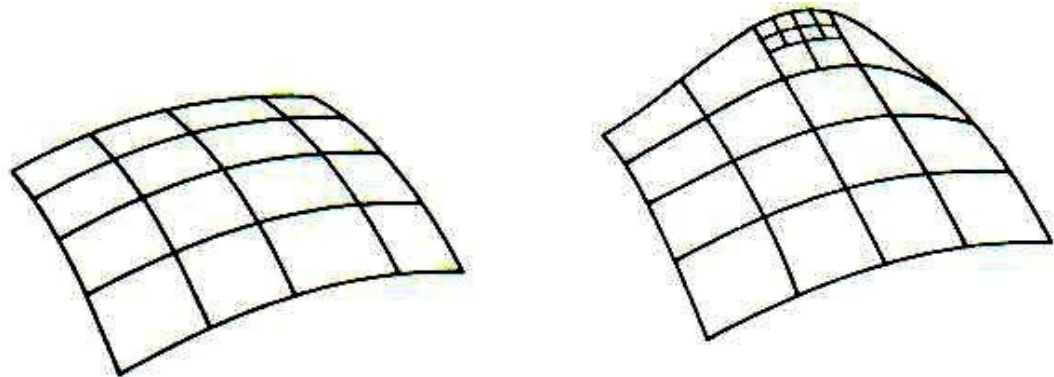
# The patch splitting process

**Figure 5.43**  
The patch splitting process.  
(a) Continue process until flat and (b) convert the vertices into two triangular facets.



# Object space subdivision

- Dealing with object space first we can list the following simple categories:
  - Uniform subdivision
    - This is the simplest case and involves a **user specifying** a level at which uniform subdivision of all patches is to terminate.
  - Non-uniform subdivision
    - This mean stopping the subdivision when the subdivision products meet a patch **flatness criterion**.



**Figure 5.44**  
Uniform and non-uniform  
subdivision of a Bézier  
patch.

# Uniform subdivision

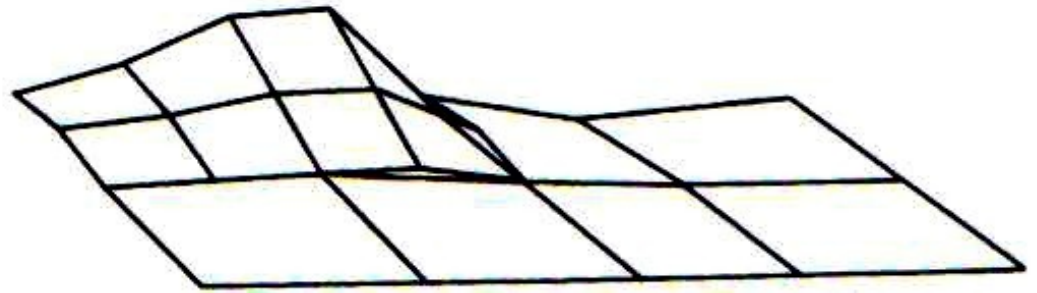
---

- Iteratively divide the patches using iso-parametric curves for a fixed level of iteration.
- Use the derived net of points to define the vertices for the mesh of polygons.
- Disadvantages
  - Visible boundary and silhouette edges may exhibit discontinuities.
  - Internal silhouette edges will generally be of higher degree than cubic.

# Non-uniform subdivision

- The patch is subdivided to a degree that depends on local curvature
  - Areas of the patch that are flattish are subject to few subdivision
  - Areas where local curvature is high are subject to more subdivisions.
- Advantage
  - Fast
  - Allow LOD control for interactive system
- Disadvantage
  - Holes may appear between patches with different level of subdivision.

**Figure 5.45**  
Tears produced by non-uniform subdivision of patches.



# Curve subdivision

---

- By Lane et al.(1980)
- Given a Bezier curve segment Q defined by  $Q_0, Q_1, Q_2, Q_3$ , the subdivision of Q can be given by:

$$R_0 = Q_0$$

$$S_0 = R_3$$

$$R_1 = (Q_0 + Q_1) / 2$$

$$S_1 = (Q_1 + Q_2) / 4 + S_2 / 2$$

$$R_2 = R_1 / 2 + (Q_1 + Q_2) / 4$$

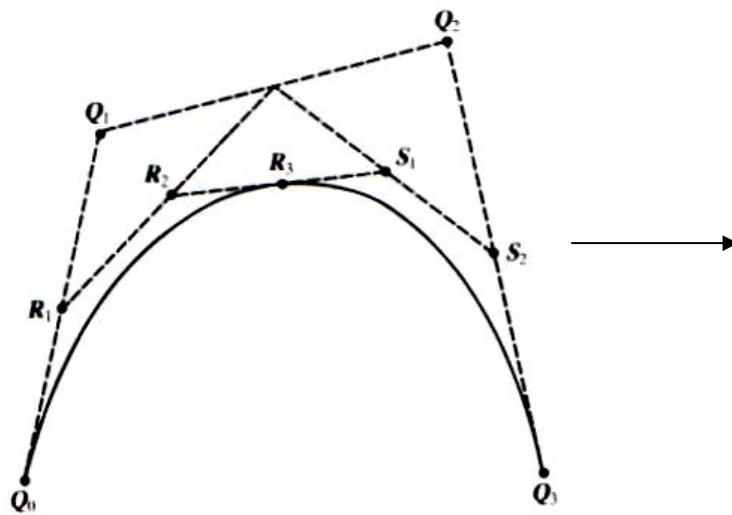
$$S_2 = (Q_2 + Q_3) / 2$$

$$R_3 = (R_2 + S_1) / 2$$

$$S_3 = Q_3$$

# Curve subdivision

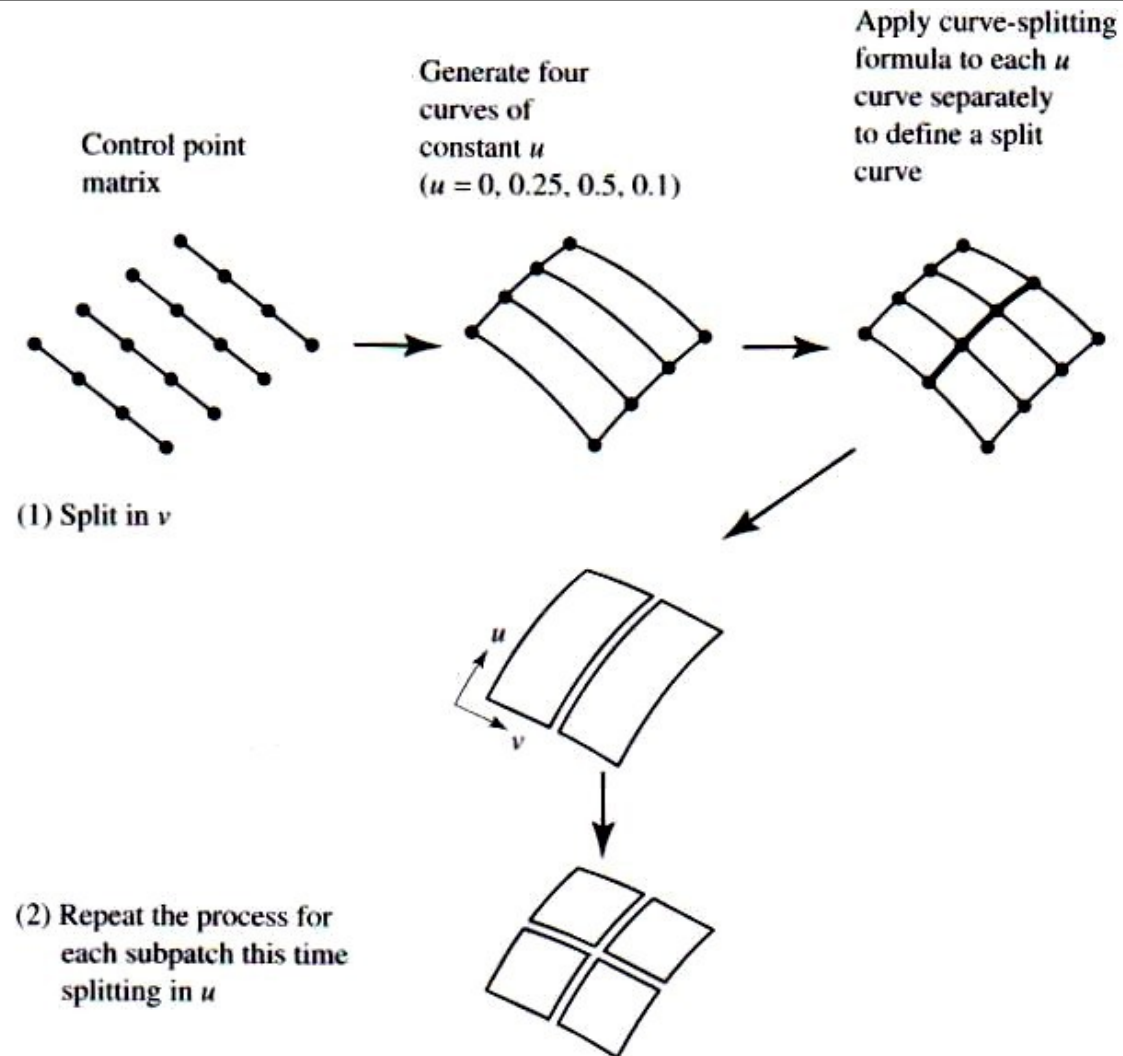
**Figure 5.46**  
Splitting a bi-cubic Bézier curve.



**Figure 5.47**  
Drawing the control points at each level of subdivision.



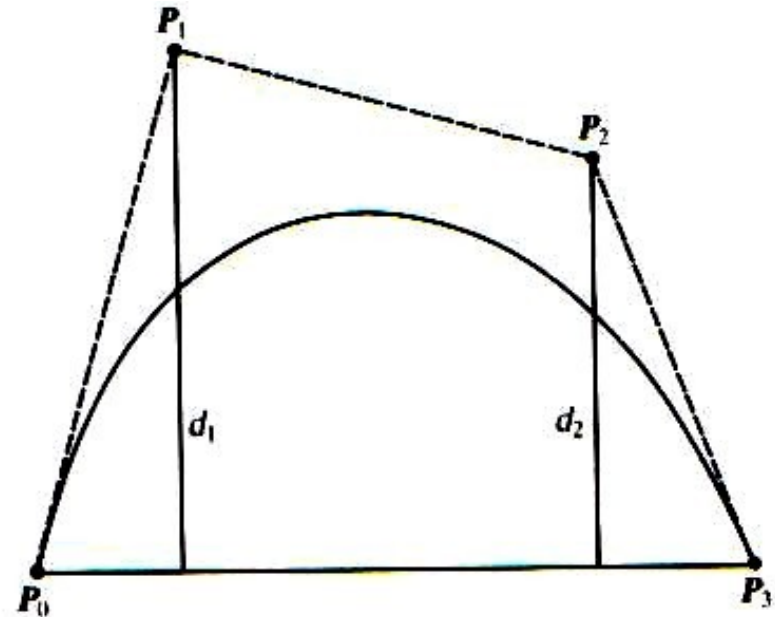
# Using curve splitting to subdivide a patch into four



**Figure 5.48**  
Using curve splitting to  
subdivide a patch into four.

# Linearity criteria

- Curve subdivision
  - Distances from the middle two points to the end point joining line.



**Figure 5.49**  
A cubic Bézier curve with  
control points  $P_0$ ,  $P_1$ ,  $P_2$ ,  $P_3$ .



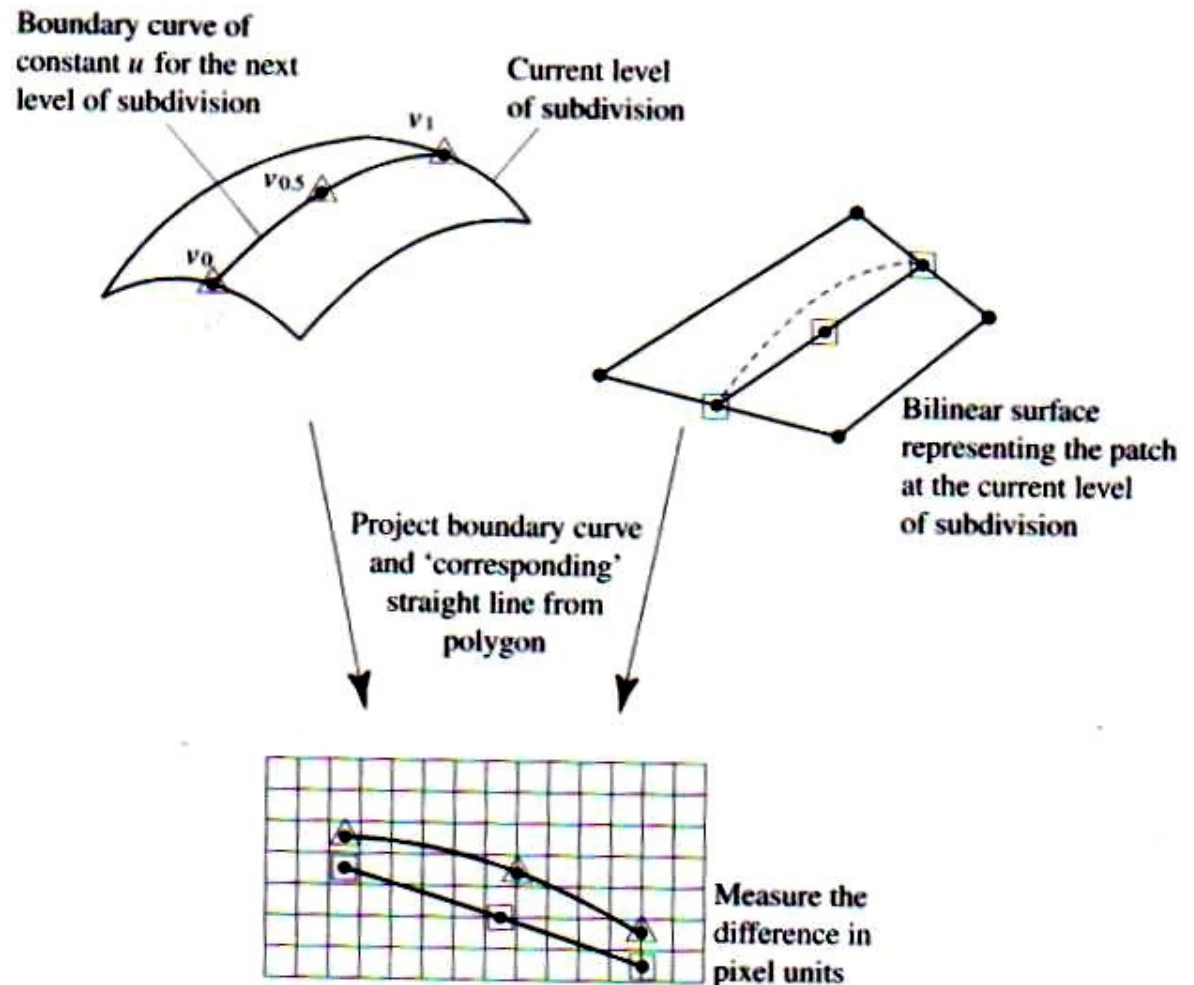
# Image space subdivision

---

- Controlling the depth of subdivision using screen space criteria.
- Called view dependent or screen space controlled
- Three ways to do this
  - Minimum space of the patch
  - Screen space flatness of the patch
  - Screen space flatness of the silhouette edge

# Screen space flatness of the patch

**Figure 5.51**  
Screen space subdivision  
termination increasing the  
flatness of the patch in  
screen space at its current  
level of subdivision.



## 5.8 Practical Bezier technology for games

---

- Down-sampling bi-quadratic meshes
- Curves, objects and subdivision

## 5.8.1 Down-sampling bi-quadratic meshes

---

- A bi-quadratic patch is defined as:

$$Q(u, v) = \sum_{i=0}^2 \sum_{j=0}^2 P_{ij} B_i(u) B_j(v)$$

- Where the quadratic blending functions are:

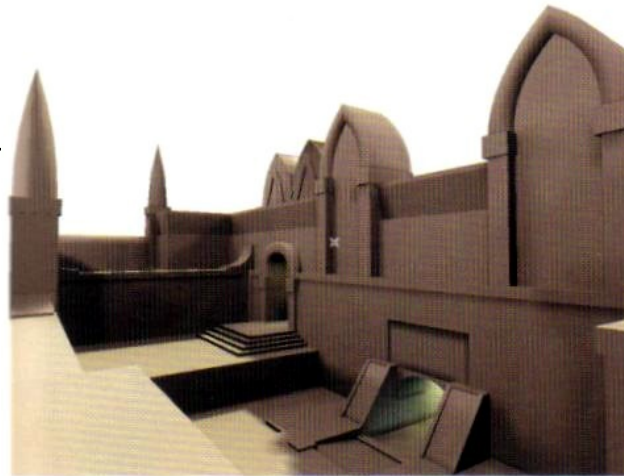
$$B_0(u) = (1 - u)^2$$

$$B_1(u) = 2(1 - u)u$$

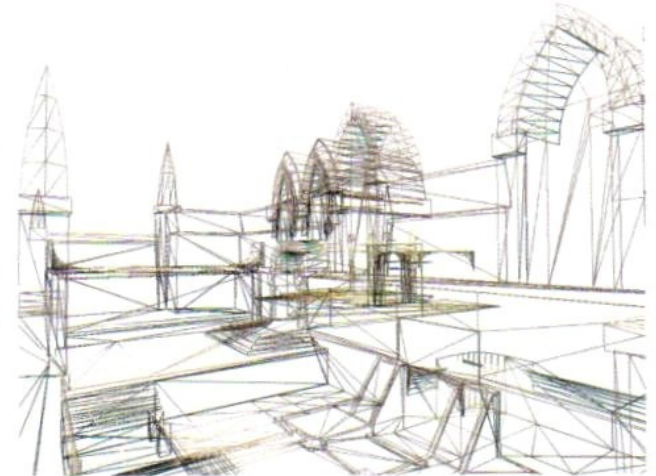
$$B_2(u) = u^2$$

# 5.8.1 Down-sampling bi-quadratic meshes

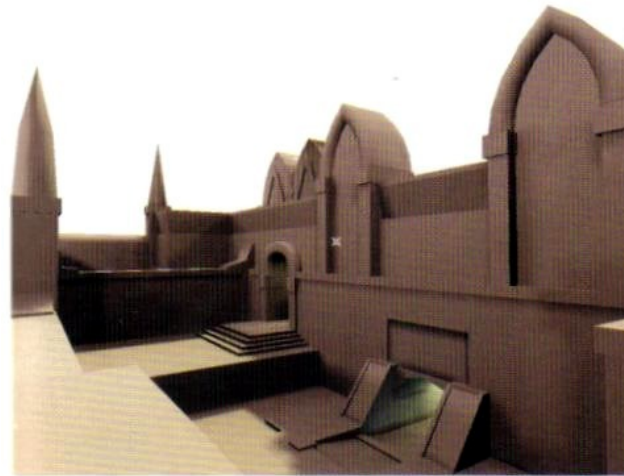
## □ Example



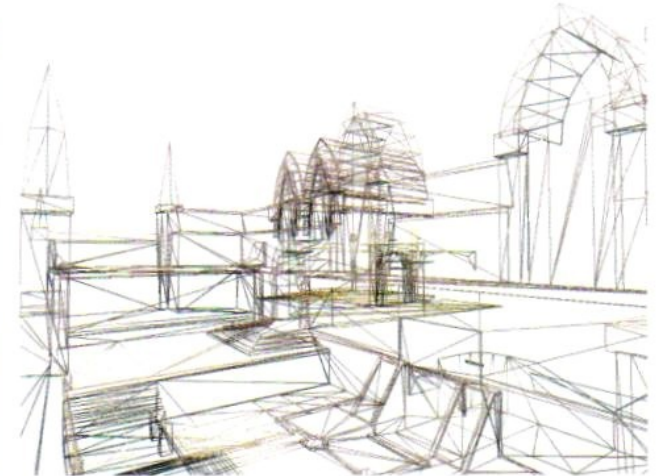
(a)



(b)



(c)



(d)

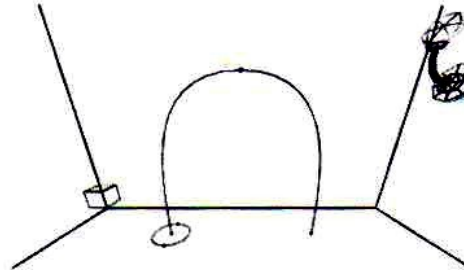
Figure 5.52

Two consecutive LODs (generated from a Quake 3 level) shown in wireframe and rendered. These are generated by down-sampling the pre-calculated Bezier mesh. Only light map rendering is used to emphasize the silhouette edges.

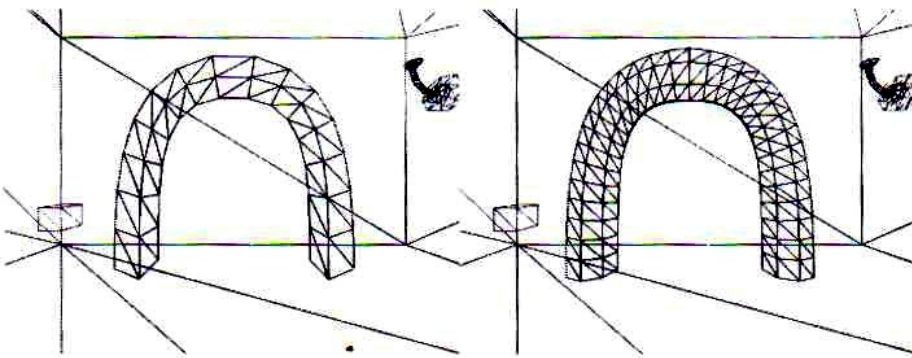
## 5.8.2 Curves, objects and subdivision

- Converting a ducted solid generator

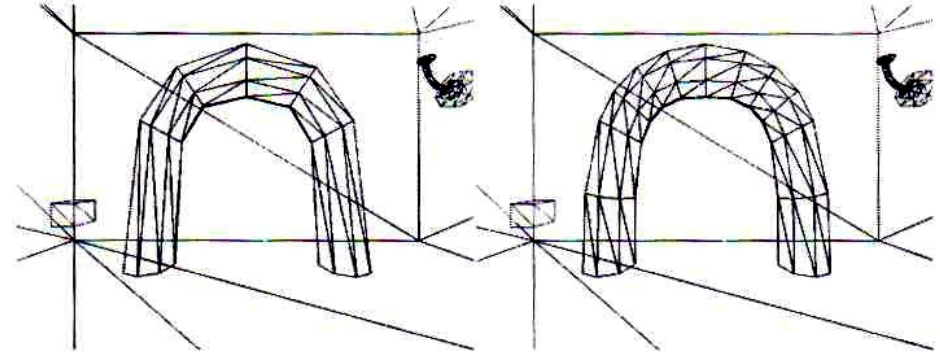
**Figure 5.53**  
Converting a ducted solid  
generator into triangles.



Generator: 3 Bézier curves



2 levels of uniform subdivision



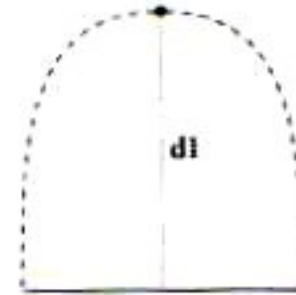
2 levels of non-uniform subdivision

# Non-uniform curve subdivision

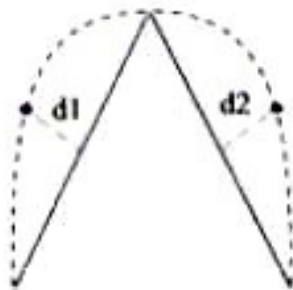
**Figure 5.54**  
Non-uniform curve subdivision.



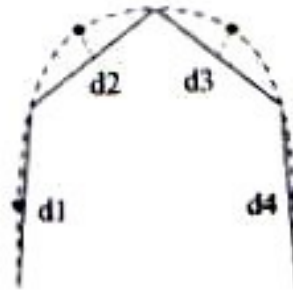
Bézier Curve  
2 Segments  
7 Control Points



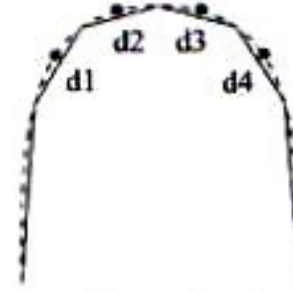
$d1 > \text{err}(\text{subdiv})$



$d1 > \text{err}(\text{subdiv})$   
 $d2 > \text{err}(\text{subdiv})$



$d1 < \text{err}(\text{stop})$   
 $d2 > \text{err}(\text{subdiv})$   
 $d3 > \text{err}(\text{subdiv})$   
 $d4 < \text{err}(\text{stop})$



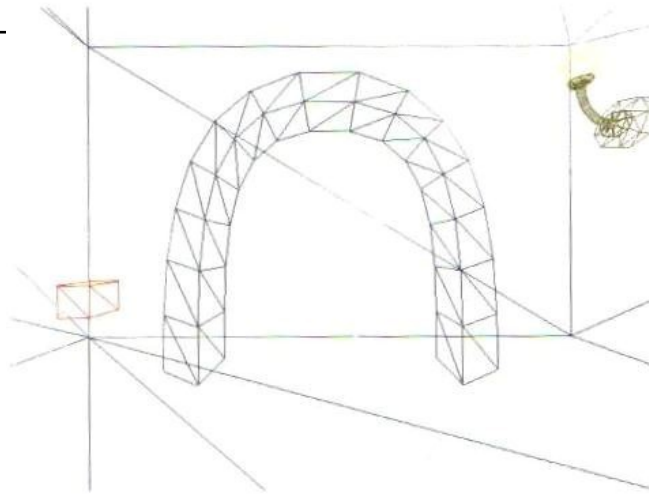
$d1 < \text{err}(\text{stop})$   
 $d2 < \text{err}(\text{stop})$   
 $d3 < \text{err}(\text{stop})$   
 $d4 < \text{err}(\text{stop})$



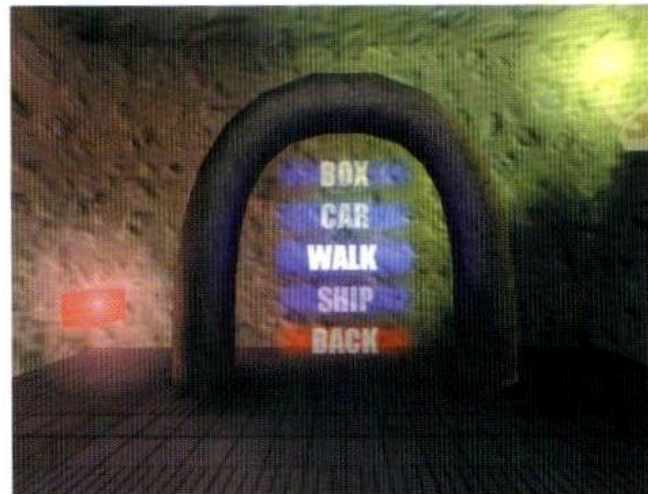
# 2 levels of uniform subdivision

Figure 5.53

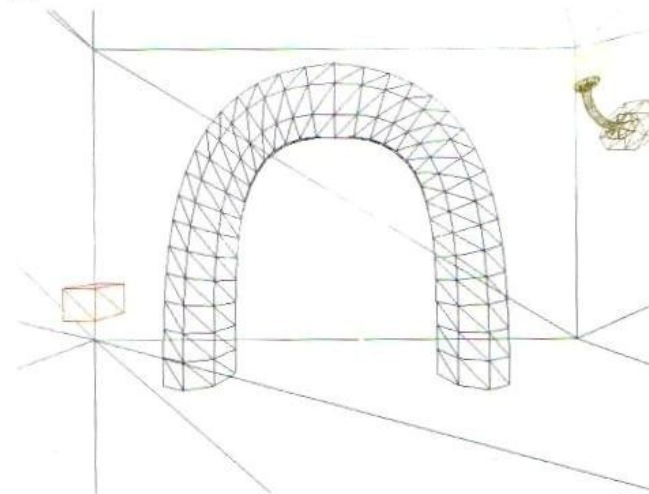
Rendered imagery for uniform and non-uniform subdivision.  
2 levels of uniform subdivision



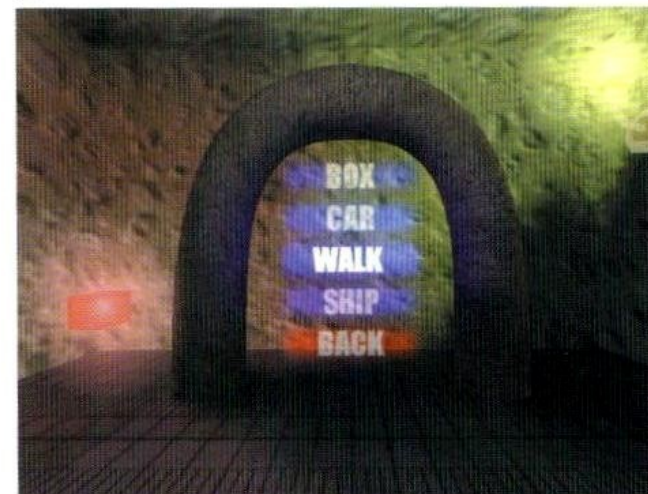
(a)



(b)



(c)



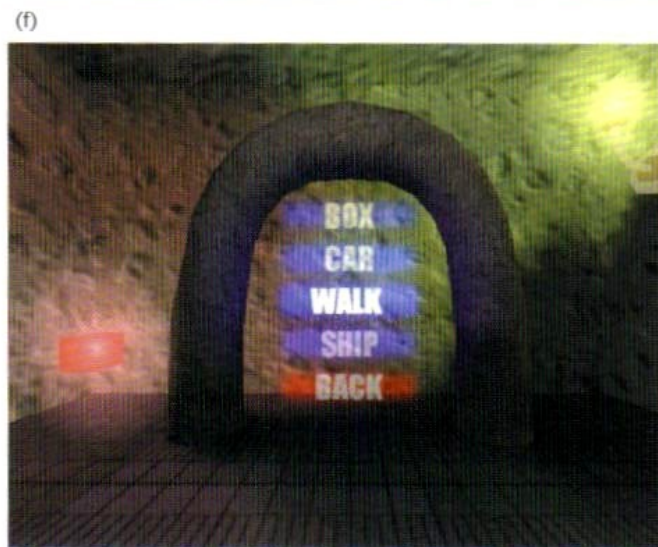
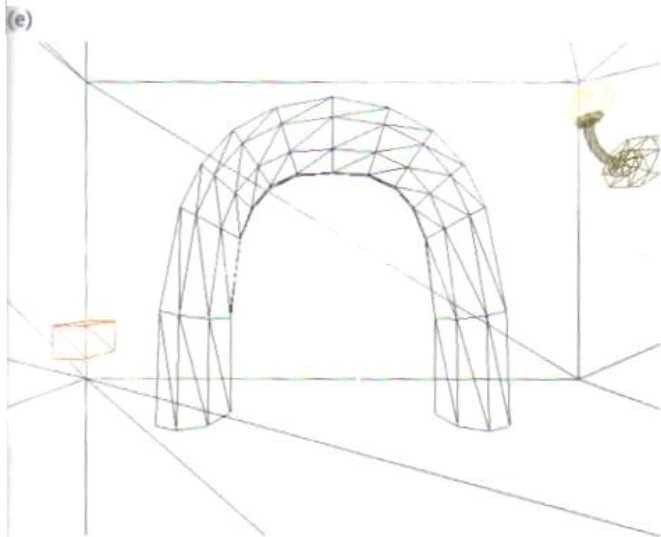
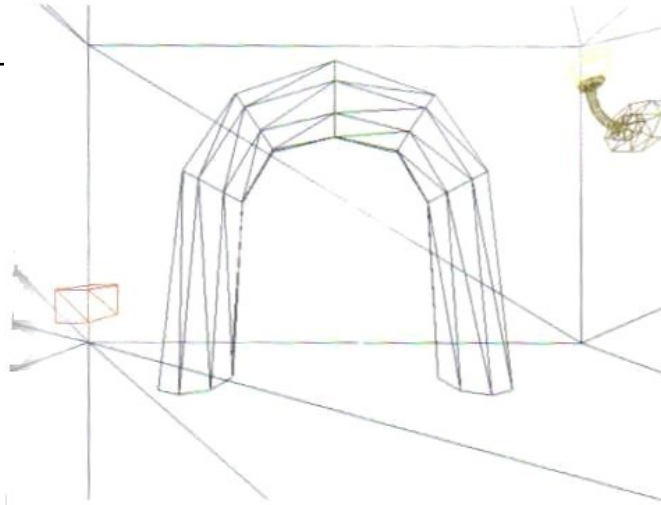
(d)



# 2levels of non-uniform subdivision

Figure 5.53

Rendered imagery for uniform and non-uniform subdivision.  
2 levels of non-uniform subdivision



(g)

(h)

# No subdivision in the cross-section

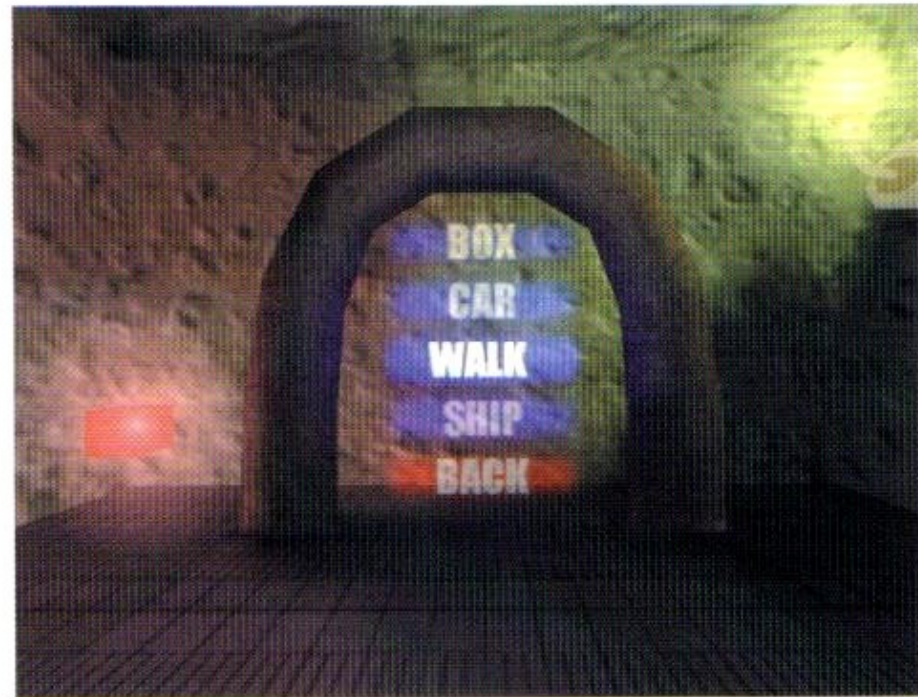
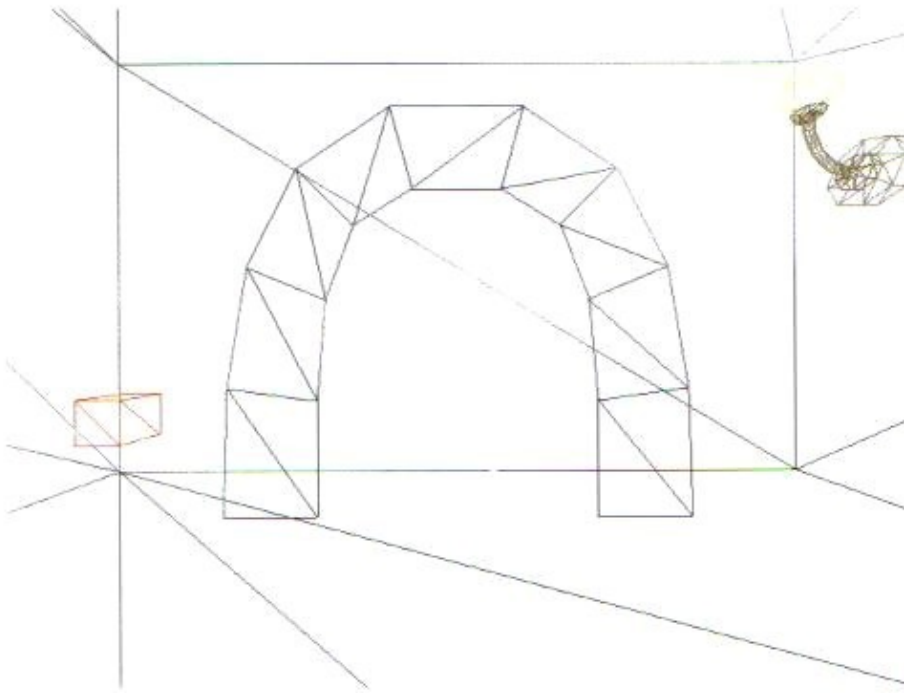


Figure 5.55

No subdivision in the cross-section. The arch is completely flat. Despite this when viewed from the front it still looks curved.



## 5.9 Subdivision surfaces

---

5.9.1 Catmull-Clark subdivision

5.9.2 Butterfly subdivision

5.9.3 Modified butterfly

## 5.9.1 Catmull-Clark subdivision

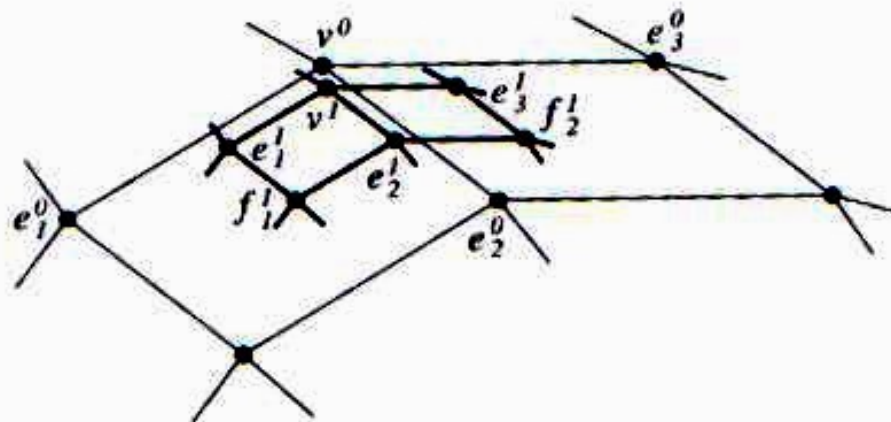
- Catmull-Clarks' original convention:
  - New vertices are placed at the centre of each original face. These are calculated as the average of the positions of original vertices of the face.
  - New edge points are created. These are calculated as the average of the midpoints of the original edge with the average of the two new face points sharing the edge.
  - New vertex points are formed as the average:
    - $Q/n + 2R/n + S(n-3)/n$ 
      - Q is the average of the new face points surrounding the old vertex
      - R is the average of the midpoints of the edges that share the old vertex
      - S is the old vertex point
      - n number of edges that share the vertex
  - The new mesh is then formed by :
    - Connecting each new face point to the new edge points of the edges that form the original face
    - Connecting each new vertex point to the new edge points of all original edges forming the original vertex.

## 5.9.1 Catmull-Clark subdivision

- De Rose et al. (1998) formulated these rules in a succinct convention **notates** the **subdivision level** and the **vertices**
- Denote the control mesh as  $M^0$  and subdivision meshes as  $M^1, M^2, \dots$
- They generated formulae for **edges** and **vertices** as :

$$e_j^{i+1} = \frac{v^i + e_j^i + f_{j-1}^{i+1} + f_j^{i+1}}{4}$$

$$v^{i+1} = \frac{n-2}{n} v^i + \frac{1}{n^2} \sum_j e_j^i + \frac{1}{n^2} \sum_j f_j^{i+1}$$

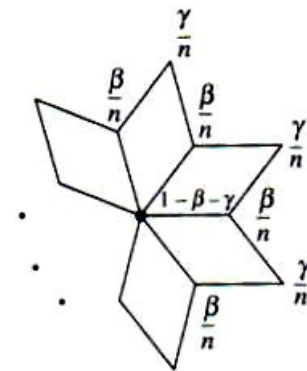
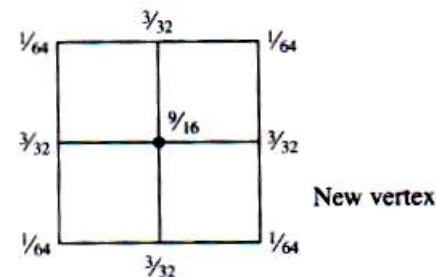
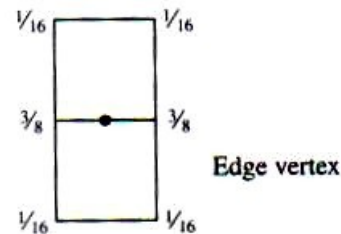
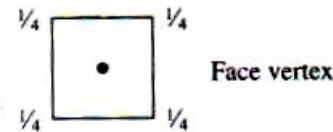


**Figure 5.56**  
Notation of De Rose et al.  
(1998).

# Mask notation

- Mask is shown simply as a **black dot** together with a set of coefficients.
- The vertices of the mask indicates where it is to be placed on the mesh and the coefficients define the subdivision rule.

**Figure 5.57**  
Masks for Catmull-Clark subdivision rules.



Extraordinary points  
(Catmull and Clark suggest

$$\beta = \frac{3}{2n} \quad \gamma = \frac{1}{4n}$$

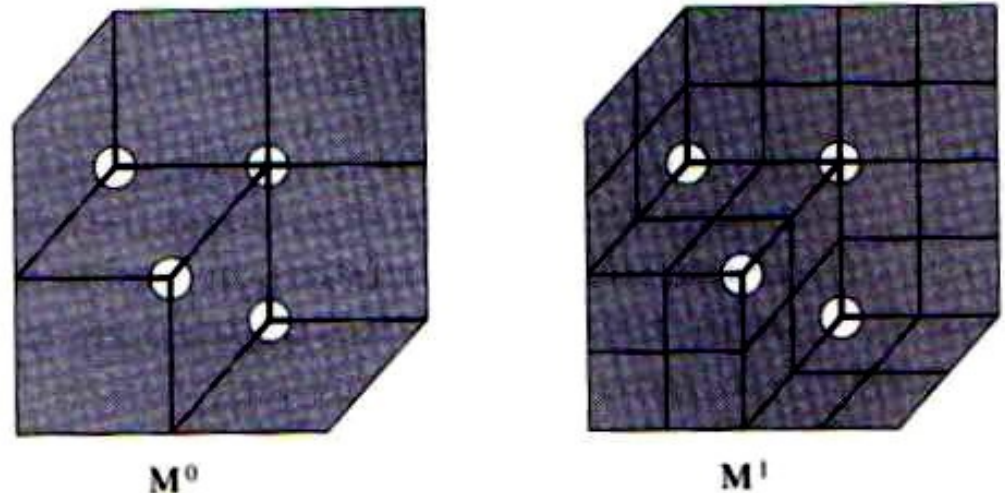


# Valence of vertex

- The valence of a vertex is defined as the **number of edges** incident on it and the vertices corresponding to the original control points retain the valence of these points.
- Vertices in the surface which do not have valence 4 are called **extraordinary** points and it is only at these points that the subdivision surface is not a standard B-spline surface.

**Figure 5.58**

Extraordinary points in a subdivided mesh. The valence of these points in the control mesh is retained in the subdivided mesh.



## 5.9.2 Butterfly subdivision

---

- Butterfly scheme (Dyn et al., 1990) is a popular subdivision approach method.
- This is an **interpolating method** – the limit surface passes through the  $M^0$  points.
- This mean it can be used to refine existing models.



## 5.9.2 Butterfly subdivision

- To implement a butterfly scheme we can process as follows (Figure 5.65 (a) ):

For each edge in the model, add a new vertex as:

- edge defined by vertex  $v_0[0]$  and  $v_0[1]$
- edge also defined by the faces  $f_0[0]$  and  $f_0[1]$
- find vertices  $v_1[0]$  and  $v_1[1]$  (other edges from  $f_0[0..1]$  that are not  $v_0[0..1]$ )
- find edges  $e[0..3]$  using  $v_0[0..1]$  and  $v_1[0..1]$
- find faces  $f_1[0..3]$  using  $e[0..3]$  (other faces from  $e[0..3]$  that are not  $f_0[0..1]$ )
- find vertices  $v_2[0..4]$  using the  $f_1[0..3]$  and  $e[0..3]$  (vertices from  $f_1[0..3]$  that are not from  $e[0..3]$ )
- add  $v_0[0..1]$ ,  $v_1[0..1]$  and  $v_2[0..3]$  with their weights to form  $v$

where:

$v$  – the vertex being added and  $v_0[0..1]$  – the current edge vertices (weight  $1/2$ )

$v_1[0..1]$  – vertices with weight  $1/8 + 2w$

$v_2[0..3]$  – vertices with weight  $-1/16 - w$

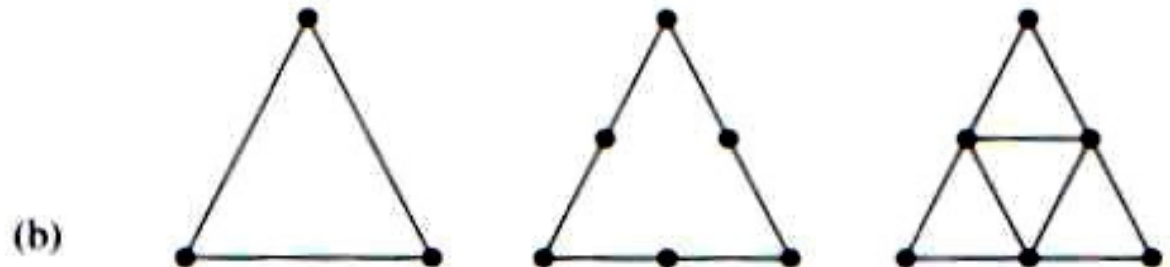
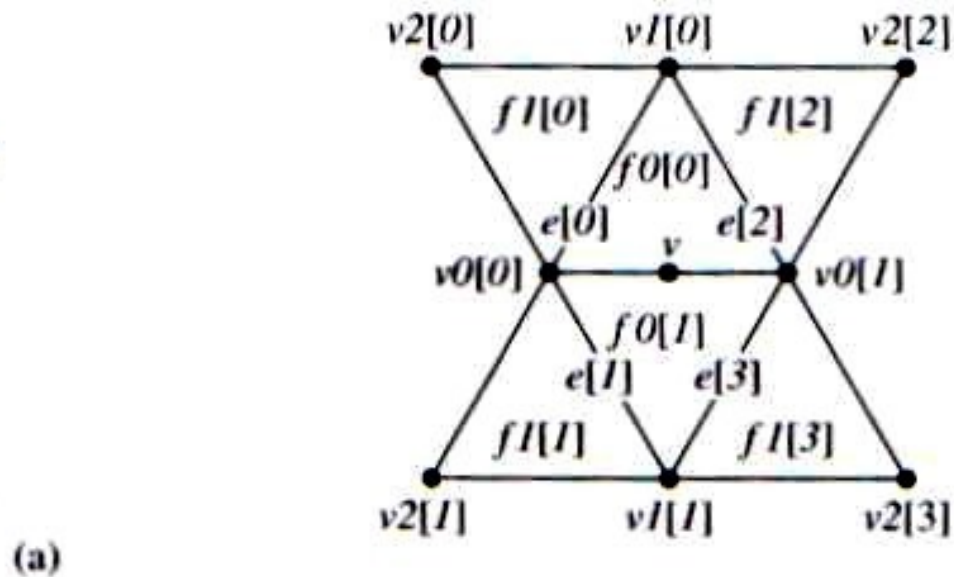
$f_0[0..1]$  – the two faces sharing the current edge  $f_1[0..4]$  – the faces connected to  $f_0[0..1]$

Then each face is converted into four using its edge vertices (Figure 5.65(b)).

## 5.9.2 Butterfly subdivision

**Figure 5.65**

Implementing a butterfly scheme. (a) Adding a new vertex  $v$  to an edge. (b) Convert each face into four using new edge vertices.



# Mesh refinement using butterfly scheme

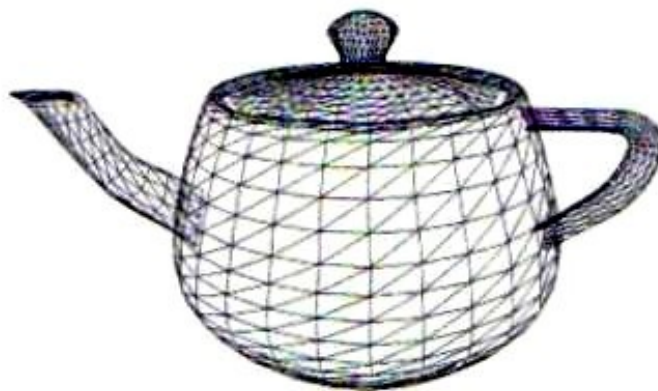
**Figure 5.59**  
Mesh refinement using  
(unmodified) butterfly  
scheme.



Original mesh



Level 1



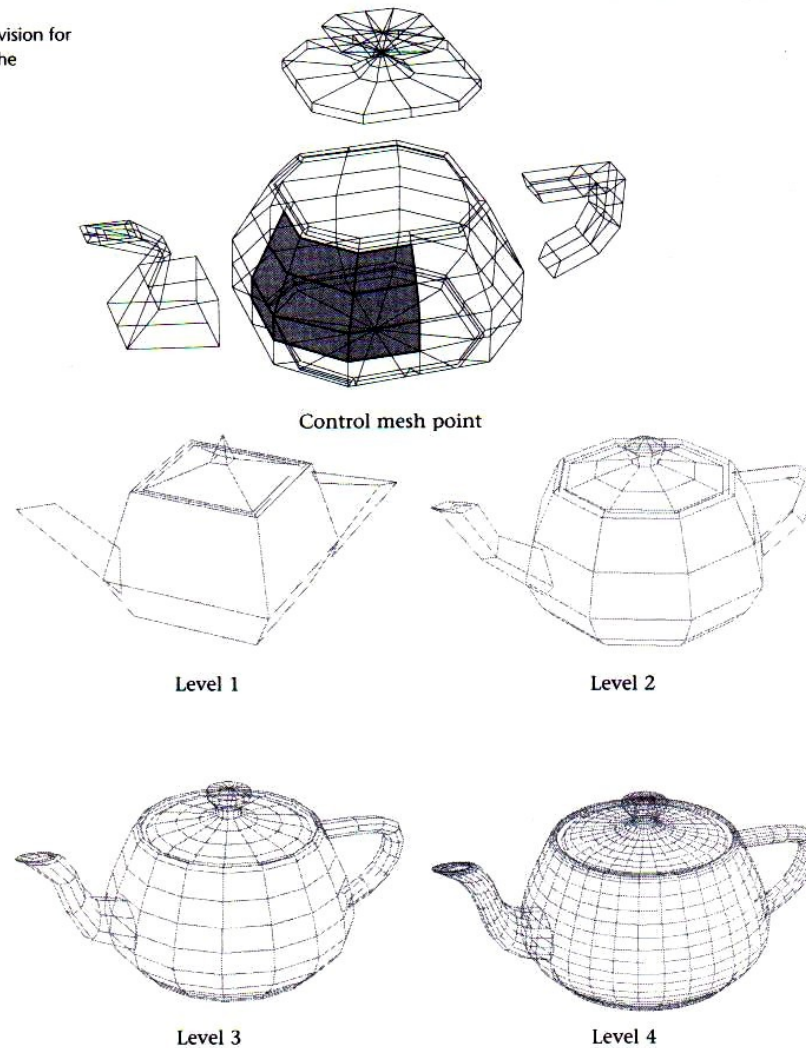
Level 2



Level 3

# Bezier patch subdivision

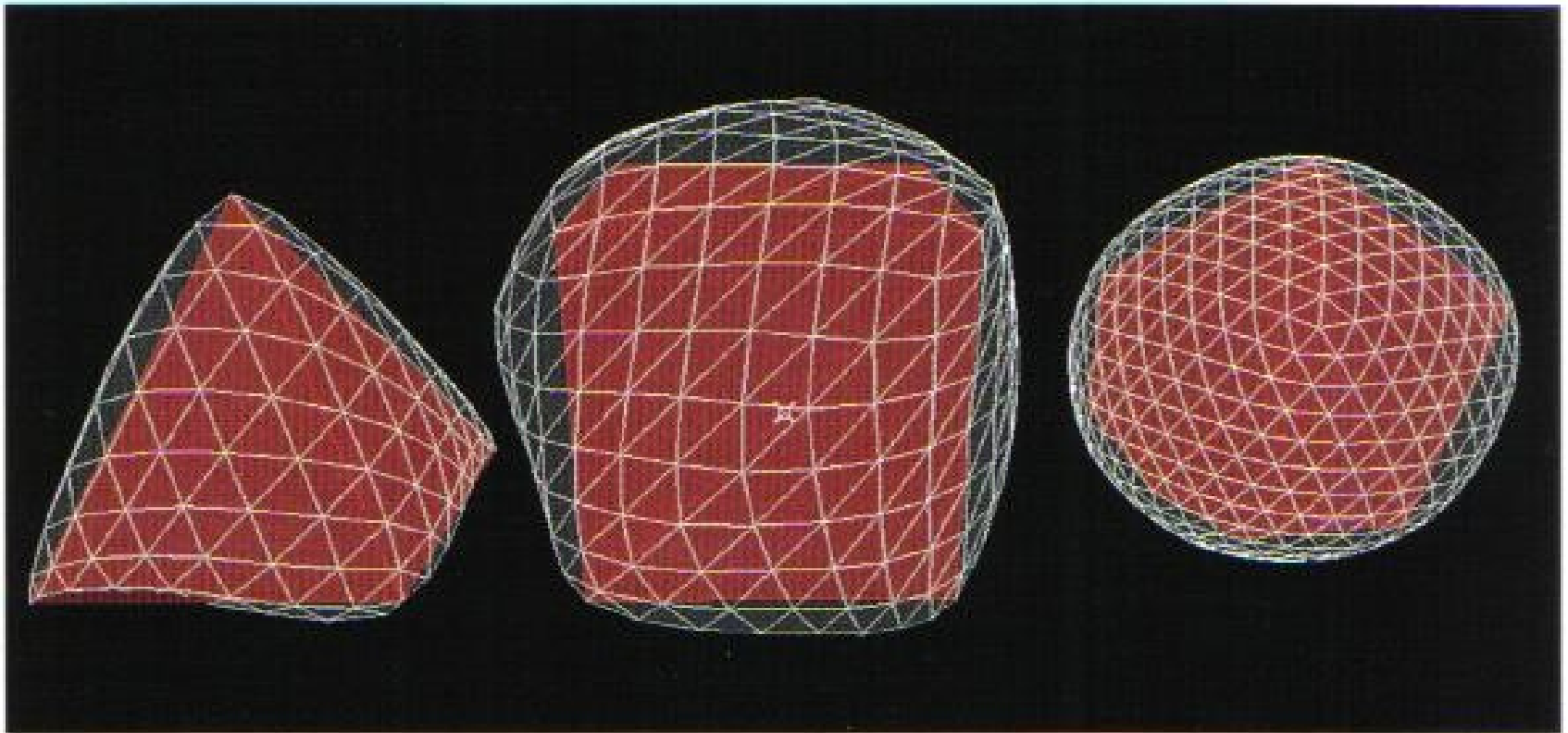
Figure 5.60  
Bézier patch subdivision for  
comparison with the  
previous figure.



# Objects after three levels of butterfly subdivision

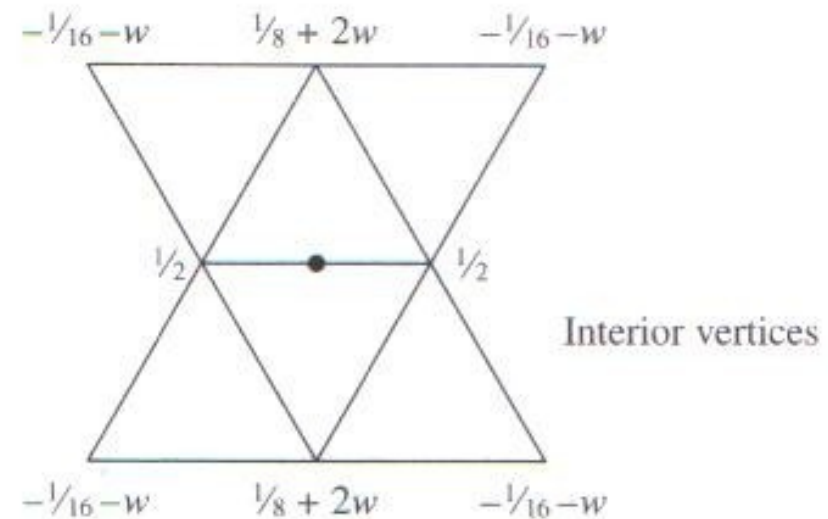
Figure 5.61

Three objects (shown as solid rendered) after three levels of butterfly subdivision.



# The mask for butterfly subdivision

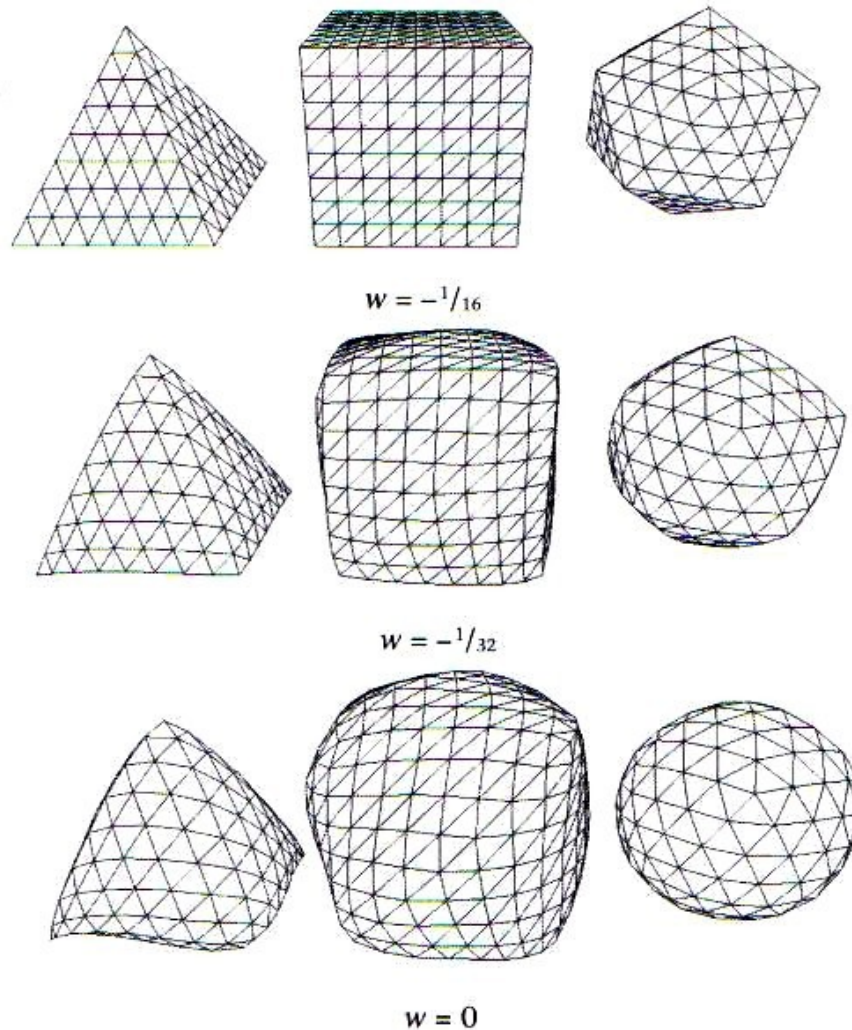
**Figure 5.62**  
Masks for the Butterfly  
subdivision rule.





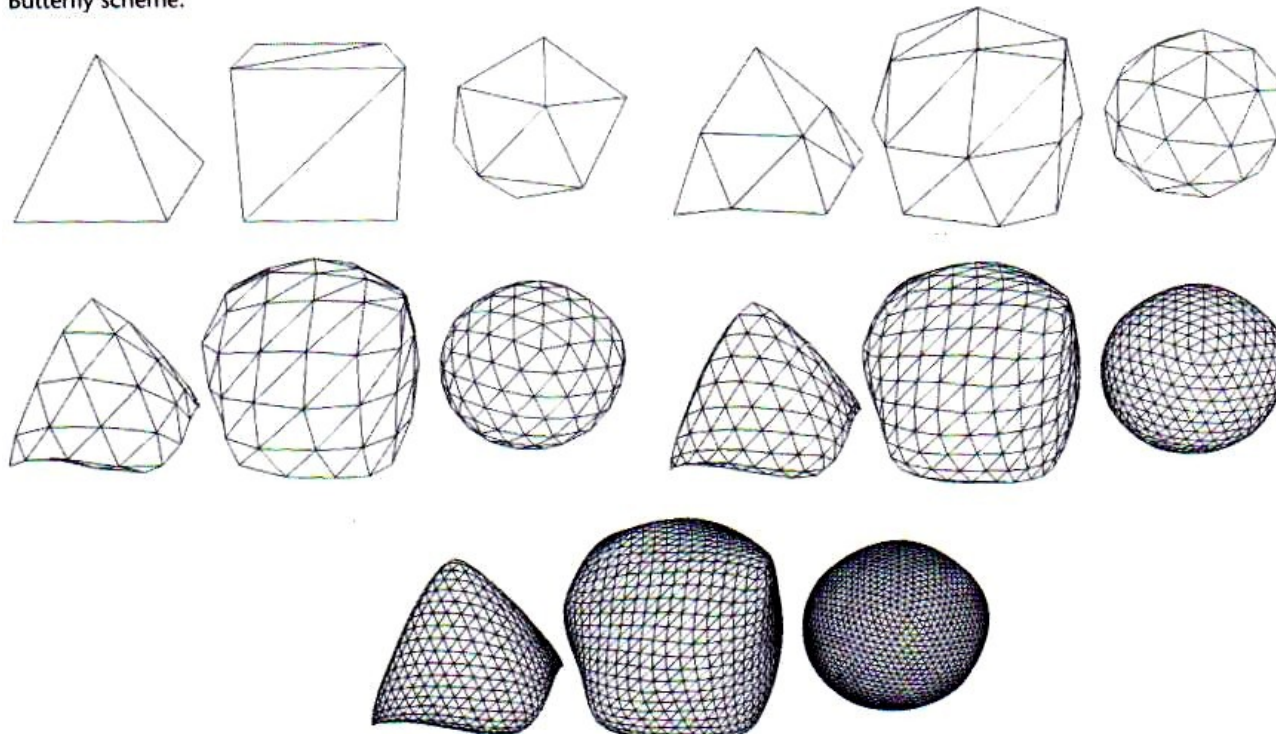
# Varying $w$ in the Butterfly scheme

**Figure 5.63**  
Varying  $w$  in the  
(unmodified) Butterfly  
scheme.



# Four levels of sub-division using the Butterfly scheme

**Figure 5.64**  
Four levels of sub-division  
using the (unmodified)  
Butterfly scheme.





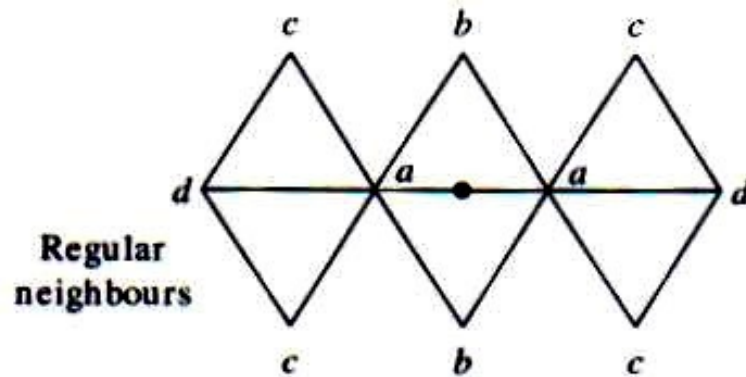
## 5.9.3 Modified butterfly

---

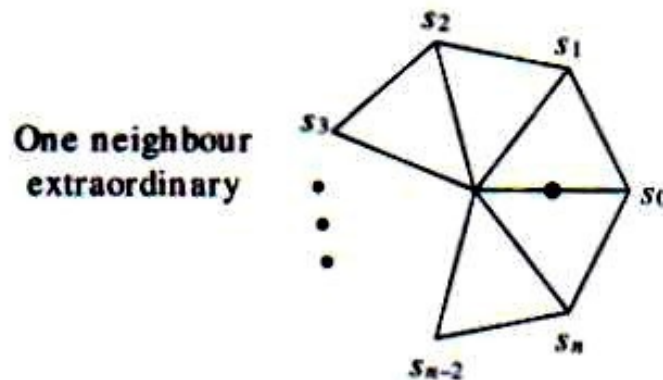
- Zorrin et al. (1996) developed a modification of the butterfly scheme.
- Retained its advantage of **interpolation** and **locality**.
- Dealt with the artefacts that the original scheme introduces.

# Masks for the modified butterfly scheme

**Figure 5.66**  
Masks for the modified butterfly scheme.



$$\begin{aligned} a &= \frac{1}{2} - w \\ b &= \frac{1}{8} + 2w \\ c &= -\frac{1}{16} - w \\ d &= w \end{aligned}$$



$$n \geq 5$$

$$s_j = \frac{\frac{1}{4} + \cos 2\pi_j/n + \frac{1}{2}(\cos 4\pi_j/n)}{n}$$

$$n = 3$$

$$s_0 = \frac{5}{12} \quad s_{1,2} = -\frac{1}{2}$$

$$n = 4$$

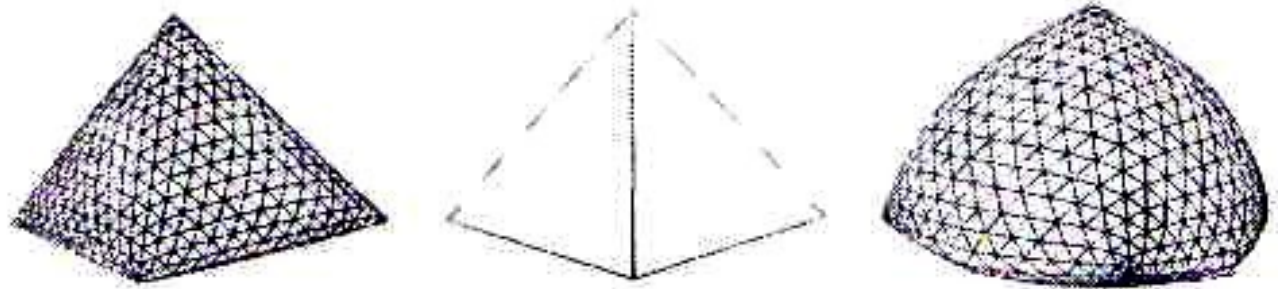
$$s_0 = \frac{3}{8} \quad s_2 = -\frac{1}{8} \quad s_{1,3} = 0$$




# The modified butterfly

---

**Figure 5.67**  
The modified butterfly  
( $w = -1/32$ , original,  $w = 0$ ).





## 5.10 Scalability – polygon meshes, patch meshes and subdivision surfaces

---

5.10.1 LOD polygon meshes

5.10.2 Patch meshes

5.10.3 Subdivision surfaces

## 5.10.1 LOD polygon meshes

---

- The object is modelled to a **degree of accuracy** that depends on the **number of polygons** used. The generality of the representation means that there are no restrictions on the shapes representable and detail is easy to add.
- The database representation of the high-resolution mesh is straightforward.
- Generation of **lower-resolution** meshes in a way that keeps as close as possible to original surface is difficult and is generally **done offline**.
- **Pre-processed** LOD structures are **bi-directional**. From the current level we can go to a **higher** or lower level.

## 5.10.2 Patch meshes

---

- Depending on the object this **can be an exact representation**. There are **difficulties** in **building a mesh** because of **continuity constraints**.
- The database representation is **simple** – a 2D array – and memory/bandwidth requirements are **low**.
- Subdivision is **fast** and, **depending** on the **complexity**, can be processed in **real time**.
- **Non-uniform** subdivision is **possible**, which optimises the representation's scalability, but cracking problems must be dealt with.

## 5.10.3 Subdivision surfaces

---

- The object is **modelled** as a **low-resolution mesh** using **software** that enables the creator to **visualise** the **limit surface**.
- The database representation can be **any subdivision** level from  $M^0$  upwards. The memory/bandwidth requirements are **low**. Although the refinement rules are straightforward, the arbitrary topology makes the data structure manipulations difficult. A **vertex x edge** and edge **vertex map** need to be **maintained**.
- Subdivision is **fast** and, **depending** on the **complexity**, can proceed in **real time**.
- **Only uniform subdivision** is possible.