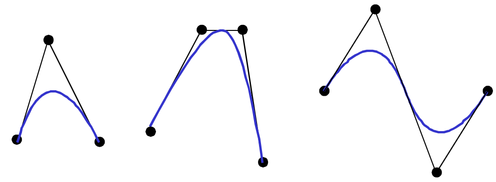
## SPLINES

2D Splines
- Bézier curves
- Spline curves

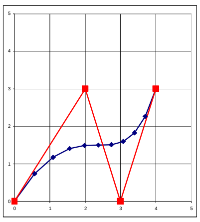
## 2D Splines

- Smooth curves generated from an input set of user-specified control points (knots)



## Raster conversion problem

- Given a set of control points, plot the spline
- A spline curve is usually approximated by a set of short line segments



Control points

Computed points on the spline, inked with short line segments into a smooth curve

## Spline defined by parametric equations

**P(u) = ( x(u), y(u) )**

- u is the parameter
- x(u) and y(u) are functions of the parameter u which generate x and y coordinates of the curve P

## Spline curves can

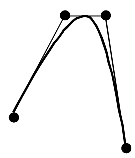
- interpolate control points
- (the curve does not have to pass through them)

or

- approximate control points
- (the curve passes through the points).

## Bézier curves

- Bézier curve is generated by forming a set of polynomial functions formed from the coordinates of the control points



## Bézier curves: equations

Input

- a set of n+1 control points
- $p_k = (x_k, y_k)$, k = 0, n

- Bézier function P(u) in parametric form:

$$P(u) = \sum_{k=0}^{n} p_k B_{kn}(u)$$

$$P(u) = \sum_{k=0}^{n} p_k B_{kn}(u) = p_0 B_{0n}(u) + p_1 B_{1n}(u) + \ldots + p_n B_{nn}(u)$$

---

$$P(u) = \sum_{k=0}^{n} p_k B_{kn}(u)$$

$p_k$ – a control point, $p_k = (x_k, y_k)$

$B_{kn}(u)$ is a blending function (a polynomial):

$$B_{kn}(u) = C(n, k) \cdot u^k \cdot (1 - u)^{n-k}$$

function C( n, k ) provides values for coefficients:

$$C(n, k) = \frac{n!}{k!\,(n-k)!}$$

---

$$P(u) = \sum_{k=0}^{n} p_k B_{kn}(u)$$

- P(u) in a form showing explicit equations for the individual curve coordinates:

$$x(u) = \sum_{k=0}^{n} x_k B_{kn}(u) \qquad y(u) = \sum_{k=0}^{n} y_k B_{kn}(u)$$

$$x(u) = \sum_{k=0}^{n} x_k B_{kn}(u) = x_0 B_{0n}(u) + x_1 B_{1n}(u) + \ldots + x_n B_{nn}(u)$$

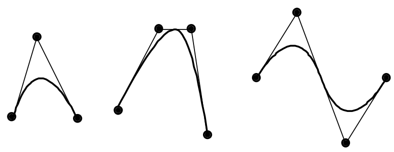
$$y(u) = \sum_{k=0}^{n} y_k B_{kn}(u) = y_0 B_{0n}(u) + y_1 B_{1n}(u) + \ldots + y_n B_{nn}(u)$$

---

$$P(u) = \sum_{k=0}^{n} p_k B_{kn}(u)$$

- The function P(u) is a polynomial of degree n (one less than the number of the control points used)

- For example, 3 points generate parabola, 4 points generate a cubic curve etc.



---

## Bézier curves: example

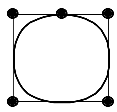
- See a worked-out example of generating a Bézier spline at

http://www.cs.bham.ac.uk/~exc/Teaching/Graphics/ComputingSpline.pdf

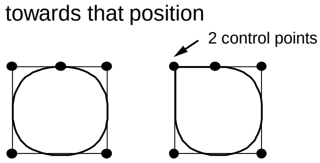
---

## Multiple control points

Condition for closed curves: $p_0 = p_n$



Multiple control points at a same position the curve is "pulled" more towards that position



- In theory Bézier curves can be constructed for any number of points.
- This would lead to polynomials of a very high order which are inefficient to calculate.

## Solution

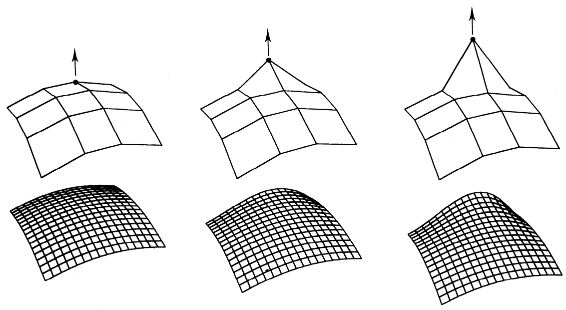
- A set of points is divided into smaller subsets
- Bézier curve specified for each small set
- A very useful property of Bézier curves is that the curve fragments are always smoothly joined

## Spline curves

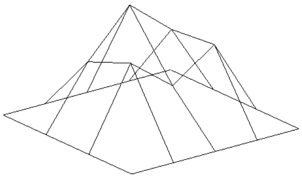
- B-splines are similar to Bézier curves, they differ in the choice of blending functions.
- In a Bézier curve the shape of the curve is influenced by all the control points.
- In B-splines only up to four nearest control points are taken into consideration (i.e. the highest degree of a polynomial is 3, a cubic spline).

## 3D Bezier patches

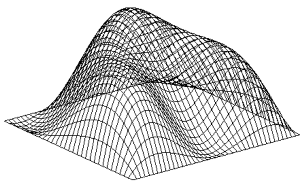
defined on a regular grid



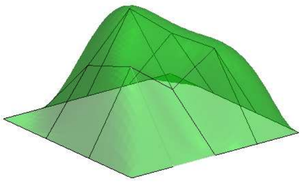
## 3D Bezier patches



random height map on 5x6 grid

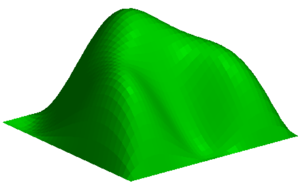


Bezier interpolation (50x60 grid)

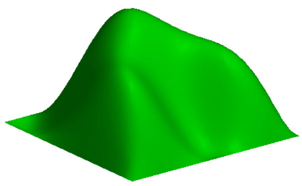


Bezier interpolation (50x60 grid), surface, flat shading

## 3D Bezier patches



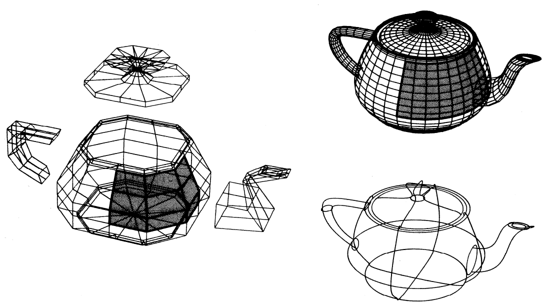
Bezier interpolation surface, flat shading



Bezier interpolation surface, Phong shading

## 3D Bezier patches – Utah teapot

defined at arbitrary positions



## 3D Bezier patches

- Interactive animation at
- http://www.nbb.cornell.edu/neurobio/land/OldStudentProjects/cs490-96to97/anson/BezierPatchApplet/

## Homework

- Given a set of control points
  (0, 6) , (2, -2), (6, 2), (8, 6)
  compute Bezier curve with parameter u changing in steps of 0.2.

- [For adventurous students]
- Given a set of control points
  (0, 6, 3) , (2, -2, -1), (6, 2, 2), (8, 6, -3)
  compute 3D Bezier curve with parameters *u* and *v* changing in steps of 0.2.

## Next lecture

Raster conversion algorithms
Line and circle