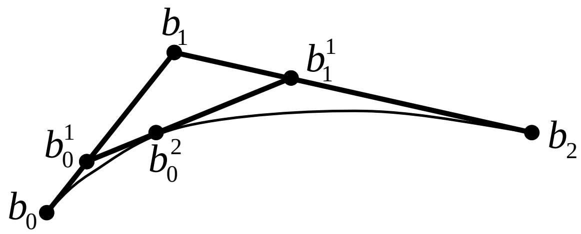
Curves & Surfaces

- So far, we have worked with polygonal objects. How do we represent and manipulate more general surfaces?

- Goals:
  - Compact representation
  - Intuitive control
  - Guaranteed smoothness

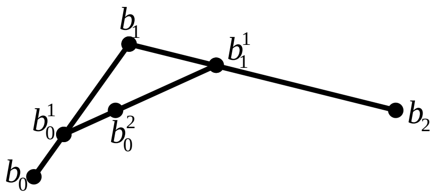
The de Casteljau Algorithm

- Goal: a simple and intuitive mechanism for defining and manipulating the shape of a curve.
- De Casteljau: Given a sequence of control points (a control polygon) define a smooth approximating curve by repeated linear interpolation.



Closed-form expression (3 pts)

$$b_0^1(t) = (1-t)b_0 + tb_1$$
$$b_1^1(t) = (1-t)b_1 + tb_2$$
$$b_0^2(t) = (1-t)b_0^1 + tb_1^1$$



$$C(t) = b_0^2(t) = (1-t)[(1-t)b_0 + tb_1] + t[(1-t)b_1 + tb_2]$$
$$= (1-t)^2 b_0 + 2t(1-t)b_1 + t^2 b_2$$

The curve is a quadratic polynomial in t!

Generalization to n+1 points

- Given control points: $b_0^0, b_1^0, \ldots, b_n^0$
- For any parameter value t, compute:

$$b_i^r(t) = (1-t)b_i^{r-1} + tb_{i+1}^{r-1}, \qquad r = 1, \ldots, n$$

- The curve C(t) is given by $C(t) = b_0^n(t)$

Closed-form expression

- The curve defined by the de Casteljau algorithm is a polynomial of degree n:

$$C(t) = \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} b_i$$

- These curves are commonly known as "Bezier curves"

Bernstein Polynomials

- Bernstein polynomials of degree n:

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} \quad \text{where} \quad \binom{n}{i} = \begin{cases} \frac{n!}{i!(n-i)!} & \text{if } 0 \le i \le n \\ 0 & \text{else} \end{cases}$$

- Bezier curve of degree n (defined by n+1 control points $b_0^0, b_1^0, \ldots, b_n^0$):

$$C(t) = \sum_{i=0}^{n} B_i^n(t)\, b_i$$

Bernstein Polynomials

- Recursive definition:

$$B_i^n(t) = (1-t)\, B_i^{n-1}(t) + t\, B_{i-1}^{n-1}(t)$$

with

$$B_0^0(t) \equiv 1$$

$$B_j^n(t) \equiv 0 \quad for \quad j \notin \{0, \ldots, n\}$$

- Important property (partition of unity):

$$\sum_{j=0}^{n} B_j^n(t) = 1$$

Properties of Bezier Curves

- Affine invariance
- Convex hull property
- Endpoint interpolation
- Symmetry
- Invariance under affine combinations
- Pseudo-local control
- Variation diminishing property

Derivatives

- The derivative of a Bezier curve:

$$\frac{d}{dt}C(t) = n\sum_{j=0}^{n-1}\left(b_{j+1}-b_j\right)B_j^{n-1}(t)$$

- As a result:

$$\left.\frac{dC}{dt}\right|_{t=0} = n\left(b_1 - b_0\right)$$

$$\left.\frac{dC}{dt}\right|_{t=1} = n\left(b_n - b_{n-1}\right)$$

Matrix Notation

- Any polynomial of degree n can be written as a scalar product:

$$a_0 + a_1 t + \cdots + a_n t^n = \left[a_0, a_1, \ldots, a_n\right]\begin{bmatrix}1\\ t\\ \vdots\\ t^n\end{bmatrix}$$

- Example: for Bernstein polynomials of degree 3 we have 4 row vectors of coefficients:

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

Matrix Notation (continued)

- So, we can write a cubic Bezier curve as:

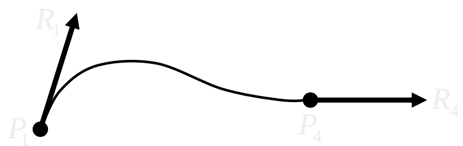
$$C(t) = \begin{bmatrix}b_{0x} & b_{1x} & b_{2x} & b_{3x}\\ b_{0y} & b_{1y} & b_{2y} & b_{3y}\\ b_{0z} & b_{1z} & b_{2z} & b_{3z}\end{bmatrix}\begin{bmatrix}1 & -3 & -3 & -1\\ 0 & 3 & -6 & 3\\ 0 & 0 & 3 & -3\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1\\ t\\ t^2\\ t^3\end{bmatrix}$$

- Or, in general, for degree n:

$$C(t) = \left[\mathbf{b}_0 \quad \mathbf{b}_1 \quad \cdots \quad \mathbf{b}_n\right]M_{\text{Bezier}}\begin{bmatrix}1\\ t\\ \vdots\\ t^n\end{bmatrix}$$

Hermite Curves

- A cubic curve that interpolates two points with given tangent vectors:



- is given by: $C(t) = \left(2t^3 - 3t^2 + 1\right)P_1 + \left(-2t^3 + 3t^2\right)P_4 + \left(t^3 - 2t^2 + t\right)R_1 + \left(t^3 - t^2\right)R_4$

$$C(t) = \left[P_1 \quad P_4 \quad R_1 \quad R_4\right]M_{\text{Hermite}}\begin{bmatrix}1\\ t\\ t^2\\ t^3\end{bmatrix} \quad \text{where} \quad M_{\text{Hermite}} = \begin{bmatrix}1 & 0 & -3 & 2\\ 0 & 0 & 3 & -2\\ 0 & 1 & -2 & 1\\ 0 & 0 & -1 & 1\end{bmatrix}$$

Changing Representations

- Suppose we're given the Hermite control points of a cubic curve and we want to convert them to Bezier control points:

$$C(t) = G_{\text{Bezier}} M_{\text{Bezier}} T \quad = \quad G_{\text{Hermite}}\, M_{\text{Hermite}}\, T$$

$$G_{\text{Bezier}} M_{\text{Bezier}} = G_{\text{Hermite}}\, M_{\text{Hermite}}$$

$$G_{\text{Bezier}} = G_{\text{Hermite}}\, M_{\text{Hermite}}\, M_{\text{Bezier}}^{-1}$$

Piecewise Curves

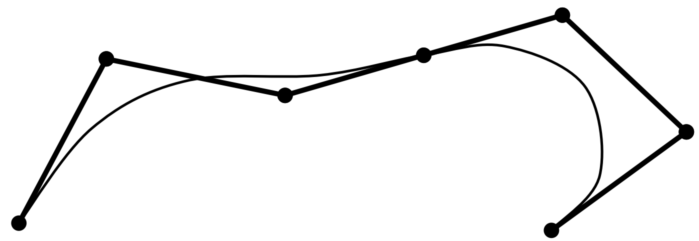
- How do we construct and manipulate complex curves?
  - Use a Bezier curve with many control points → high degree.
  - Construct a complex curve by joining several low-degree curves.
- How do we smoothly connect two Bezier curves?

Curve Smoothness

- Parametric smoothness: a curve is C^k continuous (over a parametric interval [a,b]) if it's continuously differentiable k times for every point (in [a,b]).
- Examples:
  - C^{-1} curves are discontinuous
  - C^0 curves: continuous, but not smooth
  - C^1 curves: smooth (continous 1st derivative)
  - C^2 curves: smoother
  - Etc.

Piecewise Cubic Bezier Curve

- A sequence of cubic Bezier curves, joined together such that the curve and the 1st derivative are continuous: the curve is C^1!



Tensor-Product Surfaces

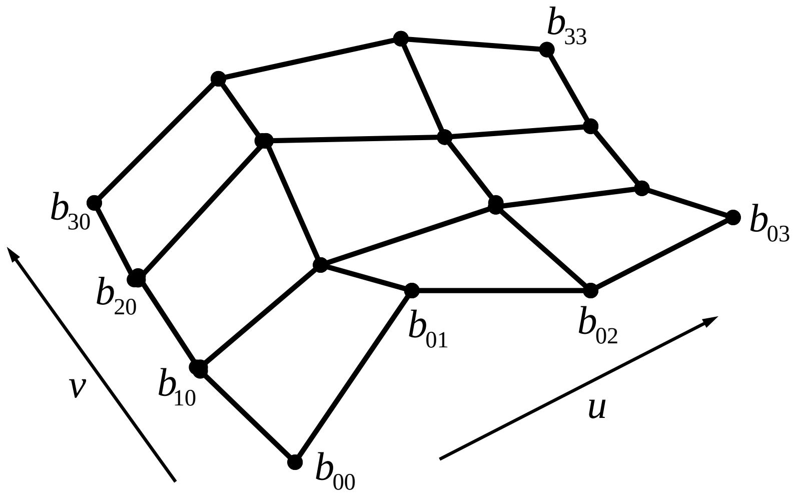
- Parametric surfaces:

$$S(u,v):[0,1]^2 \rightarrow \mathrm{E}^3 \qquad S(u,v)=\begin{cases} S_x(u,v) \\ S_y(u,v) \\ S_z(u,v) \end{cases}$$

- The Bernstein-Bezier "tensor product" surfaces:

$$S(u,v)=\sum_{i=0}^{n}\sum_{j=0}^{m} B_i^n(u)\, B_j^m(v)\, b_{ji}$$

Control Mesh



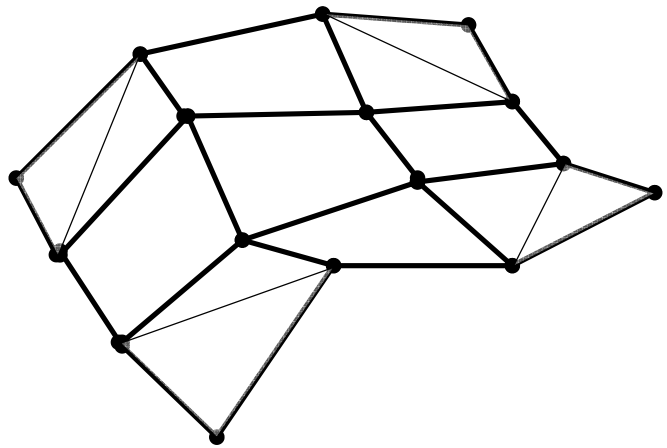
Properties of Bezier Surfaces

- Most properties of Bezier curves still hold:
  - Affine invariance
  - Convex hull property
  - Corner interpolation
- Boundaries are Bezier curves.
- Corner derivatives:

$$\left.\frac{\partial S(u,v)}{\partial u}\right|_{(0,0)} = n\left(b_{01}-b_{00}\right)$$

$$\left.\frac{\partial S(u,v)}{\partial v}\right|_{(0,0)} = m\left(b_{10}-b_{00}\right)$$

Corner Tangent Planes



Connecting Patches Smoothly

- To ensure continuity (C^0) across boundary, the boundary control points must coincide.
- How do we ensure C1 continuity across boundaries?