Advanced Modeling 2
Katja Bühler, Andrej Varchola, Eduard Gröller
Institute of Computer Graphics and Algorithms
Vienna University of Technology
Parametric curves and surfaces
Polynomial curves
Rational curves
Tensor product surfaces
Triangular surfaces
Parametric Curves

\[ c: \quad c(u) = \begin{pmatrix} x(u) \\ y(u) \\ z(u) \end{pmatrix}; \quad u \in [a,b] = :I \subset IR \]

- \( x(u), y(u), z(u) \) are differentiable functions in \( u \)
- Tangent vector: \( t(u) = \frac{d}{du} c(u) \)
- \( c \) regular in \( c(u_0) \) \( \iff \) \( t(u_0) \neq 0 \)
- \( c \) regular \( \iff \) \( c \) can be parametrized in a way that all curve points are regular.
Parametric Surfaces

\[ s : \mathbf{s}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix}; \quad (u,v) \in [a,b] \times [c,d] =: I \times J \subseteq \mathbb{R}^2 \]

- \( x(u,v), \ y(u,v), \ z(u,v) \) are differentiable functions in \( u \) and \( v \)
- Tangent plane: \( \mathbf{t}_0(l,m) = \mathbf{s}(u_0,v_0) + l\mathbf{s}_u(u_0,v_0) + m\mathbf{s}_v(u_0,v_0) \)
- Normal vector: \( \mathbf{n}(u_0,v_0) = \mathbf{s}_u(u_0,v_0) \times \mathbf{s}_v(u_0,v_0) \)
- \( s \) regular in \( s(u_0,v_0) \iff \mathbf{n}(u_0,v_0) \neq \mathbf{0} \)
- \( s \) regular \iff \( s \) can be parametrized in a way that all surface points are regular.
Bézier Curves: The de Casteljau Algorithm

Given \( n + 1 \) points \( b_0, \ldots, b_n \in IE^3 \) and an arbitrary \( t \in IR \).

Set

\[
\begin{align*}
b_i^r &= (1-t) b_{i-1}^r + t b_{i+1}^r \quad \text{with} \quad b_0^0 &= b_0. 
\end{align*}
\]

Then \( b_n \) is a curve point on the corresponding Bézier curve.

The points \( b_{n-1} \) and \( b_{n-1} \) determine the tangent line of the curve at point \( b_0 \).

The de Casteljau scheme
Bézier curve with respect to Bézier points \( b_i, \ i = 0, \ldots, n \):

\[
b(t) = \sum_{i=0}^{n} b_i B_i^n(t)
\]

Bernstein polynomial s:

\[
B_i^n(t) = \binom{n}{i} (1-t)^{n-i} t^i
\]
Properties of Bézier Curves

- Affine invariance
- Convex hull property
- Endpoint interpolation
- Linear precision
- Variation diminishing property

- Disadvantages
  - Only pseudo local control
  - High degree
Important Algorithms for Bézier Curves

- **Degree elevation**
  - to increase flexibility

- **Subdivision**
  - to increase flexibility
  - to approximate the curve

The subdivision is a byproduct of the de Casteljau algorithm
The de Casteljau algorithm is numerical stabil, but inefficient for evaluation.

Horner scheme like evaluation is more efficient.

\[ b(t) = \left( \ldots (((n)_{s}b_{0} + (n)_{1}t b_{1})s + (n)_{2}t^{2}b_{2})s + \ldots )s + (n)_{n}t^{n}b_{n} \right) \text{ with } s = 1-t \]

Repeated subdivision gives in a fast way a good approximation of the curve.
B-Spline Curves

- B-Spline curves
- are piecewise polynomial curves of degree $k-1$
- have a degree (almost) independent of the number of control points
- allow local control over the shape of a curve
B-Spline Curves: Definition

Given:

- \( n + 1 \) control points \( \mathbf{d}_i \in \mathbb{R}^3 \), \( i = 0, \ldots, n \)
- knot vector \( U = (u_0, \ldots, u_{n+k}) \)

B-spline curve:

\[
s(u) = \sum_{i=0}^{n} \mathbf{d}_i N_i^k(u), \quad u \in [u_0, u_{n+k}]
\]

with the B-spline basis functions \( N_i^k(u) \) of order \( k \).
B-Spline Basis Functions

\[ k = 0 : \quad N_i^0(u) = \begin{cases} 
1 & u \in [u_i, u_{i+1}) \\
0 & \text{sonst} 
\end{cases} \]

\[ k > 0 : \quad N_i^k(u) = \frac{u - u_i}{u_{i+k-1} - u_i} N_i^{k-1}(u) + \frac{u_{i+k} - u}{u_{i+k} - u_{i+1}} N_{i+1}^{k-1}(u) \]

Properties of the basis functions:

- partition of unity: \( \sum_{i=0}^{n} N_i^k(u) \equiv 1 \)
- positivity: \( N_i^k(u) \geq 0 \)
- local support: \( N_i^k(u) = 0 \quad \text{if} \quad u \not\in [u_i, u_{i+k}) \)
Properties of B-Spline Curves

- Affine invariance
- Strong convex hull property
- Variation diminishing property
- Local support
- Knot points of multiplicity $k$ are coincident with one of the control points.
- A B-Spline curve of order $k$ which has only knots of multiplicity $k$ is a Bézier curve
Evaluating B-Spline Curves: The de Boor Algorithm

Given

\[ n + 1 \text{ control points } \mathbf{d}_0, \ldots, \mathbf{d}_n \in \mathbb{R}^3, \text{ a knot vector } U = (u_0, \ldots, u_{n+k}) \]

and an arbitrary \( t \in [u_0, u_{n+k}) \).

Set

\[ \mathbf{d}_i^r := (1 - \alpha_i^r) \mathbf{d}_{i-1}^{r-1} + \alpha_i^r \mathbf{d}_i^{r-1} \]

with \( \mathbf{d}_i^0 := \mathbf{d}_i \) and \( \alpha_i^r := \frac{t - u_i}{u_{i+k-r} - u_i} \)

Then \( \mathbf{d}_m^{k-1} \) is the point for \( x(t_0), t_0 \in [t_m, t_{m+1}] \) on the corresponding B-spline curve.

The points \( \mathbf{d}_{m-1}^{k-2} \) and \( \mathbf{d}_m^{k-2} \) determine the tangent line of the curve at point \( \mathbf{b}_m^{k-1} \).
Find the knot span \([u_i, u_{i+1})\) in which the parameter value \(t\) lies

Compute all non zero basis functions

Multiply the values of the nonzero basis functions with the corresponding control points
Special B-Spline Curves

- **open:**
  \[ u_0 = \ldots = u_k < u_{k+1} < \ldots < u_n = \ldots = u_{n+k} \]

- **closed:**
  \[ d_{n+1} := d_0, \ldots, d_{n+k} := d_{k-1} \quad \text{and} \quad U = (u_0, \ldots, u_{n+k}, \ldots, u_{n+2k-2}) \]

- **uniform:**
  \[ U = (u_0, u_0 + d, \ldots, u_0 + (n + k)d) \]
Important Algorithms for B-Spline Curves

- Knot insertion
  - to increase flexibility
  - to compute derivatives
  - to split curves (subdivision algorithms)
  - to evaluate the curve (see de Boor algorithm)
  - to approximate the curve

- Degree elevation
  - to adapt curve degrees
Rational Curves

Rational curve:

\[ \mathbf{c}(u) = \frac{1}{w(u)} \begin{pmatrix} x(u) \\ y(u) \\ z(u) \end{pmatrix}, \quad u \in I \subset IR \]

Homogeneous representation:

\[ \mathbf{c}_H(u) = \begin{pmatrix} w(u) \\ x(u) \\ y(u) \\ z(u) \end{pmatrix}, \quad u \in I \subset IR \]

Example: conic section

\[ \mathbf{c}(u) = \frac{1}{1+u^2} \begin{pmatrix} a (1-u^2) \\ b 2u \end{pmatrix} \]

\[ \mathbf{c}_H(u) = \begin{pmatrix} 1+u^2 \\ a (1-u^2) \\ b 2u \end{pmatrix} \]
A rational Bézier curve is defined as

\[ b(u) = \frac{\sum_{i=0}^{n} w_i b_i B^n_i(u)}{\sum_{i=0}^{n} w_i B_i^n(u)} \], \quad u \in I \subset IR

The \( w_i > 0, \ i = 0, \ldots, n \) are called weights.

Homogeneous representation:

\[ b_H(u) = \sum_{i=0}^{n} b_{H_i} B^n_i(u), \quad u \in I \subset IR \]

with the homogeneous Bézier points

\[ b_{H_i} = \begin{pmatrix} w_i \\ w_i b_i \end{pmatrix} \]
Properties:
- the same properties like polynomial curve, and
- projective invariance
- the weights are an additional design parameter

Algorithms
- All algorithms of polynomial Bézier curves can be applied without any change to the homogeneous representation of rational Bézier curves.
A NURBS curve with respect to the control points \( d_i, i = 0, ..., n \) and the knot vector \( U = (u_0, ..., u_{n+k}) \) is defined as

\[
\mathbf{n}(u) = \frac{\sum_{i=0}^{n} w_i d_i N_i^k (u)}{\sum_{i=0}^{n} w_i N_i^k (u)}, \quad u \in [u_0, u_{n+k}] \subset \mathbb{R}
\]

The \( w_i > 0, i = 0, ..., n \) are called weights.

Homogeneous representation:

\[
\mathbf{n}_H (u) = \sum_{i=0}^{n} \mathbf{n}_{Hi} \mathbf{N}_i^k (u), \quad u \in [u_0, u_{n+k}] \subset \mathbb{R}
\]

with the homogeneous B-Spline points

\[
\mathbf{n}_{Hi} = \begin{pmatrix} w_i \\ w_i \mathbf{n}_i \end{pmatrix}
\]
Properties:
- the same properties like polynomial curve, and
- projective invariance
- changing the weight \( w_i \) affects only the interval \([u_i, u_{i+k})\)

Algorithms
- All algorithms of B-spline curves can be applied without any change to the homogeneous representation of NURBS curves.
"A surface is the locus of a curve that is moving through space and thereby changing the shape"

Given a curve

$$f(u) = \sum_{i=0}^{n} c_i F_i(u), \quad u \in I \subset \mathbb{IR}$$

moving the control points yields

$$c_i(v) = \sum_{j=0}^{m} a_{ij} G_j(v), \quad v \in J \subset \mathbb{IR}$$

Combining both yields a tensor-product surface

$$s(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} a_{ij} F_i(u) G_j(v), \quad (u, v) \in I \times J \subset \mathbb{IR}^2$$
A tensor-product Bézier surface is given by

\[ b(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} b_{ij} B_i^n(u) B_j^m(v), \quad (u, v) \in I \times J \subset IR^2 \]

The Bézier points \( b_{ij} \) form the control net of the surface.
Tensor-Product Bézier Surfaces

- Properties:
  analogue to that of Bézier curves

- Algorithms:
  Apply algorithms for curves in two steps:
  - Apply to
    \[ b_i(v) = \sum_{j=0}^{m} b_{ij} B_j^m(v), \quad i = 0, \ldots, n \]
  - Apply to
    \[ b(u, v) = \sum_{i=0}^{n} b_i(v) B_i^n(u) \]
A tensor product B-spline surface with respect to the knot vectors

\[ \mathbf{U} = (u_0, \ldots, u_{n+k}), \mathbf{V} = (v_0, \ldots, v_{m+l}) \]

is given by

\[ \mathbf{d}(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} d_{ij} N_i^k(u) N_j^l(v), \quad (u, v) \in [u_0, u_{n+k}) \times [v_0, v_{m+l}) \subset \mathbb{R}^2. \]

The control points \( \mathbf{d}_{ij} \) form the control net of the surface.

Properties and Algorithms are analogue to the description for Bézier tensor product surfaces.
A triangular Bézier patch is defined by
\[ b(u, v, w) = \sum_{i+j+k=n} b_{ijk} B_{ijk}^n (u, v, w) \]

The \( u, v, w \) are barycentric coordinates of the triangular parameter domain.

Generalized Bernstein polynomials:
\[ B_{ijk}^n (u, v, w) = \frac{n!}{i! j! k!} u^i v^j w^k \]
Bézier Triangles: Properties and Algorithms

- **Properties:**
  - the same as in the univariate case

- **Algorithms:**
  - **De Casteljau:**
    \[
    b_{ijk}^l = ub_{i+1jk}^{l-1} + vb_{ij+1k}^{l-1} + wb_{ijk+1}^{l-1}
    \]
  - **Subdivision**
  - **Degree elevation**
Subdivision Surfaces

- Polygon-mesh surfaces generated from a base mesh through an iterative process that smoothes the mesh while increasing its density.
- Represented as functions defined on a parametric domain with values in $\mathbb{R}^3$.
- Allow to use the initial control mesh as the domain.
- Developed for the purpose of CG and animation.
Subdivision Surfaces: The Basic Idea

- In each iteration
  - Refine the initial control mesh
  - Increase the number of vertices / faces
- The mesh vertices converge to a limit surface
Loop’s Scheme (‘87)

- Old vertex
- Edge vertex
- New vertex

\[ \beta = \frac{1}{n} \left( \frac{5}{8} - \left( \frac{3}{8} + \frac{2}{8} \cos \left( \frac{2\pi}{n} \right) \right)^2 \right) \]

http://www.cs.technion.ac.il/~cs236716/
Catmull-Clark Scheme ’78

- **Face vertex**
  \[ v_f = \frac{1}{4} \sum_{i=1}^{4} v_i \]

- **Edge vertex**
  \[ v_e = \frac{v_1 + v_2 + v_{f1} + v_{f2}}{4} \]

- **Vertex**
  \[ v = \frac{Q}{n} + \frac{2R}{n} + \frac{p(n-3)}{n} \]

- **Q** – Average of face vertices
- **R** – Average of edge vertices
- **v** – new vertex

http://www.cs.technion.ac.il/~cs236716/
Comparison of the Loop and the Catmull-Clark Scheme

Loop subdivision scheme:

Catmull-Clark subdivision scheme:

http://www.holmes3d.net/graphics/subdivision/
Subdivision Surfaces: Classification

- The type of refinement rule
  - Vertex insertion
  - Corner cutting

- The type of generated mesh
  - Triangular
  - Quadrilateral

- Approximating vs. Interpolating
Subdivision Surfaces: Comparison

Catmull-Clark  Doo-Sabin

Loop  Butterfly

Catmull-Clark  Doo-Sabin

Loop  Butterfly

[http://www.cs.technion.ac.il/~cs236716/]