Topics

- Parametric curves and surfaces
- Polynomial curves
- Rational curves
- Tensor product surfaces
- Triangular surfaces

Parametric Curves

- \( c(u) = \begin{cases} x(u) \\ y(u) \\ z(u) \end{cases} ; u \in [a,b] = J \subset IR \)
- \( x(u), y(u), z(u) \) are differentiable functions in \( u \)
- Tangent vector : \( t(u) = u \frac{d}{du} c(u) \)
- \( c \) regular in \( c(u) \) \( \iff \) \( t(u_0) \neq \mathbf{0} \)
- \( c \) regular \( \iff \) \( c \) can be parametrized in a way that all curve points are regular.

Parametric Surfaces

\( s(u,v) = \begin{cases} x(u,v) \\ y(u,v) \\ z(u,v) \end{cases} ; (u,v) \in [a,b] \times [c,d] = J \times L \subset IR^2 \)
- \( x(u,v), y(u,v), z(u,v) \) are differentiable functions in \( u \) and \( v \)
- Tangent plane : \( t(L_m) = s(u_{0,m}) + t s(u_{0,m}) + m s(u_{0,m}) \)
- Normal vector : \( n(u_{0,m}) = s(u_{0,m}) \times s(u_{0,m}) \)
- \( s \) regular in \( s(u_{0,m}) \) \( \iff \) \( n(u_{0,m}) \neq \mathbf{0} \)
- \( s \) 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 \( b_0, ..., b_n \in IR^3 \) and an arbitrary \( t \in IR \).

Set \( b'_i := (1-t) b_{i+1} + t b'_i \) with \( b'_n := b_n \).

Then \( b'_i \) is a curve point on the corresponding Bézier curve.

The points \( b_{i-1}' \) and \( b_{i+1}' \) determine the tangent line of the curve at point \( b_i' \).

Bézier Curves and Bernstein Polynomials

Bézier curve with respect to Bézier points \( b_i, i = 0, ..., n \) :

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

Bernstein polynomial \( s \) :

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

Evaluation/Approximation of Bézier Curves

- The de Casteljau algorithm is numerical
  stabil, but inefficient for evaluation
- Horner scheme like evaluation is more efficient
  \[ b(t) = (\ldots((-\binom{n}{0}a_0 + \binom{n}{1}a_1)t + \binom{n}{2}a_2)t + \ldots)t + \binom{n}{n}a_n \text{ with } t = 1-t \]
- 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 \( d_i \in IE^3, i = 0,...,n \)
- knot vector \( U = (u_0,...,u_{n+1}) \)

B - spline curve :
\[ s(u) = \sum_{i=0}^{n} d_i N^k_i(u), \quad u \in [u_i, u_{i+1}) \]
with the B - spline basis functions \( N^k_i(u) \) of order \( k \)

B-Spline Basis Functions

\[ k = 0: \quad N^0_i(u) = \begin{cases} 1 & u \in [u_i, u_{i+1}) \\ 0 & \text{sonst} \end{cases} \]
\[ k > 0: \quad N^k_i(u) = \frac{u-u_i}{u_{i+k}-u_i} N^{k-1}_i(u) + \frac{u_{i+k}-u}{u_{i+k}-u_{i+1}} N^{k-1}_{i+1}(u) \]

Properties of the basis functions :
- partition of unity : \( \sum_{i=0}^{n} N^k_i(u) = 1 \)
- positivity : \( N^k_i(u) \geq 0 \)
- local support : \( N^k_i(u) = 0 \) if \( u \notin [u_i, u_{i+1}) \)
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

Direct Evaluation of B-Spline Curves

- 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

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

Evaluating B-Spline Curves: The de Boor Algorithm

Given
\[ n + 1 \text{ control points } d_0, \ldots, d_n \in \mathbb{R}^3, \text{ a knot vector } U = (u_0, \ldots, u_m) \]
and an arbitrary \( t \in (u_i, u_{i+1}) \).
Set
\[ d_i^t := (1 - a_i') d_i + a_i' d_{i+1} \]
with \( d_i' = 1 - a_i' \) and \( a_i' = \frac{t - u_i}{u_{i+1} - u_i} \).
Then \( d_i^t \) is the point for \( x(t), y(t), z(t) \) on the corresponding B-spline curve.

The points \( d_i^t \) and \( d_{i+1}^t \) determine the tangent line of the curve at point \( \nu_i^{t} \).

Special B-Spline Curves

- open:
  \[ U = (u_0, \ldots, u_{n+1}) \]
- closed:
  \[ d_{n+1} := d_0, \ldots, d_{n+1} := d_1 \]
  \[ U = (u_0, \ldots, u_n, u_{n+k-1}) \]
- uniform:
  \[ U = (u_0, u_0 + d, \ldots, u_0 + (n+k)d) \]

Rational Curves

Rational curve:
\[ c(u) = \begin{pmatrix} x(u) \\ y(u) \\ z(u) \end{pmatrix}, \quad u \in I \subset \mathbb{R} \]

Example: conic section

Homogeneous representation:
\[ c(u) = \begin{pmatrix} a(1-u^2) \\ b(1-u^2) \end{pmatrix}, \quad u \in I \subset \mathbb{R} \]
\[ c(u) = \begin{pmatrix} a(1-u^2) \\ b(1-u^2) \end{pmatrix}, \quad u \in I \subset \mathbb{R} \]
Rational Bézier Curves

A rational Bézier curve is defined as
\[ \mathbf{b}(u) = \frac{\sum_{i=0}^{n} w_i \mathbf{B}_i^r(u)}{\sum_{i=0}^{n} w_i \mathbf{B}_i^r(u)}, \quad u \in I \subset \mathbb{R} \]
The \( w_i > 0, \) \( i = 0, \ldots, n \) are called weights.

Homogeneous representation:
\[ \mathbf{b}_{ij}(u) = \frac{\sum_{i=0}^{n} w_i \mathbf{B}_i^r(u)}{\sum_{i=0}^{n} w_i \mathbf{B}_i^r(u)}, \quad u \in I \subset \mathbb{R} \]
with the homogeneous Bézier points
\[ \mathbf{b}_{ij} = \left( \begin{array}{c} w_i \\ w_i \mathbf{b}_i \end{array} \right) \]

Non Uniform Rational B-Splines

A NURBS curve with respect to the control points \( d_i, \) \( i = 0, \ldots, n \)
and the knot vector \( U = (u_0, \ldots, u_{n+k}) \) is defined as
\[ \mathbf{n}(u) = \frac{\sum_{i=0}^{n} w_i \mathbf{d}_i N_i^k(u)}{\sum_{i=0}^{n} w_i N_i^k(u)}, \quad u \in [u_i, u_{i+k}) \subset \mathbb{R} \]
The \( w_i > 0, \) \( i = 0, \ldots, n \) are called weights.

Homogeneous representation:
\[ \mathbf{n}_{ij}(u) = \frac{\sum_{i=0}^{n} w_i \mathbf{d}_i N_i^k(u)}{\sum_{i=0}^{n} w_i N_i^k(u)}, \quad u \in [u_i, u_{i+k}) \subset \mathbb{R} \]
with the homogeneous B-Spline points
\[ \mathbf{n}_{ij} = \left( \begin{array}{c} w_i \\ w_i \mathbf{d}_i \end{array} \right) \]

Tensor-Product Surfaces

*A surface is the locus of a curve that is moving through space and thereby changing the shape*

Given a curve
\[ f(u) = \sum_{i=0}^{m} \mathbf{c}_i F_i(u), \quad u \in I \subset \mathbb{R} \]
moving the control points yields
\[ \mathbf{c}(v) = \sum_{j=0}^{n} \mathbf{a}_j G_j(v), \quad v \in J \subset \mathbb{R} \]
Combining both yields a tensor-product surface
\[ s(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{c}_i F_i(u) G_j(v), \quad (u,v) \in I \times J \subset \mathbb{R}^2 \]

Tensor-Product Bézier Surfaces

A tensor-product Bézier surface is given by
\[ \mathbf{b}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{b}_{ij} B_i^r(u) B_j^r(v), \quad (u,v) \in I \times J \subset \mathbb{R}^2 \]
The Bézier points \( \mathbf{b}_{ij} \) 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 $b_i(v) = \sum b_j B_j^r(v)$, $i = 0,...,n$
    - Apply to $b(u,v) = \sum b_i B_i^r(u)B_j^s(v)$

Tensor-Product B-Spline Surfaces

A tensor product B-spline surface with respect to the knot vectors $U = (u_0, u_1, ..., u_n)$, $V = (v_0, v_{m-1})$ is given by
$$d(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} d_{ij} B_i^r(u)B_j^s(v), \quad (u,v) \in [u_0, u_n) \times [v_0, v_m) \subset IR^2.$$ The control points $d_{ij}$ form the control net of the surface.

Properties and Algorithms are analogous to the description for Bézier tensor product surfaces.

Bézier Triangles

A triangular Bézier patch is defined by
$$b(u,v,w) = \sum_{i,j,k=0}^{n} b_{ijk} B_i^r(u)B_j^s(v)B_k^t(w).$$

The $u,v,w$ are barycentric coordinates of the triangular parameter domain.

Generalized Bernstein polynomials:
$$B_{ijk}^r(u,v,w) = \frac{n!}{i!j!k!} u^i v^j w^k.$$ In each iteration
- Refine the initial control mesh
- Increase the number of vertices / faces
- The mesh vertices converge to a limit surface

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 $R^3$
- Allow to use the initial control mesh as the domain
- Developed for the purpose of CG and animation
Loop’s Scheme ('87)

- Old vertex
- Edge vertex
- New vertex

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

Catmull-Clark Scheme '78

- Face vertex
  \[ v_f = \frac{1}{4} \sum_{j=1}^{5} v_j \]
- Edge vertex
  \[ v_e = \frac{v_1 + v_2 + v_{j1} + v_{j2}}{4} \]
- Vertex
  \[ v = \frac{Q}{n} + \frac{2R}{n} + \frac{P(n-3)}{n} \]

Comparison of the Loop and the Catmull-Clark Scheme

Loop subdivision scheme:

Catmull-Clark subdivision scheme:

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