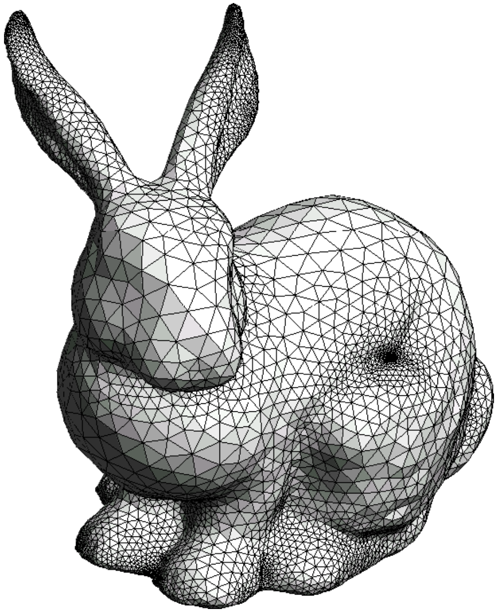


Geometric Modeling in Graphics

Part 7: Surfaces

Martin Samuelčík

www.sccg.sk/~samuelcik

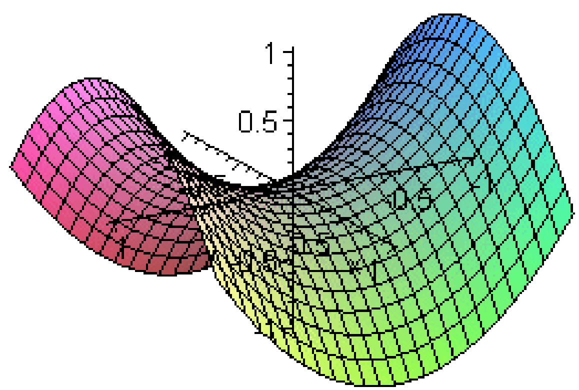
samuelcik@sccg.sk

Surface

- 2D set of points, embedded in space E^3
- f: $\mathbf{R}^2 \rightarrow E^3$
- Parametric surfaces
  - Set of all points $X \in E^3$ such that $X = f(u,v)$, $u \in <u_0,u_1>, v \in <v_0,v_1>$
  - Plane: $f(u,v)= S + uD_1 + vD_2$
  - Sphere: $f(u,v)=(r.\cos(u).\cos(v), r.\cos(u).\sin(v), r.\sin(v))$, $u \in <0,2\pi>, v \in <0, \pi>$
- Implicit surfaces
  - Set of all points $X \in \mathbf{E}^3$ such that $f(X)=0$
  - Plane: $ax+by+cz+d = 0$
  - Sphere: $(x-s_x)^2 +(y-s_y)^2 + (z-s_z)^2 - r^2 = 0$

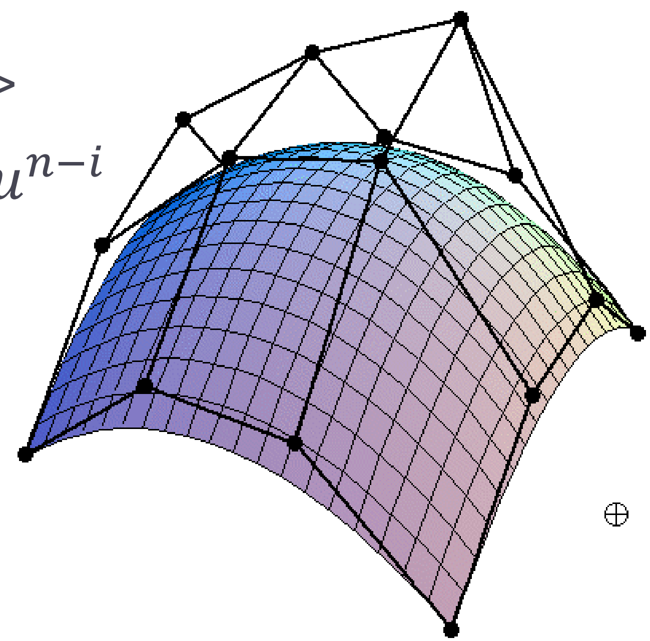
Parametric surface

- Two parameters in surface function
- Similar properties, algorithms like in curve case – putting one parameter constant leads to isocurve
- Visualization
  - Sampling domain using 2D grid points
  - Computing surface points using sampled points and f
  - Connecting surface points based on domain grid connections and forming triangle or quad mesh
  - Uniform sampling
  - Adaptive sampling
  - Raytracing



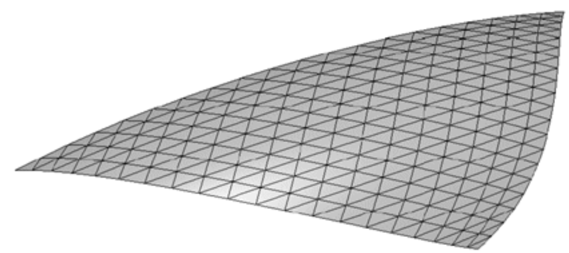
Polynomial surface

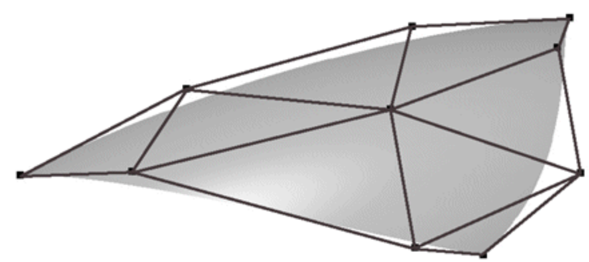
- f is polynomial function in both parameters
- Monomial basis
  - $f(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} V_{ij} u^i v^j$
- Bezier surface
  - $f(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} V_{ij} B^n{}_i(u) B^m{}_j(v)$
  - Square domain: $u \in < 0,1 >, v \in < 0,1 >$
  - Bernstein basis: $B^n{}_i(u) = \binom{n}{i} (1-u)^i u^{n-i}$
  - Tensor product surface
  - Approximation surface
  - Interpolating $V_{00}, V_{n0}, V_{0m}, V_{nm}$
  - Boundary curves are Bezier curves
  - Algorithms adopted from curve case



Polynomial surface

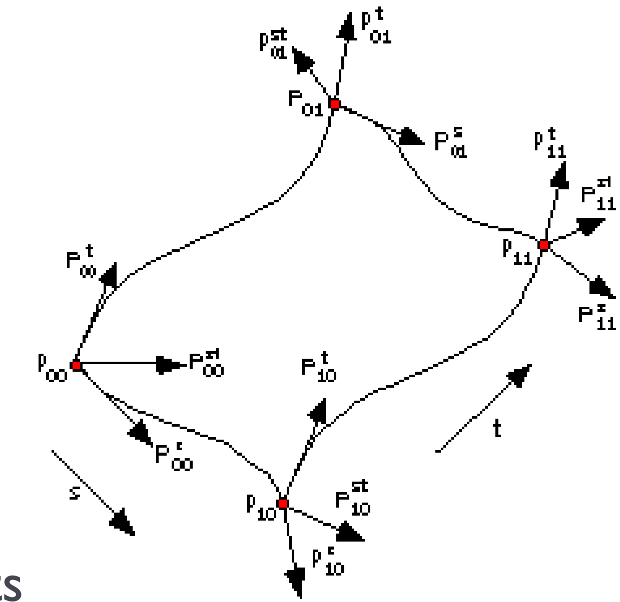
- Bezier triangle
  - $f(u,v) = \sum_{\substack{i=0,j=0,k=0 \\ i+j+k=n}}^{n} V_{ijk} B^n{}_{ijk}(u,v,1-u-v)$
  - Triangle domain: $u \in < 0,1 >, v \in < 0,1 >, u + v \leq 1$
  - Generalized Bernstein basis: $B^n{}_{ijk}(u,v,w) = \frac{n!}{i!j!k!} u^i v^j w^k$
  - u, v, w – barycentric coordinates in domain
  - Approximation surface of order n
  - Interpolating $V_{n00}, V_{0n0}, V_{00n}$
  - Special adaptation of curve algorithms





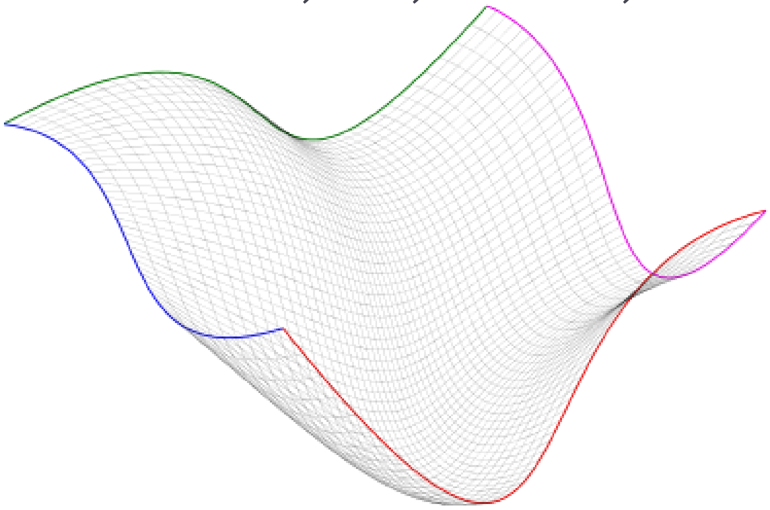
Polynomial surface

- Hermite bicubic surface
  - $f(u,v) = UHPH^TV^T$
  - $U = (u^3 \quad u^2 \quad u \quad 1), \mathrm{V} = (v^3 \quad v^2 \quad v \quad 1)$
  - $H = \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$
  - $P = \begin{pmatrix} P_{00} & P_{01} & P_{00}{}^v & P_{01}{}^v \\ P_{10} & P_{11} & P_{10}{}^v & P_{11}{}^v \\ P_{00}{}^u & P_{01}{}^u & P_{00}{}^{uv} & P_{01}{}^{uv} \\ P_{10}{}^u & P_{11}{}^u & P_{10}{}^{uv} & P_{11}{}^{uv} \end{pmatrix}$
  - $P_{00}, P_{10}, P_{01}, P_{11}$- interpolated corner points
  - $P_{ij}{}^u, P_{ij}{}^v$- tangent vectors in corner points
  - $P_{ij}{}^{uv}$- second order derivatives, twists, in corner points
  - Square domain: $u \in < 0,1 >, v \in < 0,1 >$



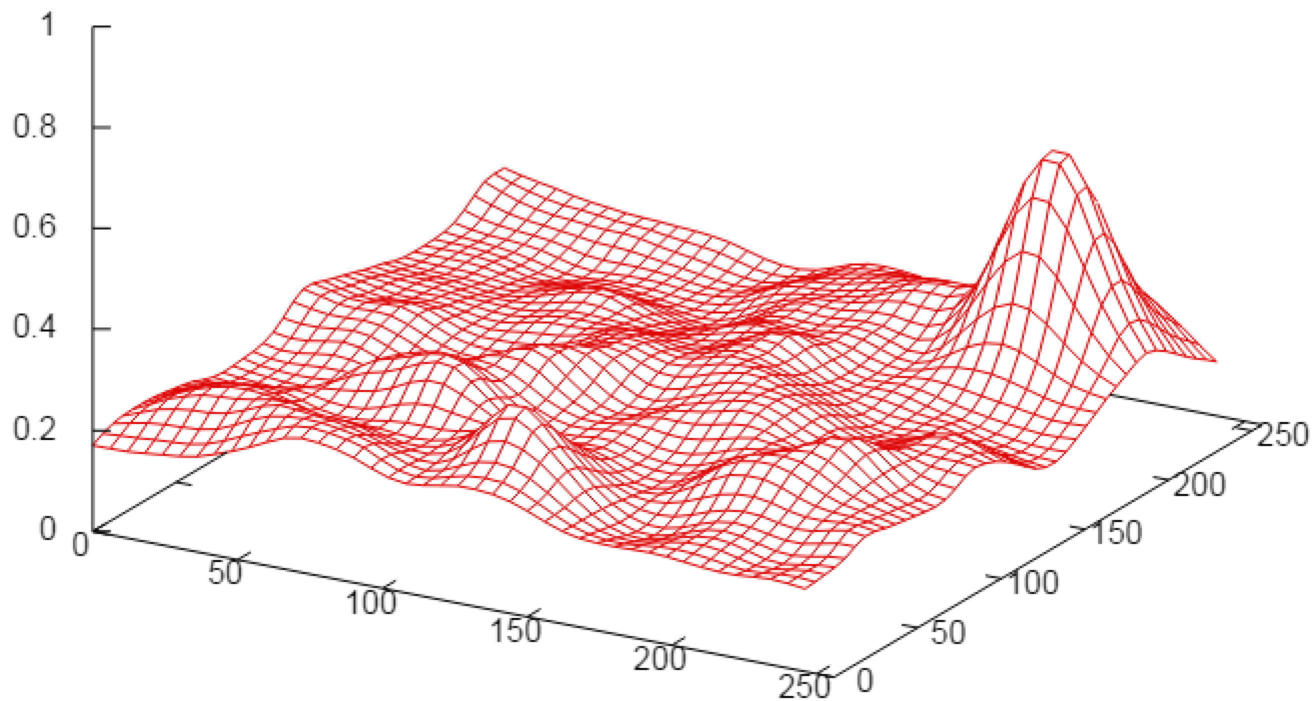
Polynomial surface

- Coons surface (patch)
  - Given four boundary parametric curves p$(u,0), p(u,1), p(0,v), p(1,v)$ meeting at four corners
  - $f(u,v) = p(u,0)(1-v) + p(u,1)w + p(0,v)(1-u) + p(1,v)u - p(0,0)(1-u)(1-v) - p(0,1)(1-u)v - p(1,0)u(1-v) - p(1,1)uv$
  - Square domain: $u \in < 0,1 >, v \in < 0,1 >$



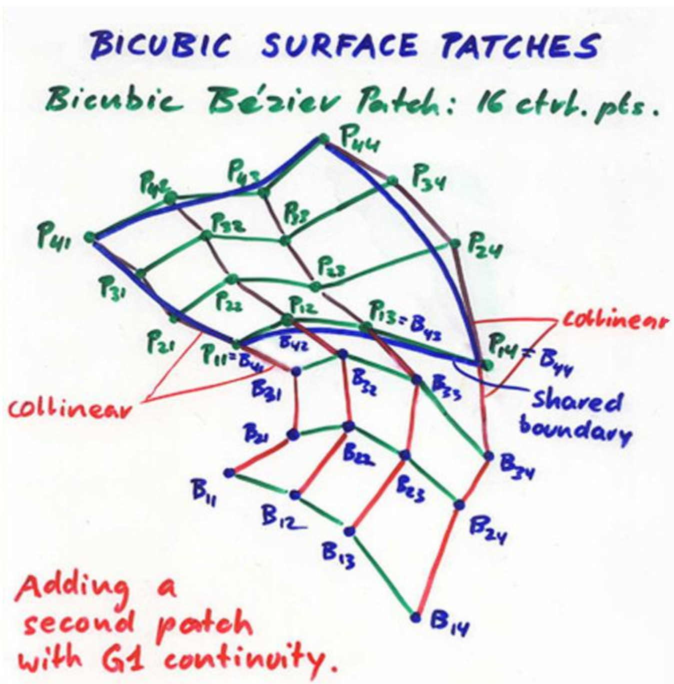
Spline surface

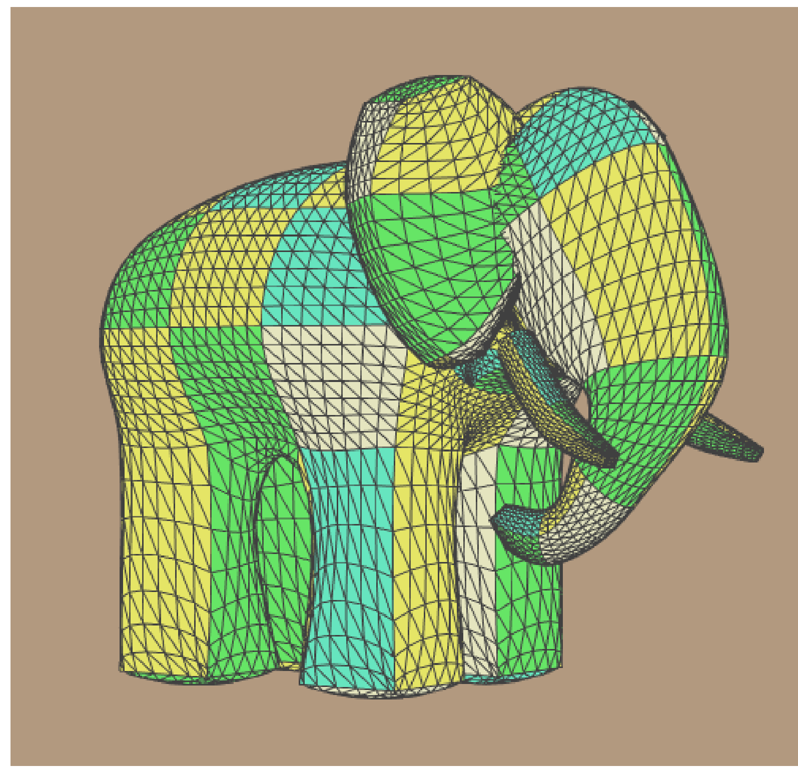
- Piecewise polynomial in both parametric directions
- Segments are polynomial surfaces with small order
- Expecting order of continuity in both directions



Bezier spline surface

- Each segment is represented as Bezier surface
- Usually linear, quadratic or cubic segments
- Continuity guaranteed by constraints on control points near boundary



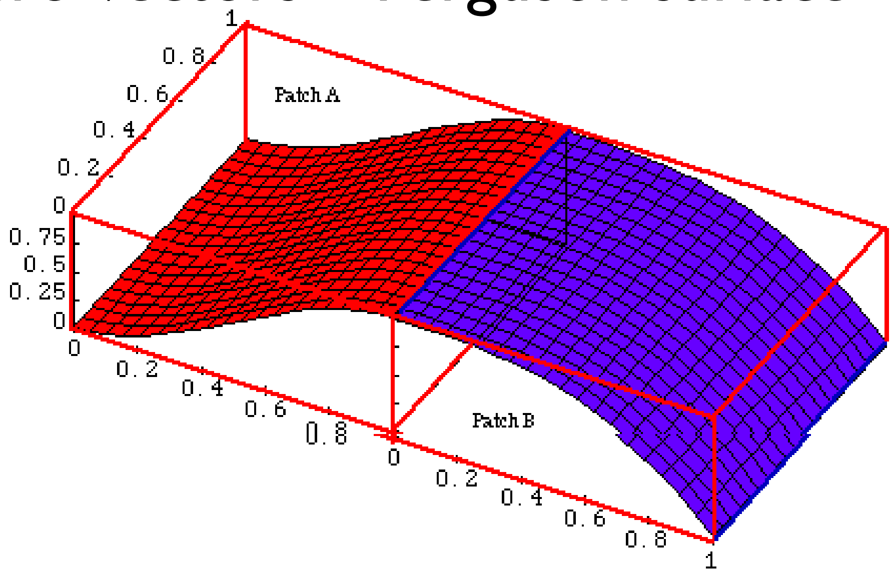


Hermite bicubic spline surface

- Given 2D grid of vertex points $V_{ij}; i = 0,1, \dots, n; j = 0,1, \dots, m$, grid of tangent vectors for vertex points in both directions ${V_{ij}}^u, {V_{ij}}^v; i = 0,1, \dots, n; j = 0,1, \dots, m$, grid of twist vectors for each vertex point ${V_{ij}}^{uv}; i = 0,1, \dots, n; j = 0,1, \dots, m$ two vectors of knot parameters $u_0 < u_1 < \cdots < u_n, v_0 < v_1 < \cdots < v_m$
- Interpolation surface, interpolating each given vertex V_{ij} and maintaining tangent vectors and twists at V_{ij}
- Interpolation of tangents and twists - C^1 continuity
- Each segment is represented in Hermite cubic surface form
  - For $u \in < u_0, u_n >, v \in < v_0, v_m >$, pick span kl such that $u \in < u_k, u_{k+1} >, v \in < v_l, v_{l+1} >$
  - $\bar{u} = \frac{u - u_k}{u_{k+1} - u_k}, \bar{v} = \frac{v - v_l}{v_{l+1} - v_l}$
  - Compute point on Hermite bicubic spline surface using Hermite bicubic surface for corners $V_{kl}, V_{k+1l}, V_{kl+1}, V_{k+1l+1}$ and parameters $\bar{u}, \bar{v}$

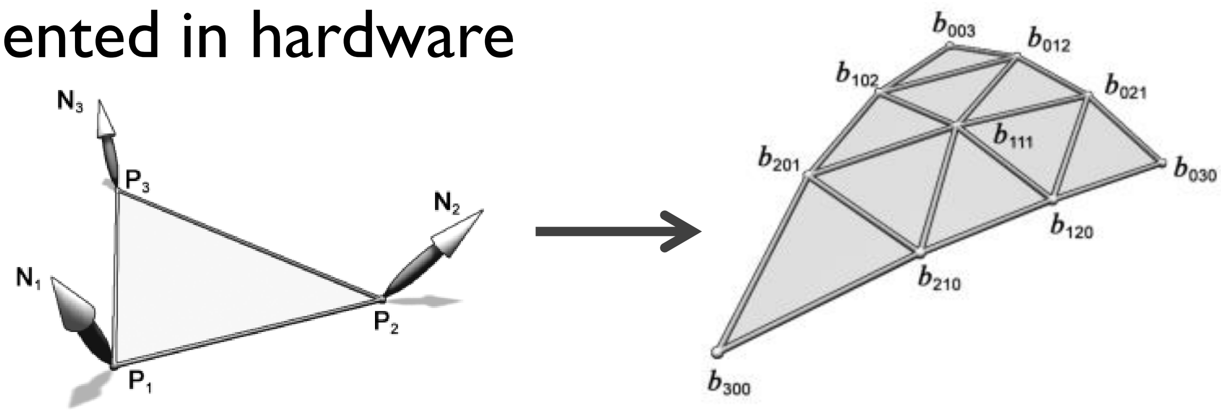
Hermite cubic spline surface

- Automatic computation of tangent vectors, knots from given points and knot parameters
- Automatic computation of knot vectors
- Using approaches from curve Hermite cubic spline case for each parameter separately
- Twists – zero vectors – Ferguson surface



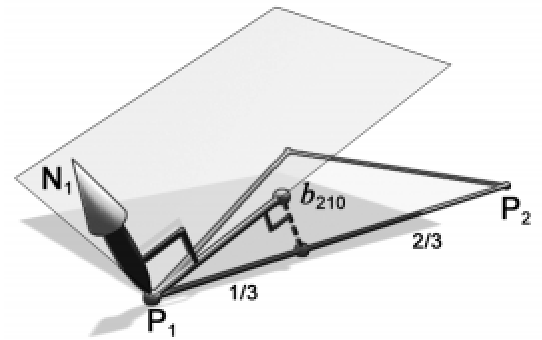
Curved PN triangles

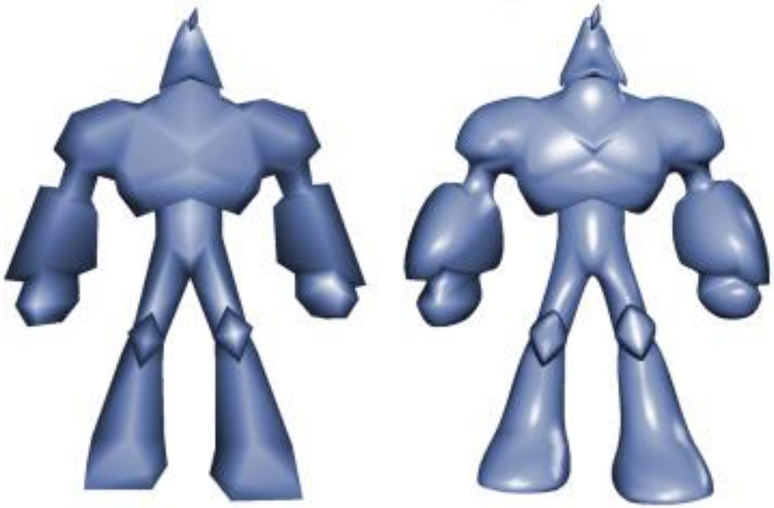
- https://www.cise.ufl.edu/research/SurfLab/papers/00ati.pdf
- Given triangular mesh with vertex normals
- Creating surface interpolating vertices of mesh and having given normals in that vertices
- Piecewise polynomial mesh, creating one Bezier triangle for each triangle of mesh
- Interpolating geometry – cubic Bezier triangle
- Interpolating normals – quadratic Bezier triangle
- Implemented in hardware



Curved PN triangles

- $b_{300} = P_1, b_{030} = P_2, b_{003} = P_3$
- $w_{ij} = (P_j - P_i).N_i$
- $b_{210} = \frac{2}{3}P_1 + \frac{1}{3}P_2 - \frac{w_{12}}{3}N_1$
- $b_{120} = \frac{2}{3}P_2 + \frac{1}{3}P_1 - \frac{w_{21}}{3}N_2$
- $b_{021} = \frac{2}{3}P_2 + \frac{1}{3}P_3 - \frac{w_{23}}{3}N_2$
- $b_{012} = \frac{2}{3}P_3 + \frac{1}{3}P_2 - \frac{w_{32}}{3}N_3$
- $b_{102} = \frac{2}{3}P_3 + \frac{1}{3}P_1 - \frac{w_{31}}{3}N_3$
- $b_{201} = \frac{2}{3}P_1 + \frac{1}{3}P_3 - \frac{w_{13}}{3}N_1$
- $V = \frac{1}{3}P_1 + \frac{1}{3}P_2 + \frac{1}{3}P_3$
- $E = \frac{1}{6}b_{210} + \frac{1}{6}b_{120} + \frac{1}{6}b_{021} + \frac{1}{6}b_{012} + \frac{1}{6}b_{102} + \frac{1}{6}b_{201}$
- $b_{111} = \frac{3}{2}E - \frac{1}{2}\mathrm{V}$

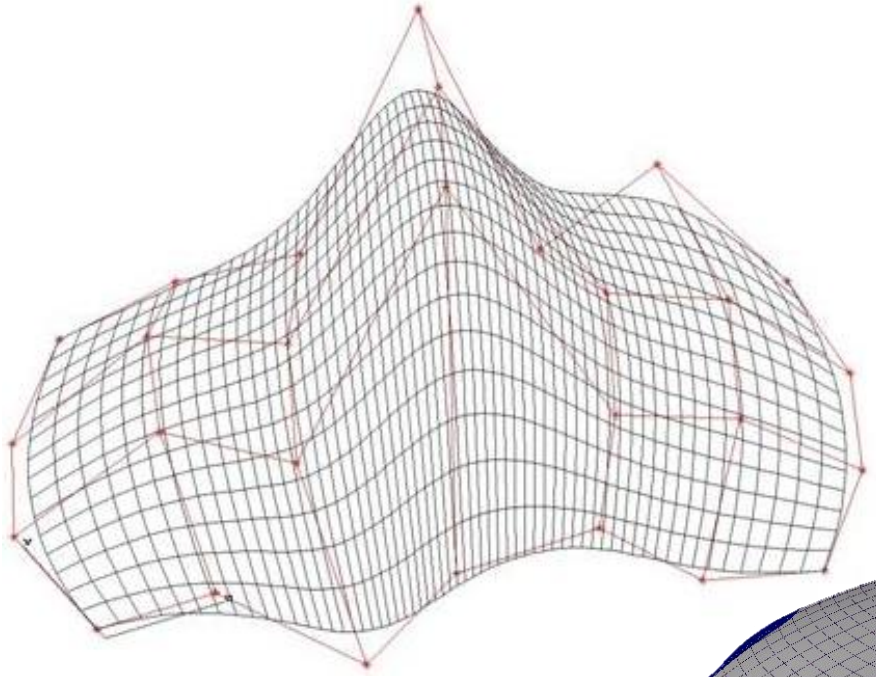


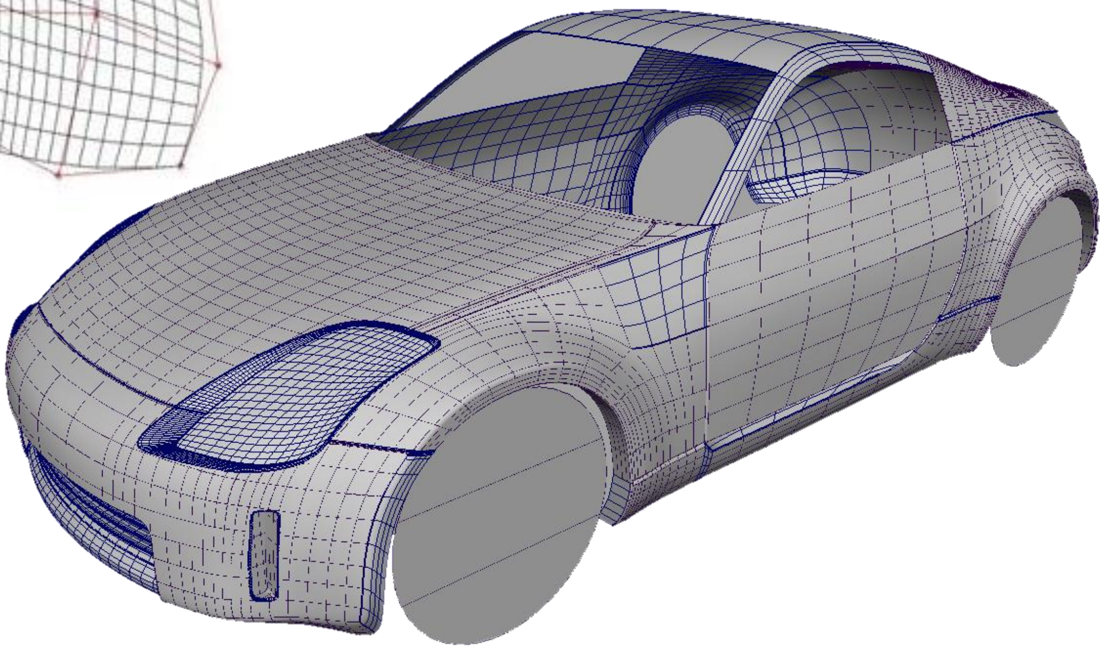


B-spline surface

- Compact representation of approximating spline surfaces
- Tensor product surface
- Input
  - Polynomial degrees d_u, d_v
  - 2D grid of control points$V_{ij}; i = 0, \dots, n_u; j = 0, \dots, n_v$
  - 2 vectors of knot parameters $(u_0, u_1, \dots, u_{m_u}), (v_0, v_1, \dots, v_{m_v})$
  - $m_u = n_u + d_u + 1, m_v = n_v + d_v + 1$
- $BSS^{d_u d_v}(u, v) = \sum_{i=0}^{n_u} \sum_{j=0}^{n_v} V_{ij} N^{d_u}{}_i(u) N^{d_v}{}_j(v)$
- Rectangle domain: $u \in < u_{d_u}, u_{n_u+1}), v \in < v_{d_v}, v_{n_v+1})$
- Using B-spline basis function same as in curve case
- Similar properties and algorithms as in curve case, treating each parameter separately

B-spline surface

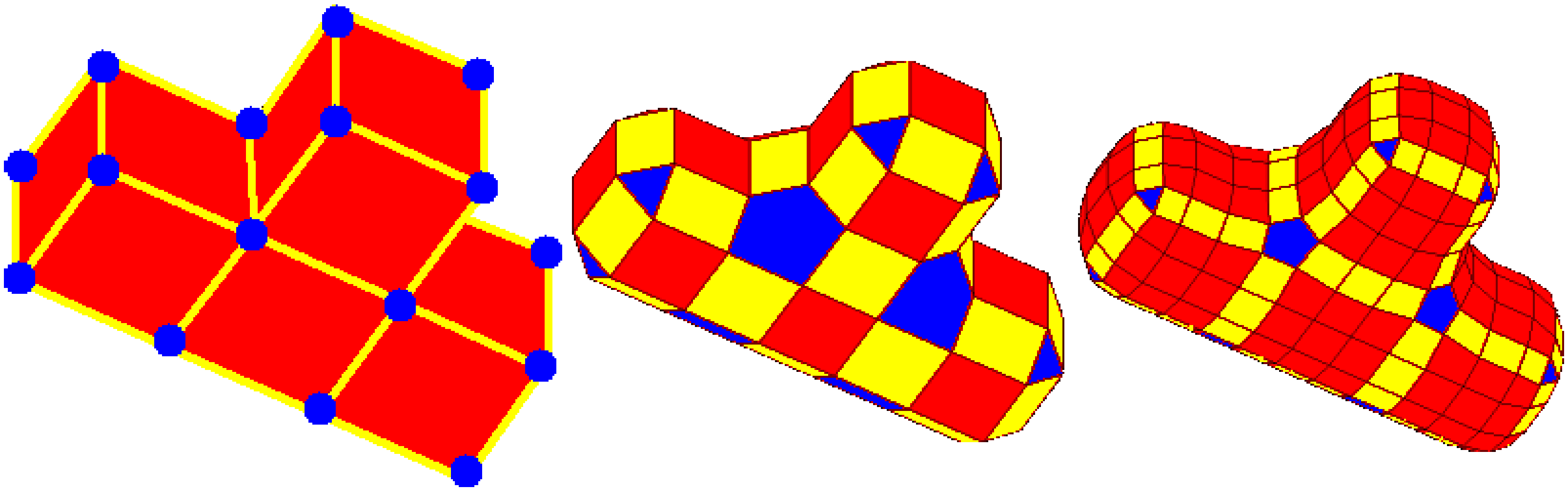


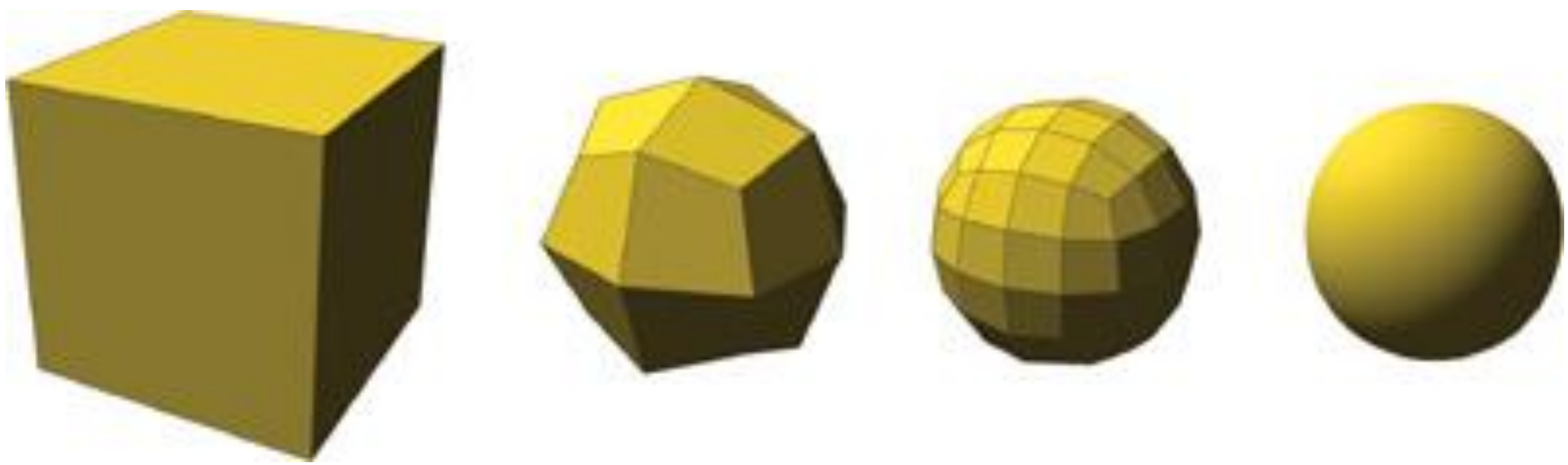


Surface subdivision algorithms

- Producing extended set of control points without change in shape of original surface
- Knot insertion, Boehm algorithm, degree elevation
- Doo-Sabin subdivision
  - Corner and edge cutting algorithm
  - Uniform knot insertion into biquadratic B-spline surface
  - Originally for regular 2D grid of control points extended for arbitrary meshes, producing polygons of arbitrary size
- Catmull-Clark subdivision
  - Uniform knot insertion into bicubic B-spline surface
  - Originally for regular 2D grid of control points extended for arbitrary meshes, producing only quads

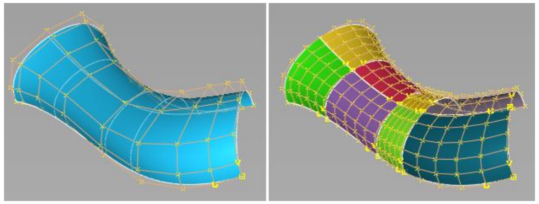
Surface subdivision algorithms





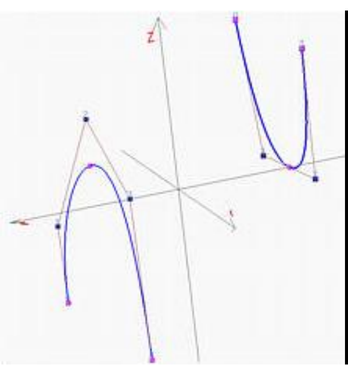
NURBS surface

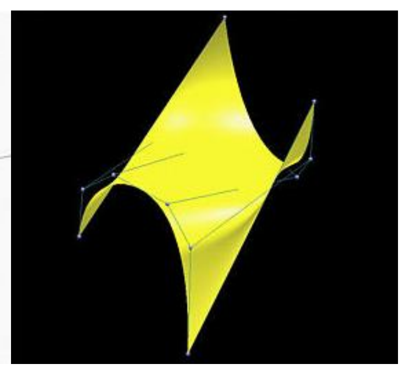
- Non-Uniform Rational B-spline surface
- Defining weights (real numbers) w_{ij} for each control point
- Embedding B-spline surface into space with additional dimension – into projective, homogenous space
  - $V_{ij} = (x_{ij}, y_{ij}, z_{ij}) \rightarrow PV_{ij} = (w_{ij}x_{ij}, w_{ij}y_{ij}, w_{ij}z_{ij}, w_{ij})$
- Evaluation, algorithms in projective space
- Projection of result point back to affine space
  - $PX = (x, y, z, w) \rightarrow X = (\frac{x}{w}, \frac{y}{w}, \frac{z}{w})$
- $RBSS^{d_u d_v}(u, v) = \dfrac{\sum_{i=0}^{n_u} \sum_{j=0}^{n_v} w_{ij} V_{ij} N^{d_u}{}_i(u) N^{d_v}{}_j(v)}{\sum_{i=0}^{n_u} \sum_{j=0}^{n_v} w_{ij} N^{d_u}{}_i(u) N^{d_v}{}_j(v)}$

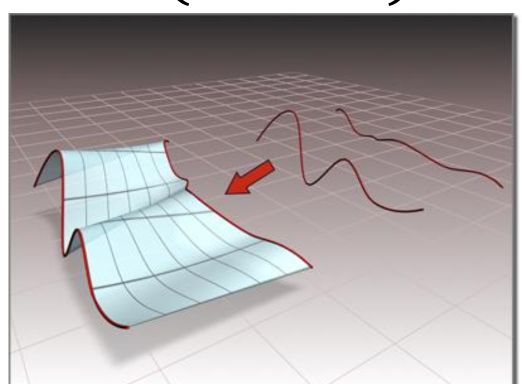


NURBS ruled surface

- For each point there is line (segment) passing through that point and lying on surface
- Connecting two NURBS curves using line segments
- Compacting both curves to have same degree and same knot vector – linear transformation of parameter, knot insertion, degree elevation
- Putting control points of curves into 2D
- $d_v = 1$
- Knot vector for v direction - $(0,0,1,1)$

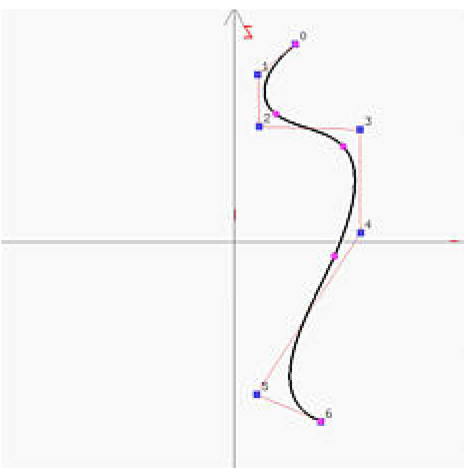
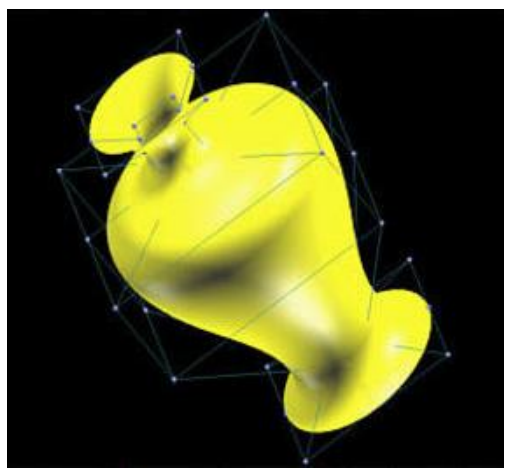
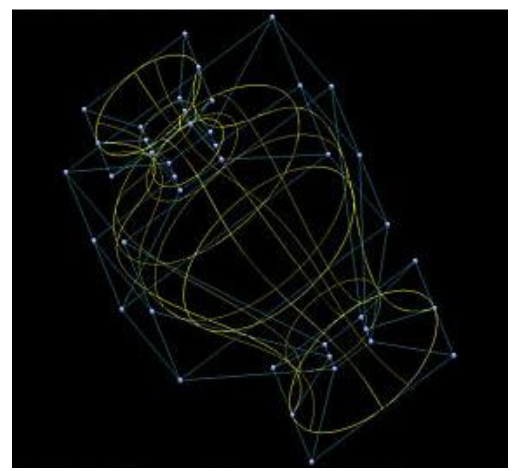






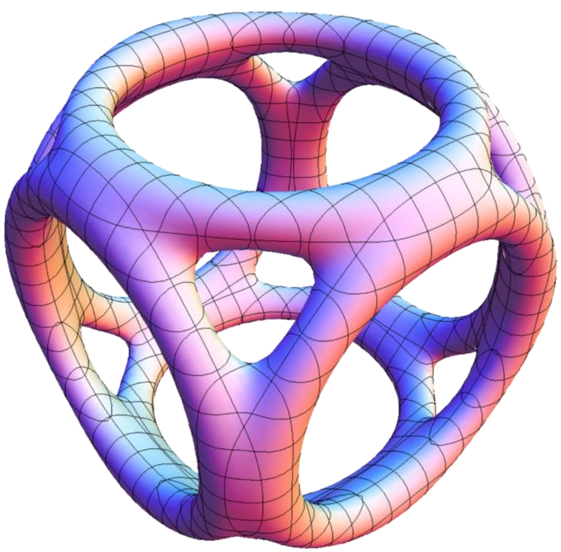
NURBS surface of revolution

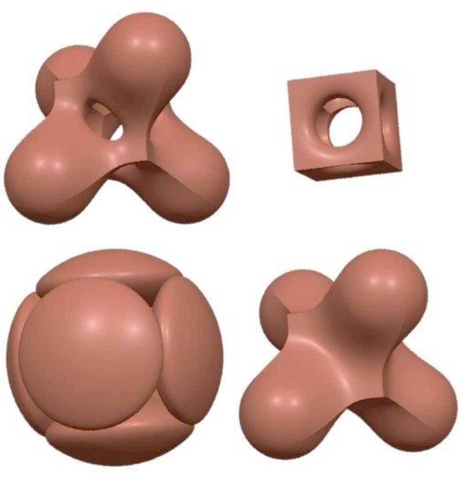
- Rotating NURBS curve around line (coordinate axis)
- u-direction – given NURBS curve
- v-direction – parameters of circular arc as NURBS curve
- Control points – rotated control points of given NURBS curve around given line forming control points for circular arc as NURBS curve

Implicit surface

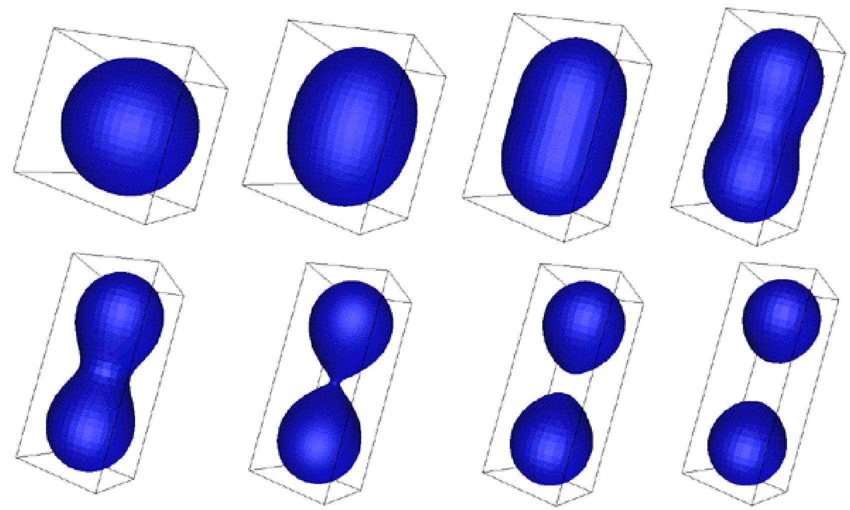
- Set of all points $X \in \boldsymbol{E}^3$ such that $f(X) = 0$
  - Sphere: $x^2+y^2+z^2-r^2=0$
- Easy computation if some point is on surface
- Defining interior, exterior, border regions by sign of f
- Hard to generate points on surface
- "Metaballs", "Blobbies", "Soft objects“
- Smooth





Implicit surface

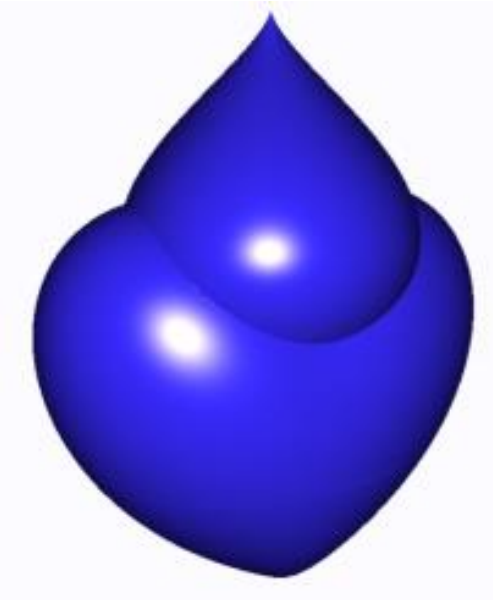
- Generation from primitives (points, lines, …)-$P_1, P_2, \ldots, P_n$
- Simulating energy field around primitives
- $D_i(X)$ - Distance of point X and primitive P_i
- $f(X) = \sum_{i=0}^{n} B(D_i(X)) - \mathrm{F}$
- F – isovalue, field strength
- Blobby molecules
  - $B(r) = ae^{-br^2}, B(r) = \frac{a}{r^2}$
- Metaballs
  - $B(r) = a(1 - \frac{3r^2}{b^2})$ for $0 \leq r \leq \frac{b}{3}$
  - $B(r) = \frac{3a}{2}(1 - \frac{r}{b})^2$ for $\frac{b}{3} \leq r \leq b$
  - $B(r) = 0$ for $b \leq r$
- Soft Objects
  - $B(r) = a(1 - \frac{4r^6}{9b^6} + \frac{17r^4}{9b^4} - \frac{22r^2}{9b^2})$ for $0 \leq r \leq b$
  - $B(r) = 0$ for $b \leq r$



Two point primitives, varying isovalue F

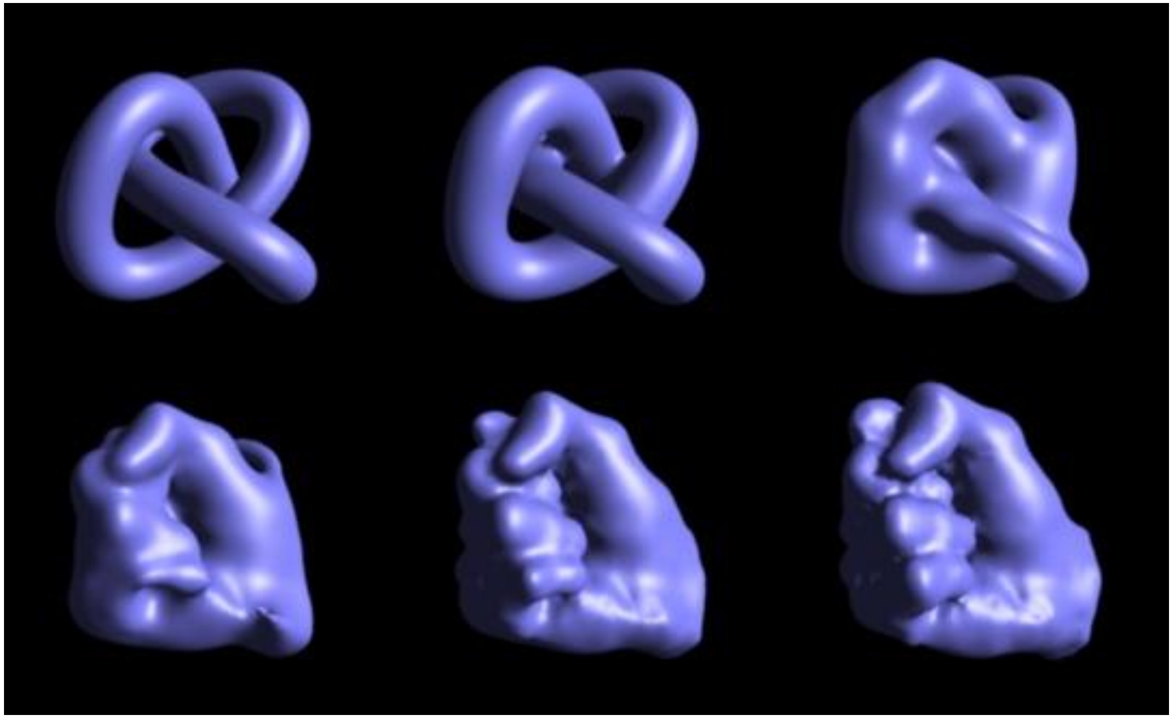
Implicit surface

- Boolean operations on two objects represented as implicit surfaces with functions f_A, f_B
- Union
  - $f_{A\cup B}(X) = \min(f_A(X), f_B(X))$
  - $f_{A\cup B}(X) = -e^{-bf_A(X)} - e^{-bf_A(X)} + 1$
- Intersection
  - $f_{A\cap B}(X) = \max(f_A(X), f_B(X))$
  - $f_{A\cap B}(X) = -e^{-bf_A(X)} + e^{-bf_A(X)} + 1$
- Difference
  - $f_{A-B}(X) = \max(f_A(X), -f_B(X))$
  - $f_{A-B}(X) = e^{bf_A(X)} + e^{bf_A(X)} + 1$



Implicit surface

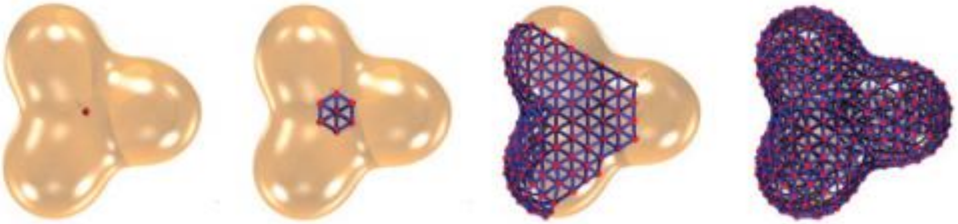
- Smooth approximation of several implicit surfaces
  - $f(X) = f_1(X).f_2(X) \dots f_n(X) - C$
- Morphing, metamorphosis of two surfaces
  - $f(X) = (1-\mu)f_1(X) + \mu f_2(X), \mu \in <0,1>$

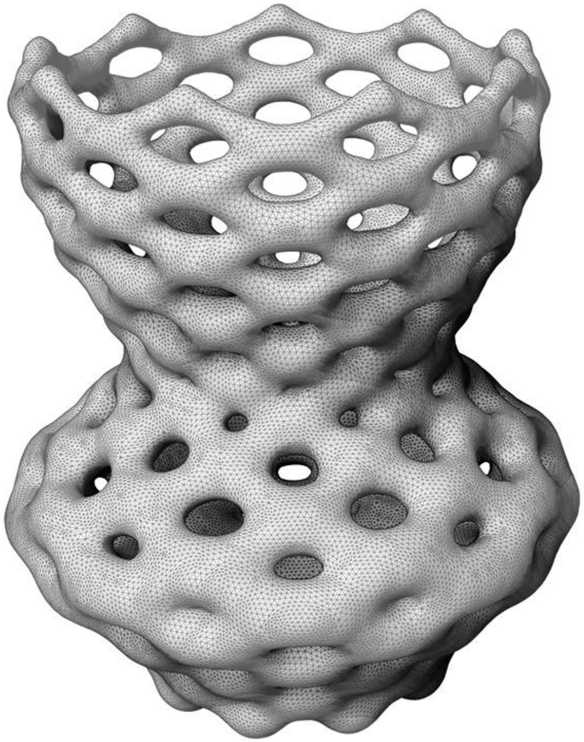


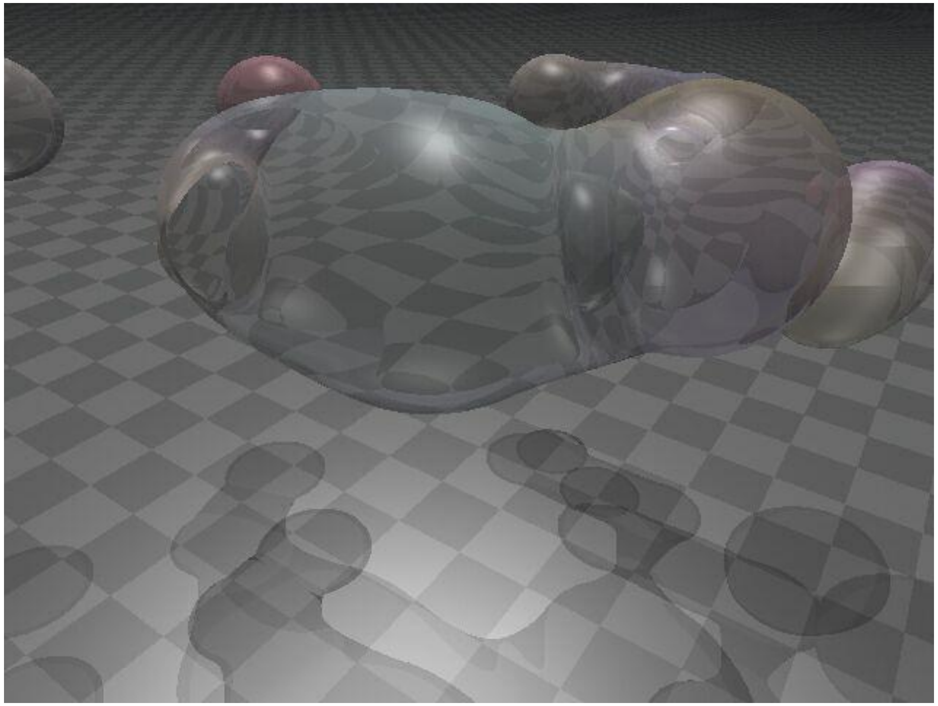
Implicit surface

- Visualization algorithms
- http://dl.acm.org/citation.cfm?id=2732197
- Points generation
  - Distributing particles over implicit surface
- Spatial decomposition
  - Sampling implicit function in finite uniform grid points
  - Generating surface triangles for each cell separately
  - Marching cubes, marching tetrahedra
- Surface tracing
  - Creating triangles by tracing surface from starting point
  - Marching triangles
- Ray-tracing
  - Simulating rays from eye through screen into scene
  - Each ray given in parametric form $X = S + tD, t \in \boldsymbol{R}$
  - Finding intersection of ray and surface
  - Solving $f(S + tD) = 0$ directly or using numerical methods (Newton..)

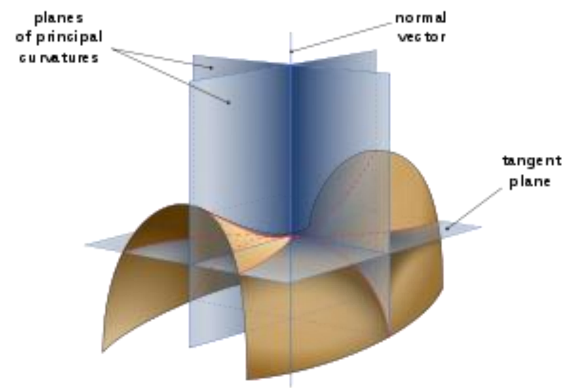
Implicit surface



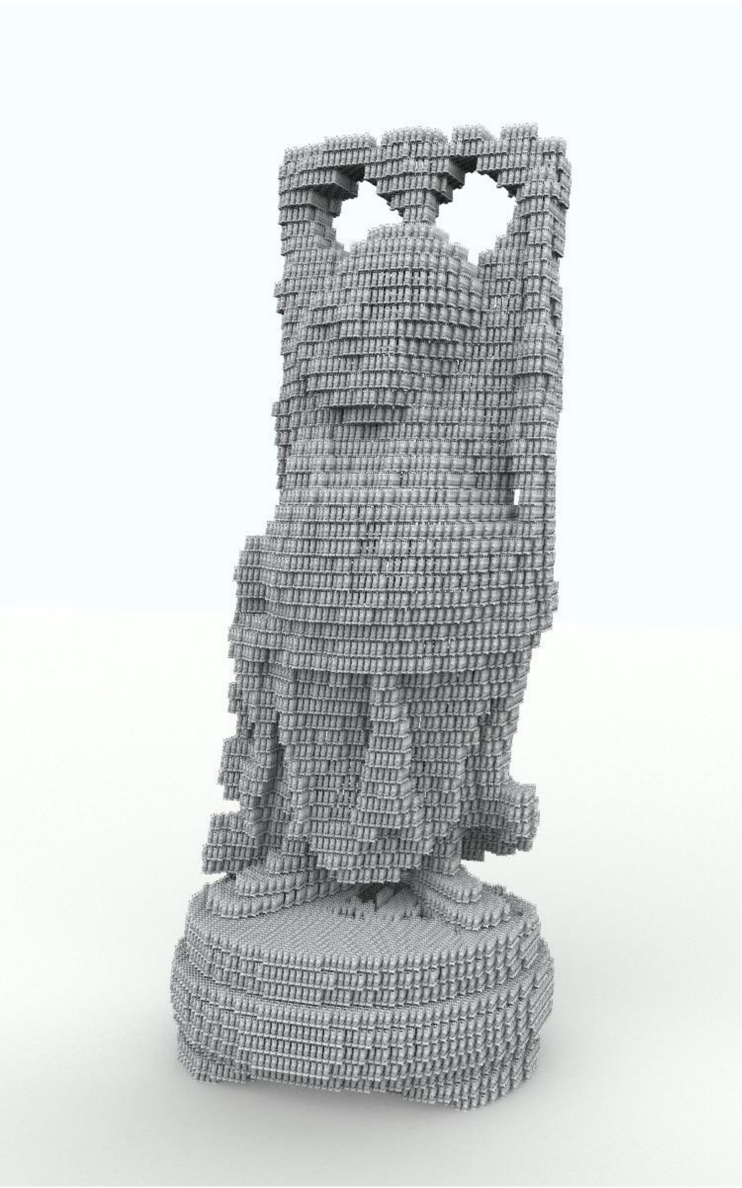




Differential geometry



- Parametric surface
  - Tangent vectors – $T_u = \frac{\partial f(u,v)}{\partial u}, T_v = \frac{\partial f(u,v)}{\partial v}$
  - Normal vector – $\mathrm{N} = T_u x T_v$
  - Curvature is based on curve case
  - For each direction from tangent plane → perpendicular plane to surface → intersection curve → curvature
  - Principal curvatures = min, max curvatures k_1, k_2
  - Mean curvature - $H = \frac{k_1+k_2}{2}$, Gaussian curvature - $K = k_1.k_2$
- Implicit surface
  - Gradient, normal vector - $\nabla \mathrm{f} = \mathrm{N} = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right) = (f_x, f_y, f_z)$
  - Surface is regular at point if gradient is not zero vector
  - Curvatures determined from parametric case



The End

for today