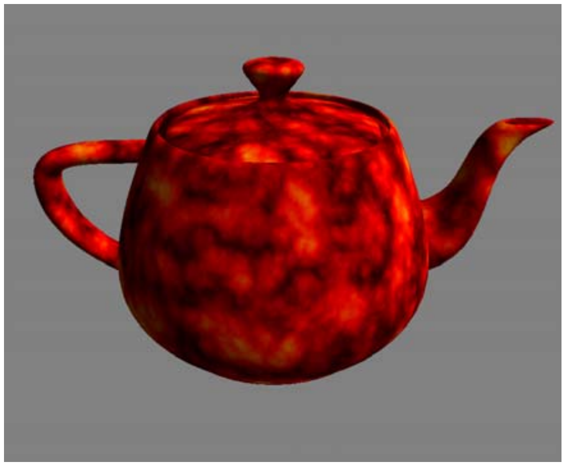
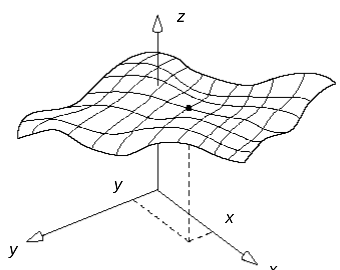
Surfaces



Height Fields

$z = f(x,y)$



Example Height Fields

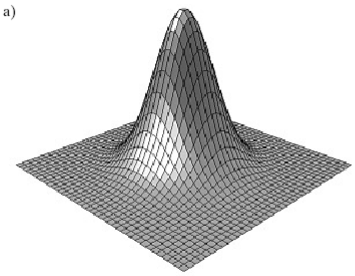
Gaussian

$$z = f(x, y) = e^{-ax^2 - by^2}$$

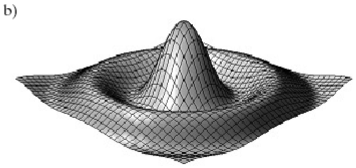
Sinc

$$z = f(x, y) = \frac{\sin\left(\sqrt{x^2 + y^2}\right)}{\sqrt{x^2 + y^2}}$$

a)



b)



Surface Representations

Explicit: *z = f(x,y)*

Implicit: *f(x,y,z) = 0*

Surface normal: $$\mathbf{n} = \underbrace{\nabla}_{\text{gradient operator}} f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{bmatrix}$$

Parametric: $x = f_x(u,v)$, $y = f_y(u,v)$, $z = f_z(u,v)$

Computing Surface Normals

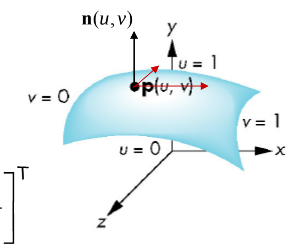
Parametric surface patch

$$\mathbf{p}(u,v) = \left[x(u,v) \quad y(u,v) \quad z(u,v)\right]^{\mathrm{T}}$$

Tangent vectors to surface

$$\mathbf{p}_u(u,v) = \left[\frac{\partial x}{\partial u} \quad \frac{\partial y}{\partial u} \quad \frac{\partial z}{\partial u}\right]^{\mathrm{T}}$$

$$\mathbf{p}_v(u,v) = \left[\frac{\partial x}{\partial v} \quad \frac{\partial y}{\partial v} \quad \frac{\partial z}{\partial v}\right]^{\mathrm{T}}$$



The tangent vectors are also tangent to the ***isoparametric curves*** $\mathbf{p}(u{=}c,v)$ and $\mathbf{p}(u,v{=}c)$

Unit normal vector to parametric surface

$$\mathbf{n}(u,v) = \frac{\mathbf{p}_u \times \mathbf{p}_v}{\left\|\mathbf{p}_u \times \mathbf{p}_v\right\|}$$

Quadric Surfaces

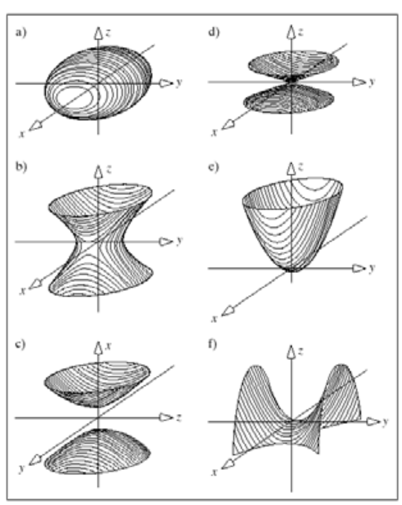


FIGURE 6.70 The six quadric surfaces: (a) Ellipsoid. (b) Hyperboloid of one sheet. (c) Hyperboloid of two sheets. (d) Elliptic cone. (e) Elliptic paraboloid. (f) Hyperbolic paraboloid.

Quadric Surfaces

Sphere

$$f(x, y, z) = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 - R^2 = 0$$

$$x(\phi, \theta) = R\cos(\phi)\cos(\theta)$$
$$y(\phi, \theta) = R\cos(\phi)\sin(\theta)$$
$$z(\phi, \theta) = R\sin(\phi)$$

$$-\pi/2 \leq \phi \leq \pi/2$$
$$-\pi \leq \theta \leq \pi$$

Quadric Surfaces

Ellipsoid

$$f(x, y, z) = \left(\frac{x - x_0}{R_x}\right)^2 + \left(\frac{y - y_0}{R_y}\right)^2 + \left(\frac{z - z_0}{R_z}\right)^2 - 1 = 0$$

$$x(\phi, \theta) = R_x\cos(\phi)\cos(\theta)$$
$$y(\phi, \theta) = R_y\cos(\phi)\sin(\theta)$$
$$z(\phi, \theta) = R_z\sin(\phi)$$

$$-\pi/2 \leq \phi \leq \pi/2$$
$$-\pi \leq \theta \leq \pi$$

Parametric Formulations

Ruled surfaces:

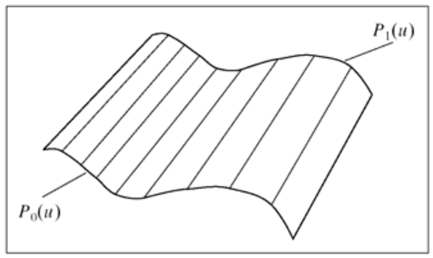
Convex linear combination of two curves

- Through every point on the surface there passes at least one line that lies on the surface

$$P(v) = (1 - v)P_0 + vP_1$$

Making P_0 and P_1 curves:

$$P(u,v) = (1 - v)P_0(u) + vP_1(u)$$

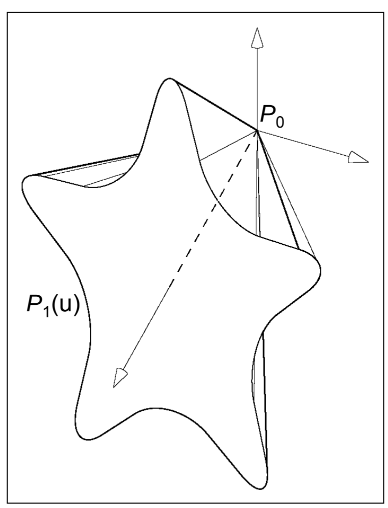


Special Cases

Generalized cone

$$P(u,v) = (1 - v)P_0 + vP_1(u)$$

P_0 is the apex

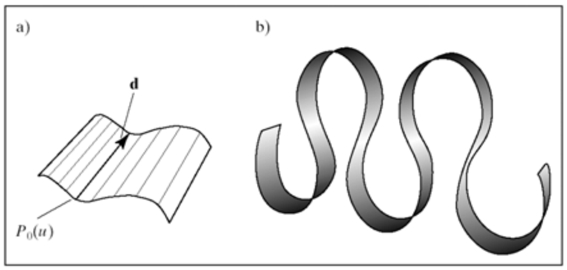


Special Cases

Generalized cylinder

P_1 is a translated version of P_0

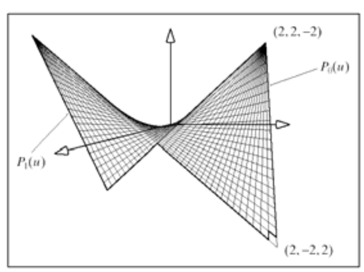
$$P(u,v) = (1-v)P_0(u) + v(P_0(u) + \mathbf{d}) \Rightarrow P(u,v) = P_0(u) + v\mathbf{d}$$



Bilinear Patches

Both P_1 and P_0 are lines

$$\begin{aligned} P(u,v) &= (1-v)P_0(u) + vP_1(u) \\ &= (1-v)[(1-u)P_{00} + uP_{01}] + v[(1-u)P_{10} + uP_{11}] \\ &= (1-v)(1-u)P_{00} + (1-v)uP_{01} + v(1-u)P_{10} + vuP_{11} \end{aligned}$$

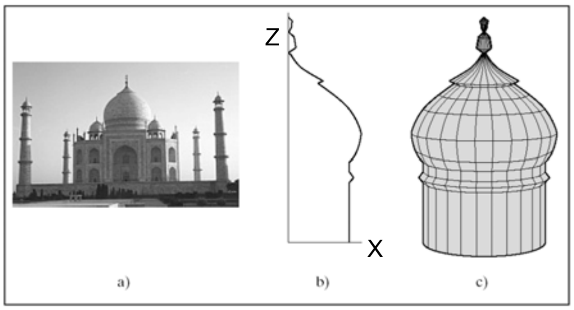


Surfaces of Revolution

Sweep profile curve around an axis:

$C(v) = [x(v), z(v)]^T$

$P(u,v)=[x(v)\cos(u), x(v)\sin(u), z(v)]^T$



Spline Surface Patches

Our prior examples of surfaces are useful, but…

- We generated them by hand from first principles
- The parameterization is completely customized

It would be nice to have a common building block

- Just as we can build curves out of many spline segments, we can build surfaces out of spline patches

Formulating Spline Patches

Our spline curves had the form

$$\mathbf{p}(u) = \sum_{i=0}^{n} \mathbf{p}_i B_i^n(u) \qquad 0 \le u \le 1$$

- A linear combination of control points $\mathbf{p}_i$
- Controlled by blending functions B_i^n

Our spline patches will have an analogous form

$$\mathbf{p}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} \mathbf{p}_{ij} B_{ij}^{mn}(u,v) \qquad 0 \le u,v \le 1$$

Tensor Product Patches

We assumed a set of* nm *basis functions

$$\mathbf{p}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} \mathbf{p}_{ij} B_{ij}^{mn}(u,v) \qquad 0 \le u,v \le 1$$

We will only consider "tensor product" patches

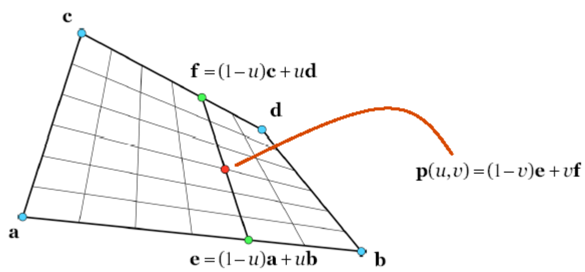
- Each basis function is the product of two 1-D basis functions

$$B_{ij}^{mn}(u,v) = B_i^m(u) B_j^n(v)$$

- Giving us the general spline equation

$$\mathbf{p}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} \mathbf{p}_{ij} B_i^m(u) B_j^n(v) \qquad 0 \le u,v \le 1$$

Bilinear Interpolation



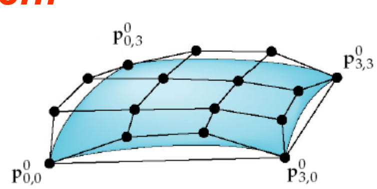
de Casteljau Algorithm for Bézier Patches

Repeated bilinear interpolation

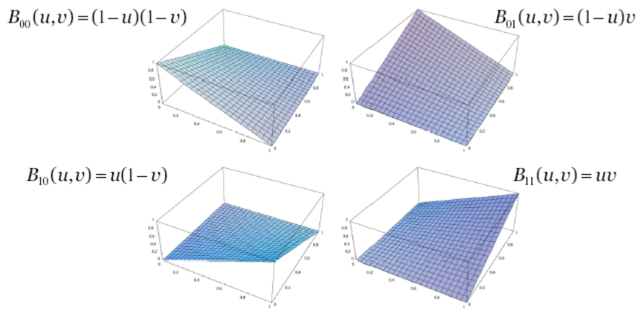
$$\mathbf{p}_{i,j}^{r}(u,v) = (1-u)(1-v)\mathbf{p}_{i,j}^{r-1} + u(1-v)\mathbf{p}_{i,j+1}^{r-1} + (1-u)v\,\mathbf{p}_{i+1,j}^{r-1} + uv\,\mathbf{p}_{i+1,j+1}^{r-1}$$

Producing the Bézier patch:

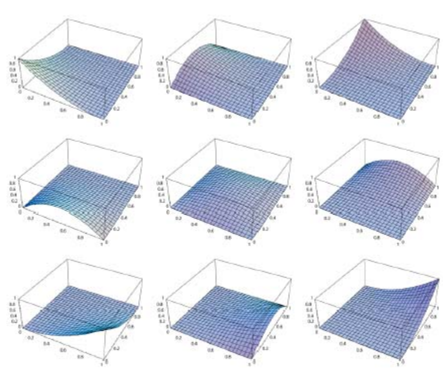
$$\mathbf{p}(u,v) = \mathbf{p}_{0,0}^{n}(u,v)$$



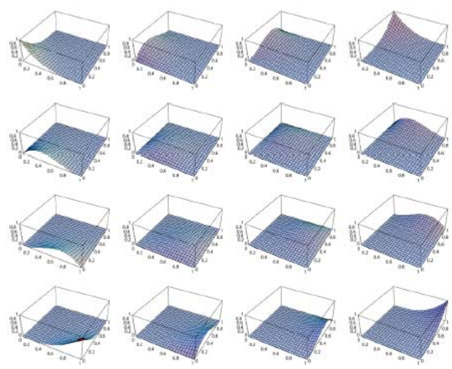
Bilinear Bézier-Bernstein Basis



Biquadratic Bézier-Bernstein Basis



Bicubic Bézier-Bernstein Basis



Properties of Bezier Surfaces

Affine invariance

Convex hull

Plane precision

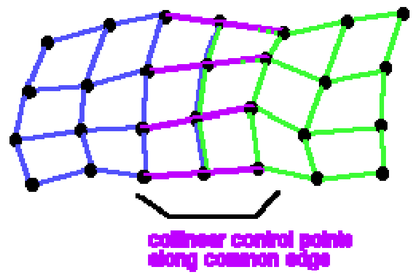
Variation diminishing

Piecewise Cubic Bezier Surfaces

G1 continuity

Common edge

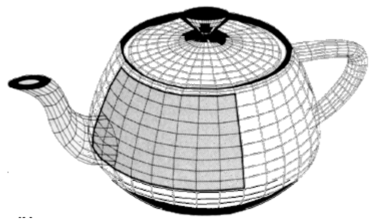
Make 2 sets of 4 control points collinear



Modeling Objects with Patches

Paste together multiple patches to cover entire object

- For example, the Utah Teapot is built from 32 patches



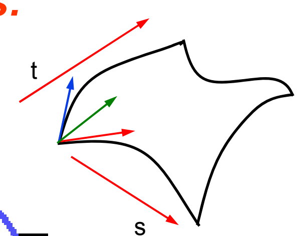
This raises some tricky questions

- How many patches are needed?
- How to guarantee the continuity of patches?
  - *While animating?!*
- How can we cut holes in the surface?
  - *Trimming curves – create boundary spline curves on surface*

Hermite Surfaces

Constraints at the four corners:

- Position, Tangent, Twist



$$G_x = \begin{bmatrix} x(0,0) & x(0,1) & \frac{\partial x}{\partial t}(0,0) & \frac{\partial x}{\partial t}(0,1) \\ x(1,0) & x(1,1) & \frac{\partial x}{\partial t}(1,0) & \frac{\partial x}{\partial t}(1,1) \\ \frac{\partial x}{\partial s}(0,0) & \frac{\partial x}{\partial s}(0,1) & \frac{\partial^2 x}{\partial s \partial t}(0,0) & \frac{\partial^2 x}{\partial s \partial t}(0,1) \\ \frac{\partial x}{\partial s}(1,0) & \frac{\partial x}{\partial s}(1,1) & \frac{\partial^2 x}{\partial s \partial t}(1,0) & \frac{\partial^2 x}{\partial s \partial t}(1,1) \end{bmatrix}$$

Rendering Parametric Curves and Surfaces

Transform into primitives we know how to handle

Curves

- Line segments

Surfaces

- Quadrilaterals
- Triangles

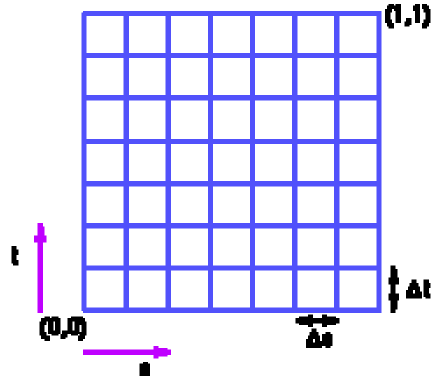
Converting to Quadrilaterals

Straightforward

Uniform subdivision

Evaluation of P(s,t) at each grid point

Isoparametric lines become isoparametric curves



Drawing Spline Curves and Surfaces

Method 1 – Direct evaluation of curves

- We have a function that generates points on the curve
- Vary parameter *u* between 0 and 1
- Substitute into formula and compute a position
- Connect consecutive points with line segments
- Method 1a – Direct evaluation with forward differencing
  - *Instead of evaluating polynomials directly, incrementalize polynomials to cut down on multiplies*

This approach has some problems

- Uniform parameter spacing is not uniform in space
  - *Length of segments will vary over curve*
- Control over length is important
  - *Too long – jagged curves*
  - *Too short – excessive drawing time*

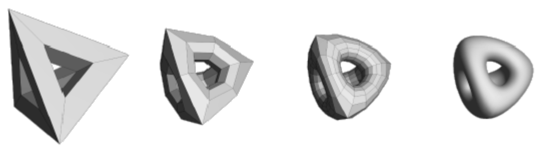
Modeling by Subdivision

Recall that we can draw spline curves by subdivision

- Start with the control polyline
- Recursively subdivide until “smooth enough”
- And draw the individual line segments

We can use this as a modeling primitive

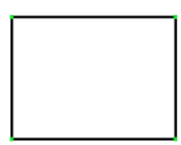
- Define the curve (or surface) as the limit of an infinite number of subdivision steps
- Discard all our polynomials!



Developing Subdivision Curves

Assume that we have some control polygon

- A closed piecewise linear curve in the plane



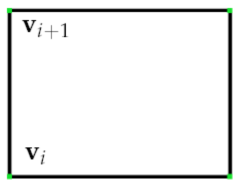
We need two fundamental operations:

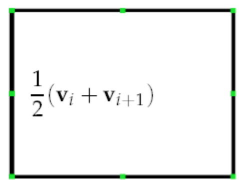
- Linear subdivision — introduce new vertices
- Linear smoothing — modify position of vertices

Linear Subdivision of Curves

Split each edge of the curve at its midpoint (*barycenter*)

- Thus doubling the number of vertices

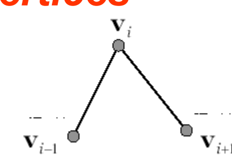




Linear Smoothing of Curves

Reposition each vertex at a weighted combination of neighbor vertices

$$\mathbf{v}_i' = \alpha_1 \mathbf{v}_{i-1} + \alpha_2 \mathbf{v}_i + \alpha_3 \mathbf{v}_{i+1}$$
$$\sum_i \alpha_i = 1$$



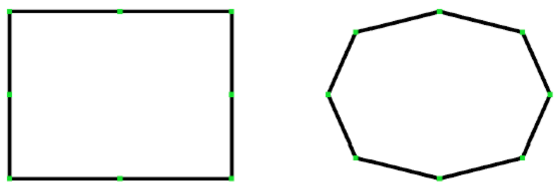
We can also write the above in matrix form

$$\mathbf{v}_i' = \begin{bmatrix} \mathbf{v}_{i-1} & \mathbf{v}_i & \mathbf{v}_{i+1} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}$$

Linear Smoothing of Curves

We are generally interested in symmetric weighting schemes

$$\mathbf{v}_i' = \left(\frac{1-\alpha}{2}\right)\mathbf{v}_{i-1} + \alpha\,\mathbf{v}_i + \left(\frac{1-\alpha}{2}\right)\mathbf{v}_{i+1}$$

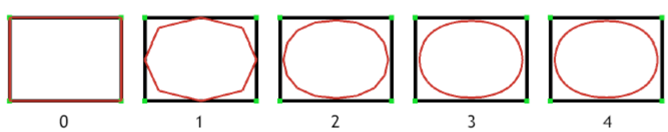


Weights $\begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix}$

Creating Smooth Curves by Subdivision

Repeat subdivision and smoothing operations

- Converges to some limit curve (determined by weights)
- For weights $\begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix}$ the resulting curve is a piecewise B-spline!



Subdivision as Linear Operator

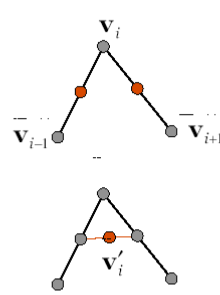
Points after** k **steps are linear combinations of previous points

- We can therefore write the subdivision step as a matrix operation

$$\mathbf{p}_k = \mathbf{p}_{k-1}\mathbf{S}_{k-1}$$

$$\begin{bmatrix} x_1 & \cdots & x_{2i} & x_{2i+1} & \cdots & x_{2n} \\ y_1 & \cdots & y_{2i} & y_{2i+1} & \cdots & y_{2n} \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & \ldots & x_n \\ y_1 & y_2 & \cdots & y_n \end{bmatrix}\mathbf{S}_{k-1}$$

Smoothing as Barycentric Averaging



Compute barycenters of adjacent edges

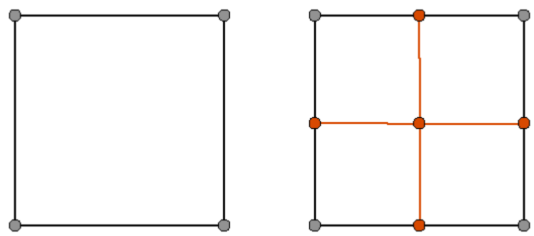
Compute barycenter of barycenters

- Same as weights $\begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix}$ but works in higher dimensions

Surfaces: Quadrilateral Subdivision of Polygons

Split face in middle and connect to edge midpoints

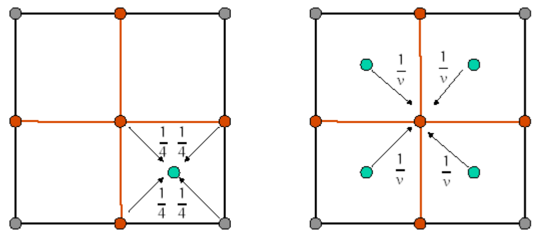
- Converts any polygon into set of quadrilaterals



Smoothing by Barycentric Averaging

Works just like it did with curves

- Compute barycenters around vertex
- Move vertex to barycenter of barycenters



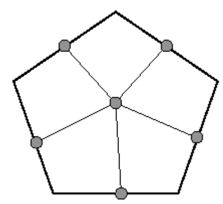
Extraordinary Points

All the points we introduce by quad subdivision are "valence 4"

- They all have 4 edges/faces connected to them

But there are other points with valence ≠ 4

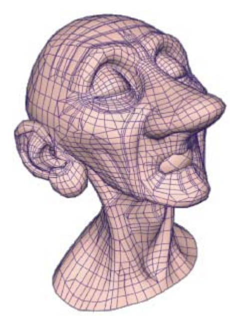
- These are called ***extraordinary points***
- Most of the smoothness analysis action happens here



Subdivision Surfaces

Have become a very successful primitive

- The subject of a lot of recent research
- Naturally ***multiresolution*** representation
- Continuum from polygon meshes to splines



Like spline surfaces

- Represent smooth surfaces well
- Can be built automatically with scanners
- Easier than polygons for manipulation

Demos:

- www.subdivision.org

Pixar's "Geri"

Modeled using subdivision surfaces