CMSC427
Parametric surfaces
(and alternatives)
Generating surfaces

- From equations
- From data
- From curves
  - Extrusion
    - Straight
    - Along path
  - Lathing (rotation)
  - Lofting
Constructive Solid Geometry (CSG)

• Alternative/supplement to parametric shapes

• Vocabulary:
  • Basic set of shapes (sphere, box, cylinder, etc)
  • Set operations on shapes
    • Union
    • Intersection
    • Difference

• Demo
  • Tinkercad
Constructive Solid Geometry (CSG)

- Computer Aided Design (CAD)
  - Precise 3D modeling for industrial design
  - Less freeform, more control and feedback on shapes
  - Often compiled (openScad.org)

```plaintext
cube([2,3,4]);
translate([3,0,0])
{
  cube([2,3,4]);
}

color([1,0,0]) cube([2,3,4]);
translate([3,0,0])
  color([0,1,0]) cube([2,3,4]);
translate([6,0,0])
  color([0,0,1]) cube([2,3,4]);
```
• Stale but interesting ray tracing software
• Scene description language (SDL)
• Pixar’s Renderman

#include "colors.inc"
background { color Cyan }
camera {
    location <0, 2, -3>
    look_at <0, 1, 2>
}
sphere {
    <0, 1, 2>, 2
    texture {
        pigment { color Yellow }
    }
}
light_source {
    <2, 4, -3>
    color White
}
• Support CSG operations

union {
  box { <1, 1, 1>, <2, 2, 2> }
  sphere{ <1.5, 1.5, 1.5>, 1 }
}
• Each segment spans four control points
• Each segment contains four Bernstein polynomials
• Each control point belongs to one Bernstein polynomial
Curved surfaces

Curves
• Described by a 1D series of control points
• A function $x(t)$
• Segments joined together to form a longer curve

Surfaces
• Described by a 2D mesh of control points
• Parameters have two dimensions (two dimensional parameter domain)
• A function $x(u,v)$
• Patches joined together to form a bigger surface
Parametric surface patch

• $\mathbf{x}(u,v)$ describes a point in space for any given $(u,v)$ pair
  • $u,v$ each range from 0 to 1

2D parameter domain
**Parametric surface patch**

- \( x(u,v) \) describes a point in space for any given \((u,v)\) pair
  - \(u, v\) each range from 0 to 1

- Parametric curves
  - For fixed \( u_0 \), have a \( v \) curve \( x(u_0, v) \)
  - For fixed \( v_0 \), have a \( u \) curve \( x(u, v_0) \)
  - For any point on the surface, there is one pair of parametric curves that go through point
• The tangent to a parametric curve is also tangent to the surface
• For any point on the surface, there are a pair of (parametric) tangent vectors
• Note: not necessarily perpendicular to each other
Tangents

Notation

- Tangent along $u$ direction
  \[
  \frac{\partial \mathbf{x}}{\partial u}(u, v) \quad \text{or} \quad \frac{\partial}{\partial u} \mathbf{x}(u, v) \quad \text{or} \quad \mathbf{x}_u(u, v)
  \]

- Tangent along $v$ direction
  \[
  \frac{\partial \mathbf{x}}{\partial v}(u, v) \quad \text{or} \quad \frac{\partial}{\partial v} \mathbf{x}(u, v) \quad \text{or} \quad \mathbf{x}_v(u, v)
  \]

- Tangents are vector valued functions, i.e., vectors!
Surface normal

- Cross product of the two tangent vectors
  \[ \mathbf{x}_u(u, v) \times \mathbf{x}_v(u, v) \]
- Order matters (determines normal orientation)
- Usually, want unit normal
  - Need to normalize by dividing through length
Bilinear patch

- Control mesh with four points $p_0, p_1, p_2, p_3$
- Compute $x(u, v)$ using a two-step construction
Bilinear patch (step 1)

• For a given value of $u$, evaluate the linear curves on the two $u$-direction edges

• Use the same value $u$ for both:

\[ \mathbf{q}_1 = \text{Lerp}(u, \mathbf{p}_2, \mathbf{p}_3) \]

\[ \mathbf{q}_0 = \text{Lerp}(u, \mathbf{p}_0, \mathbf{p}_1) \]
Bilinear patch (step 2)

- Consider that $q_0$, $q_1$ define a line segment
- Evaluate it using $v$ to get $x$

$$x = \text{Lerp}(v, q_0, q_1)$$
Bilinear patch

• Combining the steps, we get the full formula

\[ x(u,v) = \text{Lerp}(v, \text{Lerp}(u, p_0, p_1), \text{Lerp}(u, p_2, p_3)) \]
Bilinear patch

• Try the other order
• Evaluate first in the $v$ direction

\[ r_0 = \text{Lerp}(v, p_0, p_2) \quad r_1 = \text{Lerp}(v, p_1, p_3) \]
Bilinear patch

- Consider that $\mathbf{r}_0, \mathbf{r}_1$ define a line segment
- Evaluate it using $u$ to get $\mathbf{x}$

$$\mathbf{x} = \text{Lerp}(u, \mathbf{r}_0, \mathbf{r}_1)$$
• The full formula for the $v$ direction first:

$$\mathbf{x}(u,v) = \text{Lerp}(u, \text{Lerp}(v, \mathbf{p}_0, \mathbf{p}_2), \text{Lerp}(v, \mathbf{p}_1, \mathbf{p}_3))$$
Bilinear patch

• It works out the same either way!

\[ x(u,v) = \text{Lerp}(v, \text{Lerp}(u,p_0,p_1), \text{Lerp}(u,p_2,p_3)) \]

\[ x(u,v) = \text{Lerp}(u, \text{Lerp}(v,p_0,p_2), \text{Lerp}(v,p_1,p_3)) \]
Bilinear patch

• Visualization
Bilinear patches

- Weighted sum of control points
  \[ x(u, v) = (1-u)(1-v)p_0 + u(1-v)p_1 + (1-u)v p_2 + uv p_3 \]

- Bilinear polynomial
  \[ x(u, v) = (p_0 - p_1 - p_2 + p_3)uv + (p_1 - p_0)u + (p_2 - p_0)v + p_0 \]

- Matrix form exists, too
Properties

- Interpolates the control points
- The boundaries are straight line segments
- If all 4 points of the control mesh are co-planar, the patch is flat
- If the points are not coplanar, get a curved surface
  - saddle shape, AKA hyperbolic paraboloid
- The parametric curves are all straight line segments!
  - a (doubly) ruled surface: has (two) straight lines through every point
Bicubic Bézier patch

• Grid of 4x4 control points, $p_0$ through $p_{15}$
• Four rows of control points define Bézier curves along $u$
  $p_0, p_1, p_2, p_3; p_4, p_5, p_6, p_7; p_8, p_9, p_{10}, p_{11}; p_{12}, p_{13}, p_{14}, p_{15}$
• Four columns define Bézier curves along $v$
  $p_0, p_4, p_8, p_{12}; p_1, p_6, p_9, p_{13}; p_2, p_6, p_{10}, p_{14}; p_3, p_7, p_{11}, p_{15}$
Bicubic Bézier patch (step 1)

- Evaluate four $u$-direction Bézier curves at $u$
- Get intermediate points $q_0 \ldots q_3$

$q_0 = \text{Bez}(u, p_0, p_1, p_2, p_3)$
$q_1 = \text{Bez}(u, p_4, p_5, p_6, p_7)$
$q_2 = \text{Bez}(u, p_8, p_9, p_{10}, p_{11})$
$q_3 = \text{Bez}(u, p_{12}, p_{13}, p_{14}, p_{15})$
Bicubic Bézier patch (step 2)

- Points \( q_0 \ldots q_3 \) define a Bézier curve
- Evaluate it at \( v \)

\[
x(u, v) = Bez(v, q_0, q_1, q_2, q_3)
\]
Bicubic Bézier patch

• Same result in either order (evaluate $u$ before $v$ or vice versa)

\[
\begin{align*}
q_0 &= \text{Bez}(u, p_0, p_1, p_2, p_3) \\
q_1 &= \text{Bez}(u, p_4, p_5, p_6, p_7) \\
q_2 &= \text{Bez}(u, p_8, p_9, p_{10}, p_{11}) \\
q_3 &= \text{Bez}(u, p_{12}, p_{13}, p_{14}, p_{15}) \\
x(u, v) &= \text{Bez}(v, q_0, q_1, q_2, q_3)
\end{align*}
\]

\[
\begin{align*}
r_0 &= \text{Bez}(v, p_0, p_4, p_8, p_{12}) \\
r_1 &= \text{Bez}(v, p_1, p_5, p_9, p_{13}) \\
r_2 &= \text{Bez}(v, p_2, p_6, p_{10}, p_{14}) \\
r_3 &= \text{Bez}(v, p_3, p_7, p_{11}, p_{15}) \\
x(u, v) &= \text{Bez}(u, r_0, r_1, r_2, r_3)
\end{align*}
\]
Tensor product formulation

• Corresponds to **weighted average** formulation

• Construct two-dimensional weighting function as product of two one-dimensional functions
  • Bernstein polynomials $B_i, B_j$ as for curves

• Same **tensor product** construction applies to higher order Bézier and NURBS surfaces

\[ \mathbf{x}(u, v) = \sum_i \sum_j p_{i,j} B_i(u) B_j(v) \]
Bicubic Bézier patch: properties

• Convex hull: any point on the surface will fall within the convex hull of the control points
• Interpolates 4 corner points
• Approximates other 12 points, which act as “handles”
• The boundaries of the patch are the Bézier curves defined by the points on the mesh edges
• The parametric curves are all Bézier curves
Tangents of Bézier patch

- Remember parametric curves \( \mathbf{x}(u, v_0), \mathbf{x}(u_0, v) \) where \( v_0, u_0 \) is fixed
- Tangents to surface = tangents to parametric curves
- Tangents are partial derivatives of \( \mathbf{x}(u, v) \)
- Normal is cross product of the tangents
Tangents of Bézier patch

\[ q_0 = \text{Bez}(u, p_0, p_1, p_2, p_3) \]
\[ q_1 = \text{Bez}(u, p_4, p_5, p_6, p_7) \]
\[ q_2 = \text{Bez}(u, p_8, p_9, p_{10}, p_{11}) \]
\[ q_3 = \text{Bez}(u, p_{12}, p_{13}, p_{14}, p_{15}) \]

\[ \frac{\partial x}{\partial v}(u, v) = \text{Bez}'(v, q_0, q_1, q_2, q_3) \]

\[ r_0 = \text{Bez}(v, p_0, p_4, p_8, p_{12}) \]
\[ r_1 = \text{Bez}(v, p_1, p_5, p_9, p_{13}) \]
\[ r_2 = \text{Bez}(v, p_2, p_6, p_{10}, p_{14}) \]
\[ r_3 = \text{Bez}(v, p_3, p_7, p_{11}, p_{15}) \]

\[ \frac{\partial x}{\partial u}(u, v) = \text{Bez}'(u, r_0, r_1, r_2, r_3) \]
Tessellating a Bézier patch

- **Uniform tessellation** is most straightforward
  - Evaluate points on uniform grid of $u$, $v$ coordinates
  - Compute tangents at each point, take cross product to get per-vertex normal
  - Draw triangle strips (several choices of direction)

- **Adaptive tessellation/recursive subdivision**
  - Potential for “cracks” if patches on opposite sides of an edge divide differently
  - Tricky to get right, but can be done
Piecewise Bézier surface

- Lay out grid of adjacent meshes of control points
- For $C^0$ continuity, must share points on the edge
  - Each edge of a Bézier patch is a Bézier curve based only on the edge mesh points
  - So if adjacent meshes share edge points, the patches will line up exactly
- But we have a crease…
\( C^1 \) continuity

- Want parametric curves that cross each edge to have \( C^1 \) continuity
  - Handles must be equal-and-opposite across edge

[http://www.spiritone.com/~english/cyclopedia/patches.html]
Modeling with Bézier patches

- Original Utah teapot specified as Bézier Patches
Subdivision surfaces

• Goal
  • Create smooth surfaces from small number of control points, like splines
  • More flexibility for the topology of the control points (not restricted to quadrilateral grid)

• Idea
  • Start with initial coarse polygon mesh
  • Create smooth surface recursively by
    1. Splitting (subdividing) mesh into finer polygons
    2. Smoothing the vertices of the polygons
    3. Repeat from 1.
Subdivision surfaces

http://en.wikipedia.org/wiki/Catmull%E2%80%93Clark_subdivision_surface

Input mesh → Subdivision & smoothing → Subdivision & smoothing → Subdivision & smoothing

Limit surface
Loop subdivision

- Subdivision
  - Split each triangle into four

- Smoothing
  - New vertex positions as weighted average of neighbors
  - Different cases

Cases for $\beta$:

$$\beta = \begin{cases} 
\frac{3}{8n} & n > 3 \\
\frac{3}{16} & n = 3 
\end{cases}$$

Number of neighbors $n$

http://en.wikipedia.org/wiki/Loop_subdivision_surface

http://graphics.stanford.edu/~mdfisher/subdivision.html
Subdividing sphere

- Divide triangle $ABC$ into four new triangles
- Extend rays to sphere surface to compute new vertices
Subdivision surfaces

• Arbitrary mesh of control points
• Arbitrary topology or connectivity
  • Not restricted to quadrilateral topology
  • No global $u, v$ parameters
• Work by recursively subdividing mesh faces
• Used in particular for character animation
  • One surface rather than collection of patches
  • Can deform geometry without creating cracks