Surfaces

• Many objects we want to model are not flat:
  – Cars, animals, plants, buildings.
Accuracy/Space Trade-off

Problem

• Piecewise linear approximations require many pieces to look good (realistic, smooth, etc.).
• Set of individual surface points would take large amounts of storage.

Solution

• Higher-order formulae for coordinates on surface.
• If a simple formula won’t work, subdivide surface into pieces that can be represented by simple formulae.
• May still be an approx., but uses much less storage.
• Downside: harder to specify and render.

Surface Representations

• **Parametric**: \((x,y,z) = (f(u,v), g(u,v), h(u,v))\)
  – e.g. plane, sphere, cylinder, torus, bi-cubic surface, swept surface
  – parametric functions let you *iterate* over the surface by incrementing \(u\) and \(v\)
  – great for making polygon meshes, etc
  – complex for intersections: ray/surface, point-inside-boundary, etc

• **Implicit**: \(F(x,y,z) = 0\)
  – e.g. plane, sphere, cylinder, quadric, torus, blobby models
  – terrible for iterating over the surface
  – great for intersections, morphing
Examples

**Parametric: Ellipsoid**

\[
\begin{align*}
x &= r_x \cos \phi \cos \theta \\
y &= r_y \cos \phi \sin \theta \\
z &= r_z \sin \phi
\end{align*}
\]

Implicit functions:
Quadrics and other

Swept Surfaces

Obtained by sweeping generator entities along director entities.
Rotational Surfaces

Generated by rotating a curve about an axis.

Every point of the generating curve describes a circle whose supporting plane lies orthogonally to the Axis.

Meshes

We can approximate a surface with a *polygonal mesh*. 
Curved Surfaces

• Remember the overview of curves:
  – Described by a series of control points.
  – A function $Q(t)$.
  – Segments joined together to form a longer curve.

• Same for surfaces, but now two dimensions
  – Described by a mesh of control points.
  – A function $S(u,v)$.
  – *Patches* joined together to form a bigger surface.

Parametric Surface Patch

• $S(u,v)$ 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 $S(u_0,v)$.
  – For fixed $v_0$, have a $u$ curve $S(u,v_0)$.
  – For any point on the surface, there are a pair of parametric curves that go through point.
Polynomial Surface Patches

- $S(s,t)$ is typically polynomial in both $s$ and $t$
  
  - **Bilinear:**
    
    $S(s,t) = ast + bs + ct + d$
    
    $S(s,t) = (at + b)s + (ct + d)$ — hold $t$ constant $\Rightarrow$ linear in $s$
    
    $S(s,t) = (as + c)t + (bs + d)$ — hold $s$ constant $\Rightarrow$ linear in $t$

  - **Bicubic:**
    
    $S(s,t) = as^3t^3 + bs^3t^2 + cs^3t^1 + ds^3 + es^3t^3 + fs^3t^2 + gs^3t + hs^3$
    
    $+ as^2t^3 + bs^2t^2 + cs^2t + ds^2 + es^2t^3 + fs^2t^2 + gs^2t + hs^2$
    
    $S(s,t) = (at^3 + bt^2 + ct + d)s^3 + (et^3 + ft^2 + gt + h)s^2$ — hold $t$ constant $\Rightarrow$ cubic in $s$
    
    $+ (as^3 + bs^2 + cs + d)s + (ms^3 + ns^2 + os + p)$

    $S(s,t) = (as^3 + bs^3 + cs + m)t^3 + (bs^3 + fs^2 + js + n)t^2$ — hold $s$ constant $\Rightarrow$ cubic in $t$
    
    $+ (cs^3 + gs^2 + ks + o)t + (ds^3 + hs^2 + ls + p)$

Bilinear Patch

- A **bilinear patch** is defined by a control mesh with four points $p_0$, $p_1$, $p_2$, $p_3$ defining a (possibly-non-planar) quadrilateral.

- Compute $S(u,v)$ 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:

\[ q_0 = \text{Lerp}(u, p_0, p_1) \]
\[ q_1 = \text{Lerp}(u, p_2, p_3) \]

Bilinear Patch (step 2)

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

\[ S = \text{Lerp}(v, q_0, q_1) \]
Bilinear Patch (full)

• Combining the steps, we get the full formula

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

Bilinear Patch (either order)

• It works out the same either way!

\[ S(u, v) = \text{Lerp}(v, \text{Lerp}(u, p_0, p_1), \text{Lerp}(u, p_2, p_3)) \]
\[ S(u, v) = \text{Lerp}(u, \text{Lerp}(v, p_0, p_2), \text{Lerp}(v, p_1, p_3)) \]
Properties of the bilinear patch

– Interpolates the control points.
– The boundaries are straight line segments connecting the control points.
– If the all 4 points of the control mesh are co-planar, the patch is flat.
– If the points are not coplanar, get a curved surface.
– The parametric curves are all straight line segments:
  • Is a (doubly) ruled surface: has (two) straight lines through every point.

Ruled Surfaces

• Linear interpolation between 2 curves
  – All point lie in one line

\[ f(s,t) = (1-t)f_1(s) + t f_2(s) \]

Particular case

Cylinder:
\[ f(s,t) = f_1(s) + t d \]
\[ f_1(s) = (\cos s, \sin s, 0) \]

Cone:
\[ f(s,t) = (1-t)f_1 + t f_2(s) \]
\[ f_2(s) = (\cos s, \sin s, 0) \]
Coons patches

- Interpolation between 4 curves
- Build a ruled surface between pairs of curves

\[ f_1(s, t) = (1-t)f(s, 0) + t f(s, 1) \]
\[ f_2(s, t) = (1-s)f(0, t) + s f(1, t) \]
Coons patches

- "Correct" surface to make boundaries match
  - Create a linear interpolation surface between the 4 extremes:
    \[ f_3(s,t) = (1-s)t \cdot f(0,1) + s(1-t) \cdot f(1,0) + st \cdot f(1,1) \] (bilinear patch).
  - Combine surface as \( f_1 + f_2 - f_3 \)
    \[ f_1(s,t) + f_2(s,t) - ((1-s)(1-t) \cdot f(0,0) + (1-s)t \cdot f(0,1) + s(1-t) \cdot f(1,0) + st \cdot f(1,1)) \]

Properties of Coons patches

- Interpolate arbitrary boundaries
- Smoothness of surface equivalent to minimum smoothness of boundary curves
- Don’t provide higher continuity across boundaries
Bézier Control Mesh

- A bicubic patch has a grid of 4x4 control points:
  \[ P_{0,0}, \ldots, P_{0,3} \]
  \[
  \ldots 
  \]
  \[
  \ldots 
  \]
  \[ P_{3,0}, \ldots, P_{3,3}. \]
- Defines four Bézier curves along \( u \) and four Bézier curves along \( v \).
- Evaluate using same approach as bilinear.

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.
Cubic Bezier Surfaces: Algebraic Formulation

Cubic Bezier curves can be extended to surfaces on unit squares:

\[
S(u, v) = \sum_{i=0}^{3} \sum_{j=0}^{3} B_i^3(u) B_j^3(v) P_{ij}
\]

\[
B_i^3(t) = \binom{3}{i} t^i (1-t)^{3-i}
\]

Building a surface from Bézier patches

Building complex surfaces by putting together Bézier patches.

- Lay out grid of adjacent meshes.
- For C₀ 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.
C¹ continuity across Bézier edges

• We want the parametric curves that cross each edge to have C¹ continuity:
  – So the handles must be equal-and-opposite across the edge.

B-spline patches

For the same reason as using B-spline curves:

  – More uniform behavior.
  – Better mathematical properties.
  – Doesn’t interpolate any control points.
B-Spline Surfaces

A B-spline surface \( S(u,v) \), is defined by:

\[
S(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{n'} B_{i,d}(u) B_{j,d'}(v) P_{i,j}
\]

where:

- The \( P_{ij} \) are the \((n+1) \times (n'+1)\) control points.
- \( d \) and \( d' \) are the orders in the \( u \) and \( v \) directions.
- We have two non-decreasing knot sequences of parameters \( u_0, \ldots, u_{n+d} \) and \( v_0, \ldots, v_{n'+d'} \).
- \( B_{i,d} \) are the uniform B-Spline basis or blending functions of degree \( d-1 \).

NURBS Surfaces

\[
S(u,v) = \frac{\sum_{i=0}^{n} \sum_{j=0}^{n'} w_{i,j} B_{i,d}(u) B_{j,d'}(v) P_{i,j}}{\sum_{i=0}^{n} \sum_{j=0}^{n'} w_{i,j} B_{i,d}(u) B_{j,d'}(v)}
\]

- Can take on more shapes:
  - conic sections.
- Can blend, merge, … .
- Still has rectangular topology.
Surface examples

Subdivision Surfaces

- Defined by a control mesh and a recursive subdivision procedure
- Arbitrary mesh, not rectangular topology:
  - No $u,v$ parameters.
- Can make surfaces with arbitrary topology or connectivity.
- Work by recursively subdividing mesh faces:
  - Per-vertex annotation for weights, corners, creases.
- Good for interactive design
  - Used in particular for character animation:
  - One surface rather than collection of patches.
  - Can deform geometry without creating cracks.
The Basic Idea

- In each iteration
  - Refine a control net (mesh)
  - Increases the number of vertices / faces
- The mesh vertices converges to a limit surface
- Each subdivision scheme has:
  - Rules to calculate the locations of new vertices.
  - A method to generate the new net topology.