Announcements

- Homework assignment #5 due Friday, Nov 5
- Phi’s office hour this Friday, 1-2pm
- Midterm grading completed
- Midterm review:
  - Exams returned
  - Presentation of results
  - Exams recollected
Rational Curves

- Weight causes point to “pull” more (or less)
- Can model circles with proper points and weights,
- Below: rational quadratic Bézier curve (three control points)
B-Splines

- B as in Basis-Splines
- Basis is blending function
- Resolves problem with Bézier splines:
  - Control points have global scope (a change in one control point affects the global shape of the curve)
- Difference to Bézier blending function:
  - B-spline blending function can be zero outside a particular range (limits scope over which a control point has influence)
- B-Spline is defined by control points and range in which each control point is active. Ranges are specified through knot vector
NURBS

- **Non Uniform Rational B-Splines**
- Generalization of Bézier curves
- Invariant under projective transformation: if two objects touch in object space, they will still touch after projection
- Very similar to B-Splines, but with modifications made to accommodate points specified using homogeneous coordinates
- Can exactly model conic sections (circles, ellipses)
- OpenGL support: see gluNurbsCurve
- [http://mathworld.wolfram.com/NURBSCurve.html](http://mathworld.wolfram.com/NURBSCurve.html)
Lecture Overview

- Bi-linear patch
- Bi-cubic Bézier patch
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

- $x(u,v)$ 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 are a pair of parametric curves through that point

2D parameter domain
Tangents

- 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: these vectors are not necessarily perpendicular to each other.
Tangents

- Notation:
  - The tangent along a $u$ curve, AKA the tangent in the $u$ direction, is written as:
    \[
    \frac{\partial \mathbf{x}}{\partial u}(u, v) \text{ or } \frac{\partial}{\partial u} \mathbf{x}(u, v) \text{ or } \mathbf{x}_u(u, v)
    \]
  - The tangent along a $v$ curve, AKA the tangent in the $v$ direction, is written as:
    \[
    \frac{\partial \mathbf{x}}{\partial v}(u, v) \text{ or } \frac{\partial}{\partial v} \mathbf{x}(u, v) \text{ or } \mathbf{x}_v(u, v)
    \]

- Note that each of these is a vector-valued function:
  - At each point $\mathbf{x}(u, v)$ on the surface, we have tangent vectors $\frac{\partial}{\partial u} \mathbf{x}(u, v)$ and $\frac{\partial}{\partial v} \mathbf{x}(u, v)$
Surface Normal

- Normal is cross product of the two tangent vectors
- Order matters!

\[ \mathbf{n}(u, v) = \frac{\partial \mathbf{x}}{\partial u}(u, v) \times \frac{\partial \mathbf{x}}{\partial v}(u, v) \]

Typically we are interested in the unit normal, so we need to normalize

\[ \mathbf{n}^*(u, v) = \frac{\frac{\partial \mathbf{x}}{\partial u}(u, v) \times \frac{\partial \mathbf{x}}{\partial v}(u, v)}{\left| \frac{\partial \mathbf{x}}{\partial u}(u, v) \times \frac{\partial \mathbf{x}}{\partial v}(u, v) \right|} \]
Bilinear Patch

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

$$x = Lerp(v, q_0, q_1)$$
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

$$r_0 = Lerp(v, p_0, p_2) \quad r_1 = Lerp(v, p_1, p_3)$$
Consider that \( \mathbf{r}_0, \mathbf{r}_1 \) define a line segment

Evaluate it using \( u \) to get \( \mathbf{x} \)

\[
\mathbf{x} = \text{Lerp}(u, \mathbf{r}_0, \mathbf{r}_1)
\]
Bilinear Patch

- The full formula for the $v$ direction first:

$$x(u, v) = \text{Lerp}(u, \text{Lerp}(v, p_0, p_2), \text{Lerp}(v, p_1, p_3))$$
Bilinear Patch

- Patch geometry is independent of the order of $u$ and $v$

\[
x(u, v) = \text{Lerp}(v, \text{Lerp}(u, p_0, p_1), \text{Lerp}(u, p_2, p_3))
\]
\[
x(u, v) = \text{Lerp}(u, \text{Lerp}(v, p_0, p_2), \text{Lerp}(v, p_1, p_3))
\]
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 co-planar, we get a curved surface
  - saddle shape (hyperbolic paraboloid)
- *The parametric curves are all straight line segments!*
  - a (doubly) ruled surface: has (two) straight lines through every point

- Not terribly useful as a modeling primitive
Lecture Overview

- Bi-linear patch
- Bi-cubic Bézier patch
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 $v$
  - $p_0, p_4, p_8, p_{12}; p_1, p_5, p_9, p_{13}; p_2, p_6, p_{10}, p_{14}; p_3, p_7, p_{11}, p_{15}$
Bézier Patch (Step 1)

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

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

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

\[
x(u, v) = \text{Bez}(v, q_0, q_1, q_2, q_3)
\]
Bézier Patch

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

$$q_0 = \text{Bez}(u, p_0, p_1, p_2, p_3) \quad r_0 = \text{Bez}(v, p_0, p_4, p_8, p_{12})$$

$$q_1 = \text{Bez}(u, p_4, p_5, p_6, p_7) \quad r_1 = \text{Bez}(v, p_1, p_5, p_9, p_{13})$$

$$q_2 = \text{Bez}(u, p_8, p_9, p_{10}, p_{11}) \quad r_2 = \text{Bez}(v, p_2, p_6, p_{10}, p_{14})$$

$$q_3 = \text{Bez}(u, p_{12}, p_{13}, p_{14}, p_{15}) \quad r_3 = \text{Bez}(v, p_3, p_7, p_{11}, p_{15})$$

$$x(u, v) = \text{Bez}(v, q_0, q_1, q_2, q_3) \quad x(u, v) = \text{Bez}(u, r_0, r_1, r_2, r_3)$$
**Bézier Patch: Matrix Form**

\[
U = \begin{bmatrix} u^3 \\ u^2 \\ u \\ 1 \end{bmatrix}, \quad V = \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}, \quad B_{\text{Bez}} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} = B_{\text{Bez}}^T
\]

\[
C_x = B_{\text{Bez}}^T G_x B_{\text{Bez}}, \quad C_y = B_{\text{Bez}}^T G_y B_{\text{Bez}}, \quad C_z = B_{\text{Bez}}^T G_z B_{\text{Bez}}
\]

\[
G_x = \begin{bmatrix} p_{0x} & p_{1x} & p_{2x} & p_{3x} \\ p_{4x} & p_{5x} & p_{6x} & p_{7x} \\ p_{8x} & p_{9x} & p_{10x} & p_{11x} \\ p_{12x} & p_{13x} & p_{14x} & p_{15x} \end{bmatrix}, \quad G_y = \cdots, \quad G_z = \cdots
\]

\[
x(u, v) = \begin{bmatrix} V^T C_x U \\ V^T C_y U \\ V^T C_z U \end{bmatrix}
\]
Bézier Patch: Matrix Form

- \( C_x \) stores the coefficients of the bicubic equation for \( x \)
- \( C_y \) stores the coefficients of the bicubic equation for \( y \)
- \( C_z \) stores the coefficients of the bicubic equation for \( z \)
- \( G_x \) stores the geometry (\( x \) components of the control points)
- \( G_y \) stores the geometry (\( y \) components of the control points)
- \( G_z \) stores the geometry (\( z \) components of the control points)
- \( B_{Bez} \) is the basis matrix (Bézier basis)
- \( U \) and \( V \) are the vectors formed from the powers of \( u \) and \( v \)

- Compact notation
- Leads to efficient method of computation
- Can take advantage of hardware support for 4x4 matrix arithmetic
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 a 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 a Bézier patch

\[
\begin{align*}
q_0 &= \text{Bez}(u, p_0, p_1, p_2, p_3) \\
q_1 &= \text{Bez}(u, p_4, p_5, p_6, p_7) \\
q_2 &= \text{Bez}(u, p_8, p_9, p_{10}, p_{11}) \\
q_3 &= \text{Bez}(u, p_{12}, p_{13}, p_{14}, p_{15}) \\
\frac{\partial x}{\partial v}(u, v) &= \text{Bez}'(v, q_0, q_1, q_2, q_3) \\
\frac{\partial x}{\partial u}(u, v) &= \text{Bez}'(u, r_0, r_1, r_2, r_3)
\end{align*}
\]
Tessellating a Bézier patch

- **Uniform tessellation is most straightforward**
  - Evaluate points on a 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

- We want the parametric curves that cross each edge to have C\(^1\) continuity
  - So the handles must be equal-and-opposite across the edge:

http://www.spiritone.com/~english/cyclopedia/patches.html
Modeling With Bézier Patches

- Original Utah teapot was specified with Bézier Patches
Next Lecture

- Advanced surface modeling
- Advanced shader programming