Surfaces

Tensor product Bézier surfaces

Given a grid of control points $V_{ij}$ forming a control net, construct a surface $S(u,v)$ by:

- treating rows of $V$ as control points for curves $V_{i0}(u), \ldots, V_{in}(u)$.
- treating $V_{i0}(u), \ldots, V_{in}(u)$ as control points for a curve parameterized by $v$.

Building surfaces from curves

Let the geometry vector vary by a second parameter $v$:

$$S(u,v) = U \cdot M \cdot G(v)$$

$$G_i(v) = V \cdot M \cdot g$$

$g_i = [g_{i1}, g_{i2}, g_{i3}, g_{i4}]^T$

Reading

Foley et al., Section 11.3

Recommended:

Geometry matrices

By transposing the geometry curve we get:

\[ G_i(v)^T = (V \cdot M \cdot g_i)^T \]
\[ = g_i^T M^T \cdot V^T \]
\[ = [v_1 \ v_2 \ v_3 \ v_4] \cdot M^T \cdot V^T \]

Combining

And

\[ S(u, v) = U \cdot M \]
\[ G_i(v) \]
\[ G_i(v) \]
\[ G_i(v) \]

We get

\[ S(u, v) = U \cdot M \]
\[ [v_1 \ v_2 \ v_3 \ v_4] \cdot M^T \cdot V^T \]

Tensor product surfaces, cont.

Let’s walk through the steps:

- Control net
- Control curves in \( u \)
- Control polygon at \( u=1/2 \)
- Curve at \( \delta(1/2, v) \)

Which control points are interpolated by the surface?

Bezier Blending Functions

a.k.a. Bernstein polynomials

\[ Q(t) = \begin{bmatrix} t^3 & t^2 & t \end{bmatrix} \]
\[ = B_3(t) \]
\[ B_3(t) \]

\[ B_0(t) \]
\[ B_1(t) \]
\[ B_2(t) \]
\[ B_3(t) \]

1 0 0 0

1 3 -3 1

3 -6 3 0

-3 3 0 0

\[ B_3(t) \]

1 0 0 0
Matrix form

Tensor product surfaces can be written out explicitly:

\[ S(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} V_i B_i^r(u) B_j^s(v) \]

\[ = \begin{bmatrix} v^3 & v^2 & v & 1 \end{bmatrix} M_{\text{Bezier}} \begin{bmatrix} u^3 \\ u^2 \\ u \\ 1 \end{bmatrix} \]

Tensor product B-spline surfaces

As with spline curves, we can piece together a sequence of Bézier surfaces to make a spline surface. If we enforce C2 continuity and local control, we get B-spline curves:

- treat rows of \( B \) as control points to generate Bézier control points in \( u \).
- treat Bézier control points in \( u \) as B-spline control points in \( v \).
- treat B-spline control points in \( v \) to generate Bézier control points in \( u \).
Trimmed NURBS surfaces

Uniform B-spline surfaces are a special case of NURBS surfaces. Sometimes, we want to have control over which parts of a NURBS surface get drawn.

For example:

We can do this by trimming the u-v domain.

- Define a closed curve in the u-v domain (a trim curve)
- Do not draw the surface points inside of this curve.

It’s really hard to maintain continuity in these regions, especially while animating.

Surfaces of revolution

Idea: rotate a 2D profile curve around an axis.

What kinds of shapes can you model this way?

Variations

Several variations are possible:

- Scale \( C(u) \) as it moves, possibly using length of \( T(v) \) as a scale factor.
- Morph \( C(u) \) into some other curve \( C'(u) \) as it moves along \( T(v) \).
- …

Constructing surfaces of revolution

Given: A curve \( C(u) \) in the yz-plane:

\[
\begin{bmatrix}
0 \\
c_z(u) \\
c_y(u) \\
1
\end{bmatrix}
\]

Let \( R_x(\theta) \) be a rotation about the x-axis.

Find: A surface \( S(u,v) \) which is \( C(u) \) rotated about the z-axis.

\[
S(u,v) = R_x(v) \cdot C(u)
\]
**General sweep surfaces**

The *surface of revolution* is a special case of a *swept surface*.

**Idea:** Trace out surface \( S(u, v) \) by moving a *profile curve* \( C(u) \) along a *trajectory curve* \( T(v) \).

\[
S(u, v) = T(T(v)) \cdot C(u)
\]

More specifically:

- Suppose that \( C(u) \) lies in an \((x, y, z)\) coordinate system with origin \( O_c \).
- For every point along \( T(v) \), lay \( C(u) \) so that \( O_c \) coincides with \( T(v) \).

**Orientation**

The big issue:

- How to orient \( C(u) \) as it moves along \( T(v) \)?

Here are two options:

1. **Fixed (or static):** Just translate \( O_c \) along \( T(v) \).
2. Moving. Use the *Frenet frame* of \( T(v) \).

- Allows smoothly varying orientation.
- Permits surfaces of revolution, for example.

**Frenet frames**

Motivation: Given a curve \( T(v) \), we want to attach a smoothly varying coordinate system.

To get a 3D coordinate system, we need 3 independent direction vectors.

\[
\begin{align*}
\hat{t}(v) &= \text{normalize} T'(v) \\
\hat{b}(v) &= \text{normalize} \hat{t}(v) \times \hat{n}(v) \\
\hat{n}(v) &= \hat{b}(v) \times \hat{t}(v)
\end{align*}
\]

As we move along \( T(v) \), the Frenet frame \( \hat{t}, \hat{b}, \hat{n} \) varies smoothly.

**Frenet swept surfaces**

Orient the profile curve \( C(u) \) using the Frenet frame of the trajectory \( T(v) \):

1. Put \( C(u) \) in the *normal plane* \( nb \).
2. Place \( O_c \) on \( T(v) \).
3. Align \( x \) for \( C(u) \) with \( n \).
4. Align \( y \) for \( C(u) \) with \( b \).

If \( T(v) \) is a circle, you get a surface of revolution exactly?
Summary

What to take home:

- How to construct tensor product Bézier surfaces
- How to construct tensor product B-spline surfaces
- Surfaces of revolution
- Construction of swept surfaces from a profile and trajectory curve
  - With a fixed frame
  - With a Frenet frame