

## Parametric surfaces

1

## Reading

Required:

- ♦ Angel readings for “Parametric Curves” lecture, with emphasis on 11.1.2, 11.1.3, 11.1.5, 11.6.2, 11.7.3, 11.9.4.

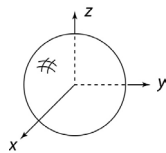
Optional

- ♦ Bartels, Beatty, and Barsky. *An Introduction to Splines for use in Computer Graphics and Geometric Modeling*, 1987.

2

## Mathematical surface representations

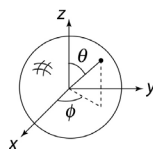
- ♦ Explicit  $z=f(x,y)$  (a.k.a., a “height field”)
  - what if the curve isn’t a function, like a sphere?



- ♦ Implicit  $g(x,y,z) = 0$

- ♦ Parametric  $S(u,v)=(x(u,v),y(u,v),z(u,v))$

- For the sphere:
  - $x(u,v) = r \cos 2\pi v \sin \pi u$
  - $y(u,v) = r \sin 2\pi v \sin \pi u$
  - $z(u,v) = r \cos \pi u$



As with curves, we’ll focus on parametric surfaces.

3

## Surfaces of revolution

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

What kinds of shapes can you model this way?

4

## Constructing surfaces of revolution

**Given:** A curve  $C(u)$  in the  $xy$ -plane:

$$C(u) = \begin{bmatrix} c_x(u) \\ c_y(u) \\ 0 \\ 1 \end{bmatrix}$$

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

**Find:** A surface  $S(u,v)$  which is  $C(u)$  rotated about the  $x$ -axis.

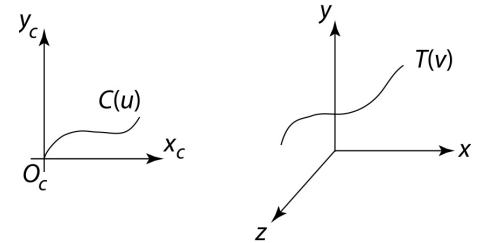
**Solution:**

5

## 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**  $T(v)$ .



More specifically:

- Suppose that  $C(u)$  lies in an  $(x_c, y_c)$  coordinate system with origin  $O_c$ .
- For every point along  $T(v)$ , lay  $C(u)$  so that  $O_c$  coincides with  $T(v)$ .

6

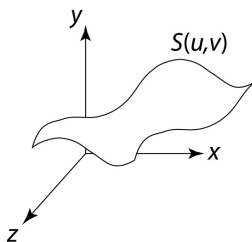
## Orientation

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  $T(v)$ .



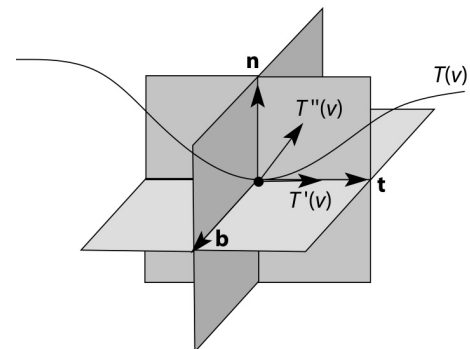
2. Moving. Use the **Frenet frame** of  $T(v)$ .

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

7

## 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.

$$\begin{aligned} \mathbf{t}(v) &= \text{normalize}[T'(v)] \\ \mathbf{b}(v) &= \text{normalize}[T'(v) \times T''(v)] \\ \mathbf{n}(v) &= \mathbf{b}(v) \times \mathbf{t}(v) \end{aligned}$$

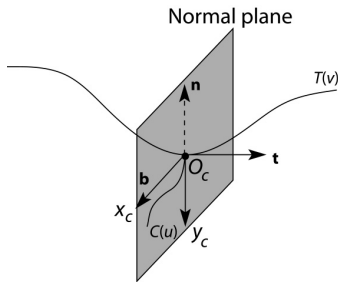
As we move along  $T(v)$ , the Frenet frame  $(t, b, n)$  varies smoothly.

8

## Frenet swept surfaces

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

- ♦ Put  $C(u)$  in the **normal plane**.
- ♦ Place  $O_c$  on  $T(v)$ .
- ♦ Align  $x_c$  for  $C(u)$  with  $\mathbf{b}$ .
- ♦ Align  $y_c$  for  $C(u)$  with  $-\mathbf{n}$ .



If  $T(v)$  is a circle, you get a surface of revolution exactly!

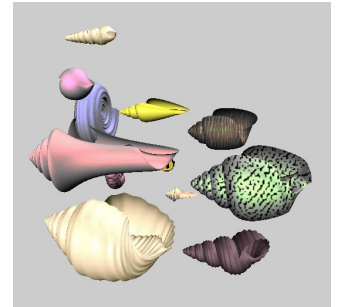
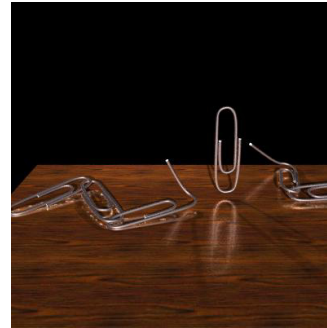
What happens at inflection points, i.e., where curvature goes to zero?

9

## Variations

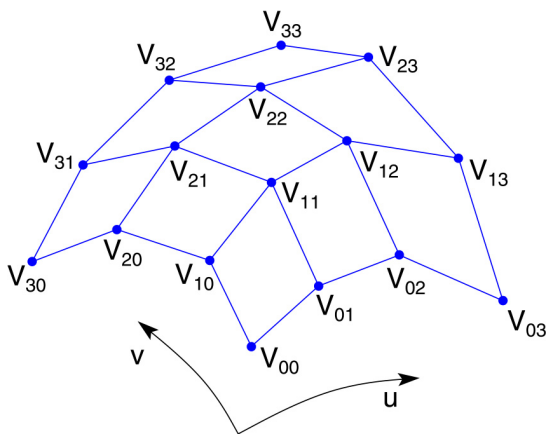
Several variations are possible:

- ♦ Scale  $C(u)$  as it moves, possibly using length of  $T(v)$  as a scale factor.
- ♦ Morph  $C(u)$  into some other curve  $\tilde{C}(u)$  as it moves along  $T(v)$ .
- ♦ ...



10

## Tensor product Bézier surfaces



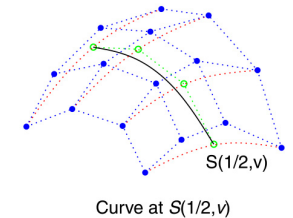
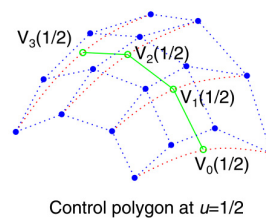
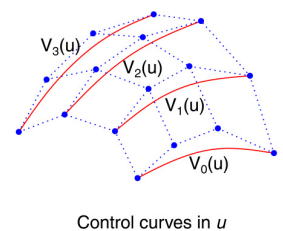
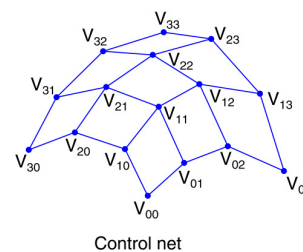
Given a grid of control points  $V_{ij}$  forming a **control net**, construct a surface  $S(u,v)$  by:

- ♦ treating rows of  $V$  (the matrix consisting of the  $V_{ij}$ ) as control points for curves  $V_0(u), \dots, V_n(u)$ .
- ♦ treating  $V_0(u), \dots, V_n(u)$  as control points for a curve parameterized by  $v$ .

11

## Tensor product Bézier surfaces, cont.

Let's walk through the steps:



Which control points are interpolated by the surface?

12

## Matrix form of Bézier curves and surfaces

Recall that Bézier curves can be written in terms of the Bernstein polynomials:

$$Q(u) = \sum_{i=0}^n V_i b_i(u)$$

They can also be written in a matrix form:

$$Q^T(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} V_0^T \\ V_1^T \\ V_2^T \\ V_3^T \end{bmatrix}$$

$$= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \mathbf{M}_{\text{Bezier}} \mathbf{V}_{\text{curve}}$$

Tensor product surfaces can be written out similarly:

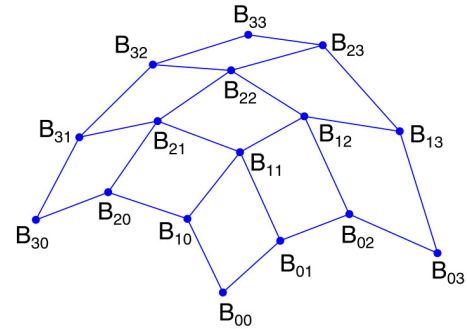
$$S(u, v) = \sum_{i=0}^n \sum_{j=0}^n V_{ij} b_i(u) b_j(v)$$

$$= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \mathbf{M}_{\text{Bezier}} \mathbf{V}_{\text{surface}} \mathbf{M}_{\text{Bezier}}^T \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

13

## Tensor product B-spline surfaces

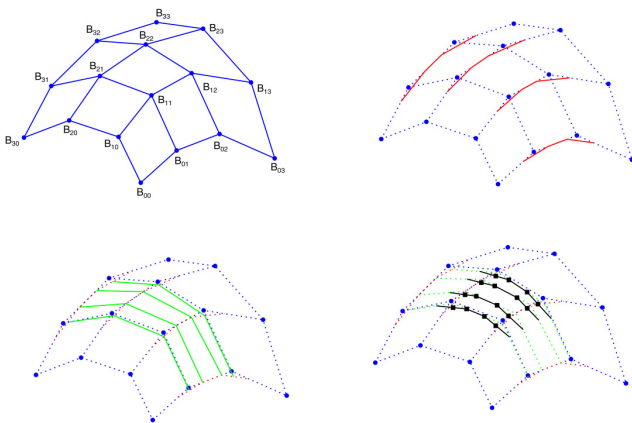
As with spline curves, we can piece together a sequence of Bézier surfaces to make a spline surface. If we enforce  $C^2$  continuity and local control, we get B-spline curves:



- ♦ 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  $u$ .

14

## Tensor product B-spline surfaces, cont.



Which B-spline control points are interpolated by the surface?

15

## Matrix form of B-spline surfaces

For curves, we can write a matrix that generates Bézier control points from B-spline control points:

$$\begin{bmatrix} V_0^T \\ V_1^T \\ V_2^T \\ V_3^T \end{bmatrix} = \frac{1}{6} \begin{bmatrix} 1 & 4 & 1 & 0 \\ 0 & 4 & 2 & 0 \\ 0 & 2 & 4 & 0 \\ 0 & 1 & 4 & 1 \end{bmatrix} \begin{bmatrix} B_0^T \\ B_1^T \\ B_2^T \\ B_3^T \end{bmatrix}$$

$$\mathbf{V}_{\text{curve}} = \mathbf{M}_{\text{B-spline}} \mathbf{B}_{\text{curve}}$$

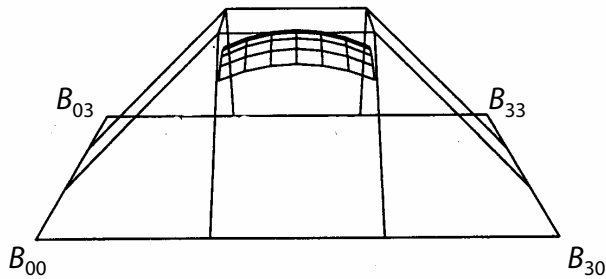
We can arrive at a similar form for tensor product B-spline surfaces:

$$\mathbf{V}_{\text{surface}} = \mathbf{M}_{\text{B-spline}} \mathbf{B}_{\text{surface}} \mathbf{M}_{\text{B-spline}}^T$$

16

## Tensor product B-splines, cont.

Another example:



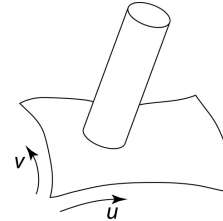
17

## 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.

18

## Summary

What to take home:

- ◆ How to construct swept surfaces from a profile and trajectory curve:
  - with a fixed frame
  - with a Frenet frame
- ◆ How to construct tensor product Bézier surfaces
- ◆ How to construct tensor product B-spline surfaces

19