COMP3271 Computer Graphics

Curves & Surfaces (III)

2019-20

Objectives

The de Casteljau algorithm for evaluating Bézier curves

Other curves and surfaces:

- Conics & Quadrics
- Extrusion surfaces
- Surface of revolutions
- Sweep surfaces

The de Casteljau Algorithm

We can use the convex hull property of Bézier curves to obtain an efficient recursive method that does not require any function evaluations

- Uses only the values at the control points

Based on the idea that "any polynomial and any part of a polynomial is a Bézier polynomial for properly chosen control data"

The de Casteljau Algorithm

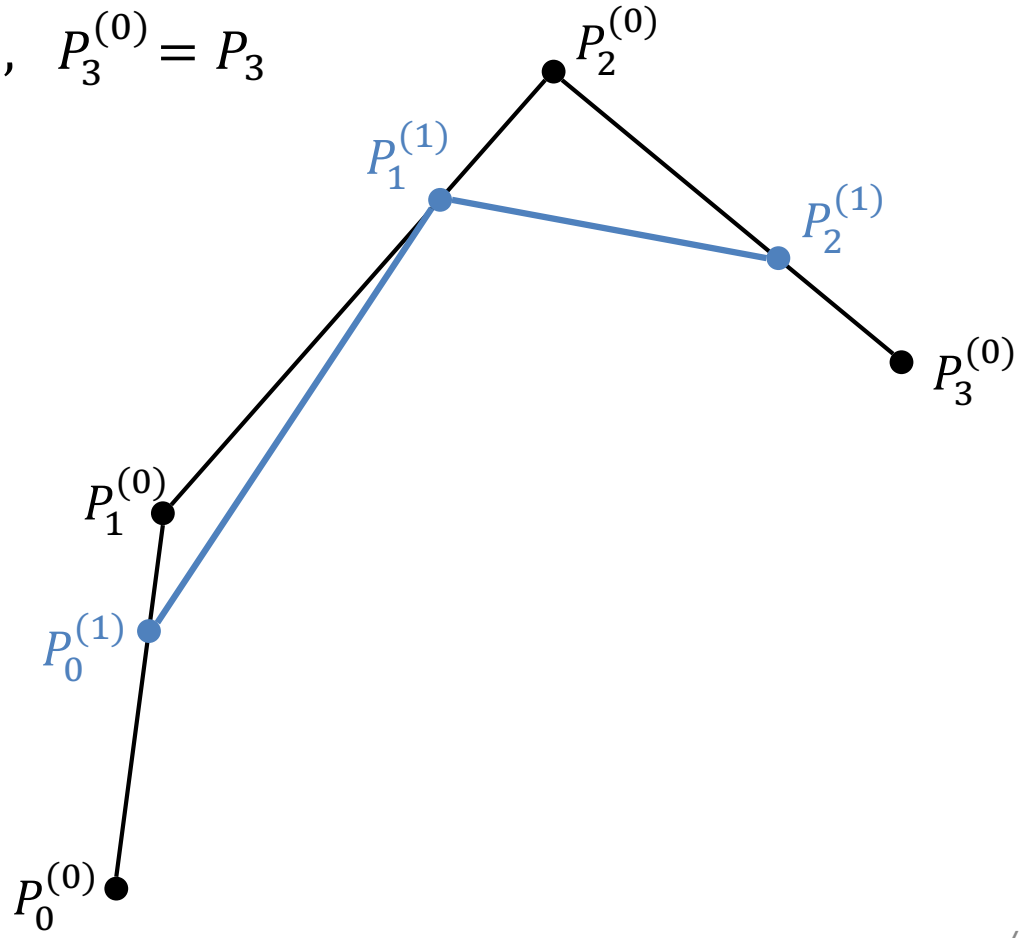
Given a cubic Bézier curve $P(t) = \sum_{i=0}^{3} P_i B_{i,3}(t)$. The following procedure produces a point $P(t)$ on the curve.

$$P_0^{(0)} = P_0,\ P_1^{(0)} = P_1,\ P_2^{(0)} = P_2,\ P_3^{(0)} = P_3$$

$$P_0^{(1)} = (1-t)P_0 + tP_1$$

$$P_1^{(1)} = (1-t)P_1 + tP_2$$

$$P_2^{(1)} = (1-t)P_2 + tP_3$$



The de Casteljau Algorithm

Given a cubic Bézier curve $P(t) = \sum_{i=0}^{3} P_i B_{i,3}(t)$. The following procedure produces a point $P(t)$ on the curve.

$$P_0^{(0)} = P_0,\ P_1^{(0)} = P_1,\ P_2^{(0)} = P_2,\ P_3^{(0)} = P_3$$

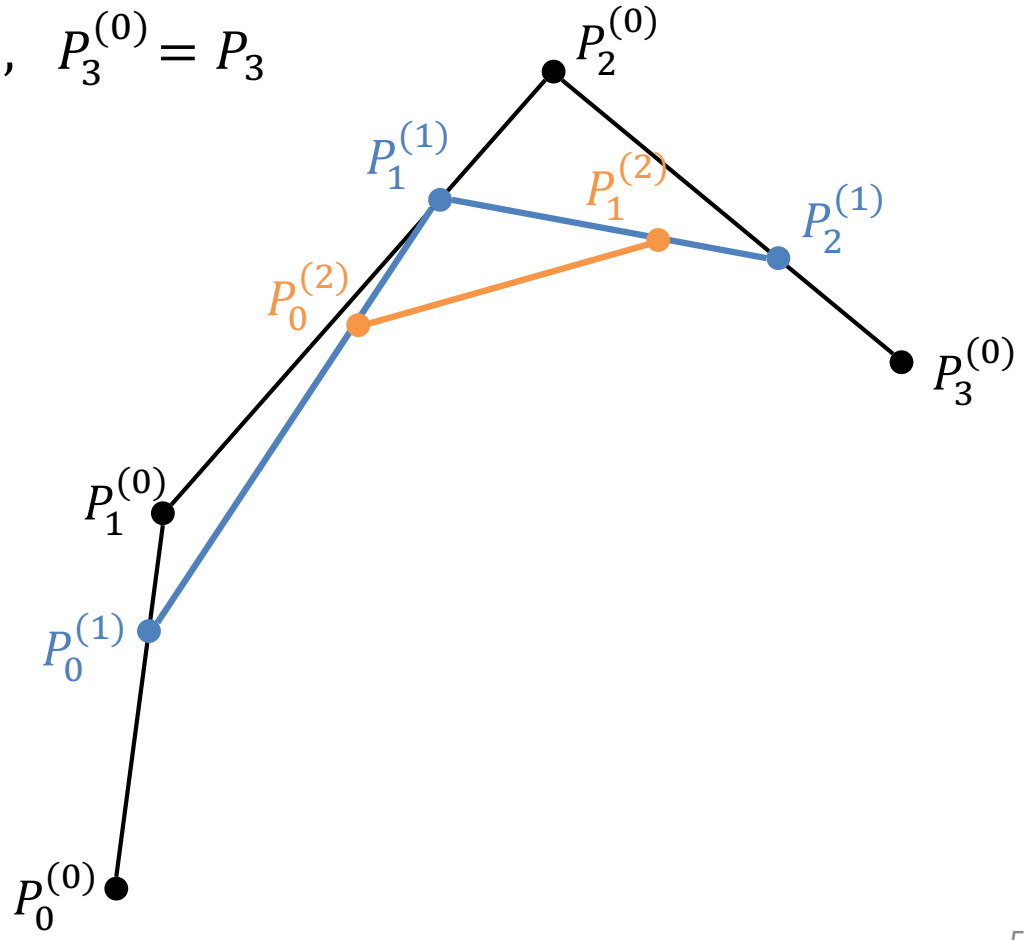
$$P_0^{(1)} = (1-t)P_0 + tP_1$$

$$P_1^{(1)} = (1-t)P_1 + tP_2$$

$$P_2^{(1)} = (1-t)P_2 + tP_3$$

$$P_0^{(2)} = (1-t)P_0^{(1)} + tP_1^{(1)}$$

$$P_1^{(2)} = (1-t)P_1^{(1)} + tP_2^{(1)}$$



The de Casteljau Algorithm

Given a cubic Bézier curve $P(t) = \sum_{i=0}^{3} P_i B_{i,3}(t)$. The following procedure produces a point $P(t)$ on the curve.

$$P_0^{(0)} = P_0,\ P_1^{(0)} = P_1,\ P_2^{(0)} = P_2,\ P_3^{(0)} = P_3$$

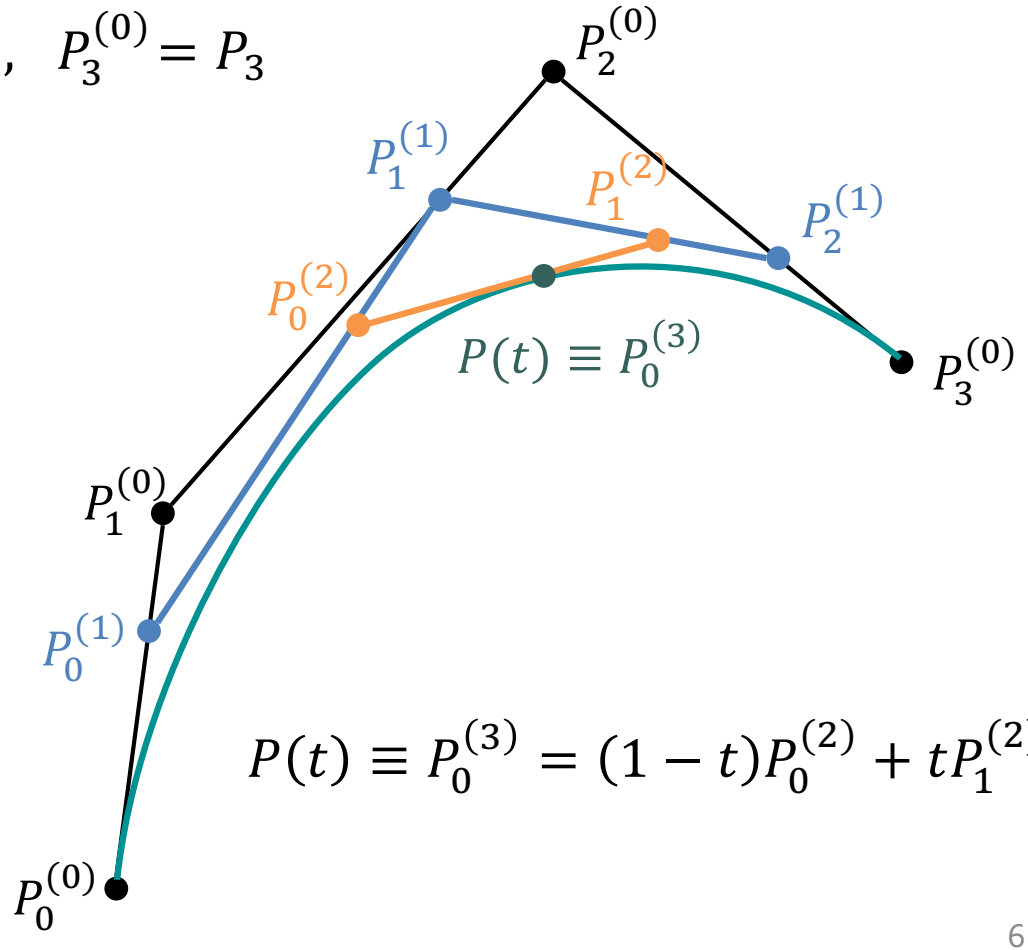
$$P_0^{(1)} = (1-t)P_0 + tP_1$$

$$P_1^{(1)} = (1-t)P_1 + tP_2$$

$$P_2^{(1)} = (1-t)P_2 + tP_3$$

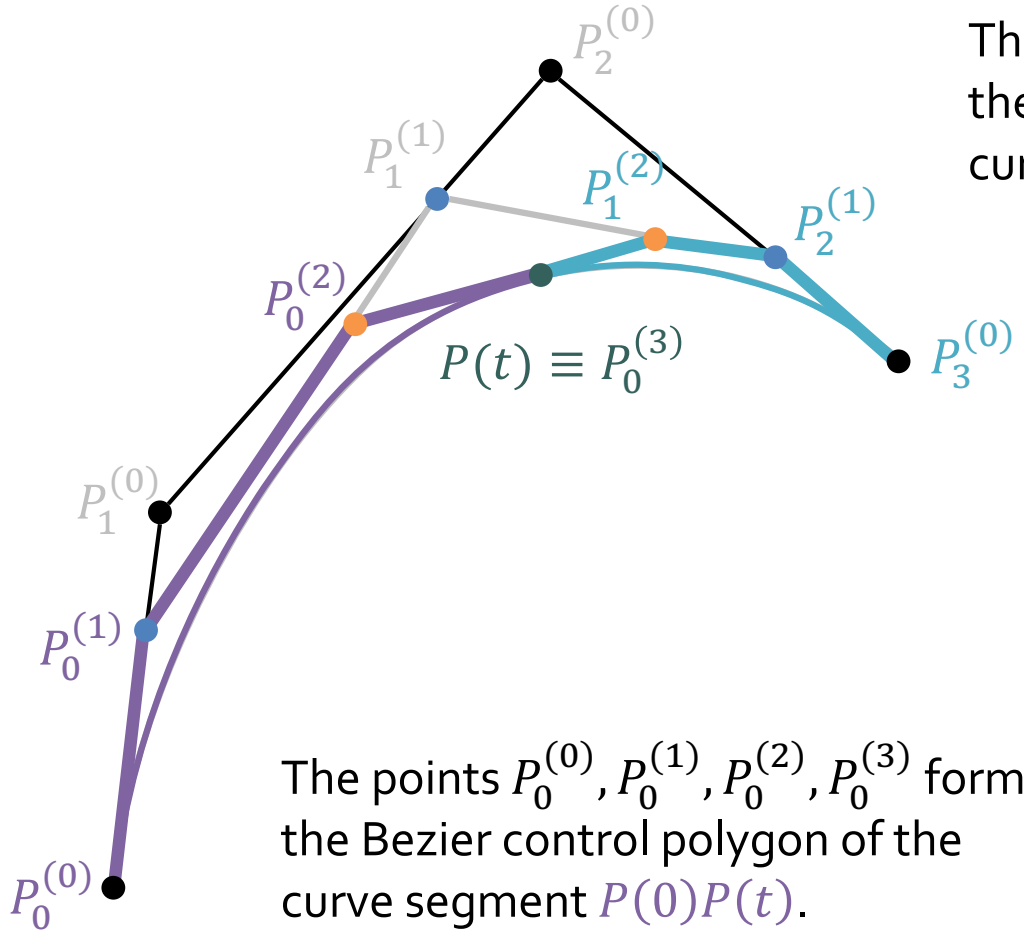
$$P_0^{(2)} = (1-t)P_0^{(1)} + tP_1^{(1)}$$

$$P_1^{(2)} = (1-t)P_1^{(1)} + tP_2^{(1)}$$



$$P(t) \equiv P_0^{(3)} = (1-t)P_0^{(2)} + tP_1^{(2)}$$

Splitting the Control Polygon



The points $P_0^{(3)}, P_1^{(2)}, P_2^{(1)}, P_2^{(0)}$ form the Bezier control polygon of the curve segment $P(t)P(1)$.

The points $P_0^{(0)}, P_0^{(1)}, P_0^{(2)}, P_0^{(3)}$ form the Bezier control polygon of the curve segment $P(0)P(t)$.

Middle-point Subdivision

When setting $t = \frac{1}{2}$, the arithmetic operations of the above algorithm are simplified to

$$P_i^{(k+1)} = \frac{P_i^{(k)} + P_{i+1}^{(k)}}{2}.$$

So only addition and right shift (division by 2) are required.

The middle-point subdivision scheme works by computing the Bézier control polygons of the two sub-curves of $P(t), t \in [0,1]$, over subintervals $[0, \frac{1}{2}]$ and $[\frac{1}{2}, 0]$. Then each sub-curve is recursively subdivided.

Note that the depth of subdivision can be made adaptive to the local curvature or the error of approximation.

Rational Bézier Curves

A rational Bézier curve is represented by

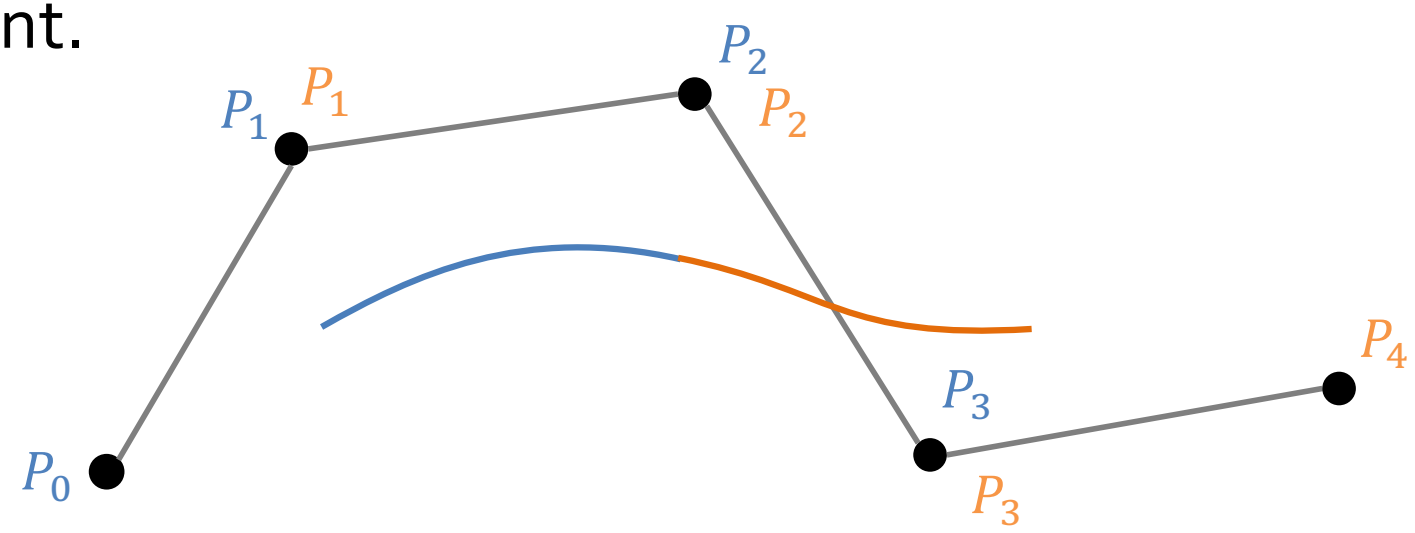
$$P(t) = \frac{\sum_{i=0}^{n} w_i B_{i,n}(t) P_i}{\sum_{i=0}^{n} w_i B_{i,n}(t)}, \qquad t \in [0,1].$$

- w_i can be thought of weights
- Increasing the weight w_i pulls a portion of $P(t)$ towards the control point P_i, and decreasing w_i pushes $P(t)$ away from P_i
- An advantage of the rational curve is that it encompasses all conic sections (with $n = 2$).

B-Splines

Basis splines: use the control points $p_{i-2}, p_{i-1}, p_i, p_{i+1}$ to define curve only between p_{i-1} and p_i (for cubic curves)

Allows us to apply more continuity conditions to each segment.



NURBS: Nonuniform Rational B-Spline curves and surfaces

Conic Sections

A common class of curves with a very long history

- defined by intersections of a plane with a cone
- defined implicitly by the 2nd degree polynomial

$$F(x,y) = ax^2 + 2bxy + 2cx + dy^2 + 2ey + f$$

In matrix form

$$F(x,y) = \mathbf{v}^{\mathsf{T}}\mathbf{Q}\mathbf{v} = \begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} a & b & c \\ b & d & e \\ c & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

This describes several generally useful kinds of curves

- circles, ellipses, parabolas, hyperbolas, and lines

Quadric Surfaces

Quadrics are the 3D analogue of conics

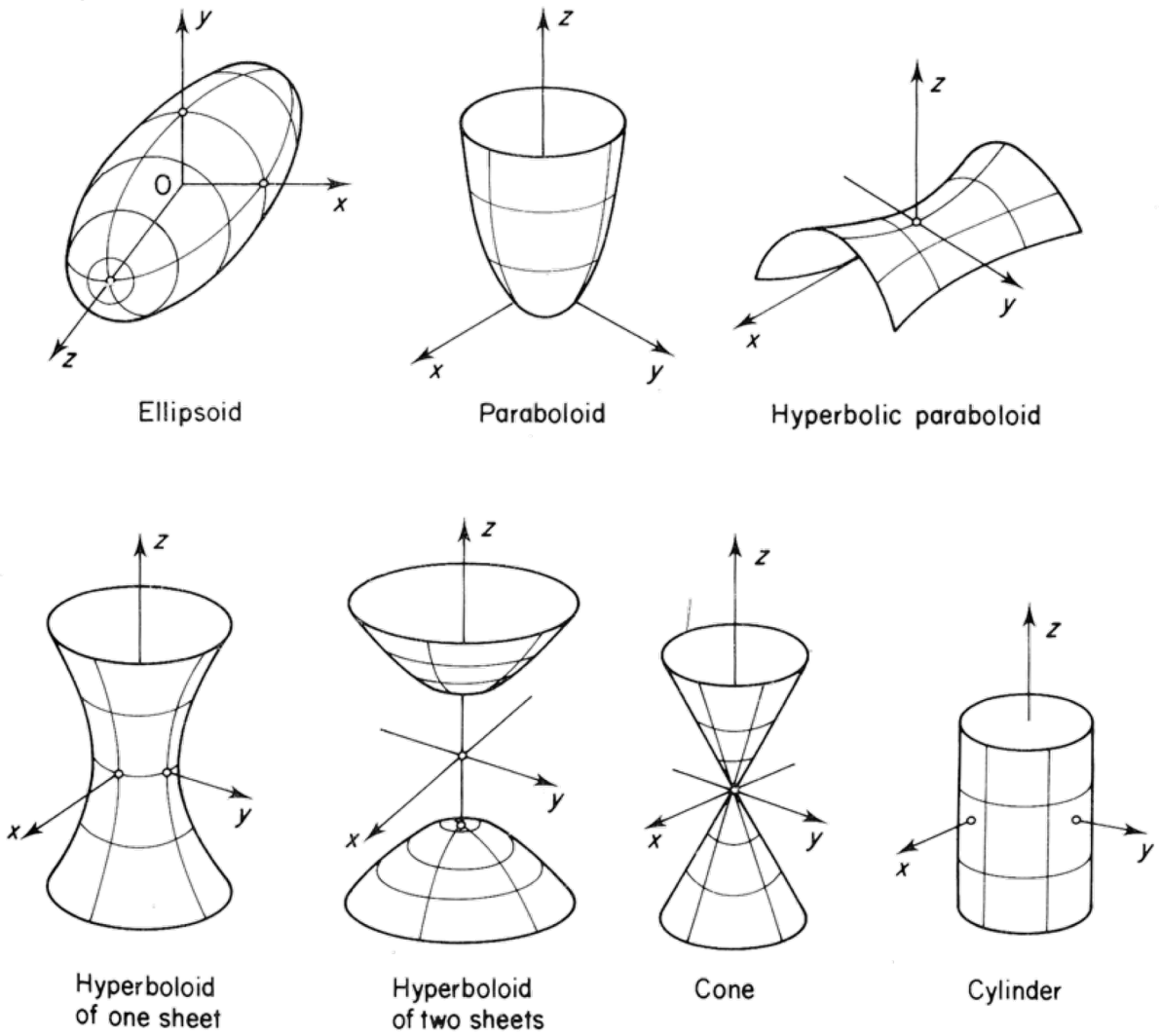
- defined by the 2nd degree polynomial

$$F(x,y,z) = ax^2 + 2bxy + 2cxz + 2dx + ey^2 + 2fyz + 2gy + hz^2 + 2iz + j$$

- or in matrix form

$$F(x,y,z) = \mathbf{v}^{\mathsf{T}}\mathbf{Q}\mathbf{v} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} a & b & c & d \\ b & e & f & g \\ c & f & h & i \\ d & g & i & j \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Quadric Surfaces



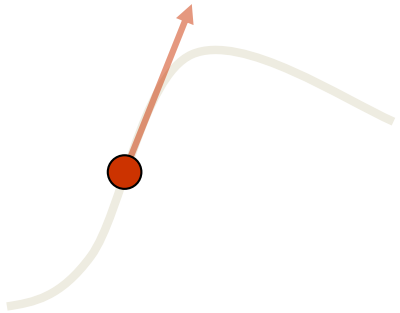
Quadric classes (from Paul Heckbert).

Sweeping out Surfaces

We view space curves as being swept out by a moving point

$$\mathbf{p}(u) = \begin{bmatrix} x(u) & y(u) & z(u) \end{bmatrix}$$

- as we vary u the point moves through space
- the curve is the path the point takes



Essentially looked at surfaces the same way

$$\mathbf{p}(u,v) = \begin{bmatrix} x(u,v) & y(u,v) & z(u,v) \end{bmatrix}$$

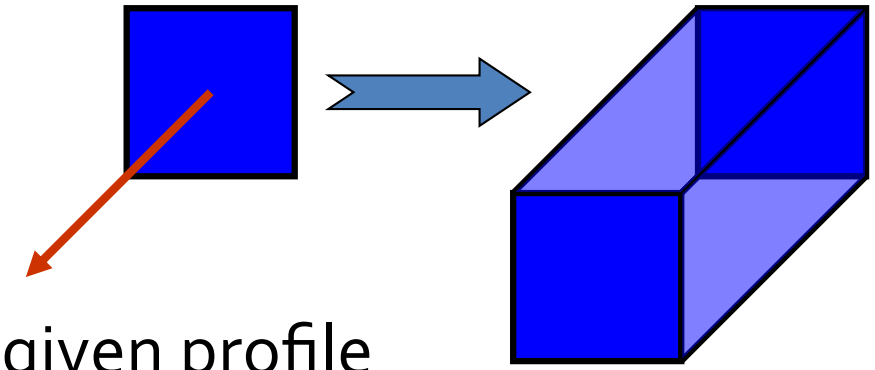
Now let's think about sweeping curves through space instead

- this will define a surface
- the set of all points visited by the curve during its motion

Extrusion Surfaces

Here's a particularly simple method

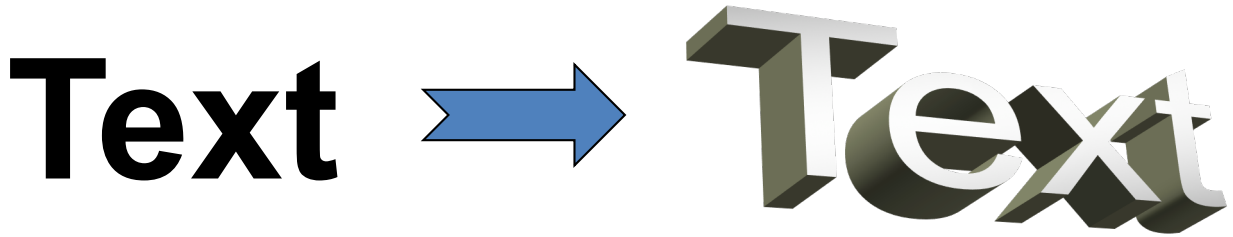
- specify initial (closed) curve
- pick an axis to move along
- and a distance to move



Sweeps out something with the given profile

- open curve defines a surface with an open boundary
- closed curve defines something like a cylinder

This is a common technique used to create 3D text



Extrusion Surfaces

Can be generated by translating a 2D cross-section curve along a fixed direction.

Let the cross-section curve be $C(u)$, and the given direction vector be D. Then the surface is defined by

$$E(u, v) = vD + C(u), \qquad v \in [v_0, v_1].$$

Surfaces of Revolution

Extrusion moves curves via translation

- we can just as easily use rotation

Start with some curve

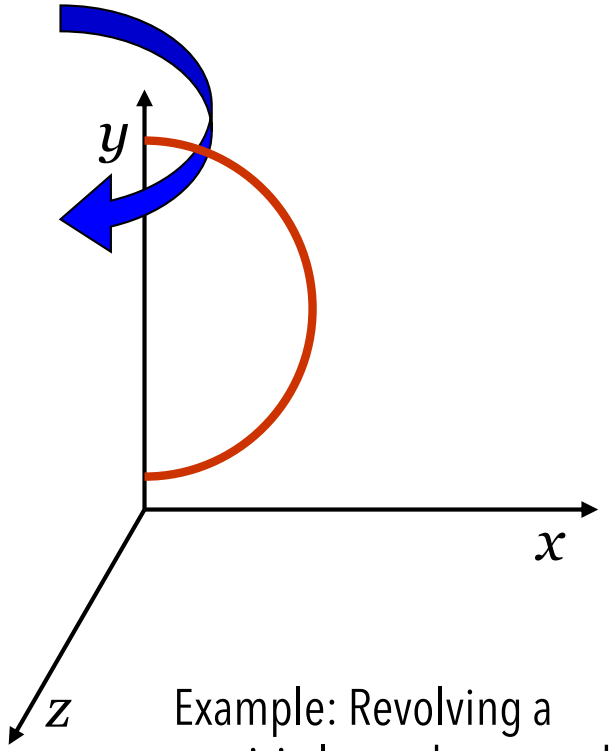
- pick an axis of rotation
- rotate about axis by 360°

Characteristics of revolved surfaces

- closed if endpoints on axis
- open otherwise
- but we can always fill in top/bottom
- by construction, they're symmetric

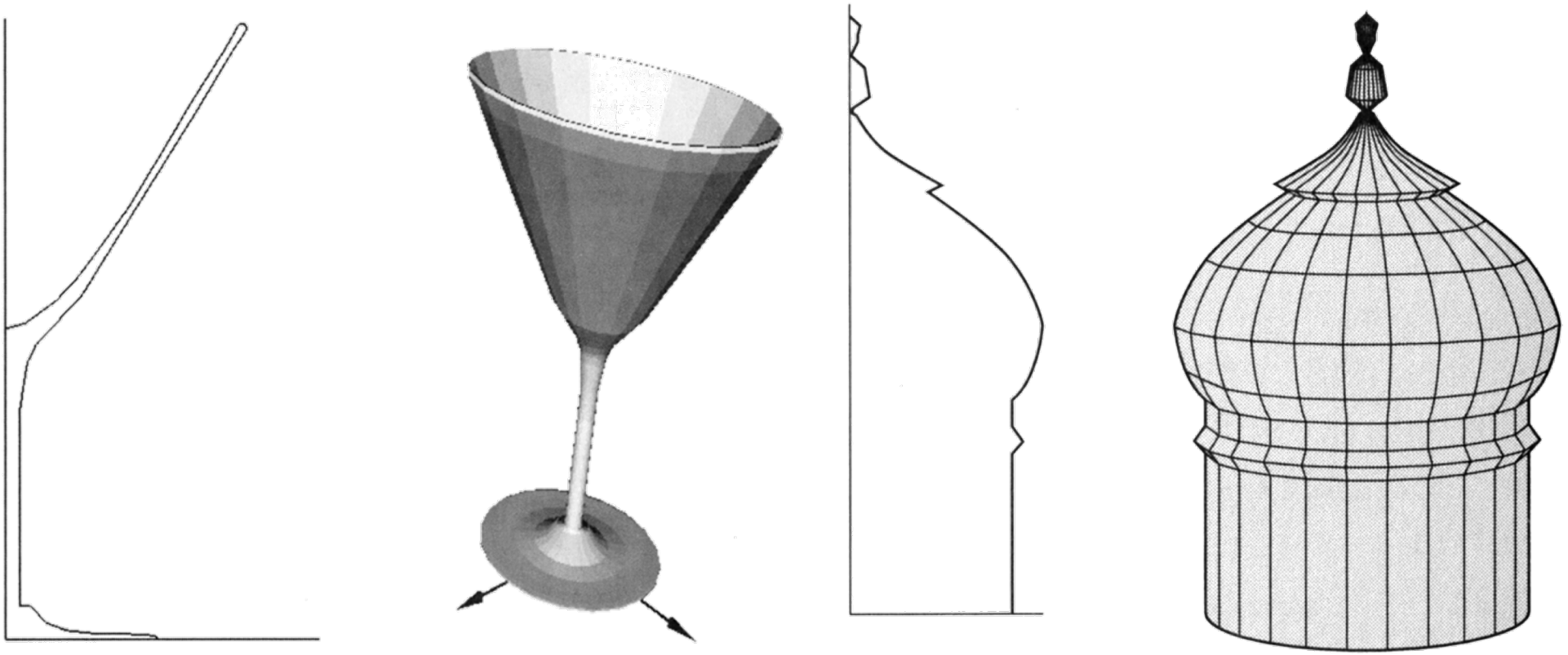
Lots of other easy examples:

- cylinder, cone, paraboloid, …



Example: Revolving a semicircle produces a sphere.

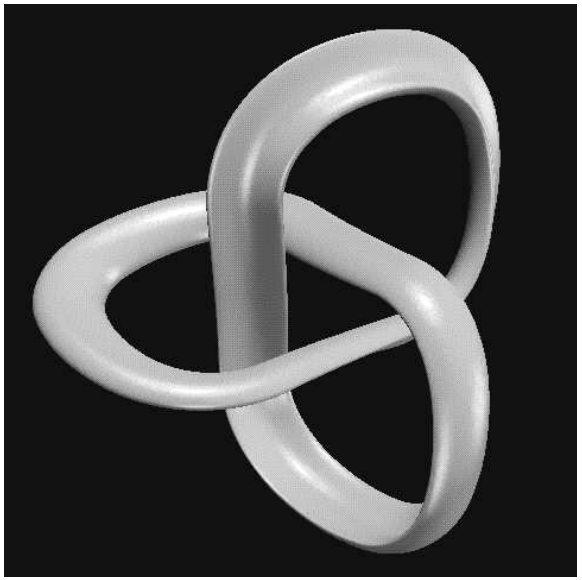
More Complex Examples of Revolution



Sweep Surfaces

The sweep surface is generated by sweeping a 2D cross-section curve along an axis curve.

The plane of the cross-section curve is usually kept perpendicular to the tangent of the axis curve.



A sweep surface along a cubic spline curve.

Sweep Surfaces

Let the cross-section curve be $C(u)$ and the axis curve be $A(v)$.
Let $F(v)$ be the matrix representing a rotation frame attached at $A(v)$. Then the sweep surface is defined by

$$S(u,v) = A(v) + F(v)C(u), \qquad v \in [v_0, v_1].$$

To use $s(v)$ to change the size of the cross-section curve while sweeping it, we have

$$S(u,v) = A(v) + s(v)F(v)C(u), \qquad v \in [v_0, v_1].$$