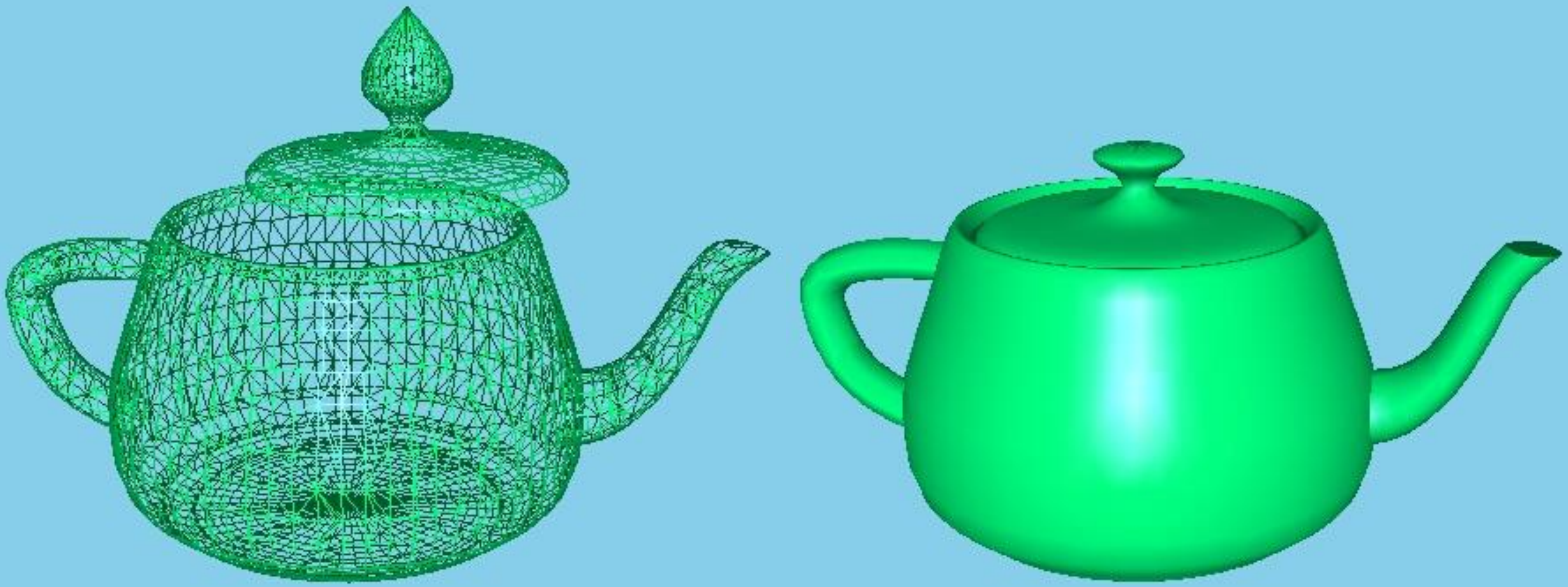
Modeling Curved Surfaces



Curved Surfaces

The use of curved surfaces allows for a higher level of modeling, especially for the construction of highly realistic models.

There are several approaches to modeling curved surfaces:

We can represent curved surfaces using mesh of curves. So we learn to create curves first then move to curved surfaces.

Curved Surfaces

There are two ways to construct a model:

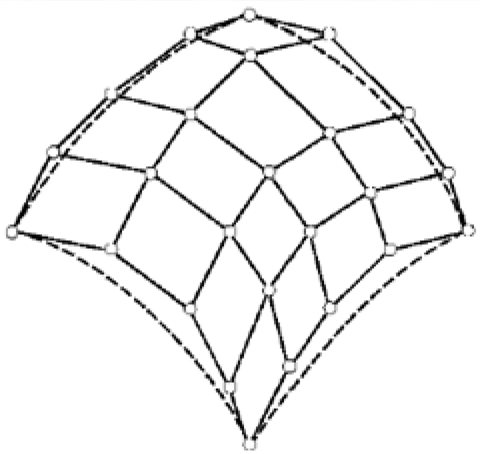
Additive Modeling

This is the process of building the model by assembling many simpler objects.

Subtractive Modeling

This is the process of removing pieces from a given object to create a new object.

For example, creating a (cylindrical) hole in a sphere or a cube.



Curved Surface Patch

Curve Representation

- There are three ways to represent a curve
  - Explicit: $y = f(x)$

    $\mathbf{y = mx + b}$ $\qquad\qquad$ $\mathbf{y = x^2}$

    (–) Must be a single valued function

    (–) Vertical lines, say $x = d$?
  - Implicit: $f(x,y) = 0$

    $\mathbf{x^2 + y^2 - r^2 = 0}$

    (+) **y** can be multiple valued function of x

    (–) Vertical lines? (Continuity hard to detect)
  - Parametric: $(x, y) = (x(t), y(t))$

    $\mathbf{(x, y) = (\cos\mathit{t}, \sin\mathit{t})}$

    (+) Easy to specify, modify and control

    (–) Extra hidden variable t, the parameter

Explicit Representation

- Curve in 2D: $y = f(x)$
- Curve in 3D: $y = f(x)$, $z = g(x)$
- Surface in 3D: $z = f(x,y)$
- Problems:
  - How about a vertical line $x = c$ as $y = f(x)$?
  - Circle $y = \pm (r^2 - x^2)^{1/2}$ two or zero values for x
- • Rarely used in computer graphics

Implicit Representation

- Curve in 2D: $f(x,y) = 0$
  - Line: $ax + by + c = 0$
  - Circle: $x^2 + y^2 - r^2 = 0$
- Surface in 3d: $f(x,y,z) = 0$
  - Plane: $ax + by + cz + d = 0$
  - Sphere: $x^2 + y^2 + z^2 - r^2 = 0$
- $f(x,y,z)$ can describe 3D object:
  - Inside: $f(x,y,z) < 0$
  - Surface: $f(x,y,z) = 0$
  - Outside: $f(x,y,z) > 0$

Parametric Form for Curves

- Curves: single parameter t (e.g. time)
  - $x = x(t), y = y(t), z = z(t)$
- Circle:
  - $x = \cos(t), y = \sin(t), z = 0$
- Tangent described by derivative

$$p(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} \qquad \frac{dp(t)}{dt} = \begin{bmatrix} \frac{dx(t)}{dt} \\ \frac{dy(t)}{dt} \\ \frac{dz(t)}{dt} \end{bmatrix}$$

- Magnitude is "velocity"

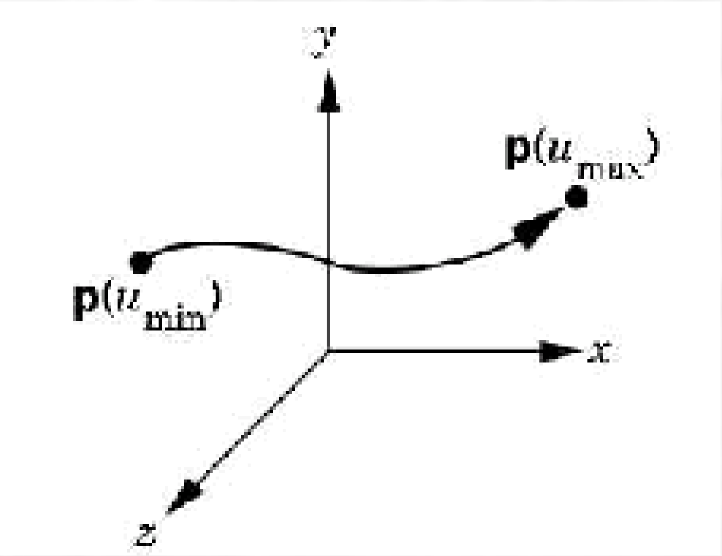
Parametric Form for Surfaces

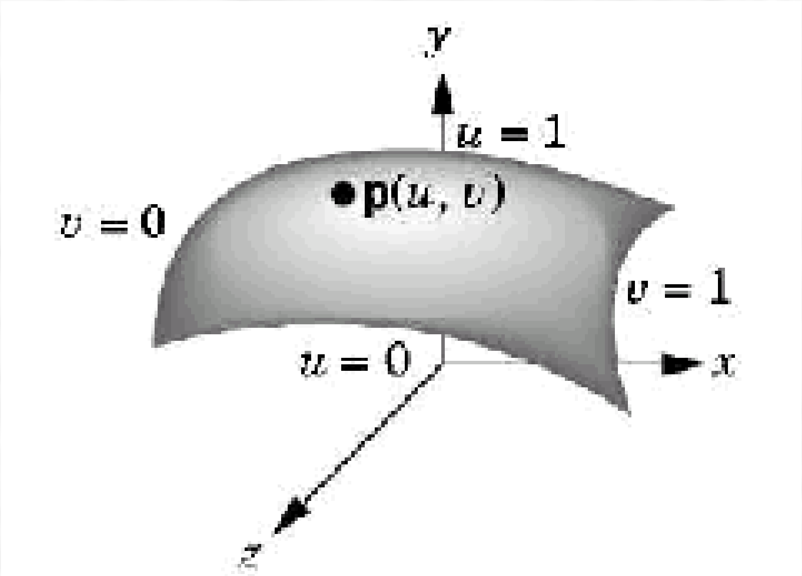
- Use parameters u and v
  - $x = x(u,v)$, $y = y(u,v)$, $z = z(u,v)$
- Describes surface as both u and v vary
- Partial derivatives describe tangent plane at each point $p(u,v) = [x(u,v)\ y(u,v)\ z(u,v)]^T$

$$\frac{\partial p(u,v)}{\partial u} = \begin{bmatrix} \frac{\partial x(u,v)}{\partial u} \\ \frac{\partial y(u,v)}{\partial u} \\ \frac{\partial z(u,v)}{\partial u} \end{bmatrix} \qquad \frac{\partial p(u,v)}{\partial v} = \begin{bmatrix} \frac{\partial x(u,v)}{\partial v} \\ \frac{\partial y(u,v)}{\partial v} \\ \frac{\partial z(u,v)}{\partial v} \end{bmatrix}$$

Advantages of Parametric Form

- Parameters often have natural meaning
- Easy to define and calculate
  - Tangent and normal
  - Curves segments (for example, $0 \leq u \leq 1$)
  - Surface patches (for example, $0 \leq u,v \leq 1$)





Lagrange Polynomial

- Given n+1 points (x_0, y_0), (x_1, y_1) …….. (x_n, y_n)
- To construct a curve that passes through these points we can use Lagrange polynomial defined as follows:.

$$y = f(x) = \sum_{k=0}^{n} y_k L_{n,k}$$

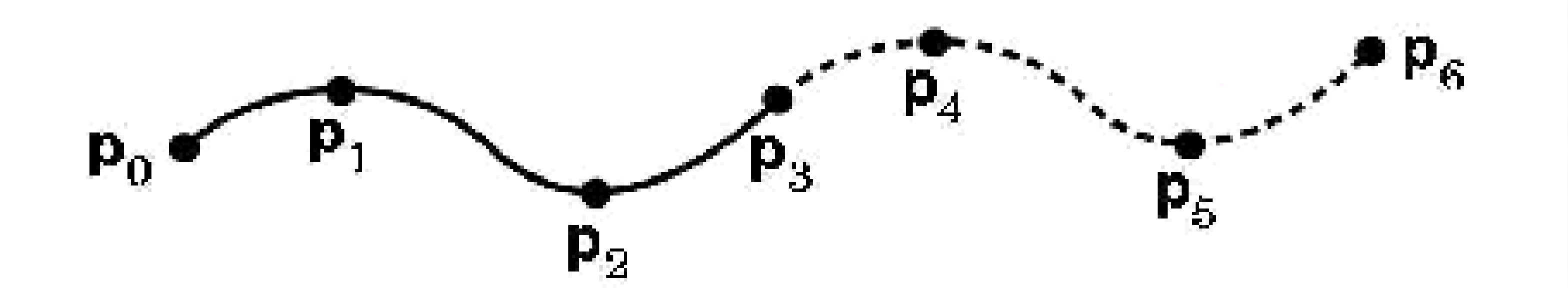
$$L_{n,k} = \frac{(x - x_o)(x - x_1)\cdots(x - x_{k-1})(x - x_{k+1})\cdots(x - x_n)}{(x_k - x_o)(x_k - x_1)\cdots(x_k - x_{k-1})(x_k - x_{k+1})\cdots(x_k - x_n)}$$

Problems:

- y=f(x), no multiple values
- Higher order functions tend to oscillate
- No local control (change any (x_i, y_i) changes the whole curve)
- Computationally expensive due to high degree.

Piecewise Linear Polynomial

- To overcome the problems with Lagrange polynomial
  - Divide given points into overlap sequences of 4 points
  - construct **3rd degree** polynomial that passes through these points, p_0, p_1 , p_2 , p_3 then p_3, p_4 , p_5 , p_6 etc.
  - Then glue the curves so that they appear **sufficiently smooth** at joint points.



Questions:

1. Why 3^{rd} Degree curves used?
2. How to measure smoothness at joint point?

Why Cubic Curves?

A curve is approximated by a piecewise polynomial curve.

Cubic polynomials are most often used because:

(1) Lower-degree polynomials offer too little flexibility in controlling the shape of the curve.

(2) Higher-degree polynomials can introduce unwanted wiggles and also require more computation.

(3) No lower-degree representation allows a curve segment to be defined by two given endpoints with given derivative at each endpoints.

(4) No lower-degree curves are nonplanar in 3D.

Measure of Smoothness

G^0 Geometric Continuity ⇔ C^0 Parametric Continuity

If two curve segments join together.

G^1 Geometric Continuity

If the **directions** (but not necessarily the magnitudes) of the two segments' tangent vectors are equal at a join point.

C^1 Parametric Continuity

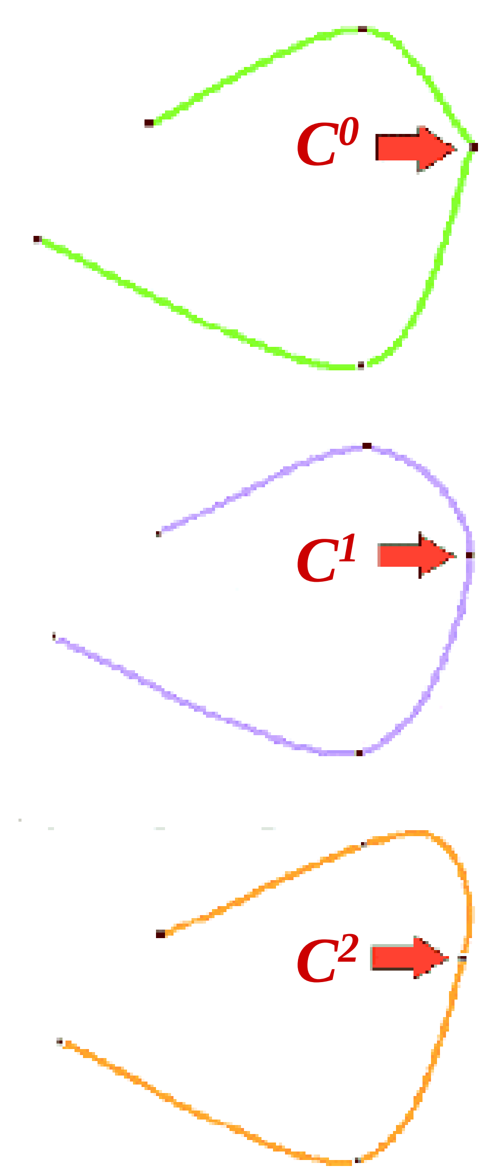
If the **directions and magnitudes** of the two segments' tangent vectors are equal at a join point.

C^2 Parametric Continuity

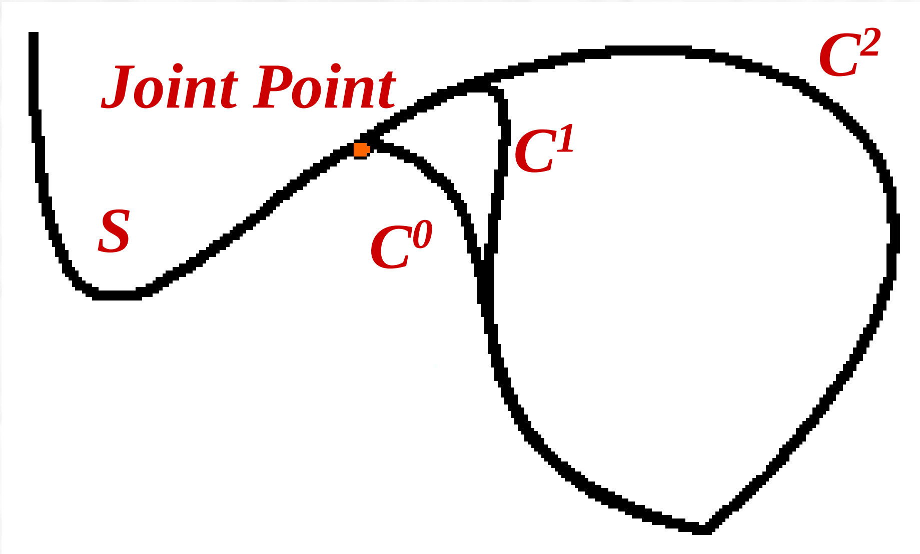
If the direction and magnitude of $Q^2(t)$ (curvature or **acceleration**) are equal at the join point.

C^n Parametric Continuity

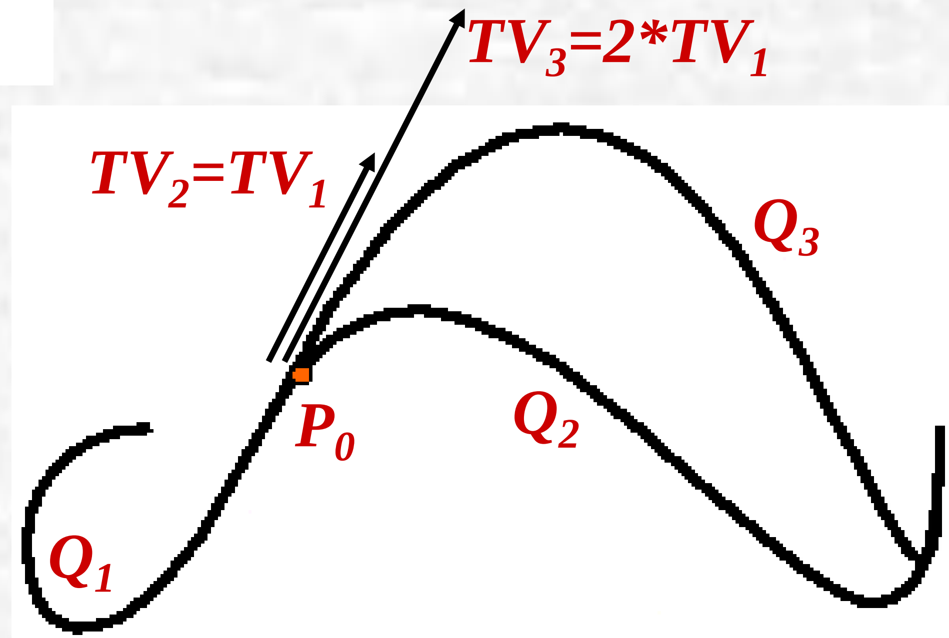
If the direction and magnitude of $Q^n(t)$ through the nth derivative are equal at the join point.



Measure of Smoothness



- By increasing parametric continuity we can increase smoothness of the curve.



- Q_1& Q_2 are C^1 and G^1 continuous
- Q_1& Q_3 are G^1 continuous only as Tangent vectors have different magnitude.
- Observe the effect of increasing in magnitude of TV

Interpolation Vs. Approximation

Given $n + 1$ points $P_0(x_0, y_0), P_1(x_1, y_1), \ldots, P_n(x_n, y_n)$

we wish to find a curve that, in some sense, fits the shape outlined by these points.

Based on requirements we are faced with two problems:

Interpolation

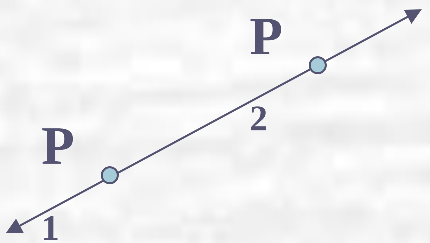
If we require the curve to pass through all the points.

Approximation

If we require only that the curve be near these points.

Parametric Representation of Lines

- Interpolation of two points
- In Parametric form:



$$P(t) = P_1 + t \cdot (P_2 - P_1)$$

$$x(t) = x_1 + t \cdot (x_2 - x_1)$$

$$y(t) = y_1 + t \cdot (y_2 - y_1)$$

$$x(t) = TC_x = TMG_x = BG_x$$

$$y(t) = TC_y = TMG_y = BG_y$$

$$x(t) = \underbrace{\begin{bmatrix} t & 1 \end{bmatrix}}_{\substack{\textbf{Parameter} \\ \textbf{T}}} \underbrace{\begin{bmatrix} x_2 - x_1 \\ x_1 \end{bmatrix}}_{\substack{\textbf{Co-eff} \\ \textbf{C}}} = \begin{bmatrix} t & 1 \end{bmatrix} \underbrace{\begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix}}_{\substack{\textbf{Basis} \\ \textbf{Matrix} \\ \textbf{M}}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{\substack{\textbf{Geometry} \\ \textbf{G}}} = \underbrace{\begin{bmatrix} 1-t & t \end{bmatrix}}_{\substack{\textbf{Blending} \\ \textbf{Function} \\ \textbf{B}}} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Parametric Cubic Curves

$$Q(t) = [x(t)\ y(t)\ z(t)] \begin{cases} x(t) = a_x t^3 + b_x t^2 + c_x t + d_x, \\ y(t) = a_y t^3 + b_y t^2 + c_y t + d_y, \\ z(t) = a_z t^3 + b_z t^2 + c_z t + d_z,\ 0 \le t \le 1 \end{cases}$$

$$\therefore\ Q(t) = \underbrace{\begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}}_{T} \underbrace{\begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix}}_{C}$$

$$\therefore\ Q(t) = T \cdot C$$

Parametric Cubic Curves

- Now co-efficient matrix **C** can be expressed as a multiple of basis(weight) matrix **M** and geometry matrix **G**.

$$Q(t) = [x(t) \quad y(t) \quad z(t)] = T \cdot C = T \cdot M \cdot G$$

$$Q(t) = [t^3 \quad t^2 \quad t \quad 1] \underbrace{\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix}}_{\textit{basis matr ix}} \underbrace{\begin{bmatrix} G_1 \\ G_2 \\ G_3 \\ G_4 \end{bmatrix}}_{\textit{geometry vector}}$$

- Each element of geometry vector **G** has 3 component for x, y and z.
- Components of **G** can be expressed as $\mathbf{G_x}$, $\mathbf{G_y}$ and $\mathbf{G_z}$.

Parametric Cubic Curves

- Multiplying out only the x-component we get

$$x(t) = T \cdot M \cdot G_x = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \begin{bmatrix} g_{1x} \\ g_{2x} \\ g_{3x} \\ g_{4x} \end{bmatrix}$$

$$\begin{aligned} x(t) = & \left(t^3 m_{11} + t^2 m_{21} + t m_{31} + m_{41}\right) g_{1x} \\ & + \left(t^3 m_{12} + t^2 m_{22} + t m_{32} + m_{42}\right) g_{2x} \\ & + \left(t^3 m_{13} + t^2 m_{23} + t m_{33} + m_{43}\right) g_{3x} \\ & + \left(t^3 m_{14} + t^2 m_{24} + t m_{34} + m_{44}\right) g_{4x} \end{aligned}$$

a blending function

- The curve is a weighted sum of the elements of geometry matrix
- The weights are each cubic polynomials of t called *blending function*
- *B=T*M*

Derivative of *Q(t)*

- Derivative of ***Q(t)*** is the parametric *tangent vector* of the curve.

$$\frac{dQ(t)}{dt} = Q'(t) = \left[\frac{d}{dt}x(t) \quad \frac{d}{dt}y(t) \quad \frac{d}{dt}z(t) \right]$$

$$Q'(t) = \frac{d}{dt}T \cdot C = \left[3t^2 \quad 2t \quad 1 \quad 0 \right] \cdot C$$

$$Q'(t) = \left[3a_x t^2 + 2b_x t + c_x \qquad 3a_y t^2 + 2b_y t + c_y \qquad 3a_z t^2 + 2b_z t + c_z \right]$$

Curve Design : Determining C

A curve segment $Q(t)$ is defined by constraints on:

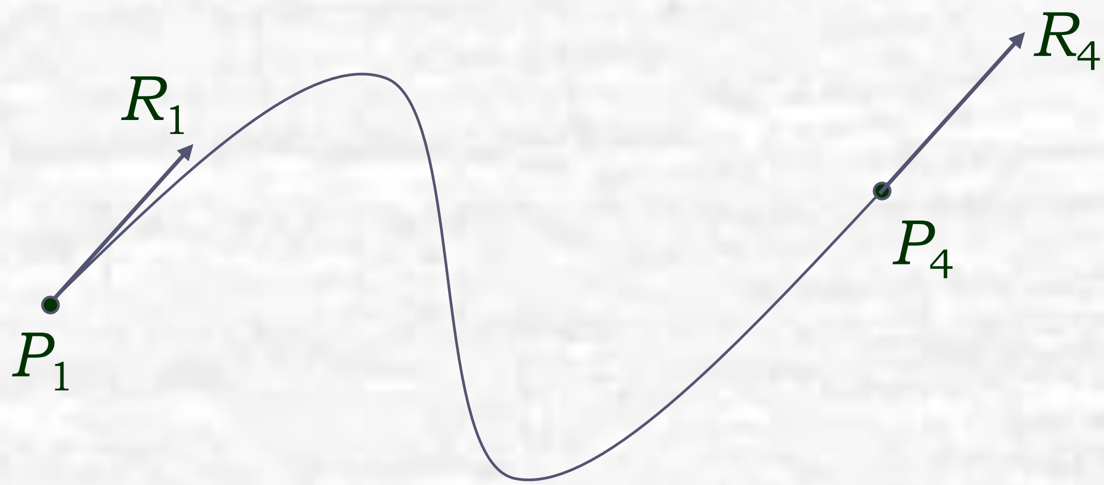
(1) endpoints

(2) tangent vectors

and (3) continuity between segments

Each cubic polynomial of $Q(t)$ has 4 coefficients,
so 4 constraints will be needed,
allowing us
to formulate 4 equations in the 4 unknowns,
then solving for the unknowns.

Hermit Curves



A cubic Hermite curve segment interpolating the endpoints P_1 and P_4 is determined by constraints on
the endpoints P_1 and P_4
and
tangent vectors at the endpoints R_1 and R_4

Hermit Curves

The Hermite Geometry Vector: $G_H = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix}$

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x = T \cdot C_x = T \cdot M_H \cdot G_{H_x}$$

$$= \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} M_H \cdot G_{H_x}$$

The constraints on $x(0)$ and $x(1)$:

$$x(0) = P_{1x} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} M_H \cdot G_{H_x}$$

$$x(1) = P_{4x} = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} M_H \cdot G_{H_x}$$

Hermit Curves

$$x'(t) = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} M_H \cdot G_{H_x}$$

Hence the tangent-vector constraints:

$$x'(0) = R_{1x} = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} M_H \cdot G_{H_x}$$
$$x'(1) = R_{4x} = \begin{bmatrix} 3 & 2 & 1 & 0 \end{bmatrix} M_H \cdot G_{H_x}$$

The 4 constraints can be written as:

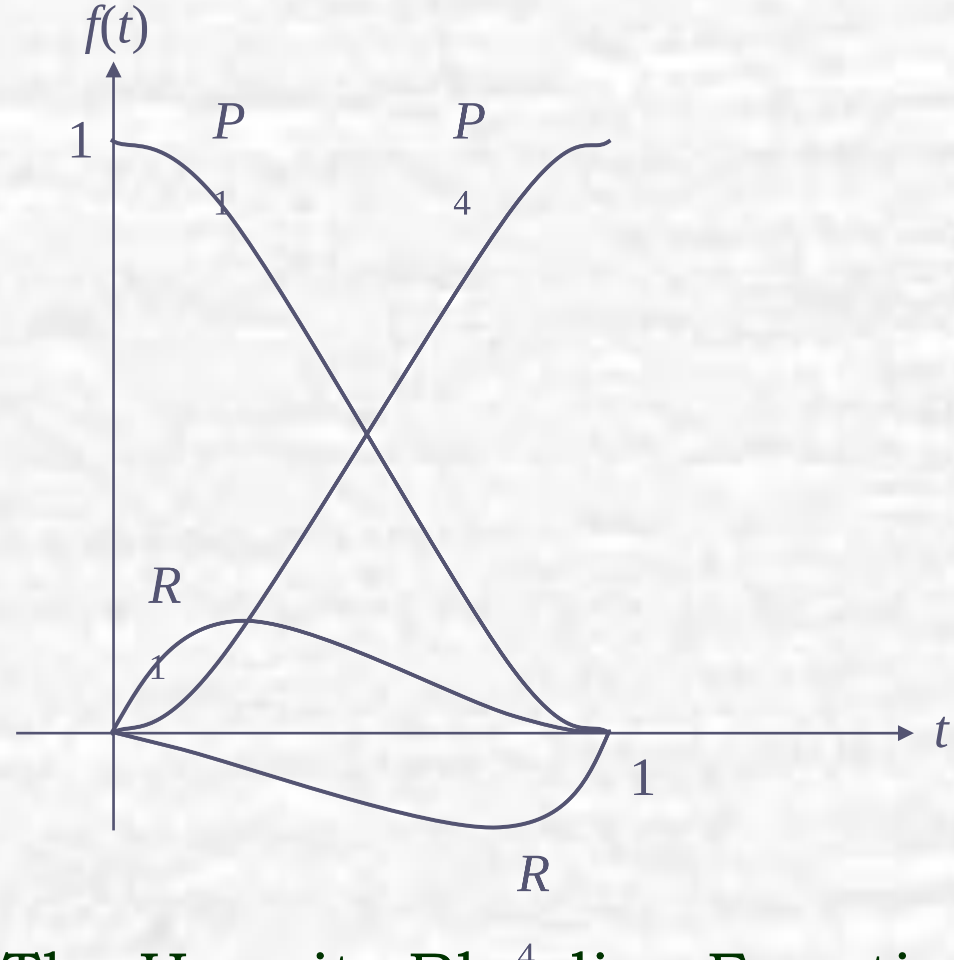
$$\begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix}_x = G_{H_x} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} M_H \cdot G_{H_x}$$

Hermit Curves

$$M_H = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

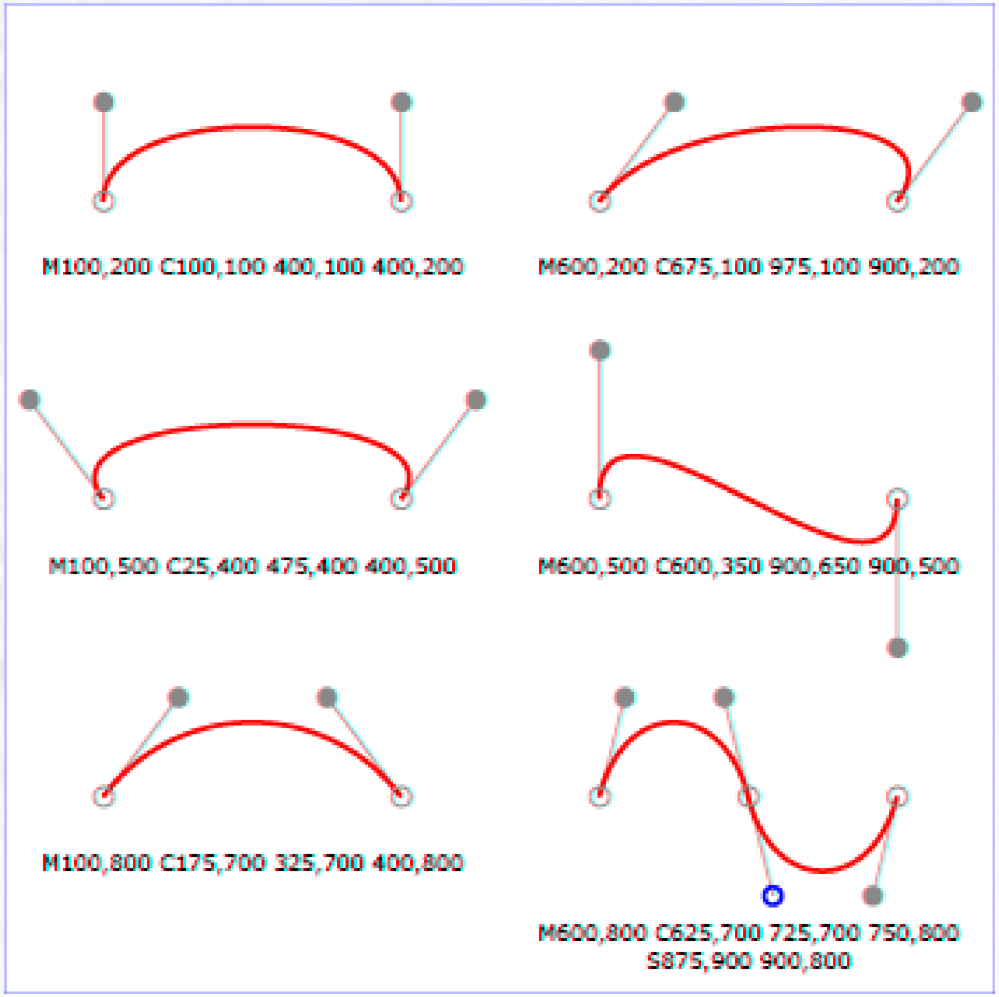
$$\begin{aligned} Q(t) = [x(t) \quad y(t) \quad z(t)] &= T \cdot M_H \cdot G_H = B_H \cdot G_H \\ &= \left(2t^3 - 3t^2 + 1\right)P_1 + \left(-2t^3 + 3t^2\right)P_4 \\ &\quad + \left(t^3 - 2t^2 + t\right)R_1 + \left(t^3 - t^2\right)R_4 \end{aligned}$$

Hermit Curves



The Hermite Blending Functions

Hermit Curves

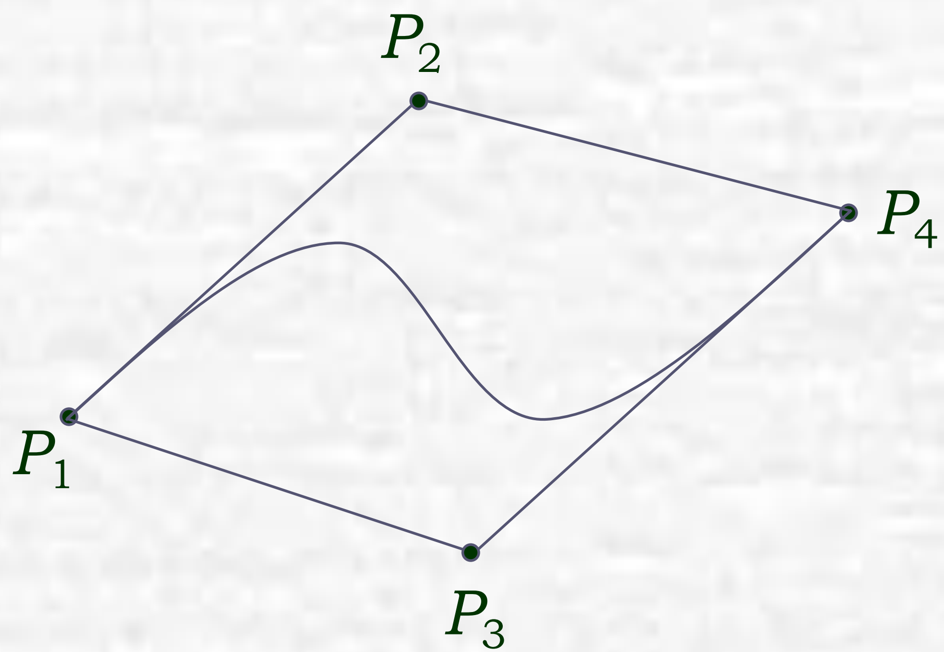


$$P_1, P_2, P_3, P_4, P_5, P_6, P_7$$

$$\begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix} = \begin{bmatrix} P_4 \\ P_7 \\ kR_4 \\ R_7 \end{bmatrix}$$

$$G^0, G^1, C^1$$

Bézier Curves



Indirectly specifies the endpoint tangent vectors by specifying two intermediate points that are not on the curve.

$$R_1 = Q'(0) = P_1P_2 = 3(P_2 - P_1)$$
$$R_4 = Q'(1) = P_3P_4 = 3(P_4 - P_3)$$

Bézier Curves

The Bézier Geometry Vector: $G_B = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$

$$G_H = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix} = M_{HB} \cdot G_B$$

$$\begin{aligned} Q(t) &= T \cdot M_H \cdot G_H = T \cdot M_H \cdot (M_{HB} \cdot G_B) \\ &= T \cdot (M_H \cdot M_{HB}) \cdot G_B = T \cdot M_B \cdot G_B \end{aligned}$$

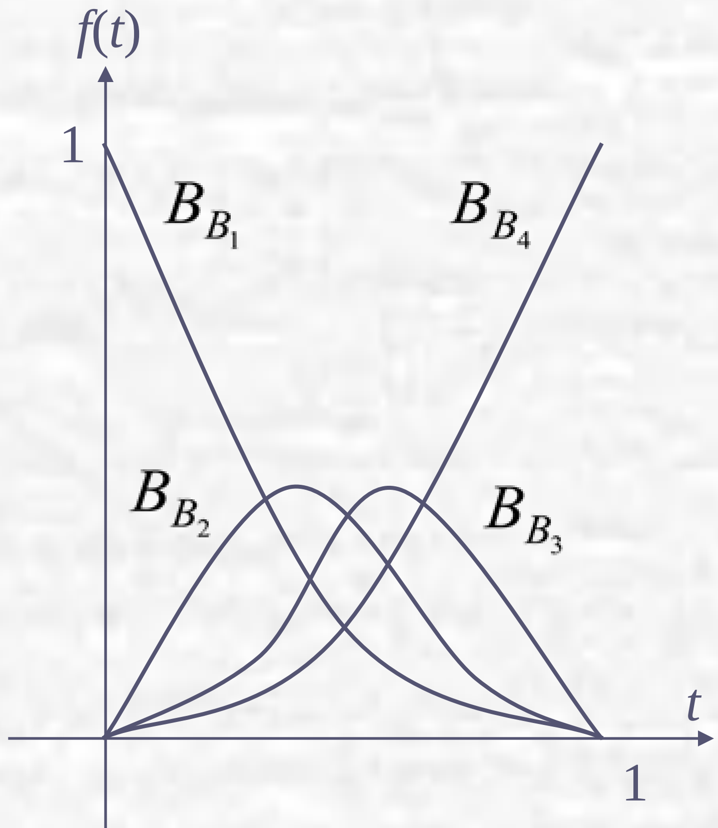
Bézier Curves

$$M_B = M_H \cdot M_{HB} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$Q(t) = T \cdot M_B \cdot G_B$$
$$= (1-t)^3 P_1 + 3t(1-t)^2 P_2 + 3t^2(1-t) P_3 + t^3 P_4$$

The 4 polynomials in $B_B = T \,.\, M_B$ are called the Bernstein polynomials.

Bézier Curves



<u>The Bernstein Polynomials</u>

A Bézier curve is bounded by the convex hull of its control points. (B : sum 1 and nonnegative)

Curve Rendering

- Brute-Force method
- Forward differencing
- Recursive sub-division

Brute-Force method

```
t = 0;
for (i=0; i <= 100; i++) {
    x(t) = a_x t^3 + b_x t^2 + c_x t + d_x
    y(t) = a_y t^3 + b_y t^2 + c_y t + d_y
    z(t) = a_z t^3 + b_z t^2 + c_z t + d_z
    Plot3d( x(t), y(t), z(t) );
    t += 0.01;
}
```

$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x$

$y(t) = a_y t^3 + b_y t^2 + c_y t + d_y$

$z(t) = a_z t^3 + b_z t^2 + c_z t + d_z$

Cost: 9 multiplication
10 Sum

Forward differencing method

$$f(t) = t^3 + 3t^2 + 3x + 4 \quad ; \quad \sum_{t=a}^{b} f(t)$$

	$f(0)$	$f(1)$	$f(2)$	$f(3)$	$f(4)$	$f(5)$	$f(6)$
$\Rightarrow$	4	11	30	67	128	219	346
$\Rightarrow$	7	19	37	61	91	127	
$\Rightarrow$	12	18	21	30	36		
$\Rightarrow$	6	6	6	6			

Curve Rendering

Forward differencing method

$$f(t) = at^3 + bt^2 + ct + d$$

$$f(t+\delta) = a(t+\delta)^3 + b(t+\delta)^2 + c(t+\delta) + d$$

$$\Delta f(t) = f(t+\delta) - f(t)$$

$$= 3a\delta t^2 + (3a\delta^2 + 2b\delta)t + (a\delta^3 + b\delta^2 + c\delta)$$

$$\delta n = t_n \qquad \delta = \frac{t_n}{n}$$

$$f_{n+1} = f_n + \Delta f_n \quad \text{—— (i)}$$

$$\underline{f(t+\delta) = f(t) + \Delta f(t)}$$

$$\Delta f(t) = 3a\delta t^2 + (3a\delta^2 + 2b\delta)t + (a\delta^3 + b\delta^2 + c\delta)$$

$$\Delta f(t+\delta) = 3a\delta(t+\delta)^2 + (3a\delta^2 + 2b\delta)(t+\delta) + (a\delta^3 + b\delta^2 + c\delta)$$

$$\Delta^2 f(t) = \Delta f(t+\delta) - \Delta f(t)$$

$$= 6a\delta^2 t + (6a\delta^3 + 2b\delta^2)$$

$$\underline{\Delta f(t+\delta) = \Delta f(t) + \Delta^2 f(t)}$$

$$\Delta^2 f_n = \Delta f_{n+1} - \Delta f_n$$

$$\Delta^2 f_{n-1} = \Delta f_n - \Delta f_{n-1}$$

$$\Delta f_n = \Delta f_{n-1} + \Delta^2 f_{n-1} \quad \text{—— (ii)}$$

Curve Rendering

Forward differencing method

$$\Delta^2 f(t) = 6a\delta^2 t + (6a\delta^3 + 2b\delta^2)$$

$$\Delta^2 f(t+\delta) = 6a\delta^2 (t+\delta) + (6a\delta^3 + 2b\delta^2)$$

$$\Delta^3 f(t) = \Delta^2 f(t+\delta) - \Delta^2 f(t)$$

$$= 6a\delta^3$$

$$\underline{\Delta^2 f(t+\delta) = \Delta^2 f(t) + \Delta^3 f(t)}$$

$$\Delta^2 f_{n+1} = \Delta^2 f_n + \Delta^3 f_n$$

$$\Delta^2 f_{n-1} = \Delta^2 f_{n-2} + \Delta^3 f_{n-2} \quad \text{—— (iii)}$$

$$\boxed{\begin{aligned} f_o &= d \\ \Delta f_o &= a\delta^3 + b\delta^2 + c\delta \\ \Delta^2 f_o &= 6a\delta^3 + 2b\delta^2 \\ \Delta^3 f_o &= 6a\delta^3 \end{aligned}}$$

Ref:

Foley: Chapter 11