Curves

Why do we care about curves? Better approximate natural shapes and motion.

How? Use at least a cubic polynomial, which can be represented in either polynomial (parametric) or matrix form.

Parametric Form

$$x(u) = a_x u^3 + b_x u^2 + c_x u + dx$$

$$y(u) = a_y u^3 + b_y u^2 + c_y u + dy$$

where $0 \le u \le 1$

Matrix Form

$$x(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} a_x \\ b_x \\ c_x \\ dx \end{bmatrix} \qquad y(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} a_y \\ b_y \\ c_y \\ dy \end{bmatrix}$$

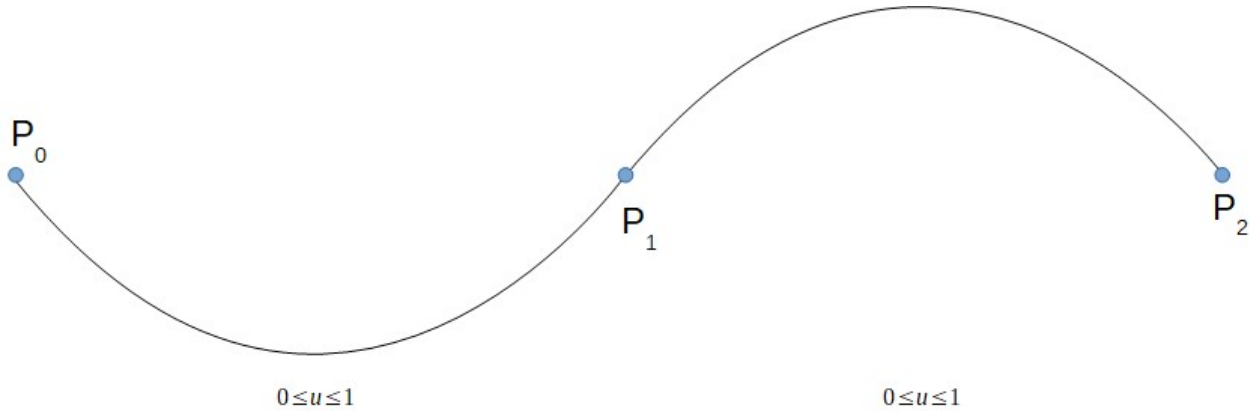
Which can be written as

$$x(u) = U \cdot C \quad \text{where} \quad C = M_{spline} \cdot M_{geom}$$

M_{spline} : Matrix of the spline character

M_{geom} : Control points and other boundary constraints

Spline: piece-wise computation of the curve



Mention the “Runge” phenomenon for why choosing cubic curves instead of higher order.

Problems

1. Need the points along the curve, P_0, P_1, … P_n : easy, we provide them
2. Need slopes at these points (to create a smooth curve) : not so easy (actually, pretty easy)

Solution to The Slope Problem

Remember, 1st derivative of an equation is the slope.

Going back to our parametric form of a curve...

$$x(u) = a_x u^3 + b_x u^2 + c_x u + dx$$

$$y(u) = a_y u^3 + b_y u^2 + c_y u + dy$$

1st derivative is…

$$\frac{\delta x}{\delta u} = 3 a_x u^2 + 2 b_x u + c_x$$

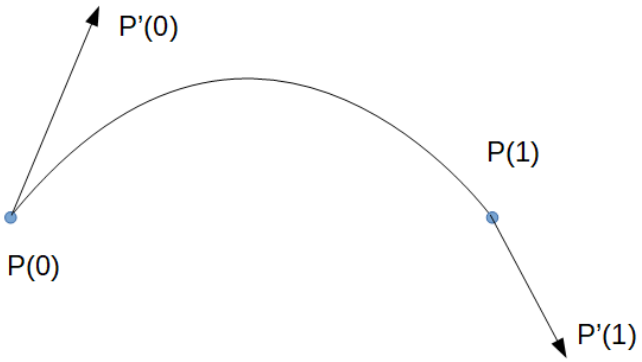
$$\frac{\delta y}{\delta u} = 3 a_y u^2 + 2 b_y u + c_y$$

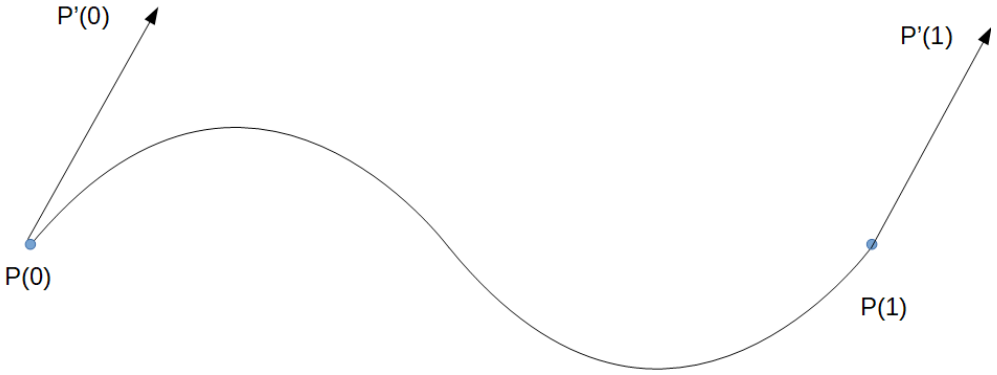
Therefore the slope is…

$$\frac{\delta y}{\delta x} = \frac{\delta y / \delta u}{\delta x / \delta u}$$

Hermite Curve (Spline)

This is an interpolating piece-wise cubic polynomial where the tangent is specified at each control point. *Interpolating* means the control points are on the curve itself.





Tangents are $\frac{\delta x}{\delta u}$ and $\frac{\delta y}{\delta u}$

Need to find: a_x, b_x, c_x, d_x and a_y, b_y, c_y, d_y with

$$x(0)=P(0)_x \qquad x'(0)=P'(0)_x$$

$$x(1)=P(1)_x \qquad x'(1)=P'(1)_x$$

plus same set of constraints for y(u) and y'(u)

Remember $x(u)=a_x u^3+b_x u^2+c_x u+dx$ In matrix form $x(u)=\begin{bmatrix}u^3 & u^2 & u & 1\end{bmatrix}\begin{bmatrix}a_x\\b_x\\c_x\\dx\end{bmatrix}$ or $U\cdot C_x$

C can be thought of as "control points". It is the two points and the slopes at those points.

Substitute for u and end points

$$x(0)=P(0)_x=[0\,0\,0\,1]\cdot C_x \quad \text{and} \quad x(1)=P(1)_x=[1\,1\,1\,1]\cdot C_x$$

Tangent vectors become...

$$\delta x/\delta u=x'(u)=\left[3u^2\,2u\,1\,0\right]\cdot C_x$$

$$x'(0)=[0\,0\,1\,0]\cdot C_x \quad \text{and} \quad x'(1)=[3\,2\,1\,0]\cdot C_x$$

This gives us 4 simultaneous equations. Putting them together in matrix form, we have...

$$\begin{bmatrix} P(0) \\ P(1) \\ P'(0) \\ P'(1) \end{bmatrix}_x = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} a_x \\ b_x \\ c_x \\ dx \end{bmatrix}$$

Rearranging to solve for a, b, c, d…

$$\begin{bmatrix} a_x \\ b_x \\ c_x \\ dx \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \cdot \begin{bmatrix} P(0) \\ P(1) \\ P'(0) \\ P'(1) \end{bmatrix}_x$$ we need M_H, but have M_H^{-1} $\quad P(u)=M_H\cdot M_g$

Need to invert M_H : $\quad H^{-1}\cdot H=I$; after inverting, we get...

$$\begin{bmatrix} a_x \\ b_x \\ c_x \\ dx \end{bmatrix} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} P(0) \\ P(1) \\ P'(0) \\ P'(1) \end{bmatrix}_x \qquad M_H = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

The resulting polynomial is...

$$P(u)=U\cdot M_H\cdot M_g \quad \text{where} \quad U=\left[u^3\,u^2\,u\,1\right]$$

M_H is the Hermite matrix

Going back to parametric form

$$x(u)=U\cdot M_H\cdot M_{g_x} \qquad y(u)=U\cdot M_H\cdot M_{g_y}$$

Expanding we get…

$$x(u)=P(0)_x(2u^3-3u^2+1)+P(1)_x(-2u^3+3u^2)+P'(0)_x(u^3-2u^2+u)+P'(1)_x(u^3-u^2)$$

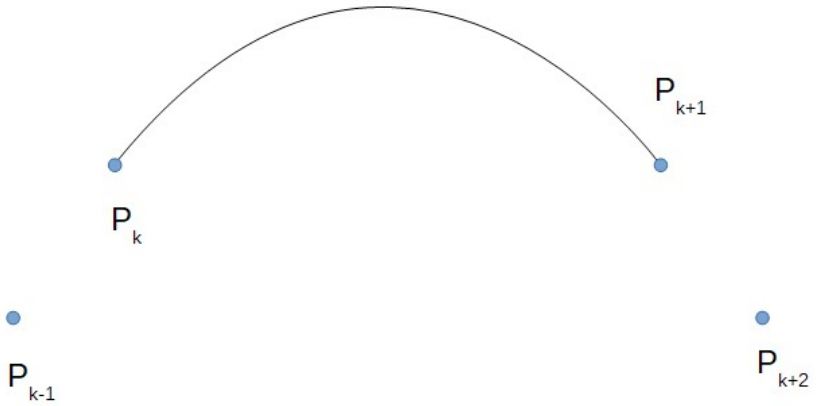
$$y(u)=P(0)_y(2u^3-3u^2+1)+P(1)_y(-2u^3+3u^2)+P'(0)_y(u^3-2u^2+u)+P'(1)_y(u^3-u^2)$$

This is called the Hermite blending function.

Cardinal Splines

Also known as Catmull-Rom spline/curve when t = 0

An interpolating curve, very similar to Hermite splines, with the difference being that rather than specifying tangents at the control points, the slope is computed from points adjacent to the curve control points.



We then define the curve boundary conditions as follows:

$$P(0)=P_k \qquad P(1)=P_{k+1}$$

$$P'(0)=\frac{1}{2}(1-t)(P_{k+1}-P_{k-1}) \qquad P'(1)=\frac{1}{2}(1-t)(P_{k+2}-P_k)$$

Parameter *t* is called the *tension*, which controls how tightly or loosely the curve fits the control points.

The general form of the curve equation is:

$$P(u)=U\cdot M_{spline}\cdot M_{geometry}$$

where $M_{spline}=\begin{bmatrix} -s & 2-s & s-2 & s \\ 2s & s-3 & 3-2s & -s \\ -s & 0 & s & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$ with $s=\frac{(1-t)}{2}$

$$P(u)=[u^3\,u^2\,u1]\cdot\begin{bmatrix} -s & 2-s & s-2 & s \\ 2s & s-3 & 3-2s & -s \\ -s & 0 & s & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}\cdot\begin{bmatrix} P_{k-1} \\ P_k \\ P_{k+1} \\ P_{k+2} \end{bmatrix}$$

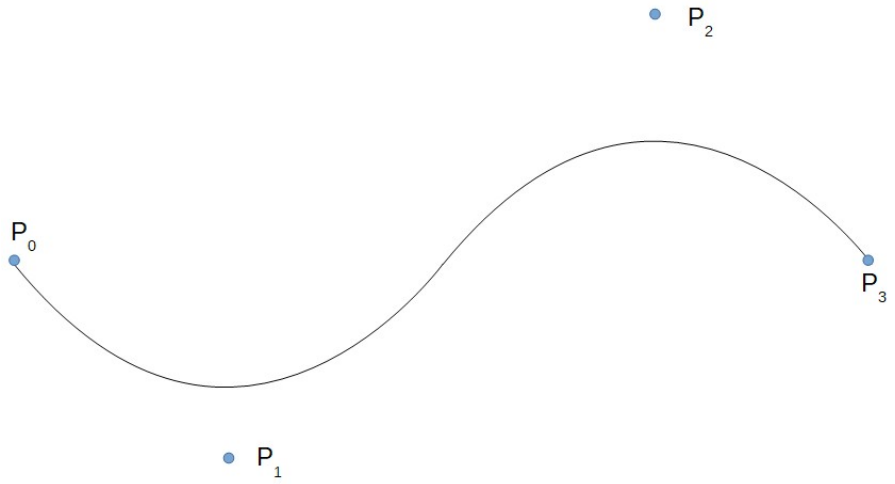
Expanding into polynomial form, gives us the blending function:

$$P(u)=P_{k-1}(-su^3+2su^2-su)+P_k[(2-s)u^3+(s-3)u^2+1]+P_{k+1}[(s-2)u^3+(3-2s)u^2+su]+P_{k+2}(su^3-su^2)$$

...where this equation is computed once for x and once for y...

Bézier Curves

An interpolating spline (through start and end points, but not intermediate) developed by Pierre Bézier for use in designing Renault automobile bodies. Because of their properties, they are used to define the curves for fonts and used for font rendering.



A Bézier curve can be fitted to any number of control points using a blending function, where the number of points (in practice) is generally kept small, such as 3 or 4. The general form for the curve looks like…

$$P(u)=\sum_{k=0}^{n} P_k \, BEZ_{k,n}(u) \quad \text{for} \quad 0 \le u \le 1$$

with n = degree of curve; e.g., a Bézier curve of 4 points is degree 3.

where

$$BEZ_{k,n}(u)=C(n,k)u^k(1-u)^{n-k}$$ This comes from the *Bernstein polynomials*

where C(n,k) are the binomial coefficients

$$C(n,k)=\frac{n!}{k!(n-k)!}$$

This looks bad, but it isn't, because *n* and *k* are small, and all values can be pre-computed to improve performance.

For a cubic (4 control points) Bézier curve, can follow a similar procedure as with Hermite curves to find the Bézier matrix. When we do, we find it to be:

$$P(u)=\begin{bmatrix}u^3 u^2 u 1\end{bmatrix}\cdot\begin{bmatrix}1 & -3 & 3 & -1\\ 0 & 3 & -6 & 3\\ 0 & 0 & 3 & -3\\ 0 & 0 & 0 & 1\end{bmatrix}\cdot\begin{bmatrix}P_0\\ P_1\\ P_2\\ P_3\end{bmatrix}$$