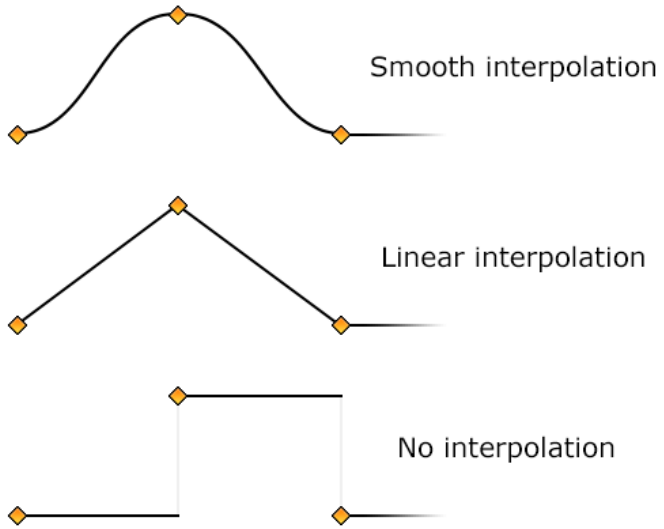
关键帧插值和速度控制



金小刚

Email: jin@cad.zju.edu.cn

浙江大学CAD&CG系统全国重点实验室

紫金港校区蒙民伟楼512

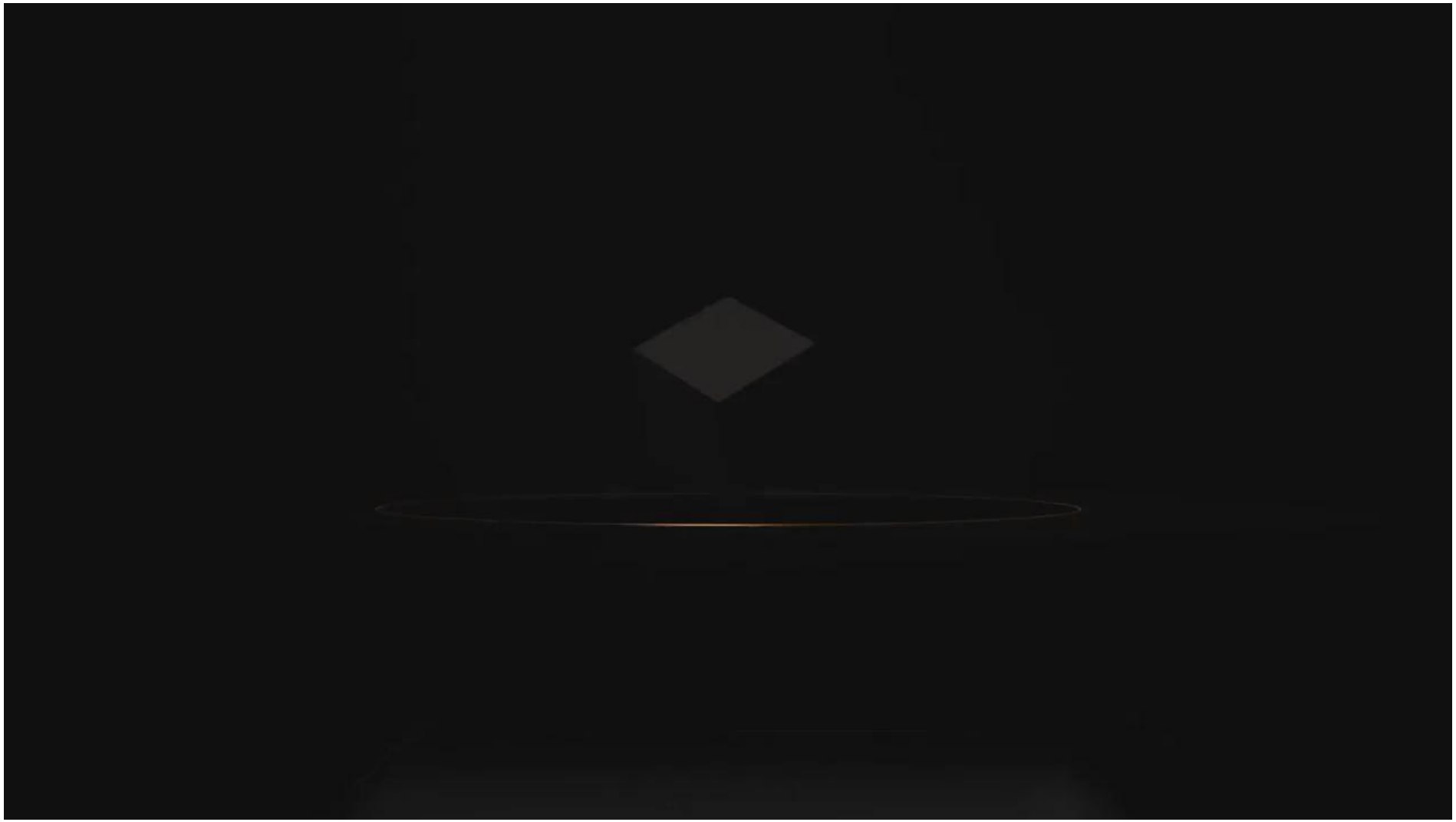
动画中的运动控制

- 运动的**表示**：用数学方法来表示各类物体的运动
- 运动的**控制和编辑**：给动画师提供能够表达他意图的方便、直观的控制工具
- 运动的**生成**：计算每一帧所需的各种运动参数

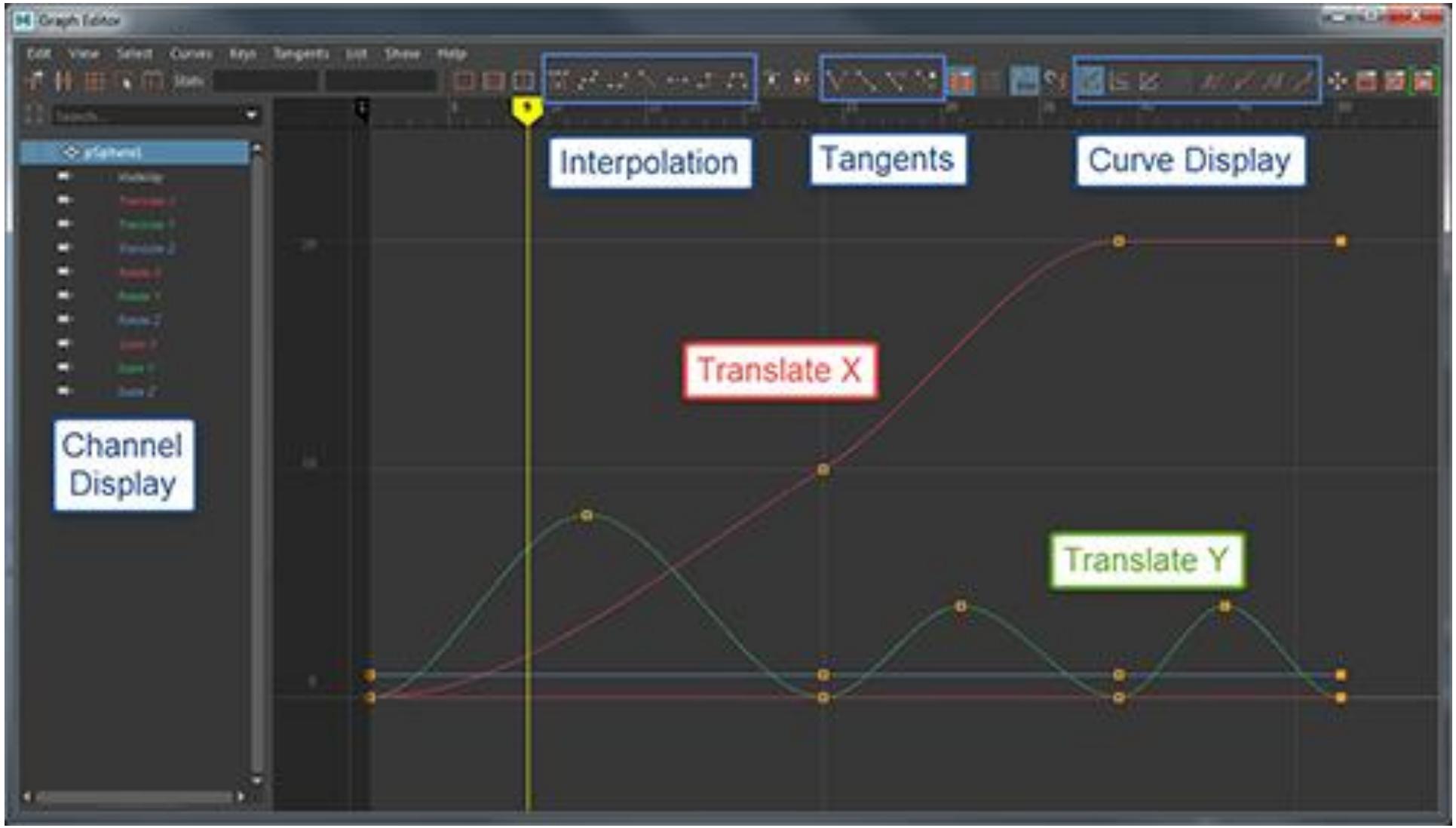
目标：再现物理世界的运动，超越物理世界的运动

Maya中的插值（Keyframe)

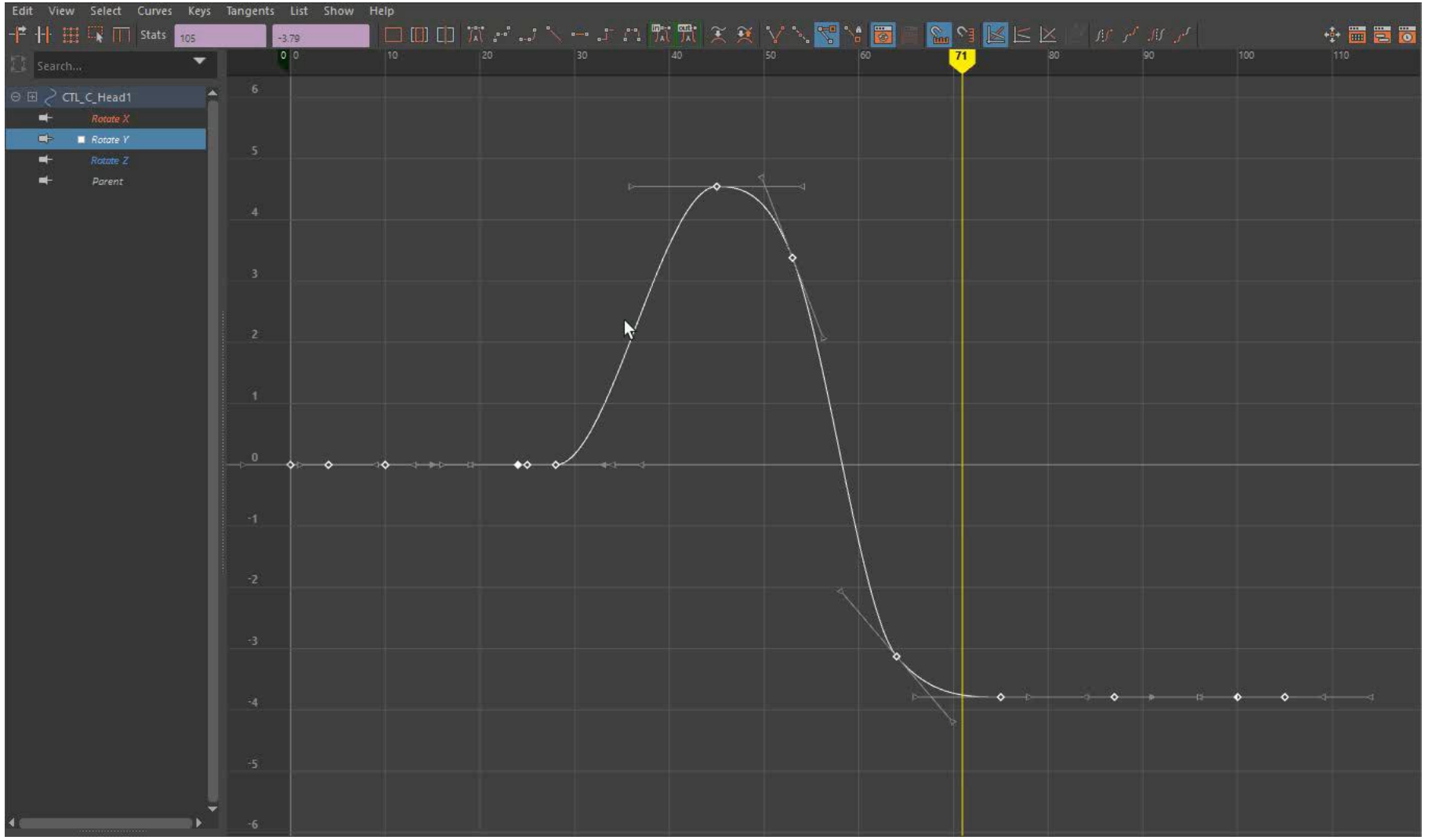
——位置、方向、比例缩放、颜色、形状...



Maya曲线图编辑器(Graph Editor)



Maya曲线图编辑器(Graph Editor)-样条



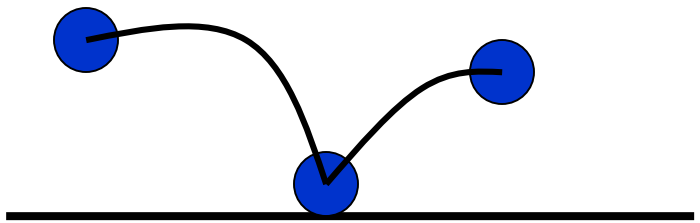
Interpolation between key frames

- Parameters to be interpolated
  - Position of an object
  - Scale of an object
  - Orientation of an object
  - Joint angle between two joints
  - Color attribute of an object
  - …

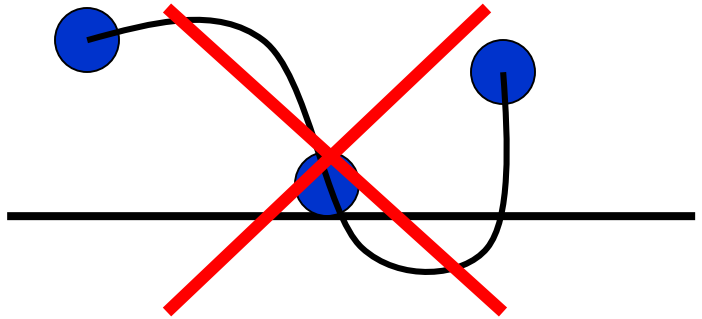
Interpolation between key frames

- Interpolation between frames is not trivial
  - Appropriate **interpolating function**
  - **Parameterization** of the function
  - Maintain the **desired control** of the interpolated values over time
- Example
  - (-5,0,0) at frame 22, (5,0,0) at frame 67
    - Stop at frame 22 and accelerate to reach a max speed by frame 34
    - Start to decelerate at frame 50 and come to a stop at frame 67

Interpolation between key frames



desired result

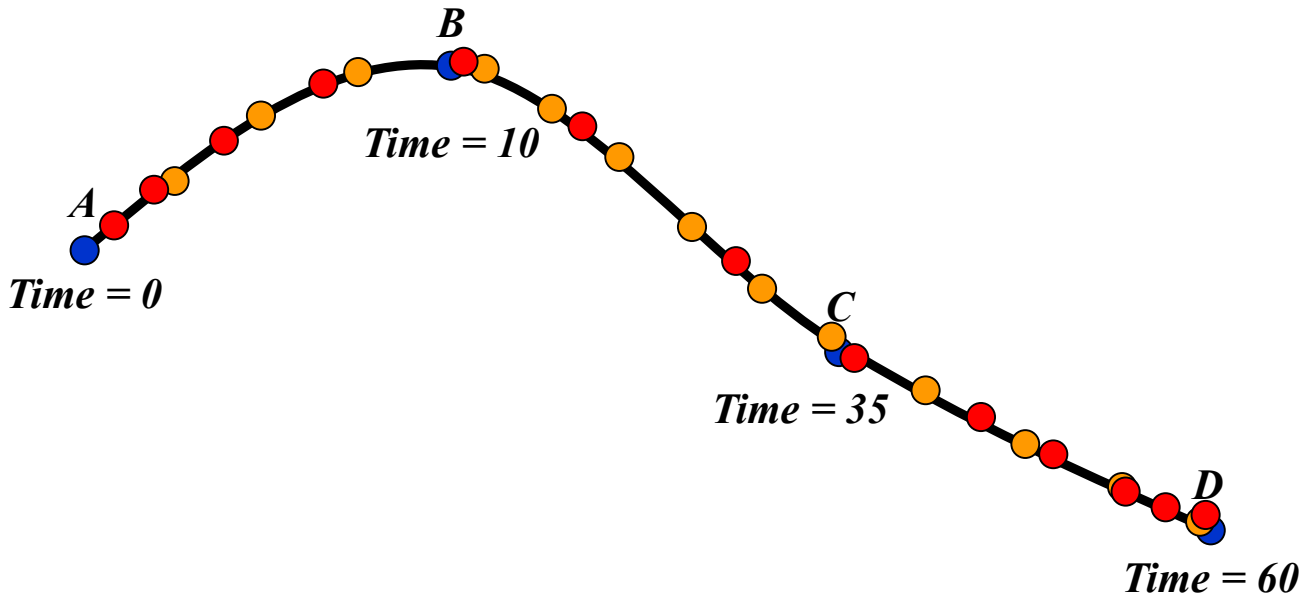


undesired result

Need a smooth interpolation with user control

Interpolation between key frames

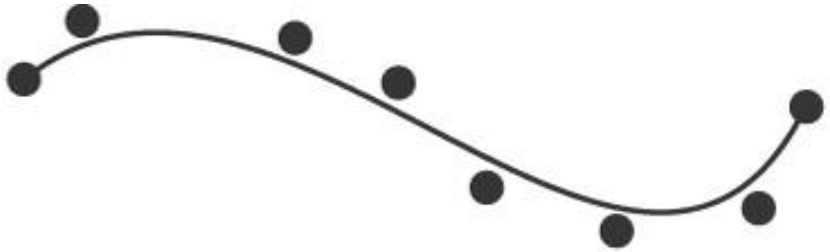
- Solution
  - Generate a space curve
  - Distribute points evenly along curve
  - Speed control: vary points temporally(时间上)



Interpolation functions

- Interpolation vs. approximation

An interpolating spline in which the spline passes through the interior control points

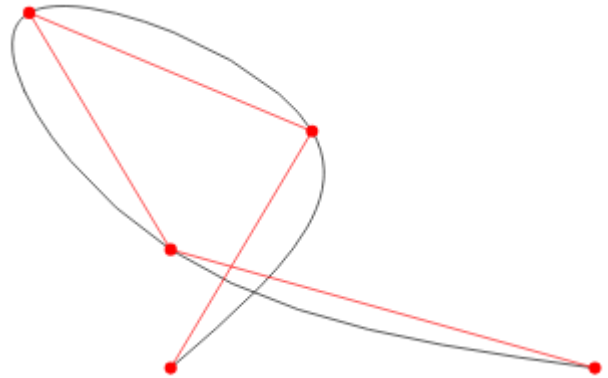


An approximating spline in which only the endpoints are interpolated; the interior control points are used only to design the curve

- Interpolation
  - Hermite, Bezier, Catmull-Rom, …
- Approximation
  - Bezier, B-spline, NURBS(**N**on-**U**niform **R**ational **B**-**S**pline), …

Interpolation functions

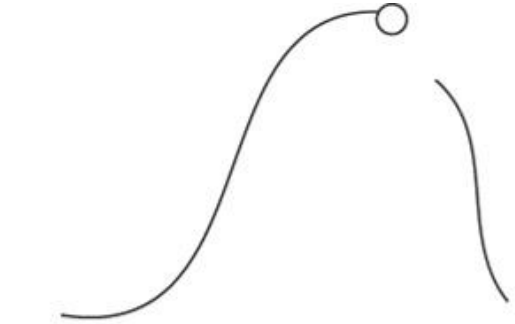
- Complexity => computational efficiency
  - Polynomials
    - Lower than **cubic(三次）**
      - No inflection point (拐点), may not fit smoothly to some data points
    - Higher than cubic
      - Doesn’t provide any significant advantages, costly to evaluate
    - **Piecewise(分段)** cubic
      - Provides sufficient smoothness
      - Allows enough flexibility to satisfy constraints such as end-point position and tangents



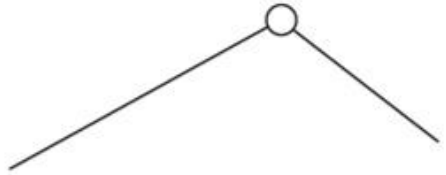
Interpolation functions

- Continuity within a curve
  - Zero-order (位置连续)
  - First order (tangential) (切向连续)
    - Suffices for most animation applications
  - Second order (曲率连续)
    - Important when dealing with time-distance curve
- Continuity between curve segments
  - Hermite, Catmull-Rom, cubic Bezier provide first order continuity between segments

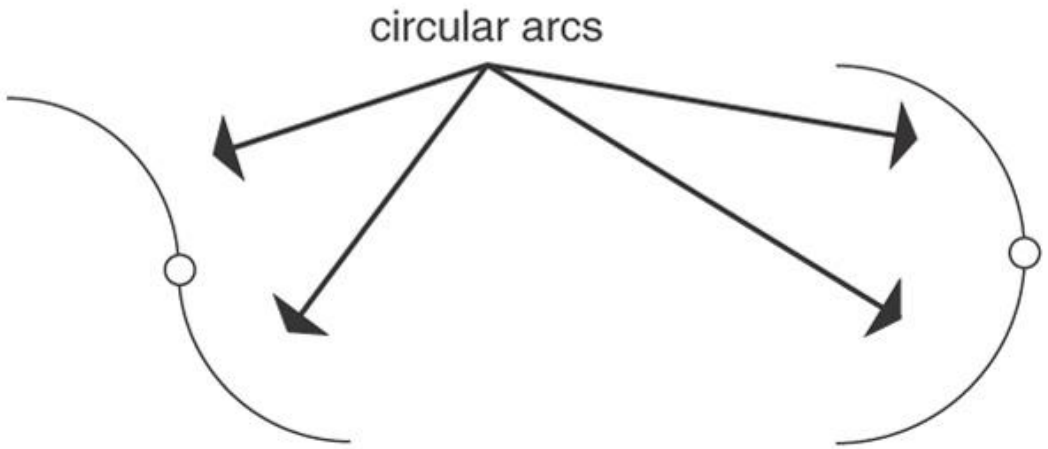
Interpolation functions



Positional discontinuity at the point



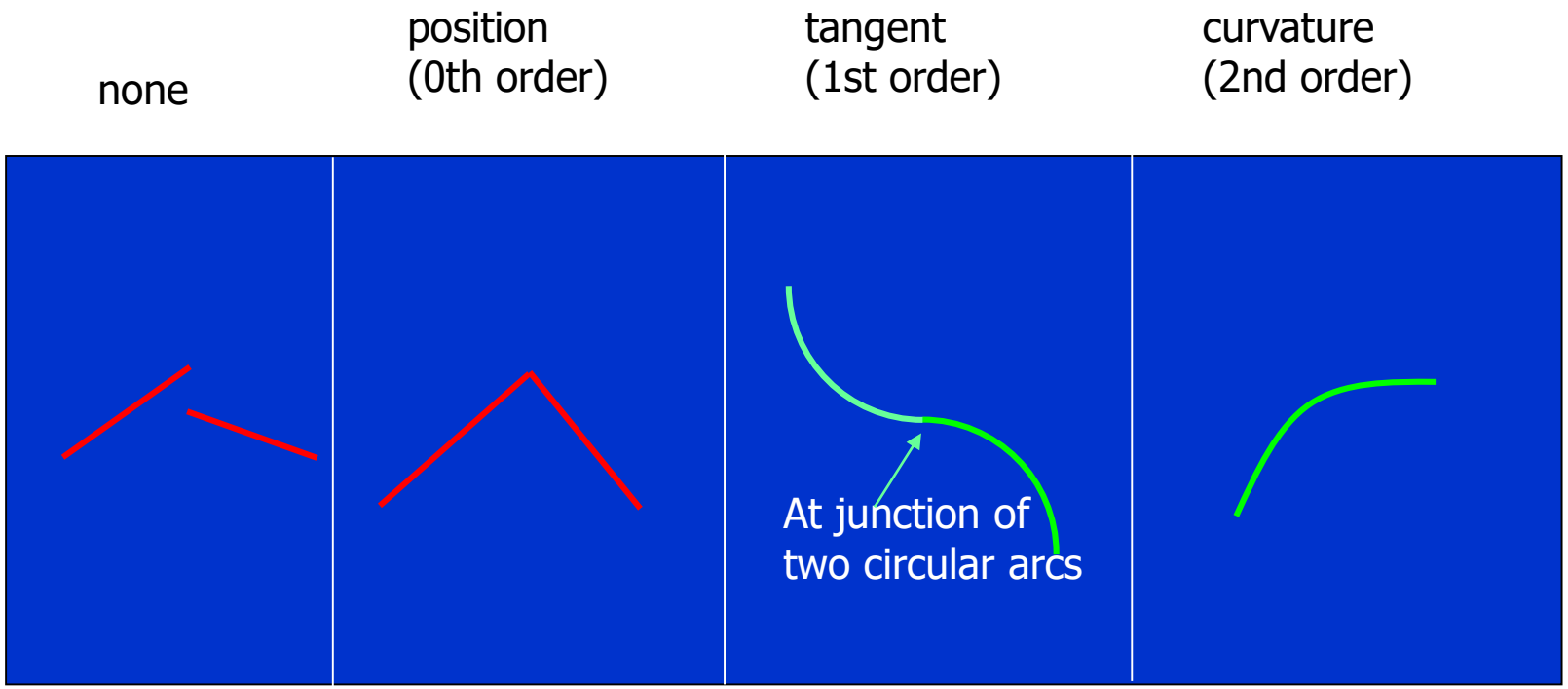
Positional continuity but not tangential continuity at the point



Positional and tangential continuity but not curvature continuity at the point

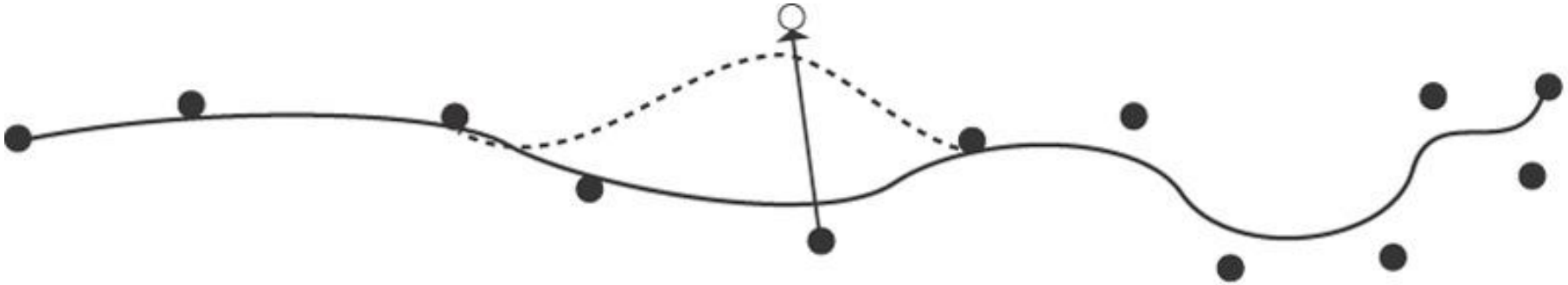
Positional, tangential, and curvature continuity at the point

Continuity

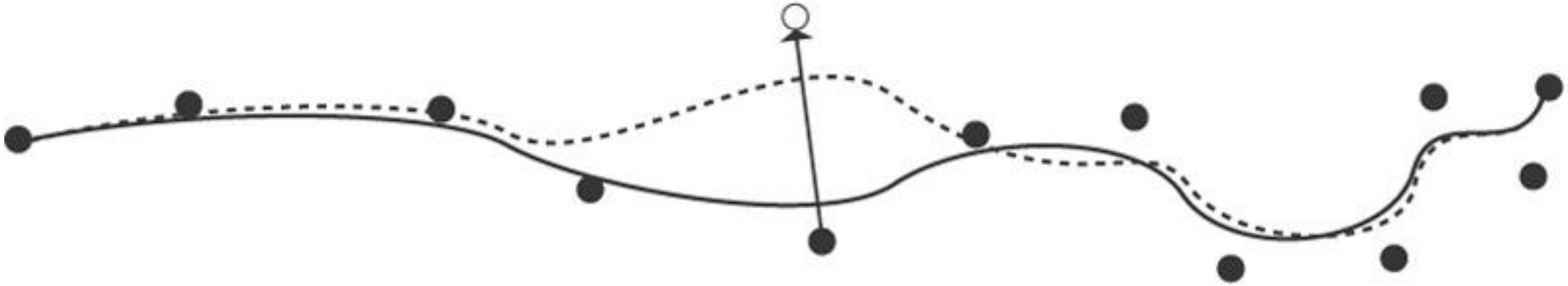

none
position
(0th order)
tangent
(1st order)
curvature
(2nd order)
At junction of
two circular arcs

Interpolation functions

- Global vs. local control

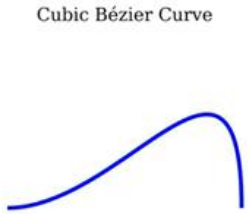


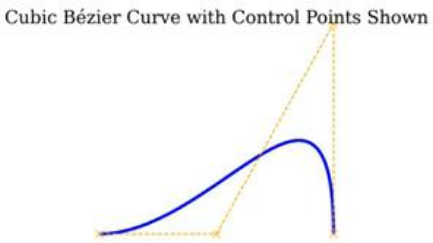
Local control: moving one control point only changes the curve over a finite bounded region



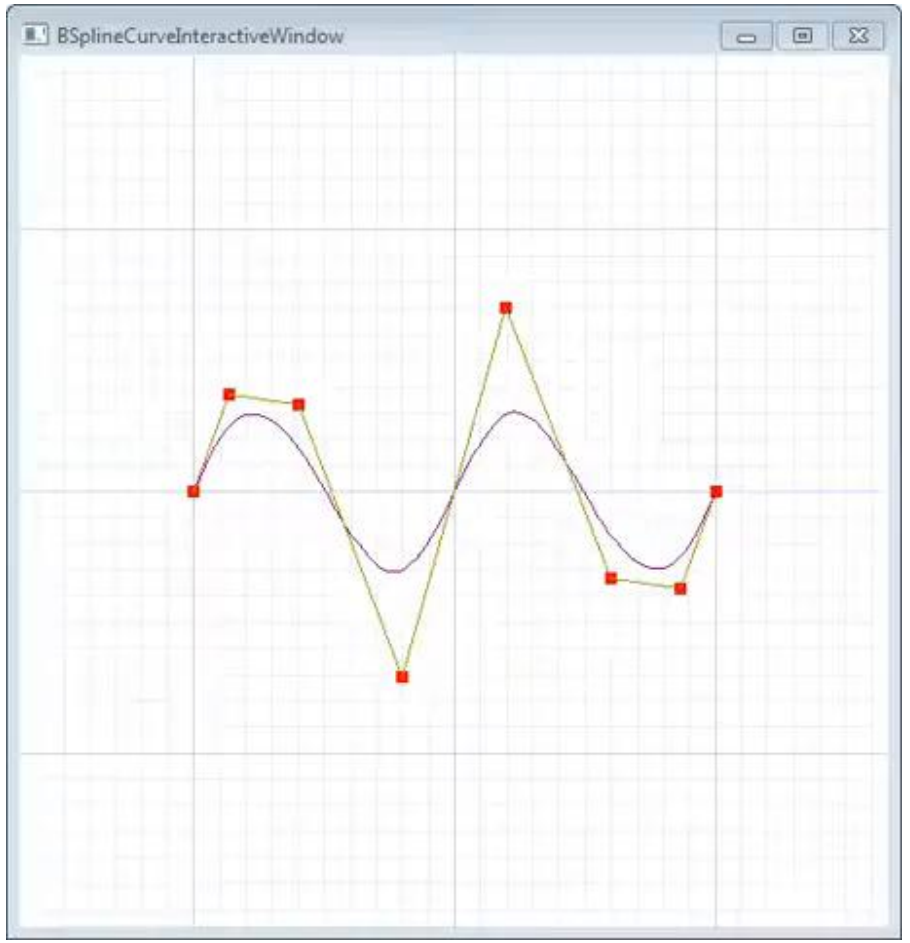
Global control: moving one control point changes the entire curve; distant sections may change only slightly

Global vs. local control





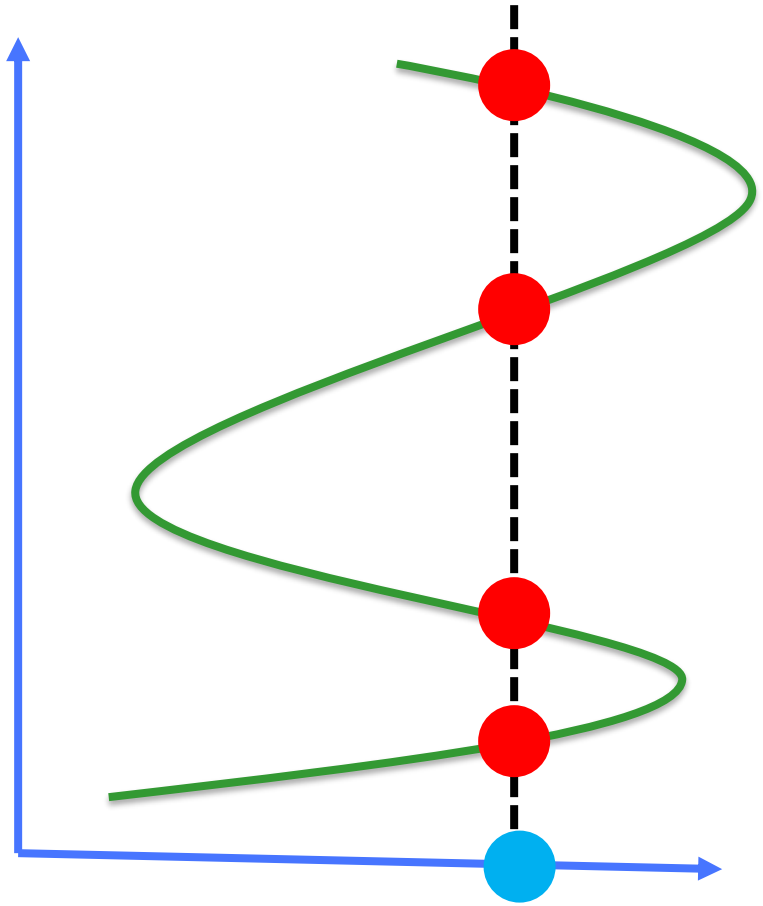
Global control



Local control

Types of Curve Representation

- Explicit $y = f(x)$
  - Good for generating points
  - For each input, there is a unique output
- Implicit $f(x,y) = 0$
  - Good for testing if a point is on a curve
  - Bad for generating a sequence of points
- **Parametric** $x = f(u), y = g(u)$
  - Good for generating a sequence of points
  - Can be used for **multi-valued** function of x



***multi-valued* function**

Example: Representing Unit Circle

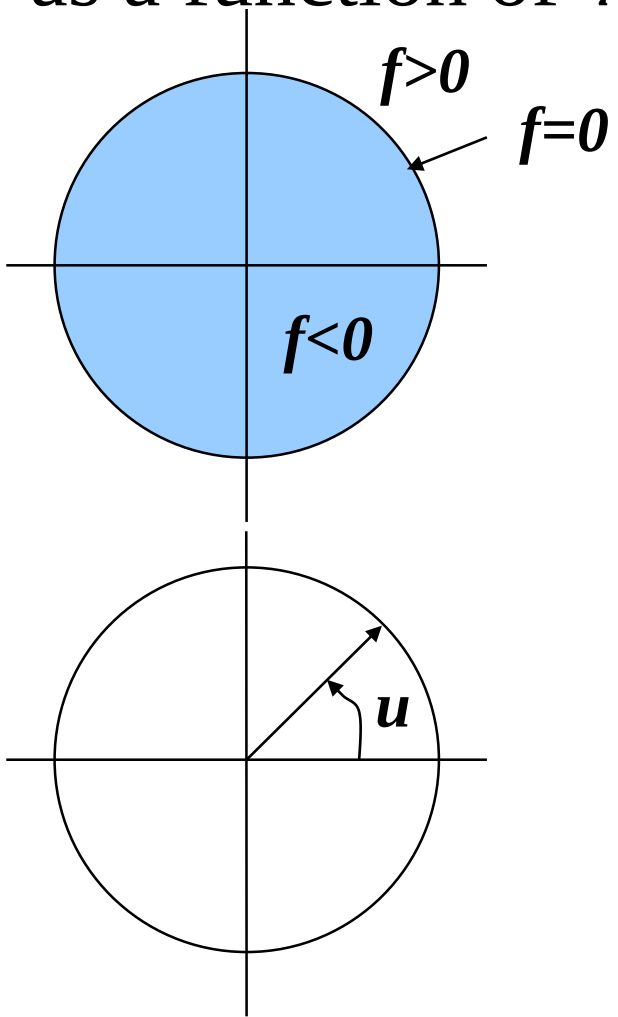
- Explicit: Cannot be represented explicitly as a function of x

- Implicit form:

  $f(x,y)=x^2+y^2-1=0$

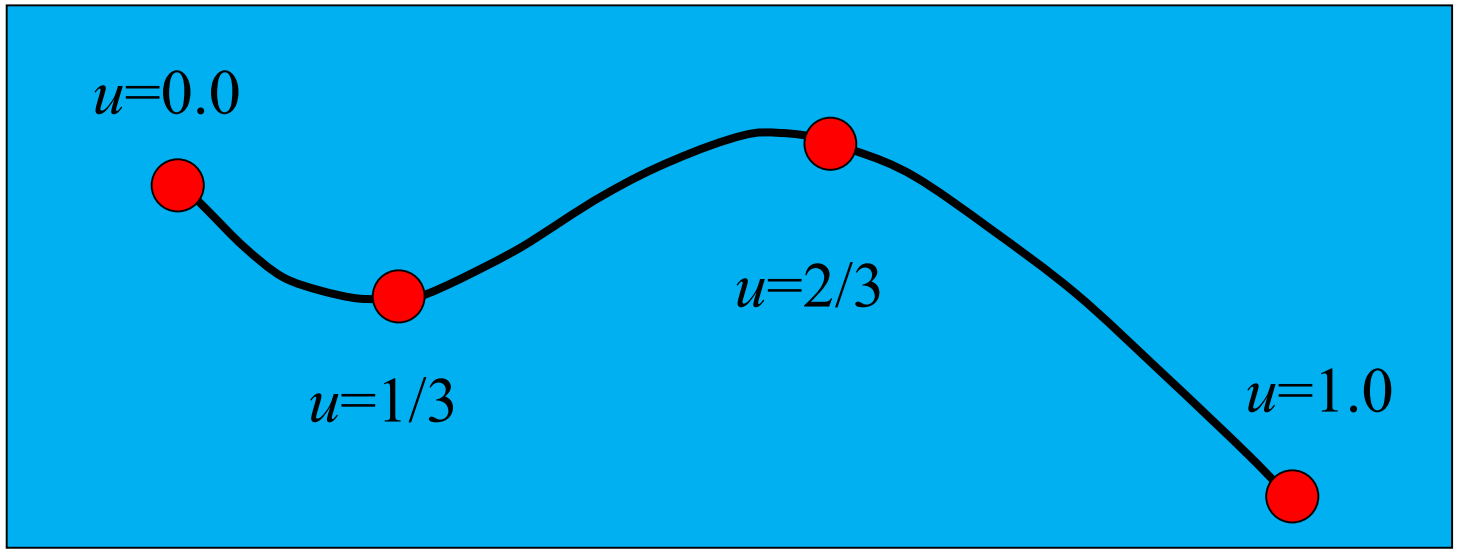
- Parametric form:

  $x=\cos(u),\ y=\sin(u),\ 0<u<2\pi$



More on 3-D Parametric Curves

- Parametric form: $P(u) = (P_x(u), P_y(u), P_z(u))$
  - $x = P_x(u), y = P_y(u), z = P_z(u)$



Space-curve $P = P(u)$ $0.0 <= u <= 1.0$

Polynomial Interpolation

- An *n*-th degree polynomial fits a curve to *n+1* points
  - Example: fit a second degree curve to three points
    - $\mathbf{y(x) = a\,x^2 + b\,x + c}$
    - points to be interpolated

      $\mathbf{(x_1, y_1), (x_2, y_2), (x_3, y_3)}$
    - solve for coefficients **(a, b, c):**

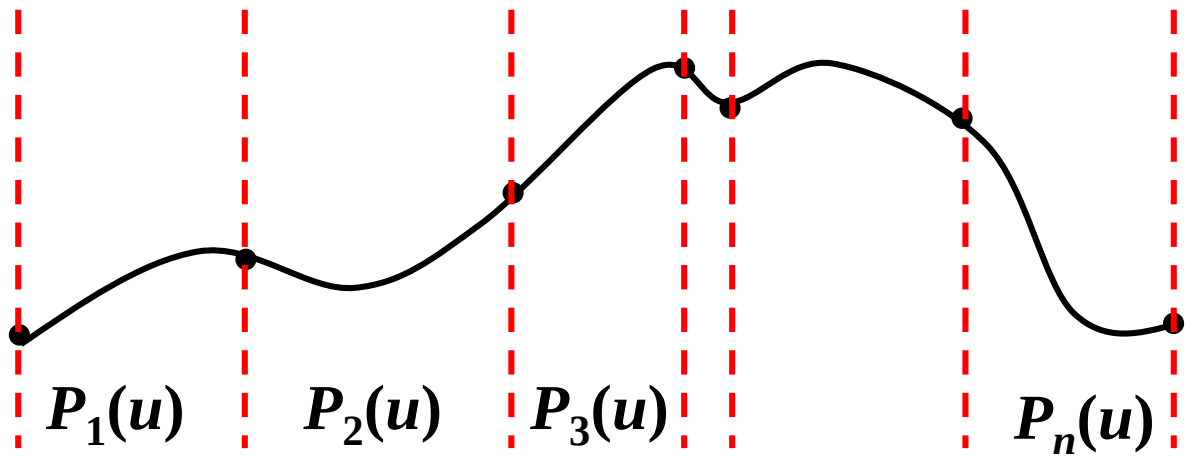
      3 linear equations, 3 unknowns
- called **Lagrange Interpolation**

Polynomial Interpolation (cont.)

- Result is a curve that is too wiggly (波动的,即样条中常提到的振荡), change to any control point affects entire curve (nonlocal) – *this method is poor*
- We usually want the curve to be as smooth as possible
  - minimize the wiggles
  - high-degree polynomials are bad
    - **Higher degree, higher the wiggles!**

Composite Segments

- Divide a curve into multiple segments
- Represent each in a parametric form
- Maintain continuity between segments
  - position
  - Tangent (C1-continuity vs. G1-continuity)
  - curvature

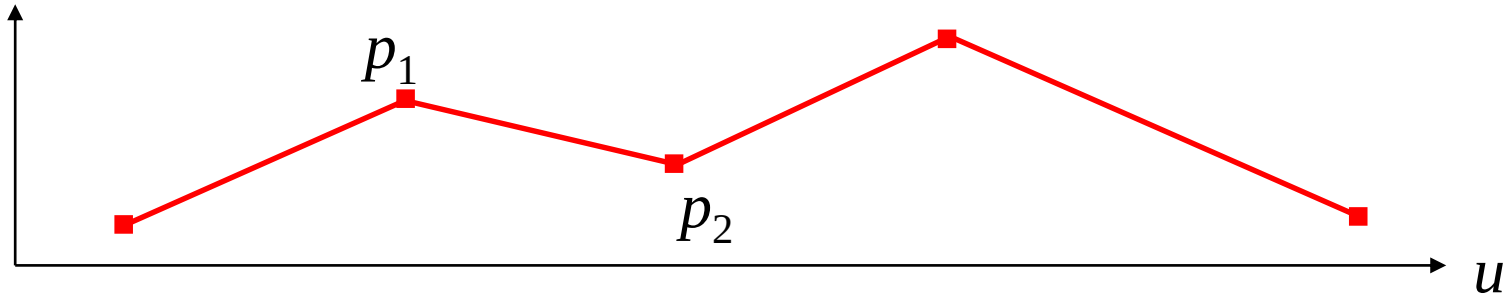


Splines: Piecewise Polynomials

- A **spline** is a **piecewise** polynomial - **many low degree polynomials** are used to interpolate (pass through) the control points
- **Cubic polynomials are the most common**
  - lowest order polynomials that interpolate two points and allow the gradient at each point to be defined - C1 continuity is possible
  - Higher or lower degrees are possible, of course

A Linear Piecewise Polynomial

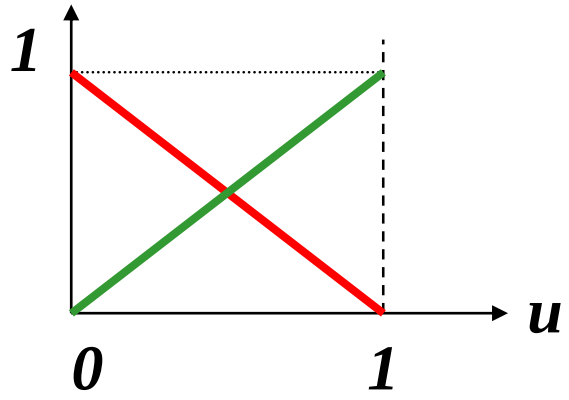
The simple form for interpolating two points.



Each segment is of the form: (this is a vector equation)

$$p(u) = up_1 + (1-u)\,p_2$$

Two basis (blending) functions



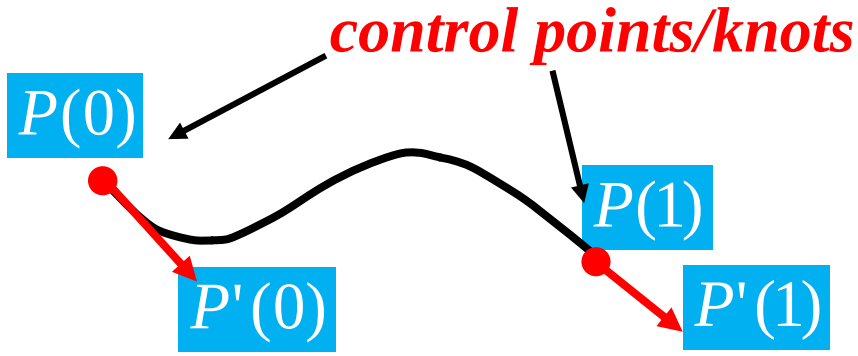
Hermite Interpolation

- Hermite Curves—cubic polynomial for two points

$$P(u) = au^3 + bu^2 + cu + d$$

$$P(u) = (P_x(u), P_y(u), P_z(u))$$

- Hermite interpolation requires
  - Endpoint positions
  - derivatives at endpoints



- To create a composite curve, use the end of one as the beginning of the other and share the tangent vector

Hermite Curve Formation

- Cubic polynomial and its derivative

$$P_x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$$

$$P_x'(u) = 3a_x u^2 + 2b_x u + c_x$$

- Given $P_x(0)$, $P_x(1)$, $P'_x(0)$, $P'_x(1)$, solve for a, b, c, d
  - 4 equations are given for 4 unknowns

$$P_x(0) = d_x$$

$$P_x(1) = a_x + b_x + c_x + d_x$$

$$P_x'(0) = c_x$$

$$P_x'(1) = 3a_x + 2b_x + c_x$$

Hermite Curve Formation (cont.)

- Problem: solve for a, b, c, d

$$P_x(0) = d_x$$
$$P_x(1) = a_x + b_x + c_x + d_x$$
$$P_x'(0) = c_x$$
$$P_x'(1) = 3a_x + 2b_x + c_x$$

- Solution:

$$a_x = 2(P_x(0) - P_x(1)) + P_x'(0) + P_x'(1)$$
$$b_x = 3(P_x(1) - P_x(0)) - 2P_x'(0) - P_x'(1)$$
$$c_x = P_x'(0)$$
$$d_x = P_x(0)$$

Hermite Curves in Matrix Form

$$P(u) = au^3 + bu^2 + cu + d$$

$$P(u) = U^T MB$$

$U^T = [u^3, u^2, u, 1]$ is the parameter

M is the coefficient matrix

B is the geometric information

$$\mathrm{M} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathrm{B} = \begin{bmatrix} P_x(0) \\ P_x(1) \\ P_x'(0) \\ P_x'(1) \end{bmatrix}$$

ith segment in composite curves →

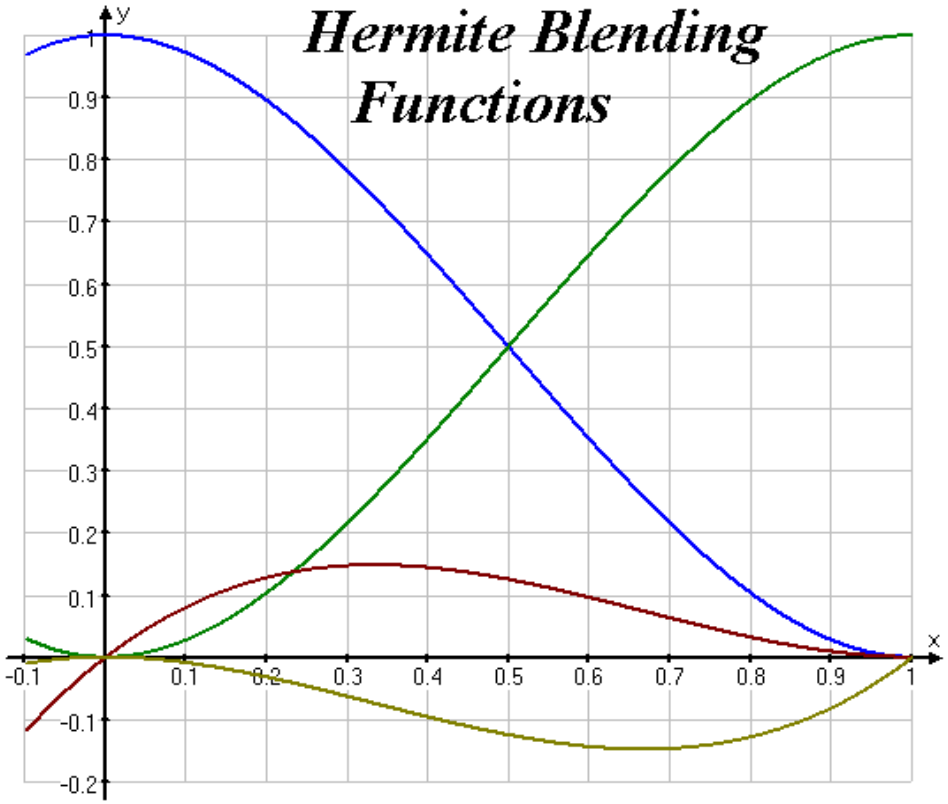
$$\mathrm{B} = \begin{bmatrix} P_i \\ P_{i+1} \\ P_i' \\ P_{i+1}' \end{bmatrix}$$

Blending Functions of Hermite Splines

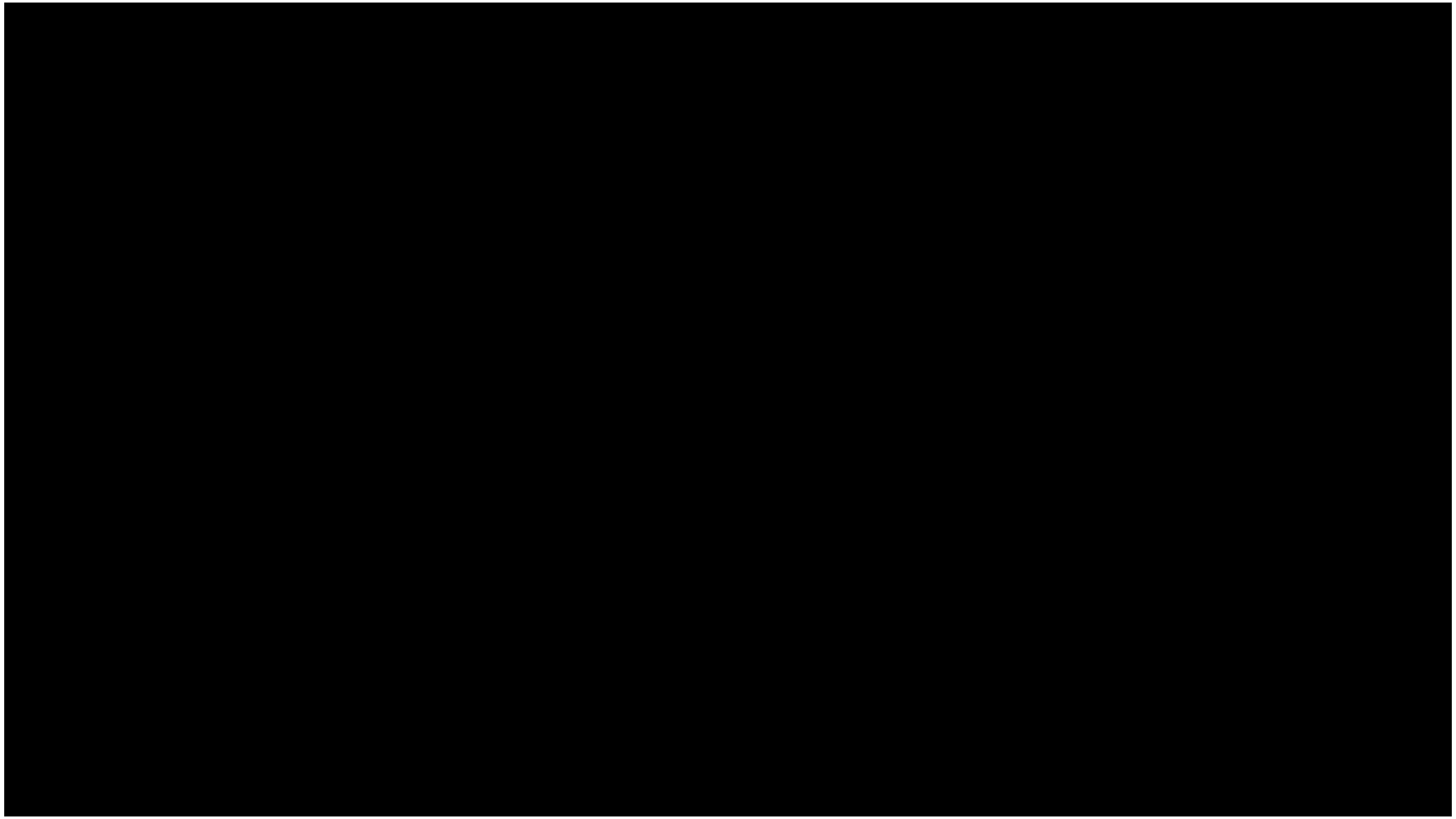
- Each cubic Hermite spline is a linear combination of 4 blending functions

$$P(u) = U^T M B$$

$$p(u) = \begin{bmatrix} 2u^3 - 3u^2 + 1 \\ -2u^3 + 3u^2 \\ u^3 - 2u^2 + u \\ u^3 - u^2 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p'_1 \\ p'_2 \end{bmatrix}$$

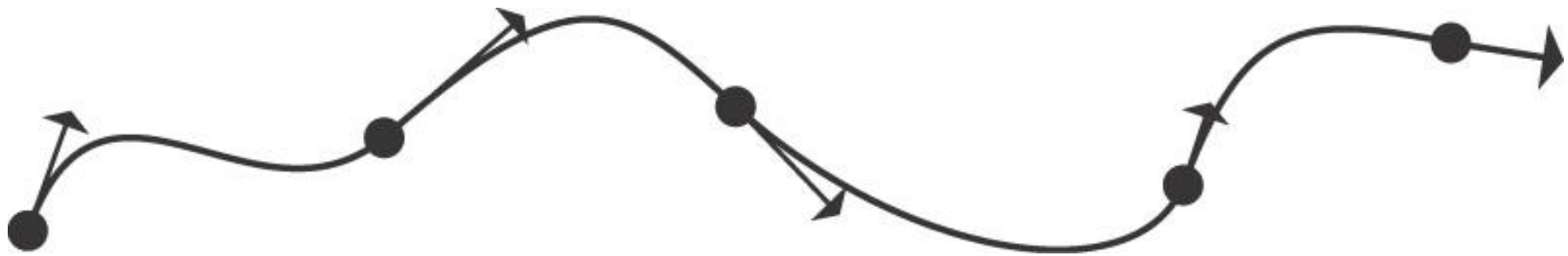


Hermite Curves demo

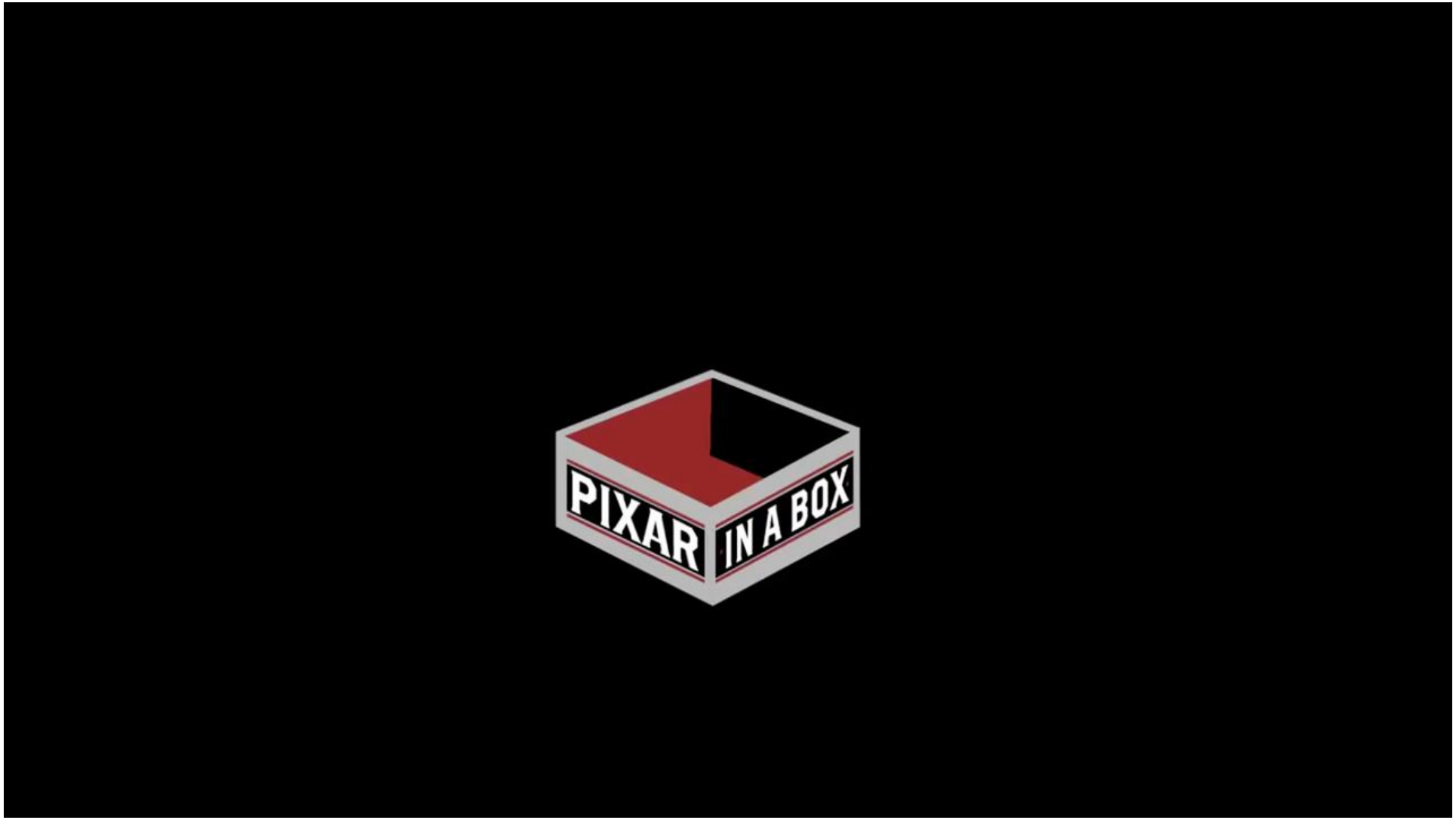


Composite Hermite Curve

- Continuity between connected segments is ensured by using ending tangent vector of one segment as the beginning tangent vector of the next.



Bezier Curves



Bezier Curves

- Similar to Hermite Curve
- Instead of endpoints and tangents, four control points are given
  - points P_1 and P_4 are on the curve
  - points P_2 and P_3 are used to control the shape
  - $p_1 = P_1$, $p_2 = P_4$,
  - $p_1' = 3(P_2 - P_1)$, $p_2' = 3(P_4 - P_3)$

$$p(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$$

Bezier Curves

- Another representation
  - Blend the control point position using Bernstein polynomials

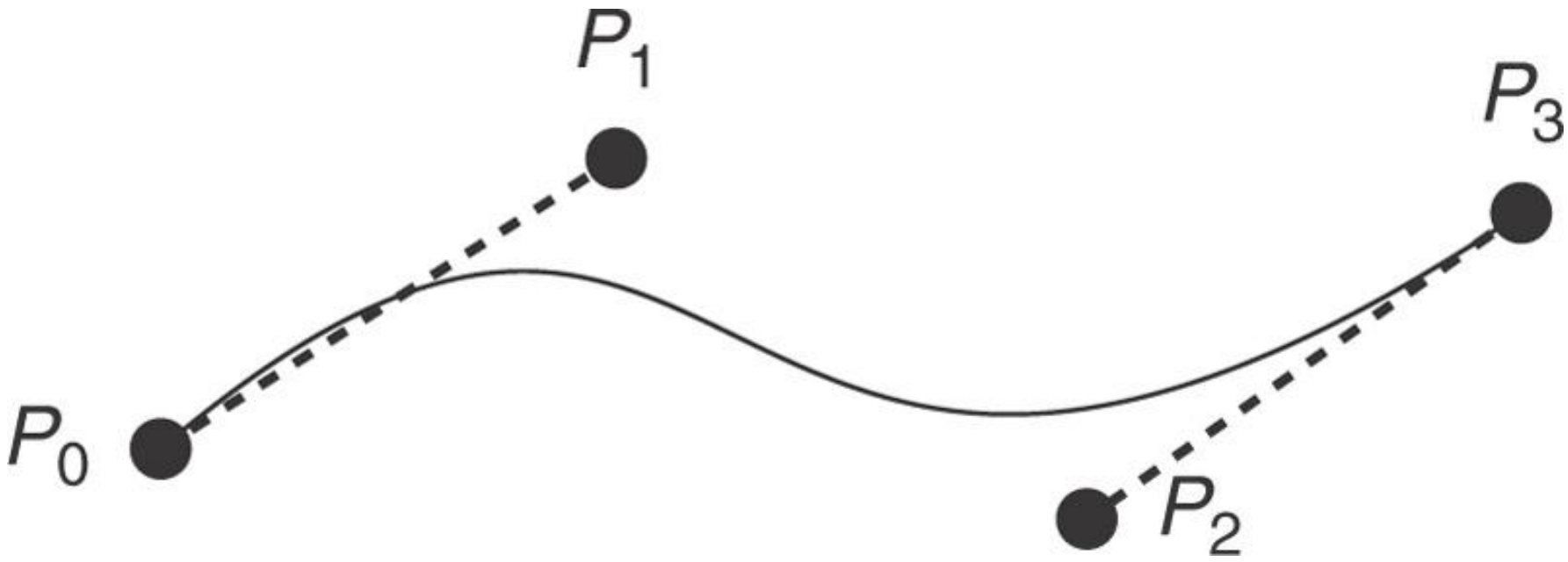
$$p(u) = \sum_{k=0}^{n} B_{k,n}(u) P_k$$

where $B_{k,n}(u)$ are Bernstein polynomials

$$B_{k,n}(u) = C(n,k) u^k (1-u)^{n-k}$$

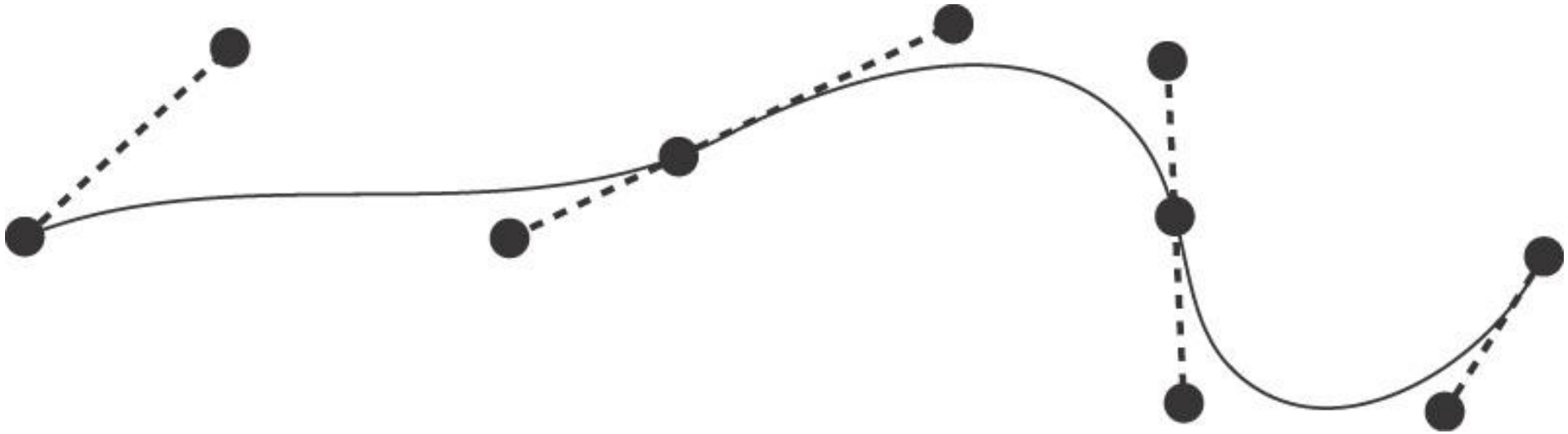
$$C(n,k) = \frac{n!}{k!(n-k)!}$$

Bezier Curves



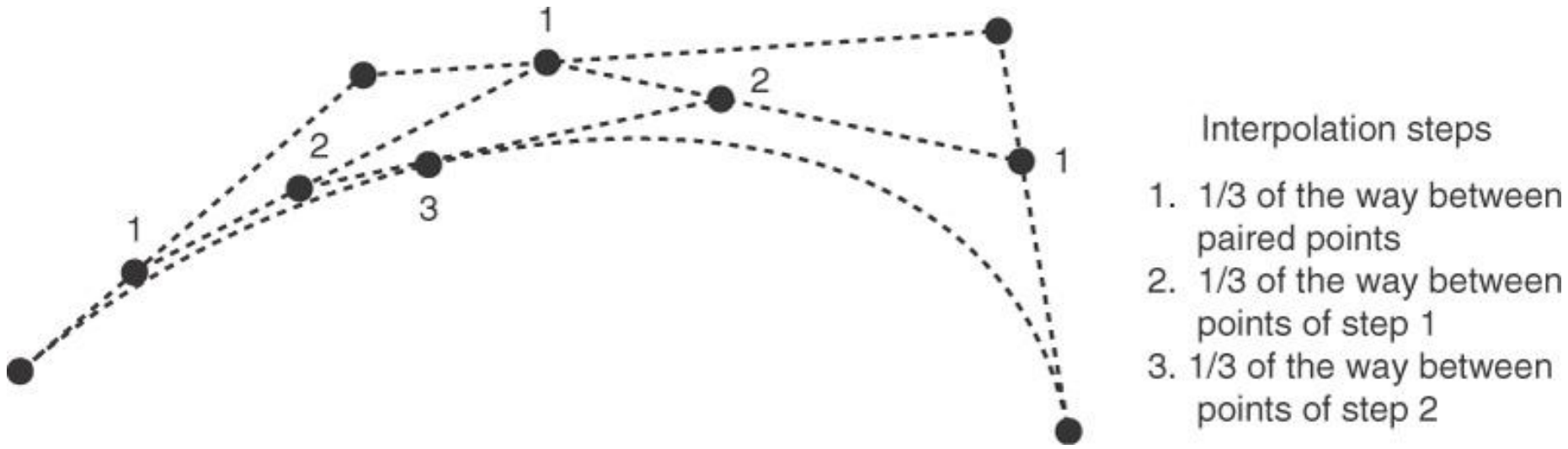
Composite Bezier Curves

- How to control the continuity between adjacent Bezier segment?
  - by using ending tangent vector of one segment as the beginning tangent vector of the next.



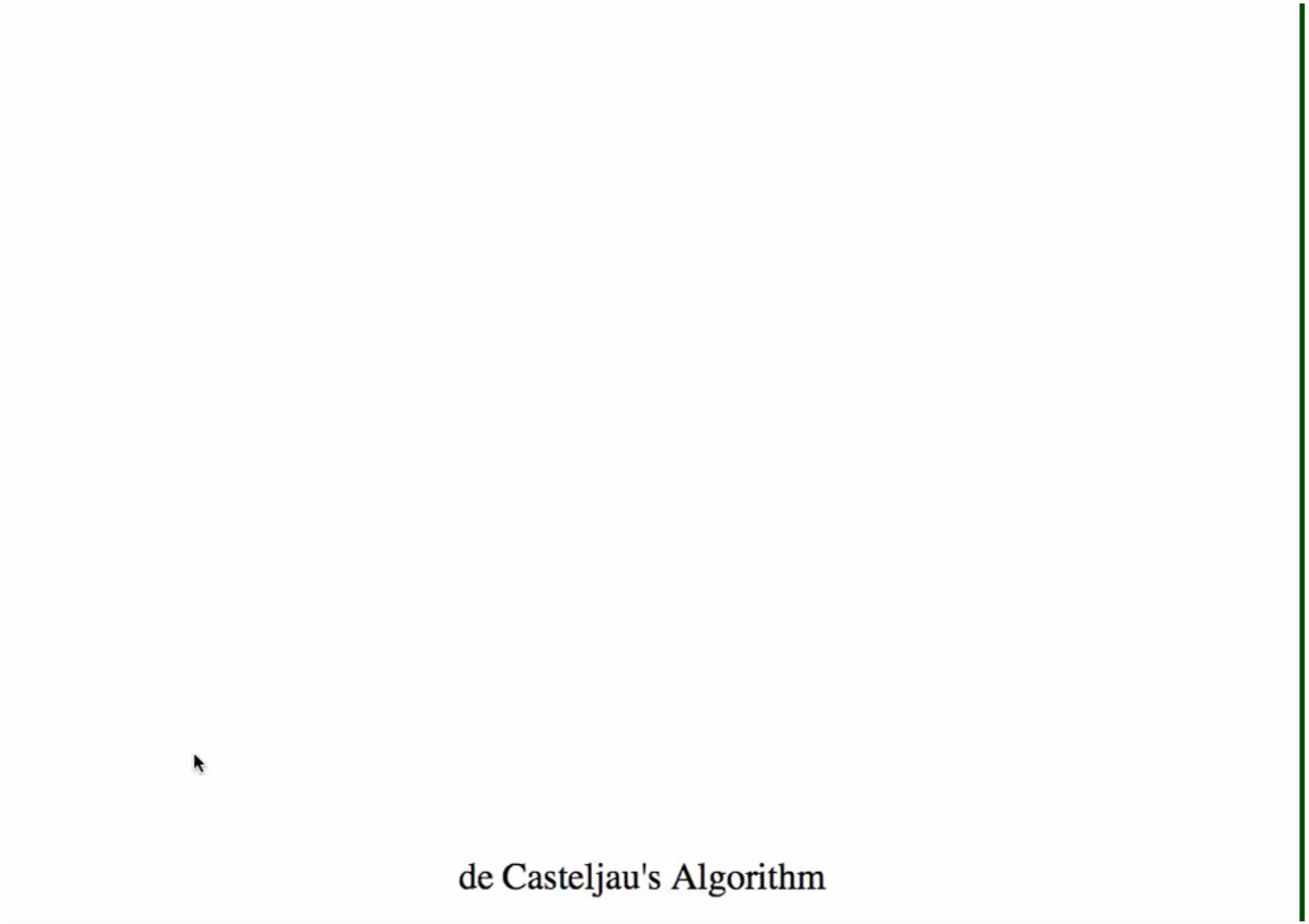
De Casteljau Construction of Bezier Curves
（德卡斯特里奥算法 ）

- How to derive a point on a Bezier curve?



计算P(1/3)的割角法

De Casteljau Construction of Bezier Curves
（德卡斯特里奥算法 ）演示



Bezier Curves (cont.)

- Variant of the Hermite spline
  - basis matrix derived from the Hermite basis (or from scratch)
- Gives more uniform control knobs (拉手) (series of points) than Hermite

Catmull-Rom Splines

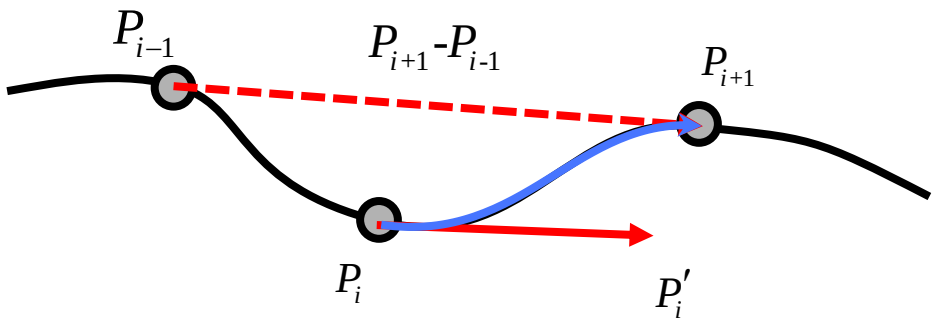
- With Hermite splines, the designer must arrange for consecutive tangents to be collinear, to get C^1 continuity. Similar for Bezier. This gets tedious.

- Catmull-Rom: an interpolating cubic spline with *built-in C^1 continuity*.

- Compared to Hermite/Bezier: fewer control points required, but less freedom.

Catmull-Rom Splines (cont.)

- Given n control points in 3-D: $p_1, p_2, \ldots, p_n$,
  - Tangent at p_i given by $s(p_{i+1} - p_{i-1})$ for i=2, ..., n-1, for some s.
  - Curve between p_i and p_{i+1} is determined by $p_{i-1}, p_i, p_{i+1}, p_{i+2}$.



$$P_i' = \frac{1}{2}(P_{i+1} - P_{i-1})$$

Catmull-Rom Splines (cont.)

- Given n control points in 3-D: $p_1, p_2, \ldots, p_n$,
  - Tangent at p_i given by $s(p_{i+1} - p_{i-1})$ for i=2, …, n-1, for some s
  - Curve between p_i and p_{i+1} is determined by $p_{i-1}, p_i, p_{i+1}, p_{i+2}$
  - What about endpoint tangents? (several good answers: extrapolate, or use extra control points p_0, p_{n+1})
  - Now we have positions and tangents at each knot – a Hermite specification.

Catmull-Rom Spline Matrix

- Derived similarly to Hermite and Bezier
- s is the tension parameter; typically s=1/2

When $s = 1/2$,

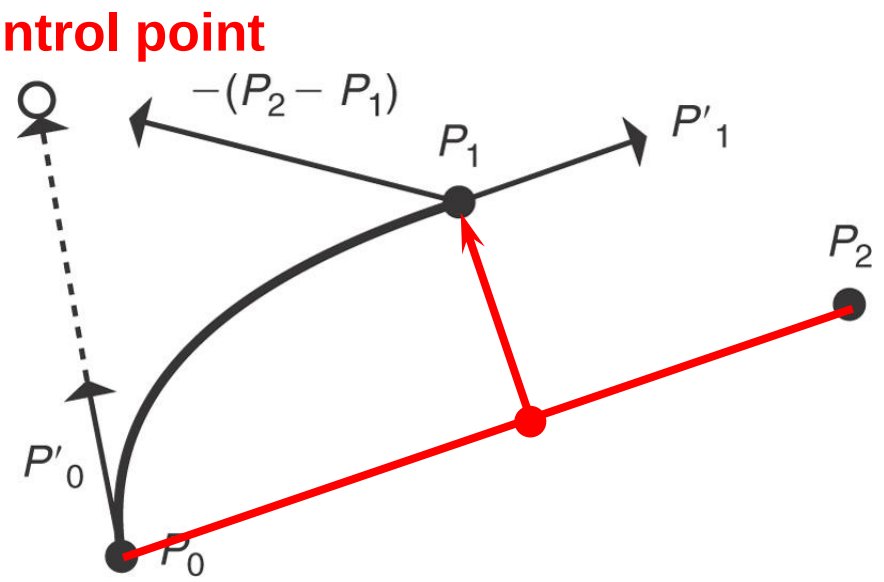
$$p(u) = U^T \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{bmatrix}$$

Catmull-Rom Spline

- What about endpoint tangents?
  - Provided by users
  - Several good answers: extrapolate, or use extra control points p_0, p_{n+1}

$$P_0' = \frac{1}{2}((P_1 - (P_2 - P_1)) - P_0)$$

$$= \frac{1}{2}(2P_1 - P_2 - P_0) = P_1 - \frac{1}{2}(P_0 + P_2)$$

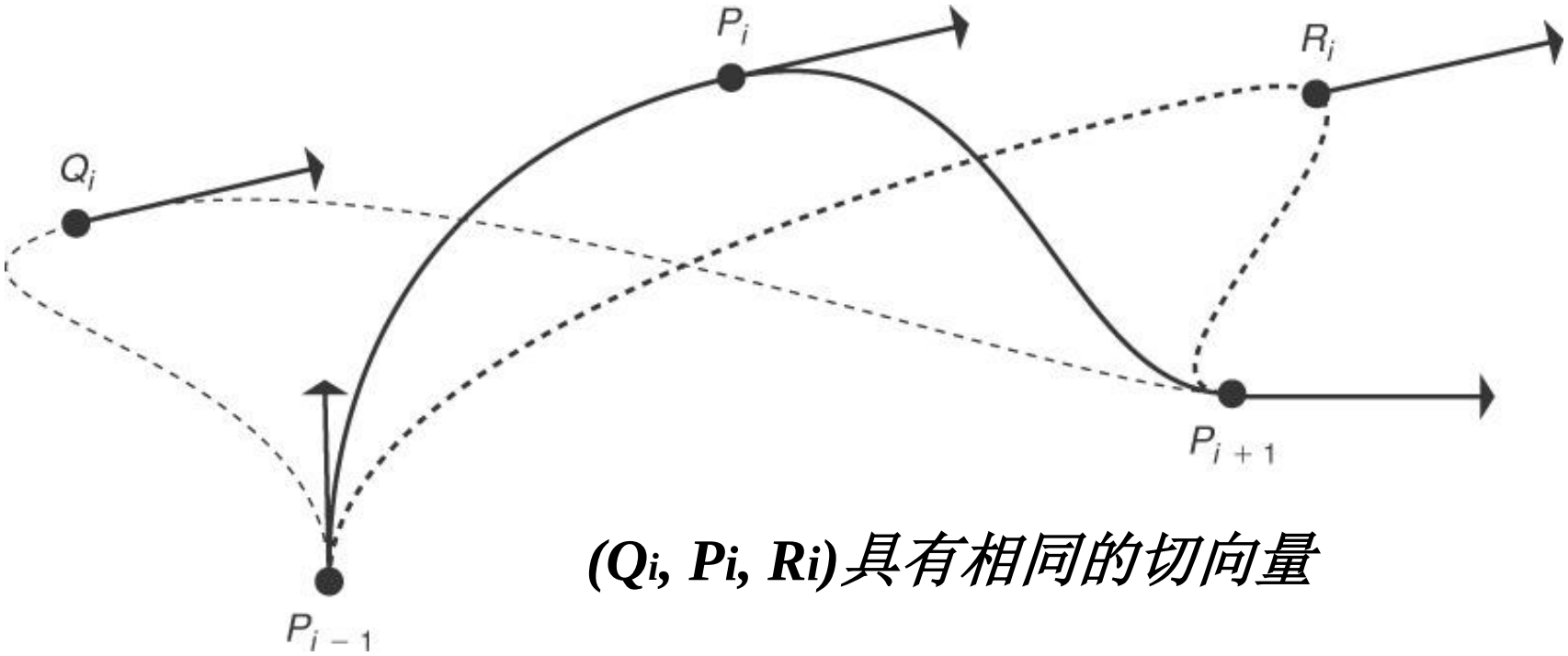


Catmull-Rom Spline

- Example

Catmull-Rom Spline

- Drawback
  - An internal tangent is not dependent on the position of the internal point relative to its two neighbors



(Q_i, P_i, R_i)具有相同的切向量

Splines and Other Interpolation Forms

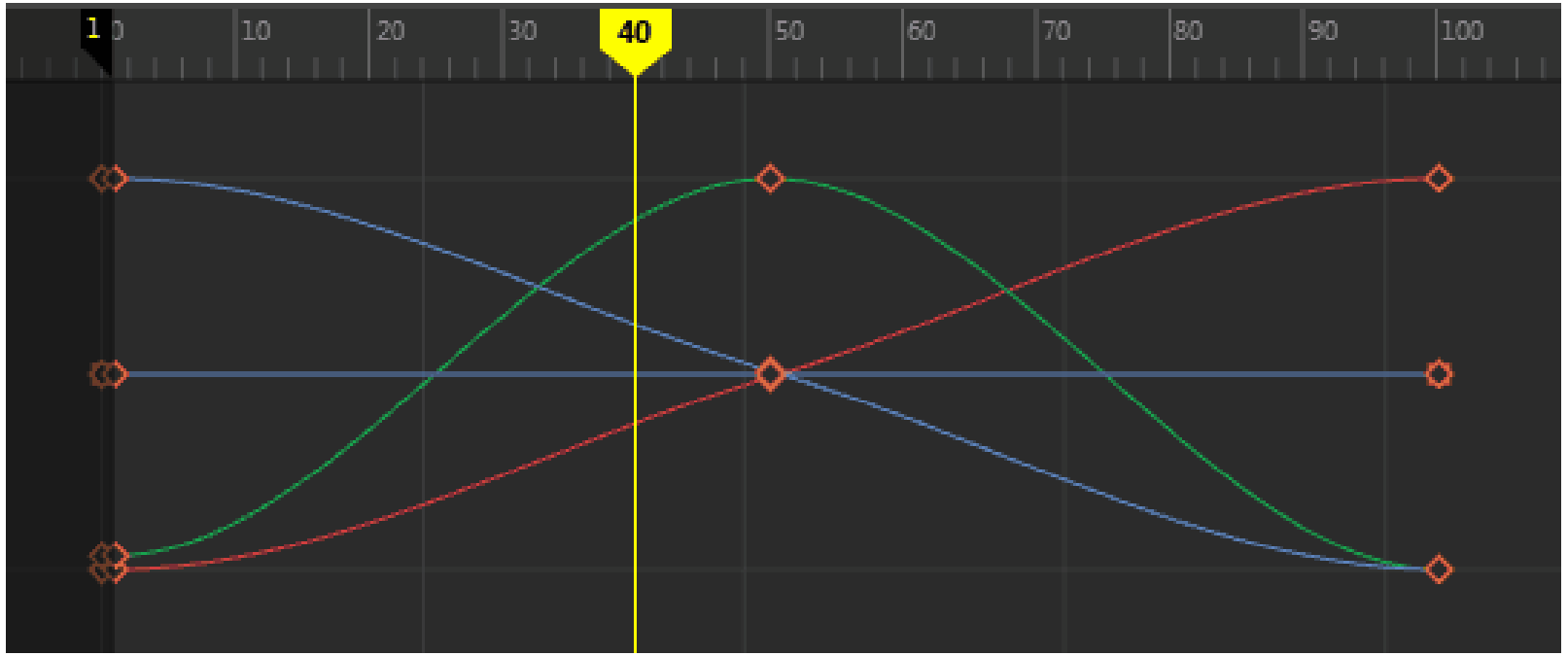
- See Computer Graphics textbooks
- Review
  - Appendix B.4 in Parent

Now What?

- We have key frames or points
- We have a way to specify the space curve
- Now we need to specify velocity to traverse the curve

Speed Curves

Speed Curve in Maya

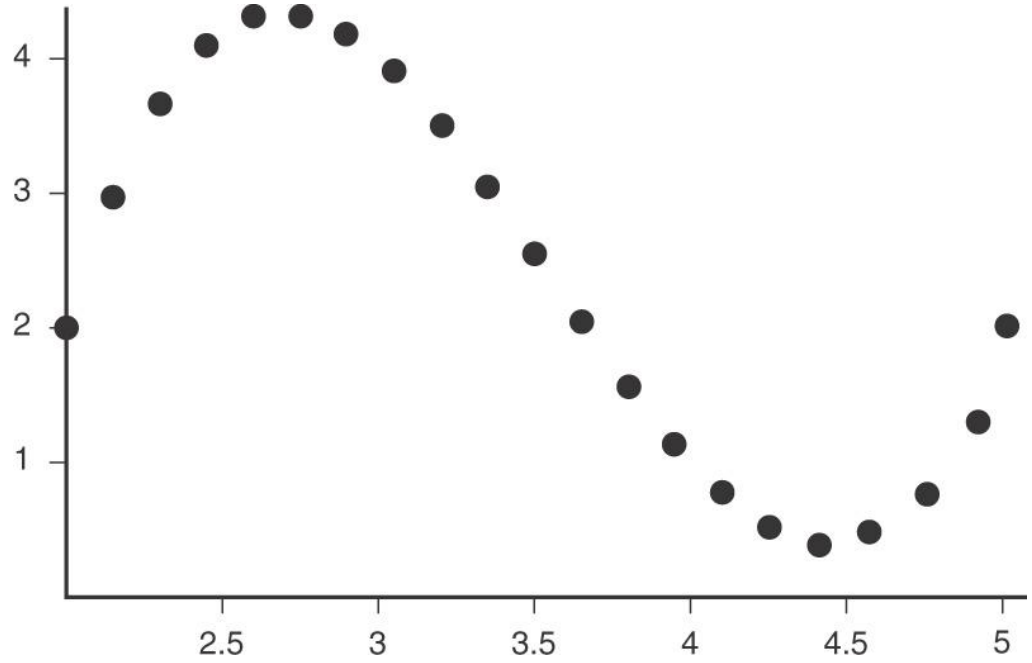


Edit the Speed Curve in the Graph Editor. The Speed Curve displays in purple.
https://help.autodesk.com/view/MAYACRE/ENU/?guid=GUID-BEE97073-A8AD-4EB6-8035-67C8F3C5492B

Speed Curve Demo

Speed Control

- The speed of tracing a curve needs to be under the direct control of the animator
- Varying u at a constant rate will not necessarily generate $P(u)$ at a constant speed.

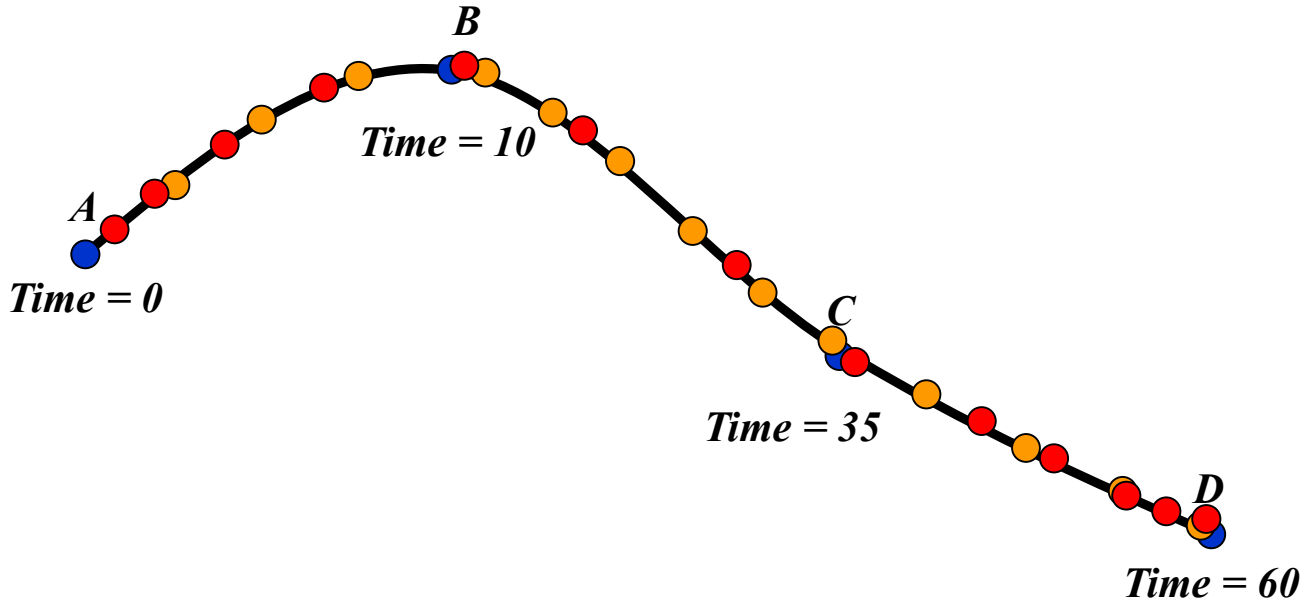


Non-uniformity in Parametrization

- Generally, equally spaced samples in parameter space are not equally spaced along the curve

$$u_2 - u_1 \neq s(u_2) - s(u_1)$$

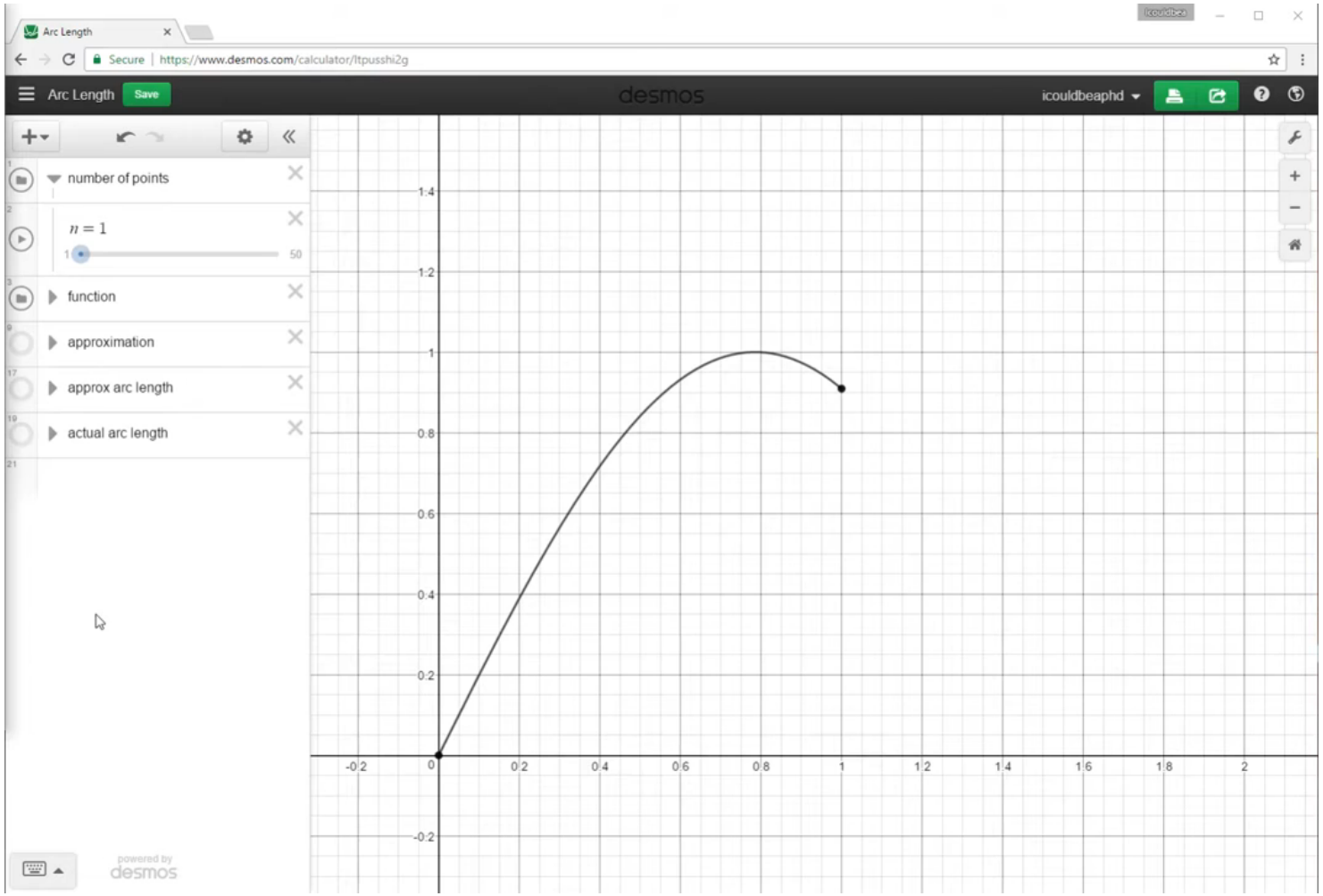
$$s(u) = \int \sqrt{P_x(u)^2 + P_y(u)^2 + P_z(u)^2}\, du$$



Arc Length Reparameterization

- To ensure a constant speed for the interpolated value, the curve has to be parameterized by arc length (for most applications)
- Computing arc length
  - Analytic method (many curves do not have, e.g., B-splines)
  - Numeric methods
    - Table and differencing
    - Gaussian quadrature (Gauss型求积公式)

Why Arc Length Reparameterization? Demo



Arc Length Reparameterization

- Space curve vs. time-distance function
  - Relates time to the distance traveled along the curve, i.e., relates time to the arc length along the curve

Arc Length Reparameterization

- Given a space curve with arc length parameterization
  - Allow movement along the curve at a constant speed by stepping at equal arc length intervals
  - Allows acceleration and deceleration along the curve by controlling the distance traveled in a given time interval
- Problems
  - Given a parametric curve and two parameter values u_1 and u_2, find $arclength(u_1,u_2)$
  - Given an arc length s, and parameter value u_1, find u_2 such that $arclength(u_1,u_2) = s$

Arc Length Reparameterization

- Converting a space curve $P(u)$ to a curve with arc-length parameterization
  - Find $s=S(u)$, and $u=S^{-1}(s)=U(s)$
  - $P^*(s)=P(U(s))$
- Analytic arc-length parameterization is difficult or impossible for most curves, e.g., B-spline curve cannot be parameterized with arc length.
  - Approximate arc-length parameterization

Forward Differencing（向前差分)

- Sample the curve at small intervals of the parameter
- Compute the distance between samples
- Build a table of arc length for the curve

u	Arc Length
0.0	0.00
0.1	0.08
0.2	0.19
0.3	0.32
0.4	0.45
…	…

Arc Length Reparameterization Using Forward Differencing

- Given a parameter *u*, find the arc length
  - Find the entry in the table closest to this *u*
  - Or take the *u* before and after it and interpolate arc length linearly

u	Arc Length
0.0	0.00
0.1	0.08
0.2	0.19
0.3	0.32
0.4	0.45
…	…

Arc Length Reparameterization Using Forward Differencing

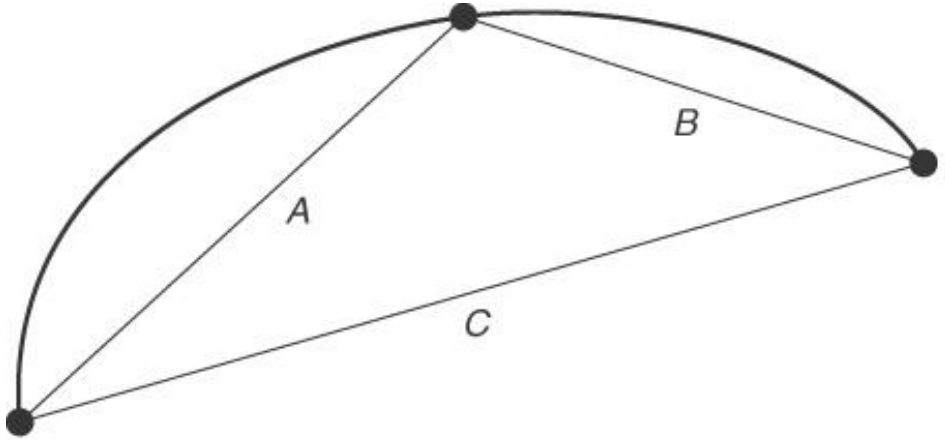
- Given an arc length s and a parameter u_1, find the parametric value u_2 such that arclength(u_1, u_2)=s
  - Find the entry in the table closest to this u using binary search
  - Or take the u before and after it and interpolate linearly

u	Arc Length
0.0	0.00
0.1	0.08
0.2	0.19
0.3	0.32
0.4	0.45
…	…

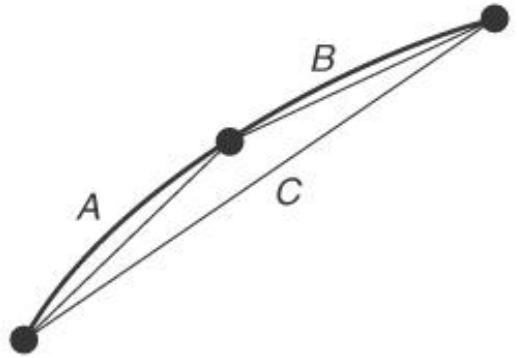
Arc Length Reparameterization Using Forward Differencing

- Easy to implement, intuitive, and fast
- Introduce errors
  - Super-sampling in forming table
    - 1000 equally spaced parameter values + 1000 entries in each interval
    - Better interpolation
    - Adaptive forward differencing

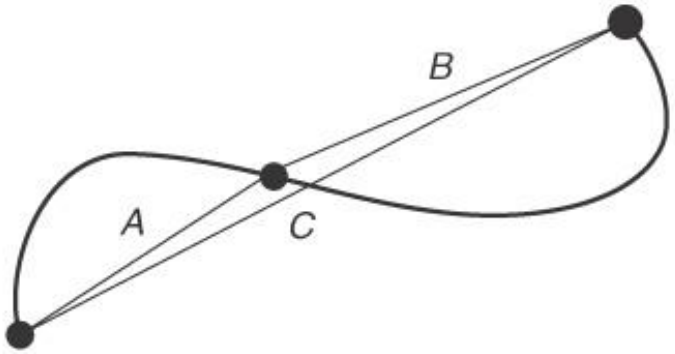
Arc Length Reparameterization Using Adaptive Forward Differencing



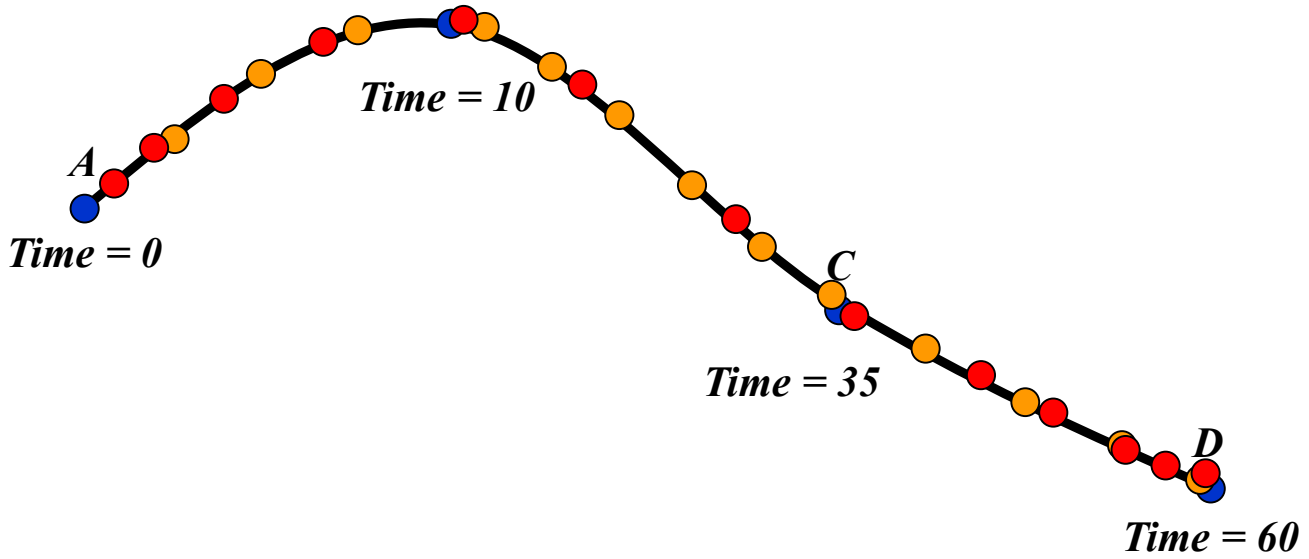
Lengths $A + B - C$ above error tolerance



Lengths $A + B - C$ within error tolerance

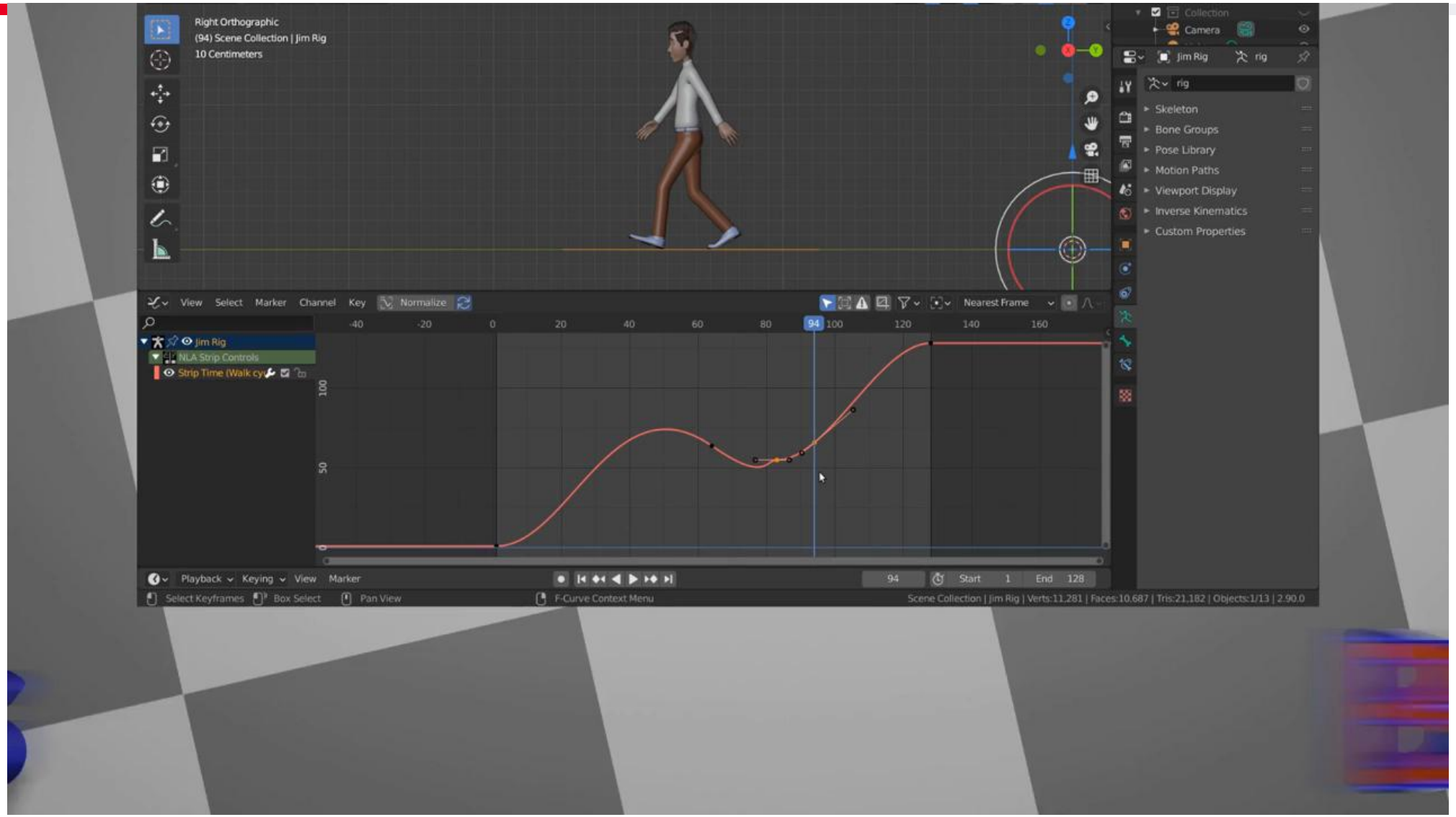


Lengths $A + B - C$ erroneously report that the error is within tolerance

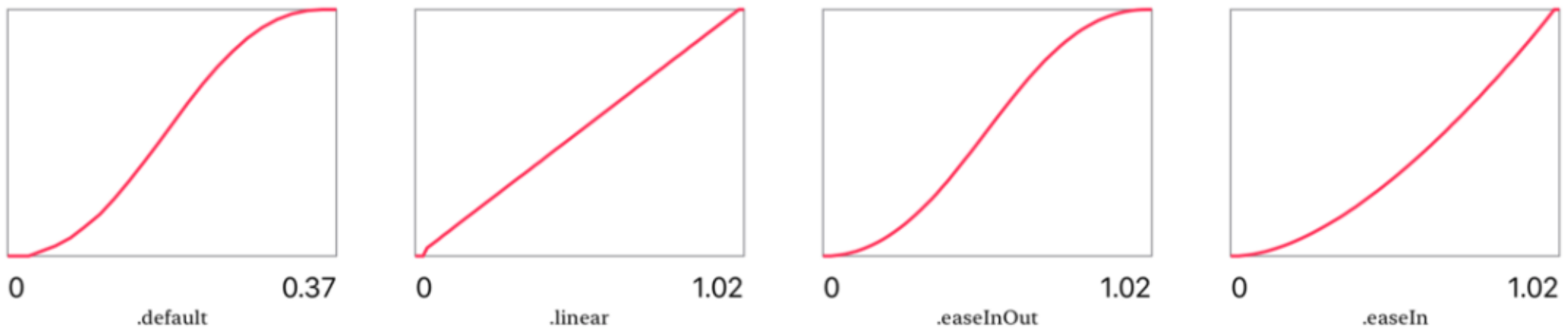


Given an arc-length parameterized space curve, now what's the next step?

速度控制曲线(以Blender为例)



速度控制曲线(Speed Control)



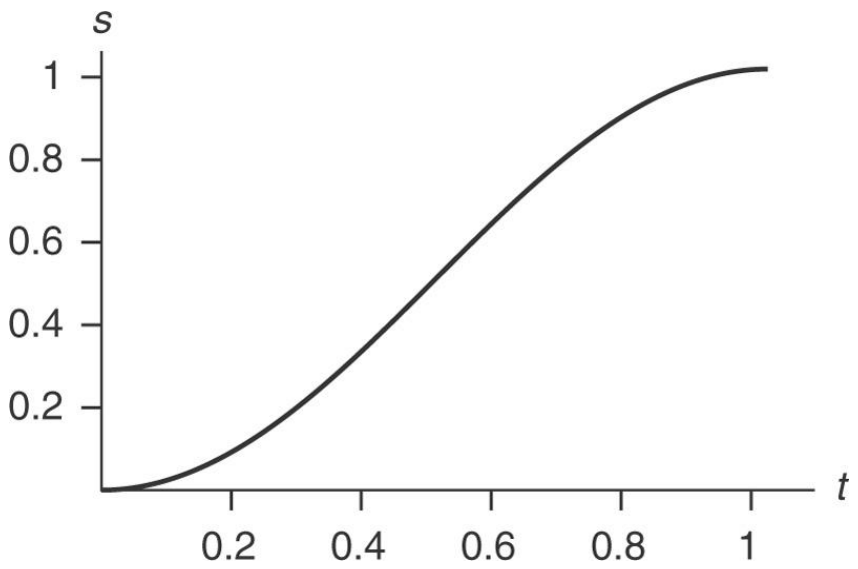
Most common speed curves

速度控制曲线(Speed Control)

- Given an arc-length parameterized space curve, how to control the speed at which the curve is traced?
  - By a **speed-control function** that relates an ***equally spaced*** *time interval* to arc length
    - Input time t, output arc length: $s = S(t)$
    - Linear function: constant speed control
    - Most common: **ease-in/ease-out**
      - Smooth motion from stopped position, accelerate, reach a max velocity, and then decelerate to a stop position

Speed Control Function

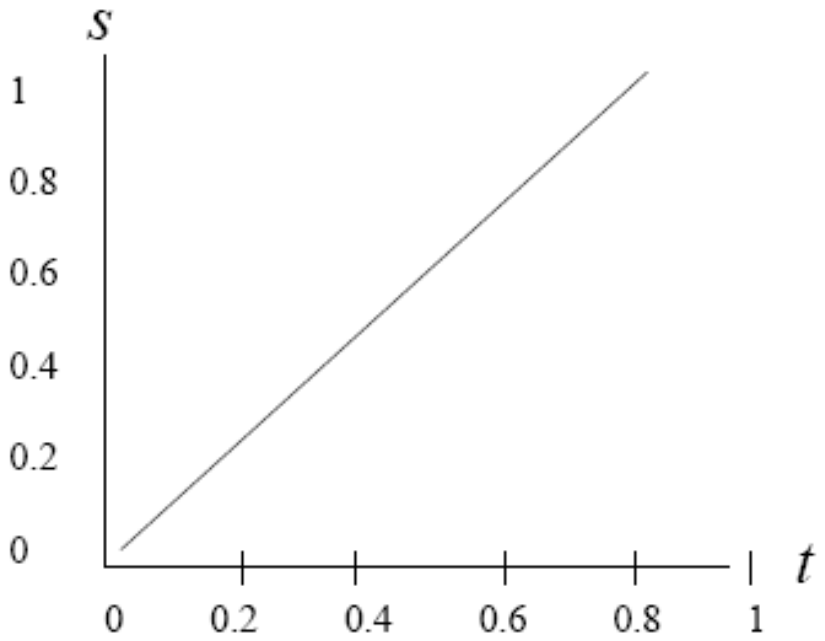
- Relates an equally spaced time interval to arc length
  - Input time t, output arc length: $s = S(t)$
  - **Normalized** arc length parameter



s=ease(t)

Start at 0, slowly increase in value and gain speed until the middle value and then decelerate as it approaches to 1.

Constant Velocity Speed Curve



- Moving at 1 m/s if meters and seconds are the units
- Too simple to be what we want

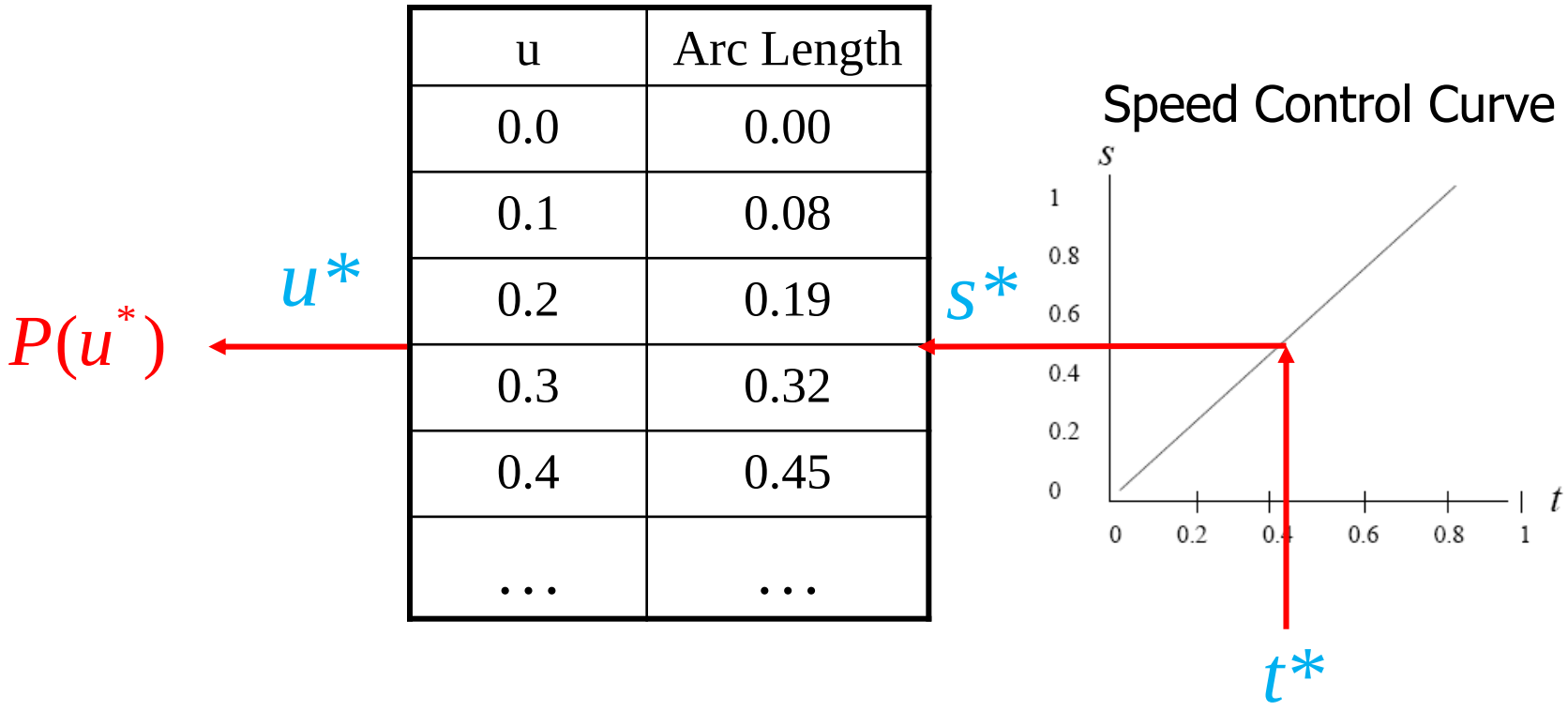
Distance Time Function and Speed Control

- We have a space curve $p=P(u)$ and a speed control function $s=S(t)$.

  For a given t, $s=S(t)$

  - Find the corresponding value $u=U(s)$ by looking up an arc length table for a given s
  - A point on the space curve with parameter u: $p=P(U(S(t)))$

Speed Control

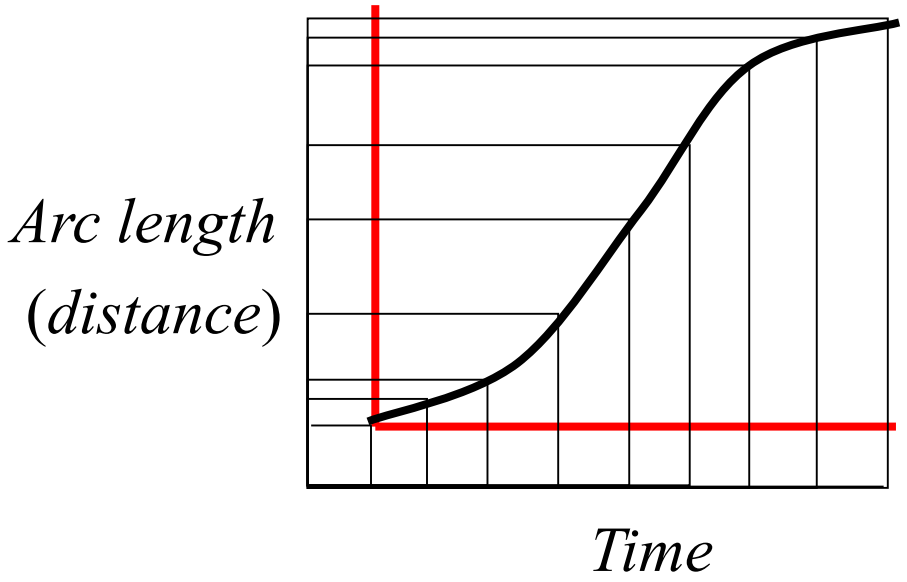


Distance Time Function

- Assumptions on distance time function
  - The entire arc length of the curve is to be traversed during the given total time
  - Additional optional assumptions
    - The function should be ***monotonic*** (单调的) in *t* （如果不是单调的，会出现倒退效果）
    - The function should be continuous

Ease-in/Ease-out

- Most useful and most common ways to control motion along a curve



Equally spaced samples in time specify arc length required for that frame

Ease-in/Ease-out

Gentleman’s Duel — Slow In and Slow Out

The End