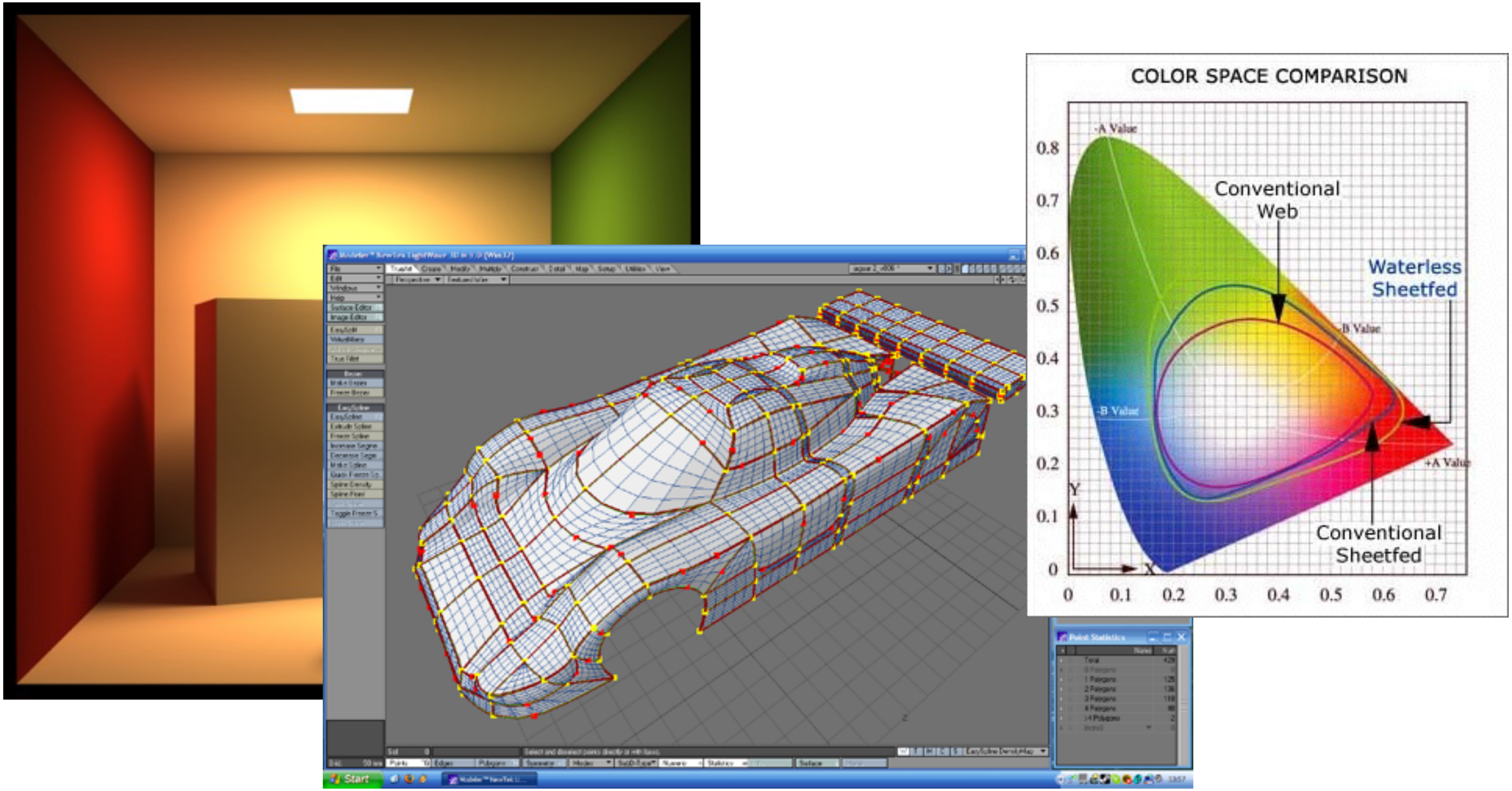


Lecture 10 : Theory of Parametric 2D curves (constraints, basis functions, cubics, Hermite curves)

Tuesday October 12th 2021

Logistics

- Grades for homework #1 have been posted.
- Grading questions: on Piazza, or at Office Hours (Among our TAs, Carter is best positioned to answer questions on grading).
- Might still need to look at/grade special cases that specifically communicated and/or obtain permission.
- Homework #3 to be released soon (ETA: released by end of Wednesday, due Friday Oct 22nd)

Logistics

- Midterm decision: In all likelihood, we'll have it as an online Canvas quiz on Oct 29th, 7:15pm (as originally announced)
- Upside:
  - Time already announced at beginning of class; students more likely to have prepared to be available at that time
  - We can devote one or two lectures of that week to exam review without setting back the pace of the class
- Downside:
  - Instructor won't be available for questions during the exam; we'll try our best to preempt any issues.

Today's lecture

- Polynomial parametric curves
  - Properties and motivation
  - Algebraic expression and implementation hints
  - Example : Linear polynomial curves (line segments)
- Cubic polynomial curves
  - Hermite curves
  - Natural cubics
  - Bezier curves, B-splines
  - Putting pieces together … (code in next lecture!)

(Recap) Polynomial curves

- Remember the differences in notation from what we saw earlier (to align with FCG Section 15)
  - Curve denoted by ***f***, parameter by ***u***
  - Curve coordinates listed as a <u>row vector</u>

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

(Piecewise) Polynomial curves

$$\mathcal{C}(t) = \begin{bmatrix} x(t) & y(t) \end{bmatrix}$$

- In earlier examples, we crafted curves using "custom" expressions for x(t) and y(t) … not an easy way to maneuver the curve around!
- Idea: use polynomial expressions to define x(t), y(t)!

$$x(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_N t^N$$
$$y(t) = b_0 + b_1 t + b_2 t^2 + \cdots + b_N t^N$$

(Piecewise) Polynomial curves

$$\mathcal{C}(t) = \begin{bmatrix} x(t) & y(t) \end{bmatrix}$$

$$x(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_N t^N$$

$$y(t) = b_0 + b_1 t + b_2 t^2 + \cdots + b_N t^N$$

- Benefits :
  - The "knobs" we can turn to maneuver the curve around are clearly defined (coefficients a_k and b_k)
  - Polynomials <u>can</u> represent the simplest curve: a line segment (it only requires linear polynomials)
  - Manipulation of these curves can be facilitated by matrix algebra (more on this later in this lecture)
  - We can adjust polynomial cubes for desired features (where to start and stop, tangents, and continuity)

(Piecewise) Polynomial curves

$$\mathcal{C}(t) = \begin{bmatrix} x(t) & y(t) \end{bmatrix}$$

$$x(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_N t^N$$

$$y(t) = b_0 + b_1 t + b_2 t^2 + \cdots + b_N t^N$$

Matrix representation

$$\mathcal{C}(t) = \begin{bmatrix} x(t) & y(t) \end{bmatrix} = \begin{bmatrix} 1 & t & t^2 \cdots & t^N \end{bmatrix} \begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ \vdots & \vdots \\ a_N & b_N \end{bmatrix}$$

(Piecewise) Polynomial curves

$$\mathcal{C}(t) = \begin{bmatrix} x(t) & y(t) \end{bmatrix}$$

$$x(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_N t^N$$

$$y(t) = b_0 + b_1 t + b_2 t^2 + \cdots + b_N t^N$$

Matrix representation
(switching to textbook notation)

$$\mathbf{f}(u) = \begin{bmatrix} x(u) & y(u) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & u & u^2 \cdots & u^N \end{bmatrix}}_{\mathbf{u}} \underbrace{\begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ \vdots & \vdots \\ a_N & b_N \end{bmatrix}}_{\mathbf{A}}$$

(Piecewise) Polynomial curves

$$\mathbf{f}(u) = \begin{bmatrix} x(u) & y(u) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & u & u^2 \cdots & u^N \end{bmatrix}}_{\mathbf{u}} \underbrace{\begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ \vdots & \vdots \\ a_N & b_N \end{bmatrix}}_{\mathbf{A}}$$

Tangents:

$$\mathbf{f}'(u) = \underbrace{\begin{bmatrix} 0 & 1 & 2u \cdots & Nu^{N-1} \end{bmatrix}}_{\mathbf{u}'} \underbrace{\begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ \vdots & \vdots \\ a_N & b_N \end{bmatrix}}_{\mathbf{A}}$$

(Piecewise) Polynomial curves

$$\mathbf{f}(u) = \begin{bmatrix} x(u) & y(u) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & u & u^2 \cdots & u^N \end{bmatrix}}_{\mathbf{u}} \underbrace{\begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ \vdots & \vdots \\ a_N & b_N \end{bmatrix}}_{\mathbf{A}}$$

Tangents:

$$\mathbf{f}'(u) = \underbrace{\begin{bmatrix} 0 & 1 & 2u \cdots & Nu^{N-1} \end{bmatrix}}_{\mathbf{u}'} \underbrace{\begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ \vdots & \vdots \\ a_N & b_N \end{bmatrix}}_{\mathbf{A}}$$

- We constructed a parametric curve ***f(u)=[x(u),y(u)]*** by allowing ***x(u)***, ***y(u)*** to take the form of polynomials
- In this case, specifying what the curve actually is reduces to specifying the polynomial coefficients a_k, b_k.

Controlling polynomial curves

$$\mathbf{f}(u) = \begin{bmatrix} x(u) & y(u) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & u & u^2 \cdots & u^N \end{bmatrix}}_{\mathbf{u}} \underbrace{\begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ \vdots & \vdots \\ a_N & b_N \end{bmatrix}}_{\mathbf{A}}$$

- General methodology for determining the polynomial coefficients a_k ,b_k for a given curve
  - Figure out what polynomial degree is needed and determine an appropriate parameter interval
  - Translate specifications for the curve (where should it go through, and/or with what tangent, at given parameter values) into algebraic equations
  - Solve these equations for the coefficients (the matrix A)

Controlling polynomial curves

$$\mathbf{f}(u) = \begin{bmatrix} x(u) & y(u) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & u & u^2 \cdots & u^N \end{bmatrix}}_{\mathbf{u}} \underbrace{\begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ \vdots & \vdots \\ a_N & b_N \end{bmatrix}}_{\mathbf{A}}$$

- Assumptions (some simplifying …)
  - Convenient to consider the parametric interval [0,1] (we will see how you reconcile this with piecewise-polynomials that span different parameter intervals)
  - Specifications that dictate the curve should traverse a specific location take the form $\boldsymbol{f}(u^*) = u^*\boldsymbol{A} = \boldsymbol{p}^*$
  - Specifications that dictate the curve should have a specific tangent take the form $\boldsymbol{f}'(u^*) = (u^*)'\boldsymbol{A} = \boldsymbol{d}^*$

Example : Line Segment

$$\mathbf{f}(u) = \underbrace{\left[\begin{array}{cc} 1 & u \end{array}\right]}_{\mathbf{u}} \underbrace{\left[\begin{array}{cc} a_0 & b_0 \\ a_1 & b_1 \end{array}\right]}_{\mathbf{A}}$$

$$\mathbf{p}_1 = \left[\begin{array}{cc} x_1 & y_1 \end{array}\right]$$

$$\mathbf{p}_0 = \left[\begin{array}{cc} x_0 & y_0 \end{array}\right]$$

- A line segment can be captured by a *linear* polynomial!
  - Why? Consider the derivative ***f’****(u)* !!
  - Why *just* linear? Matching counts of input constraints (point coordinates) and parameters (entries in **A**)

Example : Line Segment

$$\mathbf{p}_1 = \begin{bmatrix} x_1 & y_1 \end{bmatrix}$$

$$\mathbf{p}_0 = \begin{bmatrix} x_0 & y_0 \end{bmatrix}$$

$$\mathbf{f}(u) = \underbrace{\begin{bmatrix} 1 & u \end{bmatrix}}_{\mathbf{u}} \underbrace{\begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \end{bmatrix}}_{\mathbf{A}}$$

$$\mathbf{f}(0) = \mathbf{p}_0 \Rightarrow \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{A} = \mathbf{p}_0$$

$$\mathbf{f}(1) = \mathbf{p}_1 \Rightarrow \begin{bmatrix} 1 & 1 \end{bmatrix} \mathbf{A} = \mathbf{p}_1$$

Alebraic form of pass-through specifications

$$\underbrace{\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}}_{\mathbf{C}} \mathbf{A} = \underbrace{\begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \end{bmatrix}}_{\mathbf{P}}$$

C is the *constraint* matrix

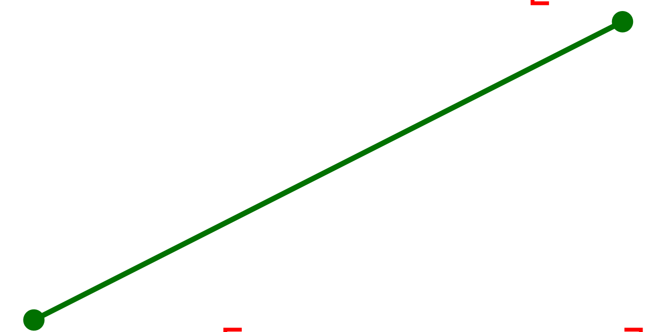
$$\mathbf{A} = \mathbf{C}^{-1}\mathbf{P} = \underbrace{\begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}}_{\mathbf{B}} \mathbf{P} \qquad \mathbf{f}(u) = \mathbf{uBP}$$

B for “basis”

Example : Line Segment

$$\mathbf{f}(u) = \underbrace{\begin{bmatrix} 1 & u \end{bmatrix}}_{\mathbf{u}} \underbrace{\begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \end{bmatrix}}_{\mathbf{A}}$$

$$\mathbf{p}_1 = \begin{bmatrix} x_1 & y_1 \end{bmatrix}$$



$$\mathbf{p}_0 = \begin{bmatrix} x_0 & y_0 \end{bmatrix}$$

$$\mathbf{f}(u) = \mathbf{p}_0 + u(\mathbf{p}_1 - \mathbf{p}_0)$$

(after carrying out operations …
let's do on whiteboard!)

A (big) step up: Cubic curves!

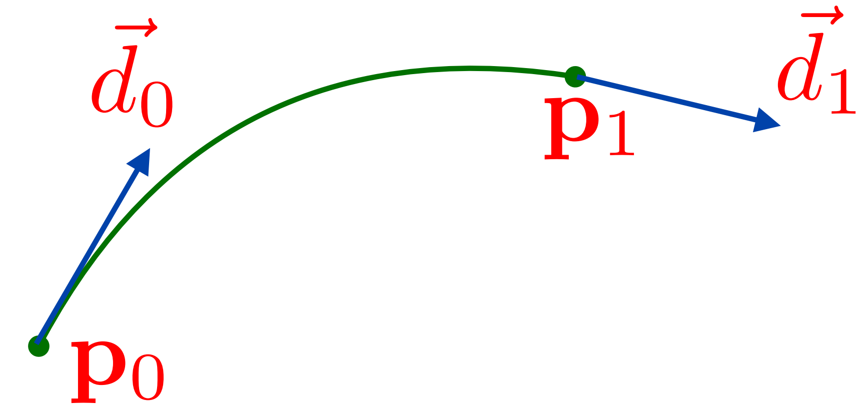
$$\mathbf{f}(u) = \underbrace{\begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix}}_{\mathbf{u}} \underbrace{\begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}}_{\mathbf{A}}$$

- Useful properties
  - Can be used to control both location of endpoints, as well as the direction of the tangent
  - Can be easily manipulated to give C1-continuity (maybe C2, under certain circumstances and restrictions)
  - Has enough degrees of freedom to allow *local control* in conjunction with C1 continuity

Properties of cubics: Pick 3!!!

- For polynomial curves, *three* of the following properties can be satisfied simultaneously (not all 4!)
  - C2 continuity of the curve — Counter-example: Hermite
  - Interpolation of all "control points" — Counter-example: B-splines
  - Local control of curve — Counter-example: Natural cubics
  - The polynomials have order no more than 3

Hermite cubics

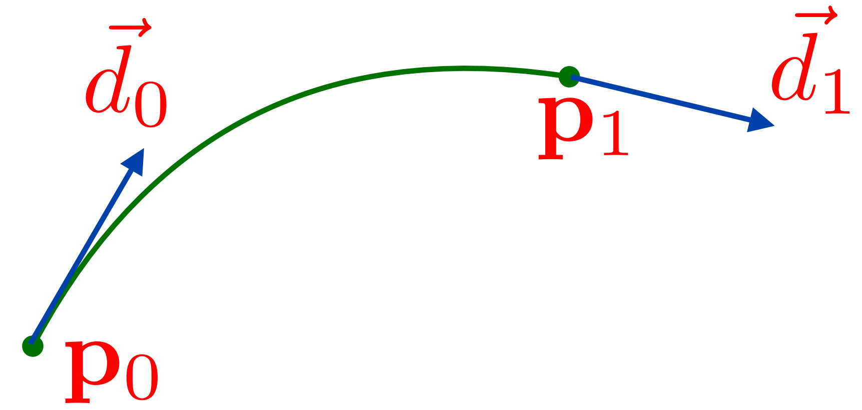


- We specify
  - Beginning and ending *positions* $\boldsymbol{p_0}$, $\boldsymbol{p_1}$
  - Beginning and ending *tangents* $\boldsymbol{d_0}$, $\boldsymbol{d_1}$
- As before the curve is written (using the *basis matrix*)

$$\mathbf{f}(u) = \mathbf{uBP}$$

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3 & -2 & 3 & -1 \\ 2 & 1 & -2 & 1 \end{bmatrix} \qquad \mathbf{P} = \begin{bmatrix} p_0 \\ p_1 \\ d_0 \\ d_1 \end{bmatrix}$$

Hermite cubics



- (derivation on whiteboard)
- Nice properties:
  - Easy to enforce C1 continuity (do you see how?)
  - Certainly amenable to local control

Basis functions?

- Standard parametric form : $\mathbf{f}(u) = \mathbf{uBP}$
- We can multiply u and B to get a *vector of basis functions* $\mathbf{b}(u) = \mathbf{uB} = \begin{bmatrix} b_0(u) & b_1(u) & b_2(u) & b_3(u) \end{bmatrix}$
- The curve is written as a linear combination of the *control points:*

$$\mathbf{f}(u) = \sum_{k=0}^{3} b_k(u)\mathbf{p}_k$$

- *Similarly, for the derivative:*

$$\mathbf{f}'(u) = \sum_{k=0}^{3} b'_k(u)\mathbf{p}_k$$

The basis functions are the ones you will implement in practice (in code!)

Basis functions?

- For Hermite :

$$\mathbf{b}(u) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3 & -2 & 3 & -1 \\ 2 & 1 & -2 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} b_0(u) & b_1(u) & b_2(u) & b_3(u) \end{bmatrix}$$

- Basis functions :

$$\begin{aligned} b_0(u) &= 2u^3 - 3u^2 + 1 \\ b_1(u) &= u^3 - 2u^2 + u \\ b_2(u) &= -2u^3 + 3u^2 \\ b_3(u) &= u^3 - u^2 \end{aligned}$$

- Curve formula :

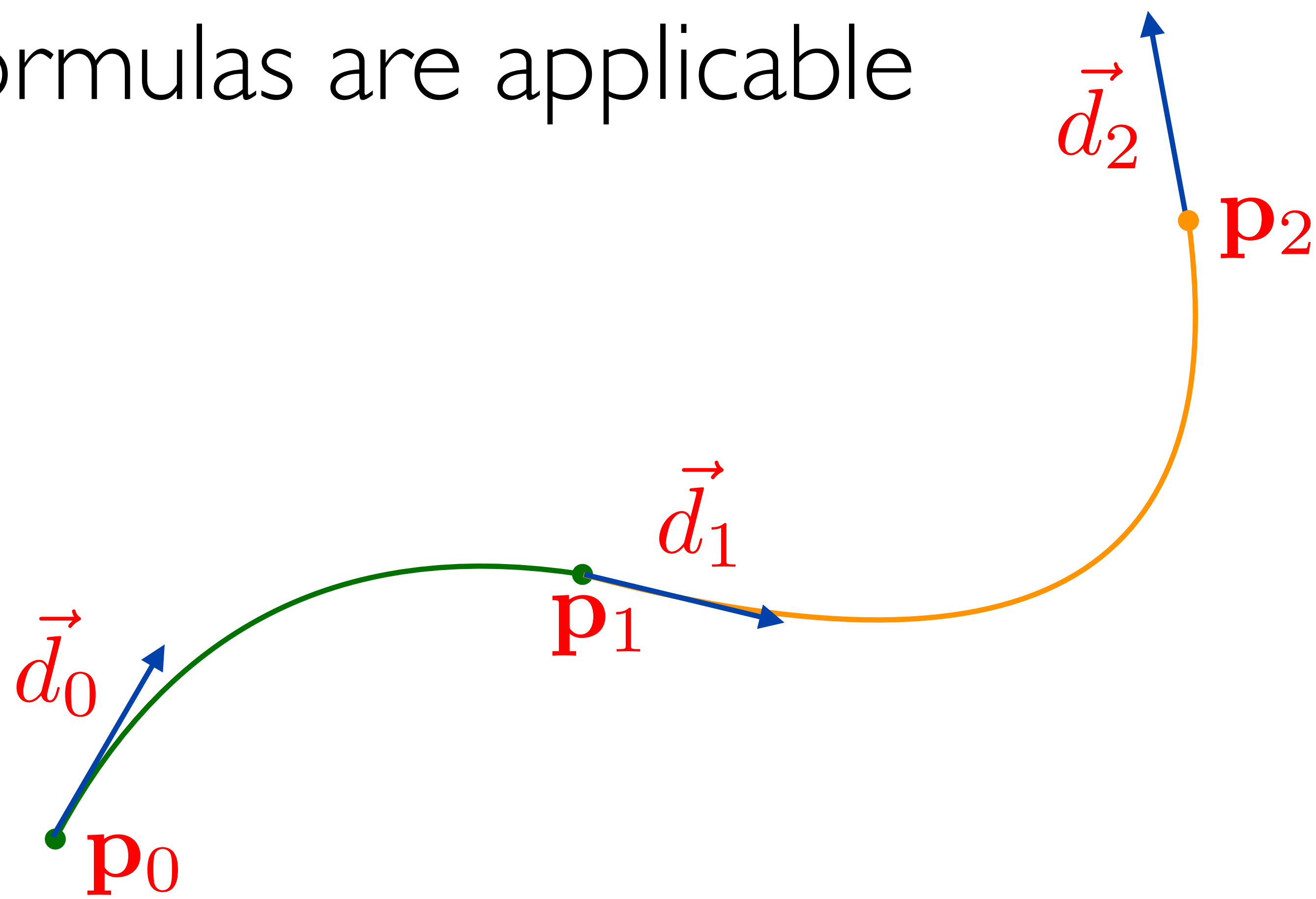
$$\mathbf{f}(u) = \sum_{k=0}^{3} b_k(u)\mathbf{p}_k$$

Hermite - Sample Implementation

jsbin.com/fawudah

Week6/Demo0

- Two Hermite curves, joined with C1 continuity
- Defined via 3 pairs of (location, tangent) control points, the middle one shared by the two curves
- The parametric interval for each piece of the curve is "translated" to the canonical interval [0,1], so that the previous formulas are applicable



Derivatives?

jsbin.com/bodorun

Week6/Demo1

- Still using Hermite basis matrix :

$$\mathbf{b}'(u) = \begin{bmatrix} 0 & 1 & 2u & 3u^2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3 & -2 & 3 & -1 \\ 2 & 1 & -2 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} b_0'(u) & b_1'(u) & b_2'(u) & b_3'(u) \end{bmatrix}$$

- Derivatives of basis functions :

$$\begin{aligned} b_0'(u) &= 6u^2 - 6u \\ b_1'(u) &= 3u^2 - 4u + 1 \\ b_2'(u) &= -6u^2 + 6u \\ b_3'(u) &= 3u^2 - 2u \end{aligned}$$

- Tangent formula :

$$\mathbf{f}'(u) = \sum_{k=0}^{3} b_k'(u)\mathbf{p}_k$$