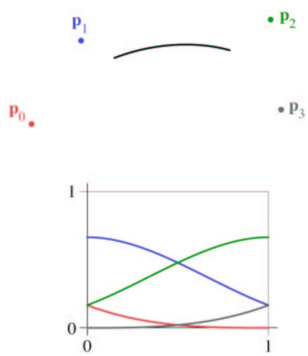
B-splines

CS 417 Lecture 13 (part 1)

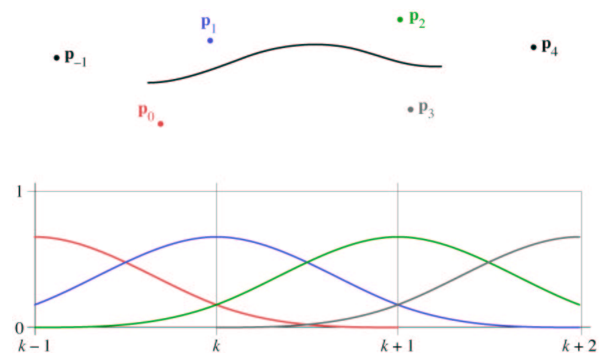
B-splines

- We may want more continuity than C^1
- We may not need an interpolating spline
- B-splines are a clean, flexible way of making long splines with arbitrary order of continuity
- Various ways to think of construction
  - a simple one is convolution
  - relationship to sampling and reconstruction

Cubic B-spline basis



Cubic B-spline basis



Construction of B-splines

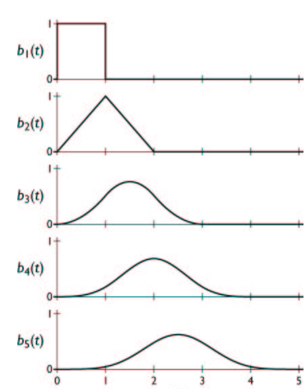
- B-splines defined for all orders
  - order *d*: degree *d* – 1
  - order *d*: *d* points contribute to value
- One definition: Cox-deBoor recurrence

$$b_1 = \begin{cases} 1 & 0 \le u < 1 \\ 0 & \text{otherwise} \end{cases}$$

$$b_d = \frac{t}{d-1} b_{d-1}(t) + \frac{d-t}{d-1} b_{d-1}(t-1)$$

B-spline construction

- Recurrence
  - ramp up/down
- Convolution
  - smoothing of basis fn
  - smoothing of curve



Cubic B-spline matrix

$$\mathbf{p}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_{k-1} \\ \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{p}_{k+2} \end{bmatrix}$$

Other types of B-splines

- Nonuniform B-splines
  - discontinuities not evenly spaced
  - allows control over continuity or interpolation at certain points
  - e.g. interpolate endpoints (commonly used case)
- Nonuniform Rational B-splines (NURBS)
  - ratios of nonuniform B-splines: $x(t)$ / $w(t)$; $y(t)$ / $w(t)$
  - key properties:
    - invariance under perspective as well as affine
    - ability to represent conic sections exactly

Converting spline representations

- All the splines we have seen so far are equivalent
  - all represented by geometry matrices

$$\mathbf{p}_S(t) = T(t) M_S P_S$$

    - where S represents the type of spline
  - therefore the control points may be transformed from one type to another using matrix multiplication

$$P_1 = M_1^{-1} M_2 P_2$$

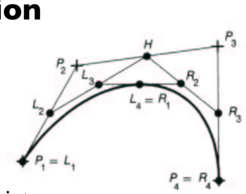
$$\begin{aligned} \mathbf{p}_1(t) &= T(t) M_1 (M_1^{-1} M_2 P_2) \\ &= T(t) M_2 P_2 = \mathbf{p}_2(t) \end{aligned}$$

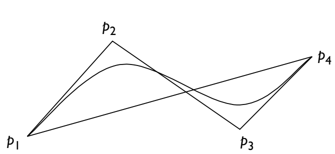
Evaluating splines for display

- Need to generate a list of line segments to draw
  - generate efficiently
  - use as few as possible
  - guarantee approximation accuracy
- Approaches
  - reccursive subdivision (easy to do adaptively)
  - uniform sampling (easy to do efficiently)

Evaluating by subdivision

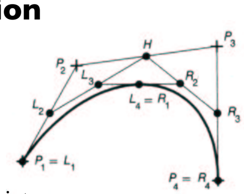
- Recursively split spline
  - stop when polygon is within epsilon of curve
- Termination criteria
  - distance between control points
  - distance of control points from line

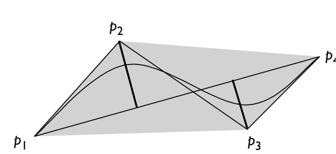




Evaluating by subdivision

- Recursively split spline
  - stop when polygon is within epsilon of curve
- Termination criteria
  - distance between control points
  - distance of control points from line





Evaluating with uniform spacing

- Forward differencing
  - efficiently generate points for uniformly spaced t values
  - evaluate polynomials using repeated differences

Intro to 3D

CS 417 Lecture 13 (part 2)

Moving from 2D to 3D

- So far, everything has been 2D
  - ultimately, that is always what goes on the screen…
  - but the most common source of 2D primitives to draw is projections of 3D scenes
- From here on everything is 3D
  - many things generalize easily
  - some things become more complex
  - some completely new issues arise

2D to 3D: geometry

- 2D: represent curves and areas
  - implicit: $f(x,y) = 0$
  - explicit: $f(t)$
- 3D: represent curves, surfaces, and volumes
  - implicit surface: $f(x, y, z) = 0$
    - implicit curve as intersection of surfaces
  - explicit curve: $\mathbf{f}(t)$
    - maps real line into 3-space
  - explicit surface: $\mathbf{f}(t, s)$
    - maps real plane into 3-space

2D to 3D: affine transformations

- Still linear plus translation
  - but rotations are considerably more complex
- Still use homogeneous coordinatess (now 4x4)
- Coordinate frame ideas still hold
  - in fact, more important because 3D is more confusing
- Still use hierarchies
  - not really any different than 2D

2D to 3D: viewing

- 2D: viewing is just selecting a rectangle
- 3D: objects are transformed into the image plane using a virtual camera
  - perspective projection: implement using homogeneous coordinate
- Hidden surfaces: an entirely new issue
  - need to work out what is occluded by what
  - for alpha blending, need to draw in the right order

2D to 3D: shading

- In 2D we draw the real shape
- In 3D we draw a projection of the real shape
  - so the eye needs additional cues

2D to 3D: rasterization

- Basic task does not change
  - we have a 2D primitive (from projected 3D primitive)
  - need to generate fragments and interpolate values
- Interpolated quantities become 3D
  - normals
  - shading information
- Perspective introduces subtleties

2D to 3D: compositing

- Same operators useful
- New operator: depth compositing
  - a very effective means to handle occlusion