

splines

- smooth functions can be used for animation
 - evolution of a robot joint angle over time
 - the path of something moving through space in time
 - the path of some rotational degree of freedom over time
- smooth curves can be used to describe smooth curves on the plane or in space
- the mathematics of smooth curves can be used as building blocks to describe smooth surfaces in space

big picture

- want to interpolate sequence of points
- want smooth function
- want locality
- could use high degree interpolatory polynomials, but this is not local, and has horrible overshooting problems
- use lots of stitched together low order polynomials pieces
- use bezier representation to describe each piece.

cubic bezier

- input 4 "control points" $p_i^0 \equiv p_i$
- visualize by placing points above $t = [0, 1/3, 2/3, 1]$ and applying same algorithm to t.
- here is an algorithm to compute the bezier interpolant at some point $t \in [0..1]$
- compute new control points as follows

$$\begin{aligned} p_i^1 &= (1-t)p_i^0 + tp_{i+1}^0 \\ p_i^2 &= (1-t)p_i^1 + tp_{i+1}^1 \\ c(t) = p_0^3 &= (1-t)p_0^2 + tp_1^2 \end{aligned}$$

- what are $c(0)$ and $c(1)$
- can be generalized to arbitrary degree.

bezier bernstein basis functions

- what is the mathematical form of this curve
- unroll the algorithm
- we get

$$c(t) = \sum_{i=0}^3 p_i \binom{3}{i} t^i (1-t)^{3-i} = \sum_{i=0}^3 p_i B_i(t)$$

$$\binom{3}{i} = \frac{3!}{i!(3-i)!}$$

- the B are the bezier bernstein basis functions

derivatives of bezier

$$c(t) = \sum_{i=0}^3 p_i \binom{3}{i} t^i (1-t)^{3-i}$$

- $c(t)$ is a polynomial of degree 3

$$d/dt \circ c(t) = \sum_{i=0}^3 p_i \binom{3}{i} [it^{i-1}(1-t)^{3-i} - (3-i)t^i(1-t)^{2-i}]$$

- so
- $c'(0) = 3(p_1 - p_0)$ and $c'(1) = 3(p_3 - p_2)$
- so slope of function matches slope of visualized control polygon

properties of basis functions

- the B form a partition of unity

$$\sum_i B_i(t) = 1 \quad \forall t$$

- apply the algorithm with all controls set to 1
- as a result $c(t)$ for all t is an “affine” combination of the p_i
 - if i add add a constant to all of the control points, i just add the constant to the whole function
- for t in $[0..1]$, $0 \leq B_i(t) \leq 1$
 - as a result, $c(t)$ is an “convex” combination of the p_i

aside: from functions to curves

- now the control points are not scalar values but x, y pairs.
- run the algorithm on the x and y separately
- think of $c(t)$ as two (or three) functions x and y (and z) of t .
- we can think of t as time, as the curve $c(t)$ traces through space
- note, that we can now visualize the algorithm in R^2 , and not use t in the visualization.

aside: properties of bezier curves

- $c'(t)$ is now a two vector
- partition of unity implies that all affine transforms “commute” with the algorithm
- convexity implies that $c(t)$ lies in the “convex hull” of the control points

stitching beziers together

- to build up a more complicated function, we stitch together different beziers
- suppose i have two beziers defined by control points l and r
- to ensure C^0 continuity, we need $l_3 = r_0$
- to ensure C^1 continuity, we need $3(l_3 - l_2) = 3(r_1 - r_0)$

catmull-rom splines

- it is inconvenient to specify non interpolated points

- given a sequence of values to interpolate $c(-1)..c(n + 1)$
- we want an interpolating function $c(t)$ for t in $[0..n]$
- use n bezier segments
- for $c(t)$ for t in $[i..i + 1]$ use one bezier curve, defined by 4 control points $p_0..p_3$
- interpolation requires us to have

$$\begin{aligned} p_0 &= c(i) \\ p_3 &= c(i + 1) \end{aligned}$$

catmull-rom derivatives

- automatically chose reasonable derivatives:

$$\begin{aligned} c'(i) &= 1/2(c(i + 1) - c(i - 1)) \\ c'(i + 1) &= 1/2(c(i + 2) - c(i)) \end{aligned}$$

so

$$\begin{aligned} (p_1 - p_0) &= 1/6(c(i + 1) - c(i - 1)) \\ (p_3 - p_2) &= 1/6(c(i + 2) - c(i)) \end{aligned}$$

quaternions splines

- want to run same algorithms for rotations $q(t)$.
- will use catmull rom to create bezier controls
 - replace $a - b$ with ab^{-1}
 - replace $1/6 * a$ with $a^{1/6}$
- will run bezier evaluation algorithm
 - replace $(1 - \alpha) * a + \alpha * b$ with $(ba^{-1})^\alpha a$