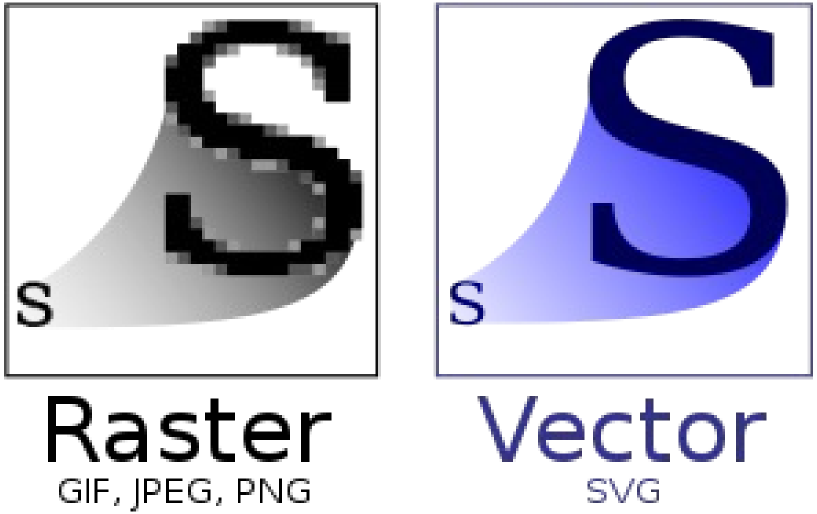
Bezier Curves

Franklin Liu

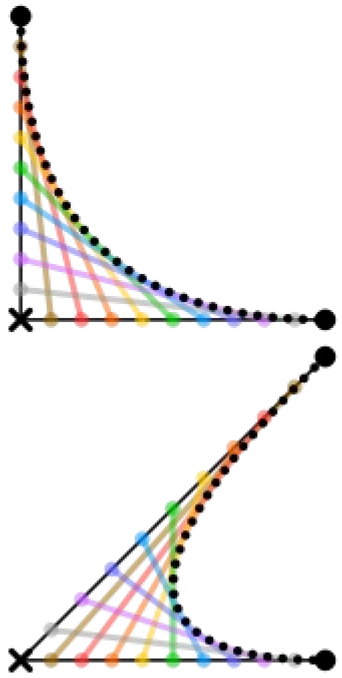
Introduction



- Raster images
  - grid of pixels
- Vector images
  - Geometric shapes, curves

Bezier curves

- Discovered independently by Paul de Casteljau and Pierre Bezier
- Uses control points to determine its shape
- Can also be represented as polynomials



De Casteljau's algorithm

- Simplest case: 3-point curve (quadratic)
- From javascript.info:
  - Draw control points.
  - Build segments between control points 1 → 2 → 3.
  - The parameter t moves from 0 to 1.
  - For each of these values of t:
    - On each segment we take a point located on the distance proportional to t from its beginning. As there are two segments, we have two points.
    - At t=0 – both points will be at the beginning of segments, and for t=0.25 – on the 25% of segment length from the beginning, for t=0.5 – 50% (the middle), for t=1 – in the end of segments.
  - Connect the points to get a curve
- Demo
- https://pomax.github.io/bezierinfo/#whatis
- https://javascript.info/bezier-curve#de-casteljau-s-algorithm

What about more control points?

- 4-point curve:
- From javascript.info:
  - Connect control points by segments: 1 → 2, 2 → 3, 3 → 4. There will be 3 brown segments.
  - For each t in the interval from 0 to 1:
  - We take points on these segments on the distance proportional to t from the beginning. These points are connected, so that we have two green segments.
  - On these segments we take points proportional to t. We get one blue segment.
  - On the blue segment we take a point proportional to t. On the example above it's red.
  - These points together form the curve.
- Demo: https://javascript.info/bezier-curve#de-casteljau-s-algorithm

Polynomial Representation

- For 3-point curve:
  - p1 is first point, p2 is second, etc
  - Parametric equation: $P = (1-t)^2 p1 + 2(1-t)t\ p2 + t^2 p3$
- $$\text{Bézier}(n,t) = \sum_{i=0}^{n} \underbrace{\binom{n}{i}}_{\text{binomial term}} \cdot \underbrace{(1-t)^{n-i} \cdot t^i}_{\text{polynomial term}} \cdot \underbrace{w_i}_{\text{weight}}$$
- Berenstein polynomials
- Demo: https://www.desmos.com/calculator/xfc7hu3cob

How to get the polynomial form

(for quadratic bezier curves)

First the linear interpolations

$$\mathbf{B}(t) = (1-t)[(1-t)\mathbf{P}_0 + t\mathbf{P}_1] + t[(1-t)\mathbf{P}_1 + t\mathbf{P}_2],\ 0 \le t \le 1$$

Expand and rearrange in terms of t and 1-t

$$\mathbf{B}(t) = (1-t)^2\mathbf{P}_0 + 2(1-t)t\mathbf{P}_1 + t^2\mathbf{P}_2,\ 0 \le t \le 1.$$

You can also rearrange for more symmetry:

$$\mathbf{B}(t) = \mathbf{P}_1 + (1-t)^2(\mathbf{P}_0 - \mathbf{P}_1) + t^2(\mathbf{P}_2 - \mathbf{P}_1),\ 0 \le t \le 1.$$

Applications

- Images (SVG)
- Animation (CSS Transitions)
- CAD/3D Modeling
- Digital Art
- Fonts