Bézier Splines

- Bézier Curves were discovered by Pierre Bézier.
- Approximating the shape of a control polygon.
- Mathematically: \( P(t) = \sum_{i=0}^{n} B_i J_{n,i}(t) \) with \( 0 \leq t \leq 1 \)
- Where \( J_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i} \) and is called the Bernstein basis.
- \( \binom{n}{i} = \frac{n!}{i!(n-i)!} \)

\[ \sum_{i=0}^{n} B_i J_{n,i}(t) = 1 \]

Partition of Unity

Positivity

\[ B_0 \to B_1 \to B_2 \to B_3 \]

\( t = 0 \)

\( t = 1 \)

Control polygon

Curve

Bézier Splines

- Cubic Bézier Splines

\[ P(t) = \sum_{i=0}^{3} B_i J_{3,i}(t) \] with \( 0 \leq t \leq 1 \)

\[ J_{3,0}(t) = (1-t)^3 \]
\[ J_{3,1}(t) = 3t(1-t)^2 \]
\[ J_{3,2}(t) = 3t^2(1-t) \]
\[ J_{3,3}(t) = t^3 \]
Bézier Splines

- Cubic Bézier Splines

\[ P(t) = \begin{bmatrix} (1-t)^3 & 3t(1-t)^2 & 3t^2(1-t) & t^3 \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix} \]

\[ = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix} \]

Bézier Splines

- Cubic Bézier Splines

\[ P(t) = \begin{bmatrix} (1-t)^3 & 3t(1-t)^2 & 3t^2(1-t) & t^3 \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix} \]

\[ = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix} \]

Bézier Splines

- The basis functions are real.
- Degree of the polynomial defining the Bézier curve is one less than the number of defining control polygon points.
- The curve generally follows the shape of the control polygon.
- Endpoint Interpolation: The first and last point of the curve are coincident with the first and last point of the control polygon.

\[ \begin{align*}
 i = 0, J_{n,v} &= \frac{n!}{v!0!(n-v)!} (0)^v(1-0)^{n-v} = 1 \\
 i \neq 0, J_{n,v} &= \frac{n!}{(n-v)v!(n-1)!} (0)^v(1-0)^{n-v-1} = 0 \\
 i = n, J_{n,v} &= \frac{n!}{0!n!(n-n)!} (1)^n(1-1)^{n-n} = 1 \\
 i \neq n, J_{n,v} &= \frac{n!}{(n-v)v!(n-1)!} (1)^v(1-1)^{n-v-1} = 0 \\
 \Rightarrow P(0) &= \sum_{i=0}^{n} B_i J_{n,i} = B_0 \\
 \Rightarrow P(1) &= \sum_{i=0}^{n} B_i J_{n,i} = B_n
\end{align*} \]

Bézier Splines

- Affine Invariance: Applying an affine transform to the curve is equivalent to applying the transformation to the control points.

\[ M x = Ax + l \]
\[ \Rightarrow MP(t) = A \sum_{i=0}^{n} B_i J_{n,i}(t) + l \]
\[ = A \sum_{i=0}^{n} B_i J_{n,i}(t) + l \sum_{i=0}^{n} J_{n,i}(t) \]
\[ = \sum_{i=0}^{n} (AB_i + l) J_{n,i}(t) = \sum_{i=0}^{n} (MB_i) J_{n,i}(t) \]
Bézier Splines

- Convex Hull: The curve lies inside the convex hull of the control points.
- Given a set of points \( x = [x_0, x_1, ..., x_n] \) the convex hull of \( x \) is given by the set of points:
  \[ \mathcal{CH}(X) = \{a_0 x_0 + \ldots + a_n x_n | \sum a_i = 1, a_i \geq 0, a_i \in \mathbb{R}\} \]
- Every point on the curve is of the form:
  \[ P(t) = \sum_{i=0}^{n} B_i J_{n,i}(t) \quad \text{with} \quad 0 \leq t \leq 1 \]
- So every point lies in the convex hull of \( B_i \).

Bézier Splines

- Symmetry: \( J_{n,i}(t) = J_{n-1-i,1-t}(1-t) \)
- The curve \( P(t) \) formed by the control points \( B_0, ..., B_n \) is the same as the curve \( P(1-t) \) formed by the control points \( B_n, ..., B_0 \).

- Parameter Domain Transformation:
  \[ t = \frac{a - u}{b - a} \]
  \[ \Rightarrow P(t) = \sum_{i=0}^{n} B_i J_{n,i}(t) = \sum_{i=0}^{n} B_i J_{n,i}(\frac{a - u}{b - a}) \]

Bézier Splines

- Variation Diminishing Property: The number of intersections of a given straight line with a planar Bézier curve is less than or equal to the number of intersections of that line with the control polygon.

- Tangent Vectors:
  \[ \frac{dP(t)}{dt} = \sum_{i=0}^{n} \frac{d}{dt} B_i J_{n,i}(t) \]
  \[ = \sum_{i=0}^{n} B_i J_{n,i}(t) \frac{d}{dt} (1-t)^{n-i} = \sum_{i=0}^{n} B_i J_{n,i}(t) \int_{0}^{1} (1-t)^{n-i} -(1-t)^{n-i-1} \]
  \[ = n \sum_{i=0}^{n-1} B_i (J_{n-1-i,1-t} - J_{n-1-i,1-t}(t)) \]
  \[ = \sum_{i=0}^{n} B_i J_{n,i}(t) n \sum_{i=0}^{n-1} B_i J_{n-1,i}(t) - n \sum_{i=0}^{n-1} B_i J_{n-1,i}(t) \]
  \[ = n \sum_{i=0}^{n} B_i J_{n,i}(t) - n \sum_{i=0}^{n-1} B_i J_{n-1,i}(t) = n \sum_{i=0}^{n-1} B_i J_{n-1,i}(t) \]
Bézier Splines

- **Tangent Vectors:**
  - i.e., tangent vectors at the ends of curve have the same direction as the first and last spans of the control polygon.
  - Continuity:\( P(t) \) of degree \( n \), defined by control vertices \( B_i \)
  - \( Q(s) \) of degree \( m \), defined by control vertices \( C_i \)

\[
P'(0) = n(B_1 - B_0) \\
P'(1) = n(B_n - B_{n-1}) \]

- \( i \), \( j \) continuity:
  - \( P'(i) = Q'(0) \)
  - \( C_1 - C_0 = \frac{n}{m} (B_n - B_{n-1}) \)
  - \( C_1 = \frac{n}{m} (B_n - B_{n-1}) + B_0 \), because \( C_0 = B_0 \)

i.e., \( B_{n-1} \), \( B_n = C_0, C_1 \) have to be collinear for the tangents to have the same direction.

**Control:**
- How does the shape of the curve change if we move one control point?
- Each control point is associated to a basis function.
- The basis functions effect the shape of the curve over a range of parameter values where the basis function is non-zero. In case of the Bernstien basis, this is the entire parameter range \([0, 1]\\).

Points on the curve: De Casteljau’s Algorithm

- The value for the entry \( j \) of column \( i \) is computed as:
  - \( B_{i,j} = (1-u)B_{i-1,j} + uB_{i-1,j+1} \) where \( 1 \leq i \leq n, \ 0 \leq j \leq n-i \)

```cpp
def deCasteljau(int i, int j)
{
  if (i == 0) return B_{i,j}
  else return (1-u)* deCasteljau(i-1, j) + u* deCasteljau(i-1, j+1)
}
```

- Is this a good way to implement the algorithm?
- Why bother with the algorithm at all?
- Subdivision and Degree Elevation.