

Lecture 4: Subdivision with de Casteljau method ¹

We have already seen how de Casteljau algorithm can generate a point on a Bézier curve. The same algorithm can be used to subdivide the control polygon and generate a new polygon with closer approximation to the curve. Repeating this process create a *subdivision* curve.

Revisiting de Casteljau algorithm

Lets assume that we have three control points $\mathbf{p}_0, \mathbf{p}_1$ and \mathbf{p}_2 to define a second degree Bézier curve. Suppose we compute the point $\mathbf{p}(u)$ for the parameter value u .

For this we computed the point \mathbf{x} on $\mathbf{p}_0\mathbf{p}_1$ given by $\mathbf{x} = \mathbf{p}_0 + u(\mathbf{p}_1 - \mathbf{p}_0)$ and also the point $\mathbf{y} = \mathbf{p}_1 + u(\mathbf{p}_2 - \mathbf{p}_1)$. The point $\mathbf{z} = \mathbf{x} + u(\mathbf{y} - \mathbf{x})$ lies on the curve. We claim that $\mathbf{p}_0, \mathbf{x}, \mathbf{z}$ constitute the control polygon for the curve between $[0, u]$ and $\mathbf{z}, \mathbf{y}, \mathbf{p}_2$ constitute the control polygon for the curve between $[u, 1]$.

To see this consider the Bézier curve:

$$\mathbf{p}(t) = (1 - t)^2\mathbf{p}_0 + 2t(1 - t)\mathbf{x} + t^2\mathbf{z}$$

Simplifying we get:

$$\mathbf{p}(t) = (1 - ut)^2\mathbf{p}_0 + 2ut(1 - ut)\mathbf{p}_1 + u^2t^2\mathbf{p}_2$$

This is the Bézier curve with the control points $\mathbf{p}_0, \mathbf{p}_1$ and \mathbf{p}_2 and the parameter $w = ut$. Thus, it is the same curve as the original in between $w = 0$ for $t = 0$ and $w = u$ for $t = 1$. Similarly, one can show that \mathbf{z}, \mathbf{y} and \mathbf{p}_2 generate the same curve between $[u, 1]$.

The new polygon $\mathbf{p}_0, \mathbf{x}, \mathbf{z}, \mathbf{y}, \mathbf{p}_2$ thus approximate the same curve, but in a better manner. We can continue this process to subdivide the two polygons between $[0, u]$ and $[u, 1]$. After m such steps we generate 2^m such polygons that are joined together at m points on the curve. This process converges to the curve very fast.

Algorithm

The procedure OneSubdivide subdivides the polygon once at the parameter value u . We use the notation poly1.poly2 to denote the concatenation of the two lists poly1 and poly2 without repeating the common end-element between the two.

```
OneSubdivide( $\mathbf{p}_0, \dots, \mathbf{p}_n, \text{poly1}, \text{poly2}, u$ )
  if  $n = 0$  output  $\text{poly1}.\{\mathbf{p}_0\}.\text{poly2}$ 
  else
     $\text{poly1} := \text{poly1}.\mathbf{p}_0; \text{poly2} := \mathbf{p}_n.\text{poly2};$ 
    compute  $\mathbf{p}'_i = \mathbf{p}_i + u(\mathbf{p}_{i+1} - \mathbf{p}_i), i = 0, \dots, n - 1$ 
    OneSubdivide( $\mathbf{p}'_0, \dots, \mathbf{p}'_{n-1}, \text{poly1}, \text{poly2}, u$ )
  endif
end
```

¹Note by Tamal K. Dey

```

Subdivide ( $\mathbf{p}_0, \dots, \mathbf{p}_n, m, u$ )
  if  $m = 1$  OneSubdivide( $\mathbf{p}_0, \dots, \mathbf{p}_n, \{\}, \{\}, u$ )
  else
     $\{\mathbf{p}'_0, \dots, \mathbf{p}'_n, \dots, \mathbf{p}'_{2n}\} :=$  OneSubdivide( $\mathbf{p}_0, \dots, \mathbf{p}_n, \{\}, \{\}, u$ );
    Subdivide( $\mathbf{p}'_0, \dots, \mathbf{p}'_n, m - 1, u$ ). Subdivide( $\mathbf{p}'_n, \dots, \mathbf{p}'_{2n}, m - 1, u$ );
  endif
end

```

This algorithm Subdivide makes 2^m calls to itself. Each of these calls makes one call to OneSubdivide which takes $O(n^2)$ time as the de Casteljau procedure. Thus the complexity of the subdivision is $O(2^m n^2)$.