Lecture 4: Subdivision with de Casteljau method

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

Let’s assume that we have three control points \( p_0, p_1, p_2 \) to define a second degree Bézier curve. Suppose we compute the point \( p(u) \) for the parameter value \( u \).

For this we computed the point \( x \) on \( p_0p_1 \) given by \( x = p_0 + u(p_1 - p_0) \) and also the point \( y = p_1 + u(p_2 - p_1) \). The point \( z = x + u(y - x) \) lies on the curve. We claim that \( p_0, x, z \) constitute the control polygon for the curve between \([0, u]\) and \( z, y, p_2 \) constitute the control polygon for the curve between \([u, 1]\).

To see this consider the Bézier curve:

\[
p(t) = (1 - t)^2p_0 + 2t(1 - t)x + t^2z
\]

Simplifying we get:

\[
p(t) = (1 - ut)^2p_0 + 2ut(1 - ut)p_1 + u^2t^2p_2
\]

This is the Bézier curve with the control points \( p_0, p_1, p_2 \) 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 \( z, y \) and \( p_2 \) generate the same curve between \([u, 1]\).

The new polygon \( p_0, x, z, y, p_2 \) 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 \( 2^m - 1 \) 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(p_0, ..., p_n, poly1, poly2, u)

if \( n = 0 \)
output poly1, \{p_0\}, poly2
else
poly1:= poly1,p_0; poly2:=p_n,poly2;
compute \( p'_i = p_i + u(p_{i+1} - p_i) \), \( i = 0, ..., n - 1 \)
OneSubdivide(\( p'_0, ..., p'_{n-1}, \)poly1,poly2,u)
endif

end

1Note by Tamal K. Dey
\textbf{Subdivide (}p_0, ..., p_n, m, u\textbf{)}

\begin{verbatim}
    if m = 1 OneSubdivide(p_0, ..., p_n, \{\}, \{\}, u)
    else
        \{p_0', ..., p_n', ..., p_{2n}'\} := OneSubdivide(p_0, ..., p_n, \{\}, \{\}, u);
        Subdivide(p_0', ..., p_n', m - 1, u). Subdivide(p_n', ..., p_{2n}', m - 1, u);
    endif
\end{verbatim}

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