MA 323 Geometric Modelling
Course Notes: Day 12
de Casteljau's Algorithm and Subdivision

David L. Finn

Yesterday, we introduced barycentric coordinates and de Casteljau's algorithm. Today, we want to go more in depth into the mechanics of de Casteljau's algorithm, and understand some of the nuances of the algorithm. We also want to discuss the efficiency of this algorithm in creating the curve. The algorithm's power is not necessarily in defining a polynomial curve, but in how the algorithm can approximate the curve it produces very quickly. In particular, de Casteljau's algorithm admits can be viewed as a subdivision method for creating an approximation to the curve.

The idea of a subdivision method is a useful idea in geometric modelling and any numerical algorithm that employs recursion. A subdivision method is a recursive method for creating an approximation. For de Casteljau's algorithm, the subdivision method is based on noticing that the control polyline to the curve basically provides the shape of the curve. Moreover, when applying de Casteljau's algorithm the curve is further approximated by the portions of lines drawn. The subdivision algorithm provides a method for defining the points needed to approximate the curve.

12.1 de Casteljau's algorithm revisited

Yesterday, we introduced the algorithm as a method for creating a curve by repeated linear interpolation from a collection of points. In particular, the algorithm is

> From the control points p_0, p_1, $\cdots$, p_n, we define points on the line segments $p_i p_{i+1}$ (the control polyline) by choosing a value t and defining $p_i^1 = (1-t)\,p_i + t\,p_{i+1}$. Notice that we defined n points in this manner as $i = 0, 1, 2, \cdots, n-1$. From the n points p_i^1, we can repeat the process and define the $n-1$ points $p_i^2 = (1-t)\,p_i^1 + t\,p_{i+1}^1$ on the line segments $p_i^1 p_{i+1}^1$. This process can be repeated defining
>
> $$p_i^{j+1} = (1-t)\,p_i^j + t\,p_{i+1}^j$$
>
> for $i = 0, 1, 2, \cdots, n-j-1$ for $j = 0, 1, 2, \cdots, n-1$ where $p_i^0 = p_i$. The end of this process produces a point p_0^n on a polynomial curve of degree n. An example of this procedure is shown in the diagram below constructing a cubic curve.

Notice to calculate one point on the curve, we need to calculate the position of n points on the control polyline. Then, we calculate $n-1$ points using the line segments of the first n control points, and so on. Thus, we need to calculate $n + (n-1) + (n-2) + \cdots + 2 + 1 = n(n+1)/2$ points to calculate one point, see diagram above

de Casteljau's algorithm applied to $n+1$ control points generates an nth degree polynomial curve. This follows from an inductive argument. We have demonstrated this algorithm for

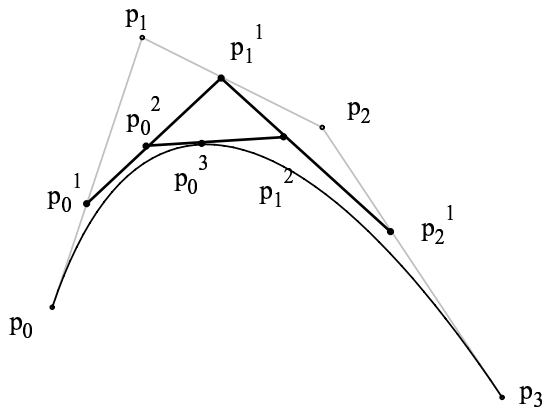

Figure 1: Construction of a curve by repeated linear interpolation

parabolas. If we assume that with n points, we get a $n-1$th degree polynomial curve. We note that the point p_0^{n-1} is obtained by applying de Casteljau's algorithm to the control points p_0, p_1, $\cdots$, p_{n-1}, and thus is a $n-1$th degree polynomial curve. Moreover, the point p_1^{n-1} is obtained by applying de Casteljau's algorithm to the control points p_1, p_2, $\cdots$, p_n, and therefore is also a $n-1$th degree polynomial curve. The point $p_0^n = (1-t)\,p_0^{n-1} + t\,p_1^{n-1}$ on the curve is thus an nth degree polynomial in t. The same sort of inductive argument can be applied to show that de Casteljau's algorithm can be written in barycentric coordinates, that is

$$c(t) = \alpha_0^n(t)\,p_0 + \alpha_1^n(t)\,p_1 + \cdots + \alpha_n^n(t)\,p_n$$

where $\alpha_0(t) + \alpha_1(t) + \cdots + \alpha_n(t) = 1$. In particular, we have

$$p_0^{n-1}(t) = \alpha_0^{n-1}(t)\,p_0 + \alpha_1^{n-1}(t)\,p_1 + \cdots \alpha_{n-1}^{n-1}(t)\,p_{n-1}$$

plus

$$p_1^{n-1}(t) = \alpha_0^{n-1}(t)\,p_1 + \alpha_1^{n-1}(t)\,p_2 + \cdots \alpha_{n-1}^{n-1}(t)\,p_n$$

with $\alpha_0^{n-1}(t) + \alpha_1^{n-1}(t) + \cdots + \alpha_{n-1}^{n-1}(t) = 1$ by the induction hypothesis. This implies with $c(t) = (1-t)\,p_0^{n-1}(t) + t\,p_1^{n-1}(t)$ that for

$$c(t) = \alpha_0^n(t)\,p_0 + \alpha_1^n(t)\,p_1 + \cdots + \alpha_n^n(t)\,p_n$$

we have

$$\begin{aligned}
\alpha_0^n(t) &= (1-t)\,\alpha_0^{n-1}(t) \\
\alpha_1^n(t) &= (1-t)\alpha_1^{n-1}(t) + t\alpha_0^{n-1}(t) \\
&\vdots \\
\alpha_i^n(t) &= (1-t)\alpha_i^{n-1}(t) + t\alpha_{i-1}^{n-1}(t) \\
&\vdots \\
\alpha_n^n(t) &= t\alpha_{n-1}^{n-1}(t)
\end{aligned}$$

and therefore

$$\begin{aligned}
&\alpha_0^n(t) + \cdots + \alpha_n^n(t) \\
&\quad = (1-t)\,(\alpha_0^{n-1}(t) + \cdots + \alpha_{n-1}^{n-1}(t)) + t\,(\alpha_0^{n-1}(t) + \cdots + \alpha_{n-1}^{n-1}(t)) \\
&\quad = (1-t) + t = 1.
\end{aligned}$$

One can instead expand the algebra, (which we look at after break), and find that each basis function $\alpha_i(t)$ is an nth degree polynomial curve, specifically

$$\alpha_i(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

where $\binom{n}{i}$ is a binomial coefficient that can be computed by Pascal's triangle.

12.2 A Simple Method for Approximating the Curve

The simplest method approximating the curve created by de Casteljau's algorithm is to divide the interval $[0,1]$ into a collection of subintervals, that is obtaining a partition $0 = t_0 < t_1 < t_2 < \cdots < t_n = 1$ of the interval $[0,1]$, and computing the points $P(t_0)$, $P(t_1)$, $\cdots$, $P(t_n)$. An approximation of the curve is given by the linear interpolant through these points. Using more points and equal subdivisions $t_1 - t_0 = t_2 - t_1 = \cdots = t_n - t_{n-1}$, we obtain a good approximation to the a continuous curve. This is a direct result of the continuity of the curve. This method does not take into account any facts about the algorithm. In fact, this method will work for any curve that is defined on a closed interval. We can increase the approximation accuracy of the approximation and the decrease the amount of time needed to compute the approximation by exploiting the nature of the algorithm.

The subdivision algorithm that we will now discuss is built on using some of the intermediate points in de Casteljau's algorithm to achieve a good approximation of the curve. This method relies on the fact that the control polyline provides a first approximation of the curve, and exploiting a theorem from Euclidean geometry called Menelaus' theorem. To explain the method, we will first use the method to construct an approximation of a parabola. Then, we will write an explicit description of the method.

Given three noncollinear points P_0, P_1, P_2 as in the diagram below. We first apply de Casteljau's algorithm with $t = 1/2$ to obtain the points P_0^1, P_1^1, P_0^2 where P_0^2 is on the parabola. Notice, in this construction the linear interpolant through the points $P_0 = P_0^0$, P_0^1, P_0^2, P_1^1 and $P_2^0 = P_2$ forms a better approximation to the parabola than the original control polyline. Moreover, if we consider the set P_0^0, P_0^1 and P_0^2 as control points we have P_0^0 and P_0^2 as points on the parabola and P_0^1 off the parabola, and the intersection of the tangent lines to the parabola at P_0^0 and P_0^2. We have a similar statement about the points P_0^2, P_1^1 and P_2^0. In fact, if we apply de Casteljau's algorithm on the points P_0^0, P_0^1, P_0^2, we obtain the same parabola. We likewise obtain the same parabola if we apply de Casteljau's algorithm on the points P_0^2, P_1^1, P_2^0. Therefore, applying the same procedure on the points $[P_0^0, P_0^1, P_0^2]$ and the points $[P_0^2, P_1^1, P_2^0]$, we will obtain a better approximation with the corresponding points.

12.3 the Subdivision Method for de Casteljau's algorithm

In our discussion of the subdivision method for de Casteljau's algorithm, we will restrict our attention to using midpoints of line segments. To approximate the curve given by de Casteljau's algorithm efficiently, we define an iteration process starting with the sequence of control points P_0, P_1, P_2, $\cdots$, P_n. The sequence of control points defines the zeroth iteration as the control polyline (the linear interpolant or piecewise linear curve through the points P_0, P_1, $\cdots$, P_n).

In the first iteration of the subdivision algorithm, we use de Casteljau's algorithm with $t = 1/2$ to calculate one point, and thus we split the curve into two segments. de Casteljau's

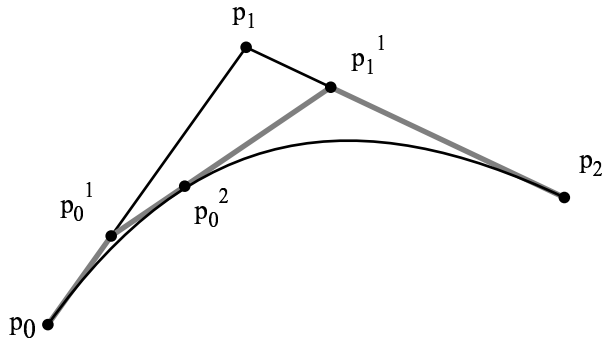

Figure 2: A better approximation to a parabola.

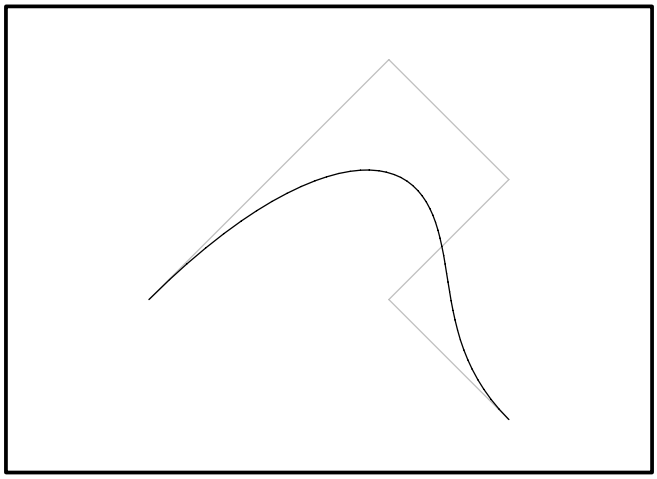

Figure 3: The zeroth iteration of subdivision

algorithm then defines a set of control points that will generate each curve segment. The control points P_0^{n-i} with $i = 0, 1, 2, \cdots, n$ generate the segment $P(t)$ with $0 \leq t \leq 1/2$ and the control points P_i^{n-i} with $i = 0, 1, 2, \cdots, n$ generates the segment $P(t)$ with $1/2 \leq t \leq 1$. The linear interpolant $\mathcal{PL}_1$ through the points P_0^0, P_0^1, $\cdots$, P_0^n, P_1^{n-1}, P_2^{n-2}, $\cdots$, P_n^0 (the control polylines for the segments $P(t)$ with $0 \leq t \leq 1/2$ and $P(t)$ with $1/2 \leq t \leq 1$) forms a better approximation to the curve than $\mathcal{PL}_0$. For an illustration see the diagram below.

We continue the process in a recursive manner applying the method onto each segment. Thus, in the second iteration of the subdivision method, we subdivide the two segments of the original curve obtaining four segments of the original curve. The third iteration obtains eight segments of the original curve and so forth. After k iterations, we have 2^k segments of the original curve.

We stated above that the control points P_0^0, P_0^1, $\cdots$, P_0^n produce the segment of the curve $P(t)$ with $0 \leq t \leq 1/2$. To show that this is true involves a version of Menelaus' theorem

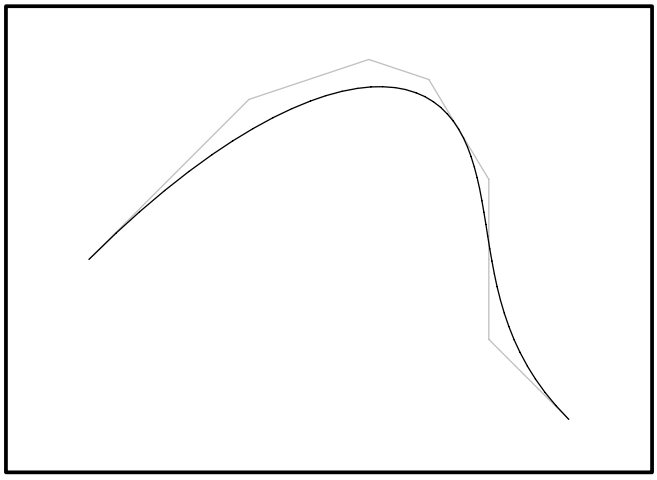

Figure 4: The first iteration of subdivision

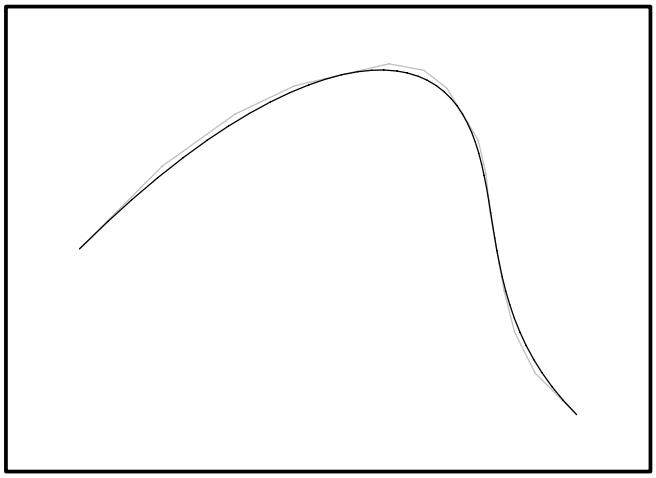

Figure 5: The second iteration of subdivision

12.4 A Version of Menelaus' Theorem

The mathematical construction that allows a multivariate version of de Casteljau's algorithm to work is Menelaus' theorem. Other formulations of de Casteljau's algorithm involve blossoming. In blossoming, each the subinterval from different iteration are given a different parameter, i.e. the construction of a parabola is written as

$$(1-s)((1-t)p_0 + tp_1) + s((1-t)p_1 + tp_2).$$

The parabola is then arrived at using $s = t$. That it does not matter which variable is computed first is a result of Menelaus' theorem.

A version of Menelaus' theorem is given as let P_0, P_1, P_2 be three points in a plane. Define the points $a_0 = (1-t)\,P_0 + t\,P_1$ and $a_1 = (1-t)\,P_1 + t\,P_2$. Also define the points $b_0 = (1-s)\,P_0 + s\,P_1$ and $b_1 = (1-s)\,P_1 + s\,P_2$. Using algebra, it is easy to show that

$$c = (1-s)\,a_0 + s\,a_1 = (1-t)\,b_0 + t\,b_1.$$

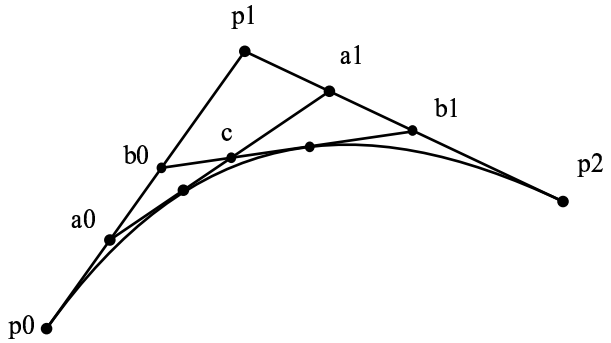

Figure 6: A CAGD version of Menelaus's Theorem

This takes on special interest, when we recast it in terms of repeated linear interpolation. Define

$$\begin{aligned}\mathbf{b}[0,t] &= (1-t)\,b_0 + t\,b_1;\\ \mathbf{b}[1,t] &= (1-t)\,b_1 + t\,b_2;\\ \mathbf{b}[s,0] &= (1-s)\,b_0 + s\,b_1;\\ \mathbf{b}[s,1] &= (1-s)\,b_1 + s\,b_2.\end{aligned}$$

Further, define

$$\begin{aligned}\mathbf{b}[s,t] &= (1-s)\,\mathbf{b}[0,t] + s\,\mathbf{b}[1,t]\\ \mathbf{b}[t,s] &= (1-t)\,\mathbf{b}[s,0] + t\,\mathbf{b}[s,1].\end{aligned}$$

Menelaus' theorem implies that $\mathbf{b}[s,t] = \mathbf{b}[t,s]$.

This gives us a multivariate way of repeatedly applying linear interpolation, and will be tremendously important later when we define surfaces using de Casteljau's algorithm. Here, we only need Menelaus' theorem to show that the subdivision method works. In other books on geometric modelling, this result is used to motivate other constructions. In particular, this version of Menelaus' theorem is used in blossoming. We will not mention in the remainder of the course, but for those of you who are interested in careers in computer graphics and CAGD should be aware of blossoming. Blossoming allows one to construct most of the geometric information about the curve as repeated linear interpolation. For more information on blossoming consult the text by Farin.

The classical theorem of Menelaus (illustrated above) is stated in terms of ratios and used to determine when points are collinear. The ratio of three collinear points P, X and Q is defined through the barycentric coordinates of the middle point X in terms of the points P and Q. Since X lies on the line P and Q, X can be written in barycentric coordinates as

$$X = (1-t)\,P + t\,Q$$

for some number t. The ratio of P, X and Q is then defined as

$$\text{ratio}(P,X,Q) = \frac{t}{1-t}$$

The ratio(P,X,Q) is equal to the proportion of PX/PQ to XQ/PQ, where PX/PQ and XQ/PQ are defined by the division of lengths. The classical theorem of Menelaus concerns the collinearity of points defined through a triangle. Let ABC be a triangle and let X be

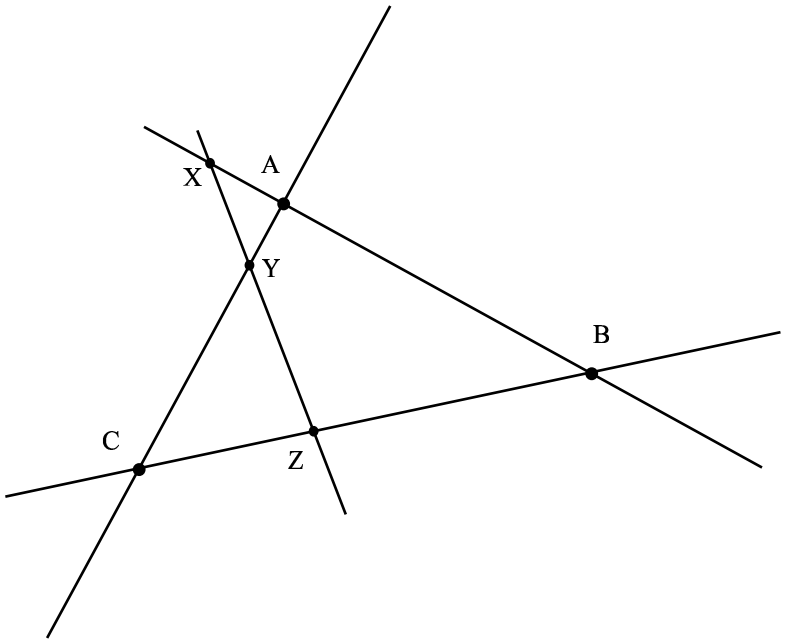

Figure 7: The classical version of the theorem of Menelaus

a point on the line AB, Y be a point on the line BC and Z be a point on the line AC. Menelaus' theorem states that the points X, Y and Z are collinear if and only if

$$\text{ratio}(A, X, B)\,\text{ratio}(B, Y, C)\,\text{ratio}(C, Z, A) = -1.$$

12.5 Why the Subdivision Method Works

From de Casteljau's algorithm (that the tangent line is given by $P_0^{n-1}(t)$ and $P_1^{n-1}(t)$) and facts about parabolas, it should be geometrically obvious that the this holds true for parabolas. A parabola is well defined by two points and two tangent lines. However, if we approach this problem (showing that the control points P_0^0, P_0^1, P_0^2 generates the segment of the curve $P(t)$ with $0 \leq t \leq 1/2$) in a more algebraic manner then we can extend the method to the general setting.

To understand the application of Menelaus' theorem to the subdivision method, consider the diagram below. The CAGD version of Menelaus' theorem states that the intersection point X of the lines $P_0^1(1/2)\,P_1^1(1/2)$ and $P_0^1(t)\,P_1^1(t)$ in the diagram below is given by

$$(1-t)\,P_0^1(1/2) + t\,P_1^1(1/2) = (1/2)[(1-t)\,P_0 + t\,P_1] + (1/2)[(1-t)\,P_1 + t\,P_2].$$

We note that the point $P_0^1(t) = (1-2t)\,P_0 + 2t\,P_0^1(1/2)$ and the thus the point $P_0^2(t) = (1-2t)\,P_0^1(1/2) + 2t\,P_0^2(1/2)$. Therefore, since the point

$$X = (1/2)[(1/2)[(1-t)\,P_0 + t\,P_1] + (1/2)[(1-t)\,P_1 + t\,P_2],$$

we have that $(1-t)^2\,P_0 + 2t(1-t)\,P_1 + t^2\,P_2$ is equal to $(1-2t)^2\,P_0^0(1/2) + 4t(1-2t)\,P_0^1(1/2) + (2t)^2\,P_0^2(1/2)$.

Exercises

1. Construct an approximation of the curve defined by de Casteljau's algorithm defined the control points below.

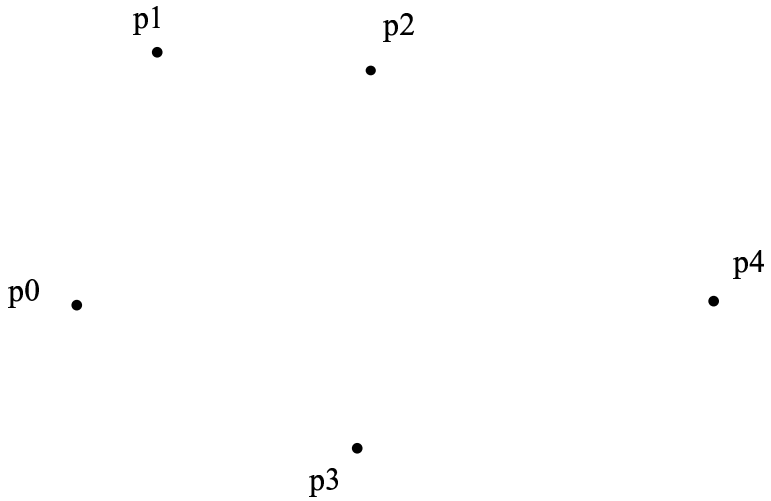

Figure 8: Apply Subdivision Method for de Casteljau's algorithm

   (a) by applying the subdivision method twice.
   (b) by applying the subdivision method four times.

2. Complete the interactive exercises for the Subdivision Method.