Bézier Curves and Splines

Douglas A. Kerr

Issue 2
May 29, 2011

ABSTRACT

A *cubic Bézier curve* is a two-dimensional curve that is completely defined by the locations of four points, two of which are the endpoints of the curve. The cubic Bézier curve is the basic ingredient of the curve constructing capacities of many technical illustration, CAD, and other graphic programs, and also has application to the description of type face glyphs. More complex curves may be created as *cubic Bézier splines*, which comprise two or more cubic Bézier curves, joined together (most commonly so the joint is smooth). In this article we describe the concept of the cubic Bézier curve, first in an intuitive way, then through a geometric construction procedure, and finally in terms of mathematical functions. Finally, we mention the *quadratic Bézier curve*, a special (and simpler) subset of the cubic Bézier curve, as well as the *linear Bézier curve* and Bézier curves of higher order. We also discuss the distinction between *paths* and *strokes*, concepts pertaining to the use of Bézier curves in illustration and image editing software.

INTRODUCTION

In many computer graphic applications, including technical illustration programs, CAD programs, and so forth, the user must be able to define curves. Ideally, everything in these programs is defined in what is sometimes spoken of as *vector* terms; that is, as a mathematical-geometric description, rather than as a collection of pixels. In this way, the path can be scaled or rotated without loss of precision due to the discrete nature of a pixel representation. The vector representation is also much better for performing various operations, such as determining the precise point of intersection between two paths, or adjusting the shape of a curve.

Many such applications today use as their basic curve ingredient a type of curve known as the *cubic Bezier curve*. Such a curve is completely and rigorously defined by the locations of four points (called *control points*), two of which are in fact the endpoints of the curve.

Since defining the location of a point requires the value of two variables, a cubic Bézier curve is thus defined by eight values. Mathematically, we say it has eight degrees of freedom.

But we cannot make a cubic Bézier curve any arbitrary shape that we might want. An infinite number can be defined, but there are infinitely more curves that are not cubic Bézier curves.

THE DETAILS

Basic concept

As we noted, a cubic Bézier curve is completely defined by the locations of four points. No other information is needed, and there are no opportunities to vary the curve from what is dictated by the points.

Figure 1 shows a cubic Bézier curve, defined by the four control points, P0-P3:

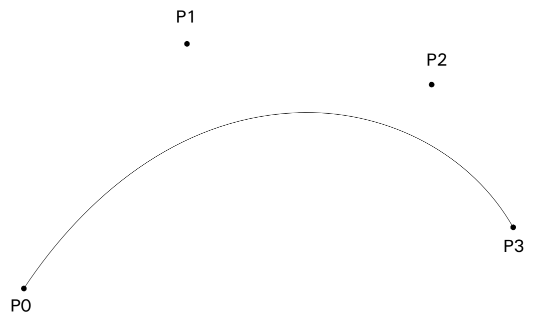


Figure 1. Cubic Bézier curve, showing control points

Control points P0 and P3 are the endpoints of the curve. In the notation of many illustration programs, these two control points are called *anchor points*, and the term *control point* is then used only for the points P2 and P3. (This notation is very descriptive, and I will generally follow it here to avoid any ambiguity.)

Points P2 and P3 define the shape of the curve in a way we will see shortly.

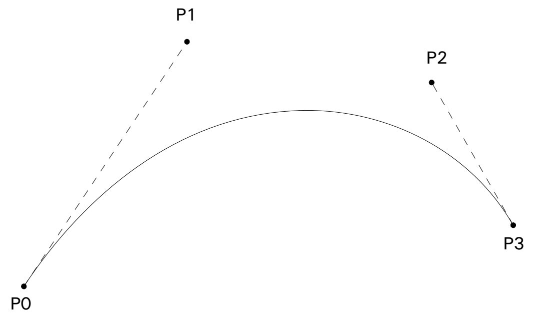


Figure 2. Cubic Bézier curve showing control vectors

We often see in our drawing program lines connecting the anchor points with the adjacent (that's in numerical order) control points. These lines may be thought of as *control vectors*.

Note that these vectors are tangent to the curve at its end points. That is, the slope of the curve at the endpoint is the same as the slope of the control vector. Said another way, the curve "takes off" from an end point initially heading straight for the associated interior control point.

Figure 3 shows this again for a different location of point P1:

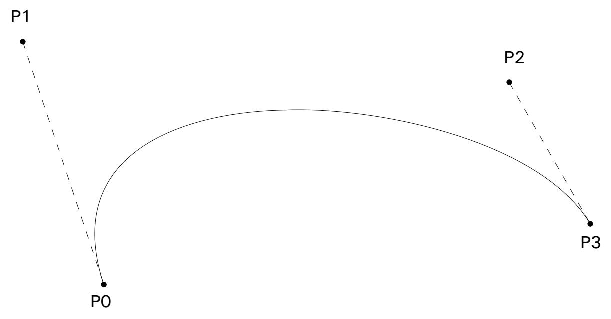


Figure 3. A different location for P1

As before, the curve "takes off" from P0 initially in the direction along the vector P0-P1.

Developing the curve itself

Of course, if we have the four points that define a cubic Bézier curve, for this to be useful we must be able to "develop" the actual curve they define.

We can do this in purely mathematical terms (and will later), or with a geometric interpretation of the mathematics (which we will do not so much later).

But first, it will be useful to get a conceptual grasp of how the control vectors control the shape of the curve, so we will not be surprised by the result the geometric and mathematical approaches give us.

An intuitive outlook: the bug metaphor

The principle by which these control vectors control the shape of the curve may be broadly understood by way of a fanciful metaphor in which the curve is the actual path of a little crawling bug. Refer again to figure 2. The bug starts off from point P0, and his mission is to get to point P3.

But as he prepares to set out, he hears a seductive voice from point P1, calling: “Over here, over here”. He initially heads straight toward the voice, but immediately realizes that his mission is to get to point P3. So (perhaps reluctantly—the voice from P1 was **very** seductive). Thus he begins steering away from the bearing to point P1 and toward the general direction of point P3.

The influential power of P1 varies with the length of the vector P0-P1. The longer the vector, the greater is the distracting influence on our bug. (That seems counterintuitive—we might think, “the ‘siren’ is farther away, and its ‘voice’ would be less potent”—but remember this is just a metaphor, not a physics model.)

To see this, we will relocate P1 so that the vector P0-P1 has the same direction as in figure 2 but is shorter (figure 4).

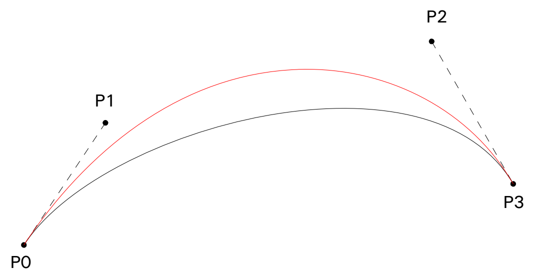


Figure 4. Shorter vector P0-P1

Now we see that our little bug much quickly veers from his initial course toward P1 than he did under the (greater) influence of P1 in its prior location. (His path in that earlier situation is shown in red for comparison.)

Similarly, P2 has an influence on the bug’s course on the right side of the figure, as it approaches P3. But the metaphor doesn’t work as cleanly in explaining that. It is perhaps best to think of things at the P3 end as just being parallel to things at the P0 end—imagine the bug leaving P3, bound for P0.

A RIGOROUS DESCRIPTION

What we have done so far is broad and qualitative. Now let’s take a more rigorous outlook on the shape of the curve.

Ultimately, we would probably like to have the curve expressed as in terms of the values of *x* and *y* (the coordinates of any given point on the curve) for all possible points along the length of the curve.

Parametric functions

It is in fact very difficult to develop an description of the curve in such a form as "*y* as a function of *x*". One deadly complication is that the function may well be two- or three-valued—for a given value of a, there might be two (or more) values of y (we can see that on the left portion of the curve in figure 3).

But we can handily finesse that and other complications by using a *parametric* function approach. There, we have both *x* and *y* as separate functions of a dummy variable, usually designated *t* (said to be the *parameter* of the system of functions). That particular choice of symbol reminds us that the dummy variable can be considered *time* in the crawling bug metaphor for the formation of the curve. The entire curve is thought of as being "drawn" by the bug while *t* runs from 0 to 1 (a period of one unit of time).

For any given value of *t*, straightforward algebraic functions will tell us the value of *x* and of *y*—that is, where the "pencil" is at that "instant" in the drawing of the entire curve. (Do not assume that the "pencil" moves along the curve at a constant rate in terms of distance per unit of this fanciful time scale. We only know that it makes the whole trip during the period between $t=0$ and $t=1$.)

The geometric construction model

This parametric outlook will serve us even if, rather than thinking in terms of mathematical functions for x and y, we use a geometric construction model. This model is theoretically rigorous; it precisely represents the mathematics involved.

First, we note that at the beginning of the scenario ($t=0$), the corresponding point on the curve is by definition at P0, and at the end of the scenario ($t=1$), the corresponding point on the curve is by definition at P3. We will then look at three other points, those for $t=0.25$, $t=0.5$, and $t=0.75$.

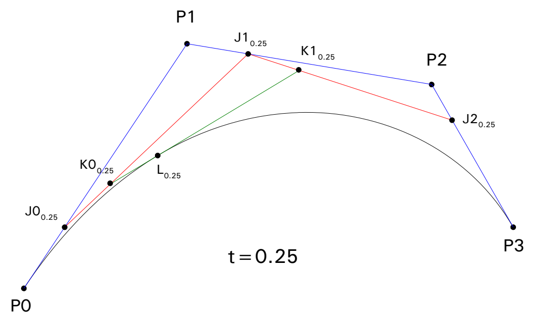


Figure 5. Geometric construction at *t*=0.25

In figure 5, we do the geometric construction for $t=0.25$—that is, determining the location the “pencil” when it is 1/4 of the way along the curve in terms of our fanciful timeline.

- We draw the lines P0-P1, P1-P2, and P2-P3 (the blue lines).
- We locate point $J0_{0.25}$ on line P0-P1, 0.25 of the way from P0 to P1.
- We locate point $J1_{0.25}$ on line P1-P2, 0.25 of the way from P1 to P2.
- We locate point $J2_{0.25}$ on line P2-P2, 0.25 of the way from P2 to P3.
- We draw lines from $J0_{0.25}$ to $J1_{0.25}$, and from $J1_{0.25}$ to $J2_{0.25}$ (red).
- We locate point $K0_{0.25}$ on line $J0_{0.25}$-$J1_{0.25}$, 0.25 of the way from $J0_{0.25}$ to $J1_{0.25}$.
- We locate point $K1_{0.25}$ on line $J1_{0.25}$-$J1_{0.25}$, 0.25 of the way from $J1_{0.25}$ to $J2_{0.25}$.
- We draw a line from $K0_{0.25}$ to $K1_{0.25}$ (green).
- We locate point $L1_{0.25}$ on line $K1_{0.25}$-$K1_{0.25}$, 0.25 of the way from $K1_{0.25}$ to $K2_{0.25}$.

Point $L1_{0.25}$ is the point on the curve that would be “drawn” at t-0.25.

Now that we have seen the process at work, let’s do the same construction at t=0.5 and t=0.75 (figure 6). I’ll spare you the blow-by-blow description.

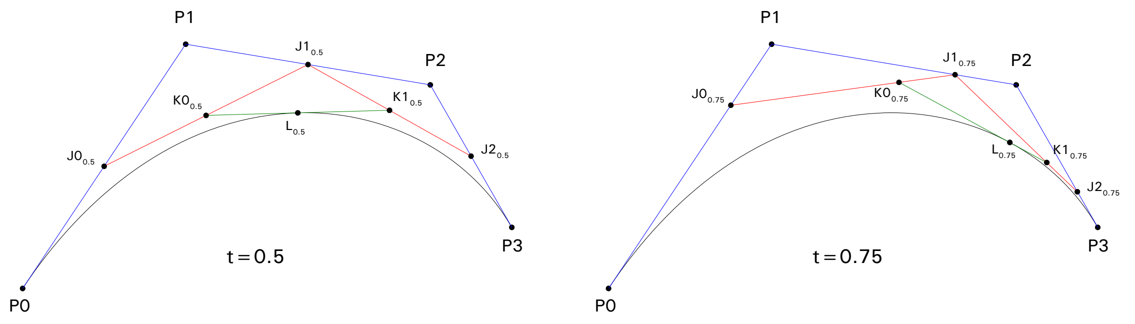


Figure 6. Geometric construction at $t=0.5$ and $t=0.75$

Remember that, at each stage of the process, we locate the point that is the fraction t along the length of line of interest (first working with the blue lines, then the red lines, then the green line). In each case, the point L_t that is the final result is the point on the curve where the “pencil” would be at time t.

A wonderful animation of the entire process (as t goes from 0 to 1) can be found here:

http://upload.wikimedia.org/wikipedia/commons/f/ff/Bezier_3_big.gif

The analytic outlook

As we mentioned above, using the parametric outlook, we can actually determine, analytically, the *x* and *y* coordinates of the point on the curve corresponding to any value of *t* from 0 through 1. The calculations of course involve the coordinates of the four control points, which we designate x_0, y_0 (for point P0) through x_3, y_3 (for point P3).

Then, for any arbitrary value of *t* (over the range 0-1), the values of the coordinates for the corresponding curve point, x_t and y_t, are given by:

$$x = At^3 + Bt^2 + Ct + D \qquad y = Et^3 + Ft^2 + Gt + H \qquad \text{(1a,b)}$$

where the values of the coefficients A-H are given by:

$$A = x_3 - 3x_2 + 3x_1 - x_0 \qquad E = y_3 - 3y_2 + 3y_1 - y_0 \qquad \text{(2a,b)}$$

$$B = 3x_2 - 6x_1 - 3x_0 \qquad F = 3y_2 - 6y_1 - 3y_0 \qquad \text{(3a,b)}$$

$$C = 3x_1 - 3x_0 \qquad G = 3y_1 - 3y_0 \qquad \text{(4a,b)}$$

$$D = x_0 \qquad H = y_0 \qquad \text{(5a,b)}$$

These functions are given, with a lovely derivation, in a useful paper by Don Lancaster, available here (at this writing):

http://www.tinaja.com/glib/cubemath.pdf

OTHER BÉZIER CURVES

The Quadratic Bézier Curve

A simpler curve, actually a special case of the Bézier cubic curve, is called the *quadratic* Bézier curve. Figure 7 shows an example.

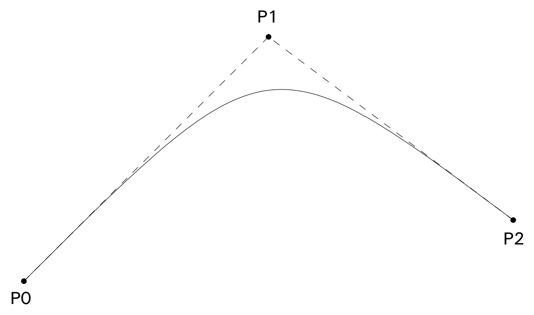


Figure 7. Bézier quadratic curve

It only has three control points, two (P0 and P2) being the endpoints of the curve (anchor points).

It can accurately be thought of as a Bézier cubic curve in which the two control points are at the same place.

The linear Bézier curve

A (straight) line segment is the degenerate case of a Bézier curve, called a linear Bézier curve. A linear is defined by two anchor points, which are its endpoints. But to fit in into the continuum of Bézier curves, we can say that its control points lie along the line segment itself, at any arbitrary distance we wish to think of.

In many cases, we consider the control points as lying at zero distance—that is, they lie at the anchor points.

Higher-order Bézier curves

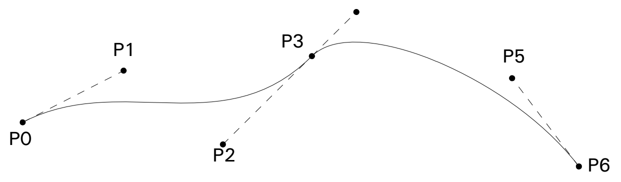
I the general case, we can have genres of Bézier curves defined by any number of points. Of course, these have correspondingly greater degrees of freedom, and thus there is more flexibility in the shapes that can be described.

But Bézier curves of order higher than three (i.e., the cubic Bézier curve) are rarely used in such areas as graphic illustration.

BÉZIER SPLINES

Basic concept

In order to expand our curve defining capabilities, we can utilize *cubic Bézier splines*[1]. These are composed of two or more cubic Bézier curves (each described by four points, and thus eight numerical values), joined together (so there is a control point that is common to two adjacent curves), most often in such a way that the joint is "smooth" (and there is a mathematical definition of what that means). Thus a spline composed of 100 cubic Bézier curves, end-to-end, will be defined by 301 points (101 of them anchor points, lying on the spline itself), and thus by 602 numerical values.



[1] The term comes from traditional drafting, where curves were often drawn with the aid of a deformable bar (traditionally made of lead), called a *spline*, which the drafter would form until it met the "specifications" for the desired curve.

Figure 8. Bézier spline (two-segment)

In figure 8, we see a two-segment Bézier spline (composed of two Bézier curves joined end-to-end). The first curve is defined by points P0-P3, and the second by points P3-P6 (point P3 being the common anchor point, at the "joint" between the two segments).

Note that in many cases, when we construct a so-called "Bézier curve" in a graphics program we are actually constructing a cubic Bézier *spline*. How we "enter" and manipulate these splines is beyond the scope of this article.

Smooth joining

I spoke just above of the fact that ordinarily, in a cubic Bézier spline, the adjacent cubic Bezier curves will be "smoothly joined". That means:

- The "right" endpoint of one curve must also be the "left" endpoint of the adjacent curve—a common anchor point.
- The common anchor point and the two associated control points must lie on a straight line.

It is not necessary that the two control vectors have the same length. (We see this for control vectors P3-P2 and P3-P4 in figure 8). In many cases, the program will initially establish the spline so that the paired control vectors at each endpoint are initially equal in length. (I had to adjust these for the example!)

Often graphic programs will have keyboard shortcuts that, if we move one such control endpoint, will force the "mated" one to move so that the straight line relationship is maintained. (I used that!)

Cusps

Suppose that at a joint between two segments, the two control point do not quite lie along a common line? In that case, the joint is to some extent "pointy"—it forms what a mathematician calls a *cusp*. Mathematically, it means that the second derivative of the function that describes the curve has a discontinuity at that point—that it, it changes suddenly.

Corners

Imagine the case where our overall curve is closed, and in fact turns out to be precisely a rectangle. How does that fit into this picture?

Well, for one thing, the segments now are line segments, which turn out to be linear Bézier curves. In the usual situation, their control points lie along the two segments, not along a common line (as required for a smooth joint). The result is often described as a *corner*. (In reality, the controls points probably lie at the anchor points, but the explanation is clearer if we consider them to lie along the segments—that way we can cleanly say “they do not lie along the same line”.)

We can see that a *corner* is just what we call a *cusp* when it is at the joint between to line segments.

BÉZIER CHIROPRACTIC—JOINT MANIPULATION

One of the common “tool functions” in program provisions for editing a Bezier spline deals with the matter of the joints. If we have a smooth joint, and hit it with a tool that is identified as “make a corner”, the two control points are drawn back to the joint itself—to the anchor point. The result is that there will be a *cusp* at that joint—we may consider it a corner.

Conversely, if we have a joint with a cusp, and hit it with a tool that is identified as “make a symmetrical curve”, the result is that the two control points are “moved out” from the anchor point along a common line, satisfying the requirement for a smooth joint.

Note that here is no unique “correct answer” as to exactly how far out the control points are put and exactly what will be the orientation of that common line. These parameters are chosen by an algorithm chosen by the program designer to give a “nice looking joint” over a range of situations.

CIRCLES

Can we make a perfect circle as a closed cubic Bézier spline? No, but we can make a very close approximation. The recipe is shown in figure 9.

The figure is made of a closed spline comprising four cubic Bézier curves. We see all four endpoints (one common to each adjacent pairs of curves), and the interior control points for one curve. For each curve, the control vectors are tangent to the intended circle and should have a length of:

$$\frac{4\left(\sqrt{2}-1\right)}{3}r \qquad (0.55228\ldots r)$$

where r is the intended radius of the “circle”.

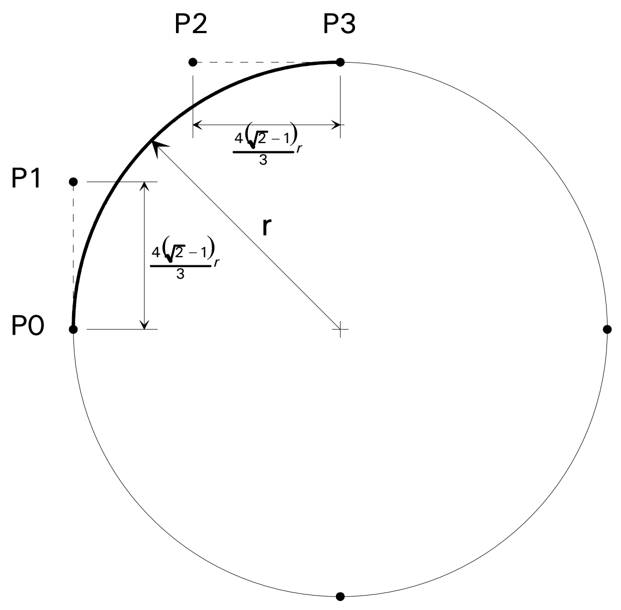


Figure 9. Cubic Bézier spline approximation of a circle

It has been said that the departure from a perfect circle on a radial basis is less than one part in a thousand.

Does that mean that the "circles" in technical illustration and CAD programs are imperfect? Not usually. The program generally has a special circle item that is defined in terms of radius and the location of the center, and is "perfect"—not as a closed Bézier spline. There are such specially-defined figures for other geometric shapes.

But if we wish to change the shape of such a special figure, we may need to convert it to a cubic Bézier spline, which the program will generally do for us gladly (often the command is something like "convert to curves"). (The program may in fact just do that without our asking if we choose "modify as a Bézier curve" in the toolbox.

PATHS AND STROKES

Paths

A Bézier curve has zero width, and thus cannot actually be "exhibited" (shown on a screen or printed page). This is of course equally true of a straight line segment (actually a special type of Bézier curve, eh?), or a geometric figure such as a rectangle or a circle.

Of course, most graphic illustration programs do us the favor of displaying or printing a zero-width figure as a string of pixels of some arbitrary width (typically a fixed pixel width regardless of the scale at which the figure is viewed or printed) so we can see it.

In order to emphasize one aspect of such "zero-width, theoretically-invisible" figures, many illustration programs refer to them (including Bézier curves and Bézier splines) as *paths*.

Stroke!

Often we will want as a design element (perhaps a stripe across a control panel, the mouth of a smiley face, or a highly-visible line around the title block of an engineering drawing) a line or curve with a finite[2] width.

Many illustration programs describe this as making a *stroke* of finite width along a path (which could well be a Bézier curve). The word stroke is meant to be evocative of a stroke with a drafting pen. And the path can be thought of as the path along which the pen is moved.

Other programs more pragmatically speak of having a line/curve of non-zero width.

Often these programs allow various options regarding strokes (or stroked paths, as they are sometimes called by the truly fastidious), such as:

- Having the end be
    - Straight across at the endpoint of the path
    - Straight across but beyond the endpoint of the path by half the width of the stroke
    - A semicircular curve with radius equal to half the stroke width and centered on the path endpoint
- Where two or more curves join (not "smoothly"), having the joint take on one of several forms (often rather comparable to the variations for a stroke end, listed just above)
- Having the stoke exhibit a pattern (such as "dotted" or "dashed')
- Having the stroke be of a certain color

Note that none of these are aspects or properties of the Bézier curve itself. (And of course, the stroke itself is not a Bézier curve, but rather a stroke along a Bézier curve.)

Figure 10 shows the path of the Bézier spline of figure 8 stroked. In the lower copy, the path itself, and its anchor points, have been overlaid in white for reference. In this case, the "round end" treatment was elected for the stroke.

---

[2] In the accepted sense (one of many) of "neither zero, infinitesimal, nor infinite".

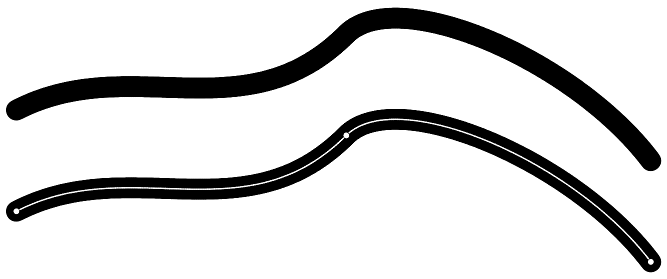


Figure 10. Stroked open Bézier spline

ACKNOWLEDGEMENTS

Thanks to Don Lancaster for his excellent presentation and derivation of the equations presented above, found in his paper cited above; to John Blasquez for his helpful comments on an earlier issue of the manuscript; and to Carla Kerr for her expert copy editing of the first edition.

#