Celtic Knotwork, Part I

Andrew Glassner

Microsoft Research

Celtic knotwork is a beautiful form of ornament. The elegance of the design, the intricacy and subtlety of the work, and the precision of the craftsmanship all combine to form a visual treasure. Knotwork probably dates from the 6th century, when Irish monks illuminated religious texts with elaborate, impeccably rendered letters and imagery. Unfortunately, very few of these manuscripts have survived.

Amazingly, nobody knows how ancient Celtic knotwork was designed or executed. The methods were probably secret, kept within the societies of religious scholars and monks that worked on the manuscripts. Perhaps they were never even written down. When the practice of creating original work in this style of illumination ended, probably in the 10th century, the techniques for that practice were lost as well.

As a result, for centuries art teachers and historians considered it impossible to create new designs and taught that the only way to use this style of ornamentation was to copy an existing drawing, stone carving, or piece of jewelry.

That state of affairs would probably persist today except for an art teacher and book illustrator named George Bain. He spent decades studying the great Celtic manuscripts, such as *The Book of Durrow, The Book of Kells,* and *The Lindisfarne Gospels*. From these examples he devised a very simple method of construction, which he published in 1951 (see the "Further Reading" sidebar for more information). He showed that his method could be used to replicate even the most complex examples in the ancient manuscripts, as well as create new and original patterns.

When Bain's book came out, it changed everything—people could create original and beautiful Celtic knotwork for their own projects. Today, Celtic knotwork is a thriving design field.

I discovered Bain's book about 25 years ago, and it made an enormous impression on me. Like many others, I started creating my own work and pushed the boundaries. Since then, I've created original knotwork designs for t-shirts, silkscreen prints, jewelry, and even magazine covers—Figure 1 shows a chemistry-themed knot I drew for the cover of the October 1998 *Communications of the ACM*.

The basic principles of Bain's method are surprisingly algorithmic. Creating a pleasing design is a matter of

Further Reading

The book that taught me about Celtic knotwork is the classic *Celtic Art: The Methods of Construction* by George Bain (William MacLellan and Co., Glasgow, 1951; reprinted by Dover Publications, 1973). Bain's son has produced a practical volume that simplifies some of his father's methods and presents the three-grid system I used in this column: *Celtic Knotwork* by Iain Bain (Sterling Publishing, New York, 1986). Both of these books are terrific.

A very clear, step-by-step explanation of grid-based knotwork construction is available in *Celtic Design: Knotwork, The Secret Method of the Scribes*, by Aidan Meehan (Thames and Hudson, New York, 1991). Meehan has produced a whole set of books on Celtic art, covering key patterns, spiral patterns, zoomorphics, lettering, and more. I recommend them to you highly. Visit Meehan's home page at http://www.geocities.com/~coracle/celtic_design/index.html.

Many years ago I managed to get a copy of a small book that doesn't even have an ISBN number. If you're able to find a copy of it, take a look at *A Handbook of Celtic Ornament* by John G. Merne (Mercier Press, Dublin and Cork, 1974).

You can find wonderful inspiration in the classics. Reprints of the *Book of Kells* are available, as are a few of the other classic manuscripts. My copy is *The Book of Kells, Described by Sir Edward Sullivan* by Edward Sullivan (Studio Editions, London, 1986). Sullivan tells the tragic story of how the pages of this precious, unique manuscript were mutilated by an unknown bookbinder about a century ago.

You can also find some modern interpretations of Celtic artwork of different forms in *The Celtic Art Source Book*, by Courtney Davis (Blandford Press, New York, 1988).

aesthetics, but the mechanics of the process fit naturally into the framework of a computer program. In this column, I'll describe Bain's construction method (as refined by his son), talk about how to write a program to implement it, discuss some ways to improve the output, and show some results.

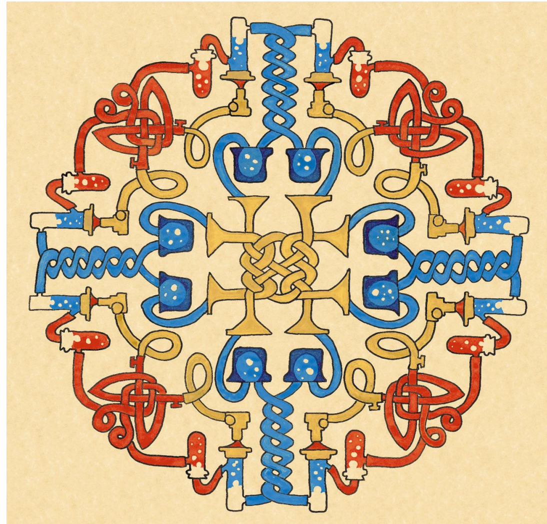
My approach will be to use the computer to develop a sketch, which then serves as the basis for a hand drawing. You can do lots of things on the computer, but getting the particular curves and patterns in a design to look just right is easier with pen and ink than mouse and keyboard. Any minor imperfections are part of the beauty of the piece.

The grid method

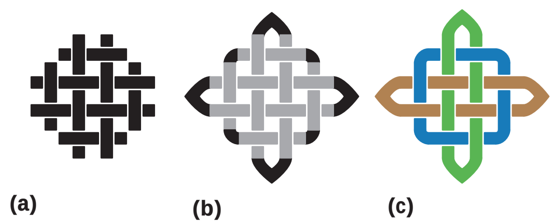
Iain Bain refined his father's approach into what he called the "three-grid technique," the method I'll present here for creating knotwork panels.

Figure 2 shows the basic idea. As shown in Figure 2a, you begin with a *plait*, or a regular weaving of perpendicular braids. Note that in a square pattern like this, the bands—sometimes called *cords*—overlap and underlap in perfect harmony. Each band goes over then under in perfect alternation. I call this the *internal weaving*. In Figure 2b I added the *external weaving*, the joining up of neighboring ends of the bands. In Figure 2c I colored the result. You can see that I created a pattern with three distinct, interwoven bands. Traditional Celtic knotwork usually had only a single band in the final pattern (though plenty of exceptions exist). I like one-band designs, but I also like making patterns with a small number of distinct bands, so in this column you'll see examples of both.

Rather than explain the process in detail, I'll present the three-grid technique with visual examples. Figure 3 shows the basic steps in creating a 2-by-3 knotwork panel. The primary grid in Figure 3a determines the overall shape of the pattern. Here it's a pattern of red dots creating a 2-by-3 grid (note that though the dots are themselves 3-by-4, the pattern is named for the 2-by-3 grid of squares for which they form the corners). Then I created a secondary grid in Figure 3b by placing a blue dot in the center of each of these red squares. Drawing the lines defined by these two sets of overlapping grids creates the tertiary grid, shown in Figure 3c. Here I marked in gray the internal region where the plait will be drawn. In Figure 3d I added the lines of the plait. The easy way to make these is to start with diagonals in the corners of the plait; the other lines are marked off along every other dot on the tertiary grid. In Figure 3e I added the external weaving. The sides are joined naturally and loops are placed at the corners. In Figure 3f I marked the alternating pattern of overlaps. Once you choose one intersection as an overlap or underlap, all the others are determined—you just need to work your way around marking them off in alternation.



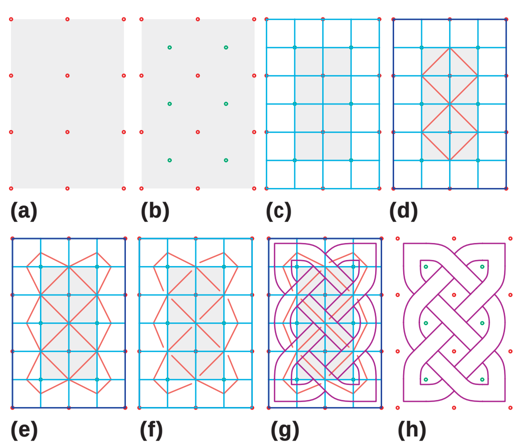
1 A chemistry-themed Celtic knot.



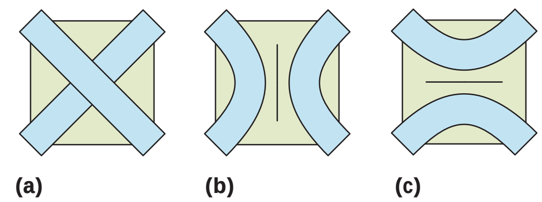
2 The basic gridwork knot. (a) The internal weaving, in this case a plait. (b) The external weaving. (c) The colored version with three bands.

Figure 3f is what I call the *skeleton*—the line with all internal and external weaves and the overlap pattern marked. Once you have the skeleton, the mechanics of the job are all done. But it's still nice to draw the more traditional band, so I show that in Figure 3g (I'll talk about how to draw the band below). Figure 3h shows the final band with everything but the primary and secondary dots removed. Notice that this example shows a single continuous band.

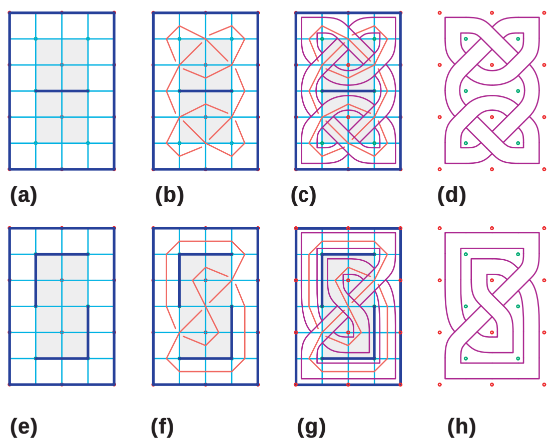
With this technique we can make a knotwork panel of any resolution—just lay out the primary grid of any size and resolution you want. Bain stated that if the number of horizontal and vertical cells in the primary grid (in this case, 2 and 3) are relatively prime, the result will be a single band. If the numbers have a com-



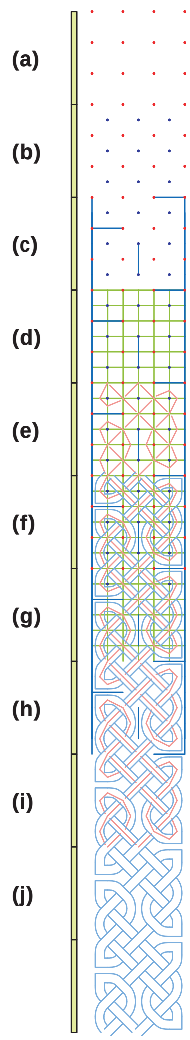
3 The three-grid technique. (a) The 2-by-3 primary grid (red dots). (b) The secondary grid (blue dots). (c) The tertiary grid (cyan lines). The gray region in the center indicates the region of internal weaving. (d) The plait (diagonal red lines). (e) Adding the external weaving. (f) Identifying the overlap pattern. (g) Adding the band. (h) The final knot.



4 (a) An intersection. (b) The result of a vertical breakline. (c) A horizontal breakline.



5 Introducing breaklines. (a) A breakline between two primary grid points. (b) The resulting skeleton. (c) The band. (d) The result. (e) Four breaklines, each between two primary grid points. (f) The skeleton. (g) The band. (h) The result.



6 The construction process for a border piece. The yellow band at the left marks off the repeating 4-by-4 primary grid. (a) The primary grid (red dots). (b) Adding the secondary grid (blue dots). (c) Adding three breaklines in blue (and breaklines along the left and right sides of the border for consistency). (d) The tertiary grid (yellow-green lines). (e) The skeleton. (f) The band. (g) Removing the grid dots. (h) Removing the tertiary grid. (i) Removing the breaklines. (j) Removing the skeleton, leaving the final border to repeat as needed.

mon factor, you'll get several bands.

We can make the knot more interesting by introducing *breaklines*. Basically a breakline re-routes an intersection. Figure 4a shows a typical intersection from inside a plait. In Figure 4b I introduced a vertical breakline. The skeleton isn't allowed to cross the breakline, so the four ends are re-routed as shown. Figure 4c shows a horizontal breakline.

Figure 5 shows the result of introducing some breaklines into the plait of Figure 3. You'll notice that even in these simple examples, the knots are generally symmetrical, a characteristic of most Celtic knotwork.

Breaklines can be drawn between two dots of either grid. Figure 6 shows the construction process for a *border element*, a piece of knotwork that can join up to itself to repeat as many times as necessary. Here I've drawn in breaklines along the sides of the border to indicate that I don't want the skeleton to escape the primary grid. Now we don't have to treat the internal and external weavings as different things—simply start weaving and avoid the breaklines.

Like everything else, breaklines have rules. They can't cross. They can only join together horizontal or vertical neighbors of the same grid—you can't connect a primary grid dot with a secondary. And you can't join up a dot with a distant neighbor—the breakline can only go horizontally or vertically, to one of the four closest neighbors on the same grid.

Figure 7 shows a more complex example worked with primary and secondary breaklines in a 5-by-4 panel. The trick with breaklines is to use them judiciously—a single breakline can change the nature of an entire piece. You'll also find that sometimes breaklines will increase or decrease the number of bands in your design. If you have a primary grid of x by y cells, you can create up to xy bands. Of course, the minimum number of bands is 1.

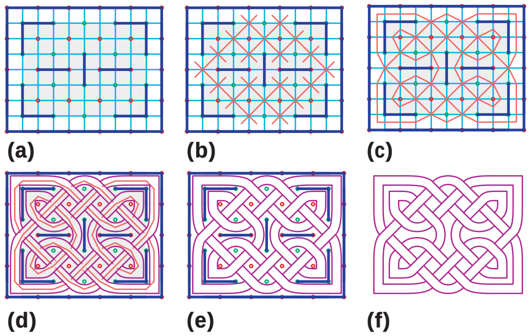
I hope these examples are enough to get you started drawing your own designs by hand. For lots more discussion of this approach and other, related techniques, see the "Further Reading" sidebar.

Programming

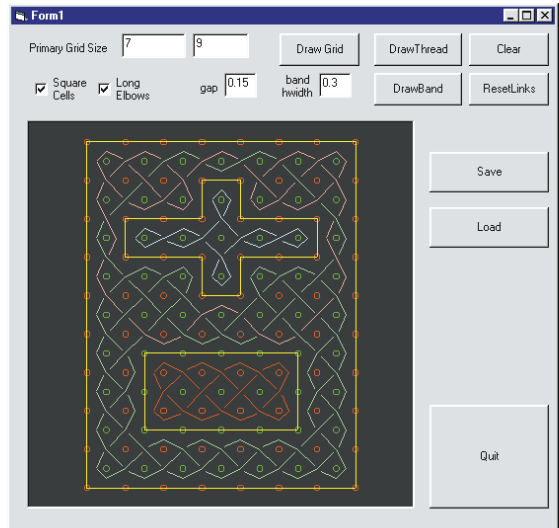
Writing a Celtic knot assistant is a fun programming project. I put mine together over a weekend in about 1,500 lines of highly unoptimized Visual Basic code. Figure 8 shows a screenshot of the program in action. It writes two output files—one for the skeleton and grids, and one for the final band—both in Postscript format. I created all the line and band knotwork figures in this column with this program. In this section I'll describe the basic outline of the code, in case you'd like to write your own.

My user interface is minimal (after all, this was just for fun). You can enter the size of the primary grid, draw breaklines, set the size of the overlap gap in the skeleton, and assign the thickness of the rendered band. You can save and recall your favorite knots. You can also draw and erase breaklines, and immediately see the resulting skeleton. If you have more than one band in the skeleton, they're drawn with different colors.

First, define the grid and breaklines. If the primary grid is x by y, then create a $2x$ by $2y$ data structure, represent-



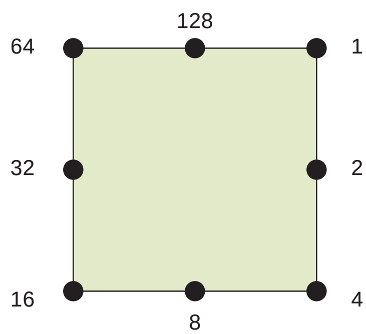
7 **Breaklines in a more complex panel. (a) A 5-by-4 primary grid with the secondary and tertiary grids and breaklines. (b) The plait constrained by the breaklines. (c) Joining up the skeleton. (d) The band. (e) Removal of the skeleton. (f) The result.**



8 **My Visual Basic knot design assistant.**

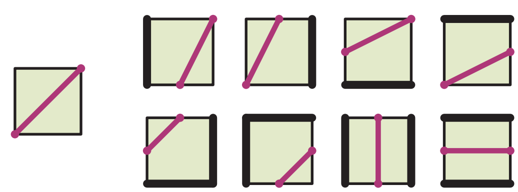
ing the tertiary grid. I call each box in this grid a *cell*. The first element in each cell's data structure is a breakline identifier, which tells me whether the upper left corner of this cell has no breaklines, a line to the right, a line going down, or both. This convention means that when I draw a breakline in the interface, it actually has to be added to two cells. When creating a new primary grid, I automatically add breaklines around the perimeter.

The cell data structure contains an all-purpose *visited* flag that I use for various bookkeeping jobs (outlined below), a *bandnumber* field, and an *edgecode* field. Figure 9 shows a cell and the values I assigned to the eight places a line can touch the cell (the four corners and the four sides). The *edgecode* is the logical OR of the codes for the appropriate points. For example, if the skeleton goes from the lower left corner to the right side, the code is $00010010_2 = 18_{10}$.



9 **Codes for the position of the skeleton endpoints in the cell.**

To draw the skeleton, the first thing I do is run through every cell in the tertiary grid and determine its edgecode. In the absence of any breaklines, the skeleton in the upper left cell will want to run from the lower left to the upper right corner. So I check the breaklines that might touch this cell and adjust the diagonal accordingly. Figure 10 shows the default diagonal and the eight possible alternatives the breaklines could re-route it into. Note that because this is a tertiary grid cell, and breaklines can't intersect, there can't be more than two breaklines per cell.



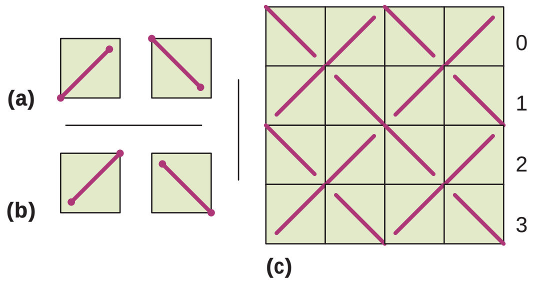
10 **Assigning codes to a diagonal in the presence of breaklines (drawn as thick lines).**

Once I label this cell, I move to the right. Now this diagonal runs the other way, from the upper left to the lower right. I label it using the mirror image of Figure 10. Then I move right again, flip the direction of the diagonal, and so on. When I've finished with the row I start the next one. The starting (left-most) cell for each even row runs from lower left to upper right, and the other way for each odd row.

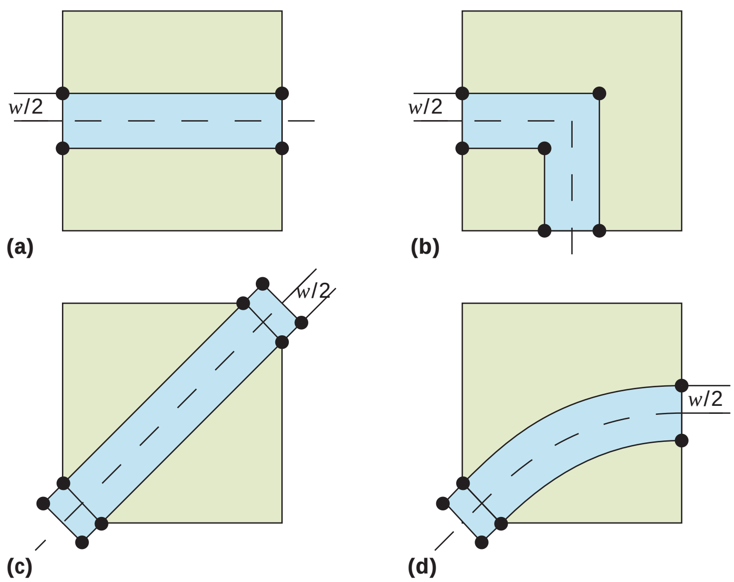
Once I've marked all the cells with edgecodes, I start labeling each cell with the band it belongs to by setting the *bandnumber* field. First, I run through every cell and set its *visited* flag to *false*. Then I set the current band number to 0. Now I enter a loop. At the top of the loop I simply scan all the cells, looking for one that hasn't been visited. If I find one, I follow its skeleton all the way around, setting the *visited* flag in each cell I walk through to *true* and the *bandnumber* field to the current band number. Finally, if I did find an unvisited cell, I increment the band number, go to the top of the loop, and look for another one. When the scan says all the cells have been visited, I'm done labeling the bands.

Following the skeleton is straightforward. Given a cell, I mark it as visited, then examine its edgecode, starting in the upper right corner and going around clockwise. Anytime I find a flagged position, I recursively call the tracking routine. The argument to the routine is the cell in the indicated direction. Since no cell can have more than one band, when I hit a cell that's been visited, I return from the tracking procedure. When the recursion is done, I've hit every cell on the band and marked it *visited* and assigned it a band number.

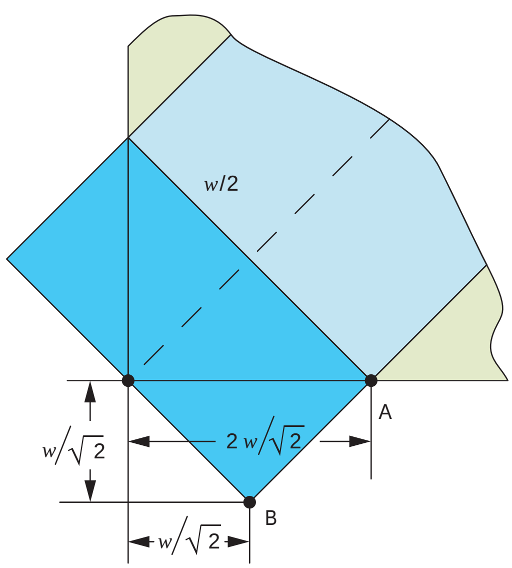
Drawing the skeleton is also straightforward—just draw a line that joins the edgecodes. The only remaining step is the overlapping. It turns out that on a rec-



11 Including overlaps. (a) Diagonals for even rows. (b) Diagonals for odd rows. (c) A few short rows assembled.



12 The four basic cell types. (a) Bar. (b) Corner. (c) Diagonal. (d) Elbow.



13 Geometry for the foot of the diagonal and elbow. The darker blue region indicates the foot.

tangular grid you can determine which way the overlap goes very easily. On even rows, the bands running northeast overlap in the lower left and underlap in the upper right. Bands running northwest overlap in the upper left and underlap in the lower right. On odd rows, it's reversed. Figure 11 summarizes this technique. This holds not just for diagonals, but also the pieces that run from a diagonal to a side. Any time you're trying to draw into a corner, this rule lets you know whether to go all the way to the corner (resulting in an overlap) or to stop short (resulting in an underlap).

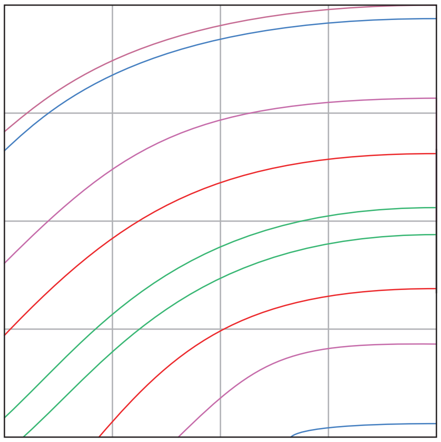
Drawing the bands is just a little bit trickier because we want them to look good. Keep in mind that I'm deliberately keeping the quality bar representing "good" pretty low here, because these bands really only provide a guide for a hand drawing. I'll detour for a moment to talk about the geometry of the band in the cells, then get back to the programming.

Figure 12 shows the four basic kinds of cells. I call these *bars*, *corners*, *diagonals*, and *elbows*. I decided early on that I wanted to control the thickness of the band from the user interface, so each of these four types has to be parameterized by thickness. Let's call the width of the band w. As we saw for the skeleton, the only question for controlling overlaps is whether we draw the little "foot" in the appropriate corners of the diagonal or one corner of the elbow. Figure 13 shows the geometry for finding the corners of the foot (in darker blue) or the corners of the shortened band (in lighter blue). The other four corners of the cell are symmetrical. If we draw the foot, then line segment AB is included in the line going to the upper right. Otherwise, we just start at B.

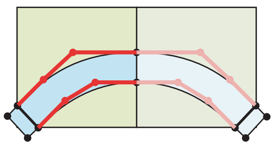
The geometry for the bar and corner is pretty easy, and we've just seen the diagonal. How about the elbow? You could approximate the lines of the elbow with a 45-degree circular arc and a straight line. But that looked a little too clunky to me, so I decided instead to use Bezier curves. My choice was motivated by the fact that they're very easy to specify and control, and they're native to Postscript. If you're not familiar with Bezier curves, all you need to know is that you need two endpoints, A and D, and two intermediate points, B and C. The curve starts at A, heads toward B for a while, then arcs toward C, and finally ends up at D. The essential bit is that it leaves A as though heading to B. It arrives at D as though coming from C. For the elbow, I want the curve leaving the lower left corner to head up at a 45-degree angle, so I just place B somewhere along the diagonal coming out of A. Similarly, C is directly to the left of D. The distances from A to B and C to D determine how much the intermediate points influence the curve's shape.

Just one curve doesn't do the trick because big, fat bands look different than thin ones. So I drew a family of five different curves, shown in Figure 14. Table 1 gives the data, so you don't have to measure the curves. The way to read this is to imagine that the elbow sits in a 32-by-32 grid. Points A and D lie at the endpoints of the elbow, while the table provides points B and C.

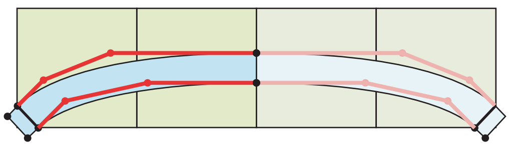
I labeled the curves from 0 to 9, in pairs. Pair 0,9 corresponds to the thickest, fattest curve, while pair 4,5 describes a very thin band. Since the grid is 32 units on



14 My curves for pleasing-looking elbows.

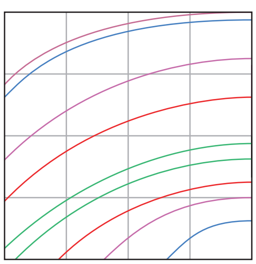


(a)



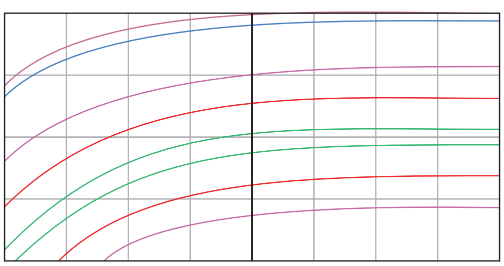
(b)

15 (a) The short arc. (b) The long arc.



(a)

16 (a) Curves for the short arc. (b) Curves for the long arc.



(b)

a side, the thickest band is $32/\sqrt{2}$, or about 22.5 units. Pair 0,9 is designed for this thickest band. Pair 1,8 is designed for 15 units, 2,7 for 9 units, 3,6 for 5 units, and 4,5 for 2 units. Given a band thickness, I find the two curves that capture it and interpolate the values B and C. Point A always lies at $(2w/\sqrt{2},0)$, and point D at $(32, C_y)$, since D lies directly to the right of C.

Table 1. Coordinates for normal elbows.

Curves	(B_x, B_y)	(C_x, C_y)
a0	(24, 0)	(30, 0)
a1	(24, 3)	(26, 5)
a2	(17, 4)	(22, 8)
a3	(15, 8)	(26, 10)
a4	(11, 10)	(23, 13)
a5	(9, 10)	(21, 14)
a6	(11, 19)	(18, 20)
a7	(10, 23)	(23, 25)
a8	(3, 23)	(10, 31)
a9	(3, 25)	(10, 31)

Table 2. Coordinates for short arc curves.

Curves	(B_x, B_y)	(C_x, C_y)
c0	(24, 0)	(29, 0)
c1	(22, 1)	(29, 1)
c2	(19, 6)	(21, 7)
c3	(13, 7)	(18, 11)
c4	(8, 6)	(16, 15)
c5	(7, 8)	(15, 17)
c6	(9, 17)	(16, 21)
c7	(9, 22)	(14, 25)
c8	(4, 25)	(11, 31)
c9	(4, 26)	(11, 32)

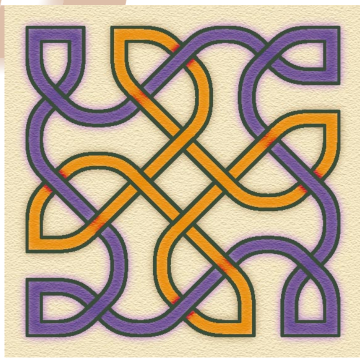
Table 3. Coordinates for long arc curves.

Curves	(B_x, B_y)	(C_x, C_y)
d0	(24, 0)	(60, 0)
d1	(22, 1)	(61, 1)
d2	(21, 8)	(56, 7)
d3	(18, 11)	(36, 11)
d4	(17, 5)	(32, 15)
d5	(17, 19)	(30, 17)
d6	(16, 23)	(40, 21)
d7	(13, 26)	(46, 25)
d8	(11, 32)	(43, 31)
d9	(11, 34)	(43, 32)

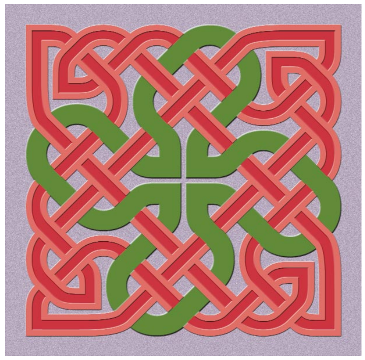
This works pretty well, but some knots still look a little too boxy for my taste. One way to make things better is to smooth some elbows to create *short arcs* and *long arcs* (see Figure 15). A short arc is created when a band crosses an edge joining two elbows. A long arc is created when the long arm of an elbow connects with a bar. Figure 16 shows the curves that I designed for these arcs. Tables 2 and 3 give the data. Notice that the long arc covers two cells.

Returning to the programming, once we have these curves all typed in, things get pretty easy.

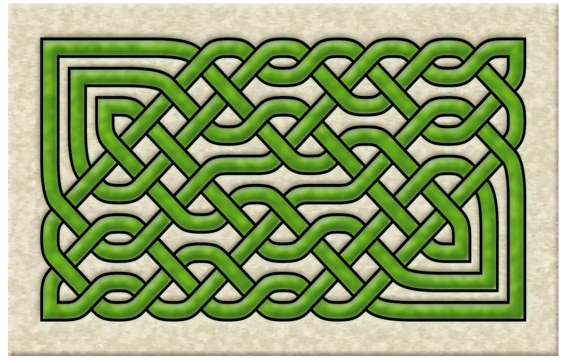
I begin by scanning the entire tertiary grid, again setting the *visited* flag to false. I first scan the whole grid for long arcs. If I find one, I mark both the elbow cell and the bar cell as visited. Then I draw the curves that correspond to them, including the foot if appropriate. Note



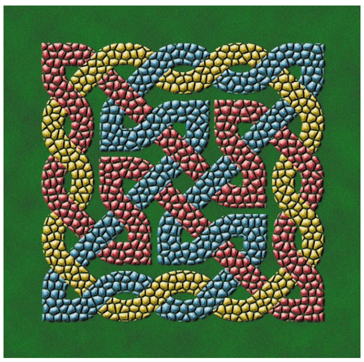
17 **A two-color knot built on a 4-by-4 primary grid.**



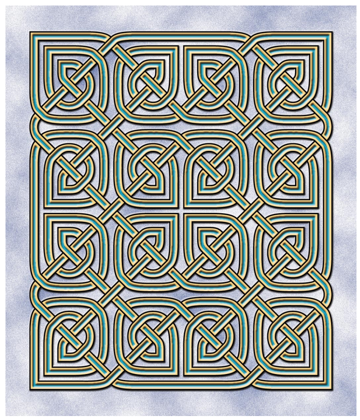
18 **A two-color knot using an internal treatment for the red band.**



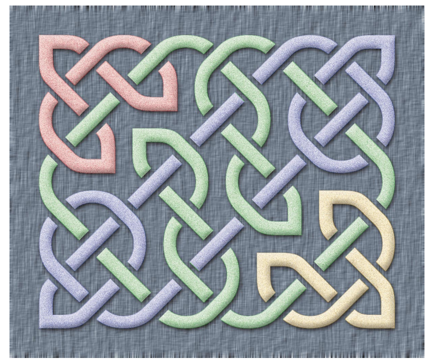
19 **A classically styled rectangular knotwork panel.**



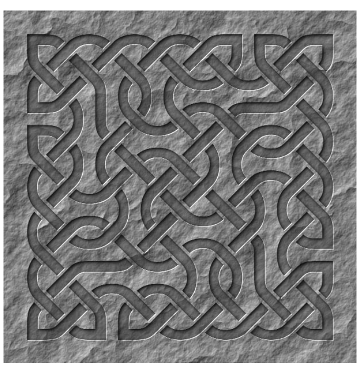
20 **A Celtic garden path.**



21 **Many Celtic knots use spirals as a design element, as in this panel.**



22 **A classically styled four-color knot.**



23 **A classical square panel knot built from a 7-by-7 grid.**

that you have to catch eight orientations. I test for them all explicitly, then call a single routine with flags for rotating and/or reflecting the curves as necessary.

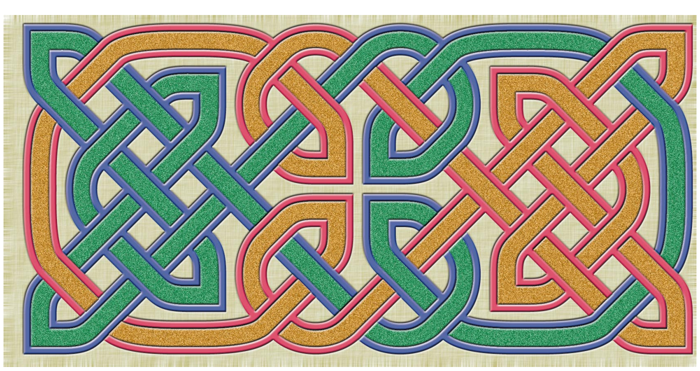
Once I've completed the scan, I run through the whole thing again looking for short arcs. If I find an elbow that hasn't been visited, I check to see if it's half of a short arc. If it is, I draw the short arc and mark both cells. The short arc has only four orientations, since it's symmetrical.

Now I do a third scan, marking and drawing all the cells that haven't already been marked.

In all, this process requires six cell-drawing routines: long arcs, short arcs, elbows, diagonals, bars, and corners. Each one is parameterized by the band width and band colors, and has a bunch of flags for rotating and reflecting the contents.

Examples

You can have a lot of fun designing and decorating your knots. Figures 17 through 24 show several knots in different styles, with different treatments. Although I usually hand draw my knots (such as Figure 1, and other examples on my Web site), these were all output from the Knot Assistant. I imported the Postscript into Adobe Photoshop to give them color and surface treatments. When I design my knots, I always try to balance the traditional nature of the medium with my own sense of aesthetics. There's a deep tradition in Celtic knotwork. I find that the most satisfying works recognize and honor that tradition, while incorporating a personal feeling of design.



24 **A knot design from the *Lindisfarne Gospels*.**

Forward, into the past

Many interesting generalizations exist in a subject as rich and fascinating as this one. Next time we'll push across some of the boundaries that served us so well in this installment. We'll break free of the rectangular grid, talk about snakes, and move out into the third dimension. ■

Readers may contact Glassner by e-mail at glassner@microsoft.com.