Using Bezier Curves for Geometric Transformations

MSM Creative Component

Catherine Kaspar

Fall 2009

Committee:

Dr. Heather Bolles

Dr. Irvin Hentzel

Dr. Alex Andreotti

Table of Contents

Introduction

I chose to investigate Bezier curves with the intent of using them in a calculator program I was working on my first summer in the MSM program. The purpose of my program was to show high school students how geometric transformations could be carried out with matrices. My idea was to use text characters as the object being transformed. I soon ran into problems, however, when I tried to use any characters containing curves. Bezier curves were the solution to my problem.

This paper is divided into three sections. The first contains some general background information on Bezier curves. The second section describes the work I did with Bezier curves, and the third includes a description of how I incorporated them into my original classroom project of transforming fonts with matrices.

Background Information

History

Whether occurring in nature or in the mind of a designer, curves and surfaces that are pleasing to the eye are not necessarily easy to express mathematically. Whatever their origin, the ability to represent and transform complex curvilinear shapes has increased markedly over the last fifty years and has resulted in marked improvements in manufacturing and computer graphics fields among others.

Progress in the area of curve representation stemmed from the need of mechanical engineers in the 1950's and 60's in the automobile and aircraft industries to accurately define freeform shapes. Transferring curves from the drawing board to the pattern shop was inefficient and inaccurate when designs contained anything other than basic lines, circles and parabolas. Although they possessed the hardware that allowed machining of complex three-dimensional shapes, software to communicate the particular specifications was lacking. (Farin, 1988).

To solve this problem, Pierre Bezier, a French engineer working for the automobile company, Renault, used parametrically defined surfaces, expressed with polynomials exhibiting special characteristics, to develop what is now viewed as the beginnings of Computer Aided Geometric Design (CAGD). Unbeknownst to Bezier, Paul De Casteljau had just finished similar work, but because De Casteljau's work wasn't published until later, Bezier's name is associated with these surfaces and curves. Bezier and de Casteljau's insight into how to use these special polynomials to represent curves has had an important impact on the computer graphics field.

Because I am interested in the plane curves used with font design, the rest of this paper will focus on two-dimensional Bezier curves, rather than 3-D Bezier surfaces.

Polynomial Representation of Curves – What Degree?

There are many ways to express a two-dimensional curve mathematically. Starting simply, the coordinates of various points on the curve can be used to create a polyline or first-degree linear approximation to the curve. But if the curvature changes quickly, infinitely many

points may be necessary to achieve a smooth appearance. This situation can create a storage problem for computers.

A more efficient approach is to use higher degree polynomial functions as approximations. Although they are still only approximations, higher degree polynomials take up less storage space and are less unwieldy when it comes to manipulating data.

Cubic polynomials are widely used in font design because they give more flexibility in forming a curve than do lower degree polynomials yet they do not involve excessive computation or introduce the extra squiggles that come with higher degree polynomials (Foley, 1990). Also, cubic polynomials are the lowest degree polynomial that allows a designer to specify two endpoints as well as independently (although indirectly) controlling the derivatives at each endpoint – all together, four unknowns. TrueType fonts are made up of second-degree polynomials where only three unknowns may be specified, i.e. the two endpoints and one intermediate point which control the derivatives at both endpoints simultaneously (Williams, 2007). Thus, TrueType fonts lack the flexibility of Adobe's PostScript fonts, which are derived from cubic polynomials (Plant, 1996). Figure 1 shows two characters drawn with cubic and quadratic polynomial curve segments. While the figures are very similar, 28 points were needed to specify the quadratic form while only 24 points were needed for the cubic version.

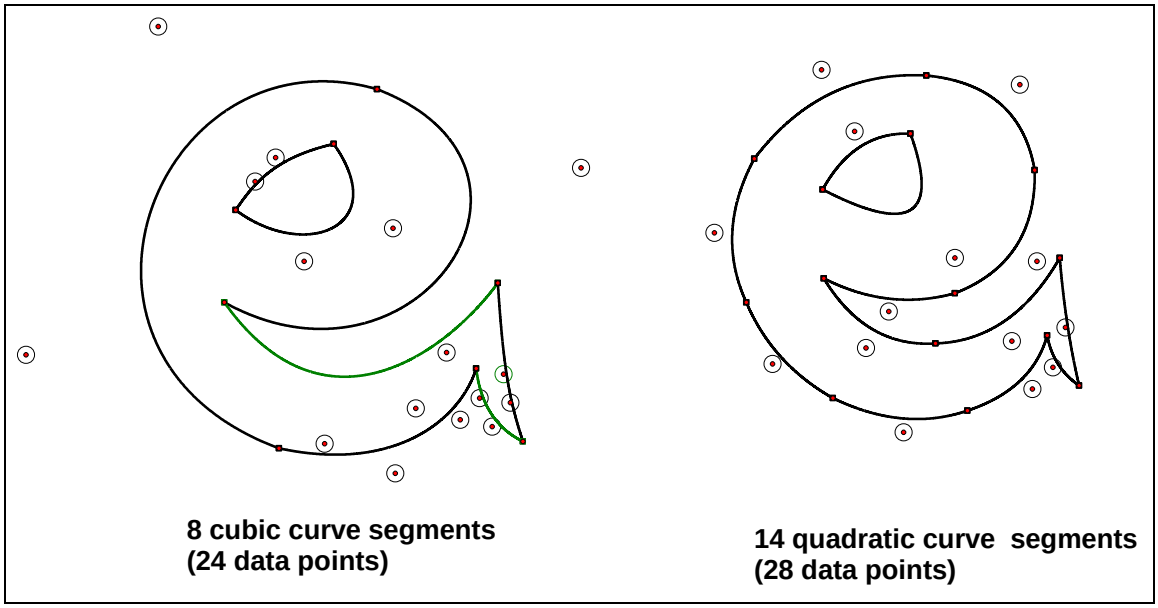


Figure 1 Showing two similar drawings, the left made with cubic curve segments and the right with quadratic curve segments.

Why Parametric Form?

These cubic polynomials can be expressed in three ways: explicitly, implicitly and parametrically (Foley, 1990). When expressing these functions explicitly, i.e. as $y = f(x)$, there are a few problems:

- Some curves such as ellipses, circles etc are not functions so they must be expressed with two equations instead of one,

- Rotation is difficult and may require further breakdown into more segments, and

- Describing curves with vertical tangents causes a problem because the value of their slope is infinity.

If trying to represent curves implicitly, i.e. as $f(x,y) = 0$, it is difficult to restrict the domain, to get only a portion of the circle for example, if that is what is needed. In this form it is also harder to determine if the ends of the two adjoining segments have tangents which are equal in direction to each other. This is often a condition specified to insure smoothness when combining two curve segments.

Both explicit and implicit expressions do have the advantage of easily determining whether or not a given point lies on the curve. In addition, these forms make computations of normals to the curve easy (Foley, 1990).

The third way to represent these curves is parametrically, i.e. as $x = x(t)$ and $y = y(t)$. Parametric form solves the problems listed above for the explicit and implicit forms. Now, geometric slopes, even vertical ones, can be expressed as parametric tangent vectors. Another advantage with parametric form is its ability to represent multiple values of y for a given value of x (a limitation of the explicit or functional form). Figure 2, from Foley, illustrates this well.

The parametric form also allows for easily combining two or more different segments of curves together while meeting certain specifications of continuity. These continuity conditions specify how two segments should be joined. This will be discussed further in a later section.

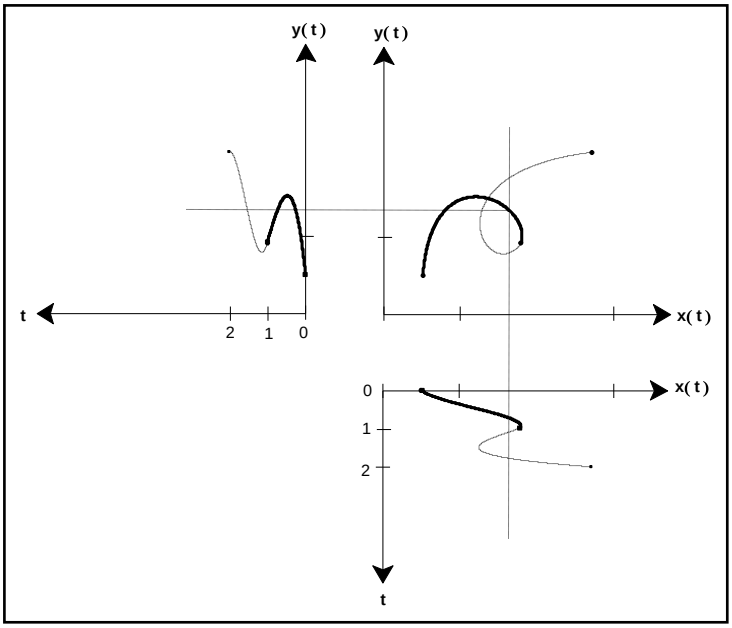


Figure 2. Two joined parametric curve segments (one solid, one dashed) and their defining parametric polynomials. The dotted lines between the *(x,y)* plot and the *x (t)* and *y (t)* plots show the correspondence between the points on the *(x,y)* curve and the defining cubic polynomials. The *x (t)* and *y (t)* plots for the second segment have been translated to begin at *t* =1, rather than at *t* =0.

Bezier Cubic Curves

Although several kinds of parametric cubic curves are used in computer graphics, Bezier curves are the type most often used for font design (Lancaster, 2005). Bezier curves differ from other types of parametric curves by the type of basis polynomials used to form them. The following discussion will explain how these polynomials contribute to the usefulness of Bezier curves.

In general, a cubic curve, expressed parametrically as *Q(t)*, is of the form:

$$Q(t) = x(t)\, y(t)\ , \text{ where} \tag{1}$$

$$x(t) = At^3 + Bt^2 + Ct + D \tag{2}$$

$$y(t) = Et^3 + Ft^2 + Gt + H, \quad 0 \le t \le 1 \tag{3}$$

For Bezier cubic curves, the coefficients A through H (more on these later) are determined by the x and y coordinates of four control points, P_0 , P_1 , P_2 , and P_3 , that are used to specify the curve. These control points and the curve specified by them, are shown in Figure 3.

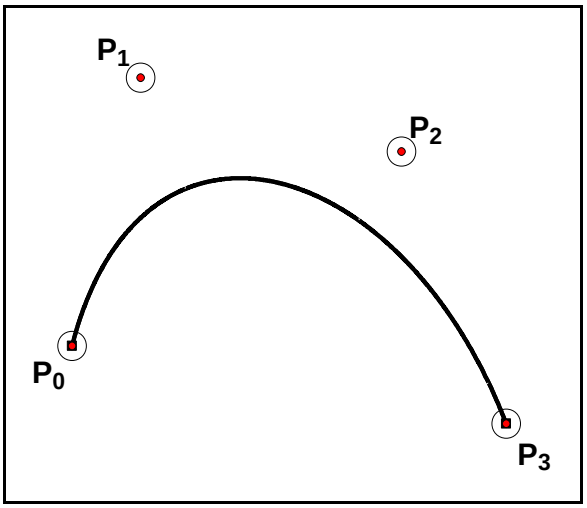


Figure 3 The four control points, P_0, , P_1, , P_2 , and P_3 and their resulting parametric cubic curve.

If the basis function idea is used to represent Bezier curves, special polynomials called Bernstein polynomials, denoted $B_i(t)$, serve as the basis functions and the four control points as the weighting coefficients. With this idea then, *Q(t), x(t)* and *y(t)* can be expressed as:

$$Q(t) = B_0(t)P_0 + B_1(t)P_1 + B_2(t)P_2 + B_3(t)P_3 \quad (4)$$

$$x(t) = B_0(t)x_0 + B_1(t)x_1 + B_2(t)x_2 + B_3(t)x_3 \quad (5)$$

$$y(t) = B_0(t)y_0 + B_1(t)y_1 + B_2(t)y_2 + B_3(t)y_3 \quad (6)$$

The cubic form of these Bezier curves is built up from the first four Bernstein polynomials. These four Bernstein polynomials are defined as follows:

$$B_0(t) = (1-t)^3 \quad (7)$$

$$B_1(t) = 3t(1-t)^2 \quad (8)$$

$$B_2(t) = 3t^2(1-t) \quad (9)$$

$$B_3(t) = t^3 \quad (10)$$

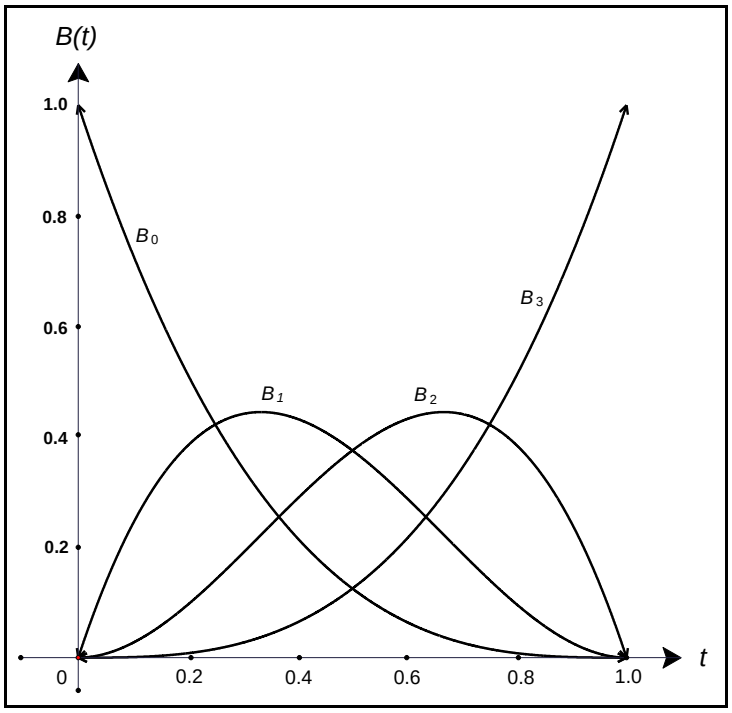


Figure 4 Variation of Bernstein polynomials with parameter, *t*.

Figure 4 shows these Bernstein basis polynomials, plotted as a function of *t*, which goes from 0 to 1. One important characteristic of these polynomials is that for every value of *t* in the interval from 0 to 1, the sum of the functions evaluated at *t* is 1. The graph, taken together with Eq.4, shows the relative importance of the four control points as *t* varies. At *t* = 0, only $B_0(t)$ is nonzero, so only the point P_0 has influence on the curve segment, Q(*t*). At *t* = 1/3 and 2/3, the points P_1 and P_2, respectively, exert the most influence on Q(*t*). Finally, at *t* = 1, the point P_3 takes over while the influence of the other control points disappears. In essence, P_0 and P_3 set the starting and ending points, respectively, of the curve segment. Once P_0 is established, the control point P_1 sets the direction of the curve as it leaves P_0. Similarly, control point P_2, along with P_3, sets the direction of the curve as it nears the ending point, P_3. Thus, with four control points it is possible to independently control the starting and ending directions of the curve (FontForge, 2007). This ability makes a cubic Bezier curve a very flexible and efficient design tool.

Substituting these Bernstein polynomials into Eq. (4), yields the following form for describing the curve segment, Q(*t*):

$$Q(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3 \qquad (11)$$

In terms of *x* and *y* coordinates of the control points (from Eq. (5) and (6)) then:

$$x(t) = (1-t)^3 x_0 + 3t(1-t)^2 x_1 + 3t^2(1-t)x_2 + t^3 x_3 \quad (12)$$

$$y(t) = (1-t)^3 y_0 + 3t(1-t)^2 y_1 + 3t^2(1-t)y_2 + t^3 y_3 \quad (13)$$

If Eq. (12) is expanded, the following expression is obtained for x(*t*).

$$x(t) = x_0(-t^3 + 3t^2 - 3t + 1) + x_1(3t^3 - 6t^2 + 3t) + x_2(-3t^3 + 3t^2) + x_3(t^3) \quad (14)$$

which, when equated to Eq. (2), yields the coefficients A through D in terms of the *x* coordinates of the control points. Gathering like powers of *t* gives the following expressions for each coefficient:

$$A = - x_0 + 3(x_1) - 3(x_2) + x_3 \quad (15)$$

$$B = 3(x_2) - 6(x_1) + 3(x_0) \quad (16)$$

$$C = 3(x_1) - 3(x_0) \quad (17)$$

$$D = x_0 \quad (18)$$

Using the same process for Eq. (13), (6) and (3), in terms of *y(t)*, yields:

$$E = - y_0 + 3(y_1) - 3(y_2) + y_3 \quad (19)$$

$$F = 3(y_2) - 6(y_1) + 3(y_0) \quad (20)$$

$$G = 3(y_1) - 3(y_0) \quad (21)$$

$$H = y_0 \quad (22)$$

Reworking these relationships allows us to also express each of the control point coordinates in terms of its cubic Bezier curve coefficients, A through H.

P_0: $x_0 = D$ $y_0 = H$ (23a,b)

P_1: $x_1 = D + C/3$ $y_1 = H + G/3$ (24a,b)

P_2: $x_2 = D + 2C/3 + B/3$ $y_2 = H + 2G/3 + F/3$ (25a,b)

P_3: $x_3 = D + C + B + A$ $y_3 = H + G + F + E$ (26a,b)

A Geometric View

Several properties of Bezier curves make them particularly well-suited for interactive design environments. (Mortenson, 1997) Three of these properties are endpoint interpolation, geometric continuity and the convex hull property. Two or more Bezier curve segments of like degree can be easily joined together to form what is termed a spline. When curve segments are joined, the starting point of the second curve segment coincides with the ending point of the first curve segment (Figure 5).

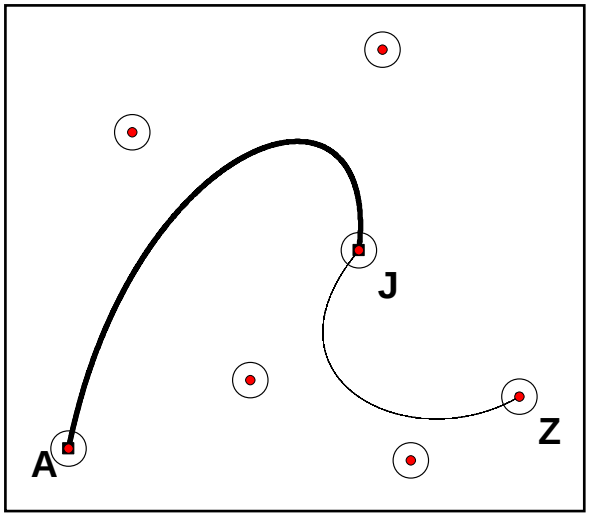


Figure 5 Spline **AZ**: Junction point **J** is the ending point of curve segment **AJ** and the starting point of curve segment **JZ**.

Because Bezier curves actually interpolate their endpoints, a designer knows that matching endpoints will result in a joined curve which is continuous. This is termed geometric continuity of order zero or G^0. In addition to these starting and ending points, cubic Bezier curve segments have two additional control points between the starting and ending points. In these cases, the tangent vector formed by the starting point and first control point is tangent to the curve as it leaves the starting point. Likewise, the tangent vector formed by the ending point and the point preceding it, determines the direction of the curve as it enters the ending point. Aligning both adjacent control points and the junction so that they are collinear will result in a smooth transition between curves, or geometric continuity of order one, designated G^1. See Figure 6.

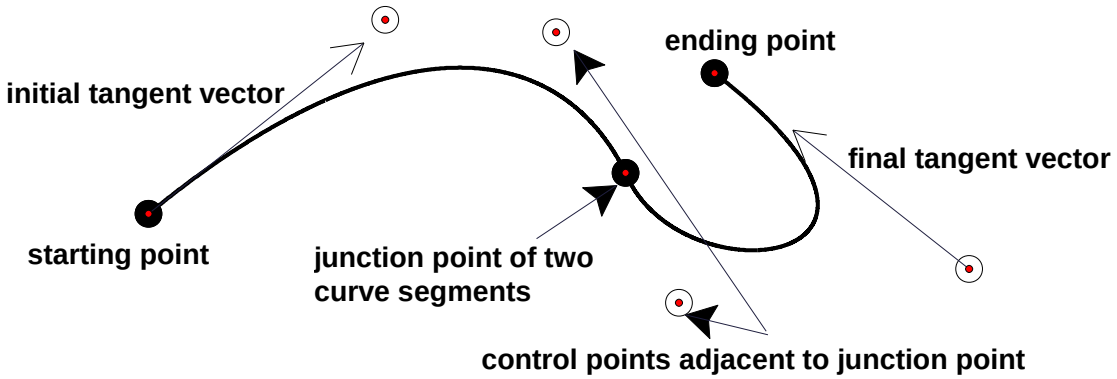


Figure 6 A cubic spline consisting of two cubic Bezier curve segments. Note the alignment of the control points on either side of the junction point. This results in a smooth transition between the two curves.

Another type of continuity, parametric continuity, or continuity with respect to the parameter, *t*, can also be calculated for these curves. This type of continuity, with respect to the parameter, *t*,

becomes important for animation applications where a changing viewpoint (a moving camera, for instance) is required without any abrupt changes in velocity and acceleration. For looking at curves in terms of font design though, this type of continuity is not important.

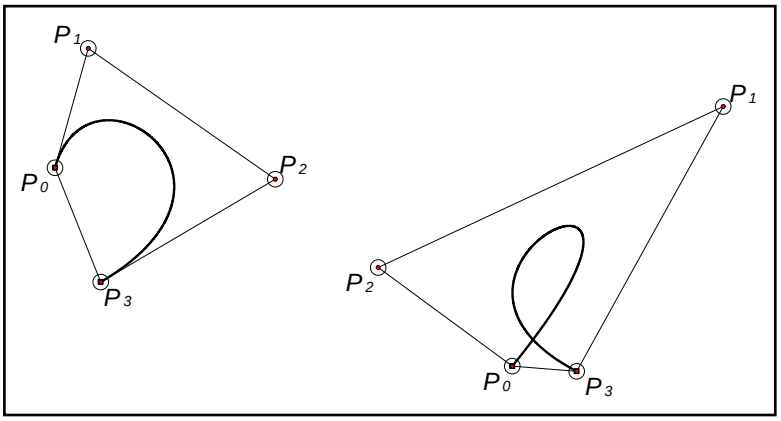


Figure 7 Two cubic Bezier curves and the convex hulls created by their four control points.

Lastly, the convex hull property contributes to the ease in working with these curves by insuring that the curve resulting from the four control points always stays within the convex polygon or hull created by the four control points (Farin, 1997). See Figure 7. Thus, four control points can be intuitively placed to mirror the general shape of the curve desired, the actual curve generated should only require small adjustments of the four control points to match the curve shape intended. There are many applets available that demonstrate this property; one of these is: http://www.doc.ic.ac.uk/~dfg/AndysSplineTutorial/Beziers.html.

Bezier Cubic Curves as Repeated Interpolation

No matter what degree, Bezier curves can be viewed as a series of linear interpolations. In their most basic form, as the parameter, t, goes from 0 to 1, a path traced from the starting point, P_0, to the ending point, P_1, is a line. The point will always be a distance *t,* from the starting point, and distance (l – *t*), from the ending point. This is shown in Figure 8.

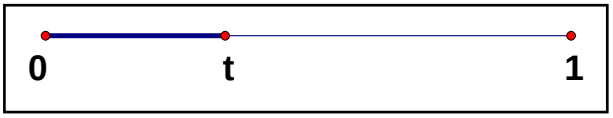


Figure 8 Linear interpolation using parameter, *t*, and the interval from 0 to 1.

Any geometric transformation of these two points will also produce a point at the same distance, *t*, along its path. The result is that for any transformation, the ratio of *t* is always preserved (Farin, 1997). This is the reason that Bezier curves are so useful in an interactive environment. No matter how the initial data set is transformed, the ratio *t*, for any point along the path, is preserved.

The same holds true when two linear parametric curves are interpolated to produce a quadratic Bezier curve. Figure 9 shows this construction.

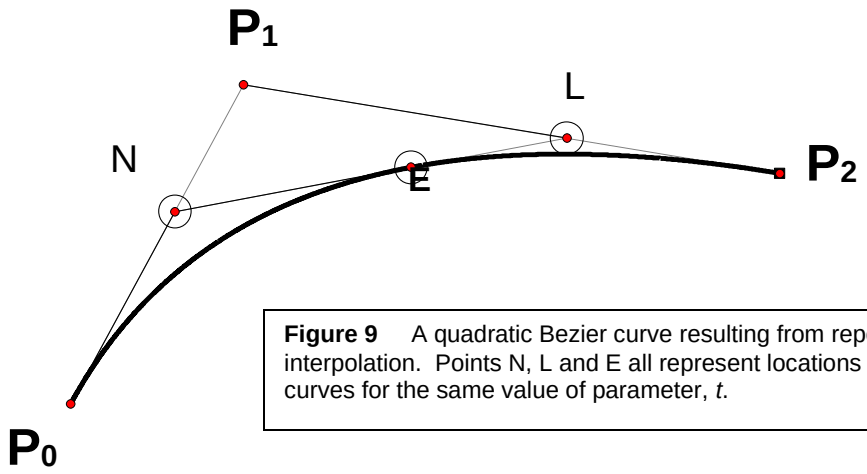


Figure 9 A quadratic Bezier curve resulting from repeated linear interpolation. Points N, L and E all represent locations on their respective curves for the same value of parameter, *t*.

The third degree, or cubic curve is generated is a similar fashion except that the interpolation is carried out for all four control points and proceeds for three rounds instead of two. Figure 10 shows the cubic case. A more detailed explanation and proof of this process is included in a later section of this paper.

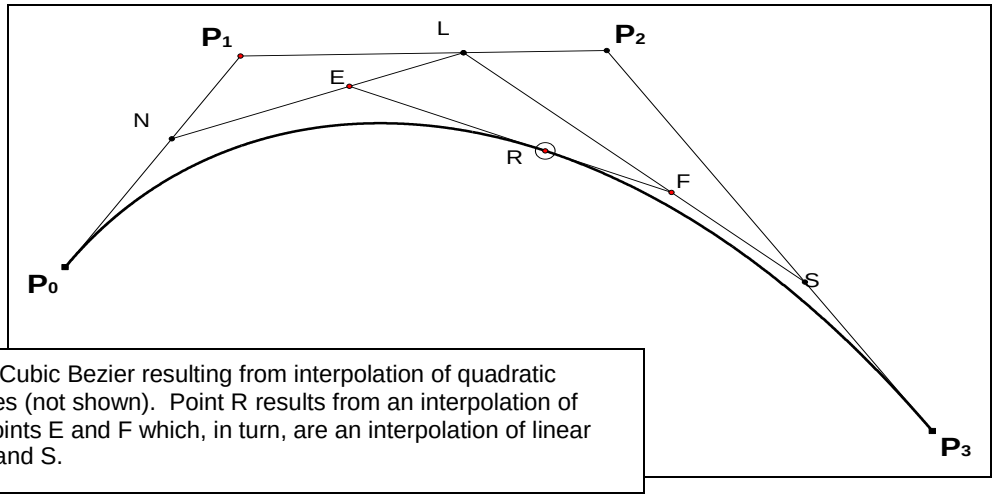


Figure 10 Cubic Bezier resulting from interpolation of quadratic Bezier curves (not shown). Point R results from an interpolation of quadratic points E and F which, in turn, are an interpolation of linear points N, L and S.

My Work with Bezier Curves

As I stated in the introduction, my original interest in these curves was as a tool for mathematically representing curved characters for use in a graphing calculator program that would perform geometric transformations on different letters using matrices. To do that, I needed to design the letters that I would use in the program. I decided to use Geometer's Sketchpad as my canvas and created a tool that I could use to draw each letter using Bezier cubic curves. After drawing the letters on Sketchpad, I transferred the data (location of control points) to a matrix that I would use in my program. The next section of this paper describes the tool I created in GSP and a proof for why it works.

Creating a GSP Tool for Cubic Bezier Curves

As discussed earlier, cubic Bezier curves are formed from repeated linear interpolation and any point along the resulting cubic curve should maintain the same value of t (or ratio) as the original linear form. There are at least a couple of ways to achieve this in GSP. One way is to use dilations (Fima, 1998); the other method, for which I found a quadratic version (Horwath, 2008), uses parallel lines to maintain similar triangles as the parameter, *t*, varies from 0 to 1. I used the second method and adapted the quadratic version to produce a cubic form.

Figure 11 shows the quadratic tool construction. At all times, a line drawn through G and parallel to $\overline{AB}$ will divide segment $\overline{BC}$ by a proportion equal to $\frac{AG}{AC}$. Likewise, a line through G and parallel to $\overline{BC}$, will divide segment $\overline{AB}$ by the same proportion. The resulting points L and N in the figure are both at a position corresponding to the same *t,* in this case, $t = 0.60$. This is true for any length or angle combination of $\overline{AB}$ and $\overline{BC}$.

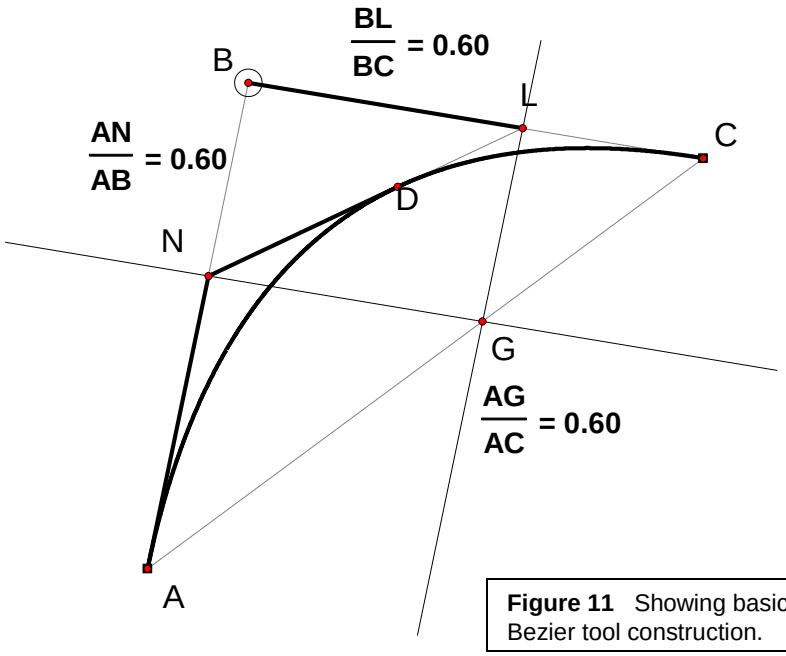


Figure 11 Showing basic ideas behind the quadratic Bezier tool construction.

If the median to $\overline{AC}$ is then constructed, a line parallel to the median and passing through G will divide segment $\overline{LN}$ by the same proportion as *t* (Horwath, 2008). The proof follows.

Proof:

Given a line, $\overleftrightarrow{m}$, constructed through point G and parallel to median $\overline{BO}$ of ΔABC, the intersection of $\overleftrightarrow{m}$ with $\overline{NL}$ (at point D) divides $\overline{NL}$ such that the ratio of ND to NL is the same as the ratio of AG to AC. See Figure 12.

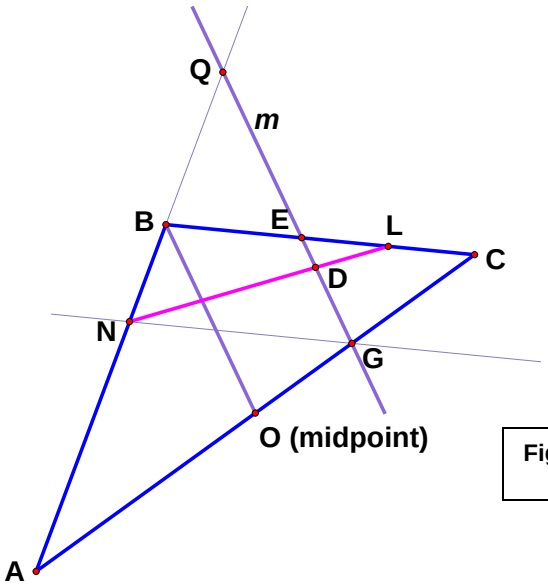


Figure 12 Setup for proof

The general idea of the proof is as follows: Use similar triangles to determine relative distances along $\overline{AB}$ and $\overline{BC}$. Then, using these distances and Menelaus' Theorem (Venema, 2002) on ΔNBL, determine the ratio of ND to NL.

Givens:

$AB = AC = BC = 1$ (corresponds to relative distance along each segment at t=1)

$$\frac{AN}{AB} = \frac{AG}{AC} = \frac{BL}{BC} = t$$

$$AO = \frac{1}{2}$$

Using Similar Triangles:

$\Delta ABO \sim \Delta AQG$

Therefore, $\frac{AB}{AO} = \frac{AQ}{AG}$ and, rearranging, $AQ = \frac{AG \times AB}{AO}$.

Substituting, $AQ = \frac{t \times 1}{\frac{1}{2}} = 2t$

By the Ruler Postulate:

$AQ = AB + BQ$
$BQ = AQ - AB$
$BQ = 2t - 1$

Also,
$NQ = NB + BQ$
$NQ = (1-t) + 2t - 1$
$NQ = t$

By Similar Triangles again:

$\Delta CGE \sim \Delta COB$

Therefore, $\frac{CG}{CE} = \frac{CO}{CB}$ and, rearranging, $CE = \frac{CG \times CB}{CO}$

$CE = \frac{(1-t)(1)}{\frac{1}{2}} = 2(1-t)$

By the Ruler Postulate:

$CE = EL + LC$, so $EL = CE - LC$
$EL = 2(1-t) - (1-t) = 1-t$
Likewise,
$BL = BE + EL$, so $BE = BL - EL$
$BE = t - (1-t) = 2t - 1$

For convenience, Menelaus' Theorem is stated here in terms of the labeling used in this proof. See, also, Figure 13.

<u>Menelaus' Theorem</u> – Let ΔNBL be a triangle. Three proper Menelaus points, D, E, and Q on the lines $\overleftrightarrow{LN}$, $\overleftrightarrow{LB}$, *and* $\overleftrightarrow{NB}$, *respectively, are collinear if and only if:* $\frac{LD}{DN} \times \frac{NQ}{QB} \times \frac{BE}{EL} = -1$. *(Since B is between N and Q,* $\frac{NQ}{QB}$ *is negative.)*

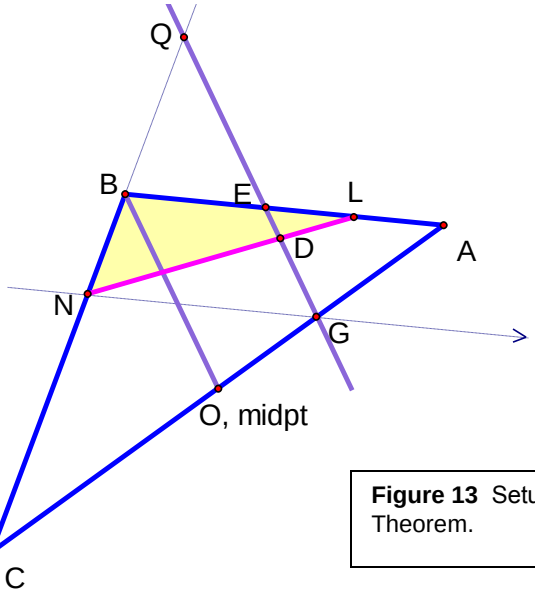


Figure 13 Setup for Menelaus' Theorem.

For reference, the previous results obtained are:

$BE = 2t - 1$
$EL = 1 - t$
$BQ = 2t - 1$
$NQ = t$

Because we have constructed D, E, and Q to be collinear, we are using the theorem to state that the ratio product will be equal to negative one. And knowing this, we can calculate $\frac{LD}{DN}$, the only unknown ratio in the product.

Then we have:

$\frac{LD}{DN} = \frac{QB}{NQ} \times \frac{EL}{BE}$ (after cancelling the negatives)

Substituting, $\frac{LD}{DN} = \frac{(2t-1)(1-t)}{t(2t-1)} = \frac{1-t}{t}$

By the ruler postulate, $LN = LD + DN = (1-t) + t = 1$

So, $\frac{DN}{LN} = t$ Q.E.D.

This is the result desired, so the construction of a line through the driver point, G, and parallel to median $\overline{BO}$ of ΔABC does indeed intersect $\overline{LN}$ at a point which produces the proper ratio for ND to NL.

For the Bezier curve tool that I constructed, these ideas were repeated several times in a sort of "nesting arrangement", with the resulting cubic Bezier curve as the result. The basic linkages used to achieve this are shown in Figure 14, where V is the controlling driver point.

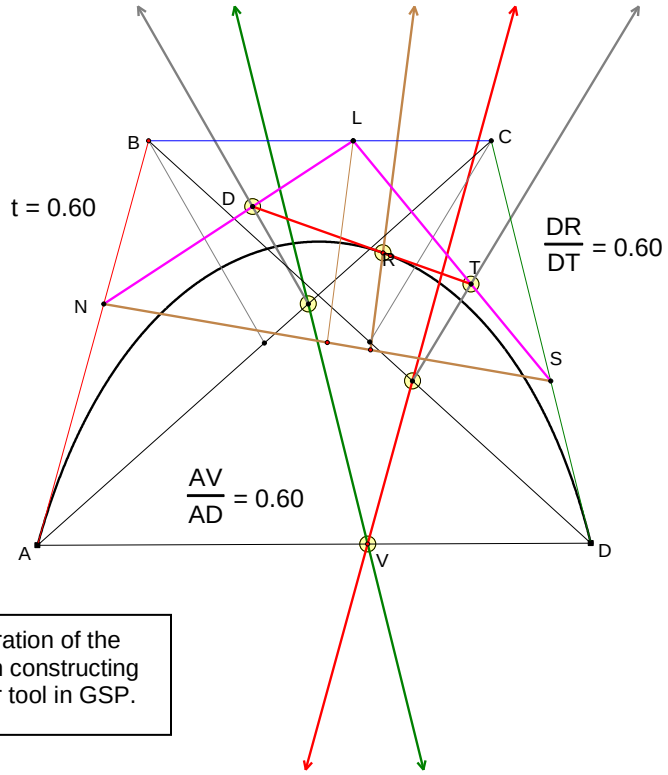


Figure 14 Illustration of the linkages used in constructing the cubic Bezier tool in GSP.

The final result was the cubic interpolating segment $\overline{DT}$ whose point, R, traces the cubic Bezier curve desired. The ratio DR to DT remains at *t* no matter what initial data points A, B, C, D are used.

Teaching Component

Using a TI-83 Calculator Program for Geometric Transformations of Fonts

Using the data from the layouts of the letters that I drew in GSP, I created a calculator program for the TI-83 Plus (see Appendix A) with the thought that it could be used for Algebra II students during a unit on matrices. Generally, in the high school curriculum, it seems that matrices are approached as a way to solve systems of equations which represent "word problems". Looking at matrices from a geometric view after just studying geometry seemed like a good way to bridge the two subjects. Matrix transformations are a powerful tool that students have at their disposal with their graphing calculators now, and connections to computer graphics may spur an interest in a technological career choice later. The small data set that is required for Bezier curves along with the graceful curves that can be produced this way makes them valuable in keeping the students engaged while working with them in this way. Also using a visual example related to computer graphics seemed like a way to interest the students in studying matrices and possibly to pique their interest in computer graphics. There are several ways I thought this program could be used. I chose to pursue an introduction to matrix multiplication as a basis for the lesson plan that follows.

Lesson Plan:

Introduction to Matrix Multiplication through Font Transformations

Student Learning Outcomes:

1. Students will visually identify reflections, rotations and dilations previously learned in geometry, as well as recall their associated coordinate rules.
2. Students will discover the relationship between coordinate rules and matrix representation associated with matrix multiplication.

Materials List:

Lesson worksheet
TI-83 calculator for each student
program: FONTTRNS
TI-GraphLink cable for program transfer

Lesson Overview:

Students will work independently and with a partner as they are introduced to the idea of matrix multiplication visually, through dilations, rotations and reflections of the letter "C". Starting simply, they connect coordinate rules with the numbers used in various positions of a 2 x 2 operation matrix. After working with several different cases they are asked to make conjectures about how numbers in each position of the 2 x 2 matrix affect the coordinates of the transformed point.

Lesson Outline:

1. It would save time for the programs to be transferred to the students' calculators ahead of time. Data for the letter "C" can be transferred ahead of time as well or students can enter data manually into matrix H (12 x 2) (see attached letter layout sheet).
2. The lesson can be launched with an example or discussion of font sizes, 12pt vs. 40pt, for example, and how computers accomplish this easily using matrix multiplication.
3. Inform students they will be working both independently and with a partner as the worksheet instructs. Pair students up and pass out worksheets.
4. Circulate as students work, listening to partner discussions regarding coordinate rules and conjectures, in particular #11 and #12, which are not as obvious as previous transformations.

5. Allow 10 minutes at the end of class for discussion of selected students' conjectures. Additional questions to pose to students:
    - *What would you expect a multiplicative identity matrix to look like?*
    - *In this investigation, dilation, rotation and reflection transformations were accomplished by matrix multiplication. What type of matrix operation do you think could accomplish a translation?*
6. Collect worksheets for assessment.

NCTM Standards for grades 9-12 addressed

Number and Operations

- understand vectors and matrices as systems that have some of the properties of the real-number system
- develop an understanding of properties of and representations for, the addition and multiplication of vectors and matrices
- develop fluency in operations with real numbers, vectors, and matrices, using mental computation or paper-and-pencil calculations for simple cases and technology for more-complicated cases.

Algebra

- generalize patterns using explicitly defined functions
- convert flexibly among and use various representations for relations and functions

Geometry

- understand and represent reflections, and dilations of objects in the plane by using sketches, coordinates, function notation and matrices

Extensions/Variations

- students can choose other letters (see attached sheet) and enter data manually
- students can try to predict coordinate rules entering by different operation matrices

Worksheet:

Introduction to Matrix Multiplication through Font Transformations

Computers easily transform the size, shape, position and orientation of fonts (lettering style) to meet a user's needs. They are able to accomplish this using extremely fast matrix multiplication. This investigation uses a TI-83 program to simulate some of those transformations on the letter "C" (at a much slower speed!).

Using your prior knowledge of geometric transformations (such as translations, dilations, reflections and rotations), this lesson will help you to explore matrix multiplication.

The program will prompt you (in Step 3) to enter the four entries of a 2 x 2 matrix into your calculator. The prompt will be for a particular matrix position, designated by its row, R, and column, C. For example, for matrix $\mathbf{A} = \begin{vmatrix} 2 & 1 \\ 4 & 5 \end{vmatrix}$, the prompt "R2C1" would signal you to enter the number "4" and "R1C1", the number "2".

Working Independently

1. Check that program is loaded into calculator by pressing **PRGM** key on calculator. **FONTTRNS** should appear in list of programs.

2. **EXEC** should be highlighted. Press **ENTER** to start program. Press **ENTER** again. Choose **MANUAL** entry when prompted.

3. Enter the entries of matrix **A**, one at a time, after each identifying prompt.

$\mathbf{A} = \begin{vmatrix} -1 & 0 \\ 0 & 1 \end{vmatrix}$

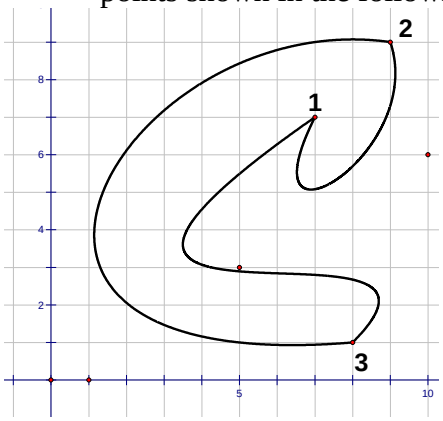
a. It will take the program about 20 seconds to display the result. The resulting display shows a dotted letter and a lined letter. The dotted letter is the original letter or pre-image, while the lined letter is the transformed letter or image. Looking at the pre-image and image displayed, what geometric transformation do you think was performed on the pre-image to obtain the image?

b. One of the following coordinate rules corresponds to this transformation. Circle the correct coordinate rule.

- $(x, y) \rightarrow (-x, y)$
- $(x, y) \rightarrow (-x, -y)$
- $(x, y) \rightarrow (x, -y)$

c. Now you will retrieve the actual coordinates for three corresponding points on both the image and pre-image. To do this, press **STAT** and then press **ENTER** when **EDIT** is highlighted. Use the right/left arrow keys to view list $\mathbf{L_3}$ (pre-image x), $\mathbf{L_4}$ (pre-image y), $\mathbf{L_5}$ (image x) and $\mathbf{L_6}$ (image y). Record these

coordinates in this table. These coordinates always correspond to the 3 labeled points shown in the following illustration (for both the pre-image and image).



point	pre-image x	pre-image y	image x	image y
1				
2				
3				

d. Compare the x-coordinates of the pre-image and image and then the y-coordinates of the pre-image and image. Does the coordinate rule you chose in **b** agree with the actual coordinates retrieved?

4. Press **PRGM** to restart the program. This time when prompted for entries, enter matrix $\mathbf{B} = \begin{vmatrix} 1 & 0 \\ 0 & -1 \end{vmatrix}$. After the result is displayed decide:

a. what type of geometric transformation was performed:

b. what coordinate rule is associated with this transformation:

c. Retrieve the actual coordinates by the same procedure as in **3c** and enter into the table.

point	pre-image x	pre-image y	image x	image y
1				
2				
3				

d. Does your coordinate rule agree with the coordinates retrieved?

5. Press **PRGM** to restart the program. This time, enter matrix $\mathbf{C} = \begin{vmatrix} 2 & 0 \\ 0 & 1 \end{vmatrix}$. Again, decide

a. Geometric transformation performed:

b. Coordinate rule associated with this transformation (estimate proportion from graph:

c. Retrieve the actual coordinates by the same procedure as in **3c** and enter into the table.

point	pre-image x	pre-image y	image x	image y
1				
2				
3				

d. Does your coordinate rule agree with the coordinates retrieved?

6. Press **PRGM** to restart the program. Enter matrix **D** = $\begin{vmatrix} 1 & 0 \\ 0 & 2 \end{vmatrix}$.

a. Geometric transformation performed:

b. Coordinate rule associated with this transformation (estimate proportions from calculated graph):

c. Retrieve the actual coordinates by the same procedure as in **3c** and enter into the table.

point	pre-image x	pre-image y	image x	image y
1				
2				
3				

d. Does your coordinate rule agree with the coordinates retrieved?

7. Look back at matrices **A, B, C, D** and their associated coordinated rules. Summarize the relationships you have discovered between entries in a matrix and the coordinate rule for a given transformation.

entry R1C1:

entry R2C2:

8. Without running the program, make a prediction, first, about the coordinate rule, and then about how the pre-image would be transformed by matrix **E** = $\begin{vmatrix} 2 & 0 \\ 0 & 2 \end{vmatrix}$.

With your partner

9. Discuss your answers to Questions 7 and 8. Write down any new insights you have gained from your discussion and refine your prediction if necessary.

10. Test your refined prediction by inputting matrix **E** and running the program as before. Were you correct? If not, what is the correct transformation and coordinate rule for matrix **E?**

11. Now run **FONTTRNS** with matrix **F** = $\begin{vmatrix} 0 & 1 \\ 1 & 0 \end{vmatrix}$.

a. Retrieve the actual coordinates by the same procedure as in **3c** and enter into the table.

point	pre-image x	pre-image y	image x	image y
1				
2				
3				

b. From the coordinates found in **a**, what is the coordinate rule for this transformation?

c. What transformation was performed?

12. Now run **FONTTRNS** with matrix **G** = $\begin{vmatrix} 0 & -1 \\ 1 & 0 \end{vmatrix}$.

a. Retrieve the actual coordinates by the same procedure as in **3c** and enter into the table.

point	pre-image x	pre-image y	image x	image y
1				
2				
3				

b. From the coordinates found in **a**, what is the coordinate rule for this transformation?

c. What transformation was performed?

12. Without running the program, make a prediction, first, about the coordinate rule, and then about how the pre-image would be transformed by matrix $\mathbf{H} = \begin{vmatrix} 0 & -1 \\ -1 & 0 \end{vmatrix}$. Check your prediction by inputting matrix **G** into the program.

13. Looking over your work up to this point, make a conjecture about how each entry in a 2 x 2 matrix affects the x and y coordinates of the resulting image.

a.

matrix entry	affects image x? (yes or no)	affects image y? (yes or no)	**how**, specifically, is image x or y coordinate affected by given matrix entry?
R1C1			
R1C2			
R2C1			
R2C2			

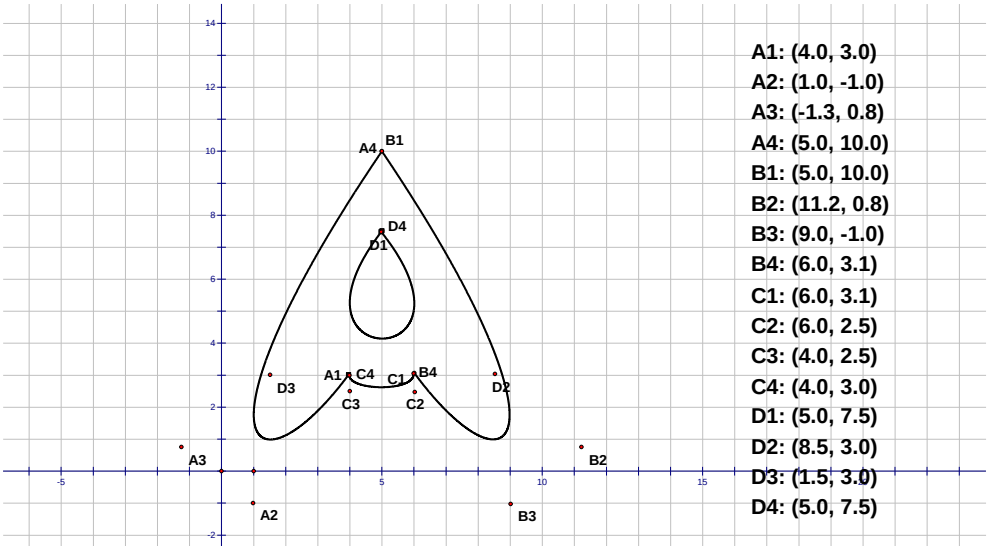
b. Using above table results, answer the following:

i.) Which row of the matrix appears to affect or contribute to the x-coordinate of the image?

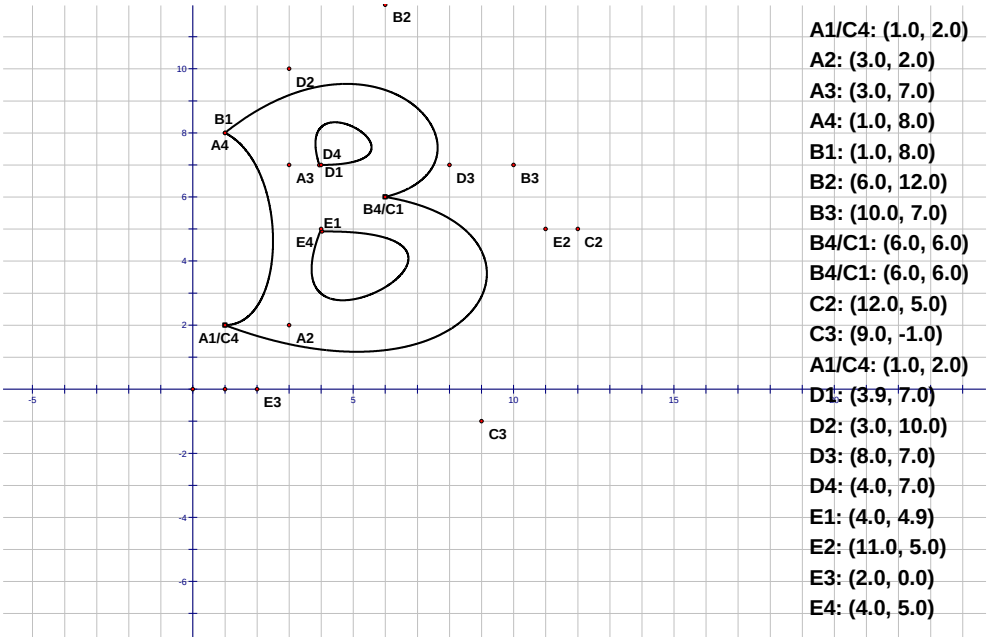
ii.) Which row of the matrix appears to affect the y-coordinate of the image?

14. Using your conjectures from the previous question, what 2 x 2 matrix could you use to produce an image that is dilated horizontally by a factor of 3 and is reflected over the y-axis? Test your matrix by entering it into your calculator. If necessary, adjust your matrix to achieve the desired image.

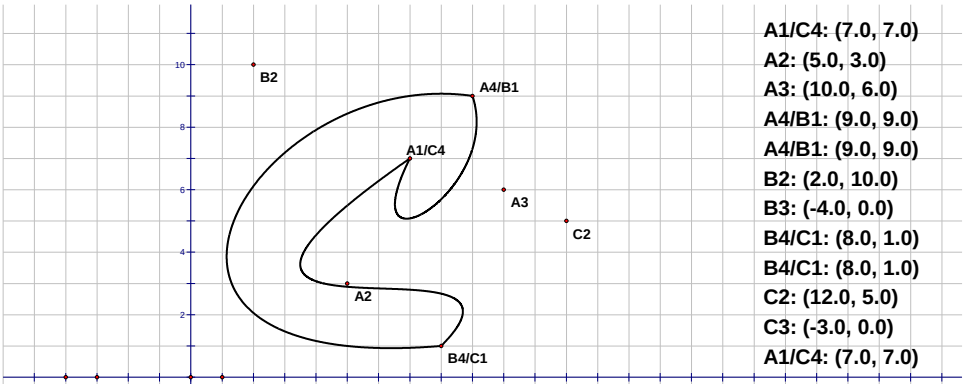
Letter layouts and coordinates



A1: (4.0, 3.0)
A2: (1.0, -1.0)
A3: (-1.3, 0.8)
A4: (5.0, 10.0)
B1: (5.0, 10.0)
B2: (11.2, 0.8)
B3: (9.0, -1.0)
B4: (6.0, 3.1)
C1: (6.0, 3.1)
C2: (6.0, 2.5)
C3: (4.0, 2.5)
C4: (4.0, 3.0)
D1: (5.0, 7.5)
D2: (8.5, 3.0)
D3: (1.5, 3.0)
D4: (5.0, 7.5)



A1/C4: (1.0, 2.0)
A2: (3.0, 2.0)
A3: (3.0, 7.0)
A4: (1.0, 8.0)
B1: (1.0, 8.0)
B2: (6.0, 12.0)
B3: (10.0, 7.0)
B4/C1: (6.0, 6.0)
B4/C1: (6.0, 6.0)
C2: (12.0, 5.0)
C3: (9.0, -1.0)
A1/C4: (1.0, 2.0)
D1: (3.9, 7.0)
D2: (3.0, 10.0)
D3: (8.0, 7.0)
D4: (4.0, 7.0)
E1: (4.0, 4.9)
E2: (11.0, 5.0)
E3: (2.0, 0.0)
E4: (4.0, 5.0)



A1/C4: (7.0, 7.0)
A2: (5.0, 3.0)
A3: (10.0, 6.0)
A4/B1: (9.0, 9.0)
A4/B1: (9.0, 9.0)
B2: (2.0, 10.0)
B3: (-4.0, 0.0)
B4/C1: (8.0, 1.0)
B4/C1: (8.0, 1.0)
C2: (12.0, 5.0)
C3: (-3.0, 0.0)
A1/C4: (7.0, 7.0)

Other ways this calculator program could be used include:

1. After they have learned the mechanics of matrix multiplication, they can use the program's AUTO mode to put in compositions of transformations to see what the result looks like visually and in the matrix form that is outputted. They can start to develop an idea of why most matrix combinations are order sensitive but some are not.
2. The program could possibly be used as an analytic exercise and as a way to further promote flexibility between matrix, numeric and graphical representations. Given a certain result (either screenshot or pre-image/image coordinates), can the student figure out how that result was achieved. This activity is more challenging because some of the transformations are about the origin so that the further away the letter's vertices are from the origin the greater the effects will be. Sample sheets for the elementary transformations are included in Appendix B. Students could use those for comparison.
3. A *Mathematics Teacher* article by Wolff (2002), suggests a way to create motion with a graphing calculator by the use of the ClrDraw command to erase a figure before the newly transformed figure is drawn and by timing this disappearance – reappearance phenomenon by the use of a delay loop in a program. That might be another exciting avenue for students to explore while working with transforming curves.

In conclusion, exploring Bezier curves has been a fascinating topic for me. Although I realize that the teaching component part of my project would have been just as effective in teaching matrix multiplication without the inclusion of any curves at all, figuring out a way to transform a curve was very intriguing to me. I think a student would find it fun also to use the GSP tool to create their own curved figures (letters or otherwise).

References

Cox, Paul (?). *The Mathematics of String Art: A Tribute to Pierre Bezier (1910-1999)*. Retrieved July 12, 2009, from Math Mistakes Web site: http://members.cox.net/mathmistakes/bezier.htm.

Farin, Gerald (1997). *Curves and Surfaces for Computer-Aided Geometric Design: A Practical Guide*. San Diego: Academic Press.

Feicht, Louis. (1998). *Matrix/Spreadsheet/Transformation Lesson*. Retrieved Oct. 23, 2009 from World Web Dynamics. Web site: http://www.worldwebdynamics.com/LOU?matst.htm

Fetter, Annie (1999). *Three Roads to a Parabola*. Retrieved July 12, 2009, from The Math Forum: Annie's Sketchpad Activities Web site: http://mathforum.org/~annie/gsp.handouts/parabola/

Fima, Adrian Oldnow. (1998). Micromaths- Dilations, Splines and CAD- As Easy As ABC (first page). *Teaching Mathematics and Its Applications*, 17(1), 42. doi:10.1093/teamat/17.1.42

Horwath, Michael (2008). *Quadratic Bezier Curve Is a Parabola*. Retrieved July 12, 2009, from Isometricland Web site: **http://www.geocities.com/Area51/Quadrant/3864/geogebra/geogebra_quadratic_bezier_curve_is_parabolic_segment.htm** (These works are licensed under a Creative Commons GNU Lesser General Public License)

Iowa Department of Education. (?). *Iowa High School Mathematics Model Core Curriculum*. Retrieved Oct. 23, 2009 from Website: http://www.iowa-city.k12.ia.us/curriculum/curriculum review reports/math curriculum review/Mathematics%20Model%20Core%20Curriculum.pdf

Kirsanov, Dmitry. (1999). *Nonlinear Design, Bezier Curves*. Retrieved Oct. 23, 2009 from dmitry's design lab. Web site: http://www.webreference.com/dlab/9902/index.html

Lancaster, Don. (1997). *Hardware Hacker #62*. Retrieved Oct. 23, 2009 from Don Lancaster's Guru's Lair Cubic Spline Library. Web site: http://www.tinaja.com/glib/hack62.pdf

Lancaster, Don. (2005). *The Math Behind Bezier Cubic Splines*. Retrieved Oct. 23, 2009 from Don Lancaster's Guru's Lair Cubic Spline Library. Web site: http://www.tinaja.com/glib/cubemath.pdf

Lockman, Kendra. (2008). *Bezier Curves*. Retrieved Oct. 23, 2009 from The Geometer's Sketchpad Resource Center. Web site:

http://www.dynamicgeometry.com/JavaSketchpad/Gallery/Other Expllorations and Amusements/Bezier Curves.html

Mathews, John. (2003). *Module for The Bezier Curve*. Retrieved Oct. 23, 2009 from Cal State Fullerton. Web site: http://math.fullerton.edu/mathews/n2003/BezierCurveMod.html

Mortenson, Michael E. (1997). *Geometric Modeling*. New York: John Wiley and Sons, Inc.

National Council of Teachers of Mathematics. (2000). *Principles and Standards for School Mathematics*. Retrieved Oct. 23, 2009 from NCTM. Web site: http://www.nctm.org/fullstandards/document/default.asp

Plant, Darrel. (1996). *What's a Bezier Curve?* Retrieved from Moshofsky/Plant Creative Services. Web site: http://moshplant.com/direct-or/bezier/index.html

Powell, Nancy (2007). *An Adventure in Line Designs*. from Powell's String Art Project Home Page Web site: http://www.district87.org/staff/powelln/stringart/index.html

Smith, Rod, Holland, Doug etal. (2005). *FontHOWTO: Fonts 101- A Quick Introduction to Fonts*. Retrieved Sept. 25, 2009 from Linux Selfhelp. Web site: http://www.linuxselfhelp.com

Venema, Gerald A. (2002). *The Foundations of Geometry*. Upper Saddle River: Pearson Prentice Hall.

Williams, George. (2000-2007). *An outline font editor for PostScript, TrueType and OpenType fonts*. Retrieved Oct. 23, 2009 form Fontforge. Web site: http://fontforge.sourceforge.net/overview.html#intro

Wolff, Kenneth C. (2002). Elementary Graphics and Animation with Your Calculator. *Mathematics Teacher*, 5(3), 172-176.

Appendix A: TI-83 Plus Program

```
ClrList
LALPHX,LALPHY,LALPHZ,LNEWX,LNEWY,LPLOTX,LPLOTY,LORX,LORY,L
TEMP,L3,L4,L5,L6
identity(3)→[A]
[A]→[B]
[A]→[C]
[A]→[D]
[A]→[E]
[A]→[F]
Lbl H
Matr>list([H],LALPHX,LALPHY)
dim(LALPHX)→ Z
Z→dim(LALPHZ)
Fill(1,LALPHZ)
List>matr(LALPHX,LALPHY,LALPHZ,[J])

Menu("MODE","MANUAL",RE,"AUTO",A)


Lbl A
Input "1 OR 2 TRF?",O
Lbl S
identity(3)→[B]
[B]→[C]
[B]→[D]
[B]→[E]
[B]→[F]

Menu("TRANSFORMATION
TYPE","ROTATION",RO,"SHEAR",SH,"TRANSLATION",TR,"REFLECTION",RE,"
DILATION",DI)

Lbl RO
Input "DEGREES?",R
cos(R)→[B](1,1):
cos(R)→[B](2,2)
sin(R)→[B](2,1)
-sin(R)→[B](1,2)
[B]*[A]→[A]
Goto Z

Lbl SH
Input "X SHEAR?",L
Input "Y SHEAR?",M
```

```
L→[C](1,2)
M→[C](2,1)
[C]*[A]→[A]
Goto Z

Lbl TR
Input "X SHIFT?",H
Input "Y SHIFT?",K
H→[D](1,3)
K→[D](2,3)
[D]*[A]→[A]
Goto Z

Lbl RE
Input "R1C1",A
Input "R1C2",B
Input "R2C1",C
Input "R2C2",D
A→[E](1,1)
B→[E](1,2)
C→[E](2,1)
D→[E](2,2)
[E]*[A]→[A]
Goto Z

Lbl DI
Input "X DILATION",U
Input "Y DILATION",V
U→[F](1,1)
V→[F](2,2)
[F]*[A]→[A]
Goto Z

Lbl Z
If O=2
Then
1→O
Goto S
End

[A]*[J]^T→[G]
[G]^T→[I]
Matr>list([I],LNEWX,LNEWY)

ClrDraw
FnOff
```

```
PlotsOff

Lbl B
For(I,1,2)
For(N,1,Z/4,1)
For(E,1,2,1)
For(P,0,1,0.1)
LNEWX(4N-3)*(1-P)^3+LNEWX(4N-2)*3(1-P)^2P+LNEWX(4N-1)*3(1-
P)P^2+LNEWX(4N)*P^3→LPLOTY(11*N-(1-P)10)
End

If E=1
Then
For(F,11*N-10,11*N)
If I=1
Then
LPLOTY(F)→LPLOTX(F)
Else
LPLOTY(F)→LORX(F)
End
End
For(W,1,4)
LNEWY(4(N-1)+W)→LNEWX(4(N-1)+W)
End
Else
For(F,11*N-10,11*N)
If I=1
LPLOTY(F)→LTEMP(F)
LPLOTY(F)→LORY(F)
LTEMP(F)→LPLOTY(F)
End
End
End
End

If I=1
Then
Matr>list([J],LNEWX,LNEWY)
End
End


N-1→N
Plot1(Scatter,LPLOTX,LPLOTY,Ò)
Plot2(Scatter,LORX,LORY,Ñ)
PlotsOn (1,2)
```

```
ZoomStat
PlotsOff (1)


For(N,1,Z/4)
11*N-10→T
LORX(T)→L3(N)
LORY(T)→L4(N)
LPLOTX(T)→L5(N)
LPLOTY(T)→L6(N)
For(Q,11*N-10,11*N-1)
Line(LPLOTX(Q),LPLOTY(Q),LPLOTX(Q+1),LPLOTY(Q+1))
End
End
Stop
```

Matrix H data file:

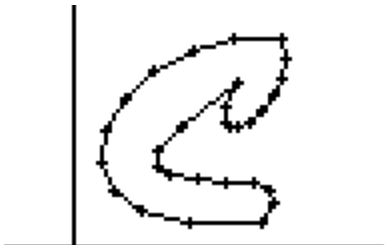
```
//H

7 7
5 3
10 6
9 9
9 9
2 10
-4 0
8 1
8 1
12 5
-3 0
7 7
```

Appendix B: Matrix Forms for Elementary Transformations (& Screen Shots)

Euclidean Transformations

Identity

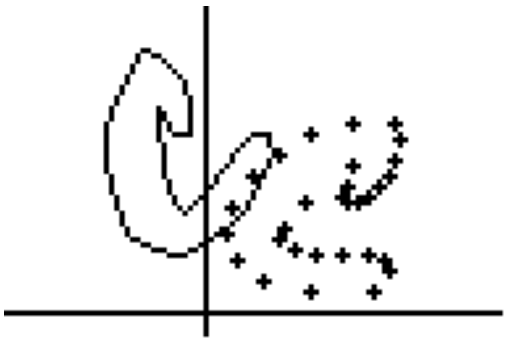
$$[A] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Screen shot (image/pre-image superimposed)
pre-image: plotted +'s
image: line drawing

Rotation

$$[A] = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

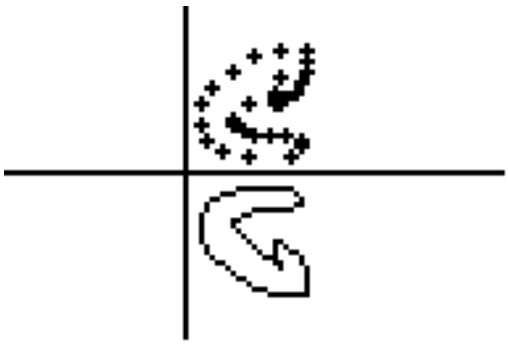


Translation

$$[A] = \begin{bmatrix} 1 & 0 & h \\ 0 & 1 & k \\ 0 & 0 & 1 \end{bmatrix}$$

Reflection

$$[A] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



through x-axis

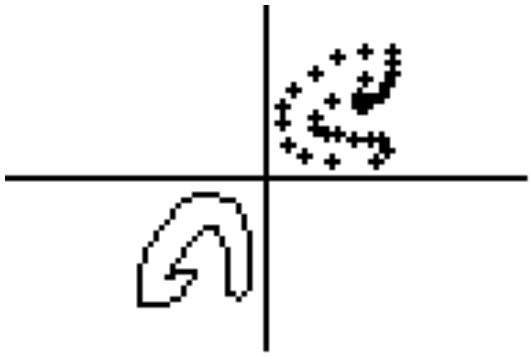
$$[A] = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

through y-axis

$$[A] = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

through y=x axis

$$[A] = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



through y=-x axis

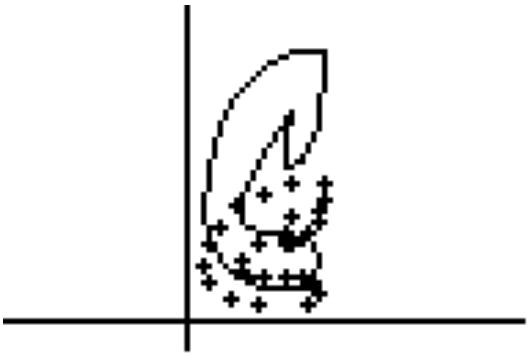
Affine Transformations
Dilation

$$[A] = \begin{bmatrix} k_x & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

horizontal

$$[A] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

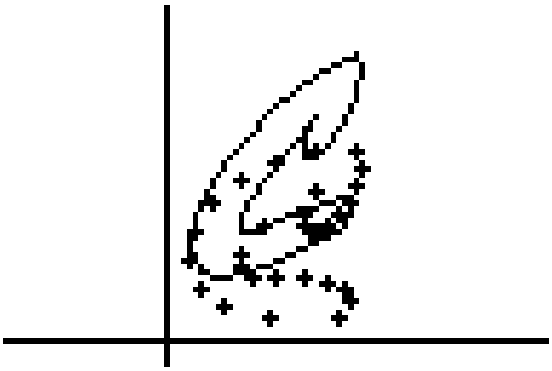
vertical



Shear

$$[A] = \begin{bmatrix} 1 & 0 & 0 \\ S_V & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

vertical



$$[A] = \begin{bmatrix} 1 & S_H & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

horizontal