A Tool for Teaching Curve Design[*]

Yuan Zhao, John L. Lowther and Ching-Kuang Shene[†]
Department of Computer Science
Michigan Technological University
Houghton, MI 49931–1295
Email: [yzhao|john|shene]@mtu.edu

1 Introduction

This paper describes a tool for teaching curve design. This tool is a component of the software tools to be used in a computing with geometry course [3, 4] that is being developed under the support of National Science Foundation. Curve design is important in computer graphics, animation, and computer aided design. Unfortunately, curve design requires very involved mathematics even though many curve design concepts are intuitive. As a result, it has been a challenging job for instructors teaching curves and surfaces in computer graphics, computer aided design, and other related courses. During past years, there have not been very many efforts dedicated to curve design tool development. Yen [7] produced a well-received video program explaining important concepts of B-spline curves and surfaces and Rockwood and Chambers [6] published a multimedia tutorial on computer aided geometric design. The former only provides a one-way communication, while the latter restricts users to a predefined environment with very limited interaction for users to carry out experiments. To fill this gap, our tool provides students with a fully interactive environment in which they are free to design, modify, and manipulate curved objects and perform experiments without constraints.

In the following, Section 2 presents design issues, Section 3 discusses general features, Section 4 enumerates basic elements, Section 5 covers advanced topics, and Section 6 is our conclusion. Interested readers should consult [1, 2, 5] for mathematical background details.

[*]This work was partially supported by the National Science Foundation under grant DUE-9653244. The third author was also partially supported by the National Science Foundation under grant CCR-9696084 (formerly CCR-9410707).

[†]Communicating author

2 System Design Issues

2.1 Design Goal

The goal of this project is to design a tool for students to learn fundamental curve concepts, gain hands-on experience in curve design, and acquire curve design skills without getting into deep and involved mathematics. We believe that once students understand the fundamentals they can easily follow the mathematical derivations in later courses. Moreover, instructors may also find that this pedagogical tool could make their courses more interesting and intuitive rather than theoretical.

2.2 Supported Activities

Our tool covers four different types of curves: Bézier, rational Bézier, B-spline and NURBS (Non-Uniform Rational B-spline). These types of curves are defined by several parameters: a set of control points, a knot vector, and the degree of the curve. For NURBS curves, each control point has a weight. Adjusting these parameters yields different curves. Our tool provides a mechanism to manipulate these parameters.

In addition to displaying the curve based on given parameters, our tool can also show important and interesting properties. These include partition of unity, the computation of basis functions, the convex hull property, de Casteljau's algorithm, de Boor's algorithm, curve subdivision, knot insertion and infinite control points. Details can be found in Section 4 and 5.

2.3 Portability

Portability is one of our major concerns. We want to make our tool available on as many platforms as possible. Although our tool is being developed on Silicon

Graphics O2's, we do not use most of the SGI's powerful tools since they may not be available on other platforms. As a result, we only use C, OpenGL and GLUT. OpenGL has become very popular recently. While not all platforms support OpenGL, there is a popular and freely available OpenGL clone, Mesa, that can run on most popular platforms. GLUT is a windowing toolkit for OpenGL available on most UNIX platforms and Windows 95/NT and OS/2. Since all development tools being used are on public domain, our tool is highly portable. We have successfully tested it with Mesa and GLUT on Sun systems. Other platforms will be tested in the near future.

3 Curve Design Elements

3.1 User Interface

Figure 1 shows a typical interface of our tool. Students can add and move control points on the drawing canvas, select the type of curves with the buttons on the top row, and trace the curve with the vertical slider. This vertical slider is also used for showing and modifying knots. For NURBS curves, students can use the horizontal slider for modifying the weight of a selected control point. The coordinates of the selected point are displayed and can be modified for finer position control. There are other buttons for selecting frequently used options such as displaying the de Casteljau and de Boor control nets, convex hull and control polygon.

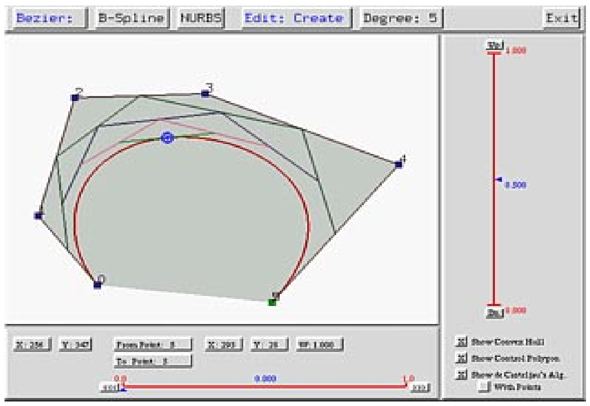


Figure 1: The Convex Hull and de Casteljau Control Net of a Bézier Curve

Our tool can save the curves to a file so that a complex design can be performed in several sessions.

3.2 Control Points

Students can create a set of control points, insert a control point after or before a selected one, and select, move and delete control points. The curve being designed is rendered on-the-fly as control points are manipulated. For a B-spline or NURBS curve, students must also specify curve's degree before the curve can be rendered.

For NURBS curves, since weights can be any real number, an exponential scale is used. More precisely, a value t representing a position on the slider is transformed by a function of type $\exp(t)$. In this way, a weight can quickly be made very large and have a finer control when t is small. Our tool supports infinite control points which have weights of zero. An infinite control point becomes a direction vector and when it is moved, the corresponding curve segment is pushed or pulled in that direction. Infinite control points provide a simpler way for constructing commonly seen curves and surfaces such as circles, spheres and surfaces of revolution (*e.g.*, tori).

3.3 Knot Sequence

There is a special relation among the three defining parameters of a B-spline or NURBS curve: $m = n + p + 1$, where m is the number of knots, n the number of control points, and p the degree of the curve. Our tool allows students to specify a set of control points and the degree of the curve. After this, students can modify the default uniform knots, and make some of them multiple ones. Students can request to have the curve clamped (*i.e.*, both ends being tangent to the control polygon) or make the curve a closed one. They can also use knot insertion to insert knots and degree elevation to increase the curve's degree (Section 5).

3.4 Multiple Curve Segments

Students can work on multiple curves for complex design tasks such as font and surface profile design. They can subdivide an existing curve into several segments and change their shapes (Section 5.2). Or, they can create a new curve and join it with existing ones to form a more complex curve.

4 Basic Topics

4.1 Tracing the Curve

For a Bézier curve defined by $n + 1$ control points $\mathbf{p}_0$, $\mathbf{p}_1$, ..., $\mathbf{p}_n$, the point on the curve that corresponds to u is defined to be

$$\mathbf{p}(u) = \sum_{i=0}^{n} B_{n,i}(u)\mathbf{p}_i, \quad u \in [0, 1]$$

where the Bézier basis function is defined as follows:

$$B_{n,i}(u) = \frac{n!}{i!(n-1)!}u^i(1-u)^{n-1}$$

As u moves, point $\mathbf{p}(u)$ traces out the curve. To better show this effect, our tool displays a curve and uses a special mark that follows the curve (Figure 1 and 4).

For B-spline and NURBS curves, knots (*i.e.*, $u_0 \le u_1 \le \cdots \le u_m$) are restricted to $[0,1]$. The point on the curve that corresponds to u is

$$\mathbf{p}(u) = \frac{1}{\sum_{i=0}^{n} w_i N_{i,p}(u)} \sum_{i=0}^{n} w_i N_{i,p}(u)\mathbf{p}_i, \quad u \in [0,1]$$

where w_i is the weight of control point $\mathbf{p}_i$ and $N_{i,p}(u)$ is the B-spline basis function defined recursively as follows:

$$\begin{aligned} N_{i,0}(u) &= \begin{cases} 1 & u \in [u_i, u_{i+1}) \\ 0 & \text{otherwise} \end{cases} \\ N_{i,p}(u) &= \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \\ &\qquad \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \end{aligned}$$

For B-spline curves, all weights, w_i, are equal to 1.

For a NURBS curve, increasing the value of w_i forces the curve to approach control point $\mathbf{p}_i$. If w_i is set to infinity, the curve passes through $\mathbf{p}_i$. The left NURBS curve in Figure 2 has all $w_i = 1$ while the right one increases w_6 to 10.

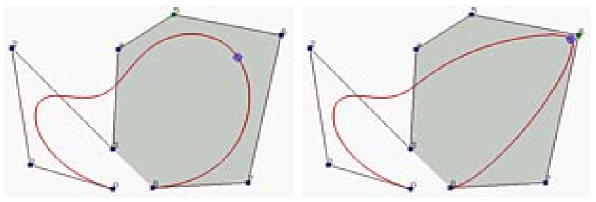


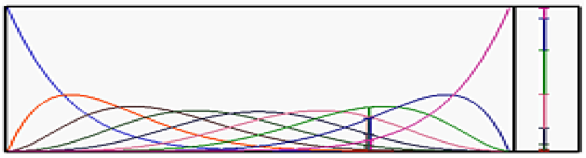
Figure 2: The Effect of Changing Weights

4.2 Partition of Unity

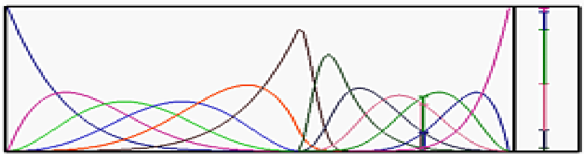
The sum of all coefficients of control points is 1. If the weights are non-negative, all coefficients are in $[0,1]$. As a result, interval $[0,1]$ is "partitioned" by these coefficients. Note that the way of partitioning $[0,1]$ depends on the value of u. Partition of unity can be considered as a weighted average of the control points and provides a way of modifying the shape of a curve.

For Bézier curves, partitions always follow a fixed pattern, since $B_{n,i}(u)$ is fixed once u is given. For B-spline curves, the partition for a particular u depends on the knot vector and basis functions $N_{i,p}(u)$. For NURBS curves, the partition also depends on weights.

Our tool can display the coefficients and partition of unity on-the-fly in a separate window as the values of u, knots and weights are changing (Figure 3). Each coefficient is shown with a different color along with a vertical bar showing the actual partition.



(a) Partition of Unity of a Bézier Curve



(b) Partition of Unity of a B-spline Curve

Figure 3: Partition of Unity

4.3 Convex Hull Property

A direct consequence of the partition of unity property is that the curve lies in the convex hull of certain defining control points. For NURBS curves, this holds only for non-negative weights.

Different types of curves require different ways for showing the convex hull. For Bézier curves, the convex hull is defined by the control points and can be displayed once the control points are available (Figure 1). For B-spline and NURBS curves, part of the curve lies in the convex hull defined by those control points whose corresponding basis functions are non-zero (Figure 4). Furthermore, as u moves in $[0,1]$, the convex hull that corresponds to non-zero basis functions at u will also change. Our tool can show the change of convex hull.

4.4 De Casteljau's Algorithm

De Casteljau's algorithm provides a beautiful geometric treatment to computing a point on a Bézier curve. Given a set of $n+1$ control points $\mathbf{p}_0^0$, $\mathbf{p}_1^0$, …, $\mathbf{p}_n^0$ and $u \in [0,1]$, de Casteljau's algorithm successively computes a set of polylines until a polyline degenerates to a point, which is the point $\mathbf{p}(u)$ on the curve. Starting with polyline $\mathbf{p}_0^0\mathbf{p}_1^0 \cdots \mathbf{p}_n^0$, a point $\mathbf{p}_i^j$ on segment

$\mathbf{p}_{i-1}^{j-1}\mathbf{p}_i^{j-1}$ is computed such that the ratio of the length of $\mathbf{p}_{i-1}^{j-1}\mathbf{p}_i^j$ and the length of $\mathbf{p}_{i-1}^{j-1}\mathbf{p}_i^{j-1}$ is u:

$$\mathbf{p}_i^j = (1-u)\mathbf{p}_{i-1}^{j-1} + u\mathbf{p}_i^{j-1} \tag{1}$$

This procedure generates n more polylines: $\mathbf{p}_0^0\mathbf{p}_1^0\cdots\mathbf{p}_n^0$, $\mathbf{p}_1^1\mathbf{p}_2^1\cdots\mathbf{p}_n^1$, $\mathbf{p}_2^2\mathbf{p}_3^2\cdots\mathbf{p}_n^2$, $\ldots$, $\mathbf{p}_k^k\mathbf{p}_{k+1}^k\cdots\mathbf{p}_n^k$, $\ldots$, and $\mathbf{p}_n^n = \mathbf{p}(u)$. Our tool can show all of these polylines (*i.e.*, de Casteljau control net) and the computation of $\mathbf{p}(u)$. The step-by-step computation of $\mathbf{P}(u)$ is similar to the triangular scheme of finite difference computation in numerical methods. Figure 1 shows a de Casteljau control net of a Bézier curve of degree 5 at $u = 0.5$.

4.5 De Boor's Algorithm

De Boor's algorithm for B-spline and NURBS curves is a generalization of de Casteljau's algorithm. Both algorithms have a similar computation scheme. Given a set of control points $\mathbf{p}_0^0$, $\mathbf{p}_1^0$, $\mathbf{p}_2^0$, $\ldots$, $\mathbf{p}_n^0$, the degree of the curve p, and a point $u \in [u_k, u_{k+1})$, since only $N_{k-p,p}(u)$, $N_{k-p+1,p}(u)$, $\ldots$, $N_{k,p}(u)$ are non-zero, $\mathbf{p}(u)$ is computed using $\mathbf{p}_{k-p}^0$, $\mathbf{p}_{k-p+1}^0$, $\ldots$, $\mathbf{p}_k^0$. Like Equation (1), de Boor's algorithm computes intermediate points with the following:

$$\mathbf{p}_i^j(u) = (1-\alpha_i^j)\mathbf{p}_{i-1}^{j-1}(u) + \alpha_i^j\mathbf{p}_i^{j-1}(u) \tag{2}$$

where α_i^j is defined as follows:

$$\alpha_i^j = \frac{u - u_i}{u_{i+p-j+1} - u_i}$$

Computation starts with polyline $\mathbf{p}_{k-p}^0\mathbf{p}_{k-p+1}^0\cdots\mathbf{p}_k^0$ and uses Equation (2) to generate a sequence of polylines $\mathbf{p}_{k-p+1}^1\mathbf{p}_{k-1+2}^1\cdots\mathbf{p}_k^1$, $\mathbf{p}_{k-p+2}^2\mathbf{p}_{k-1+3}^2\cdots\mathbf{p}_k^2$, $\ldots$, until a polyline degenerates to a point, which is the point $\mathbf{p}(u)$ on the curve. These polylines form a de Boor control net. If u is a knot of multiplicity r, the last point is $\mathbf{p}_k^{p-r}$; otherwise, the last point is $\mathbf{p}_k^p$.

De Boor's algorithm can be applied to NURBS curves. Let $\hat{\mathbf{p}}_i = w_i\mathbf{p}_i$, where $\hat{\mathbf{p}}_i$ is a four-dimensional point. Then, applying de Boor's algorithm to these new points yield $\hat{\mathbf{p}}_k^{p-r}$, which is still a four-dimensional point. Finally, converting it back to three dimensions with $\mathbf{p}(u) = \hat{\mathbf{p}}_k^{p-r}/w_k^{p-r}$ yields the result.

Our tool can display the de Boor control net on-the-fly. Figure 4 shows the convex hull and de Boor control net of a clamped B-spline curve of degree 4, 10 control points, four internal knots at 0.22, 0.42 (multiplicity 2), 0.6 and 0.75, and $u = 0.88$. Since u is in the last knot span and the number of non-zero basis functions is 5, the convex hull is defined by $\mathbf{p}_5$ to $\mathbf{p}_9$ with which the de Boor control net is computed.

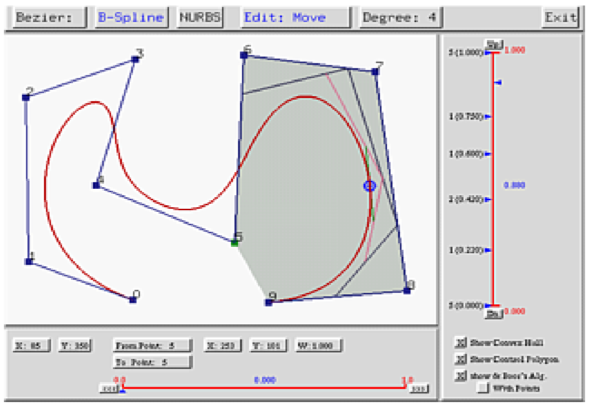


Figure 4: The Convex Hull and de Boor Control Net of a B-spline Curve

5 Advanced Topics

5.1 Degree Elevation

Degree elevation can increase the degree of the curve without changing its shape. It is an important technique for achieving higher control flexibility. For a Bézier curve defined by $\mathbf{p}_0$, $\mathbf{p}_1$, $\ldots$, $\mathbf{p}_n$, the following new set of control points $\mathbf{q}_0 = \mathbf{p}_0$, $\mathbf{q}_1$, $\ldots$, $\mathbf{q}_n$ and $\mathbf{q}_{n+1} = \mathbf{p}_n$ defines the same curve and increases the degree of the curve by 1:

$$\mathbf{q}_i = \left(\frac{i}{n+1}\right)\mathbf{p}_{i-1} + \left(1 - \frac{i}{n+1}\right)\mathbf{p}_i, \qquad 1 \le i \le n$$

Our tool allows students to increase the degree of a curve and observe an important fact that as the degree increases the control polyline moves closer to the curve and has the latter as a limit case.

5.2 Curve Subdivision

Curve subdivision is a handy technique in curve design which permits a user to subdivide a curve into segments and only work on those unsatisfactory ones without affecting the others. For a Bézier curve, a user can choose a $u \in [0,1]$ and subdivide the curve at $\mathbf{p}(u)$ into two segments, each of which is a Bézier curve of the same degree with a new set of control points. One important fact is that after subdivision the two curve segments are tangent to each other at a joining control point. Thus, while the control points of each segment can be moved freely and the degree of each segment can be further increased, the joining control point and its two neighbors must be collinear to maintain tangential continuity.

Curve subdivision for Bézier curves is a byproduct of de Casteljau's algorithm. Given u, using de Casteljau's algorithm, a control net is generated and points $\mathbf{p}_0^0$, $\mathbf{p}_1^1$,

…, $\mathbf{p}_n^n$ define the first segment and $\mathbf{p}_n^n$, $\mathbf{p}_n^{n-1}$, …, $\mathbf{p}_n^0$ define the second.

Our tool allows students to choose a $u \in [0,1]$ to perform a subdivision for a Bézier curve. After subdivision, each curve segment is reparameterized so that the domain is always $[0,1]$. Students can select the the segment they wish to work on. In the current version, students must maintain the collinearity condition at joining control points. In future versions, we plan to add an option so that all three collinear points are moved and rotated as a single unit. Figure 5 shows a subdivision at $u = 0.5$ of the Bézier curve in Figure 1 and the convex hull of its left segment. Subdivision for B-spline and NURBS curves will be available soon.

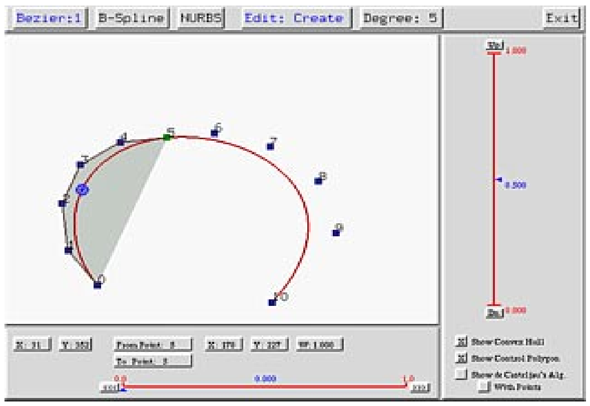


Figure 5: A Subdivided Bézier Curve

5.3 Knot Insertion

Knot insertion is a powerful tool in the study of B-spline and NURBS curves and has many applications. Its basic idea is to insert additional knots for finer shape control without affecting the current shape of the curve. Because of $m = n+p+1$, inserting one knot requires the addition of one more control point. Suppose $u \in [u_k, u_{k+1})$ initially has multiplicity s and is to be inserted r times, $r + s \le p$, where p is the degree of the B-spline or NURBS curves. The ith new control point in the jth insertion step, $\mathbf{q}_i^r$, is computed recursively:

$$\mathbf{q}_i^j = (1 - \alpha_i^j)\mathbf{q}_{i-1}^{j-1} + \alpha_i^j \mathbf{q}_i^{j-1}$$

where α_i^j is

$$\alpha_i^j = \begin{cases} 1 & i \le k - p + r - 1 \\ (u - u_i)/(u_{i+p-j+1} - u_i) & k - p + r \le i \le k - s \\ 1 & i \ge k - s + 1 \end{cases}$$

Please notice the similarity of the above equation and Equation (2). In fact, de Boor's algorithm is implemented with repeated knot insertions.

In addition to find $\mathbf{p}(u)$ on a curve, knot insertion can also be used in computing the derivatives and subdivisions of curves. Our tool allows students to insert knots one at a time. In future versions, students can insert several knots simultaneously.

6 Conclusion

In this paper, we have described important features of our tool for curve design to be used in an NSF supported course and in computer graphics and other related courses. Although we have tried our best to incorporate many important concepts and techniques into our tool, as of this writing, some of them are still in testing stage, while some advanced topics remain on our to-do list (*e.g.*, knot refinement and knot removal). Finally, one of the most important concepts that has not yet been considered is blossoming, since it is rather abstract although very useful. It will be available in the final version.

References

[1] Gerald Farin, *NURB Curves and Surfaces*, A K Peters, 1995.

[2] Gerald Farin, *Curves and Surfaces for CAGD: A Practical Guide*, forth edition, Academic Press, 1997.

[3] John L. Lowther and Ching-Kuang Shene, Geometric Computing in the Undergraduate Computer Science Curricula, *The Journal of Computing in Small Colleges*, Vol. 13 (1997), No. 2 (November), pp. 50–61.

[4] John L. Lowther, Ching-Kuang Shene and Yuan Zhao, Computing with Geometry as an Undergraduate Course, August 1997. Available at `http://www.cs.mtu.edu/~shene/edu/education.html`.

[5] Les Piegl and Wayne Tiller, *The NURBS Book*, Springer-Verlag, 1995.

[6] Alyn Rockwood and Peter Chambers, *Interactive Curves and Surfaces: A Multimedia Tutorial on CAGD*, Morgan Kaufmann, 1996.

[7] Jonathan Yen, *Knotty: A B-Spline Visualization Program*, Part I and II, Morgan Kaufmann, San Francisco, 1993.