Serdica J. Computing **5** (2011), 323–332

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

INTERACTIVE 3D VISUALIZATION OF BÉZIER CURVES USING JAVA OPEN GRAPHICS LIBRARY (JOGL)*

Krassimira Vlachkova, Marina Boikova

Abstract. We present a new program tool for interactive 3D visualization of some fundamental algorithms for representation and manipulation of Bézier curves. The program tool has an option for demonstration of one of their most important applications—in graphic design for creating letters by means of cubic Bézier curves. We use Java applet and JOGL as our main visualization techniques. This choice ensures the platform independency of the created applet and contributes to the realistic 3D visualization. The applet provides basic knowledge on the Bézier curves and is appropriate for illustrative and educational purposes. Experimental results are included.

1. Introduction. Computer Aided Geometric Design (CAGD) is an applied mathematical area which combines theoretical results with practical methods for solving problems in geometric modeling by means of computer algorithms and technologies. The aim of CAGD is to create efficient algorithms, methods and program products through which geometric objects (curves, surfaces, solids) can be described, digitally represented and visualized with a necessary precision.

ACM Computing Classification System (1998): I.3.5, J.6.
Key words: Bézier curves, applet, JOGL.
*This work was partially supported by Sofia University Science Fund Grant No.195/2011.

Bézier curves are a fundamental mathematical technique in CAGD for representation, manipulation and visualization of parametric curves. They were introduced in the late '60s by Paul de Casteljau and independently by Pierre Bézier in connection with applications to the automotive industry. Nowadays the field includes a wide range of algorithms and methods and is an independent and dynamic mathematical area, see, e. g., [5, 6, 7]. The most important applications of Bézier curves are in computer graphics, computer animation and graphic design. Many of the popular graphic packages in font design use Bézier curves. For example `TrueType`, developed by Apple Computer, uses quadratic Bézier curves, and PostScript, developed by Adobe Systems, uses cubic Bézier curves.

In this paper we present a new program tool for interactive visualization and manipulation of 3D Bézier curves and for visual demonstration of the process of creating letters in graphic design by means of cubic Bézier curves. We have used some of the most up-to-date visualization techniques—Java applet ([4, 9]) and JOGL ([2, 3]). The created applet illustrates basic concepts and algorithms from the theory of Bézier curves and can be used for educational purposes.

The rest of the paper is organized as follows. Section 2 gives a brief description of the algorithms for representation and manipulation of Bézier curves. In Section 3 we provide our motivation of the choice of the software tools. Section 4 gives detailed information about the applet's main features and presents experimental results.

2. Mathematical background.

2.1. Geometric construction of Bézier curves: de Casteljau algorithm. The de Casteljau algorithm is a fundamental algorithm in curve and surface design. Based on a geometric construction, it is surprisingly simple. Yet this is one of the most robust and numerically stable methods for evaluating polynomial curves.

Let n be a natural number, $\mathbf{b}_0$, $\mathbf{b}_1$, ..., $\mathbf{b}_n$ are $n+1$ different points in $\mathbb{R}^3$ and $t \in [0,1]$. The de Casteljau algorithm uses consecutive linear interpolations and in n steps constructs a point $\mathbf{b}_0^n(t)$ on a polynomial curve $\mathcal{B}$ of degree n. The curve $\mathcal{B}$ is called *Bézier curve*. The points $\mathbf{b}_i$, $i = 0, \ldots, n$, are called *control points* or *Bézier points*. The polygon with vertices $\mathbf{b}_0, \ldots, \mathbf{b}_n$ is called *control polygon* or *Bézier polygon* of the curve.

The algorithm is as follows:

Algorithm 1 *de Casteljau*

Input: $\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^3,\ t \in \mathbb{R}$

Output: $\mathbf{b}_0^n(t)$ *is a point on the related polynomial curve* $\mathcal{B}$ *corresponding to the parameter* t.

for $i = 0 \to n$ **set** $\mathbf{b}_i^0 := \mathbf{b}_i$
for $r = 1 \to n$
  for $i = 0 \to n - r$
    $\mathbf{b}_i^r(t) := (1-t)\mathbf{b}_i^{r-1}(t) + t\mathbf{b}_{i+1}^{r-1}(t)$
end

It is convenient to arrange the intermediate points $\mathbf{b}_i^r(t)$ into a triangular array which is called *de Casteljau scheme*: see Figure 1 for the case of a cubic Bézier curve ($n = 3$).

$$\begin{array}{llll} \mathbf{b}_0 & & & \\ \mathbf{b}_1 & \mathbf{b}_0^1(t) & & \\ \mathbf{b}_2 & \mathbf{b}_1^1(t) & \mathbf{b}_0^2(t) & \\ \mathbf{b}_3 & \mathbf{b}_2^1(t) & \mathbf{b}_1^2(t) & \mathbf{b}_0^3(t) \end{array}$$

Fig. 1. A de Casteljau scheme for a cubic Bézier curve

Detailed information about the de Casteljau algorithm and the Bézier curves and their properties can be found, e. g., in [1, 5, 6, 7].

2.2. The Bernstein form of a Bézier curve. In the previous section Bézier curves were defined using the geometric algorithm of de Casteljau. Here we give the analytical presentation of Bézier curves using *Bernstein polynomials* which are defined by

$$B_i^n(t) := \binom{n}{i} t^i (1-t)^{n-i},\ 0 \le t \le 1,$$

where

$$\binom{n}{i} := \begin{cases} \frac{n!}{i!(n-i)!} & \text{if } 0 \le i \le n \\ 0 & \text{else} \end{cases}.$$

Bézier curves can be represented analytically through Bernstein polynomials by $\mathcal{B}(t) := \mathbf{b}_0^n(t) = \mathbf{b}_0 B_0^n(t) + \mathbf{b}_1 B_1^n(t) + \cdots + \mathbf{b}_n B_n^n(t)$. Note that the evaluation of Bézier curves using Bernstein polynomials is faster but less robust.

2.3. Subdivision of a Bézier curve. Let $c \in (0,1)$. Finding the control points of the part of the Bézier curve that corresponds to the interval $[0,c]$ is called *subdivision* of the Bézier curve. The de Casteljau algorithm not only computes the value $\mathcal{B}(c) = \mathbf{b}_0^n(c)$ but also provides the control points of the Bézier curves corresponding to the intervals $[0,c]$ and $[c,1]$. Precisely, these control points are $\mathbf{b}_0^j(c)$ and $\mathbf{b}_j^{n-j}(c)$, $j = 0,\ldots,n$, respectively, as shown in the de Casteljau scheme for cubic Bézier curves in Figure 2.

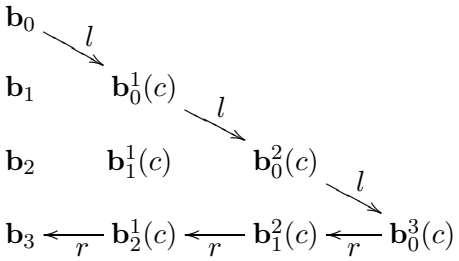


Fig. 2. The control points of the left (l) and the right (r) cubic Bézier curves obtained after subdivision. The arrows indicate the ordering of the corresponding control points

If we repeat the subdivision process, after k subdivision levels we obtain 2^k Bézier polygons which converge to the initial curve as $k \to \infty$. The convergence is very fast and for this reason the subdivision algorithm has many practical applications. For example, it can be used for finding the intersection points of two Bézier curves.

3. Overview of our approach. We have created an interactive Java applet using the graphic library Swing ([4]) for the user interface and JOGL and Java for the implementation of the algorithms. Our main purpose was to visualize 3D Bézier curves and to demonstrate their basic properties and applications. In addition we have used the applet for evaluation of the efficiency of different algorithms and approaches for representation and manipulation of Bézier curves.

3.1. Description of the applet. *An applet* is an application (mini-program) that performs one specific task for a larger computer program, often as a plug-in. *A Java applet* is an applet in the form of a compiled Java program that runs only under a web browser. When the applet is activated through a web browser, the executable file is downloaded automatically from a web server to the client's system. Then it is executed by a Java Virtual Machine (JVM).

Some advantages of using Java applets are:

– Java applets are supported by most web browsers and they are platform independent. Java applets can be executed, e.g., in Microsoft Windows, Unix, Mac OS and Linux.

– Most web browsers cache applets, so when returning to a previously visited web page, the applet loads quickly. Besides, applets improve with use: after a first applet's run, the JVM is already running and starts quickly (unless the browser was restarted).

– Java applets are executed in a sandbox by most browsers and do not have access to the client's local file system. This is an advantage with respect to security. Actually, this restriction can be overcome by using signed applets which work with more rights.

Some disadvantages are:

– The client's system must have Java plug-in installed.

– Despite the use of Java archive (jar) files, the jar file can still be too large, which slows down its downloading and execution significantly. This is the reason why we didn't include in our Java applet more algorithms for Bézier curves.

The main features of our applet are:

- 3D control points can be chosen arbitrarily on the screen using the mouse.
- The applet visualizes the corresponding Bézier curve using either a de Casteljau algorithm or the Bernstein polynomial representation as chosen by the user.
- The curve can be edited visually by moving the control points and rotation.
- The curve can be subdivided interactively using a slider for the parameter t.
- Letters can be drawn using cubic Bézier curves.

More details about the applet's functionality are presented in Section 4. The applet is available and can be tested at the internet address
`www.fmi.uni-sofia.bg/fmi/companal/krassivl/3DBezier/BezierCurve.html`.
The necessary graphics libraries download automatically after starting the applet.

3.2. Why JOGL. Open Graphics Library (OpenGL) ([8]) is probably the application programming interface (API) most used in industry for 2D and

3D computer graphics. This is due to its wide accessibility and compatibility with different operating systems and computer platforms.

JOGL is a graphic library that allows OpenGL to be used in the Java programming language. JOGL implements Java bindings for OpenGL and provides a platform for 3D graphics applications in Java. JOGL also provides full access to OpenGL functions. Using JOGL significantly improves the visualization.

4. Description, performance and results. The applet works in two modes, *Demonstration* → {*Bézier Curve* or *Letter Drawing*}:

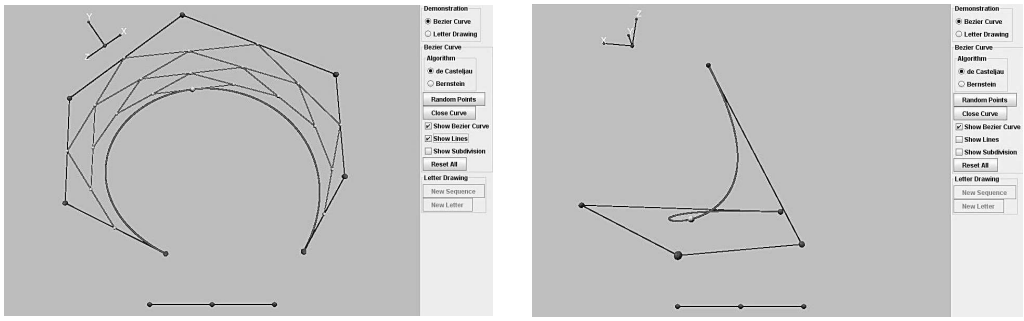
- *Bézier Curve* mode is used for visualization of 3D Bézier curves.
- *Letter Drawing* mode is used for creating letters by means of cubic Bézier curves.

After returning to the previous mode the last changes are restored so that we do not lose any information.

4.1. Visualization of 3D Bézier curves. In *Bezier curve* mode the applet provides the following features:

- Control points can be added by clicking on the screen with the left mouse button.
- Editing control points:
  - A point can be translated by pressing, holding and dragging with the left mouse button.
  - The polygon can be rotated by pressing, holding and moving the left/right mouse button on the screen (not on the control point).
  - A 3D polygon is obtained by rotating the current polygon and then moving/generating a control point.
- Anti-aliasing is included.
- A Bézier curve can be visualized in two modes: *Algorithm* → {*de Casteljau* or *Bernstein*}
  - *de Casteljau* mode
    - *Show Lines* visualizes the intermediate points $\mathbf{b}_i^r(t)$ and the intermediate control polygons, see Figure 3a.

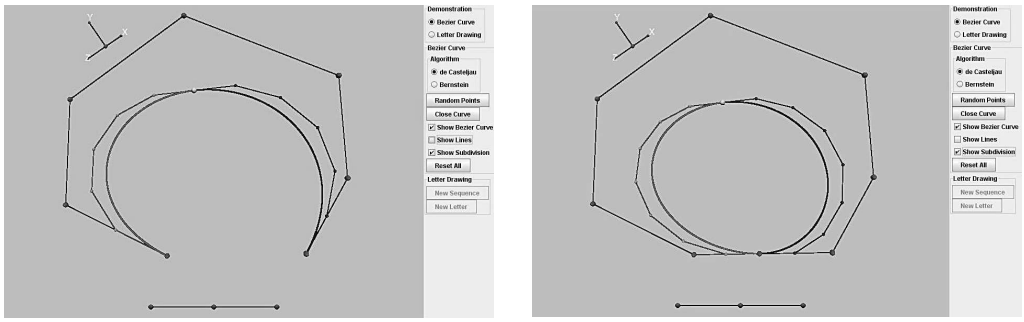
  - ∗ *Subdivision* subdivides the curve and shows in different colors the control polygons of the two curves obtained after the subdivision, see Figure 4a.
- *Bernstein* mode visualizes the Bézier curve using Bernstein polynomial representation, see Figure 3b.



(a) Bézier curve and the intermediate control polygons

(b) 3D Bézier curve

Fig. 3. Bézier curves



(a) Subdivision for $t = 1/2$

(b) Closed Bézier curve

Fig. 4. Subdivision of a Bézier curve

Both algorithms (*de Casteljau* and *Bernstein*) have the following options:

- *Close Curve* closes the current curve so that the resulting curve is smooth, see Figure 4b.
- *Random Points* generates random initial control polygon.
- *Show Bezier Curve* shows/hides the curve.

- The parameter t can be adjusted from a slider in the lower part of the applet.
- A random control point can be deleted by clicking on the point with the right mouse button.
- *Reset All* clears the screen.

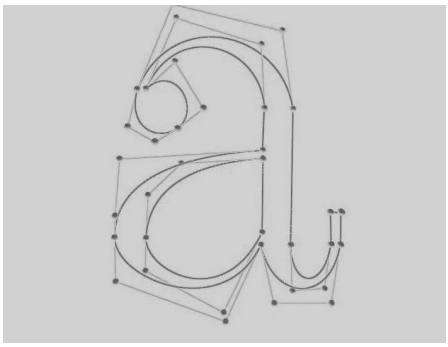


Fig. 5. Representation of the letter **a** using cubic Bézier curves

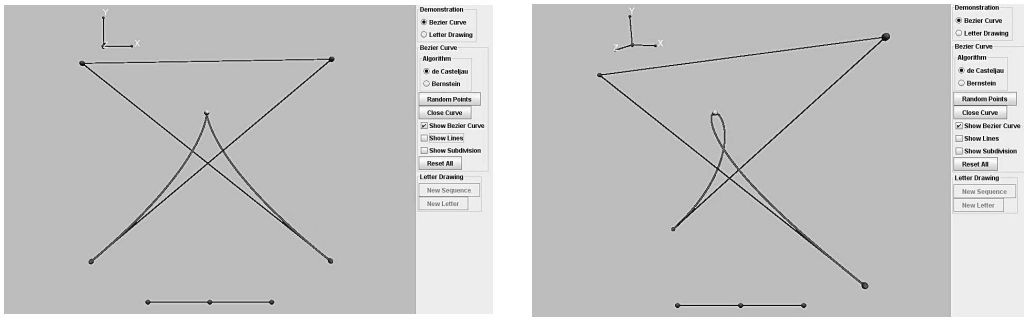
4.2. Creating letters using cubic Bézier curves. Perhaps the most important application of the Bézier curves is in font design. Representing letters using Bézier curves makes their scaling, rotation, translation very fast and easy. In our applet we have included an option to demonstrate the representation of letters as it is done in PostScript, i.e., by means of cubic Bézier curves, see Figure 5.

We use the *Letter Drawing* mode which has the following main features:

- The last control point of a cubic Bézier polygon is an initial control point of the next polygon.
- *New Sequence*: A single sequence of polygons can be insufficient for representing some letters, e.g., O, A, etc., hence an option for begining a new sequence of polygons is included.
- *New Letter* clears the screen.
- A random point can be deleted with the right mouse button but if the point is not the last for the current polygon, the whole polygon will be removed. In this case the next polygon from the sequence automatically moves to its place.

To visualize the cubic curves we use the faster algorithm with Bernstein polynomial representation. In Figure 5 the letter **a** is shown. A detailed user guide including more images from the applet's performance (e. g., Latin and Cyrillic letters—lowercase and capital) can be found at the Internet address

`www.fmi.uni-sofia.bg/fmi/companal/krassivl/3DBezier/UserGuide.pdf`.



(a) 2D cubic Bézier curve (b) 3D cubic Bézier curve

Fig. 6. 3D cubic Bézier curve cannot have a cusp

5. Conclusions and future work. We have implemented and presented an interactive Java applet using JOGL for visualization of some fundamental algorithms for 3D Bézier curves. The applet has an option for demonstration of drawing letters by means of cubic Bézier curves. The application is useful for visual demonstration of basic concepts and important properties of the Bézier curves. For example, the option *Algorithm* $\rightarrow$ {*de Casteljau* or *Bernstein*} can be used to demonstrate the rate of convergence of both algorithms. If we choose a control polygon with more than, say, 14 points, and try to move one of them or rotate the polygon, then it is seen clearly that the curve moves faster when the algorithm with Bernstein polynomials is used. Another important property that can be shown visually is that a 3D cubic Bézier curve cannot have a cusp, see Figure 6.

Currently, we have only explored some basic algorithms for Bézier curves. While these curves are by far one of the most popular techniques for curve modeling in CAGD, we have yet to examine more algorithms, e. g., a hodograph of a Bézier curve and degree elevation.

REFERENCES

[1] Boehm W., A. Müller. On de Casteljau's algorithm. *Computer Aided Geometric Design*, **16** (1999), 587–605.

[2] Chen J. X., E. J., Wegman. Foundations of 3D Graphics Programming Using JOGL and Java3D. Springer, 2006.

[3] Davis G. Learning Java Bindings for OpenGL (JOGL), AuthorHouse, 2004.

[4] Eckel B. Thinking in Java. Prentice Hall, 2003.

[5] Farin G. Curves and Surfaces for CAGD: A Practical Guide. 5th edition, Morgan-Kaufmann, 2002.

[6] Farin G., J., Hoschek, M.-S., Kim Eds. Handbook of Computer Aided Geometric Design. Elsevier, 2002.

[7] Prautzch H., W.,Boehm, M.,Paluszny. Bézier and B-Spline techniques, Springer, 2010.

[8] OpenGL, `http://www.opengl.org`, July, 2011

[9] Java applets, `http://java.sun.com/applets`, July, 2011

Krassimira Vlachkova, Marina Boikova
Faculty of Mathematics and Informatics
St. Kl. Ohridski University of Sofia
5, J. Bourchier Blvd
1164 Sofia, Bulgaria
email: `krassivl@fmi.uni-sofia.bg`
email: `marina_boikova@yahoo.com`

Received July 27, 2011
Final Accepted December 22, 2011