TP1 : Bézier curves, De Casteljau's algorithm

Tibor Stanko

February 5, 2016

1 Bézier curves

A degree n Bézier curve takes the form

$$\mathbf{x}(t) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t) \qquad t \in [0,1]$$

where

$$B_i^n(t) = \binom{n}{i} (1-t)^{n-i} t^i$$

are the degree n Bernstein polynomials, and the binomial coefficients are defined as

$$\binom{n}{i} = \frac{n!}{(n-i)!i!}.$$

The Bézier points $\mathbf{b}_i \in \mathbb{R}^d$ form the *control polygon*.

2 De Casteljau's algorithm

- input: Bézier points $\mathbf{b}_i$ for $i = 0, \ldots, n$, and parameter $t \in [0,1]$.
- output: The point $\mathbf{b}_0^n$ on the curve.
- set $\mathbf{b}_i^0 = \mathbf{b}_i$ and compute the points

$$\mathbf{b}_i^k(t) = (1-t)\mathbf{b}_i^{k-1} + t\mathbf{b}_{i+1}^{k-1} \qquad \text{for} \qquad k = 1, \ldots, n, \quad i = 0, \ldots, n-k.$$

The De Casteljau's algorithm provides an efficient means for evaluating a Bézier curve $\mathbf{x}(t)$. It is useful to look at this algorithm in its schematic form. For a quartic curve ($n = 4$):

$$\begin{array}{lccccccccc}
\mathbf{b}_0 = \mathbf{b}_0^0 & & & & & & & & & \\
 & \ddots & & & & & & & & \\
\mathbf{b}_1 = \mathbf{b}_1^0 & \ldots & \mathbf{b}_0^1 & & & & & & & \\
 & \ddots & & \ddots & & & & & & \\
\mathbf{b}_2 = \mathbf{b}_2^0 & \ldots & \mathbf{b}_1^1 & \ldots & \mathbf{b}_0^2 & & & & & \\
 & \ddots & & \ddots & & \ddots & & & & \\
\mathbf{b}_3 = \mathbf{b}_3^0 & \ldots & \mathbf{b}_2^1 & \ldots & \mathbf{b}_1^2 & \ldots & \mathbf{b}_0^3 & & & \\
 & \ddots & & \ddots & & \ddots & & \ddots & & \\
\mathbf{b}_4 = \mathbf{b}_4^0 & \ldots & \mathbf{b}_3^1 & \ldots & \mathbf{b}_2^2 & \ldots & \mathbf{b}_1^3 & \ldots & \mathbf{b}_0^4 = \mathbf{x}(t)
\end{array}$$

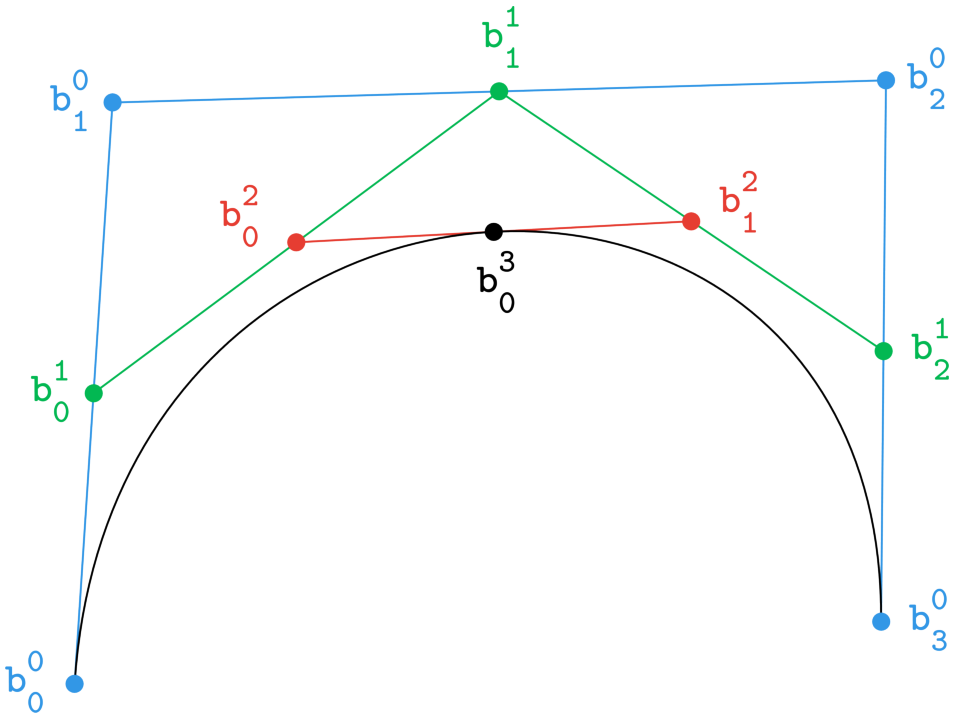

Figure 1: Visualisation of the steps of the De Casteljau's algorithm.

3 Code

```
git clone https://github.com/bbrrck/geo-num-2016.git
cd geo-num-2016/TP1
mkdir build
cd build
cmake ..
make ./geonum_TP1
```

For rendering, you can use gnuplot or matplotlib. While still in the `build/` directory, test them by running

```
gnuplot -p ../plots/plot.gnu
python ../plots/plot.py
```

Many of you have reported problems with gnuplot due to the line set terminal qt. Change it to something else to make things work, e.g. set terminal x11. For a complete list of terminals available on your machine, execute

```
echo 'set terminal' | gnuplot
```

4 ToDo

1. Implement the computation of a curve point $\mathbf{x}(t)$ using Bernstein polynomials.
2. Implement the De Casteljau algorithm for a parameter t.
3. Evaluate the curve using both methods and compare their performance the De Casteljau algorithm for various sampling densities.
4. Visualise the curve and its Bézier polygon. Use all input files from the data/ folder.
5. Visualise the intermediate polygons $\mathbf{b}_i^k$ from the De Casteljau algorithm for a fixed parameter t. (Only the simple.bcv is enough.)

5 Resources

- Handbook of CAGD, edited by Gerald Farin, Josef Hoschek, Myung-Soo Kim
- A Primer on Bézier Curves by Pomax
- Bézier Curves and Picasso by Jeremy Kun
- Bézier Curves and Type Design: A Tutorial by Fábio Duarte Martins
- The Bézier Game
- Bézier Curve Simulation