Homework - 13

CS594

Bézier curves and De Casteljau's algorithm

So far we have confined our attention to piecewise linear geometry, i.e., line-segments and polygons. While this type of geometry is well-suited for computer displays, it is inadequate for design of parts, machines, automobiles, etc., where one needs to define smooth shapes with precision. Polynomials are apt for representing smooth shapes in computers. However, the well-known power basis, comprising of $\{1, x, x^2, \cdots, x^n\}$ for polynomials of degree n, is not convenient for a number of reasons. For instance, the shape of a polynomial defined using power basis does not correlate with that of the coefficients. A different basis, called the Bernstein basis, solves this issue, besides having a number of other useful and interesting properties. Curves defined using Bernstein basis are called Bézier curves. This assignment includes two video tutorials and one article which attempt to explain Bézier curves.

In this video tutorial, the author introduces the concept of *blending*, i.e., taking convex combination of points, curves, etc. Two points p_0, p_1 may be blended using a real number $t \in [0,1]$ as $l(t) = (1-t)p_0 + tp_1$. It is easy to see that the set of points $\{l(t) | t \in [0,1]\}$ forms a line-segment with p_0, p_1 as its endpoints. This concept is generalized to yield Bézier curves by blending more than two points, as explained in the video. In this video tutorial, the author explains the relation between the power basis, which we mentioned in the above paragraph, and the Bernstein basis. Also note how the resulting curve roughly follows the shape of the coefficients in the case of Bernstein basis but not in case of the power basis.

This article by Dr. C. K. Shene explains the concept of Bézier curves in a rigorous manner and also explains De Casteljau's algorithm for evaluating points on Bézier curves. Read the first four subsections in Unit-5: Bézier curves.

Exercise: The exercise involves implementing De Casteljau's algorithm. Write code to read a planar Bézier curve from a file and display it on the screen. While a Bézier curve is smooth by construction, in order to display it on the screen we use a piecewise linear approximation. This may be done by sampling a number of points on the curve and drawing a *line-strip* in OpenGL, i.e., a connected sequence of line-segments, from the first point to the last point. The code should do the following: read the curve from the input file, evaluate the curve at the required number of points using De Casteljau's algorithm and finally, display a line-strip at sampled points on the screen. The input file should contain the degree of the curve, the number of samples for display and a list of control-points in $\mathbb{R}^2$ as the coefficients of the curve. For instance,

$$\begin{aligned} &2, 50 \\ &0.57, -0.17 \\ &0.33, 0.5 \\ &-0.4, 0.89 \end{aligned}$$

In the above example, the degree is 2, i.e., it is a quadratic Bézier curve and thus, has three control points which appear one per line, in the format x, y. The number 50 is the number of samples to use for display. So, in this example, the code will evaluate the given curve C at points $\{C\left(\frac{i}{50}\right) | 0 \leq i \leq 50\}$ and use the points to draw a line-strip in OpenGL. You must also draw the control-polygon for the curve, which is obtained by drawing a line-strip using the control-points. Two examples are shown in Figure 1, wherein the curves are displayed in black while the control-polygon in red. The control-points are shown in blue. Your code does not need to display control-points. You may edit the code for Tutorial-2 of OpenGL (red triangle). There, instead of using `GL_TRIANGLES`, use `GL_LINE_STRIP` to draw a line-strip. Submit C++ code, one sample input curve file and an image containing the screen-capture from the display of this curve.

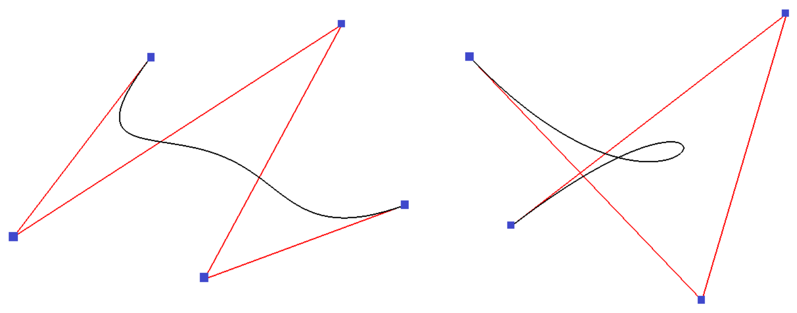

Figure 1: Two examples of Bézier curves with degrees 4 and 3.