Homework - 14

CS594

Bézier curves: Subdivision and degree elevation

The previous assignment introduced Bézier curves as a tool to model freeform shapes. In this assignment we look at some of the related algorithms for Bézier curves such as subdivision and degree elevation. Subdivision refers to cutting a given Bézier curve into two pieces at any given point. Degree elevation refers to finding an equivalent representation of a Bézier curve of degree n, as a degree $n+1$ curve.

We begin by computing derivatives of a given Bézier curve. Read the first two subsections of Unit 4: Parametric curves of this article by Dr. C. K. Shene. We have already covered the first four subsection from Unit 5: Bézier curves. Read the fifth subsection of Unit 5 which deals with computing the derivatives of a Bézier curve.

In many cases it is required to subdivide a given Bézier curve at a particular point. For instance, one may want to retain a part of the curve and discard the remaining part. This may be achieved by subdividing the curve which yields two segments of the original curve which coincide with the original curve. Read the sixth subsection of Unit 5.

It is easy to see that polynomials of degree n form a subset of polynomials of degree $n+1$. Hence it should be possible to represent a given polynomial of degree n as a degree $n+1$ polynomial. While this is trivial in case of the power basis, it needs some computation for the Bernstein basis. Read section seven of Unit 5 which discusses degree elevation of Bézier curves.

Exercise: The exercise involves implementing subdivision and degree elevation for Bézier curves. Add the following functionality to the code of HW-13. Write a function which takes as input a Bézier curve and a parameter $s \in [0, 1]$. The function must subdivide the input curve at s and return the two resulting curves. The input curve may be read from a file on the disk,

as in HW-13 and the output curves may again be written in separate files. Also display the subdivided curves on the screen, in different colors, along with the original curve. Apply some x or y offset to the new curves so that their display does not overlap with the original curve. Also display the control-polygons for all the three curves.

Add another function which takes a Bézier curve as input and elevates its degree by 1. Again display the two curves on the screen, giving some offset to the output curve to prevent overlap. Also display the control-polygons for both.

Submit the C++ code as well as sample input and output files for one example each, for the above two functionalities.