CSE 5443 Lab 2
Autumn, 2021

Implement an interactive graphics window that allows a user to place n control points in a $[0,1] \times [0,1]$ region and then draws a degree d B-Spline from the control points (once all n points are input.) Your program should have the following features:

1. Ability to interactively enter control control points;
2. Ability to interactively move the control points;
3. "Add Control Point" button to add a control point to the B-spline;
4. Read/write of the curve degree and control points from/to a file.

Program details:

1. Name the program `ibspline`.
2. Input to the program is either:
   (a) Two integers, n and d, with $n \geq d+1$, specifying the number of control points and the B-Spline degree;
   - NOTE: A B-spline of degree d has order $k = d+1$. The text uses the order k, not the degree d, in describing the de Boor algorithm and Boehmn's knot insertion. Be sure not to confuse k and d.

   (b) The name of a file containing the curve degree, knot vector and control points.

   (Your program needs to be able to process both forms of input, an integer or a file name.)
3. When input is (n,d):
   (a) The program should check that $n \geq d+1$;
   (b) The user should interactively specify the location of the n control points by drawing on a window region with a mouse;
   (c) The B-Spline curve has degree d.
4. Your program should construct a CLAMPED B-Spline.
   (a) The knot vector has $n+d+1$ knots;
   (b) The first $d+1$ knots are all 0;
   (c) The last $d+1$ knots are $n-d$;
   (d) Initially, the other knots are $(1,2,3,\ldots,n-d-1)$. With the insertion of more control points, these knots could change so that they would no longer be uniformly spaced.
5. The B-Spline should be created by using the de Boor algorithm to compute a set of suitably close points on the curve and then drawing a polygonal line through those points. Note that de Boor algorithm as described in the text uses the order $k = d+1$ as a parameter, not the degree d.
6. Using the mouse, the user should be able to interactively select any control point and move it. When any control point is moved, the program should automatically recompute and redraw the B-Spline;
7. The graphics window should have an "Add Control Point" (or "Add ControlP" or "Add CP") button for adding a control point to the B-Spline.
   (a) When the "Add Control Point" button is selected, the user should select an existing control point q_h, where $h \geq d$. (Control points are labeled starting at 0, i.e., $q_0, q_1, \ldots, q_{n-1}$.) Requiring h to be at least d ensures that knots are not inserted between two knots with values 0. (The program should prompt to select a control point.)

(b) The program creates a new knot vector with a new knot $t = (t_h + t_{h+1})/2$ between knots t_h and t_{h+1}.

(c) The program replaces the $d-1$ control points $q_{h-d+1}, q_{h-d+2}, \ldots, q_{h-1}$ with d new control points using the Boehm algorithm. (See Theorem 11.5.2.13 in the text or the course notes. If d is 1, the program just adds a control point.) Note that Boehm algorithm as described in the text uses the order $k = d + 1$ as a parameter, not the degree d.

8. The graphics window should have a "Save" button that allows the user to save the control points to a file.

   (a) The file format is similar to lab 1, but has the label "BSPLINE", and includes a line defining the knot vector. (See below.)

   (b) When input to your program is the name of a previously saved file, your program should read the curve degree, the knot vector and control points from the file and use those as the starting point for your program.

9. You can write your program in matlab, python, python jupyter notebooks, Unity, java script, or any other language that easily supports interactive windows.

   (a) For implementations in matlab or python, the code should be a function, `ibspline`. Calling `ibspline` (7,3) prompts for the user to select 7 control points using the mouse, and construct a B-spline of degree 3. Calling `ibspline` ('fname') should read all the control points from file 'fname'.

   (b) For implementations in languages that don't have interpreted environments, you can use the command line format "`ibspline` 3 7" and "`ibspline` fname".

10. DO NOT use any routines in any language that calculate points on a B-spliner curve or control points for a B-spline curve. You need to write your own code based on de Boor's algorithm for doing those calculations.

11. DO NOT look at anyone else's code or let them look at your code. You can discuss the algorithm/program with your fellow students, but don't look at code.

12. Submit file(s) of your code through carmen. Include also a README file that explains how to run your program.

File format

File format for reading, writing the B-spline curve is:

1. First line, text: BSPLINE
2. 0 or more comment lines: Each comment line starts with a '#'
3. {ndim } {nump } {ndegree }
   where

   (a) ndim is the dimension of the points, i.e., the dimension of the curve containing the points. For this lab, ndim is always 2.

   (b) nump is the number of control points.

   (c) ndegree is the degree of the B-Spline curve.

4. Single line containing a list of nump + ndegree values representing the knot vector:

   - $\{t_0\}\ \{t_1\}\ \{t_2\}\ \ldots\ \{t_{\mathsf{nump}+\mathsf{ndegree}-1}\}$

   where $t_i \leq t_{i+1}$ and $t_0 = t_1 = \ldots = t_{\mathsf{ndegree}} = 0$ and $t_{\mathsf{nump}} = t_{\mathsf{nump}+1} = \ldots = t_{\mathsf{nump}+\mathsf{ndegree}-1}$. The list represents the knot vector $(t_0, t_1, \ldots, t_{\mathsf{nump}+\mathsf{ndegree}-1})$ where

5. List of control point coordinates, one point per line. For each point $p^i = (p^i_x, p^i_y)$:

   - $\{p^i_x\}\ \{p^i_y\}$

   (The number of coordinates per line is ndim. For this lab ndim equals 2, so there are only 2 coordinates.)

Sample file text

```
BSPLINE
# Sample file containing Bspline control points
# {dimension} {number of points} {degree}
2 7 2
0 0 0 1 2 3 4 5 5 5
0.15 0.2
0.20 0.2
0.30 0.8
0.34 0.6
0.42 0.7
0.63 0.5
0.7  0.3
```

Extra Credit

Add four buttons “Uniform”, “Clamp”, “Close”, “Open”.

1. “Pressing” the “Uniform” button replaces the current knot vector with the uniform knot vector $(0, 1, 2, \ldots, n+d)$ where n is the current number of control points and changes to open mode where the curve is drawn for $u \in [t_d, t_n]$.

2. “Pressing” the “Clamp” button replaces the first d knots $t_0, t_1, \ldots, t_{d-1}$, with the value t_d and the last d knots $t_{n+1}, t_{n+2}, \ldots, t_{n+d}$, with t_n and changes to clamped mode where the curve is drawn for $u \in [t_0, t_n]$.

3. “Pressing” the “Close” button creates a closed B-spline defined on $u \in [t_d, t_{n+d}]$, defined as follows:

   (a) For each $i \in [n+1, n+d]$, set $t_i = t_n + (t_{i-n} - t_0)$. (Essentially, make the gap between t_i and t_j, $j < n$, to be the sum of the gap from t_j to t_n and the gap from t_0 to t_{i-n}.)

   (b) For $u \in [t_i, t_{i+1}]$ where $d \leq i \leq n$, the curve is defined as for an open B-spline.

   (c) For $u \in [t_i, t_{i+1}]$ where $n \leq i < n+d$, the curve is defined as for an open B-spline with control points: $(q_0, q_1, \ldots, q_{n-1}, q_0, q_1, \ldots, q_{n-1}, q_0, q_1, \ldots, q_{d-1})$,
   and knot vector: $(t_0, t_1, \ldots, t_{n+d}, t_{n+d+1}, \ldots, t_{n+2d})$,
   where $t_{n+2d} = t_{n+2d-1} = \ldots t_{n+d+1} = t_{n+d}$.

   (d) Typically, one first presses the “Uniform” button and then the “Close” button. Pressing the “Close” button should not change the knots $t_0, t_1, \ldots, t_n$.

4. “Pressing” the “Open” button changes to open mode, where the curve is drawn only for $u \in [t_d, t_n]$.

5. When an open or closed (not clamped) B-spline is saved to a file, the knot vector will NOT have the properties that $t_0 = t_1 = \ldots = t_d$ and $t_n = t_{n+1} = \ldots = t_{n+d-1}$.

   Therefore, when reading from a file, the program must determine if the knot vector is clamped, i.e. if $t_0 = t_1 = \ldots = t_d$ and $t_n = t_{n+1} = \ldots = t_{n+d-1}$. If the knot vector is not clamped, assume the curve is open and defined on $u \in [t_d, t_n]$.

6. The button “Add Control Points” should work in each of the three modes, open, clamped or closed, using Boehm algorithm.

7. Implementing “Add Control Points” to work properly in the closed mode is particularly tricky. Note that selecting q_i to add a knot in $[t_i, t_{i+1}]$ for $i < d$, requires also changing some of the knots t_i in the range $i \in [n, n+d)$ and replacing some of the control points $q_{n-d}, q_{n-d+1}, \ldots, q_{n-1}$.

Implementation of the “Clamp” and “Uniform” buttons (with proper implementation of “Add Control Points”) is worth up to an extra 5% of this lab grade.

Implementation of the “Clamp”, “Uniform”, “Closed” and “Open” buttons (with proper implementation of “Add Control Points”) is worth up to an extra 10% of this lab grade.