CSE 5443 Lab 1
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 $n-1$ Bezier curve from the control points (once all n points are input.) Your program should have the following features:

1. Ability to interactively enter Bezier control points;
2. Ability to interactively move the control points;
3. Subdivide button for subdiving the Bezier curve;
4. Read/write of the curve degree and control points from/to a file.

Program details:

1. Name the program `ibezier`.
2. Input to the program is either:
   (a) An integer $n \geq 2$ specifying the number of control points;
   (b) The name of a file containing the curve degree 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 an integer n:
   (a) The user should interactively specify the location of the n control points by drawing on a window region with a mouse;
   (b) The Bezier curve has degree $n-1$.
4. The Bezier curve should be created by using de Casteljau's algorithm to compute a set of suitably close points on the curve and then drawing a polygonal line through those points.
5. 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 Bezier curve;
6. The graphics window should have a "Subdivide" button that splits the Bezier curve of degree $n-1$ into two Bezier curves at $u = 1/2$, each controlled by n control points. (The middle control point is shared by both Bezier curves.)
   (a) Use de Casteljau's subdivision algorithm to subdivide the Bezier curve and create the new set of control points.
   (b) All the resulting control points should be displayed and movable by the user. (Note that when control points are moved, the curve may no longer be smooth at the point shared by the two Bezier curves.)
   (c) A second call to "Subdivide" should split each of the two Bezier curves into two, creating four Bezier curves.
   (d) Further calls to "Subdivide" should split each current Bezier curve into two, creating 8, then 16, then 32 Bezier curves, and so on.
7. The graphics window should have a "Save" button that allows the user to save the control points to a file.
   (a) See below for file format.
   (b) When input to your program is the name of a previously saved file, your program should read the curve degree and control points from the file and use those as the starting point for your program.

8. You can write your program in matlab, python, python jupyter notebooks, Unity, java script, or any other language that easily supports interactive windows.
   (a) Examples of matlab and python code, `iplotpoly`, creating interactive windows to draw, modify, subdivide and save polygon lines are provided in carmen. If you have no or little experience with interactive graphics, I suggest basing your programs on those examples. (The provided examples include a Rotate function. You do not have to include a Rotate function in your lab. You are free to do so, but there is no extra credit for doing so.)
   (b) For implementations in matlab or python, the code should be a function, `ibezier`. Calling `ibezier` (5) prompts for the user to select 5 control points using the mouse. Calling `ibezier` ('fname') should read all the control points from file 'fname'.
   (c) For implementations in languages that don't have interpreted environments, you can use the command line format "`ibezier` 5" and "`ibezier` fname".
9. DO NOT use any routines in any language that calculate points on a Bezier curve or control points for a Bezier curve. You need to write your own code based on de Casteljau for doing those calculations.
10. 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.
11. 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 bezier curve is:

1. First line, text: BEZIER
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 Bezier curves.
   (d) Note that the nump should equal (ndegree$*k+1$) for some integer k where k is the number of Bezier curves.
4. List of control point coordinates, one point per line. For each point $p^i = (p^i_x, p^i_y)$:
   (a) $\{p^i_x\}$ $\{p^i_y\}$

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

Note that the provide matlab and python examples of `iplotpoly` DO NOT write files in the "BEZIER" format. Even if you program in matlab or python, you will need to write your own routines for reading or writing in "BEZIER" format.

Sample file text

```
BEZIER
# Sample file containing Bezier control points
# {dimension} {number of points} {degree}
2 4 3
0.2 0.2
0.3 0.8
0.6 0.7
0.8 0.3
```

Extra Credit

1. Add a toggle button "Smooth".
2. The toggle button "Smooth" affects how control points move after the Bezier curve has been subdivided into two or more Bezier curves.
3. When "Smooth" is selected and the Bezier curves have degree d:
   (a) If q_i is the d'th control point in a Bezier curve, then moving q_i also moves q_{i+2} so that the curve remains smooth at point q_{i+1};
   (b) If q_i is the second control point in a Bezier curve, then moving q_i also moves q_{i-2} so that the curve remains smooth at point q_{i-1};
   (c) If q_i is the last control point of one Bezier curve and the first control point of another one, then moving q_i also moves q_{i-1} and q_{i+1} so that the curve remains smooth at point q_i.

   Note that if the Bezier curves have degree d, then each Bezier curve is controlled by $d + 1$ control points.
4. There is some freedom in how to move the other points based on q_i. Determine a "good" way to do so, and document in your README file. (No "correct" solution but solutions that have the most "natural" behavior are preferable.)
5. Once a call to "Subdivide" has been made, do not allow toggle button "Smooth" to be changed. (This avoids the problem of what to do if a non-smooth curve is created using subdivision, and then "Smooth" is selected.)

Implementation of the "Smooth" button is worth up to 10% of this lab grade.