CPSC 424 Assignment 3 — Programming

Term: September 2022, Instructor: Alla Sheffer, sheffa@cs.ubc.ca
`https://www.students.cs.ubc.ca/~cs-424`

Due: Oct 21, 2022 at 11:59PM

In this assignment you are to implement 2D Bézier curves in a WebGL program. You are provided with javascript&html source code for starting a simple WebGL program and moving some points around on the screen. You are also provided with a basic vector function to get you started faster. Finally, a reference video is available. You should refer to the video for clarification of any ambiguities in this assignment specification.

You will implement Bézier curve evaluation, the subdivision algorithm, and piecewise Bézier curves.

We have provided a simple user interface for manipulating curves and changing modes. You should be able to create and move control points by clicking and dragging with the mouse. This functionality is provided in `main.js`.

There are three options under the "Mode" submenu: "Basic", "Subdivision", and "Piecewise". These correspond to the three components of the assignment you will need to implement.

- In Basic mode, a Bézier curve is drawn using all the control points on the canvas.
- In Subdivision mode, the Bézier curve is approximated using the subdivision algorithm. You can set the subdivision level by the input box.
- In Piecewise mode, the control points are interpolated/approximated using piecewise Bézier curves of a chosen degree. You can set the degree of the piecewise curves by inputting numbers to textbox.

In Piecewise mode, the continuity of the curves can be set in the "Continuity" menu. We have provided menu options for C0 and C1 continuity, although only C0 continuity is mandatory for the assignment.

Finally, there is a button to clear all points from the canvas.

Assignment 3.1: Bézier Curve Evaluation (5 Points)

Implement simple evaluation and rendering of Bézier curves. You are provided with an incomplete `Bezier` function in file `bezier.js`. Implement `evaluate()` to return the point along the Bézier curve at the given t-value. We have provided a function `nChooseK()` to compute $\binom{n}{k}$. You may also find it useful to use `Math.pow()` to compute exponential expressions. Note that these computations are prone to overflow in the presence of large numbers. As such, we do not expect that your code be robust for Bézier curves of arbitrarily high degrees.

Next, implement the section of `drawCurve()` labelled "Basic Mode" in file `bezier.js`. You should draw the curve by evaluating it at uniformly-spaced t-values, and then drawing lines between those points. We have provided a function `drawLine()` to do the line-drawing for you.

To draw the line you need to uniformly sample the parameter domain of your curve, with the number of samples set to the *samples* variable in bezier.js which can be controlled from the graphics interface. Test the impact of increasing or decreasing this number on your output and program speed. Make sure your code does not lag when this value is large.

Assignment 3.2: Subdivision (5 Points)

Implement `subdivide()` in file `bezier.js`. Follow the subdivision algorithm described in class to produce two `BezierCurve` objects that have control polygons representing the two halves of the initial `BezierCurve`. For the purpose of this assignment, subdivision is always at the center of the parameter interval.

Next, implement the section of `drawCurves()` labelled "Subdivision Mode". You should start with a `BezierCurve` composed of the full set of points. Then subdivide the curve repeatedly until the number of subdivisions is equal to the `subdivide_level` variable. Once all subdivisions have been performed, draw the control polygons of the resulting curves using `drawControlPolygon()`. The control polygons should converge to the true Bézier curve.

You should be able to subdivide the control polygon any number of times, like the sample implementation. **There should be no hard limit on the number of subdivision steps that can be performed**.

Assignment 3.3: Piecewise Curves (5 Points)

Implement the section of `drawCurves()` labelled "Piecewise Mode". Create $\lceil (N-1)/d \rceil$ `BezierCurves` where N is the total number of control points and d is the degree of each piecewise curve. Use the `piecewise_degree` variable to determine what the degree of the Bézier curves will be (the final curve may need to have lower degree than the others). The curves must have at least C0 continuity. You may earn bonus points by implementing C1 continuity (see below). Note that this will likely require adding additional control points in the code to ensure that the C1 condition is enforced.

Assignment 3.4: Extensions (2 Points)

You can get up to two bonus points for this assignment (i.e. the maximum grade is 17 out of 15!). Extra points will carry over so you will be able to use these points for compensating for points you lose on the midterms, final, or other homework. Examples for what you can do are:

- Smooth transitions between multiple Bézier curves, i.e., C1 continuity.
- Animation/Implementation of the De Casteljau evaluation algorithm (this will require tweaking the UI accordingly).
- Whatever else you can think of.

Bonus points are given at the discretion of the TAs.

Getting Started

You should first download the provided template and unpack it. The sample demo is under the folder `\demo`. Take a look at it to get a feeling for what you are about to be doing.

The initial version of the code contains all user interface components and point manipulation mechanics. It does not draw any curves until you begin implementing the assignment.

You should not need to modify any files except for `bezier.js`. You are welcome to modify anything you like while implementing your extensions (though you must preserve all functionality from the mandatory components of the assignment if you want to receive full marks).

Submission

In addition to your source code, you should also have a `README` file that details your name, cs account, student number and how far you have gotten in your implementation. You should also mention any extensions you may have implemented.

Once you have all your files together, submit them with the `handin` command that you should know from other courses:

```
handin cs-424 a3
```

You can also find the instructions for `handin` at

`https://my.cs.ubc.ca/docs/handin-instructions`