Assignment 5: Bézier Curves
CS180/CS280 Fall 2022

Professor: Lingqi Yan

University of California, Santa Barbara

Assigned on Nov 8, 2022 (Tuesday)

Due at 23:59 on Nov 14, 2022 (Monday)

Notes:

- Be sure to read the Programming and Collaboration Policy on course website.
- Any updates or correction will be posted on EdStem, so check there occasionally.
- You must do your own work independently.
- Read through this document carefully before posting questions on EdStem.
- Submit your assignment on GauchoSpace before the due date.

---

1 Overview

Bézier curves are parametric curves that are used in computer graphics. In this assignment, you are expected to implement the **de Casteljau's** algorithm for drawing a Bézier curve, defined by 4 control points (although once you've implemented the algorithm correctly, there's no reason why it shouldn't work with more).

The functions you should modify are given in the **main.cpp** file.

- `bezier`: This function implements the basic functionality of implementing the Bezier curve. It takes a set of points in pixels and an `OpenCV::Mat` object as inputs, and doesn't output anything. It iterates through `t` inside a range from 0 to 1, and increases it just a little in each iteration. For each `t` that's calculated, the other function `recursive_bezier` is called, and that function returns the point that's on the Bezier curve at point `t`. Finally, that returned point is painted on the OpenCV::Mat object.
- `recursive_bezier`: This function takes a set of points (control points), and a float `t`, and implements the **de Casteljau**'s algorithm to find the point that's on the Bezier curve.

2 Algorithm

De Casteljau algorithm is explained as following:

1. Consider a Bezier curve with control points p_0 ... p_n. You first connect the consecutive points to have line segments.
2. Subdivide each line segment with the ratio $t : (1 - t)$, and find the points at the division.
3. These points are your new control points, and their number will be one fewer than the previous step.
4. If you have only one point, then return that point and terminate. Otherwise, go to step 1 with current points.

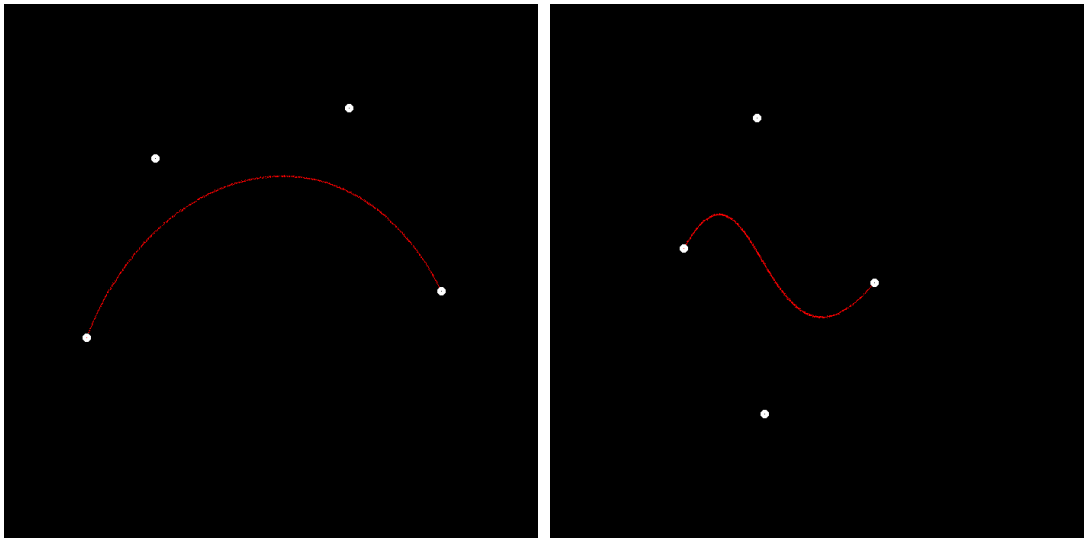
If you apply this algorithm for different **t**'s between [0, 1], you will get the corresponding Bezier curve.

3 Getting started

In this assignment, you will have a new code base, that is a lot smaller than the previous ones. Similar to the previous assignments, you can choose to work either on your own computers or on the virtual machine. Please download the project's skeleton code, and build the project like we did previously by using the following commands:

```
$ mkdir build
$ cd ./build
$ cmake ..
$ make -j
```

After this, you should be able to run the given code by using `./BezierCurve`. When it is running, the program will open a black window. Now, you will be able to click onto the window to place points. The program will wait for you to place 4 points in the window, then it will automatically draw a Bezier curve based on these points you place. The given implementation computes the Bezier curve by using its polynomial equation. This Bezier curve is drawn in red to the window. The curve with some chosen points are like this:



After making sure that everything is working, now you can start on your own implementation. Comment out the line where the `naive_bezier` function is called inside the while loop in `main` function, and uncomment the `bezier` function instead. Your Bezier curve will be drawn in green to the window.

If you want to make sure your program is working correctly, call both the `naive_bezier` and `bezier` functions, and if your implementation is correct, both should write to the approximately same pixels, therefore the line should appear **yellow**. If so, you can make sure that your implementation is correct.

You can also try to modify the code and experiment with different numbers of control points and see different Bezier curves.

4 Grading and Submission

Grading:

1. (5 points) Submission is in the correct format, include all necessary files. Code can compile and run.
2. (20 points) De Casteljau's Algorithm:
   Given the control points, your code should be able to produce the correct Bezier curve.
3. (Bonus 5 points) Bonus points:
   Implement anti-aliasing on the Bezier curve. (When you find a point that lies on the curve, instead of putting it directly to a single pixel, you need to splat it to the neighboring pixels according to the distances to the pixel centers.)

Submission:
After you finish the assignment, clean your project, remember to include the CMakeLists.txt in your folder, whether you modified it or not. Add a short report in the directory, write down your full name and perm number inside it. **Please indicate whether you have implemented the bonus points.** Then compress the entire folder and rename it with your name, like "Goksu.zip". Submit **ONLY ONE .zip** file on GauchoSpace. **Please make sure to delete the build folder before you submit.**