

CSCI 441 - Lab 03  
Friday, September 13, 2019  
LAB IS DUE BY **FRIDAY SEPTEMBER 20 11:59 PM!!**

Today, we'll add some new features to our OpenGL Toolbox and display some nifty looking curves.

### Step 1 – Loading Our Control Points

We are going to eventually be drawing a Bezièr curve. As discussed, Bezièr curves are defined by a series of control points. We will read the control points in from a file to allow greater flexibility in our Bezièr Curve Viewer (the BCV for you folk in the know). `TODO #01` will get us in the right place. Be sure to prompt the user to enter a filename for the data file to open. This allows us to rerun the program with different data files and not have to recompile the code every time.

Now we need to actually read in our data, `TODO #02` will accomplish this for us. Hopefully you figured out already that you need to pass the filename to `loadControlPoints( char* )`. This function will read through the file and create a `glm::vec3` for each line in our file. The control file has the following structure

```
n-points
x1, y1, z1
x2, y2, z2
...
xn, yn, zn
```

The first line is the number of points in the file and then there are `n` lines that follow with one point per line. Read in each point and store it in our `controlPoints` vector. You may assume that the control file will be properly formed, meaning it will contain the proper number of points to draw a complete Bezièr Curve (4, 7, 10, etc.)

There are two control files included with this lab, `controlPoints4.csv` and `controlPoints7.csv`. Compile, run, and test running/loading with each file.

### Step 3 – Draw our Control Point Cage

We want to be able to visualize where our control points are in relation to the final curve. For this step we'll fill in `TODO #03` and `TODO #04`. First for `TODO #03`, draw a green sphere at the location of each control point. Make the sphere an appropriate size relative to our grid size.

Compile and run. Run with the `controlPoints4.csv` file. How many spheres do you see?

Now for `TODO #04`, connect each pair of control points with a yellow line. Refer to how the grid gets drawn as a reminder of how to work with 2D OpenGL Primitives. You'll probably want to make the control cage lines a thickness of 3.0f so they are easier to see.

Compile and run. Run with the `controlPoints4.csv` file. How many yellow lines do you see?

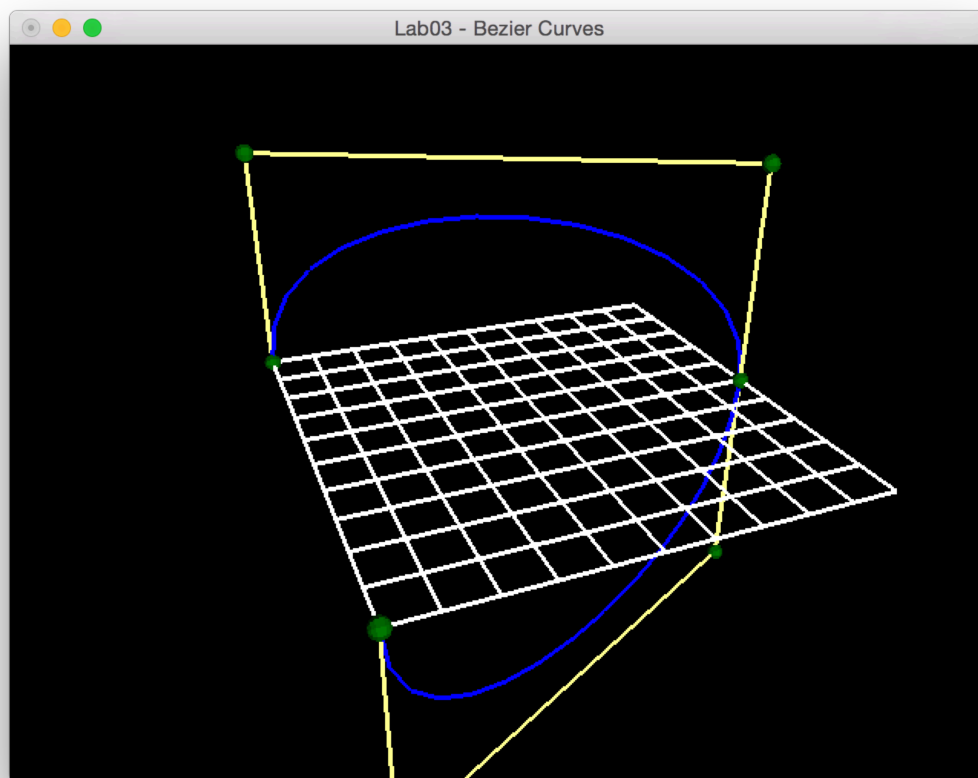
You should have not connected the last point back to the first point, so now we should see our full control point cage. Let's draw the curve!!

## Step 4 – Drawing the Bezièr Curve

Now we are ready to draw the curve. Use a blue line with thickness of 3.0f to draw the curve. You will need to fill in `TODO #05` and `TODO #06` to complete this task. `TODO #05` is responsible for segmenting our curve into lines and connecting the dots. `TODO #06` is responsible for evaluating the Bezièr Curve at a single point.

When testing, start with `controlPoints4.csv`. Once you have a single curve drawn, then test with `controlPoints7.csv`. Your program should be able to handle any number of Bezièr Curves as specified from a file. That is, if your program was run with `controlPoints10.csv`, it should display three curves and not error.

Once your view for `controlPoints7.csv` looks like below....you did it! Hooray!



## EXTRA EXTRA! Extra Credit Achievement



The following section is not required for the lab. If you want to earn an extra credit achievement, then continue. Otherwise, skip to the end to zip up your submission. 😊

If you are feeling extra ambitious and want to earn an Extra Credit Achievement, then complete the Picking Made Easy tutorial included with this lab (the tutorial was originally put together by Oregon State

University. Thank you to them for doing such a great job. I converted their write up from GLUT to GLFW.). We want to be able to click on a control point and move its location. To earn the extra credit achievement, you must allow the user to Shift + Left Click on a control point. If the user clicked a control point, then highlight that control point red. Once the control point is selected, the user can move the control point's location along the X-, Y-, and Z-axis through keyboard presses. If the user Shift + Left Click's and does not select a control point, then deactivate any previously selected control points. Only one control point can be active at a time.

**Q1: Was this lab fun? 1-10 (1 least fun, 10 most fun)**

**Q2: How was the write-up for the lab? Too much hand holding? Too thorough? Too vague? Just right?**

**Q3: How long did this lab take you?**

**Q4: Any other comments?**

To submit this lab, zip together your main.cpp and README.txt with questions. Name the zip file <HeroName>\_L03.zip. Upload this on to Canvas under the L03 section.

LAB IS DUE BY **FRIDAY SEPTEMBER 20 11:59 PM!!**