CS4815 Week08 Lab Exercise

Lab Objective: We will complete our work on the Bézier curve drawing program. We will consider two tasks today: 1) drawing in the lines that connect the points so that we have a better idea of how the curve drawing performs; and, b) dragging some of the points of the curve and watch the curve be updated.

Here's a **quick summary** of the tasks:

❶ Copy the main source file for this week's lab

❷ Modify the program so that we can see the lines that connect the control points

❸ Modify the program so that we can select control points and drag them around the screen; the Bézier drawing should update

❹ Submit your completed program using the handin command

```
handin -m cs4815 -p w08
```

In Detail

❶ This week's lab, although based on Bézier curves again, is independent of last week's. Therefore you may use as a starting point the starting code of last week's lab. However, if you were able to complete last week's lab, for the sake of a more complete program you may want to *add to that* instead of making two sets of incompatible modifications to the same initial file.

When I began thinking about this lab I thought that it would be nice to have you read in an arbitrary set of points from a file via the `argc` and `argv` command-line variables and have you compute the curve based on this set of points. This would be in place of the hard-coded 4 points of the array `ctrlPts` in the function `display()`. However, when I sat down to do the preliminary work for it I soon realised that it would be too complicated to explain to people who were only now learning C++ . So this might be good news for you. Maybe.

Eventhough I am not going to have you base your program on reading in an arbitrary set of points I want you to modify the `bezier.cc` program so that it behaves in this way. By this I mean that you should remove from the function `displayFcn()` the initialisatino of the array of points, `ctrlPts` and its associated size `nCtrlPts`, and make both of these *global variables*. That is, I want your program to "fake" reading points from a file by declaring the points as a global array; if you had more expertise in C++ then it would be possible for you

to write the function that would open a file and fill this array. You should add a few more points to the array to make the drawings more interesting; 6-ish would be an ok number of points.

There is one more change that you should make to the code that relates to faking reading the points from a file. At the top of the program the world-coordinate clipping window dimensions are hard-coded based on the 4 points given. You should generalise this process by writing a function that goes through all of the points of the array – more than 4 of them, please! – and initialises the four extremes. This will be important later.

❷ You should faintly draw in the lines that connect all of your control points. You should set these lines up as a *single* OpenGL primitive declaration, as we have seen in a previous lab. That is, only one `glBegin()` "command" should apply over all the points / lines. I will leave it to your own aesthetic judgement for the colouring, thickness, options, etc. But just remember that sledge-hammers can hurt.

❸ You should now modify your program so that it responds to mouse left clicks when within a "reasonable" distance (radius) of a point. In this case the point should be "selected" and you should be able to drag it and update the Bézier curve. The idea of a reasonable distance should be based on the scaling factors so that with more spread out points there should not be a need for such a tight radius around the points.

❹ Using the `handin` command given at the top of the lab sheet please submit your lab exercise by the usual deadline of Friday, Week09.