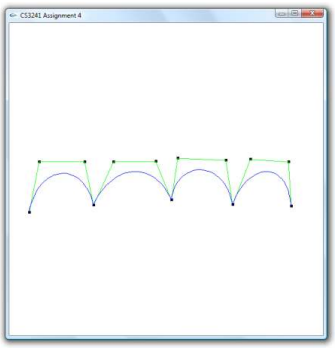
LAB 4: LET'S GET TWISTED

We will draw Bezier curves in this assignment, and some objects along them. You can add control points by left-clicking on the mouse. The points will be added to a list called "`ptList[MAXPTNO]`" and the variable "`nPt`" will record the number of points. Here are the tasks that you are supposed to do:

1. **Control Line**

   In your skeleton program, when you click a new point on the screen, it will be displayed as a black point. Your first task is to draw the control lines (green ones) by joining them. This control lines are being toggled by pressing the key "L" and by the variable "`displayControlLines`".



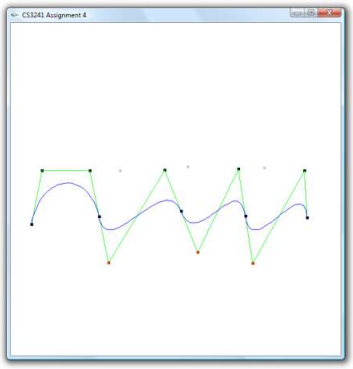
2. **Clear All**

   By pressing the key "E", you should erase all control points *on the screen* and start as new.

3. **Bezier Curves**

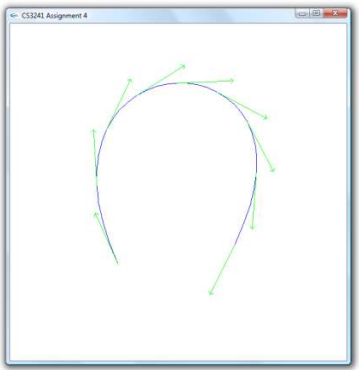
   For every four points, draw a Bezier curve. So, the first 4 points, namely the 1st, 2nd, 3rd, and 4th points of your input, will form the first curve. Then the next curve is continuously formed by the 4th to 7th points (note that the 4th point is shared by two curves, so are the 7th, 10th … etc). The second curve is **NOT** formed by the 5th to 8th points instead. The number of straight line segments to approximate the curve is in the variable `NLINESEGMENT`. It is **NOT** advisable to use the divide-and-conquer paradigm (De Casteljau Algorithm) in this assignment. This will just make things harder for you.

TOGGLED BY "C"



4. **C1 Continuity**

   By pressing the key "C" (Toggle the variable "`C1Continuity`"), you will modify the SECOND point of every curve (except the first one) and make them C1 Continuous. The new second control point will be displayed in red, and the original second control point will be dimed as grey. And this function should be reversible. And all the functions in this assignment should be working in BOTH modes (regardless in C0 or C1 continuity).

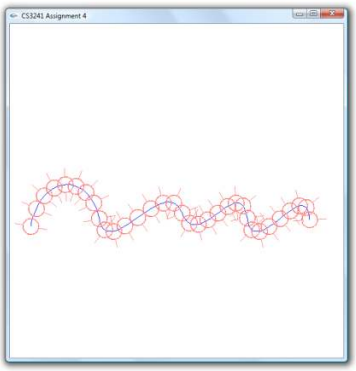


5. **Tangent Vectors**

   Draw the tangent vectors by pressing the key "T" (Toggle the variable "`displayTangentVectors`"). The number of tangent vectors you need to draw is in the variable `NOBJECTONCURVE` (Not `NLINESEGMENT`). There is already a function `drawRightArrow()` to draw an arrow for you.

6. **Draw Objects**

   At the same manner of the tangent vectors, draw `NOBJECTONCURVE` copies of an object of your own creation along the curve (Toggled by "O" and the variable "`displayObjects`"). You can reuse your Assignment 1 drawing. The objects should be drawn on the reference frame with the tangent and the normal vectors on the



curve. In the sample program, the object is simply ONE circle with two lines on the north and south poles of the circle. By repeating it on the curve, it looks like a worm.

7. **Create A Drawing with Bezier Curves**

   Create one beautiful figure (with or without the objects) and submit "`savefile.txt`" together with your submission. (There are two functions "R" and "W" provided for you to read and write your current control points into a file called "`savefile.txt`".) You have a choice to draw your figure using either C0 Continuity or C1 Continuity. Please indicate in the "`readme.txt`" so that we know which mode to look at.

Submission Instruction

Submit a zipped file with the following files: "`main.cpp`", "`savefile.txt`" for your drawing and your "`readme.txt`". Please indicate if you have any extra features that are impressive or any explanation of your program in "`readme.txt`". The zipped file should not be larger than 2MB.

Note for Xcode Users (Mac)

To be able to read and write to and from your "`savefile.txt`", right-click "lab4" under the "Products" folder in your Project Navigator on the left and select "Show in Finder". Copy and paste your "`savefile.txt`" into this folder. Now run the program and press "R". If there are 13 black points on the screen, the setup is complete.

Need more help?

If you are still confused about Bezier curves, you might want to try the following Java program written by a TA for this module. Based loosely on this assignment, this program allows you to manipulate the control points of the curve and watch it deform in real time:

http://bit.ly/cc_cs3241_bezier

Final note

All the reshape, resize functions should be already handled. Unless you've messed them up, you should not need to modify them. You can also assume all the points input are within the 600 x 600 windows of the screen. Last but not the least, you should NOT use the Bezier function of OpenGL to draw your "main" curve, namely, no `glEvalCoordinate`. (However, you can use it within your object in Step 6.)