Due: Sep 25 11:55pm

Project 1b

Points: 45 (+15 BONUS)

Purpose

Interact with smooth curves in OpenGL.

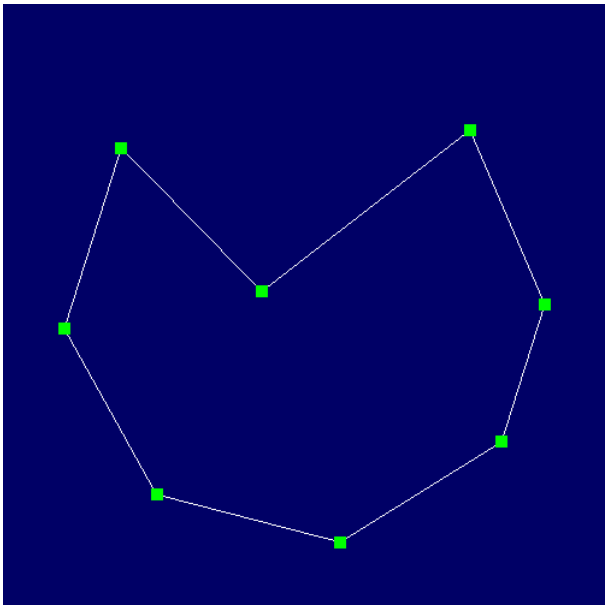
Set Up

Use the *reorganized* code base and fill in the code you wrote for Project 1a to enable picking.
Place $N = 8$ control points equally distributed to form a circle.
The title of the window to your name and your UFID, e.g. "Firstname Lastname (ufid)"

Task 1: Connect the points

Points: 5

Connect all the control points with white lines



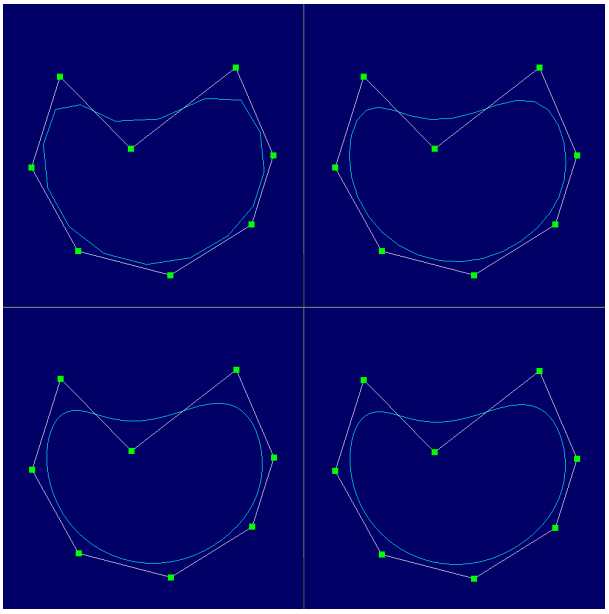
- In order to draw a closed polygon and also draw the line in white instead of blue(the points' color), you may want to create another `Vertex` array of size `ncpoints + 1`. Copy the coordinate from `cpoints`, set the color to white and make sure the curve is closed.
- Set up the VAO, VBO for this new array and call `glDrawArrays` with `GL_LINE_STRIP` to draw the polygon.

Task 2: Subdivision

Points: 10

From the polygon in task #1, generate a smoother curve by subdivision.

Initialize $P_i^0 = P_i$ (green). Double the number of control points by setting



$$P_{2i}^k := \frac{P_{i-2}^{k-1} + 10P_{i-1}^{k-1} + 5P_i^{k-1}}{16}$$

$$P_{2i+1}^k := \frac{5P_{i-1}^{k-1} + 10P_i^{k-1} + P_{i+1}^{k-1}}{16}$$

where k is the level of subdivison and
i is the index of points is in range $0 \ldots (N \times 2^k - 1)$.

When $k = 0$ show the original control polygon. When key 1 is pressed and $k = m$ then compute and draw the control polygon at level $k = m + 1$. However, when $m = 5$, m is set to 0 and the original control polygon is drawn.

Task 3: C^3 Bezier

Points: 10

Another way to generate a curve is to construct several smoothly connected Bezier pieces.

We are going to construct 8 pieces of degree 4. The ith piece has control points $\mathbf{c}_i = \{c_{i,0}, c_{i,1}, c_{i,2}, c_{i,3}, c_{i,4}\}$. One of the 5 control points is given as:

$$c_{i,2} = \frac{4P_{i-1} + 16P_i + 4P_{i+1}}{24}$$

and the points $c_{i,1}$ and c_{i_3} have the following form:

$$c_{i,1} = aP_{i-1} + bP_i + cP_{i+1}$$

$$c_{i,3} = cP_{i-1} + bP_i + aP_{i+1}$$

for constants a, b, c.

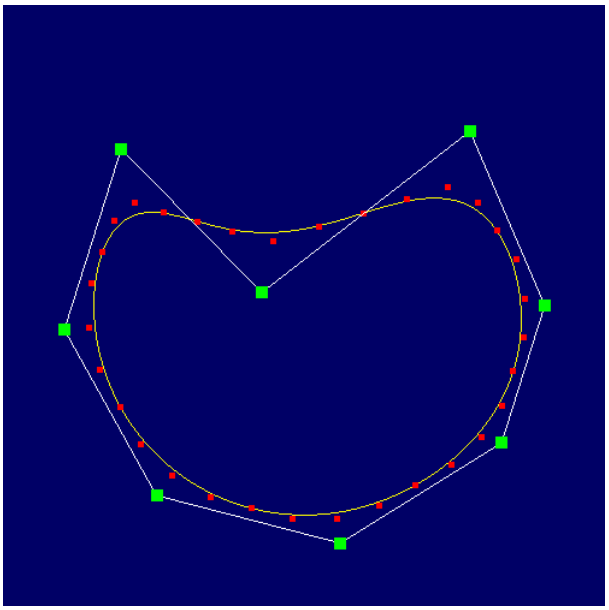
- Determine $c_{i,0}$, $c_{i,1}$, $c_{i,3}$ and $c_{i,4}$ so that the ith and the $(i + 1)$th Bezier piece join C^3 . Write down the formulas for the control points into a readme.txt file.

Task 4: Draw Bezier curve

Points: 15

For each Bezier curve piece:

Choose n, the tessellation number, i.e. the number of pieces to discretize each Bezier piece into, sufficiently high so the discretized curve looks smooth.



- For each Bezier curve piece, evaluate the points on the curve when $t = 0, \frac{1}{n}, \frac{2}{n}, \ldots, 1$ using De Casteljau's algorithm and connect the evaluation points with a polyline.
- When 2 is pressed, show/hide the Bezier curves.

Task 5: Similarity between the curves

Points: 5

- Assume the 8 points are uniformly destributed on the circle. Is the resulting curve a circle?
- Justify your answer in the readme.txt file.

BONUS

Points: 10

Implement **Task 4** using the **OpenGL 4.x**'s tessellation engine.

REMARK

Point penalty: 10

Make sure **picking still works** on the original N vertices, and your curves adapt to their movement.

WHAT TO SUBMIT

- All of your **source files** (.cpp's, shaders, readme's, etc) as in link
- A **link** to a screen capture of your running program showcasing the implementation of all of the tasks using recordit (Mac, Win) or similar software.
  For bigger tasks screen capture each step.
  NOTE: Make sure your programs compile and run on the lab machines in E313 as the grading would be done on them.

GRADING

- Failure to fulfill the submission requirements will result in a grade of 0.
- For regrading:
    - Come to the TA's office hours.
    - Be ready to download your submission from Canvas and show it running.