CS 535 Computer Graphics

Programming Assignment 1 (40 points)

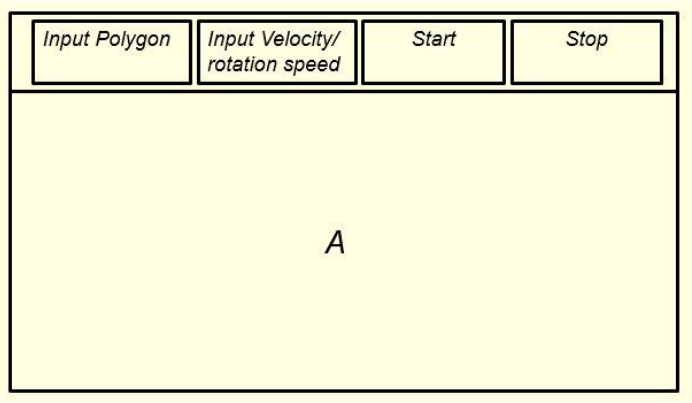
Due: 09/25/2025

[Animated 2D Bouncing]

1. Assignment Description

You job here is to convert a given (outdated) openGL program that performs 2D animation of a 2D polygon and the associated composite Bezier curve to a **modern** openGL program. The term "**modern**" means your C++ program needs to use shader program and shader objects (vertex array objects) to drive the shaders of your graphics card to do the computation and rendering. The given (outdated) openGL program can be downloaded from the class website using the link "Animated 2D Bouncing" underneath "Sample Programs for Programming Assignments" on the right panel of the class website. There are some problems with the sample program. You need to find those problems and have those problems fixed in your program as well.

Initially, your program should display a GUI on screen as follows:



The four small rectangular areas are supposed to act as menu items. By clicking in the first (left most) rectangular area (the one with a label "Input Polygon"), a specific polygon should be displayed at the center of the region (A) below (centroid of the polygon matches center point of region A). The type of polygon that you should use depends on the last digit of your UK student ID. If the last digit of your UK student ID is even, you use polygon (a) in Figure 1. Otherwise, you use (b). By clicking in the second small rectangular area (the one with a label "Input Velocity/Rotation Speed"), the user can input the velocity and rotation speed of the polygon by clicking at two points in A to input a vector

and another two points to input a degree to the program. By clicking in the third small rectangular area (the one with a label "Start"), the user starts the animation process and by clicking in the fourth area (the one with a label "Stop"), the user stops the animation. The animation can be repeated as many times as possible. The polygon and the associated composite Bezier curve should rotate (about its centroid) clockwise if your SSN is even. Otherwise, the polygon and the associated composite Bezier curve should rotate counter-clockwise.

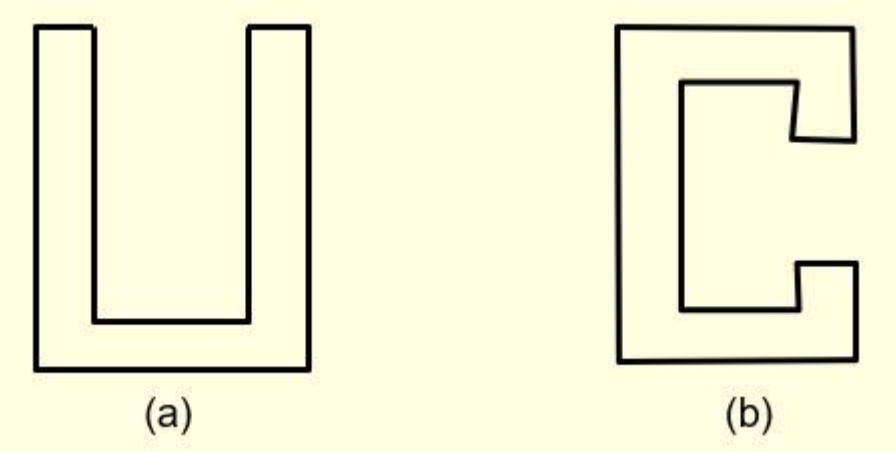


Figure 1.

(Note:
OpenGL is a Graphics API, not GUI, so you don't have any Buttons, Textboxes in OpenGL. But there are several external GUI libraries that will allow you to create buttons, textboxes, drop-down menus, etc. For example,

https://www.github.com/ocornut/imgui
http://en.wikipedia.org/wiki/GLUI

2. Original Implementation Details

For each given polygon, one can create a closed composite Bezier curve in two steps:

1. Compute mid-point of each edge of the polygon

2. For each vertex of the polygon, construct a Bezier curve of degree two using the vertex and mid-points of the adjacent edges as control points

For instance, in part (a) the following figure, once the midpoints of edge AB and edge BC have been computed, say D and E, respectively, one can define a Bezier curve segment of degree two for vertex B as follows (see part (b) of Figure 2):

$$C_B(t) = (1-t)^2\mathbf{D} + 2t(1-t)\mathbf{B} + t^2\mathbf{E}$$

where $0 \leq t \leq 1$.

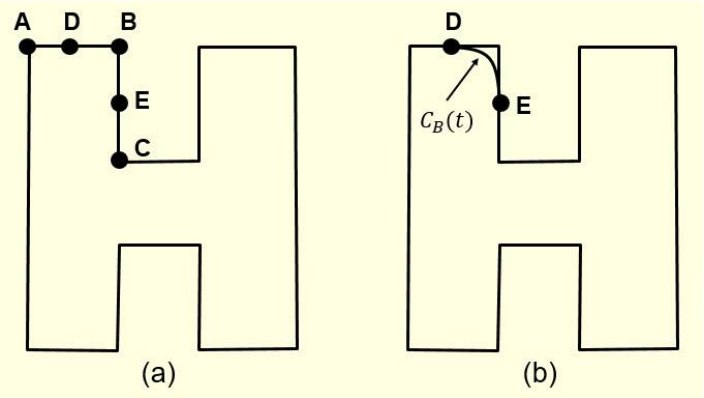


Figure 2.

By constructing a Bezier curve segment of degree two this way for each vertex of the polygon, one gets a closed smooth composite Bezier curve (see Figure 3).

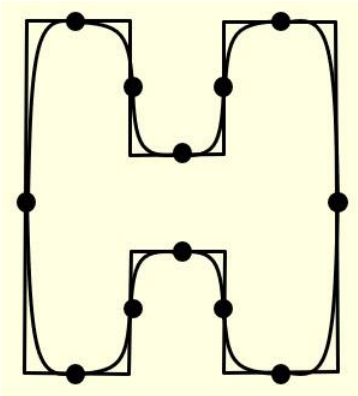


Figure 3.

When the third small rectangular area (the “Start” button) is clicked by the user, your program should construct a closed, composite Bezier curve of degree two for the shown polygon using the above approach first.

With the input velocity (the vector defined by the first input point (start point)

and the second input point (end point)) and rotation speed (number of horizontal pixels between the third and the fourth pixels) about its centroid (of the polygon), the program then moves and rotates the polygon and the Bezier curve accordingly and bounce them in the appropriate direction when the polygon and the Bezier curve hit the wall (boundary of the screen window)

HERE IS THE IMPORTANT PART

The program does not bounce the polygon and the Bezier curve immediately after one of the vertices touches the wall. The program starts to bounce the polygon and the Bezier curve only when the "centroid" of the polygon hits the wall. So the polygon and the associated curve in Figure 4 will keep moving and rotating until the centroid (the red spot) reaches the wall.

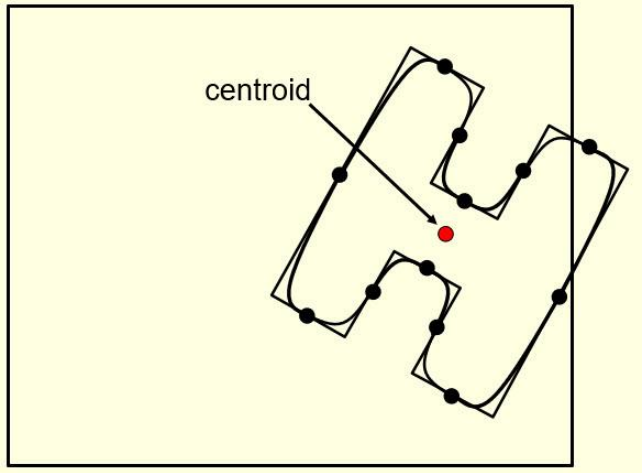


Figure 4.

This means the program will need to clip the polygon against the wall periodically and update the corresponding Bezier curve accordingly. See Figure 5 for the new shape of the polygon after the clipping process. The Bezier curve will then be recomputed using the vertices of the new polygon. At some point, both the polygon and the associated Bezier curve might be broken into several smaller, disjoint polygons and Bezier curves. The program uses the Weiler-Atherton's algorithm in the clipping process. An HTML file explaining the concept of the W-A algorithm is attached underneath this assignment for your reference.

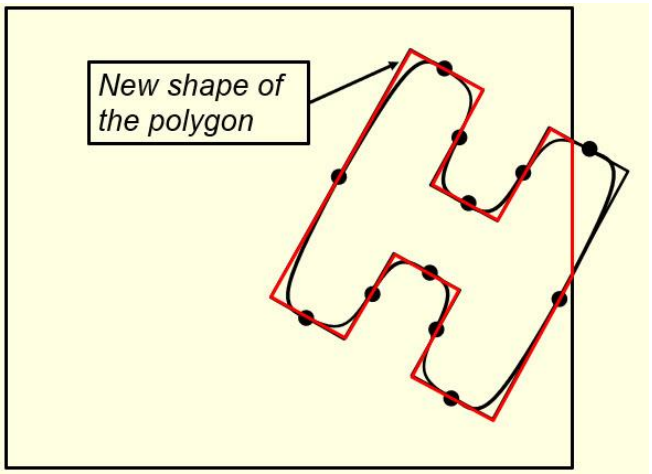


Figure 5.

3. Remarks:

(a) A lot of things have changed. You need to find out what commands should be used when you do the conversion. For instance, the following openGL command was used to handle events before:

glutMainLoop( )

Modern openGL doesn't use it any more. A while loop is used instead (see slide 25 of the notes "OpenGL and Shaders").

(b) You should be able to compile and execute the given (outdated) openGL program on your platform with the new environment. The header file "glew.h" covers "gl.h", "glu.h" and "glut.h", so you don't have to have these header files included in your program.

(c) The line drawing command used in the given program should be replaced with a different approach.

(d) Shader program and shader objects compiling and linking are standardized, just follow the example programs given in class.

(e) Important things are how to define the shader objects and how to design the shader program.

4. What to Turn In

Mail your program (in one file) to "cheng@cs.uky.edu" on or before the due date. Then come in during my office hours between 9/25/2025 and 10/3/2025 to do a demo of your program.