CS 548: Computer Graphics
Assignment 1
Due 2/26/2015, 11:59pm EST

Description:

In this project, you will implement a program that will do three things:

1. Allows the user to draw a wireframe polygon (i.e., series of lines) using Bresenham's line drawing algorithm.
2. Allows the user to change the current drawing color (for items already drawn and items to be drawn)
3. Draws circles (using the midpoint circle algorithm) automatically at each vertex of the polygon.

For the line-drawing part, here is what your program should be able to do:

1. When the user first presses and holds down the left mouse button, a line should be drawn from the initial point where the mouse button was pressed to the current location of the mouse. This should be updated dynamically as the user continues to move the mouse and hold down the left mouse button.

2. When the user releases the left mouse button, the line should be drawn on the screen, and the endpoint of the line should stop following the mouse position.

3. If the user presses and holds the left mouse button again, a new line should be drawn from the endpoint of the last line to the current mouse position.

4. If the user right-clicks, a line should be drawn connecting the LAST vertex drawn with the FIRST vertex. Note that NO NEW VERTEX should be created on a right-click.

5. If the user clicks the left mouse button again to start drawing AFTER a right-click, the current polygon should be cleared (as well as any circles drawn), and a new polygon started.

For the color-changing part, the user should be able to hit a key on the keyboard to change to one of 4 color choices: white, red, green, and blue. (So, “1” = white, “2” = red, etc.)

This should update ALL colors in your color buffer to be the same color; so, if the user hits “red”, all lines drawn and all lines to be drawn will draw in red.

For the circle-drawing part, a circle with a radius of 50 pixels should be drawn on each vertex of the polygon (including the one being dragged around by the user.

You must use Bresenham’s algorithm to draw lines. This means you will use OpenGL to draw points only (GL_POINTS). Do NOT draw the lines with GL_LINES!!!
You must deal with all 8 octants for line drawing.

You may use the following code in this project:

- Any of the sample code for this class
- The Bresenham’s line drawing code in the Hearn-Baker book (pg. 98-99 in the 3rd edition) as a starting point (you will need to adapt it for all 8 octants)
- The circle midpoint code found in the Hearn-Baker book (pg. 108-109 in the 3rd edition)

However, you may NOT use code from elsewhere (online or offline) for ANY part of this assignment. You also CANNOT share code amongst yourselves. These are INDIVIDUAL projects.

**NO CREDIT will be given for the Bresenham code if it only handles 2 octants (i.e., what the Hearn-Baker code handles).**

The current line must be updated dynamically as the user moves the mouse while pressing the left mouse button (and again, a circle must be drawn as well).

The program MUST redraw the image while idle, so you must pass in your drawing function to glutIdleFunc as well as glutDisplayFunc.

You must use OpenGL, GLUT, GLEW, and GLM for this project.

**You must use OpenGL 3.0+** (so do NOT use the fixed-function pipeline, glBegin()/glEnd(), etc.). That said, just use the simple shaders from the sample code, since you should only have to modify the C++ code.

You will need vertex buffers for pixel positions as well as the color for each pixel (but the color can be set to the same value).

You must also include a README (in plain text). This should include:

- *Title of project and class:* so, “CS 548: Assignment 1”
- *Your name and email*
- *OpenGL version:* also include what version your shaders are using (the #version part)
- *OS and Development environment:* For example, Windows 7 and Visual Studio 2013
- *Basic description of any controls:* you may use the text from this assignment to explain what the mouse is supposed to do, but include any other key commands / mouse commands you have.
- *Whether or not you attempted the bonus credit (and whether it’s supposed to be working)*

The README will be graded on clarity, neatness, and content, and it’s worth 10% of your grade.
Bonus:

For bonus credit, *when the user right-clicks and closes the polygon*, instead of drawing full circles at each vertex, only draw the part of the circle segment that goes from one edge to another:

![Image of two circle segments](image)

You want the image on the right, NOT the left.

In some cases, the circle will loop around the “outside”; this is still acceptable:

![Image of a circle segment loop](image)

To do this, you will need to:

1. Loop through each set of three vertices \( (i - 1), i, (i + 1) \)
2. Check the angle for the two lines \( (i) \) to \( (i - 1) \), and \( (i) \) to \( (i + 1) \)
   a. You will probably use the \( \text{atan2}() \) function, which is in radians, and returns from \(-\pi\) to \(\pi\). To make all angles positive: \( \text{if}(\text{angle} < 0) \text{ angle} = 2.0*\pi + \text{angle} \)
3. Pick the smaller of the two angles as your “start angle” and the other as your “end angle”
4. When you go to plot each pixel in the midpoint circle algorithm, check whether the vector formed by the pixel coordinates has an angle between the start and end angles
5. If it is, draw that pixel. If not, don’t.
Grading:

The assignment will be graded as follows:

- (10%) README
- (40%) Bresenham’s algorithm line drawing (in ALL 8 octants)
- (20%) Left mouse button functionality handled correctly (i.e., line drawn to current mouse position while left mouse button is pressed, keep last line when mouse button released)
- (10%) Right mouse button functionality (polygon closed, new left click after clears polygon)
- (10%) Midpoint circle drawing algorithm
- (10%) Change current drawing color
- (Bonus 10%) See bonus description

Submission:

You should take the following items and put them in a *.zip file:

- C++ source file → name this “Project1.cpp”
- Your vertex shader file
- Your pixel/fragment shader file
- README

Upload this zip file to Angel.

Do NOT send me your project files (e.g., Visual Studio project files).

Hints:

I would STRONGLY suggest creating a simple struct for holding 3D points:

```cpp
struct Vec3 {
    GLfloat x, y, z;
};
```

I would also add some constructors in there for ease of use.

I would suggest keeping track of the start and end points for the lines in a C++ vector (e.g., myLines); then, use your Bresenham function to fill ANOTHER vector (e.g., myPixels) with the actual pixels to draw. Finally, before you draw the lines, update the vertex buffer with the pixel data (e.g., use glVertexAttribPointer() to copy the data from myPixels to the vertex buffer).
Ideally, you should create a struct for your current polygon that contains a vector with the vertices, a vector with the color for each vertex, and finally whether the polygon is “finished” or “not finished” (i.e., closed or not closed).

You should also consider making a struct for circles as well (i.e., center and radius).

Regular Version Screenshot:

Bonus Version Screenshot: