COSC 4370 – Homework 2

Name: Austin Thibodeaux PSID: 2037865

October 2022

# 1 Problem

This assignment requires the completion of four problems, all of which involve various transformations of simple shapes in OpenGL. The first problem asks to arrange an array of teacup shapes into a circular formation. The second problem asks to make a staircase with the cube shape. The third problem asks to arrange an array of teacups into an upside-down triangle shape. Problem 4 asks to make any complex shape using various shapes and triangles using glPushMatrix/glPopMatrix and transformations.

# 2 Method

Four functions have been defined for this project:

* **problem1:** Draws 10 teapots in a circle along with a translation to the right and up. Each teapot is also rotated forward slightly on its local coordinates.
* **problem2:** Draws 20 cubes that decrease in height for each step of the staircase. The bottom of each cube is aligned to form a straight line, whereas the top of each cube decreases in height for each iteration.
* **problem3:** Draws 21 teapots in an upside-down triangle formation. Each teapot has a vertical and horizontal margin.
* **problem4:** Draws a sunflower using two cylinders representing the stem and base, and 16 petals using triangles. The sunflower sits on top of a “hill” rendered with a flattened sphere.

# 3 Implementation

In this implementation, the image size will be 800×800 and the origin of the ellipse will be translated up 400 pixels. This will ensure that the bottom half of the ellipse is not cut off by the bounds of the image. The main function will initially fill the canvas with black, and then calls *MidpointEllipse* with the parameters *offsetY=0, offsetX=400, a=12\*64, and b\*6\*64*, where *a* is the major axis and *b* is the minor axis. Then the image is saved to a bmp file.

## 3.1 MidpointEllipse

This function will draw, and only draw the first quadrant of an ellipse. The algorithm is divided into two regions with region 1 transitioning to region 2 when the slope of the ellipse reaches -1. The slope is defined by .

The first region is computed using a while loop that exits on the condition that the slope reaches -1. In the first iteration, *x* will start at 0 any *y* will start at the minor axis. The first pixel (and its mirrored counterpart) is drawn by calling *FlipAndDraw*. Before the first iteration, a *prediction variable*, named **p1**, is used to determine which of two pixels is selected to be painted. The value of *p1* is obtained by finding the value of the ellipse function *f(x+1, y-0.5)*, which is a point between the East and Southeast neighboring pixels. If the value is less than 1, the *East* pixel is selected, and *dx*, the slope numerator, and prediction function (*f(x+2, y – 0.5))* are updated for the next iteration. If the value is greater or equal to 1, the *Southeast* pixel is selected, and *dx, dy* slope numerator and denominator, and the prediction variable *(f(x+2, y – 3/2))* are updated for the next iteration.

When the slope reaches -1, the second region begins. The second region is computed using another while loop that exits on the condition that y is greater than 0 or hits the x-axis. A second prediction variable is declared **p2** is similarly used to determine which of the two pixels is selected to be painted. The value of *p2* is obtained by the value of the ellipse function *f(x+0.5, y-1)*. If the value is greater than 1, the *South* pixel is selected, *dy* is increased, and *p2* is updated to *f(x+0.5, y-2)*. Otherwise, the *Southeast* pixel is selected, *dx* and *dy* are increased, and *p2* is updated to *f(x+3/2, y-2)*. After this while loop the algorithm is finished and the program exits and saves the bmp file.

# 4 Results

When *MidpointEllipse* finishes its second region, the program will then create a bmp file. The image consists of the right half of an ellipse, centered vertically, drawn in a red color with a black background.

# References

* <https://www.geeksforgeeks.org/midpoint-ellipse-drawing-algorithm/>
* <https://www.includehelp.com/computer-graphics/mid-point-ellipse-algorithm.aspx>
* <https://www.cpp.edu/~raheja/CS445/MEA.pdf>