**CSCE A385 Programming Assignment 2**

Computer Graphics

Shanan Almario

**Program Notes**

IDE: Visual Studios 2015

**Problem 1 Code Description**

The code outputs an octagonal space where the Brownian motion is randomly placed inside. The motion will continue to move around the space until it collides with the bounds of the space. The start of the motion is shown by a green marker, and the end of the motion is shown by a red marker. The wall where the motion collides will become opaque.

**Interaction:**

*Keyboard*:

x: will move the space counter clockwise on the x-axis

X: will move the space clockwise on the x-axis

y: will move the space counter clockwise on the y-axis

Y: will move the space clockwise on the y-axis

z: will move the space counter clockwise on the z-axis

Z: will move the space clockwise on the z-axis

*Mouse: ( As specified by the assignment)*

If the mouse is moved up, then the object is rotated around the x-axis, clockwise in the y-z coordinate plane.

If the mouse is moved horizontally to the right/left, then the object is rotated around the y-axis, clockwise in the z-x coordinate plane.

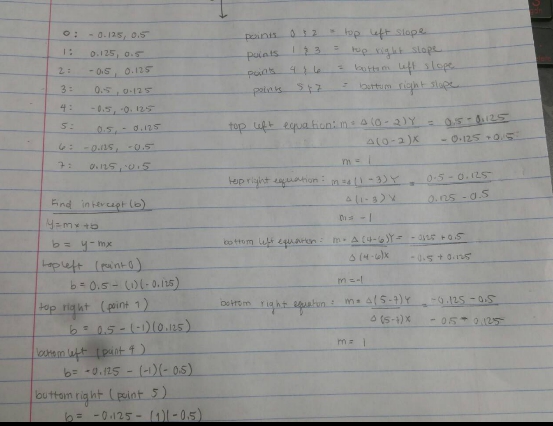
**Problem 1**

There are 7 Vertex Array Objects: the wire form of the octagon, the transparent walls of the octagon, the Brownian trajectory, the wire for the start marker object, the walls of the start marker object, the wire for the end marker object, and the walls of the end marker object. Each object is initialized in the init function, and are drawn in the display function.

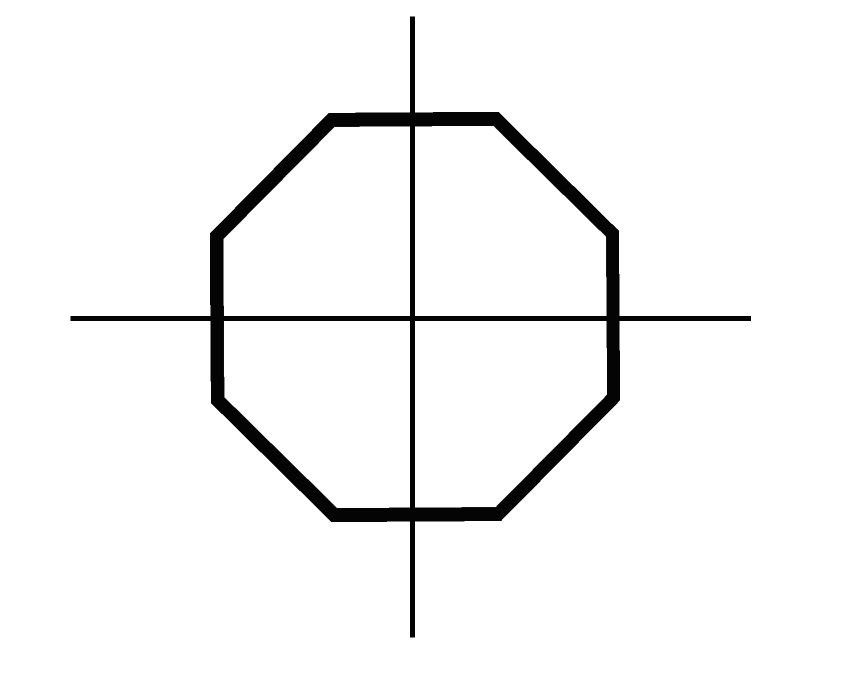
The resize routine in this code will adapt the display size to match the window screen.

**Collision Detection Math**

*Refer to Diagram 1 for the points*



The Z-Plane is just a rectangle, so checking the boundary is zMin and zMax. The xy plane is an octagon shape, so at certain sections of the octagon, the current point will be checked to see if it is past the slope lines at each side of the octagon.



**X**

**Y**

**15**

**14**

**13**

**12**

**11**

**9**

**8**

**10**

**7**

**8**

**6**

**5**

**4**

**3**

**2**

**1**

**0.125**

**0.125**

**-0.125**

**-0.125**

**\_**

**\_**

**|**

**|**

**-0.5**

**-0.5**

**0.5**

**0.5**

**Diagram 1: Vertex Numbers**

**Blue**

**Vertex: Point**

0: (-0.125, 0.5, 0.5)

1: (0.125, 0.5, 0.5)

2: (-0.5, 0.125, 0.5)

3: (0.5, 0.125, 0.5)

4: (-0.5, -0.125, 0.5)

5: (-0.5, -0.125, 0.5)

6: (-0.125, -0.5, 0.5)

7: (0.125, 0.5, 0.5)

**Red**

**Vertex: Point**

8: (-0.125, 0.5, -0.5)

9: (0.125, 0.5, -0.5)

10: (-0.5, 0.125, -0.5)

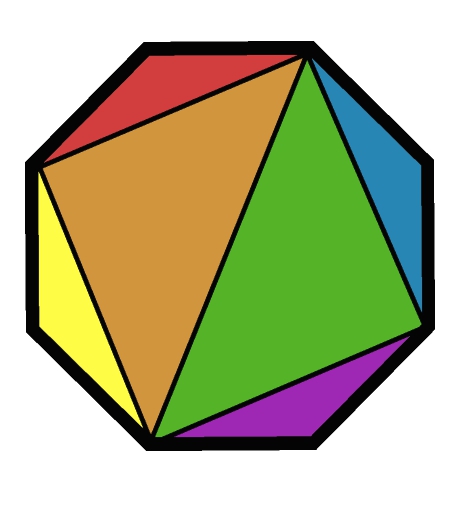
11: (0.5, 0.125, -0.5)

12: (-0.5, -0.125, -0.5)

13: (-0.5, -0.125, -0.5)

14: (-0.125, -0.5, -0.5)

15: (0.125, 0.5, -0.5)



**15**

**14**

**12**

**13**

**10**

**11**

**9**

**8**

**4**

**8**

**7**

**6**

**5**

**3**

**2**

**1**

**Diagram 2: Edge Grid Image**

**Red Triangle**

0, 1, 2

8, 9, 10

**Yellow Triangle**

2, 4, 6

10, 12, 14

**Purple Triangle**

6, 7, 5

14, 15, 13

**Blue Triangle**

5, 3, 1

13, 11, 9

**Orange Triangle**

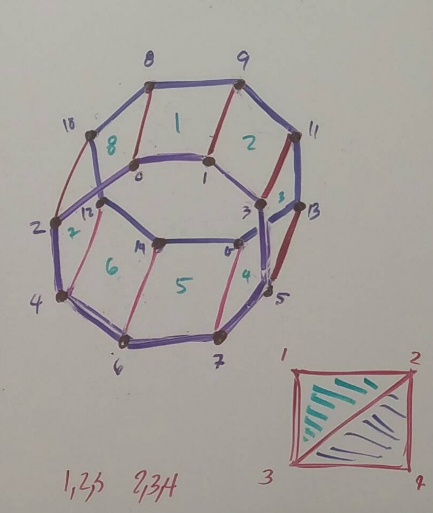
1, 2, 6

9, 10, 14

**Green Triangle**

1, 5, 6

9, 13, 14



**Diagram 3: Rectangles Binding Both Octogons**

**Problem 2 Code Description**

This code should show a cylindrical cannon with two wheels and an axle. Mouse interaction and keyboard interaction has been implemented to examine cannon.

**Interaction:**

*Keyboard*:

z/Z: roll the camera

y/Y: yaw the camera

x/X: pitch the camera

*Mouse: ( As specified by the assignment)*

If the mouse is moved up, then the object is rotated around the x-axis, clockwise in the y-z coordinate plane.

If the mouse is moved horizontally to the right/left, then the object is rotated around the y-axis, clockwise in the z-x coordinate plane.

**Problem 2**

All objects (barrel, wheels, and axle) are nodes and are set up as siblings of each other. Each object is has its own build and draw functions which are called upon in init and display. Everything except the axle is a capped cylinder. The axle is drawn without drawing the bottom and the top cap. All objects are cylinder objects, and are scaled and rotated to look like the object they need to look like.

**Problem 3**

The third problem requires mouse interaction with the first two problems and is seen in the description of the two problems.