Computer Graphics Fall 2020 **10/02/2020**

Assignment – Scene2 (S2)

Due on, October 11, 2020.

Submission Instructions:

Please carefully follow the instructions for assignment S1.

Assignment Instructions:

The goal of assignments S1 to S6 is to produce a scene of "medium complexity." For ideas about the default scene see color plate 24 in the book by Angel (available on TRACS).

The goal of this assignment (S2) is to expand the scene produced in assignment S1. Following assignments will manipulate these objects (e.g., duplicate, apply transformations, etc.) and the surroundings (light, shading) to create the actual scene.

The main requirement of S2 is to exercise GL transformations. In specific:

1. Use perspective projection with the camera placed at the center of the scene and capable of "seeing" the entire set of objects you have generated in assignment S2. Use gluLookat() to place the camera.
2. Define a window of 800x600 that is not in the "origin" of the screen, and a viewport of 500x500 that is not in the "origin" of the window. The entire scene should be rendered into this viewport.
3. Use the library of objects defined in S1. It should include a point, line, circle **which should be drawn using its parametric equations**, triangle, square, hexagon, and a cube, placed in the center of the viewing volume. Do not render the members of the library. You can use your S1 program or the enclosed program snippet by Daniel Palmer to generate the library of objects.
4. Each object of the library should be rendered in the following way:
    a. Each object should be translated and scaled so that objects do not overlap.
    b. Scale the point, line, triangle, and the square with a scaling factor of your choice; using different scaling factors for each object; but, the same scaling factor for each axis of the objects.
    c. Scale the circle and the hexagon with scaling factors of your choice using different scaling factors for each object; and different scaling factors for each axis.
    d. Each scaled object, except for the point and the cube, should be rotated around its center by a different degree (for example 30 degrees for the line and 60 degrees for the triangle). You can choose your own vector and rotate around this vector.
5. Render a cube with edge length of 0.25 anywhere in the scene and "far" from the center, then rotate it by 45 degrees around the vector (1, 1, 1) translate and render again (generating a second cube).
6. You should produce only one scene, so you may want to push and pop the Model View matrix between consecutive transformations.

Computer Graphics Fall 2020 **10/02/2020**

Assignment – Scene2 (S2)

Due on, October 9, 2020.

```
/*************************************************************************
 Author: Daniel Palmer
*************************************************************************/
   //Draw triangle
   glColor3f(1, 1, 0);
   ngon(3);

   //Draw square
   glColor3f(1, 0, 1);
   ngon(4);

   //Draw hexagon
   glColor3f(1, .5, .5);
   ngon(6);

   //Draw circle
   glColor3f(0, 0, 1);
   ngon(100);

//Function to draw a N-sided object that is upright (defined to have the bottom edge horizontal)
//at the center of the current model identity, with diameter = 1

void GLobj::ngon(int n){
   float degree, vertx, verty, rotate, degToRad;
   rotate = 360.0/n;      //Vertex rotation
   increment
   degree = rotate/2 + 180;//Current degree of vertex (starts rotated to make object upright)
  degToRad = 180/3.14159; //Conversion factor from degrees to radians

  glBegin(GL_POLYGON);
   for(int i = 0; i < n; i++, degree += rotate)
   {
      vertx = 0.5 * sin(degree/degToRad);        //next vertex's x coordinate
      verty = 0.5 * sin((90 - degree)/degToRad); //next vertex's y coordinate
      glVertex3f(vertx, verty, 0);
   }
   glEnd();
}
/*************************************************************************/
```