INF01009 – Computação Gráfica
2003.1
Manuel M. Oliveira
Programming Assignment 1

Passed in: May 21, 2003 **Total of Points of the Assignment: 100**
Due: May 28, 2003 at 1:30pm

GOALS

The goals of this assignment are familiarize you with some important OpenGL commands required for rendering 3D scenes. By completing this assignment you will learn how to:

a) Define a virtual camera in a 3D virtual scene, specifying its position, orientation and field of view;
b) Render an object using points, wireframe and solid representations;
c) Change the field of view of the camera;
d) Translate the camera in both X, Y and Z directions;
e) Produce basic shading effects by using one light source.
f) Perform backface culling

1) Study the section **Some Hints for Building Polygonal Models of Surfaces** (pages 84 to 92 of the OpenGL red book) and implement the recursive subdivision version of the algorithm used to render a icosahedron (Example 2-17 on page 91). Your program should display a view of the 3D model.

   Suggestion: start with Example 2-16 and modify it appropriately. Also note that you will need the *vdata* and *tindices* arrays defined in Example 2-13 as well as the function *normalize* defined in Example 2-15.

   a) Render the polyhedron using the recursive subdivision algorithm. **(20 points)**

   b) Render the solid using points, wireframe and solid representations. **(20 Points)**.

   c) Interactively change the horizontal and vertical field of view of the virtual camera **(15 Points)**.

   d) Translate the camera along the X, Y and Z directions using the glTranslatef() command **(15 Points)**.

   e) Produce some shading effect by enabling/disable one light source. **(20 Points)**

   f) Enable/disable backface culling **(10 points)**

Tips on How to Complete the Assignment

As you know, all C/C++ programs require a main function. In the main function you need to specify the parameters associated with the window you will be using for display your renderings. Below, you will find a sample main function (note that you may not need all these commands for your program) and many other templates for the functions you will need. Study the meaning of all such commands and make sure you understand exactly what tasks they are performing in this code.

```
int main(int argc, char *argv[])
{
...
//
//   initialize the window for rendering with OpenGL
//
  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_ALPHA | GLUT_DEPTH);
  glutInitWindowPosition(0, 0);
  glutInitWindowSize(400, 400);
  win_id =  glutCreateWindow("OpenGL");
  glutDisplayFunc(openglDisplay);
  glutReshapeFunc(openglReshape);
  glutMouseFunc(Mouse);                    //   you may not need or want to use this command
  glutMotionFunc(worldMotion);             //  you may not need or want to use this command
  glutKeyboardFunc(worldKeyboard);
  glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
  glClearColor(0.0, 0.0, 0.0, 0.0);

....
  glutMainLoop();
  return 0;
}
```

All OpenGL program requires that we specify a *Display* and a *Reshape* functions. These are specified by the commands *glutDisplayFunc* and *glutReshapeFunc*, respectively. Here are examples of these functions that you can use in your program. Again, study the meaning of all these commands. You can initialize *vfov* and *hfov* (vertical and horizontal field of view) with 60 (degrees), and *Znear* and *Zfar* with 0.1 and 1000.0, respectively (note that by using variables (as opposed to constants) we can easily change the effect of commands).

```
//*****************************************************
//  Reshape  function for OpenGL
//*****************************************************
void openglReshape(int w, int h)
{
  glViewport(0, 0, (GLsizei) w, (GLsizei) h);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluPerspective(vfov, hfov/vfov, Znear, Zfar);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
}
```

Initialize the camera position (variable <cam_pos> in the code fragment below at (0, 0, 2.5)).

```
//*****************************************************
//  Display function for OpenGL
//*****************************************************
void openglDisplay(void)
{
 glLoadIdentity();  // initialize ModelView Matrix as the Identity Matrix
 gluLookAt(<cam_pos.x>, <cam_pos.y>, <cam_pos.z> , 0.0, 0.0, -1.0, 0.0, 1.0, 0.0);

//
 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); // before rendering, clear color and depth buffers
   <your function that draws the icosahedron is called here>();

   glutSwapBuffers();                    // after you are done rendering, swap buffers
}
```

Selecting the rendering mode can be achieved with the following code fragment, which need to be executed before you draw the actual triangles.

```
switch(<renderMode>) {
        case <POINTS>:
                <render_mode> = GL_POINT;
                break;
        case <WIREFRAME>:
                <render_mode> = GL_LINE;
                break;
        case <SOLID>:
                <render_mode> = GL_FILL;
                break;
}
glPolygonMode(GL_FRONT_AND_BACK, <render_mode>);
```

You can set the color of the polyhedron (or any of its vertices) using the command *glColor3f(<r>, <r>, <b>).* For instance, to set the rendering color to white use glColor3f(1.0, 1.0, 1.0).

You can change the field of view using the command *gluPerspective(<vfov>, <hfov>/<vfov>, <znear>, <zfar>);*

In order to translate the camera, change the camera position using the *gluLookAt* command as described before.

Enabling/Disabling lighting and light sources can be achieved with the following commands:

```
If (<enable>) {
   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
 }
else
  if (<disable>) {
     glDisable(GL_LIGHTING);
     glDisable(GL_LIGHT0);
  }
```

Backface culling can be enabled and disabled with the commands *glEnable(GL_CULL_FACE)* and *glDisable(GL_CULL_FACE),* respectively.

Good luck.