Exercise 02561-01 **Primitives and Attributes - Introduction to OpenGL**

Reading
ANG: Chapter 2.2-7
PRIM: Chapter 2, 3.1

Purpose
The purpose of this exercise is to give a short introduction to create, edit, link and run C/C++ programs that are based on the graphics library OpenGL, and the libraries GLU, GLUT that handles utilities on top of OpenGL and the window system of the platform.

Startup
In order to create and run the programs you may use the description of starting Visual Studio (on the CampusNet).

Part 1
Run the demonstration program 02561-01-01-2009.cpp, which draws a rectangle in a window on the screen. This is about the simplest OpenGL program you can imagine – a kind of "Hello World" program. The program is listed in appendix 1 and accessible as a file from the Campusnet. Try to understand the purpose of each function.
Add the lines, which have been commented out in the appendix 1, to the program file. What is the function of these lines?

Part 2
Also run the program 02561-01-02-2009.cpp which draws a rectangle. The program is listed in appendix 2 and you can also access this program from the CN. The program has a few more features and can be used as a kind of template for drawing 2D drawings in OpenGL.
Try to understand the overall structure of the program and the purpose of each function.

Extend the program to include a triangle with vertices (2,2), (5,2) and (3.5, 5). Make that triangle red.

Translate the triangle with the vector [6,7]. Make the color of the vertices red, green and blue respectively and smooth shade the triangle. Rotate the rectangle (but not the triangle) 45 degree counter-clock-wise around the midpoint of the rectangle.

Part 3
Now we change to 3D graphics. Run the demonstration program 02565-01-03-2009.cpp. Also try to understand the overall structure and the purpose of each function.

Part 4
Make a program which draws a front page template.
You use appropriate primitives and attributes. Fill in a copy of the template and use it as front page when delivering an exercise.

Part 5
Draw a circle

Part 6
A window – viewport pair is defined by the coordinates $[x_{w1},x_{w2},y_{w1},y_{w2}]$ and $[x_{v1},x_{v2},y_{v1},y_{v2}]$. Such a pair defines a transformation (matrix); look at figure 2.3 and 7.4. Set up the transformation matrix which map a point $(x_{w,}y_w)$ to the point (x_v,y_v).

The window-viewport transformation can also be defined from concatenation of three simple transformations. Find which kind of transformations and set up the matrices for these three transformations. Check that the concatenated matrix is equal to the window-viewport transformation matrix.
How does OpenGL handle window – viewport transformations.

Part 7

A unit cube is defined in its own Object Coordinate system (Oo, Xo, Yo, Zo) by the diagonal vertices (0,0,0) and (1,1,1). The coordinate system is right-handed and the Yo-axis points upwards.
The object is scaled to double size and rotated 30-degree counter clockwise around the Yw-axis. Finally, the object is translated along the Yw-axis by three units.
The program 02561-01-07-2009.cpp (on CN) just makes an orthographic view of a unit cube, glutWireCube(1). We have also included the world coordinate axis Xw, Yw, and Zw, and the auxiliary line through (1,0,0) and (1,3,0).
Change the program so you can carry out the requested transformations mentioned above. Which transformation matrix did you use?
In which order did you specify the transformations.
Play with changing the order of the transformations.

Part 8

(continues next week)

In this part we shall make a perspective picture of a 10 unit cube with the diagonal vertices (0,0,0) and (10, 10, 10).

First we will like to make a front perspective view of the cube.
The Eye Point (View Point = Camera Position) is positioned in (20, 5, 5) and the View Plane goes through the origin of the world coordinate system. In a front perspective the View plane and a set of faces (or two main directions) are parallel to each other.
Find the position of the Point of Interest (=Principal Point, At-point).
Modify the program 02501-01-07-2009.cpp to make the front perspective view mentioned with the eye point (20., 5., 5.).
You use the functions gluPerspective and gluLookAt to set up the view frustum and the eye point, at-point, and up vector.
The parameter fovy=45 and up = (upx,upy,upz) = (0,1,0). Choose reasonably values for the rest of the parameters.
Make an X-perspective from the same eye point. The View Plane goes through the Yw-axis.
Play with the parameters of the two function mentioned above.

Delivery

You deliver all the exercises at the end of the course. Therefore for each situation make a screenshot of the screen window and save the programs.

NJC.02.09.2009

```
/* Appendix 1
*
*   ogl_min.cpp
*
*   02561-01-01-2009
*
*     This program draws a yellow polygon
*
*/


#include <GL/glut.h>


void Display (void);

int main (int argc, char **argv){
      glutInit (&argc, argv);
      glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
      glutInitWindowSize (700, 500);
      glutInitWindowPosition (10, 10);
      glutCreateWindow ("Hello World");

      glClearColor (0., 0., 0., 0.);
      glColor3f (1., 1., 0.);
      //glMatrixMode (GL_PROJECTION);
      //glLoadIdentity ();
      //gluOrtho2D (-10., 10., -10., 10.);
      //glMatrixMode (GL_MODELVIEW);


      glutDisplayFunc (Display);

      glutMainLoop ();
      return 0;
}


void Display (void){


      glClear (GL_COLOR_BUFFER_BIT);

      glBegin (GL_POLYGON);
            glVertex2f (-5.,-5.);
            glVertex2f (-5.,5.);
            glVertex2f (8.,5.);
            glVertex2f (8.,-5);
      glEnd ();

      glFlush ();
}
```

```
/*Appendix 2:  02561-01-02-2009
*
* The program draws a rectangle and a triangle
*
*/

#include <GL/glut.h>

void Init (void);
void Display (void);
void Reshape (int w, int h);

int main (int argc, char **argv){
      glutInit (&argc, argv);
      glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
      glutInitWindowSize (600, 500);
      glutInitWindowPosition (10, 10);
      glutCreateWindow ("Hello World 2");

      glutDisplayFunc (Display);

      glutReshapeFunc (Reshape);
      glutMainLoop ();
      return 0;
}


void Init (void) {

      glClearColor (0., 0., 0., 0.);
      glColor3f (1., 1., 0.);
      glShadeModel (GL_SMOOTH);
}


void Display (void){

      float V[][2] ={
            -5.,-5.,
            -5., 5.,
             8., 5.,
             8.,-5.
      };

      glClear (GL_COLOR_BUFFER_BIT);
      glBegin (GL_POLYGON);
            glVertex2fv (V[0]);
            glVertex2fv (V[1]);
            glVertex2fv (V[2]);
            glVertex2fv (V[3]);
      glEnd ();
      glFlush ();
}

void Reshape (int w, int h){

      glViewport (0., 0., w, h);
      glClearColor (0., 0., 0., 0.);
      glColor3f (1., 1., 0.);
      glMatrixMode (GL_PROJECTION);
      glLoadIdentity ();
      gluOrtho2D (-15., 15., -10., 15.);
      glMatrixMode (GL_MODELVIEW);
      glLoadIdentity ();
}
```

```
/* Appendix 3:  02561-01-03-2009   Tea pot
*
*/

#include <GL/glut.h>

int width = 500;
int height = 500;


void display() {

  glClear(GL_COLOR_BUFFER_BIT);
  glColor3f(0.0,1.0,0.0);

  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  glTranslatef(width/2,height/2,0);
  glRotatef( 25, 1, 0, 0 );

  glutWireTeapot(width/4);
  glFlush();
}


void reshape(int w, int h)  {

  width = w;
  height = h;

  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(0, width, 0, height, -1000, 1000 );

  glViewport(0, 0, width, height);
}


int main(int argc, char **argv) {

  glutInitWindowSize(width, height);
  glutInitDisplayMode(GLUT_RGB);

  glutCreateWindow("The Utah Teapot (modeled by Martin Newell in 1975)");

  glutReshapeFunc(reshape);
  glutDisplayFunc(display);

 // glutShowWindow();
  glutMainLoop();

  return 0;
}
```

```
/*Appendix 7:  02561-01-07-2009.cpp */

#include <GL/glut.h>

void initgl(void);

void display (void);
void axis (void);
void reshape (int w, int h);

int main (int argc, char** argv) {

        glutInit (&argc, argv);
        glutInitDisplayMode(GLUT_RGBA | GLUT_SINGLE);
        glutInitWindowSize (500, 500);
        glutInitWindowPosition (100, 100);
        glutCreateWindow ("Exercise 02501-03");
        initgl ();
        glutDisplayFunc (display);
        glutReshapeFunc (reshape);
        glutMainLoop ();
        return 0;
}


void initgl (void) {
        glClearColor (0., 0., 0., 0.);
        glShadeModel (GL_FLAT);
        glMatrixMode (GL_PROJECTION);
        glLoadIdentity ();
        glOrtho (-6., 6., -6., 6., -6., 6.);
        glMatrixMode (GL_MODELVIEW);

        glLoadIdentity ();
}

void reshape (int w, int h) {
        glViewport (0, 0, (GLsizei) w, (GLsizei) h);
        glMatrixMode (GL_PROJECTION);
        glLoadIdentity ();
        glOrtho (-6., 6., -6., 6., -6., 6.);
        glMatrixMode (GL_MODELVIEW);
        glLoadIdentity ();
}


void display (void) {

        glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glColor3f (1.,1.,1.);

            glutWireCube (1.);

        axis();
        glFlush ();

}


void axis (void) {
        /*Plot part of axis and an auxiliary line*/
        GLfloat v0[] = {0., 0., 0.};
        GLfloat vx[]= {4., 0., 0.};
        GLfloat vy[]= {0, 4., 0.};
        GLfloat vz[]= {0., 0., 4.};
```

```
        GLfloat v0x1[] = {1., 0., 0.};
        GLfloat vyx1[] = {1., 3., 0.};

        glPushAttrib(GL_CURRENT_BIT);
        glColor3f (1., 0., 0.);
        glBegin (GL_LINES);
                glVertex3fv (v0);
                glVertex3fv (vx);
        glEnd ();

        glBegin (GL_LINES);
                glVertex3fv (v0);
                glVertex3fv (vy);
        glEnd ();

        glBegin (GL_LINES);
                glVertex3fv (v0);
                glVertex3fv (vz);
        glEnd ();

        glBegin (GL_LINES);
                glVertex3fv (v0x1);
                glVertex3fv (vyx1);
        glEnd ();
        glPopAttrib();
}
```