Day5: Opengl example

#include <windows.h>  // for MS Windows

#include <GL/glut.h>  // GLUT, include glu.h and gl.h

/\* Initialize OpenGL Graphics \*/

void initGL() {

   // Set "clearing" or background color

   glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Black and opaque

}

void display() {

   glClear(GL\_COLOR\_BUFFER\_BIT);   // Clear the color buffer with current clearing color

   // Define shapes enclosed within a pair of glBegin and glEnd

   glBegin(GL\_QUADS);              // Each set of 4 vertices form a quad

      glColor3f(1.0f, 0.0f, 0.0f); // Red

      glVertex2f(-0.8f, 0.1f);     // Define vertices in counter-clockwise (CCW) order

      glVertex2f(-0.2f, 0.1f);     //  so that the normal (front-face) is facing you

      glVertex2f(-0.2f, 0.7f);

      glVertex2f(-0.8f, 0.7f);

      glColor3f(0.0f, 1.0f, 0.0f); // Green

      glVertex2f(-0.7f, -0.6f);

      glVertex2f(-0.1f, -0.6f);

      glVertex2f(-0.1f,  0.0f);

      glVertex2f(-0.7f,  0.0f);

      glColor3f(0.2f, 0.2f, 0.2f); // Dark Gray

      glVertex2f(-0.9f, -0.7f);

      glColor3f(1.0f, 1.0f, 1.0f); // White

      glVertex2f(-0.5f, -0.7f);

      glColor3f(0.2f, 0.2f, 0.2f); // Dark Gray

      glVertex2f(-0.5f, -0.3f);

      glColor3f(1.0f, 1.0f, 1.0f); // White

      glVertex2f(-0.9f, -0.3f);

   glEnd();

   glBegin(GL\_TRIANGLES);          // Each set of 3 vertices form a triangle

      glColor3f(0.0f, 0.0f, 1.0f); // Blue

      glVertex2f(0.1f, -0.6f);

      glVertex2f(0.7f, -0.6f);

      glVertex2f(0.4f, -0.1f);

      glColor3f(1.0f, 0.0f, 0.0f); // Red

      glVertex2f(0.3f, -0.4f);

      glColor3f(0.0f, 1.0f, 0.0f); // Green

      glVertex2f(0.9f, -0.4f);

      glColor3f(0.0f, 0.0f, 1.0f); // Blue

      glVertex2f(0.6f, -0.9f);

   glEnd();

   glBegin(GL\_POLYGON);            // These vertices form a closed polygon

      glColor3f(1.0f, 1.0f, 0.0f); // Yellow

      glVertex2f(0.4f, 0.2f);

      glVertex2f(0.6f, 0.2f);

      glVertex2f(0.7f, 0.4f);

      glVertex2f(0.6f, 0.6f);

      glVertex2f(0.4f, 0.6f);

      glVertex2f(0.3f, 0.4f);

   glEnd();

   glFlush();  // Render now

}

/\* Handler for window re-size event. Called back when the window first appears and

   whenever the window is re-sized with its new width and height \*/

void reshape(GLsizei width, GLsizei height) {  // GLsizei for non-negative integer

   // Compute aspect ratio of the new window

   if (height == 0) height = 1;                // To prevent divide by 0

   GLfloat aspect = (GLfloat)width / (GLfloat)height;

   // Set the viewport to cover the new window

   glViewport(0, 0, width, height);

   // Set the aspect ratio of the clipping area to match the viewport

   glMatrixMode(GL\_PROJECTION);  // To operate on the Projection matrix

   glLoadIdentity();             // Reset the projection matrix

   if (width >= height) {

     // aspect >= 1, set the height from -1 to 1, with larger width

      gluOrtho2D(-1.0 \* aspect, 1.0 \* aspect, -1.0, 1.0);

   } else {

      // aspect < 1, set the width to -1 to 1, with larger height

     gluOrtho2D(-1.0, 1.0, -1.0 / aspect, 1.0 / aspect);

   }

}

/\* Main function: GLUT runs as a console application starting at main() \*/

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

   glutInit(&argc, argv);          // Initialize GLUT

   glutInitWindowSize(640, 480);   // Set the window's initial width & height - non-square

   glutInitWindowPosition(50, 50); // Position the window's initial top-left corner

   glutCreateWindow("Viewport Transform");  // Create window with the given title

   glutDisplayFunc(display);       // Register callback handler for window re-paint event

   glutReshapeFunc(reshape);       // Register callback handler for window re-size event

   initGL();                       // Our own OpenGL initialization

   glutMainLoop();                 // Enter the infinite event-processing loop

   return 0;

}

OutPut:

