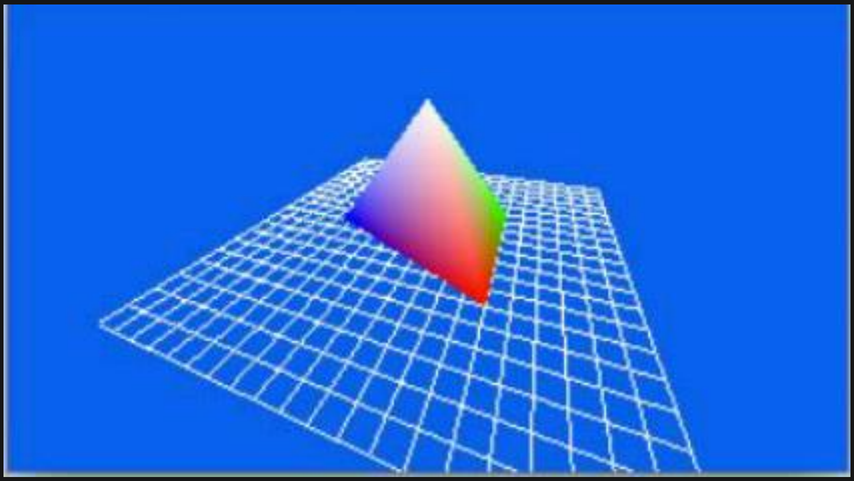
Unity Unive

Section 2

Group name	Group members ID
Beamlak Techane	04563/14
Sebaif Mohammed	04550/14
Tizita Tesfaye	04569/14
Tsion Mulugeta	04571/14
Xenos Wegayew	04557/14
Amanuel Bukune	04521/14

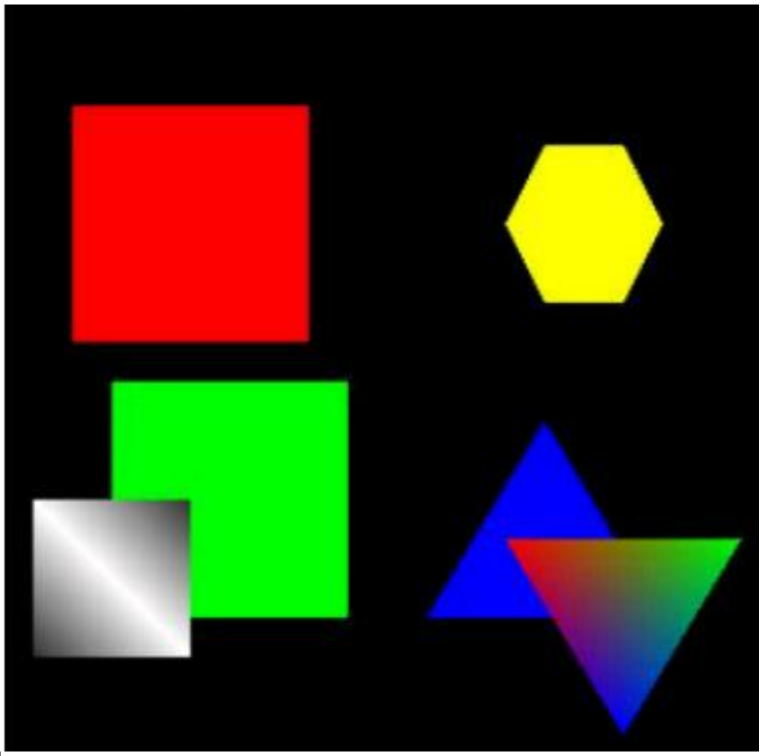
Submitted to Instructor: ketema

1. 

```
// This is a simple introductory program; its main window contains a
static
// picture of a tetrahedron, whose top vertex is white and whose bottom
// vertices are red, green and blue.  The program illustrates viewing by
// defining an object at a convenient location, then transforming it so
that
// it lies within the view volume.  This is a lousy way to do things (it's
// easier to use gluLookAt()), but it's nice to see how viewing is done at
// a very low level.

#ifdef __APPLE_CC__
#include <GLUT/glut.h>
#else
#include <GL/glut.h>
#endif

// Clears the window and draws the tetrahedron.  The tetrahedron
is  easily
// specified with a triangle strip, though the specification really isn't
very
// easy to read.
void display() {
  glClear(GL_COLOR_BUFFER_BIT);

  // Draw a white grid "floor" for the tetrahedron to sit on.
  glColor3f(1.0, 1.0, 1.0);
  glBegin(GL_LINES);
  for (GLfloat i = -2.5; i <= 2.5; i += 0.25) {
    glVertex3f(i, 0, 2.5); glVertex3f(i, 0, -2.5);
    glVertex3f(2.5, 0, i); glVertex3f(-2.5, 0, i);
  }
  glEnd();

  // Draw the tetrahedron.  It is a four sided figure, so when defining it
  // with a triangle strip we have to repeat the last two vertices.
  glBegin(GL_TRIANGLE_STRIP);
    glColor3f(1, 1, 1); glVertex3f(0, 2, 0);
    glColor3f(1, 0, 0); glVertex3f(-1, 0, 1);
    glColor3f(0, 1, 0); glVertex3f(1, 0, 1);
    glColor3f(0, 0, 1); glVertex3f(0, 0, -1.4);
    glColor3f(1, 1, 1); glVertex3f(0, 2, 0);
    glColor3f(1, 0, 0); glVertex3f(-1, 0, 1);
  glEnd();

  glFlush();
}

// Sets up global attributes like clear color and drawing color, enables
and
// initializes any needed modes (in this case we want backfaces culled),
and
// sets up the desired projection and modelview matrices. It is cleaner to
// define these operations in a function separate from main().
void init() {

  // Set the current clear color to sky blue and the current drawing color
to
```

```
  // white.
  glClearColor(0.1, 0.39, 0.88, 1.0);
  glColor3f(1.0, 1.0, 1.0);

  // Tell the rendering engine not to draw backfaces.  Without this code,
  // all four faces of the tetrahedron would be drawn and it is possible
  // that faces farther away could be drawn after nearer to the viewer.
  // Since there is only one closed polyhedron in the whole scene,
  // eliminating the drawing of backfaces gives us the realism we need.
  // THIS DOES NOT WORK IN GENERAL.
  glEnable(GL_CULL_FACE);
  glCullFace(GL_BACK);

  // Set the camera lens so that we have a perspective viewing volume
whose
  // horizontal bounds at the near clipping plane are -2..2 and vertical
  // bounds are -1.5..1.5.  The near clipping plane is 1 unit from the
camera
  // and the far clipping plane is 40 units away.
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glFrustum(-2, 2, -1.5, 1.5, 1, 40);

  // Set up transforms so that the tetrahedron which is defined right at
  // the origin will be rotated and moved into the view volume.  First we
  // rotate 70 degrees around y so we can see a lot of the left side.
  // Then we rotate 50 degrees around x to "drop" the top of the pyramid
  // down a bit.  Then we move the object back 3 units "into the screen".
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  glTranslatef(0, 0, -3);
  glRotatef(50, 1, 0, 0);
  glRotatef(70, 0, 1, 0);
}
```

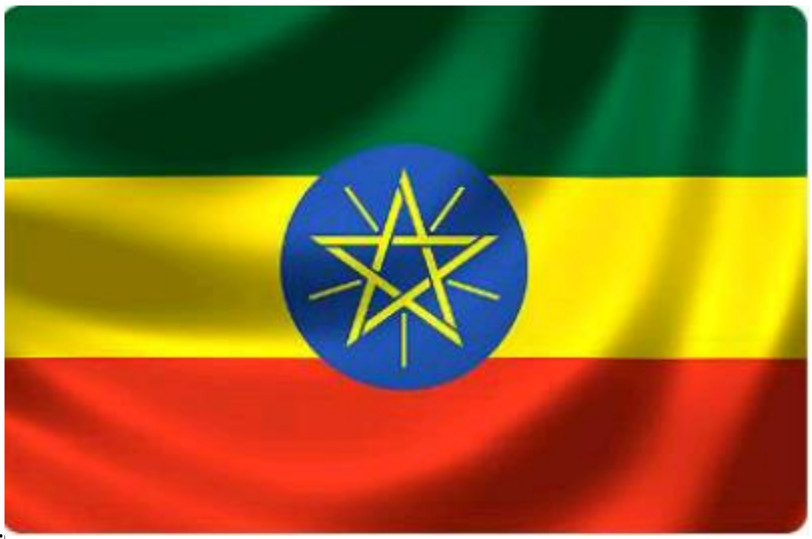
```
// Initializes GLUT, the display mode, and main window; registers
callbacks;
// does application initialization; enters the main event loop.
int main(int argc, char** argv) {
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
  glutInitWindowPosition(80, 80);
  glutInitWindowSize(800, 600);
  glutCreateWindow("A Simple Tetrahedron");
  glutDisplayFunc(display);
  init();
  glutMainLoop();
}
```

2. 

```
       /*
 * GL02Primitive.cpp: Vertex, Primitive and Color
 * Draw Simple 2D colored Shapes: quad, triangle and polygon.
 */
```

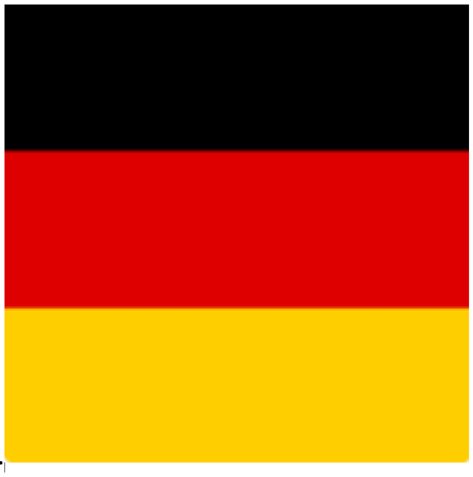
```
#include <windows.h>  // for MS Windows
#include <GL/glut.h>  // GLUT, include glu.h and gl.h

/* Initialize OpenGL Graphics */
void initGL() {
   // Set "clearing" or background color
   glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Black and opaque
}

/* Handler for window-repaint event. Call back when the window first appears and
   whenever the window needs to be re-painted. */
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
}

/* Main function: GLUT runs as a console application starting at main()  */
int main(int argc, char** argv) {
   glutInit(&argc, argv);          // Initialize GLUT
   glutCreateWindow("Vertex, Primitive & Color");  // Create window with the
given title
   glutInitWindowSize(320, 320);   // Set the window's initial width & height
   glutInitWindowPosition(50, 50); // Position the window's initial top-left
corner
   glutDisplayFunc(display);       // Register callback handler for window re-
paint event
   initGL();                       // Our own OpenGL initialization
   glutMainLoop();                 // Enter the event-processing loop
   return 0;
}
```

3.



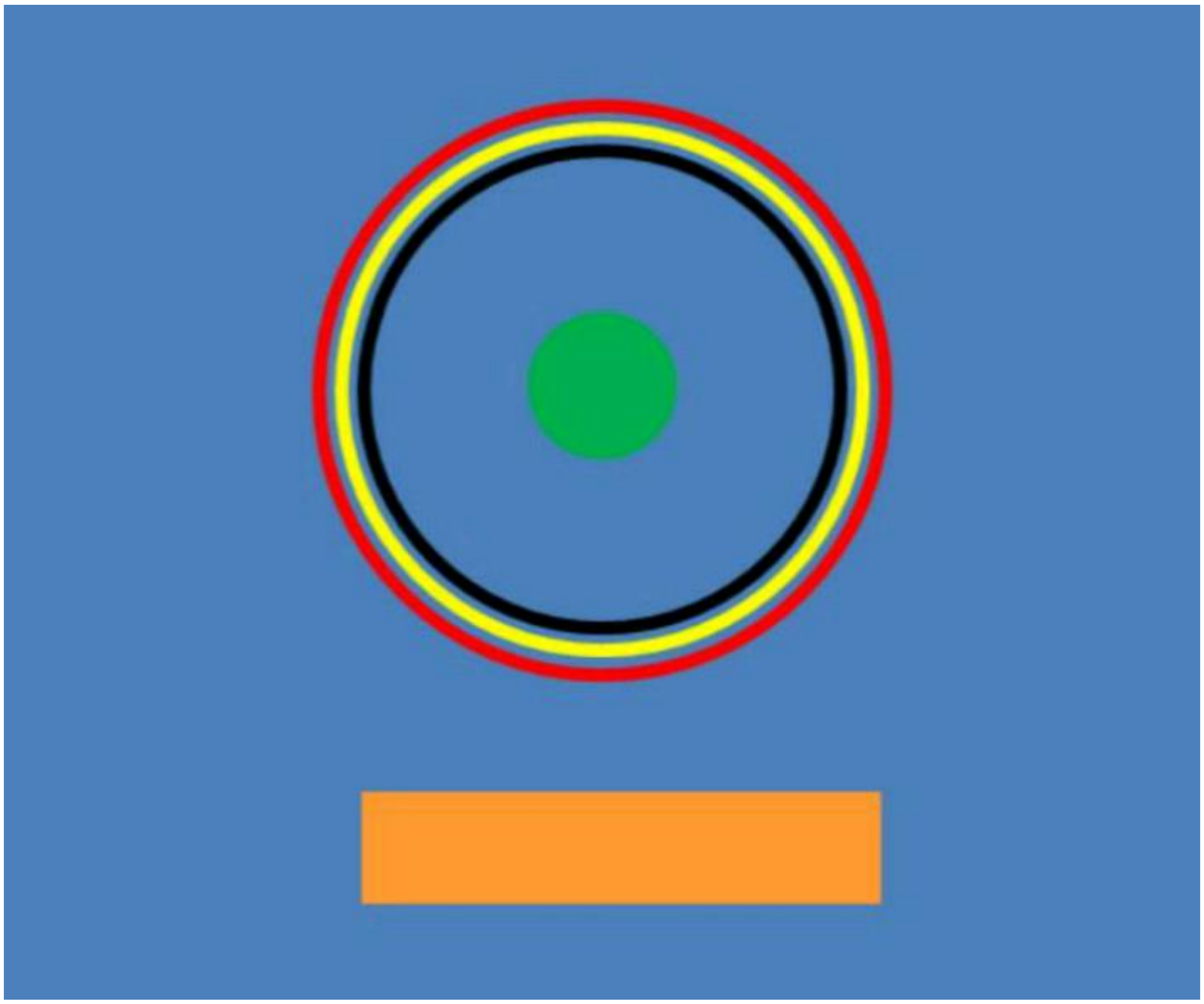
```
#include <GL/glut.h>

#include <math.h>


void drawRectangle(float x, float y, float width, float height, float r, float g, float b) {

   glColor3f(r, g, b);

   glBegin(GL_POLYGON);

   glVertex2f(x, y);

   glVertex2f(x + width, y);

   glVertex2f(x + width, y + height);

   glVertex2f(x, y + height);

   glEnd();

}


void drawStar(float x, float y, float radius, float r, float g, float b) {

   glColor3f(r, g, b);

   glBegin(GL_TRIANGLES);

   glVertex2f(x, y + radius);
```

```
    glVertex2f(x - radius * 0.866, y - radius * 0.5);

    glVertex2f(x + radius * 0.866, y - radius * 0.5);

    glVertex2f(x, y - radius);

    glVertex2f(x - radius * 0.866, y + radius * 0.5);

    glVertex2f(x + radius * 0.866, y + radius * 0.5);

    glEnd();

}


void display() {

    glClear(GL_COLOR_BUFFER_BIT);


    // Draw the green background

    drawRectangle(-1.0, -1.0, 2.0, 1.0, 0.0, 0.5, 0.0);


    // Draw the blue circle

    glColor3f(0.0, 0.0, 1.0);

    glBegin(GL_POLYGON);

    for (int i = 0; i < 360; i++) {

        float angle = i * 3.14159 / 180;

        float x = 0.5 * cos(angle);

        float y = 0.5 * sin(angle);

        glVertex2f(x, y);

    }

    glEnd();
    // Draw the yellow stars
    drawStar(0.1, 0.3, 0.1, 1.0, 1.0, 0.0);
    drawStar(-0.1, -0.3, 0.1, 1.0, 1.0, 0.0);

    glFlush();
}

int main(int argc, char* argv[]) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(640, 480);
    glutCreateWindow("Ethiopian Flag");

    glClearColor(1.0, 1.0, 1.0, 1.0); // Set clear color to white
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-1.0, 1.0, -1.0, 1.0);

    glutDisplayFunc(display);
    glutMainLoop();

    return 0;
}
```

4. 

```
#include <GL/glut.h>


void drawRectangle(float x, float y, float width, float height, float r, float g, float b) {

    glColor3f(r, g, b);

    glBegin(GL_POLYGON);

    glVertex2f(x, y);

    glVertex2f(x + width, y);

    glVertex2f(x + width, y + height);

    glVertex2f(x, y + height);

    glEnd();

}


void display() {

    glClear(GL_COLOR_BUFFER_BIT);


    // Draw the black stripe

    drawRectangle(-1.0, 0.33, 2.0, 0.33, 0.0, 0.0, 0.0);
    // Draw the red stripe

    drawRectangle(-1.0, 0.0, 2.0, 0.33, 1.0, 0.0, 0.0);


    // Draw the yellow stripe

    drawRectangle(-1.0, -0.33, 2.0, 0.33, 1.0, 1.0, 0.0);


    glFlush();

}


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

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);

    glutInitWindowSize(640, 480);

    glutCreateWindow("German Flag");


    glClearColor(1.0, 1.0, 1.0, 1.0); // Set clear color to white

    glMatrixMode(GL_PROJECTION);

    glLoadIdentity();

    gluOrtho2D(-1.0, 1.0, -1.0, 1.0);


    glutDisplayFunc(display);

    glutMainLoop();


    return 0;

}
```

5.



5.

```
#include <GL/freeglut.h>


void drawScene() {

   glClear(GL_COLOR_BUFFER_BIT);


   // Draw blue background

   glClearColor(0.0, 0.0, 1.0, 1.0); // Blue color

   glClear(GL_COLOR_BUFFER_BIT);
```

```
    // Draw green circle

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

    glPushMatrix();

    glTranslatef(0.0, 0.0, 0.0);

    glutSolidSphere(0.2, 50, 50); // You can adjust the radius as needed

    glPopMatrix();


    // Draw concentric rings in different colors

    glColor3f(0.0, 0.0, 0.0); // Black color

    glPushMatrix();

    glTranslatef(0.0, 0.0, 0.0);

    glutSolidTorus(0.3, 0.4, 50, 50);

    glPopMatrix();


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

    // Draw yellow ring


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

    // Draw blue ring


    glColor3f(1.0, 0.0, 0.0);

    glFlush();

}
```

```
int main(int argc, char* argv[]) {

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);

    glutInitWindowSize(640, 480);

    glutDisplayFunc(drawScene);

    glutMainLoop();

  initgraph(&gd, &gm, "");


   setcolor(ORANGE);

   rectangle(100, 100, 200, 200);


   getch();

   closegraph();

    return 0;

}
```