

**UNITY UNIVERSITY**

**Department of Computer Science**

**COMPUTER GRAPHICS**

**Group Assignment**

**Section: 02**

**Members Name ID. Number**

1. **Yosef tirusew 06049/14**
2. **Nafis seyfu 04589/14**
3. **Samuel sileshi 04561/14**
4. **Yohannis abu 06057/14**
5. **Biniam mulat 04565/14**

#include <GL/glut.h>

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

// Draw wireframe grid

glColor3f(1.0, 1.0, 1.0); // Set color to white

glBegin(GL\_LINES);

for (int i = -5; i <= 5; i++) {

glVertex3f(i, 0, -5);

glVertex3f(i, 0, 5);

glVertex3f(-5, 0, i);

glVertex3f(5, 0, i);

}

glEnd();

// Draw pyramid structure

glBegin(GL\_TRIANGLES);

// Front face

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

glVertex3f(0, 1, 0);

glVertex3f(-1, 0, 1);

glVertex3f(1, 0, 1);

// Other faces with different colors

// Add color gradients as needed

glEnd();

glutSwapBuffers();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | GLUT\_DOUBLE | GLUT\_DEPTH);

glutCreateWindow("3D Graphics");

glutDisplayFunc(display);

glEnable(GL\_DEPTH\_TEST);

glutMainLoop();

return 0;

}

2,#include <GL/glut.h>

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

// Draw horizontal stripes - Green, Yellow, Red

glBegin(GL\_QUADS);

// Green stripe

glColor3f(0.0, 0.5, 0.0); // Green color

glVertex2f(-1.0, 0.5);

glVertex2f(1.0, 0.5);

glVertex2f(1.0, 0.0);

glVertex2f(-1.0, 0.0);

// Yellow stripe

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

glVertex2f(-1.0, 0.0);

glVertex2f(1.0, 0.0);

glVertex2f(1.0, -0.5);

glVertex2f(-1.0, -0.5);

// Red stripe

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

glVertex2f(-1.0, -0.5);

glVertex2f(1.0, -0.5);

glVertex2f(1.0, -1.0);

glVertex2f(-1.0, -1.0);

glEnd();

// Draw the blue circle with a yellow star

// Add code to draw the blue circle with a yellow star

glutSwapBuffers();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | GLUT\_DOUBLE | GLUT\_DEPTH);

glutCreateWindow("Flag of Ethiopia");

glutDisplayFunc(display);

glEnable(GL\_DEPTH\_TEST);

glutMainLoop();

return 0;

}

3,The image you provided contains six geometric shapes of different colors and styles on a black background:

1. A solid red square in the upper left corner.

2. A yellow hexagon located above the center to the right.

3. A solid green square below the red square.

4. A gray gradient square below the yellow hexagon, slightly overlapping with the green square.

5. A solid blue triangle pointing upwards in the center towards the bottom.

6. A triangle with a rainbow gradient pointing downwards and overlapping with the blue triangle, creating a star-like shape where they intersect.

```cpp

#include <GL/glut.h>

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

// Draw geometric shapes

// Red square

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

glBegin(GL\_QUADS);

glVertex2f(-0.5, 0.5);

glVertex2f(0.0, 0.5);

glVertex2f(0.0, 0.0);

glVertex2f(-0.5, 0.0);

glEnd();

// Yellow hexagon

// Add code to draw the yellow hexagon

// Green square

// Add code to draw the green square

// Gray gradient square

// Add code to draw the gray gradient square

// Blue triangle

// Add code to draw the blue triangle

// Triangle with rainbow gradient

// Add code to draw the triangle with rainbow gradient

glutSwapBuffers();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | GLUT\_DOUBLE | GLUT\_DEPTH);

glutCreateWindow("Geometric Shapes");

glutDisplayFunc(display);

glEnable(GL\_DEPTH\_TEST);

glutMainLoop();

return 0;

}

```

4.

#include <GL/glut.h>

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

// Draw the flag of Germany

// Black band

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

glBegin(GL\_QUADS);

glVertex2f(-1.0, 1.0);

glVertex2f(1.0, 1.0);

glVertex2f(1.0, 0.33);

glVertex2f(-1.0, 0.33);

glEnd();

// Red band

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

glBegin(GL\_QUADS);

glVertex2f(-1.0, 0.33);

glVertex2f(1.0, 0.33);

glVertex2f(1.0, -0.33);

glVertex2f(-1.0, -0.33);

glEnd();

// Gold (Yellow) band

glColor3f(1.0, 0.84, 0.0); // Gold (Yellow) color

glBegin(GL\_QUADS);

glVertex2f(-1.0, -0.33);

glVertex2f(1.0, -0.33);

glVertex2f(1.0, -1.0);

glVertex2f(-1.0, -1.0);

glEnd();

glutSwapBuffers();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | GLUT\_DOUBLE | GLUT\_DEPTH);

glutCreateWindow("Flag of Germany");

glutDisplayFunc(display);

glEnable(GL\_DEPTH\_TEST);

glutMainLoop();

return 0;

}

```

You can customize the code further to match the exact proportions and dimensions of the flag of Germany.

5,#include <GL/glut.h>

#include <cmath>

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

// Blue Rectangle

glColor3f(0.23921568627450981,0.396078431372549,0.5764705882352941); // Set color to blue (RGB: 0, 0, 1)

glBegin(GL\_QUADS);

glVertex2f(-0.77, -6.0); // Bottom-left corner

glVertex2f(0.77, -6.0); // Bottom-right corner

glVertex2f(0.77, 6.0); // Top-right corner

glVertex2f(-0.77, 6.0); // Top-left corner

glEnd();

// Brown Rectangle Inside

glColor3f(0.784313725490,0.4745098039215686,0.16470588235294117); // Set color to brown (RGB: 0.6, 0.3, 0.0)

glBegin(GL\_QUADS);

// Calculate dimensions based on blue rectangle's height and width

float brownRectWidth = 0.75 \* 2 \* 3 / 4; // 3/4 of blue rectangle's width

float brownRectHeight = 4.0 \* 2 \* 0.1; // 10% of blue rectangle's height

glVertex2f(-brownRectWidth / 2.9, -4.0); // Bottom-left corner

glVertex2f(brownRectWidth / 2.9, -4.0); // Bottom-right corner

glVertex2f(brownRectWidth / 2.9, -4.0 + brownRectHeight); // Top-right corner

glVertex2f(-brownRectWidth / 2.9, -4.0 + brownRectHeight); // Top-left corner

glEnd();

// Set the color to red

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

// Draw a circle (inner red)

glLineWidth(4.0); // Increase border thickness

glBegin(GL\_LINE\_LOOP);

float inner\_red\_radius = 0.55f; // Inner red circle radius

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

float angle = i \* 3.14159f / 180;

float x = inner\_red\_radius \* cos(angle); // Adjust x-coordinate to fit within the rectangle

float y = 5 \* inner\_red\_radius \* sin(angle); // Adjust y-coordinate to fit within the rectangle and increase height by 5x

glVertex2f(x, y);

}

glEnd();

// Set the color to yellow

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

// Draw a smaller yellow circle inside the red circle

glBegin(GL\_LINE\_LOOP);

float smaller\_yellow\_radius = 0.95f \* inner\_red\_radius; // 95% of the original red circle radius

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

float angle = i \* 3.14159f / 180;

float x = smaller\_yellow\_radius \* cos(angle); // Adjust x-coordinate to fit within the rectangle

float y = 5 \* smaller\_yellow\_radius \* sin(angle); // Adjust y-coordinate to fit within the rectangle and increase height by 5x

glVertex2f(x, y);

}

glEnd();

// Set the color to black

glColor3f(0.0f, 0.0f, 0.0f);

// Draw a smaller black circle inside the yellow circle

glBegin(GL\_LINE\_LOOP);

float smaller\_black\_radius = 0.95f \* smaller\_yellow\_radius; // 95% of the yellow circle radius

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

float angle = i \* 3.14159f / 180;

float x = smaller\_black\_radius \* cos(angle); // Adjust x-coordinate to fit within the rectangle

float y = 5 \* smaller\_black\_radius \* sin(angle); // Adjust y-coordinate to fit within the rectangle and increase height by 5x

glVertex2f(x, y);

}

glEnd();

// Set the color to green

glColor3f(0,0.5411764705882353,0.24705882352941178);

// Draw a smaller green circle inside the black circle and fill it

glBegin(GL\_POLYGON);

float smaller\_green\_radius = 0.25f \* smaller\_black\_radius; // 35% of the black circle radius

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

float angle = i \* 3.14159f / 180;

float x = smaller\_green\_radius \* cos(angle); // Adjust x-coordinate

float y = 5 \* smaller\_green\_radius \* sin(angle); // Adjust y-coordinate to fit within the rectangle and increase height by 5x

glVertex2f(x, y);

}

glEnd();

glFlush();

}

void init() {

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

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-1.5, 1.5, -5.0, 5.0); // Set the viewing area

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(400, 800); // Set window size

glutCreateWindow("Nested Rectangles");

init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}