**Rotating Orbit**

**Code:**#include <GL/glut.h>

#include <cmath>

#define PI 3.1415926535898

// Circle properties

float yellowCircleRadius = 0.2f;

float blueCircleRadius = 0.1f;

float redCircleRadius = 0.05f;

// Rotation angles

float blueOrbitAngle = 0.0f; // Blue circle around yellow

float redOrbitAngle = 0.0f; // Red circle around blue

// Speeds

float blueSpeed = 0.5f; // Blue circle orbit speed

float redSpeed = 1.5f; // Red circle orbit speed

void drawCircle(float cx, float cy, float radius, int numSegments, float r, float g, float b) {

glColor3f(r, g, b); // Circle color

glBegin(GL\_POLYGON);

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

float theta = 2.0f \* PI \* float(i) / float(numSegments);

float x = radius \* cosf(theta);

float y = radius \* sinf(theta);

glVertex2f(cx + x, cy + y);

}

glEnd();

}

void drawOrbit(float cx, float cy, float radius) {

glColor3f(0.0f, 0.0f, 0.0f); // Black orbit lines

glBegin(GL\_LINE\_LOOP);

for (int i = 0; i < 100; i++) {

float theta = 2.0f \* PI \* float(i) / float(100);

float x = radius \* cosf(theta);

float y = radius \* sinf(theta);

glVertex2f(cx + x, cy + y);

}

glEnd();

}

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

// Draw orbit lines (black)

drawOrbit(0.0f, 0.0f, 0.5f); // Orbit for blue circle

float blueX = cos(blueOrbitAngle) \* 0.5f;

float blueY = sin(blueOrbitAngle) \* 0.5f;

drawOrbit(blueX, blueY, 0.2f); // Orbit for red circle

// Draw Yellow Circle (center)

drawCircle(0.0f, 0.0f, yellowCircleRadius, 100, 1.0f, 1.0f, 0.0f);

// Draw Blue Circle position (orbiting yellow circle)

drawCircle(blueX, blueY, blueCircleRadius, 50, 0.0f, 0.0f, 1.0f);

// Draw Red Circle position (orbiting blue circle)

float redX = blueX + cos(redOrbitAngle) \* 0.2f;

float redY = blueY + sin(redOrbitAngle) \* 0.2f;

drawCircle(redX, redY, redCircleRadius, 50, 1.0f, 0.0f, 0.0f);

glutSwapBuffers();

}

void update(int value) {

// Update angles for rotation

blueOrbitAngle += blueSpeed \* PI / 180.0f; // Convert speed to radians

if (blueOrbitAngle > 2 \* PI) blueOrbitAngle -= 2 \* PI;

redOrbitAngle += redSpeed \* PI / 180.0f;

if (redOrbitAngle > 2 \* PI) redOrbitAngle -= 2 \* PI;

glutPostRedisplay(); // Request to redraw the screen

glutTimerFunc(16, update, 0); // Approx ~60 FPS (16ms delay)

}

void init() {

glClearColor(1.0, 1.0, 1.0, 1.0); // White background

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-1.0, 1.0, -1.0, 1.0); // Set coordinate system

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(800, 600);

glutCreateWindow("Rotating Circles Animation with White Background");

init();

glutDisplayFunc(display);

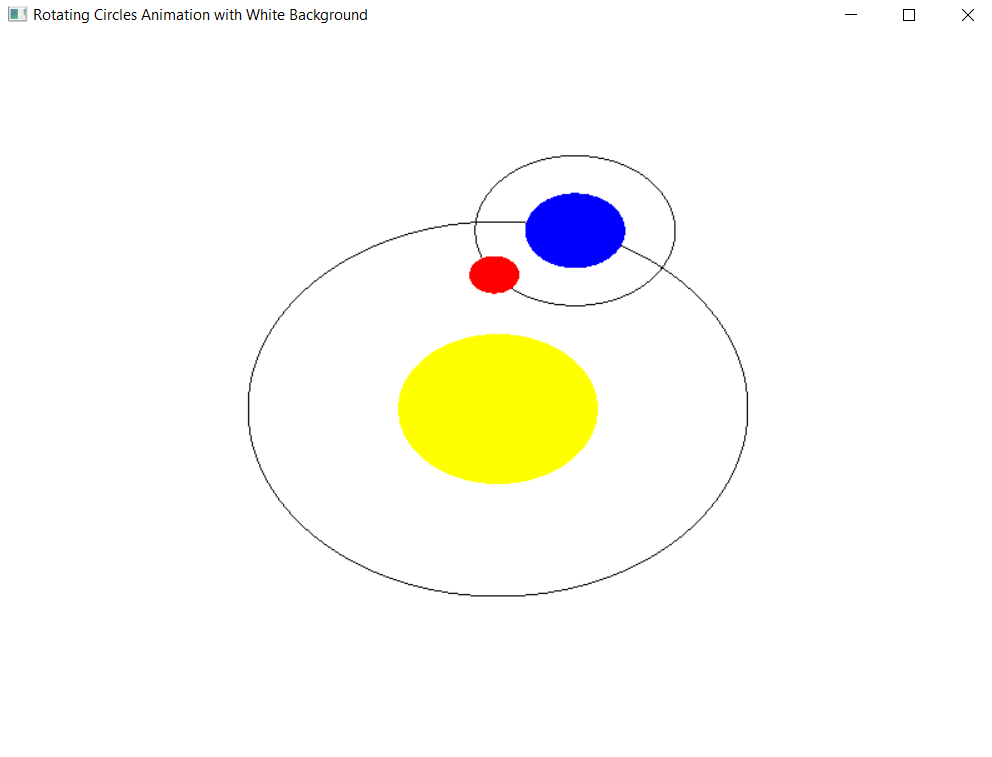
glutTimerFunc(0, update, 0);

glutMainLoop();

return 0;

}

**Output:**

****