LAPORAN RESMI

MINGGU 9-11

Grafika Komputer

Nama : Muhammad Zaid Abdillah

Kelas : 3 D4 Informatika A

NRP : 2110191013

Minggu 9

Listing Program:

```
#include <GL/glut.h>
#include <math.h>
#include <iostream>
typedef struct {
    float x;
    float y;
} Point2D_t;
typedef struct {
    float r;
    float g;
    float b;
} Color_t;
typedef struct {
    float m[3][3];
} Matrix3D_t;
typedef struct {
    float v[3];
} Vector3D_t;
typedef struct {
    float x;
    float y;
    float z;
} Point3D_t;
typedef struct {
    int NumberofVertices;
    long int pnt[32];
} face_t;
typedef struct {
    int NumberofVertices;
    Point3D_t pnt[100];
    int NumberofFaces;
    face_t fc[32];
} Object3D_t;
Vector3D_t vec3D;
Matrix3D_t matrix3DX, matrix3DY, matrix3DZ;
float sudut = 0.0;
void drawDot(int x, int y)
{
    glColor3f(0.0, 0.0, 1.0);
    glPointSize(5);
    glBegin(GL_POINTS);
    glVertex2i(x, y);
```

```
    glEnd();
}
void drawPolyline(Point3D_t pnt[], int n, Color_t col)
{
    int i;
    glColor3d(col.r, col.g, col.b);
    glBegin(GL_LINE_LOOP);
    for (i = 0; i < n; i++) {
        glVertex2f(pnt[i].x, pnt[i].y);
    }
    glEnd();
}
void drawLine(Point2D_t pnt[], int n, Color_t col) {
    int i;
    glColor3d(col.r, col.g, col.b);
    glBegin(GL_LINES);
    for (i = 0; i < n; i++) {
        glVertex2f(pnt[i].x, pnt[i].y);
    }
    glEnd();
}
void sumbu_koordinat() {
    Point2D_t sumbuX[2] = { { -300.0,0.0 }, { 300.0,0.0 } };
    Point2D_t sumbuY[2] = { { 0.0,-300.0 }, { 0.0,300.0 } };
    Color_t col = { 0.0,0.0,1.0 };
    drawLine(sumbuX, 2, col);
    drawLine(sumbuY, 2, col);
}
Matrix3D_t createIdentity()
{
    Matrix3D_t rotate;
    rotate.m[0][0] = 0.0; rotate.m[0][1] = 0.0; rotate.m[0][2] = 0.0;
    rotate.m[1][0] = 0.0; rotate.m[1][1] = 0.0; rotate.m[1][2] = 0.0;
    rotate.m[2][0] = 0.0; rotate.m[2][1] = 0.0; rotate.m[2][2] = 0.0;
    return rotate;
}
Matrix3D_t rotationX(float teta)
{
    Matrix3D_t rotate = createIdentity();
    rotate.m[0][0] = 1.0; rotate.m[0][1] = 0.0; rotate.m[0][2] = 0.0;
    rotate.m[1][0] = 0.0; rotate.m[1][1] = cos(teta / 57.3);
    rotate.m[1][2] = -sin(teta / 57.3);
    rotate.m[2][0] = 0.0; rotate.m[2][1] = sin(teta / 57.3);
    rotate.m[2][2] = cos(teta / 57.3);
    return rotate;
```

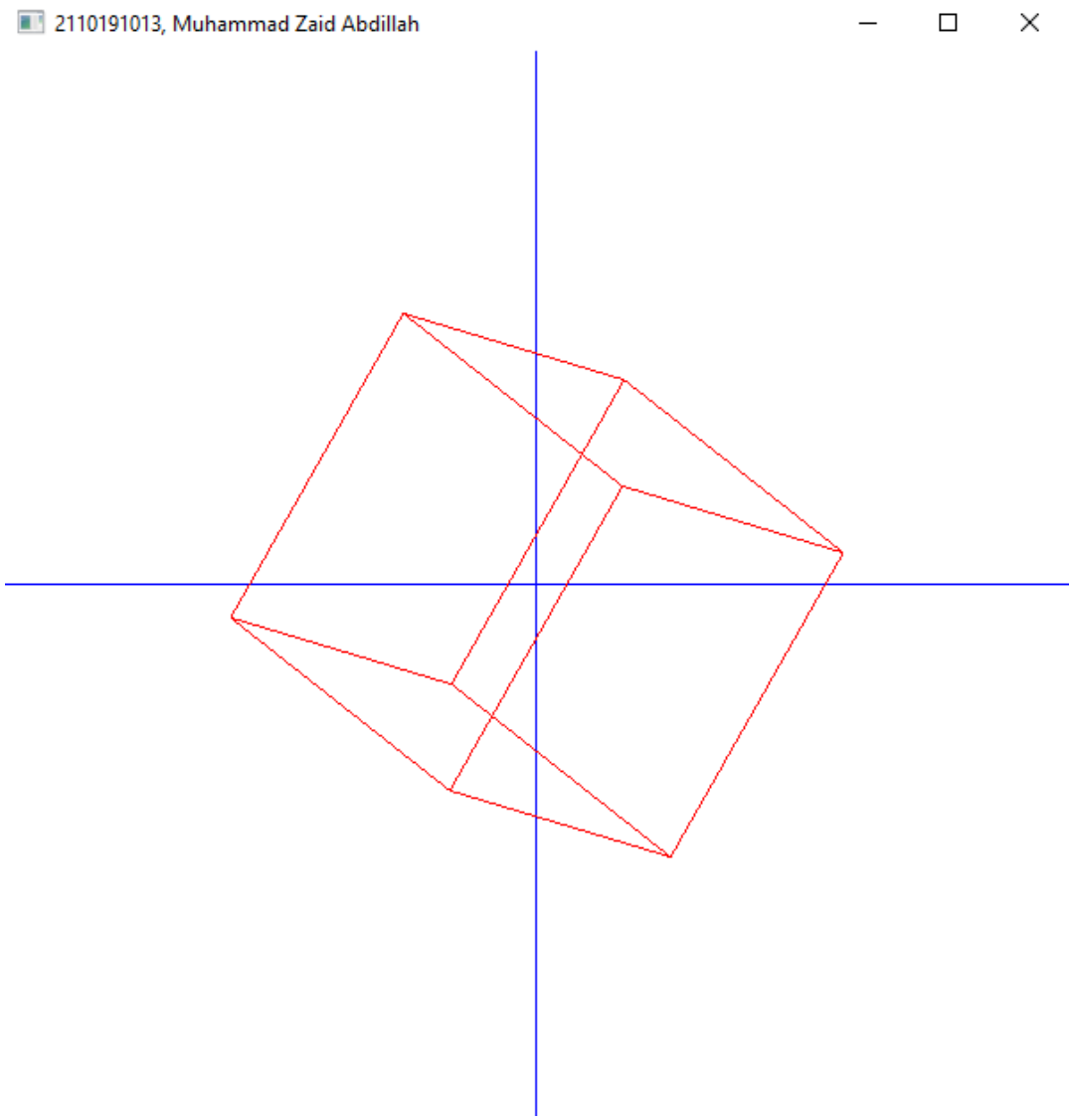
```
}
Matrix3D_t rotationY(float teta)
{
    Matrix3D_t rotate = createIdentity();
    rotate.m[0][0] = cos(teta / 57.3); rotate.m[0][1] = 0.0;
    rotate.m[0][2] = sin(teta / 57.3);
    rotate.m[1][0] = 0.0; rotate.m[1][1] = 1.0; rotate.m[1][2] = 0.0;
    rotate.m[2][0] = -sin(teta / 57.3); rotate.m[2][1] = 0.0;
    rotate.m[2][2] = cos(teta / 57.3);
    return rotate;
}
Matrix3D_t rotationZ(float teta)
{
    Matrix3D_t rotate = createIdentity();
    rotate.m[0][0] = cos(teta / 57.3); rotate.m[0][1] = -sin(teta /
        57.3); rotate.m[0][2] = 0.0;
    rotate.m[1][0] = sin(teta / 57.3); rotate.m[1][1] = cos(teta / 57.3);
    rotate.m[1][2] = 0.0;
    rotate.m[2][0] = 0.0; rotate.m[2][1] = 0.0; rotate.m[2][2] = 1.0;
    return rotate;
}
Vector3D_t Point2Vector3D(Point3D_t pnt)
{
    Vector3D_t vec;
    vec.v[0] = pnt.x;
    vec.v[1] = pnt.y;
    vec.v[2] = pnt.z;
    return vec;
}
Point3D_t Vector2Point3D(Vector3D_t vec)
{
    Point3D_t pnt;
    pnt.x = vec.v[0];
    pnt.y = vec.v[1];
    pnt.z = vec.v[2];
    return pnt;
}
Vector3D_t operator *(Matrix3D_t a, Vector3D_t b)
{
    Vector3D_t c;
    int i, j, k;
    for (i = 0; i < 3; i++) {
        c.v[i] = 0;
        for (j = 0; j < 3; j++) {
            c.v[i] += a.m[i][j] * b.v[j];
```

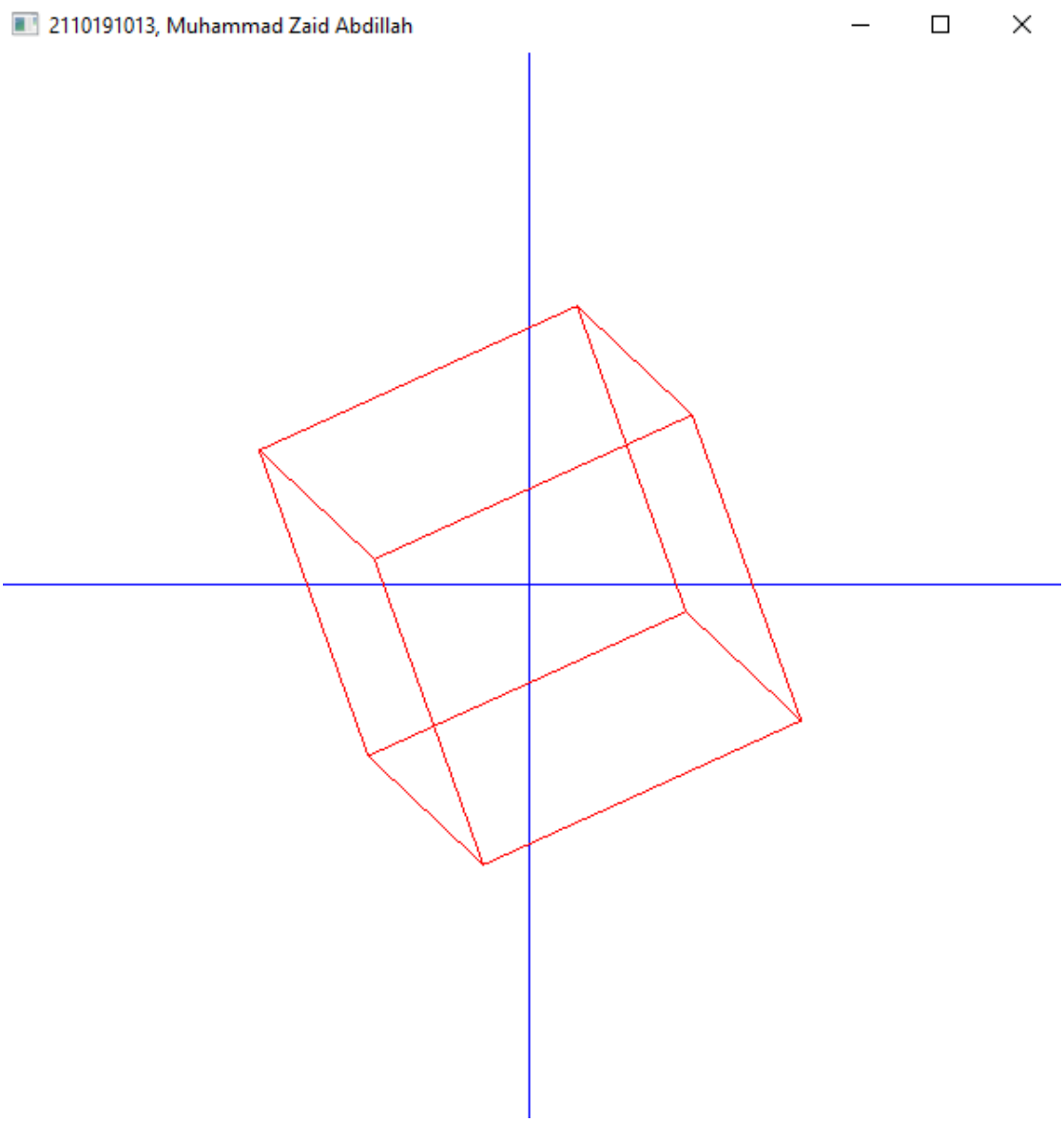
```
        }
    }
    return c;
}
void drawObject3D(Object3D_t obyek, int n)
{
    matrix3DX = rotationX(sudut);
    matrix3DY = rotationY(sudut);
    matrix3DZ = rotationZ(sudut);
    int numTitikFace;
    Vector3D_t vecbuff[3];
    Color_t col = { 1.0,0.0,0.0 };
    int result;
    //Ditransformasi dulu
    for (int i = 0; i < obyek.NumberofVertices; i++) {
        vec3D = Point2Vector3D(obyek.pnt[i]);
        vec3D = operator*(matrix3DZ, vec3D);
        vec3D = operator*(matrix3DY, vec3D);
        vec3D = operator*(matrix3DX, vec3D);
        obyek.pnt[i] = Vector2Point3D(vec3D);
    }
    sudut = sudut + 1;
    Point3D_t pnt3D[4];
    Point3D_t pnt2D[4];
    Vector3D_t normalVector;
    Vector3D_t vecbuff1[4];
    for (int i = 0; i < obyek.NumberofFaces; i++) {
        for (int j = 0; j < obyek.fc[i].NumberofVertices; j++) {
            pnt3D[j] = obyek.pnt[obyek.fc[i].pnt[j]];
            pnt2D[j].x = pnt3D[j].x;
            pnt2D[j].y = pnt3D[j].y;
            vecbuff1[j] = Point2Vector3D(pnt3D[j]);
        }
        drawPolyline(pnt2D, obyek.fc[i].NumberofVertices, col);
    }
}
void buat_kubus01()
{
    Object3D_t kubus =
    { 8,
    {
    {-100,-100,100},
    {100,-100,100},
    {100,100,100},
    {-100,100,100},
```

```
    {-100,-100,-100},
    {100,-100,-100},
    {100,100,-100},
    {-100,100,-100}
    },
     6,
    {
    {4,{0,1,2,3}},
    {4,{1,5,6,2}},
    {4,{4,7,6,5}},
    {4,{0,3,7,4}},
    {4,{4,5,1,0}},
    {4,{2,3,7,6}}
    }
    };
    /**std::cout << "\nJumlah titik : " << kubus.NumberofVertices;
    std::cout << "\nJumlah face : " << kubus.NumberofFaces;
    std::cout << "\nTitik ke-0 x : " << kubus.pnt[0].x;
    std::cout << "\nTitik ke-0 y : " << kubus.pnt[0].y;
    std::cout << "\nTitik ke-0 z : " << kubus.pnt[0].z;
    std::cout << "\nFace ke-0.0 : " << kubus.fc[0].pnt[0];*/
    drawObject3D(kubus, 4);
}
void draw_3D()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 0.0, 0.0);
    sumbu_koordinat();
    buat_kubus01();
    glFlush();
}
void timer(int value)
{
    glutPostRedisplay();
    glutTimerFunc(50, timer, 0);
}
void Initialize() {
    glClearColor(1.0, 1.0, 1.0, 0.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-300, 300, -300, 300);
}
int main(int iArgc, char** cppArgv) {
    glutInit(&iArgc, cppArgv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
```

```
    glutInitWindowPosition(50, 50);
    glutInitWindowSize(600, 600);
    glutCreateWindow("2110191013, Muhammad Zaid Abdillah");
    Initialize();
    glutDisplayFunc(draw_3D);
    glutTimerFunc(1, timer, 0);
    glutMainLoop();
    return 0;
}
```

Output:





Minggu 10:

Listing Program:

```
#include <GL/glut.h>
#include <math.h>
#include <iostream>
typedef struct {
    float x;
    float y;
} Point2D_t;
typedef struct {
    float r;
    float g;
```

```
    float b;
} Color_t;
typedef struct {
    float m[3][3];
} Matrix3D_t;
typedef struct {
    float v[3];
} Vector3D_t;
typedef struct {
    float x;
    float y;
    float z;
} Point3D_t;
typedef struct {
    int NumberofVertices;
    long int pnt[32];
} face_t;
typedef struct {
    int NumberofVertices;
    Point3D_t pnt[100];
    int NumberofFaces;
    face_t fc[32];
} Object3D_t;
Vector3D_t vec3D;
Matrix3D_t matrix3DX, matrix3DY, matrix3DZ;
float sudut = 0.0;
void drawDot(int x, int y)
{
    glColor3f(0.0, 0.0, 1.0);
    glPointSize(5);
    glBegin(GL_POINTS);
    glVertex2i(x, y);
    glEnd();
}
void drawPolyline(Point3D_t pnt[], int n, Color_t col)
{
    int i;
    glColor3d(col.r, col.g, col.b);
    glBegin(GL_LINE_LOOP);
    for (i = 0; i < n; i++) {
        glVertex2f(pnt[i].x, pnt[i].y);
    }
    glEnd();
}
void drawLine(Point2D_t pnt[], int n, Color_t col) {
```

```
    int i;
    glColor3d(col.r, col.g, col.b);
    glBegin(GL_LINES);
    for (i = 0; i < n; i++) {
        glVertex2f(pnt[i].x, pnt[i].y);
    }
    glEnd();
}
void sumbu_koordinat() {
    Point2D_t sumbuX[2] = { { -300.0,0.0 }, { 300.0,0.0 } };
    Point2D_t sumbuY[2] = { { 0.0,-300.0 }, { 0.0,300.0 } };
    Color_t col = { 0.0,0.0,1.0 };
    drawLine(sumbuX, 2, col);
    drawLine(sumbuY, 2, col);
}
Matrix3D_t createIdentity()
{
    Matrix3D_t rotate;
    rotate.m[0][0] = 0.0; rotate.m[0][1] = 0.0; rotate.m[0][2] = 0.0;
    rotate.m[1][0] = 0.0; rotate.m[1][1] = 0.0; rotate.m[1][2] = 0.0;
    rotate.m[2][0] = 0.0; rotate.m[2][1] = 0.0; rotate.m[2][2] = 0.0;
    return rotate;
}
Matrix3D_t rotationX(float teta)
{
    Matrix3D_t rotate = createIdentity();
    rotate.m[0][0] = 1.0; rotate.m[0][1] = 0.0; rotate.m[0][2] = 0.0;
    rotate.m[1][0] = 0.0; rotate.m[1][1] = cos(teta / 57.3);
    rotate.m[1][2] = -sin(teta / 57.3);
    rotate.m[2][0] = 0.0; rotate.m[2][1] = sin(teta / 57.3);
    rotate.m[2][2] = cos(teta / 57.3);
    return rotate;
}
Matrix3D_t rotationY(float teta)
{
    Matrix3D_t rotate = createIdentity();
    rotate.m[0][0] = cos(teta / 57.3); rotate.m[0][1] = 0.0;
    rotate.m[0][2] = sin(teta / 57.3);
    rotate.m[1][0] = 0.0; rotate.m[1][1] = 1.0; rotate.m[1][2] = 0.0;
    rotate.m[2][0] = -sin(teta / 57.3); rotate.m[2][1] = 0.0;
    rotate.m[2][2] = cos(teta / 57.3);
    return rotate;
}
Matrix3D_t rotationZ(float teta)
{
```

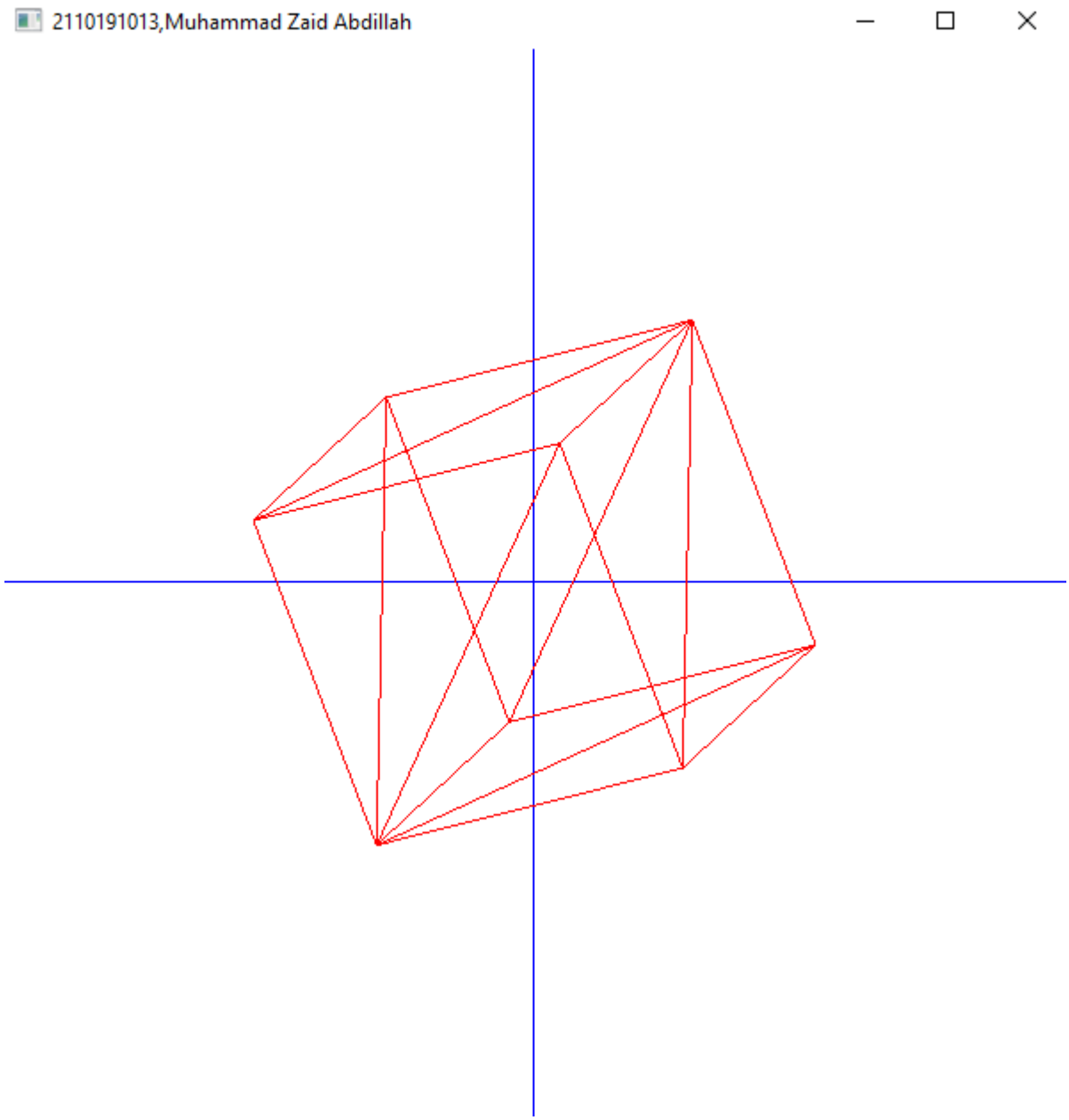
```
    Matrix3D_t rotate = createIdentity();
    rotate.m[0][0] = cos(teta / 57.3); rotate.m[0][1] = -sin(teta /
        57.3); rotate.m[0][2] = 0.0;
    rotate.m[1][0] = sin(teta / 57.3); rotate.m[1][1] = cos(teta / 57.3);
    rotate.m[1][2] = 0.0;
    rotate.m[2][0] = 0.0; rotate.m[2][1] = 0.0; rotate.m[2][2] = 1.0;
    return rotate;
}
Vector3D_t Point2Vector3D(Point3D_t pnt)
{
    Vector3D_t vec;
    vec.v[0] = pnt.x;
    vec.v[1] = pnt.y;
    vec.v[2] = pnt.z;
    return vec;
}
Point3D_t Vector2Point3D(Vector3D_t vec)
{
    Point3D_t pnt;
    pnt.x = vec.v[0];
    pnt.y = vec.v[1];
    pnt.z = vec.v[2];
    return pnt;
}
Vector3D_t operator *(Matrix3D_t a, Vector3D_t b)
{
    Vector3D_t c;
    int i, j, k;
    for (i = 0; i < 3; i++) {
        c.v[i] = 0;
        for (j = 0; j < 3; j++) {
            c.v[i] += a.m[i][j] * b.v[j];
        }
    }
    return c;
}
void drawObject3D(Object3D_t obyek, int n)
{
    matrix3DX = rotationX(sudut);
    matrix3DY = rotationY(sudut);
    matrix3DZ = rotationZ(sudut);
    int numTitikFace;
    Vector3D_t vecbuff[3];
    Color_t col = { 1.0,0.0,0.0 };
    int result;
```

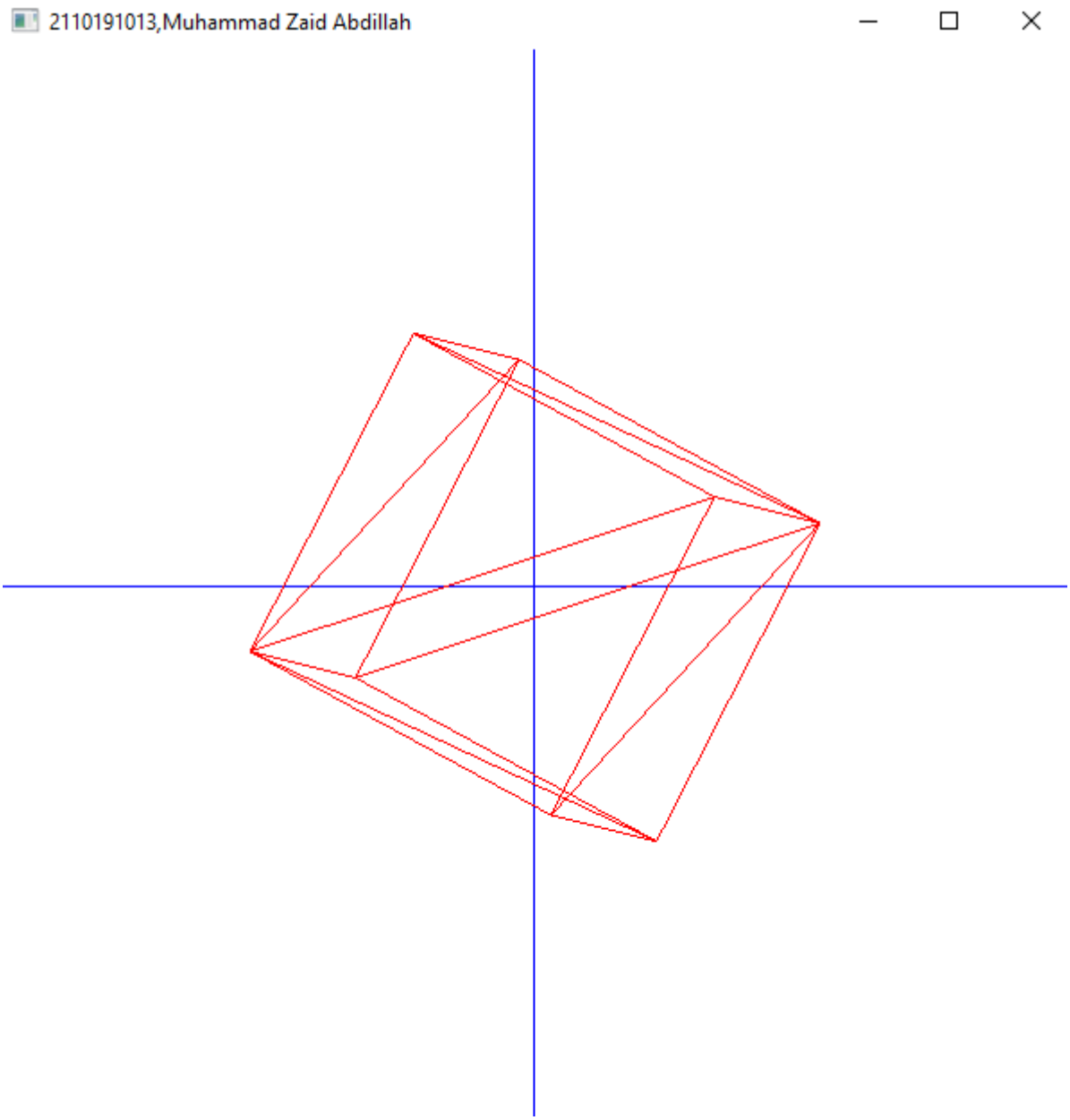
```
    //Ditransformasi dulu
    for (int i = 0; i < obyek.NumberofVertices; i++) {
        vec3D = Point2Vector3D(obyek.pnt[i]);
        vec3D = operator*(matrix3DZ, vec3D);
        vec3D = operator*(matrix3DY, vec3D);
        vec3D = operator*(matrix3DX, vec3D);
        obyek.pnt[i] = Vector2Point3D(vec3D);
    }
    sudut = sudut + 1;
    Point3D_t pnt3D[4];
    Point3D_t pnt2D[4];
    Vector3D_t normalVector;
    Vector3D_t vecbuff1[4];
    for (int i = 0; i < obyek.NumberofFaces; i++) {
        for (int j = 0; j < obyek.fc[i].NumberofVertices; j++) {
            pnt3D[j] = obyek.pnt[obyek.fc[i].pnt[j]];
            pnt2D[j].x = pnt3D[j].x;
            pnt2D[j].y = pnt3D[j].y;
            vecbuff1[j] = Point2Vector3D(pnt3D[j]);
        }
        drawPolyline(pnt2D, obyek.fc[i].NumberofVertices, col);
    }
}
void buat_kubus02()
{
    Object3D_t kubus =
    { 8,
    {
    {-100,-100,100},
    {100,-100,100},
    {100,100,100},
    {-100,100,100},
    {-100,-100,-100},
    {100,-100,-100},
    {100,100,-100},
    {-100,100,-100}
    },
     12,
    {
    {3,{0,3,2}},
    {3,{0,2,1}},
    {3,{1,6,2}},
    {3,{1,5,6}},
    {3,{4,5,6}},
    {3,{4,6,7}},
```

```
    {3,{3,0,7}},
    {3,{0,4,7}},
    {3,{3,2,6}},
    {3,{3,6,7}},
    {3,{0,1,5}},
    {3,{0,5,4}},
    }
    };
    /**std::cout << "\nJumlah titik : " << kubus.NumberofVertices;
    std::cout << "\nJumlah face : " << kubus.NumberofFaces;
    std::cout << "\nTitik ke-0 x : " << kubus.pnt[0].x;
    std::cout << "\nTitik ke-0 y : " << kubus.pnt[0].y;
    std::cout << "\nTitik ke-0 z : " << kubus.pnt[0].z;
    std::cout << "\nFace ke-0.0 : " << kubus.fc[0].pnt[0];*/
    drawObject3D(kubus, 3);
}
void draw_3D()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 0.0, 0.0);
    sumbu_koordinat();
    buat_kubus02();
    glFlush();
}
void timer(int value)
{
    glutPostRedisplay();
    glutTimerFunc(50, timer, 0);
}
void Initialize() {
    glClearColor(1.0, 1.0, 1.0, 0.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-300, 300, -300, 300);
}
int main(int iArgc, char** cppArgv) {
    glutInit(&iArgc, cppArgv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowPosition(50, 50);
    glutInitWindowSize(600, 600);
    glutCreateWindow("2110191013,Muhammad Zaid Abdillah");
    Initialize();
    glutDisplayFunc(draw_3D);
    glutTimerFunc(1, timer, 0);
    glutMainLoop();
```

```
    return 0;
}
```

Output:





Minggu 11:

Listing Program:

```
#include <iostream>
using namespace std;
#include <GL/glut.h>
#include <math.h>
using namespace std;
typedef struct {
    float x;
    float y;
```

```
    float z;
} Point3D_t;
typedef struct {
    float x;
    float y;
} Point2D_t;
typedef struct {
    float r;
    float g;
    float b;
} Color_t;
typedef struct {
    float v[3];
} Vector3D_t;
typedef struct {
    float m[3][3];
} Matrix3D_t;
typedef struct {
    int NumberOfVertices;
    short int point[32];
} Face_t;
typedef struct {
    int NumberOfVertices;
    Point3D_t point[100];
    int NumberOfFaces;
    Face_t face[32];
} Object3D_t;
Vector3D_t point2vector(Point2D_t pnt) {
    Vector3D_t vec = { pnt.x,pnt.y,1 };
    return vec;
}
Vector3D_t Point2Vector3D(Point3D_t point) {
    Vector3D_t vector;
    vector.v[0] = point.x;
    vector.v[1] = point.y;
    vector.v[2] = point.z;
    return vector;
}
Point3D_t Vector2Point3D(Vector3D_t vector) {
    Point3D_t point;
    point.x = vector.v[0];
    point.y = vector.v[1];
    point.z = vector.v[2];
    return point;
}
```

```
Matrix3D_t operator*(Matrix3D_t a, Matrix3D_t b) {
    Matrix3D_t c; //c = a * b
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            c.m[i][j] = 0;
            for (int k = 0; k < 3; k++)
                c.m[i][j] += a.m[i][k] * b.m[k][j];
        }
    }
    return c;
}
Vector3D_t operator*(Matrix3D_t a, Vector3D_t b) {
    Vector3D_t c; //c = a * b
    for (int i = 0; i < 3; i++) {
        c.v[i] = 0;
        for (int j = 0; j < 3; j++)
            c.v[i] += a.m[i][j] * b.v[j];
    }
    return c;
}
Point2D_t vector2point(Vector3D_t vec) {
    Point2D_t pnt = { vec.v[0],vec.v[1] };
    return pnt;
}
Matrix3D_t createIdentity() {
    Matrix3D_t i = { {
    {1,0,0},
    {0,1,0},
    {0,0,1}
    } };
    return i;
}
Matrix3D_t translationMatrix(float dx, float dy) {
    Matrix3D_t trans = createIdentity();
    trans.m[0][2] = dx;
    trans.m[1][2] = dy;
    return trans;
}
Matrix3D_t scalingMatrix(float mx, float my) {
    Matrix3D_t scl = createIdentity();
    scl.m[0][0] = mx;
    scl.m[1][1] = my;
    return scl;
}
Matrix3D_t rotationMatrix(float theta) {
```

```
    Matrix3D_t rot = createIdentity();
    float cs = cos(theta);
    float sn = sin(theta);
    rot.m[0][0] = cs;
    rot.m[0][1] = -sn;
    rot.m[1][0] = sn;
    rot.m[1][1] = cs;
    return rot;
}
Matrix3D_t rotationX(float theta) {
    Matrix3D_t rotate = createIdentity();
    rotate.m[0][0] = 1.0; rotate.m[0][1] = 0.0; rotate.m[0][2] = 0.0;
    rotate.m[1][0] = 0.0; rotate.m[1][1] = cos(theta / 57.3);
    rotate.m[1][2] = -
        sin(theta / 57.3);
    rotate.m[2][0] = 0.0; rotate.m[2][1] = sin(theta / 57.3);
    rotate.m[2][2] =
        cos(theta / 57.3);
    return rotate;
}
Matrix3D_t rotationY(float theta) {
    Matrix3D_t rotate = createIdentity();
    rotate.m[0][0] = cos(theta / 57.3); rotate.m[0][1] = 0.0;
    rotate.m[0][2] =
        sin(theta / 57.3);
    rotate.m[1][0] = 0.0; rotate.m[1][1] = 1.0; rotate.m[1][2] = 0.0;
    rotate.m[2][0] = -sin(theta / 57.3); rotate.m[2][1] = 0.0;
    rotate.m[2][2] =
        cos(theta / 57.3);
    return rotate;
}
Matrix3D_t rotationZ(float theta) {
    Matrix3D_t rotate = createIdentity();
    rotate.m[0][0] = cos(theta / 57.3); rotate.m[0][1] = -sin(theta /
        57.3);
    rotate.m[0][2] = 0.0;
    rotate.m[1][0] = sin(theta / 57.3); rotate.m[1][1] = cos(theta /
        57.3);
    rotate.m[1][2] = 0.0;
    rotate.m[2][0] = 0.0; rotate.m[2][1] = 0.0; rotate.m[2][2] = 1.0;
    return rotate;
}
Matrix3D_t operatorKali(Matrix3D_t a, Matrix3D_t b) {
    Matrix3D_t c;
    for (int i = 0; i < 3; i++) {
```

```
        for (int j = 0; j < 3; j++) {
            c.m[i][j] = 0;
            for (int k = 0; k < 3; k++) {
                c.m[i][j] += a.m[i][k] * b.m[k][j];
            }
        }
    }
    return c;
}
Vector3D_t operatorKali(Matrix3D_t a, Vector3D_t b) {
    Vector3D_t c;
    for (int i = 0; i < 3; i++) {
        c.v[i] = 0;
        for (int j = 0; j < 3; j++) {
            c.v[i] += a.m[i][j] * b.v[j];
        }
    }
    return c;
}
void timer(int value) {
    glutPostRedisplay();
    glutTimerFunc(50, timer, 0);
}
void set_color(Color_t col) {
    glColor3f(col.r, col.g, col.b);
}
void draw_dot(Point2D_t pnt, Color_t col) {
    set_color(col);
    glPointSize(10);
    glBegin(GL_POINTS);
    glVertex2d(pnt.x, pnt.y);
    glEnd();
}
void draw_polyline(Point2D_t pnt[], int n, Color_t col) {
    set_color(col);
    glBegin(GL_LINE_STRIP);
    for (int i = 0; i < n; i++) {
        glVertex2f(pnt[i].x, pnt[i].y);
    }
    glEnd();
}
void draw_polygon(Point2D_t pnt[], int n, Color_t col) {
    set_color(col);
    glBegin(GL_LINE_LOOP);
    for (int i = 0; i < n; i++) {
```

```
        glVertex2f(pnt[i].x, pnt[i].y);
    }
    glEnd();
}
void fill_polygon(Point2D_t pnt[], int n, Color_t col) {
    set_color(col);
    glBegin(GL_POLYGON);
    for (int i = 0; i < n; i++) {
        glVertex2f(pnt[i].x, pnt[i].y);
    }
    glEnd();
}
void Initialize() {
    glClearColor(1.0, 1.0, 1.0, 0.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-400, 400, -400, 400);
}
float angle;
Matrix3D_t matrix3DX;
Matrix3D_t matrix3DY;
Matrix3D_t matrix3DZ;
Vector3D_t vector3D;
void lingkaran(Point2D_t point, float size, Color_t col) {
    glBegin(GL_POINT);
    glPointSize(size);
    draw_dot(point, col);
    glEnd();
}
void sumbu_koordinat() {
    Point2D_t sumbuX[2] = { {-320,0},{320,0} };
    Point2D_t sumbuY[2] = { {0,-240},{0,240} };
    Color_t col = { 0,0,1 };
    draw_polyline(sumbuX, 2, col);
    draw_polyline(sumbuY, 2, col);
}
void drawFillLingkaran(float r, int x, int y, Color_t color) {
    set_color(color);
    Point2D_t lingkaran[360];
    for (int i = 0; i < 360; i++) {
        lingkaran[i].x = (float)(r * sin(i * 3.14 / 180) + x);
        lingkaran[i].y = (float)(r * cos(i * 3.14 / 180) + y);
    }
    fill_polygon(lingkaran, 360, color);
}
```

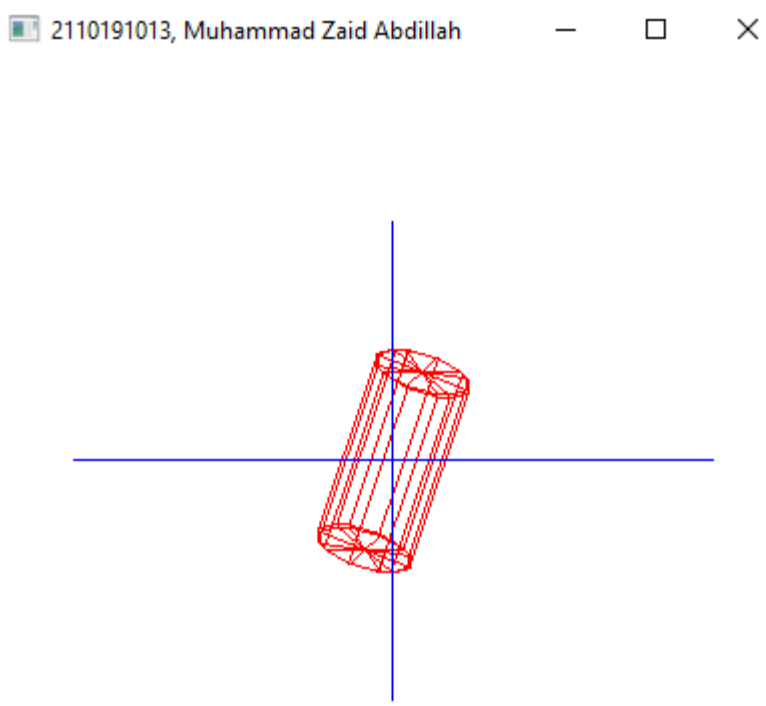
```
void drawLingkaran(float r, int x, int y, Color_t color) {
    set_color(color);
    Point2D_t lingkaran[360];
    for (int i = 0; i < 360; i++) {
        lingkaran[i].x = (float)(r * sin(i * 3.14 / 180) + x);
        lingkaran[i].y = (float)(r * cos(i * 3.14 / 180) + y);
    }
    draw_polyline(lingkaran, 360, color);
}
void drawObject3D(Object3D_t object, int n) {
    matrix3DX = rotationX(angle);
    matrix3DY = rotationY(angle);
    matrix3DZ = rotationZ(angle);
    int numFacePoint;
    Vector3D_t buffVector[3];
    Color_t color = { (0.0,0.0,1.0)
    };
    int result;
    std::cout << "\nBanyak titik : " << object.NumberOfVertices;
    std::cout << "\nBanyak sisi : " << object.NumberOfFaces;
    std::cout << "\n";
    for (int i = 0; i < object.NumberOfVertices; i++) {
        vector3D = Point2Vector3D(object.point[i]);
        vector3D = operator*(matrix3DZ, vector3D);
        vector3D = operator*(matrix3DY, vector3D);
        vector3D = operator*(matrix3DX, vector3D);
        object.point[i] = Vector2Point3D(vector3D);
    }
    angle += 1;
    Point3D_t point3D[4];
    Point2D_t point2D[4];
    Vector3D_t normalVector;
    Vector3D_t buffVector1[4];
    for (int i = 0; i < object.NumberOfFaces; i++) {
        for (int j = 0; j < object.face[i].NumberOfVertices; j++) {
            point3D[j] = object.point[object.face[i].point[j]];
            point2D[j].x = point3D[j].x;
            point2D[j].y = point3D[j].y;
            buffVector1[j] = Point2Vector3D(point3D[j]);
        }
        draw_polyline(point2D, object.face[i].NumberOfVertices,
            color);
    }
}
void selimut() {
```

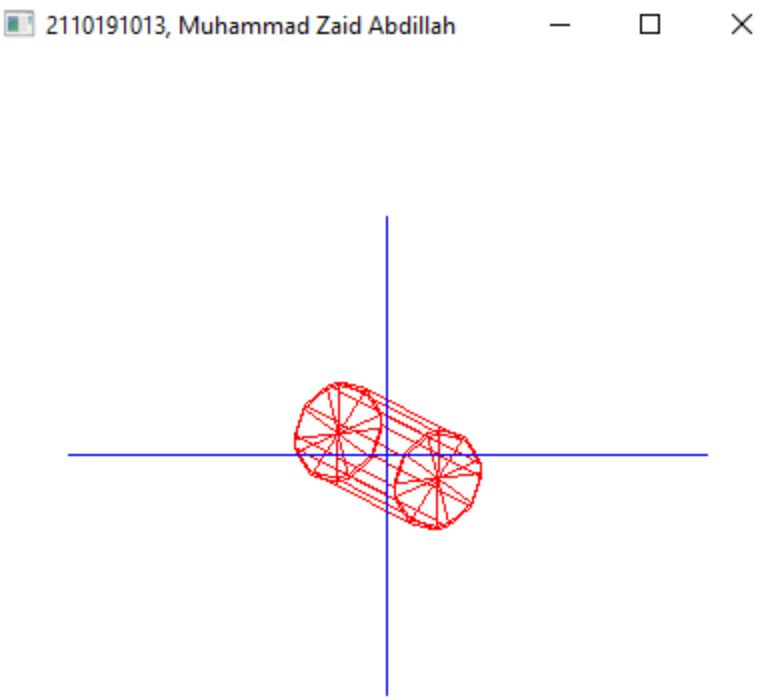
```
    Object3D_t selimut = {
    24, {
    {50 * 0,100,50 * 1},
    {50 * 0.6,100,50 * 0.8},
    {50 * 0.8,100,50 * 0.6},
    {50 * 1,100,50 * 0},
    {50 * 0.8,100,50 * (-0.6)},
    {50 * 0.6,100,50 * (-0.8)},
    {50 * 0,100,50 * (-1)},
    {50 * (-0.6),100,50 * (-0.8)},
    {50 * (-0.8),100,50 * (-0.6)},
    {50 * (-1),100,50 * 0},
    {50 * (-0.8),100,50 * 0.6},
    {50 * (-0.6),100,50 * 0.8},
    //bawah
    {50 * 0,-100,50 * 1},
    {50 * 0.6,-100,50 * 0.8},
    {50 * 0.8,-100,50 * 0.6},
    {50 * 1,-100,50 * 0},
    {50 * 0.8,-100,50 * (-0.6)},
    {50 * 0.6,-100,50 * (-0.8)},
    {50 * 0,-100,50 * (-1)},
    {50 * (-0.6),-100,50 * (-0.8)},
    {50 * (-0.8),-100,50 * (-0.6)},
    {50 * (-1),-100,50 * 0},
    {50 * (-0.8),-100,50 * 0.6},
    {50 * (-0.6),-100,50 * 0.8},
    },
    12, {
    {4,{0,1,13,12}},
    {4,{1,2,14,13}},
    {4,{2,3,15,14}},
    {4,{3,4,16,15}},
    {4,{4,5,17,16}},
    {4,{5,6,18,17}},
    {4,{6,7,19,18}},
    {4,{7,8,20,19}},
    {4,{8,9,21,20}},
    {4,{9,10,22,21}},
    {4,{10,11,23,22}},
    {4,{11,0,12,23}}
    }
    };
    drawObject3D(selimut, 12);
}
```

```
void lingkaran() {
    Object3D_t lingkaran = {
    26, {
    {50 * 0,100,50 * 1},
    {50 * 0.6,100,50 * 0.8},
    {50 * 0.8,100,50 * 0.6},
    {50 * 1,100,50 * 0},
    {50 * 0.8,100,50 * (-0.6)},
    {50 * 0.6,100,50 * (-0.8)},
    {50 * 0,100,50 * (-1)},
    {50 * (-0.6),100,50 * (-0.8)},
    {50 * (-0.8),100,50 * (-0.6)},
    {50 * (-1),100,50 * 0},
    {50 * (-0.8),100,50 * 0.6},
    {50 * (-0.6),100,50 * 0.8},
    //bawah
    {50 * 0,-100,50 * 1},
    {50 * 0.6,-100,50 * 0.8},
    {50 * 0.8,-100,50 * 0.6},
    {50 * 1,-100,50 * 0},
    {50 * 0.8,-100,50 * (-0.6)},
    {50 * 0.6,-100,50 * (-0.8)},
    {50 * 0,-100,50 * (-1)},
    {50 * (-0.6),-100,50 * (-0.8)},
    {50 * (-0.8),-100,50 * (-0.6)},
    {50 * (-1),-100,50 * 0},
    {50 * (-0.8),-100,50 * 0.6},
    {50 * (-0.6),-100,50 * 0.8},
    //tengah
    {0,100,0},
    {0,-100,0},
    },
    24, {
    {3,{0,1,24}},
    {3,{1,2,24}},
    {3,{2,3,24}},
    {3,{3,4,24}},
    {3,{4,5,24}},
    {3,{5,6,24}},
    {3,{6,7,24}},
    {3,{7,8,24}},
    {3,{8,9,24}},
    {3,{9,10,24}},
    {3,{10,11,24}},
    {3,{11,0,24}},
```

```
    {3,{12,13,25}},
    {3,{13,14,25}},
    {3,{14,15,25}},
    {3,{15,16,25}},
    {3,{16,17,25}},
    {3,{17,18,25}},
    {3,{18,19,25}},
    {3,{19,20,25}},
    {3,{20,21,25}},
    {3,{21,22,25}},
    {3,{22,23,25}},
    {3,{23,12,25}},
    }
    };
    drawObject3D(lingkaran, 12);
}
void Draw0() {
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(0.0, 0.0, 1.0);
    selimut();
    lingkaran();
    sumbu_koordinat();
    glFlush();
}
int main(int iArgc, char** cppArgv) {
    glutInit(&iArgc, cppArgv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(400, 400);
    glutInitWindowPosition(200, 200);
    glutCreateWindow("2110191013, Muhammad Zaid Abdillah");
    Initialize();
    glutDisplayFunc(Draw0);
    glutTimerFunc(1, timer, 0);
    glutMainLoop();
    return 0;
}
```

Output:


2110191013, Muhammad Zaid Abdillah


2110191013, Muhammad Zaid Abdillah