KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

ELECTRONIC ENGINEERING POLITECHNIC INSTITUTE OF SURABAYA (EEPIS)

JL. RAYA ITS KEPUTIH SUKOLILO SURABAYA 60111 INDONESIA

TELP. (031) 5947280 FAX. (031) 5946114

E-MAIL : pens@eepis-its.edu

http://www.pens.ac.id

GRAFIKA KOMPUTER

LAPORAN	:	PRAKTIKUM 05
JUDUL	:	TRANSFORMASI 3D
PERCOBAAN	:	-
NAMA	:	ROSYIDAH AMINI SUCI
KELAS	:	3 D3 TEKNIK INFORMATIKA B
NRP.	:	2103181045
DOSEN	:	HERO YUDO MARTONO
ASISTEN	:	-
TANGGAL	:	9 OKTOBER 2020

- **Source Code**

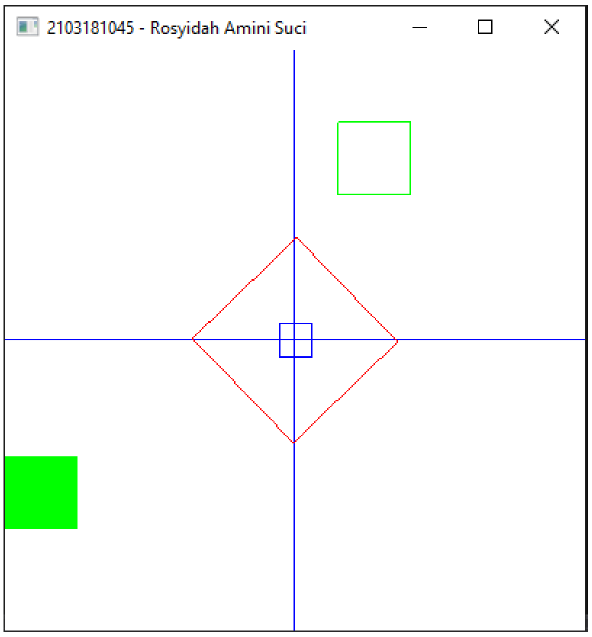
```
#include <GL/glut.h>
#include <iostream>
#include <math.h>
using namespace std;
// Stuct untuk menyimpan data titik, vektor dan matriks
typedef struct
{
	float x;
	float y;
} Point2D_t;
typedef struct
{
	int x;
	int y;
} Point2D_i;
typedef struct
{
	float x, y, z;
} Point3D_t;
typedef struct
{
	float x, y, z, r, g, b;
} Point3D_color_t;
typedef struct
{
	float v[2];
} Vector2D_t;
typedef struct
{
	float v[3];
} Vector3D_t;
typedef struct
{
	float m[2][2];
} Matrix2D_t;
typedef struct
{
	float m[3][3];
} Matrix3D_t;
typedef struct
{
	int m[3][3];
} Matrix3D_i;
typedef struct
{
	float r;
	float g;
	float b;
} Color_t;
// Function untuk mengubah point menjadi vektor dan sebaliknya
void setColor(Color_t col)
{
	glColor3f(col.r, col.g, col.b);
}
Vector3D_t point2DToVector3D(Point2D_t pnt)
{
	Vector3D_t vector;
	vector.v[0] = pnt.x;
```

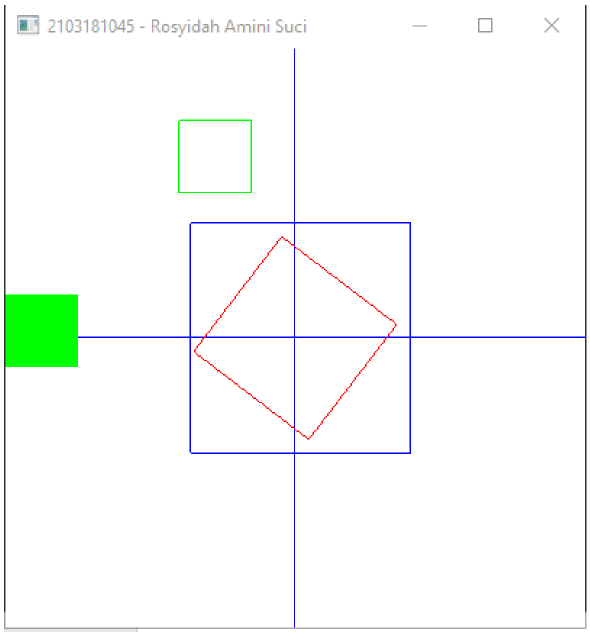
```c
	vector.v[1] = pnt.y;
	vector.v[2] = 1.0;
	return vector;
}
Vector3D_t point3DToVector3D(Point3D_t pnt)
{
	Vector3D_t vector;
	vector.v[0] = pnt.x;
	vector.v[1] = pnt.y;
	vector.v[2] = pnt.z;
	return vector;
}
Point2D_t point3DToPoint2D(Point3D_t pnt)
{
	Point2D_t point;
	point.x = pnt.x;
	point.y = pnt.y;
	return point;
}
Point2D_t vector2DToPoint2D(Vector2D_t vector)
{
	Point2D_t point;
	point.x = vector.v[0];
	point.y = vector.v[1];
	return point;
}
Point2D_t vector3DToPoint2D(Vector3D_t vector)
{
	Point2D_t point;
	point.x = vector.v[0];
	point.y = vector.v[1];
	return point;
}
Point3D_t vector3DToPoint3D(Vector3D_t vector)
{
	Point3D_t point;
	point.x = vector.v[0];
	point.y = vector.v[1];
	point.z = vector.v[2];
	return point;
}
// Function untuk membuat matiks identitas (diagonal utama 1)
Matrix3D_t createIdentityMatrix()
{
	Matrix3D_t matrix;
	matrix.m[0][0] = 1.0;
	matrix.m[0][1] = 0.0;
	matrix.m[0][2] = 0.0;
	matrix.m[1][0] = 0.0;
	matrix.m[1][1] = 1.0;
	matrix.m[1][2] = 0.0;
	matrix.m[2][0] = 0.0;
	matrix.m[2][1] = 0.0;
	matrix.m[2][2] = 1.0;
	return matrix;
}
Matrix3D_t rotationZ(float theta)
{
	Matrix3D_t matrix = createIdentityMatrix();
	matrix.m[0][0] = cos(theta / 57.3);
	matrix.m[0][1] = -sin(theta / 57.3);
	matrix.m[1][0] = sin(theta / 57.3);
	matrix.m[1][1] = cos(theta / 57.3);
	return matrix;
}
Matrix3D_t flipZ(float theta)
{
	Matrix3D_t matrix = createIdentityMatrix();
	matrix.m[0][0] = cos(theta / 57.3);
	matrix.m[0][1] = 0.0;
	matrix.m[0][2] = sin(theta / 57.3);
	matrix.m[1][0] = 0.0;
	matrix.m[1][1] = -1.0;
	matrix.m[1][2] = 0.0;
	matrix.m[2][0] = -sin(theta / 57.3);
	matrix.m[2][1] = 0.0;
	matrix.m[2][2] = cos(theta / 57.3);
	return matrix;
}
Matrix3D_t scaleZ(float m)
{
	Matrix3D_t matrix = createIdentityMatrix();
	matrix.m[0][0] = m;
	matrix.m[1][1] = m;
	return matrix;
}
Vector3D_t operator*(Matrix3D_t matrix, Vector3D_t vector)
{
	Vector3D_t vectorHasil;
	for (int i = 0; i < 3; i++)
	{
		vectorHasil.v[i] = 0;
		for (int j = 0; j < 3; j++)
		{
			vectorHasil.v[i] += matrix.m[i][j] * vector.v[j];
		}
	}
	return vectorHasil;
}
// Pre Draw Function
void drawLine(Point2D_t pnt[], int n, Color_t color)
{
	int i;
	setColor(color);
	glBegin(GL_LINES);
	for (i = 0; i < n; i++)
	{
		glVertex2f(pnt[i].x, pnt[i].y);
	}
	glEnd();
}
void drawPolygon(Point3D_t pnt[], int n, Color_t color)
{
	int i;
	setColor(color);
	glBegin(GL_POLYGON);
	for (i = 0; i < n; i++)
	{
		glVertex2f(pnt[i].x, pnt[i].y);
	}
	glEnd();
}
void drawPolyline(Point3D_t pnt[], int n, Color_t color)
{
	int i;
	setColor(color);
	glBegin(GL_LINE_LOOP);
	for (i = 0; i < n; i++)
	{
		glVertex2f(pnt[i].x, pnt[i].y);
	}
	glEnd();
}
void drawPoint(int x, int y)
{
	glColor3f(0.0, 0.0, 1.0);
	glPointSize(5);
	glBegin(GL_POINTS);
	glVertex2i(x, y);
	glEnd();
}
// Variabel global objek
Point3D_t kotakRotasi[4] = {
{-50.0, -50.0, 0.0}, {-50.0, 50.0, 0.0}, {50.0, 50.0, 0.0}, {50.0, -50.0,
0.0} };
Point3D_t kotakVertikal[4] = {
{-200.0, -200.0, 0.0}, {-200.0, -150.0, 0.0}, {-150.0, -150.0, 0.0}, {-
150.0, -200.0, 0.0} };
Point3D_t kotakHorizontal[4] = {
{100.0, 100.0, 0.0}, {100.0, 150.0, 0.0}, {150.0, 150.0, 0.0}, {150.0,
100.0, 0.0} };
Point3D_t kotakScale[4] = {
{-5.0, -5.0, 0.0}, {-5.0, 5.0, 0.0}, {5.0, 5.0, 0.0}, {5.0, -5.0, 0.0} };
// Function untuk menggambar sumbu koordinat
void drawSumbuKoordinat()
{
	Point2D_t sumbuX[2] = { {-200.0, 0.0}, {200.0, 0.0} };
	Point2D_t sumbuY[2] = { {0.0, -200.0}, {0.0, 200.0} };
	Color_t col = { 0.0, 0.0, 1.0 };
	drawLine(sumbuX, 2, col);
	drawLine(sumbuY, 2, col);
}
void drawSegiempatRotasi()
{
	int n = 4;
	Color_t col = { 1.0, 0.0, 0.0 };
	Vector3D_t vec3D;
	Matrix3D_t matrix3DZ = rotationZ(-2);
	drawPolyline(kotakRotasi, n, col);
	for (int i = 0; i < n; i++)
	{
		vec3D = point3DToVector3D(kotakRotasi[i]);
		vec3D = operator*(matrix3DZ, vec3D);
		kotakRotasi[i] = vector3DToPoint3D(vec3D);
	}
}
void drawSegiempatVertical()
{
	int n = 4;
	Color_t col = { 0.0, 1.0, 0.0 };
	drawPolygon(kotakVertikal, n, col);
	for (int i = 0; i < n; i++)
	{
		if (kotakVertikal[0].y >= 250)
		{
			kotakVertikal[0].y = -240;
```

```
				kotakVertikal[1].y = -190;
				kotakVertikal[2].y = -200;
				kotakVertikal[3].y = -250;
			}
			else
			{
				kotakVertikal[i].y = kotakVertikal[i].y + 10;
			}
		}
}
void drawSegiempatHorizontal()
{
		int n = 4;
		Color_t col = { 0.0, 1.0, 0.0 };
		drawPolyline(kotakHorizontal, n, col);
		for (int i = 0; i < n; i++)
		{
			if (kotakHorizontal[0].x <= -250)
			{
				kotakHorizontal[0].x = 240;
				kotakHorizontal[1].x = 240;
				kotakHorizontal[2].x = 300;
				kotakHorizontal[3].x = 300;
			}
			else
			{
				kotakHorizontal[i].x = kotakHorizontal[i].x - 10;
			}
		}
}
void drawSegiempatScale()
{
		int n = 4;
		Color_t col = { 0.0, 0.0, 1.0 };
		Vector3D_t vec3D;
		Matrix3D_t matrix3DZ = scaleZ(1.1);
		drawPolyline(kotakScale, n, col);
		for (int i = 0; i < n; i++)
		{
			if (kotakScale[0].x <= -200)
			{
				kotakScale[0].x = -5.0;
				kotakScale[0].y = -5.0 * 1.1;
				kotakScale[1].x = -5.0;
				kotakScale[1].y = 5.0 * 1.1;
				kotakScale[2].x = 5.0;
				kotakScale[2].y = 5.0;
				kotakScale[3].x = 5.0;
				kotakScale[3].y = -5.0;
			}
			else
			{
				vec3D = point3DToVector3D(kotakScale[i]);
				vec3D = operator*(matrix3DZ, vec3D);
				kotakScale[i] = vector3DToPoint3D(vec3D);
			}
		}
}
void userdraw(void)
{
		// Disini tempat untuk menggambar
		drawSumbuKoordinat();
```

```
        drawSegiempatRotasi();
        drawSegiempatScale();
        drawSegiempatVertical();
        drawSegiempatHorizontal();
}
void display(void)
{
        glClear(GL_COLOR_BUFFER_BIT);
        userdraw();
        glFlush();
}
// Pengaturan
void Initialize()
{
        glClearColor(1.0, 1.0, 1.0, 0.0);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(-200.0, 200.0, -200.0, 200.0);
}
void timer(int)
{
        glutPostRedisplay();
        glutTimerFunc(100, timer, 0);
}
int main(int argc, char** argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowPosition(200, 200);
        glutInitWindowSize(400, 400);
        glutCreateWindow("2103181045 - Rosyidah Amini Suci");
        Initialize();
        glutDisplayFunc(display);
        glutTimerFunc(100, timer, 0);
        glutMainLoop();
        return 0;
```

- **Output**


2103181045 - Rosyidah Amini Suci


2103181045 - Rosyidah Amini Suci