

**Laporan Tugas Besar 2 IF-2123 Aljabar Geometri
Simulasi Transformasi Linier pada Bidang 2D dan 3D
Dengan Menggunakan *OpenGL API*
Semester I Tahun 2018/2019**

Disusun oleh:

Winston Wijaya - 13517018

Irfan Sofyana Putra – 13517078

Jofiandy Leonata Pratama – 13517135



**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2018**

DAFTAR ISI

BAB I : DESKRIPSI MASALAH	2
BAB II : <u>TEORI</u> DASAR	4
BAB III : <u>IMPLEMENTASI</u> PROGRAM DALAM PYTHON DAN <u>PEMBAGIAN</u> TUGAS ANTAR KELOMPOK	7
BAB IV : <u>EKSPERIMEN</u>	25
BAB V : <u>PENUTUP</u>	34
DAFTAR REFERENSI	36

BAB I

DESKRIPSI MASALAH

Pada tugas kali ini, mahasiswa diminta **membuat program** yang mensimulasikan transformasi linier untuk melakukan operasi translasi, refleksi, dilatasi, rotasi, dan sebagainya pada sebuah objek **2D dan 3D**. Objek dibuat dengan mendefinisikan sekumpulan titik sudut lalu membuat bidang 2D/3D dari titik-titik tersebut. Contoh objek 2D: segitiga, segiempat, polygon segi-n, lingkaran, rumah, gedung, mobil, komputer, lemari, dsb. Contoh objek 3D: kubus, pyramid, silinder, terompet, dll.

Program akan memiliki dua buah window, window pertama (*command prompt*) berfungsi untuk menerima *input* dari *user*, sedangkan *window* kedua (*GUI*) berfungsi untuk menampilkan output berdasarkan input dari user. Kedua window ini muncul ketika user membuka file *executable*.

Untuk objek 2D, saat program baru mulai dijalankan, program akan menerima input **N**, yaitu jumlah titik yang akan diterima. Berikutnya, program akan menerima input **N** buah **titik** tersebut (pasangan nilai **x dan y**). Setelah itu program akan menampilkan output sebuah bidang yang dibangkitkan dari titik-titik tersebut. Selain itu juga ditampilkan dua buah garis, yaitu **sumbu x** dan **sumbu y**. Nilai x dan y memiliki rentang minimal **-500 pixel** dan maksimum **500 pixel**. Pastikan window *GUI* yang Anda buat memiliki ukuran yang cukup untuk menampilkan kedua sumbu dari ujung ke ujung. Hal yang sama juga berlaku untuk objek 3D tetapi dengan tiga sumbu: x, y, dan z.

Berikutnya, program dapat menerima input yang didefinisikan pada tabel dibawah.

Input	Keterangan
<code>translate <dx> <dy></code>	Melakukan translasi objek dengan menggeser nilai x sebesar dx dan menggeser nilai y sebesar dy .
<code>dilate <k></code>	Melakukan dilatasi objek dengan faktor scaling k .

<code>rotate <deg> <a> </code>	Melakukan rotasi objek secara berlawanan arah jarum jam sebesar <i>deg</i> derajat terhadap titik <i>a,b</i>
<code>reflect <param></code>	Melakukan pencerminan objek. Nilai <i>param</i> adalah salah satu dari nilai - nilai berikut: <i>x, y, y=x, y=-x</i> , atau <i>(a,b)</i> . Nilai (a,b) adalah titik untuk melakukan pencerminan terhadap.
<code>shear <param> <k></code>	Melakukan operasi <i>shear</i> pada objek. Nilai <i>param</i> dapat berupa <i>x</i> (terhadap sumbu x) atau <i>y</i> (terhadap sumbu y). Nilai <i>k</i> adalah faktor <i>shear</i> .
<code>stretch <param> <k></code>	Melakukan operasi <i>stretch</i> pada objek. Nilai <i>param</i> dapat berupa <i>x</i> (terhadap sumbu x) atau <i>y</i> (terhadap sumbu y). Nilai <i>k</i> adalah faktor <i>stretch</i> .
<code>custom <a> <c> <d></code>	Melakukan transformasi linier pada objek dengan matriks transformasi sebagai berikut:
<code>multiple <n> ... // input 1 ... // input 2 // input n</code>	Melakukan transformasi linier pada objek sebanyak <i>n</i> kali berurutan. Setiap baris input 1.. <i>n</i> dapat berupa <i>translate, rotate, shear, dll</i> tetapi bukan <i>multiple, reset, exit</i> .
<code>reset</code>	Mengembalikan objek pada kondisi awal objek didefinisikan.
<code>exit</code>	Keluar dari program.

BAB II

TEORI DASAR

2.1 Transformasi Linier

Secara singkat, transformasi linear dapat diartikan sebagai fungsi peubah suatu vektor, misalkan V menjadi vektor W jika kedua aturan dibawah ini memenuhi kedua properti u dan v di vektor V untuk seluruh skalar k :

- (i) $T(ku) = kT(u)$ (Homogeneity property)
- (ii) $T(u+v) = T(u) + T(v)$ (Additivity property)

Transformasi Linier dapat dibedakan menjadi beberapa fungsi yaitu:

1. Translasi
2. Dilatasi
3. Refleksi
4. Pergeseran (Shear)
5. Rotasi
6. Peregangan (Stretch)

I. Translasi

Translasi merupakan metode transformasi linier yang melakukan perubahan posisi dari suatu titik maupun bidang dari satu posisi ke posisi lainnya dengan jarak tertentu. Operasi dari Translasi ini digunakan dengan menambahkan atau mengurangi suatu konstanta terhadap titik - titik dari suatu koordinat.

Contoh : Translasi $(100,100)$ terhadap titik $(200,200)$ menghasilkan $(300,300)$

II. Dilatasi

Dilatasi merupakan metode transformasi linier yang mengubah suatu nilai posisi dari titik dengan melakukan ekspansi ataupun kompresi dengan suatu faktor pada sumbu tertentu. Hal ini menyebabkan tidak hanya ukuran yang berubah namun juga posisi dari benda tersebut dapat berubah.

Operasi dari Dilatasi ini menggunakan suatu faktor ekspansi yang mengubah nilai dari koordinat suatu titik.

Contoh : Dilatasi Titik $(100,100)$, $(100,150)$, $(150,100)$ terhadap faktor 2 menghasilkan Titik $(200, 200)$, $(200, 300)$, $(300, 200)$

III. Refleksi

Refleksi merupakan metode transformasi linier yang menggunakan konsep pencerminan sehingga dihasilkan titik yang merupakan 'bayangan' dari titik yang ingin di-operasikan.

Operasi Refleksi yang digunakan pada kasus ini merupakan operasi refleksi terhadap suatu titik.

Contoh : Refleksi Titik (100,100) terhadap titik (0,0) (Origin) adalah (-100, -100)

IV. Pergeseran (Shear)

Pergeseran (Shear) merupakan metode transformasi linier yang menggerakkan suatu sumbu koordinat dengan koefisien k terhadap sumbu koordinat lainnya. Hal ini menyebabkan apabila nilai suatu sumbu koordinat besar, menyebabkan shear dari koordinat lainnya menjadi signifikan

Operasi shear yang digunakan pada kasus ini menerima 2 input yaitu sumbu dimana operasi diterapkan dan faktor pergeserannya

Contoh : Shear Titik (100,200) dengan sumbu x dengan koefisien 2 menghasilkan nilai (500,200)

V. Rotasi

Rotasi merupakan metode transformasi linier yang memutar titik dari suatu objek dengan suatu sumbu dan besaran derajat perputarannya.

Operasi Rotasi yang digunakan pada kasus ini adalah perputaran suatu titik dengan derajat yang diberikan terhadap suatu sumbu yang berupa titik

Contoh : Rotasi Titik (100, 0) dengan besar perputaran 90° akan menghasilkan titik berupa (0,100)

VI. Peregangannya (Stretch)

Peregangan merupakan metode transformasi linier yang mengubah nilai dari suatu titik terhadap suatu sumbu dengan mengkalikannya dengan faktor k . Peregangan ini hampir mirip dengan ekspansi namun hanya dilakukan pada suatu sumbu saja

Operasi Stretch ini menerima input dari sumbu yang ingin diregangkan dan besar faktor peregangannya

Contoh : Peregangan Titik (100,100), (150,200), (100,150) diregangkan terhadap faktor 2 pada sumbu x menghasilkan nilai titik (200,100), (300,200), (200,150).

2.2 Matriks Transformasi

Matriks transformasi merupakan matriks yang digunakan untuk melakukan transformasi terhadap suatu titik. Tujuan adanya matriks transformasi ini agar mempermudah melakukan operasi transformasi tersebut terhadap suatu titik.

2.3 OpenGL

OpenGL (Open Graphics Library) adalah spesifikasi standar yang mendefinisikan sebuah lintas-bahasa, lintas platform API untuk mengembangkan aplikasi yang menghasilkan grafis komputer 2D maupun 3D. Antarmuka terdiri dari lebih dari 250 panggilan fungsi yang

berbeda yang dapat digunakan untuk menggambar tiga dimensi yang adegan-adegan kompleks dari bentuk-bentuk primitif sederhana. OpenGL dikembangkan oleh Silicon Graphics Inc (SGI) pada tahun 1992 [2] dan secara luas digunakan dalam CAD, realitas maya, visualisasi ilmiah, visualisasi informasi, dan simulasi penerbangan. Hal ini juga digunakan dalam video game, di mana bersaing dengan Direct3D on Microsoft Windows platform (lihat vs OpenGL Direct3D). OpenGL dikelola oleh sebuah teknologi konsorsium nirlaba yaitu Khronos Group.

Fungsi dasar dari OpenGL adalah untuk mengeluarkan koleksi perintah khusus atau executable ke sistem operasi. Dengan demikian, program ini bekerja dengan perangkat keras grafis yang ada yang berada pada hard drive atau sumber tertentu lainnya. Setiap perintah dalam dirancang untuk melakukan tindakan tertentu, atau memulai efek khusus tertentu yang terkait dengan grafis.

OpenGL adalah suatu spesifikasi grafik yang low-level yang menyediakan fungsi untuk pembuatan grafik primitive termasuk titik, garis, dan lingkaran. OpenGL digunakan untuk keperluan-keperluan pemrograman grafis. OpenGL bersifat Open-Source, multi-platform dan multi-language serta digunakan mendefinisikan suatu objek, baik objek 2 dimensi maupun objek 3 dimensi. OpenGL juga merupakan suatu antarmuka pemrograman aplikasi (application programming interface (API) yang tidak tergantung pada piranti dan platform yang digunakan, sehingga OpenGL dapat berjalan pada sistem operasi Windows, UNIX dan sistem operasi lainnya.

OpenGL pada awalnya didesain untuk digunakan pada bahasa pemrograman C/C++, namun dalam perkembangannya OpenGL dapat juga digunakan dalam bahasa pemrograman yang lain seperti Java, Tcl, Ada, Visual Basic, Delphi, maupun Fortran. Namun OpenGL di-package secara berbeda-beda sesuai dengan bahasa pemrograman yang digunakan. Oleh karena itu, package OpenGL tersebut dapat di-download pada situs <http://www.opengl.org> sesuai dengan bahasa pemrograman yang akan digunakan.

OpenGL melayani dua tujuan :

- Untuk menyembunyikan kompleksitas dari interfacing dengan berbagai 3D accelerators, memamerkan oleh programmer dengan satu, seragam API.
- Untuk menyembunyikan kemampuan yang berbeda dari hardware platform, oleh semua yang memerlukan mendukung implementasi penuh fitur OpenGL set (menggunakan software emulation jika diperlukan).

BAB III

IMPLEMENTASI PROGRAM DALAM PYTHON DAN PEMBAGIAN TUGAS ANTAR KELOMPOK

3.1 Implementasi Program

3.1.1 Main.py

```
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
from transformation2D import *
from transformation3D import *
from random import *
import sys
import math
import threading

#declaration of neededs variables
Wireframe = True
width = 700
height = 700
tra_x = 0
tra_y = 0
tra_z = 0
x_rot = 0
y_rot = 0
angle = 0
lx = 0
lz = -1
xx = 0
zz = 5
yy = 0
ly = 1
```



```

#define class point. Point has 3 components (x,y,z)

class Point:
    def __init__(self, x, y, z):
        self.x = x
        self.y = y
        self.z = z
        self.arrPoint = []

    def add_point(self, x, y, z):
        temp = Point(float(x), float(y), float(z))
        self.arrPoint.append(temp)

#initialize variables for points
initial = Point(0,0,0)
titik = Point(0,0,0)

2D was selected (calling all transformation and method for 2D)
def run2d():
    print("Masukan banyak titik: ")
    n = int(input())
    print("Masukan titik-titik tersebut:")
    for i in range(0, n):
        x,y = input().split()
        titik.add_point(x,y,0)
        initial.add_point(x,y,0)
    makeWindow2D()

#Draw Shape of the Blocks (So every changes be made, call this method)
def drawShape2D(x):
    glBegin(GL_POLYGON)
    for i in x.arrPoint:
        glColor3f(random(), random(), random())
        glVertex2f(i.x, i.y)
    glEnd()

#Draw Axis
def drawAxis():
    glBegin(GL_LINES)

    glColor3f(random(), random(), random())
    glVertex2f(0,width)
    glVertex2f(0, -width)

    glColor3f(random(), random(), random())
    glVertex2f(height,0)
    glVertex2f(-height,0)
    glEnd()

```

```

#Draw all the parts for the window
def draw2D():
    while (True):
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT) # clear
the screen
        glLoadIdentity()
        refresh2d(width, height)
        drawAxis()
        drawShape2D(titik)
        glutSwapBuffers()
        comm = input()
        if (comm == "exit"):
            sys.exit()
        elif (comm != "multiple"):
            doCommand2D(comm)
        elif (comm == "multiple"):
            tmp = input()
            n = int(tmp)
            for i in range(0, n):
                comm = input()
                doCommand2D(comm)

```

```

#do the command for 2d only
def doCommand2D(comm):
    if (comm == "translate"):
        x, y = input().split()
        P = translate(titik, float(x), float(y))
    elif (comm == "dilate"):
        x = input()
        P = dilate(titik, float(x))
    elif (comm == "reflect"):
        tmp = input()
        if (len(tmp) == 1):
            if (tmp == "x"):
                #reflect to x axis
                P = reflect(titik, 0, 0, 0)
            else:
                #reflect to y axis
                P = reflect(titik, 1, 0, 0)
        else:
            if (tmp[0]=='y'):
                if (len(tmp) == 3):
                    #reflect to y=x
                    P = reflect(titik, 2, 0, 0)
                else:
                    #reflect to y=-x
                    P = reflect(titik, 3, 0, 0)
            else:
                #reflect to point(a, b)

```

```

elif (comm == "shear"):
    a, k = input().split()
    P = shear(titik, a, float(k))
elif (comm == "rotate"):
    deg, a, b = input().split()
    P = rotate(titik, float(deg), float(a), float(b))
elif (comm == "stretch"):
    param, x = input().split()
    P = stretch(titik, param, float(x))
elif (comm == "custom"):
    a,b,c,d = input().split()
    P = custom(titik, float(a), float(b), float(c), float(d))
elif (comm == "reset"):
    P = reset(titik, initial)

```

```

#Make the window of the program
def makeWindow2D():
    glutInit() # initialize glut
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_ALPHA | GLUT_DEPTH)
    glutInitWindowSize(width, height) # set window size
    glutInitWindowPosition(0, 0) # set window
    position
    window = glutCreateWindow("Tubes Algeo 2D") # create window
    with title
    glutDisplayFunc(draw2D) # set draw
    function callback
    glutIdleFunc(draw2D) # draw all the
    time
    glutMainLoop()

```

```

#Buffer the drawing for 2D
def refresh2d(width, height):
    glViewport(0,0 , width, height)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glOrtho(-width, width, -height, height, 0.0, 1.0)
    glMatrixMode (GL_MODELVIEW)
    glLoadIdentity()

#3D was selected (calling all transformation and method for 3D)
def run3d():
    tmpArr = [(-0.5, -0.5, 0.5),
              (0.5, -0.5, 0.5), (0.5, 0.5, 0.5), (-0.5, 0.5, 0.5),
              (-0.5, -0.5, -0.5), (-0.5, 0.5, -0.5), ( 0.5, 0.5, -0.5), ( 0.5, -0.5,
0.5), (-0.5, -0.5, 0.5)
              , (-0.5, 0.5, 0.5), (-0.5, 0.5, -0.5), (-0.5, -0.5, -0.5), ( 0.5, -0.5, -
0.5), ( 0.5, 0.5, -0.5),
              ( 0.5, 0.5, 0.5), ( 0.5, -0.5, 0.5), (-0.5, 0.5, 0.5), ( 0.5, 0.5, 0.5),
              ( 0.5, 0.5, -0.5),
              (-0.5, 0.5, -0.5), (-0.5, -0.5, 0.5), (-0.5, -0.5, -0.5), ( 0.5, -0.5, -
0.5), ( 0.5, -0.5, 0.5)]

    for i in tmpArr:
        titik.add_point(i[0], i[1], i[2])
        initial.add_point(i[0], i[1], i[2])
    makeWindow3D()
    return

```

```

#Make the window of the program 3D
def makeWindow3D():
    global window
    glutInit()
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_ALPHA | GLUT_DEPTH)
    glutInitWindowSize(width, height)
    glutInitWindowPosition(0, 0)
    window = glutCreateWindow("Tubes Algeo 3D")
    glutDisplayFunc(draw3D)
    glutIdleFunc(draw3D)
    glutReshapeFunc(ReSizeGLScene)
    InitGL(width, height)
    glutMainLoop()

#draw the axis for 3D
def drawAxis3D():
    glBegin(GL_LINES)

    glColor3f(1,0,0)
    glVertex3f(0,500,0)
    glVertex3f(0,-500,0)

    glColor3f(0,0,1)
    glVertex3f(500,0,0)
    glVertex3f(-500,0,0)

    glColor3f(0,1,0)
    glVertex3f(0,0,500)
    glVertex3f(0,0,-500)
    glEnd()

```

```

#initialize openGl window

def InitGL(Width, Height):
    glClearColor(0.0, 0.0, 0.0, 0.0)
    glClearDepth(1.0)
    glDepthFunc(GL_LESS)
    glEnable(GL_DEPTH_TEST)
    glPolygonMode(GL_FRONT, GL_LINE)
    glPolygonMode(GL_BACK, GL_LINE)
    glShadeModel(GL_SMOOTH)

    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()

    gluPerspective(45.0, float(Width)/float(Height), 0.1, 100.0)

    glMatrixMode(GL_MODELVIEW)

```

```
#resize openGl
def ReSizeGLScene(Width, Height):
    if Height == 0:
        Height = 1
    glViewport(0, 0, Width, Height)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    gluPerspective(45.0, float(Width)/float(Height), 0.1, 100.0)
    glMatrixMode(GL_MODELVIEW)
```

```
#Draw all the parts for the window 2D
def draw3D():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT) # clear the screen
    glMatrixMode(GL_MODELVIEW)

    glLoadIdentity()

    gluLookAt(
        xx, 1, zz,
        xx+lx, yy+ly, zz+lz,
        0, 1.0, 0)

    drawAxis3D()
    glutKeyboardFunc(keyPressed)
    glTranslatef(0.0, 0.0, 0.0)

    glRotatef(x_rot, 1.0, 0.0, 0.0)
    glRotatef(y_rot, 0.0, 1.0, 0.0)
    # glTranslatef(tra_x, tra_y, tra_z)

    drawShape3D(titik)
    glFlush()
    glutSwapBuffers()
```

```

#do all commands for 3D program
def doCommand3D():
    comm = input()
    if (comm == "translate"):
        x, y, z = input().split()
        P = translate3D(titik, float(x), float(y), float(z))
    elif (comm == "dilate"):
        n = input()
        P = dilate3D(titik, float(n))
    elif (comm == "reflect"):
        x, y, z = input().split()
        P = reflect3D(titik, float(x), float(y), float(z))
    elif (comm == "shear"):
        a, b, c = input().split()
        P = shear3D(titik, a, float(b), float(c))
    elif (comm == "rotate"):
        a, deg = input().split()
        P = rotate3D(titik, a, float(deg))
    elif (comm == "stretch"):
        param, x = input().split()
        P = stretch3D(titik, param, float(x))
    elif (comm == "custom"):
        a, b, c = input().split()
        d, e, f = input().split()
        g, h, i = input().split()
        P = custom3D(titik, float(a), float(b), float(c), float(d),
float(e), float(f), float(g), float(h), float(i))
    elif (comm == "reset"):
        P = reset3D(titik, initial)
    elif (comm == "multiple"):
        n = input()
        for i in range(0, int(n)):
            doCommand3D()
    elif (comm == "exit"):
        sys.exit()

```

```

#keypressed procedure for camera's looking in 3D
def keyPressed(*args):
    global tra_x
    global tra_y
    global tra_z
    global x_rot
    global y_rot
    global lx, lz, xx, zz, ly, yy
    global angle
    fraction = 0.1
    if (args[0] == b"w"):
        xx += lx * fraction
        zz += lz * fraction
    elif (args[0] == b"s"):
        xx -= lx * fraction
        zz -= lz * fraction
    elif (args[0] == b"a"):
        angle -= 0.1
        lx = math.sin(angle)
        lz = -math.cos(angle)
    elif (args[0] == b"d"):
        angle += 0.1
        lx = math.sin(angle)
        lz = -math.cos(angle)
    elif (args[0] == b"z"):
        yy += ly * fraction
        # xx += lx * fraction
    elif (args[0] == b"c"):
        yy -= ly * fraction
        # xx -= lx * fraction
    elif (args[0] == b"u"):
        x_rot += 2.0
        y_rot += 2.0
    elif (args[0] == b"y"):
        x_rot -= 2.0
        y_rot -= 2.0
    elif (args[0] == b"x"):
        global Wireframe
        if Wireframe==False:
            glPolygonMode(GL_FRONT, GL_LINE)
            glPolygonMode(GL_BACK, GL_LINE)
            Wireframe=True
        elif Wireframe ==True:
            glPolygonMode(GL_FRONT, GL_FILL)
            glPolygonMode(GL_BACK, GL_FILL)
            Wireframe=False
    elif (args[0] == b"f"):
        doCommand3D()

```



```

#procedure for drawing 3D object
def drawShape3D(x):
    glBegin(GL_QUADS)
    glColor3f(1.0, 0.0, 0.0)
    #front
    glVertex3f(x.arrPoint[0].x, x.arrPoint[0].y, x.arrPoint[0].z)
    glColor3f(0.0, 1.0, 0.0)
    glVertex3f(x.arrPoint[1].x, x.arrPoint[1].y, x.arrPoint[1].z)
    glVertex3f(x.arrPoint[2].x, x.arrPoint[2].y, x.arrPoint[2].z)
    glColor3f(1.0, 0.0, 0.0)
    glVertex3f(x.arrPoint[3].x, x.arrPoint[3].y, x.arrPoint[3].z)
    #back
    glVertex3f(x.arrPoint[4].x, x.arrPoint[4].y, x.arrPoint[4].z)
    glVertex3f(x.arrPoint[5].x, x.arrPoint[5].y, x.arrPoint[5].z)
    glColor3f(0.0, 1.0, 0.0)
    glVertex3f(x.arrPoint[6].x, x.arrPoint[6].y, x.arrPoint[6].z)
    glVertex3f(x.arrPoint[7].x, x.arrPoint[7].y, x.arrPoint[7].z)
    glColor3f(0.0, 1.0, 0.0)
    #left
    glVertex3f(x.arrPoint[8].x, x.arrPoint[8].y, x.arrPoint[8].z)
    glVertex3f(x.arrPoint[9].x, x.arrPoint[9].y, x.arrPoint[9].z)
    glColor3f(0.0, 0.0, 1.0)
    glVertex3f(x.arrPoint[10].x, x.arrPoint[10].y, x.arrPoint[10].z)
    glColor3f(1.0, 0.0, 0.0)
    glVertex3f(x.arrPoint[11].x, x.arrPoint[11].y, x.arrPoint[11].z)
    #right
    glVertex3f(x.arrPoint[12].x, x.arrPoint[12].y, x.arrPoint[12].z)
    glVertex3f(x.arrPoint[13].x, x.arrPoint[13].y, x.arrPoint[13].z)
    glColor3f(0.0, 1.0, 0.0)
    glVertex3f(x.arrPoint[14].x, x.arrPoint[14].y, x.arrPoint[14].z)
    glColor3f(0.0, 0.0, 1.0)
    glVertex3f(x.arrPoint[15].x, x.arrPoint[15].y, x.arrPoint[15].z)
    glColor3f(0.0, 0.0, 1.0)
    #top
    glVertex3f(x.arrPoint[16].x, x.arrPoint[16].y, x.arrPoint[16].z)
    glVertex3f(x.arrPoint[17].x, x.arrPoint[17].y, x.arrPoint[17].z)
    glColor3f(0.0, 1.0, 0.0)
    glVertex3f(x.arrPoint[18].x, x.arrPoint[18].y, x.arrPoint[18].z)
    glVertex3f(x.arrPoint[19].x, x.arrPoint[19].y, x.arrPoint[19].z)
    glColor3f(1.0, 0.0, 0.0)

    #bottom
    glVertex3f(x.arrPoint[20].x, x.arrPoint[20].y, x.arrPoint[20].z)
    glColor3f(0.0, 0.0, 1.0)
    glVertex3f(x.arrPoint[21].x, x.arrPoint[21].y, x.arrPoint[21].z)
    glVertex3f(x.arrPoint[22].x, x.arrPoint[22].y, x.arrPoint[22].z)
    glVertex3f(x.arrPoint[23].x, x.arrPoint[23].y, x.arrPoint[23].z)

    glEnd()

```

```

#list of command
def listofcommand():
    print("Untuk list of command 2D, pilih 2")
    print("Untuk list of command 3D, pilih 3")
    print("Masukkan pilihan: ", end = " ")
    n = int(input())
    if (n == 2):
        print("Untuk 2D:")
        print("Pisahkan kata kunci perintah dengan newline")
        print("contoh perintah: ")
        print("translate")
        print("1 2")
        print("")
        print("----List Perintah")
        print("1. translate dx dy")
        print("    Translasi objek dengan menggeser nilai x sebesar dx dan
menggeser y sebesar dy")
        print("2. dilate k")
        print("    melakukan dilatasi untuk objek dengan faktor scalling k")
        print("3. rotate deg a b")
        print("    melakukan rotasi objek berlawanan arah jarum jam sebesar
deg derajat terhadap titik (a,b)")
        print("4. reflect x y")
        print("    melakukan pencerminan terhadap titik (x,y)")
        print("5. shear x k")
        print("    melakukan operasi shear pada objek terhadap sumbu x
dengan faktor scalling k")
        print("6. stretch x k")
        print("    melakukan operasi stretch pada objek terhadap sumbu x
dengan faktor scalling k")
        print("7. custom a b c d")
        print("    melakukan transformasi linier dengan matriks transformasi
[(a,b),(c,d)]")
        print("8. multiple n")
        print("    melakukan transformasi linier pada objek sebanyak n
kali")

```

```

else:
    print("Untuk 3D:")
    print("Pisahkan kata kunci perintah dengan newline")
    print("contoh perintah: ")
    print("translate")
    print("1 2 3")
    print("")
    print("----List Perintah")
    print("1. translate dx dy dz")
    print("    Translasi objek dengan menggeser nilai x sebesar dx dan
menggeser y sebesar dy dan menggeser z sebesar dz")
    print("2. dilate k")
    print("    melakukan dilatasi untuk objek dengan faktor scalling
k")
    print("3. rotate deg a")
    print("    melakukan rotasi objek berlawanan arah jarum jam
sebesar deg derajat terhadap sumbu-a")
    print("4. reflect x y z")
    print("    melakukan pencerminan terhadap titik (x,y,z)")
    print("5. shear x k1 k2")
    print("    melakukan operasi shear pada objek terhadap sumbu x
dengan faktor scalling k1 dan k2")
    print("6. stretch x k")
    print("    melakukan operasi stretch pada objek terhadap sumbu x
dengan faktor scalling k")
    print("7. custom a b c d e f g h i")
    print("    melakukan transformasi linier dengan matriks
transformasi [(a,b,c),(d,e,f),(g,h,i)]")
    print("8. multiple n")
    print("    melakukan transformasi linier pada objek sebanyak n
kali")
    print("Untuk kembali ke program utama, pilih 1")
    print("Masukkan pilihan: ", end = " ")
    n = int(input())
    if (n == 1):
        mainProgram()

```

```
#Main program (include making the windows)
def mainProgram():
    print("Selamat datang pada program simulasi transformasi
linier")
    print("Terdapat dua menu pada program ini yaitu simulasi 2
dimensi dan 3 dimensi")
    print("1. Tampilkan list perintah, silakan input 1")
    print("2. Untuk Memilih 2 dimensi, silakan input 2")
    print("3. Untuk Memilih 3 dimensi, silakan input 3")
    print("Masukan pilihan : ")
    x = int(input())
    if (x == 1):
        listofcommand()
    elif (x == 2):
        run2d()
    else:
        run3d()

#run the main program
mainProgram()
```

3.1.2 transformation2D.py

```
import math

def translate(self, x, y):
    for i in self.arrPoint:
        i.x += x
        i.y += y
    return self

def dilate(self, k):
    for i in self.arrPoint:
        i.x = i.x*k
        i.y = i.y*k
    return self

def reflect(self, comm, x, y):
    if (comm == 0):
        #reflect to x axis
        for i in self.arrPoint:
            i.y = -1 * i.y
    elif (comm == 1):
        #reflect to y axis
        for i in self.arrPoint:
            i.x = -1 * i.x
    elif (comm == 2):
        #reflect to y = x
        for i in self.arrPoint:
            i.x, i.y = i.y, i.x
    elif (comm == 3):
        #reflect to y = -x
        for i in self.arrPoint:
            tmp = i.x
            i.x = -i.y
            i.y = -tmp
    else:
        #reflect to point(x, y)
        for i in self.arrPoint:
            tmp = i.x - x
            i.x = i.x - 2*tmp
            tmp = i.y - y
            i.y = i.y - 2*tmp
    return self
```

```

def shear(self, a, k):
    if (a=='x'):
        for i in self.arrPoint:
            i.x += i.y*k
    elif (a=='y'):
        for i in self.arrPoint:
            i.y += i.x*k
    return self

def rotate(self, deg, a, b):
    deg = (2*deg*math.acos(-1))/360.0
    for i in self.arrPoint:
        i.x -= a
        i.y -= b
        tmpx = i.x * math.cos(deg) - i.y * math.sin(deg)
        tmpy = i.x * math.sin(deg) + i.y * math.cos(deg)
        i.x = tmpx
        i.y = tmpy
    return self

def stretch(self, param, k):
    if (param == 'x'):
        for i in self.arrPoint:
            i.x *= k
    else:
        for i in self.arrPoint:
            i.y *= k
    return self

def custom (self, a, b, c, d):
    for i in self.arrPoint:
        tx = i.x * a + i.y * b
        ty = i.x * c + i.y * d
        i.x = tx
        i.y = ty
    return self

def reset(self, point):
    for i, j in zip(self.arrPoint, point.arrPoint):
        i.x = j.x
        i.y = j.y
    return self

```

3.1.2 transformation3D.py

```
import math
def translate3D(self, x, y, z):
    for i in self.arrPoint:
        i.x += x
        i.y += y
        i.z += z
    return self

def dilate3D(self, k):
    for i in self.arrPoint:
        i.x *= k
        i.y *= k
        i.z *= k
    return self

def reflect3D(self, x, y, z):
    for i in self.arrPoint:
        tmp = i.x - x
        i.x -= 2*tmp
        tmp = i.y - y
        i.y -= 2*tmp
        tmp = i.z - z
        i.z -= 2*tmp
    return self

def shear3D(self, a, b, c):
    if (a=='x'):
        for i in self.arrPoint:
            i.x += b * i.y + c * i.z
    elif (a=='y'):
        for i in self.arrPoint:
            i.y += b * i.x + c * i.z
    elif (a == 'z'):
        for i in self.arrPoint:
            i.z += b * i.x + c * i.y
    return self
```

```

def rotate3D(self,a, deg):
    deg = (2*deg*math.acos(-1))/360.0
    cos = math.cos(deg)
    sin = math.sin(deg)
    if (a == 'x'):
        for i in self.arrPoint:
            tmp1 = i.y
            tmp2 = i.z
            i.y = tmp1 * cos - tmp2 * sin
            i.z = tmp1 * sin + tmp2 * cos
    elif (a == 'y'):
        for i in self.arrPoint:
            tmp1 = i.x
            tmp2 = i.z
            i.x = tmp1 * cos + tmp2 * sin
            i.z = -sin * tmp1 + tmp2 * cos
    elif (a == 'z'):
        for i in self.arrPoint:
            tmp1 = i.x
            tmp2 = i.y
            i.x = tmp1 * cos - tmp2 * sin
            i.y = tmp1 * sin + tmp2 * cos
    return self

def stretch3D(self, param, k):
    if (param == 'x'):
        for i in self.arrPoint:
            i.x *= k
    elif (param == 'y'):
        for i in self.arrPoint:
            i.y *= k
    elif (param == 'z'):
        for i in self.arrPoint:
            i.z *= k
    return self

def custom3D (self, a, b, c, d, e, f, g, h, j):
    for i in self.arrPoint:
        tx = (i.x) * a + (i.y) * d + (i.z) * g
        ty = (i.x) * b + (i.y) * e + (i.z) * h
        tz = (i.x) * c + (i.y) * f + (i.z) * j
        i.x = tx
        i.y = ty
        i.z = tz
    return self

```



```
def reset3D(self, point):
    for i, j in zip(self.arrPoint, point.arrPoint):
        i.x = j.x
        i.y = j.y
        i.z = j.z
    return self
```

3.2 Beberapa asumsi yang kami gunakan pada program ini:

1. Untuk fungsi *shear* pada 3 dimensi, kami menerima 3 buah *input*. *Input* yang pertama adalah sumbu yang akan digeser dan 2 *input* lainnya adalah besaran pergeseran dari dua sumbu selain sumbu yang mengalami pergeseran.
2. Untuk fungsi *rotate* pada 3 dimensi, kami menerima 2 buah *input*. *Input* yang pertama adalah besar rotasi yang diinginkan. Kemudian satu input lagi merupakan sumbu yang akan dirotasikan.

3.3 Pembagian Tugas Antar Kelompok

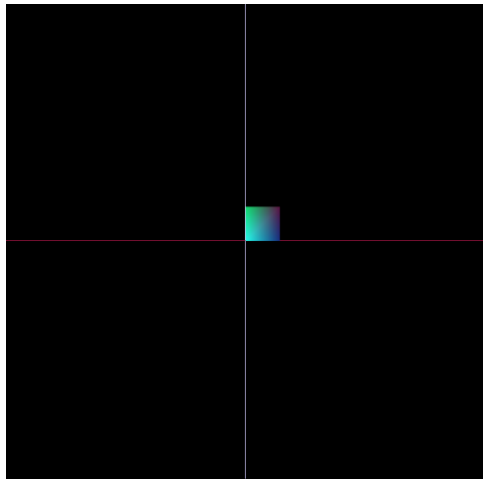
1. Winston Wijaya (13517018)
 - a. Pembuatan Operasi Fungsi dari 2-Dimensi
 - b. Pembuatan Operasi Fungsi dari 3-Dimensi
 - c. Penggambaran Objek untuk 2-Dimensi
 - d. Penggambaran Axis untuk Objek dari 2-Dimensi dan 3-Dimensi
 - e. Pengerjaan Laporan
2. Irfan Sofyana Putra (13517078)
 - a. Pembuatan Operasi Fungsi dari 2-Dimensi
 - b. Pembuatan Operasi Fungsi dari 3-Dimensi
 - c. Pengaturan Camera View pada Objek 3 Dimensi
 - d. Penggambaran Objek untuk 3-Dimensi
 - e. Pengerjaan Laporan
3. Jofiandy Leonata Pratama (13517135)
 - a. Pembuatan Operasi Fungsi dari 2-Dimensi
 - b. Pembuatan Operasi Fungsi dari 3-Dimensi
 - c. Pengaturan Fungsi Keyboard untuk mengatur Camera View dan fungsi pada Objek 3 Dimensi
 - d. Pengaturan Bentuk dan Alur pembuatan Keseluruhan Program
 - e. Pengerjaan Laporan

BAB IV

EKSPERIMEN

4.1 Eksperimen 2D

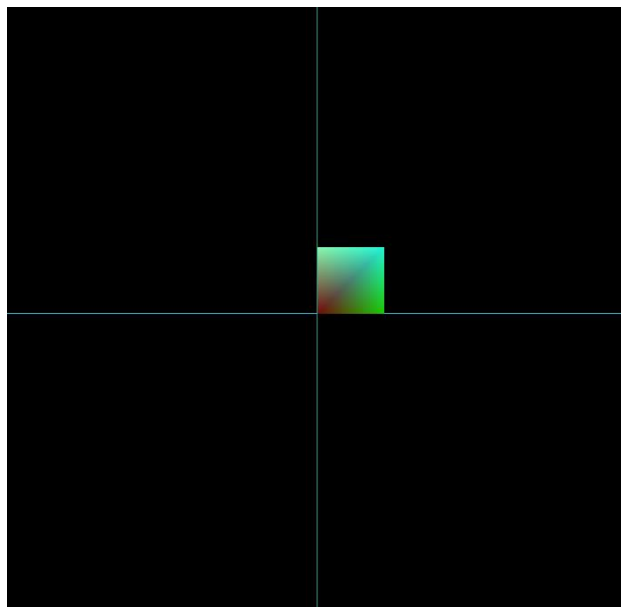
Pada percobaan 2D ini, kami menginputkan empat buah titik yaitu (0,0), (0,100), (100, 100), (100,0). Output dari objek ini adalah sebuah segiempat.



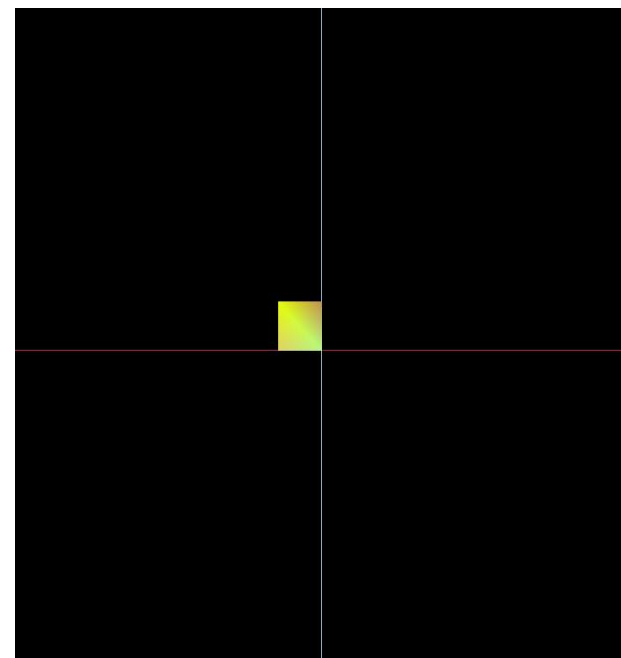
Hasil eksekusi program terhadap contoh-contoh kasus yang diberikan:

Input	Output
<pre>translate 200 100</pre>	

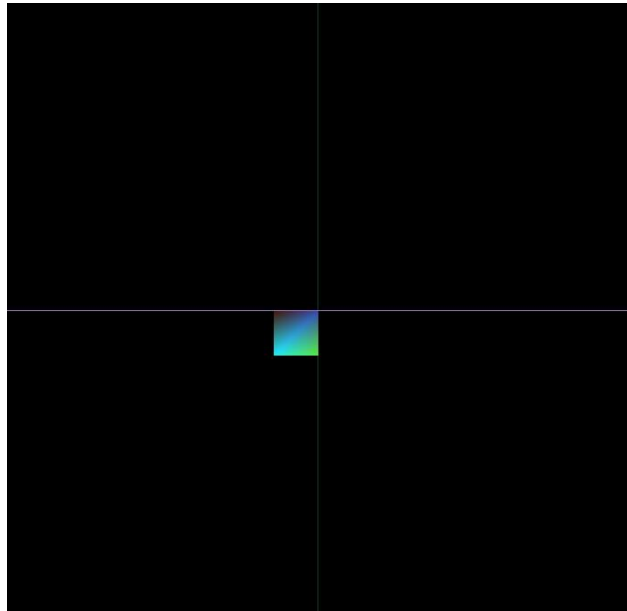
Dilate
1.5



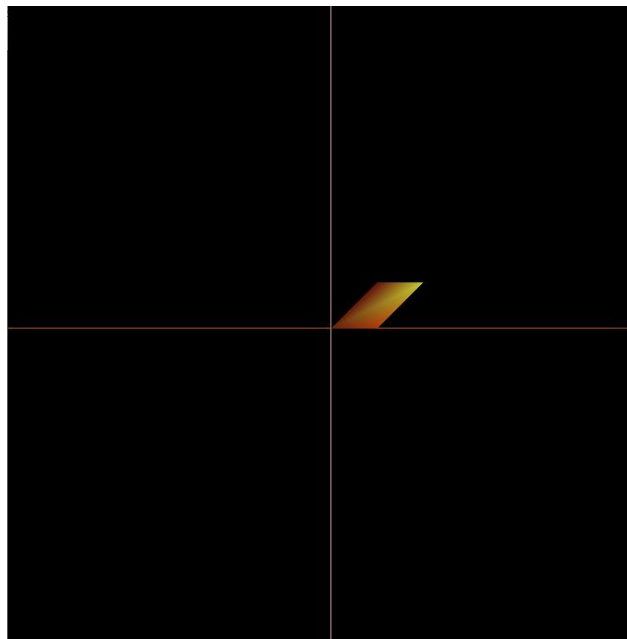
rotate 90 0 0



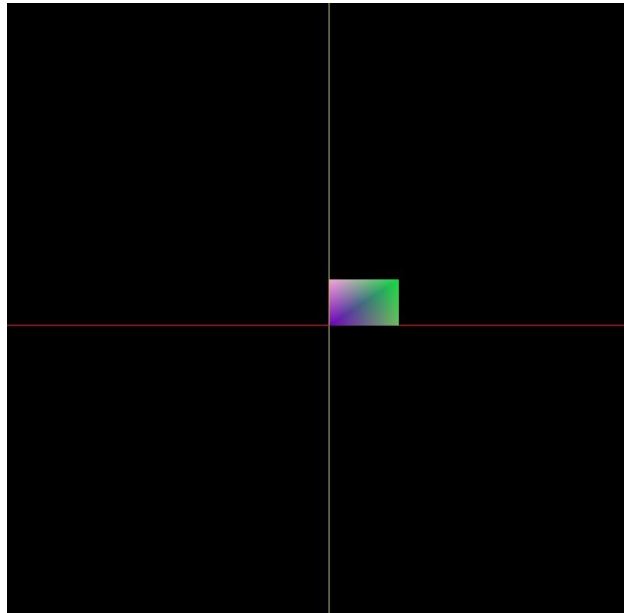
```
reflect  
0 0
```



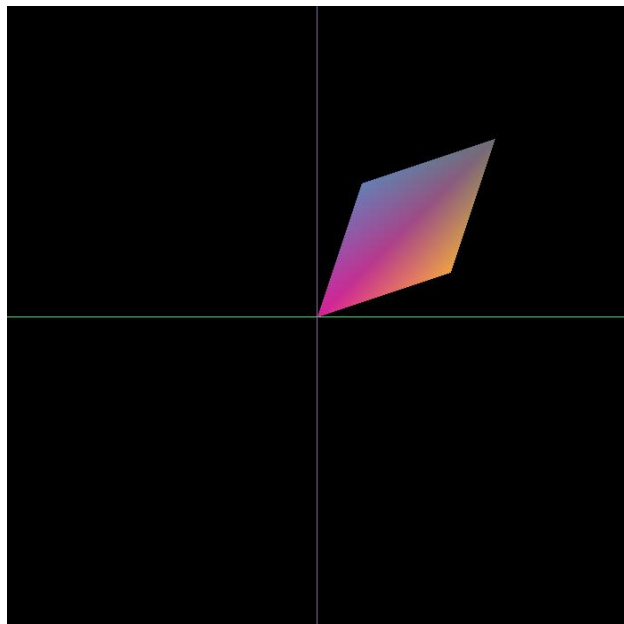
```
Shear  
x 1
```



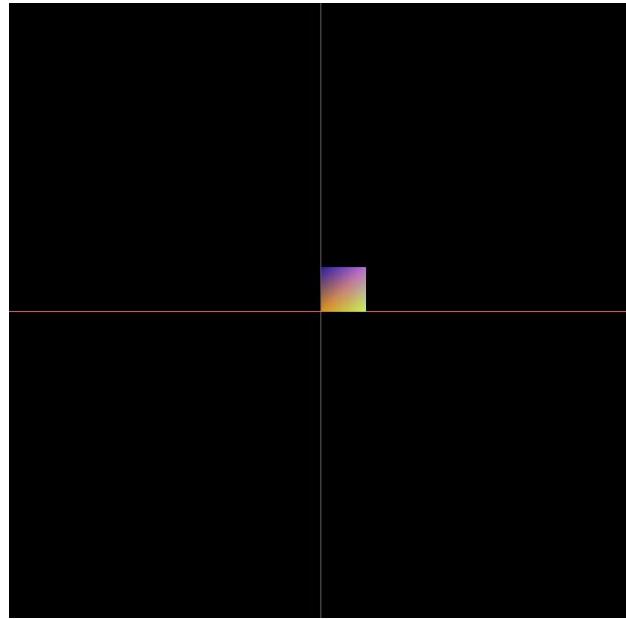
```
stretch  
x 1.5
```



```
custom  
1 3 3 1
```

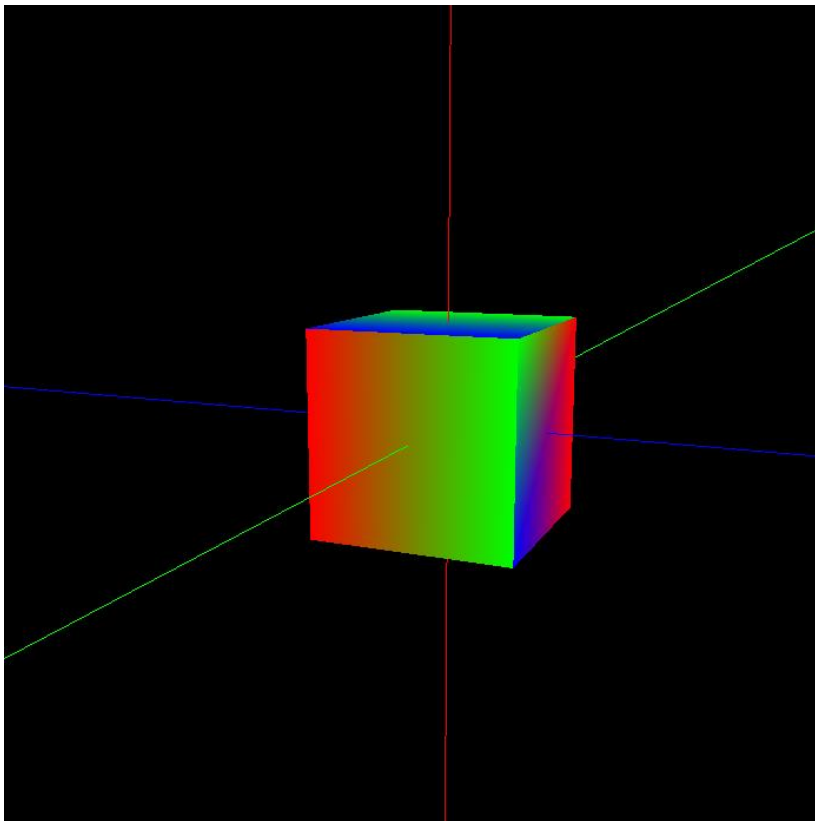


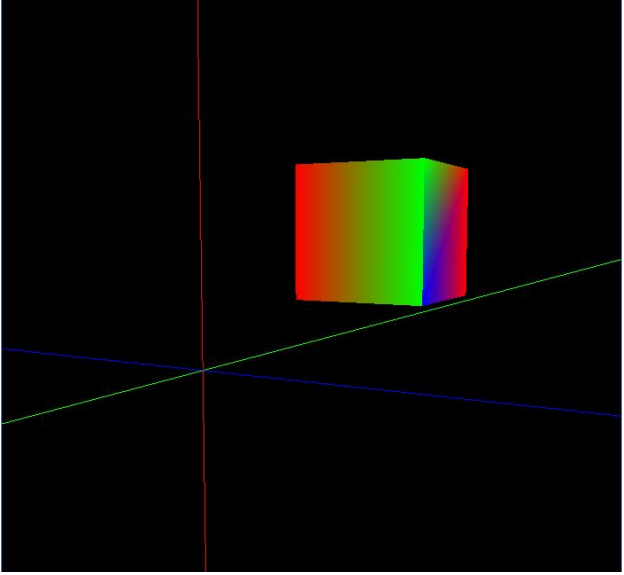
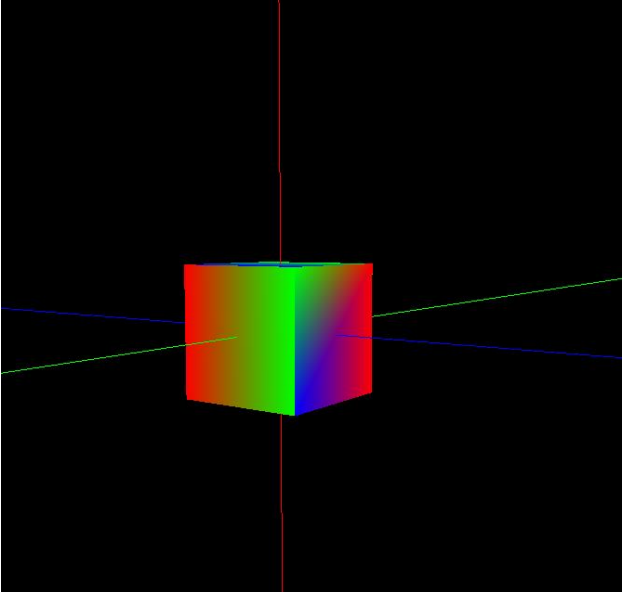
reset



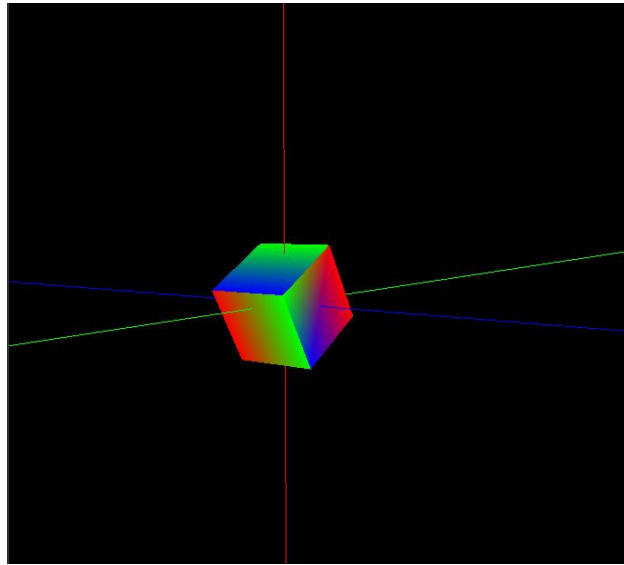
4.2 Eksperimen 3D

Pada percobaan 3D ini, kami akan membuat sebuah kubus yang berbentuk seperti gambar dibawah ini:

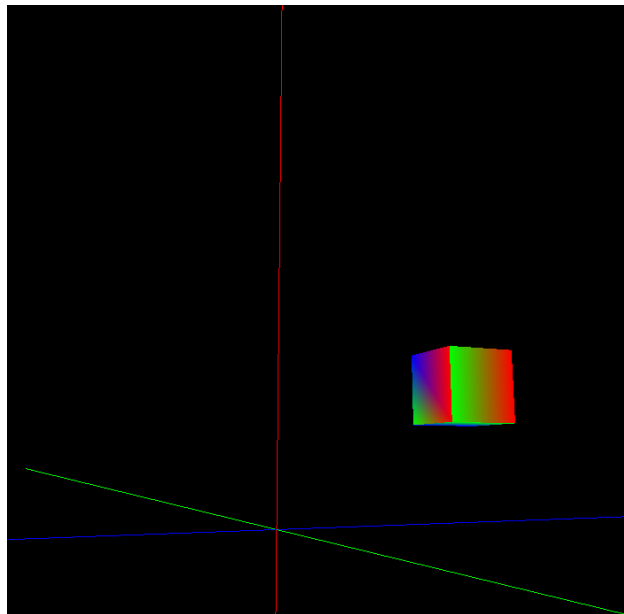


Input	Output
<pre>translate 1 1 -1</pre>	
<pre>dilate 1.5</pre>	

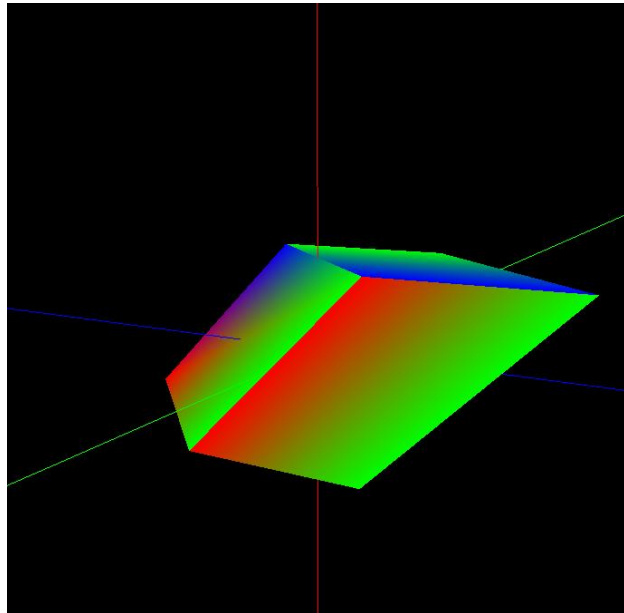
```
rotate  
x 30
```



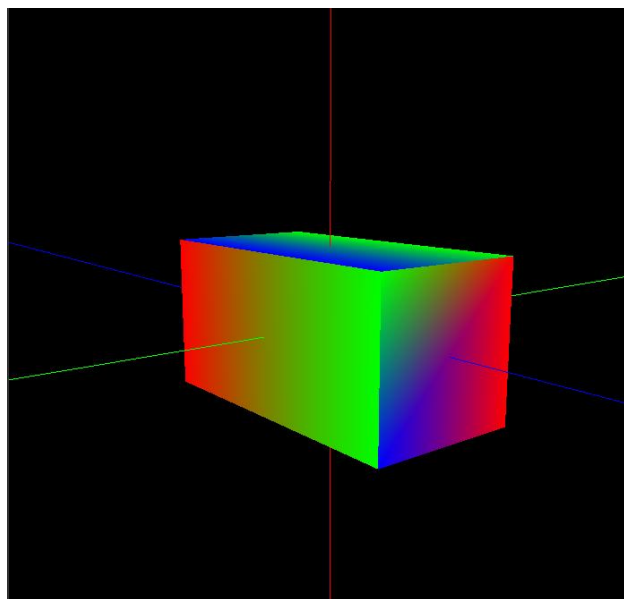
```
reflect  
1 1 1
```



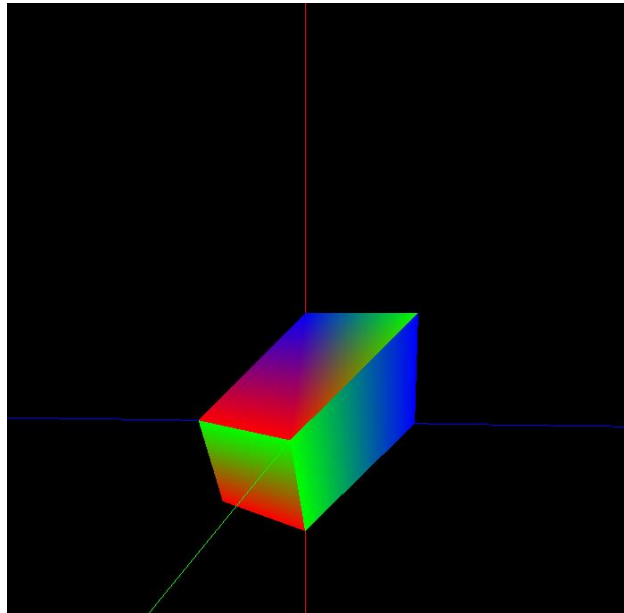

```
shear  
x 1 1
```



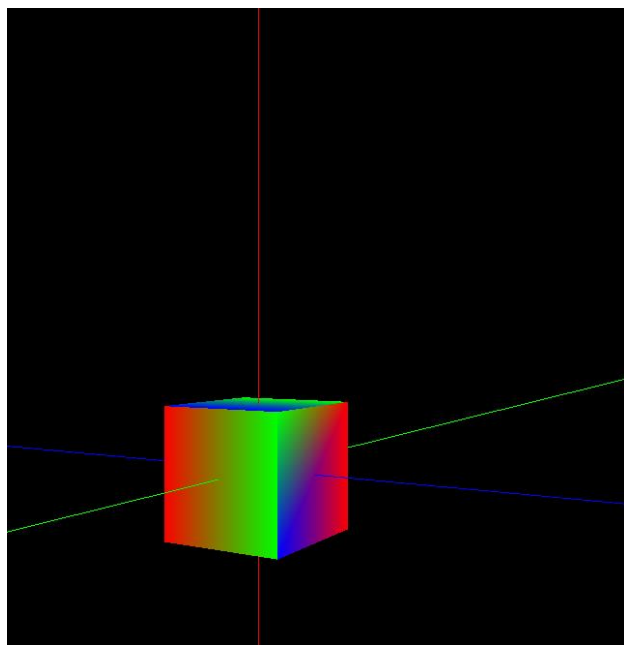
```
stretch  
x 2
```



```
custom  
0 1 1  
1 0 1  
1 1 0
```



```
reset
```



BAB V

PENUTUP

5.1 Kesimpulan

Transformasi linier adalah salah satu metode dalam Aljabar Geometri yang digunakan untuk melakukan perubahan posisi dari suatu objek khususnya titik pada sumbu koordinat. Transformasi linier ini memiliki banyak fungsi berupa Translasi, Dilatasi, Rotasi, Refleksi, Pergeseran, Peregangan. Metode Transformasi Linier ini banyak diaplikasikan pada banyak hal terutama untuk visualisasi grafis seperti pergerakan pemain pada Games, dan banyak lagi. Melalui metode ini, kita dapat memvisualisasikan perubahan yang terjadi akan suatu objek dan juga pengaplikasiannya yang tepat untuk dilaksanakan sehingga didapat hasil yang diharapkan.

OpenGL (**Open Graphics Library**) merupakan suatu spesifikasi standar yang mendefinisikan sebuah lintas-bahasa, lintas platform API untuk mengembangkan aplikasi yang menghasilkan grafis komputer 2D maupun 3D. OpenGL ini digunakan untuk visualisasi dari objek yang ingin digambarkan baik dari inputan titik-titik maupun juga mengeksekusikan fungsi-fungsi terhadap suatu titik. Selain itu, pada OpenGL ini kita dapat menggerakkan pandangan kita atau Camera View pada pembuatan Objek 3D. Hal ini memudahkan kita untuk memvisualisasikan objek yang ingin dibuat dan ini tidak menutup kemungkinan untuk diperlakukan beberapa fungsi pada objek 3D tersebut.

5.2 Saran

Untuk kedepannya, semoga spesifikasi yang diberikan pada tugas besar mata kuliah IF-2123 Aljabar Geometri mengenai transformasi linier ini lebih jelas, karena pada tugas ini ada beberapa fungsi transformasi yang menghasilkan penafsiran yang berbeda-beda dari tiap orang.

5.3 Refleksi

Melalui tugas besar IF-2123 Aljabar Geometri mengenai transformasi linier ini kami mendapatkan banyak sekali ilmu yang bermanfaat. Kami mendapatkan ilmu mengenai bagaimana cara menggunakan bahasa pemrograman *python* dan beserta *library*-nya. Kami juga mengulas kembali materi yang sudah kami pelajari mengenai matriks dan operasinya. Kami juga menjadi kenal dengan OpenGL karena tugas ini memang menuntut kita untuk menggunakan hal tersebut. Kami rasa OpenGL akan sangat membantu kami jika di kesempatan yang akan datang kami mendapat tugas yang membutuhkan visualisasi. Hal terpenting yang kami dapatkan dari

tugas ini adalah kami menjadi lebih paham dari transformasi linier itu sendiri karena langsung melihatnya secara visual.

DAFTAR REFERENSI

Sumber internet:

http://www.academia.edu/5271706/Matriks_transformasi

<http://jejak-kamera.blogspot.com/2015/03/pengertian-opengl-open-graphic-library.html>

Sumber buku:

Anton, Howard. (2011). *Elementary Linear Algebra Application Version 10th Edition*.

Anton, H., & Rorres, C. (2014). *Elementary Linear Algebra 11th Edition : Application Version*.

Wiley