LAPORAN TUGAS BESAR II

IF2123 ALJABAR GEOMETRI

//INSERT PHOTO HERE

DISUSUN OLEH :

EDWARD ALEXANDER JAYA

TIMOTHY

LOUIS CAHYADI (13517126)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2018

DAFTAR ISI

BAB I

DESKRIPSI MASALAH

Pada tugas kali ini, mahasiswa diminta membuat program yang mensimulasikan transformasi linier untuk melakukan operasi translasi, refleksi, rotasi, dan sebagainya padasebuah objek 2D dan 3D. Objek dibuat dengan mendefinisikan sekumpulan titik sudut lalu membuat bidang 2D/3D dari titik-titik tersebut. Contoh objek 2S : 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** |
| translate <dx> <dy> | Melakukan translasi objek dengan menggeser nilai x sebesar dx dan menggeseer nilai y sebesar dy |
| dilate <k> | Melakukan dilatasi objek dengan faktor scalling k |
| rotate <deg> <a> <b> | Melakukan rotasi objek secara berlawanan arah jarum jam sebesar deg derajat terhadap titik (a,b) |
| reflect <param> | Melakukan pencerminan objek. Nilai param adalah salah satu dari nilai berikut : x, y, y = x, y = -x, atau (a,b). Nilai (a,b) adalah titik untuk melakukan pencerminan terhadap. |
| shear <param> <k> | Melakukan operasi shear pada objek. Nilai param dapat berupa x (terhadap sumbu x) atau y (terhadap sumbu y). Nilai k adalah faktor shear |
| Stretch <param> <k> | Melakukan operasi stretch pada objek. Nilai param dapat berupa x (terhadap sumbu x) atau y (terhadap sumbu y). Nilai k adalah faktor stretch |
| custom <a><b><c><d> | Melakukan transformasi linier pada objek dengan matrix transformasi sebagai berikut : |
| multiple <n>  … // input 1  … // input 2  … // input 3  ...  … // input n | Melakukan transformasi linier pada objek sebanyak n kali berurutan. Setiap baris input 1..n dapat berupa translate, rotate, shear, dll tetapi bukan multiple, reset, exit. |
| reset | Mengembalikan objek pada kondisi awal objek didefinisikan. |
| exit | Keluar dari program. |

BAB II

DASAR TEORI

1. **Transormasi Linier**

Misalkan **x** adalah sebuah vektor pada *n-*dimensi. Ketika sebuah matriks **A** dengan ukuran n x n dikalikan dengan **x**, hal tersebut mentransformasi vektor menjadi vektor yang baru **Ax**. Hal ini terjadi pada semua titik **x** pada ruang n-dimensi **Rn**. Seluruh ruang ditransformasi, atau dipetakan ke dirinya sendiri oleh matriks **A**.

Sebuah transformasi disebut transformasi linier jika memenuhi ketiga syarat berikut ini :

1. Tidak memindah titik origin, karena **A0 = 0** untuk setiap matriks **A**
2. Jika vektor **x** dipetakan ke **x’** maka **cx** dipetakan ke **cx’** karena **A(cx) = c(Ax)**
3. Jika vektor **x**  dan vektor **y** dipetakan ke **x’** dan **y’**, maka vektor **x + y** juga dipetakan ke **x’ + y’** karena **A(x + y) = Ax + Ay**
4. **Matriks Transformasi**

Pada pengerjaan tugas ini, digunakan matrix tranformasi homogen yaitu untuk titik pada 2D terdapat element ketiga yang bernilai 1, dan bernilai nol jika matrix menunjukan vektor. Untuk 3D terdapat element keempat yang bernilai 1 jika menunjukan titik dan bernilai nol jika menunjukan vektor. Sebagai contoh

dan

Pada 2 dimensi, setiap titik ditulis sebagai baris pada matrix T yang dibentuk dari masukan pengguna. Semisal sebuah segitiga dengan titik pertama di (x1,y1), titik kedua di (x2,y2) dan titik ketiga di (x3,y3) maka segitiga ditulis sebagai matrix

Untuk melakukan transformasi, maka transformasi dilakukan pada setiap titik yang ada pada bangun dua dimensi tersebut sehingga matrix T akan dikali dengan sebuah matrix transformasi yang bersesuaian. Pada dua dimensi, berikut ini merupakan daftar matrix transformasi yang digunakan :

1. Translation <dx> <dy>
2. Dilate <k>
3. Rotate <deg> <a> <b>
4. Reflect <param>
   1. <x>
   2. <y>
   3. <y = x>
   4. <y = -x>
   5. <a,b>
5. Shear <param> <k>
   1. <x>
   2. <y>
6. Stretch <param>
   1. <x>
   2. <y>
7. Custom <a><b><c><d>

Pada 3D :

1. Translate <dx><dy><dz>
2. Dilate <k>
3. Rotate <param> <deg>
   1. <x>
   2. <y>
   3. <z>
4. Reflect <param>
   1. xy (Terhadap bidang xy)
   2. yz
   3. zx
   4. (0,0,0)
5. Shear <param> <k>
   1. x
   2. y
   3. z
6. Stretch <param> <k>
   1. x
   2. y
   3. z
7. **OpenGL**

Open Graphics Library (OpenGL) adalah sebuah Application Programming Interface (API) yang berfungsi untuk melakukan rendering grafik 2D dan 3D. OpenGL bersifat *cross-language, cross-platform,*  dan *open source*. OpenGL umumnya digunakan untuk melakukan interaksi dengan *Graphics Processing Unit (GPU)* untuk mencapai hasil render yang diakseleeerasi dengan hardware. Anda diharapkan untuk melakukan eksplorasi penggunaan OpenGL. Berikut ini adalah contoh kode program yang menggunakan library OpenGL :

|  |  |
| --- | --- |
| **Kode Program (khusus untuk objek 2D)** | **Keterangan** |
| Glfloat triangleVertices[] = {  100, 100, 0,  300, 100, 0,  250, 250, 0  }; | Mendefinisikan tiga buah titik, yaitu (100,100,0), (300,100,0), dan (250,250,0). Perhatikan nilai ketiga dari titik adalah nol supaya titik berupa 2D |
| GLFWindow \*window;  Window = glfwCreateWindow (600, 600,  “MyWindowName”, NULL, NULL); | Membuat sebuah window yang akan Anda gunakan untuk menampilkan output program. |
| glEnableClientState(GL\_VERTEX\_ARRAY);  glVertexPointer(3,GL\_FLOAT, 0, triangleVertices);  glDrawArrays(GL\_POLYGON,0,3);  glDisableClientState(GL\_VERTEX\_ARRAY); | Menggambar sebuah poligon sesuai titik – titik yang sudah didefinisikan pada triangleVertices. |

BAB III

IMPLEMENTASI

**I. Source Code**

import OpenGL

import math, string

from OpenGL.GLUT import \*

from OpenGL.GLU import \*

from OpenGL.GL import \*

from copy import copy, deepcopy

import sys

import time

import threading

#Gunakan tombol panah kanan,kiri,atas,bawah untuk menggerakan sudut kamera

#Gunakan 'W','A','S','D' untuk menggerakan kamera ke atas, kiri, bawah, kanan.

#Gunakan 'Z' dan 'X' untuk zoom in dan zoom out.

#Deklarasi Variabel Global

dim = 120.0 #msk

width = 1200 #lebar window

height = 675 #tinggi window

th = 0 #azimut / horizon angle

ph = 0 #elevation angle

#fov = 1.2 #field of view (sudut pandang)

asp = width / height #aspect ratio

length = 500.0 #panjang axis

movex = 0

movey = 0

movez = 0

global koordinatkubus

koordinatkubus = [

[1.0, 1.0, -1.0, 1.0],

[-1.0, 1.0, -1.0, 1.0],

[-1.0, 1.0, 1.0, 1.0],

[1.0, 1.0, 1.0, 1.0],

[1.0, -1.0, 1.0, 1.0],

[-1.0, -1.0, 1.0, 1.0],

[-1.0, -1.0, -1.0, 1.0],

[1.0, -1.0, -1.0, 1.0],

[1.0, 1.0, 1.0, 1.0],

[-1.0, 1.0, 1.0, 1.0],

[-1.0, -1.0, 1.0, 1.0],

[1.0, -1.0, 1.0, 1.0],

[1.0, -1.0, -1.0, 1.0],

[-1.0, -1.0, -1.0, 1.0],

[-1.0, 1.0, -1.0, 1.0],

[1.0, 1.0, -1.0, 1.0],

[-1.0, 1.0, 1.0, 1.0],

[-1.0, 1.0, -1.0, 1.0],

[-1.0, -1.0, -1.0, 1.0],

[-1.0, -1.0, 1.0, 1.0],

[1.0, 1.0, -1.0, 1.0],

[1.0, 1.0, 1.0, 1.0],

[1.0, -1.0, 1.0, 1.0],

[1.0, -1.0, -1.0, 1.0]

]

global firstkoordinatkubus

firstkoordinatkubus = deepcopy(koordinatkubus)

global matrix

global firstmatrix

global command

matrix=[] #define empty matrix

#Deklarasi Array dan Variabel2nya yang dibutuhkan

Neff = 100

#PosX[Neff] , Array berisi posisi X yang dimasukkan oleh user

#PosY[Neff] , Array berisi posisi Y yang dimasukkan oleh user

#PosZ[Neff] , Array berisi posisi Z yang dimasukkan oleh user

#Perintah[Neff], Array berisi perintah yang dimasukkan oleh user, kalau perintah tidak sesuai maka program looping ulang

#Deklarasi Input Awal

global msk

msk = "halo"

def MakeMatrix():

global firstmatrix

rows = int(input('Masukkan jumlah titik :'))

for i in range(0,rows): #total row

row=[]

for j in range(0,3): #total column

row.append(1.0) #adding 0 value for each column for this row

matrix.append(row) #add fully defined column into the row

for i in range(0,rows): #total row

elemen = input("titik {i+1} :")

z = elemen.split(' ')

matrix[i][0] = float(z[0]);

matrix[i][1] = float(z[1]);

firstmatrix = deepcopy(matrix)

def translate():

global koordinatkubus

global matrix

global msk

#global dx,dy,dz,deltadx,deltady,deltadz

deltadx = 0

deltady = 0

deltadz = 0

if(msk == 2):

timetemp = time.time()

tempkoordinat = matrix

dx = float(input('dx :'))

dy = float(input('dy :'))

while ((abs(deltadx) < abs(dx)) or (abs(deltady) < abs(dy))):

if (abs(deltadx) <abs(dx)):

deltadx = deltadx + 0.01\*dx

if (abs(deltady) <abs(dy)):

deltady = deltady + 0.01\*dy

b = [[1,0,0],[0,1,0],[deltadx,deltady,1]]

matrix = matmult(tempkoordinat,b)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

glutPostRedisplay()

elif(msk == 3):

dx = float(input('dx :'))

dy = float(input('dy :'))

dz = float(input('dz :'))

timetemp = time.time()

tempkoordinat = koordinatkubus

while ((abs(deltadx) < abs(dx)) or (abs(deltady) < abs(dy)) or (abs(deltadz) < abs(dz))):

if (abs(deltadx) <abs(dx)):

deltadx = deltadx + 0.01\*dx

if (abs(deltady) <abs(dy)):

deltady = deltady + 0.01\*dy

if (abs(deltadz) <abs(dz)):

deltadz = deltadz + 0.01\*dz

b = [[1,0,0,0],[0,1,0,0],[0,0,1,0],[deltadx,deltady,deltadz,1]]

koordinatkubus = matmult(tempkoordinat,b)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

glutPostRedisplay()

return

def dilate():

global koordinatkubus

global matrix

global msk

deltak = 1

if(msk == 2):

k = float(input("k :"))

tempkoordinat = matrix

timetemp = time.time()

if (abs(k) > 1):

while (abs(k) > abs(deltak)):

deltak = round ((deltak + (k-1)/10),7)

b = [[deltak,0,0],[0,deltak,0],[0,0,1]]

matrix = matmult(tempkoordinat,b)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

glutPostRedisplay()

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

else:

while (deltak > k):

deltak = round ((deltak - (1-k)/10),7)

b = [[deltak,0,0],[0,deltak,0],[0,0,1]]

matrix= matmult(tempkoordinat,b)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

glutPostRedisplay()

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

elif(msk == 3):

k = float(input("k :"))

timetemp = time.time()

tempkoordinat = koordinatkubus

#MASIH ADA BUG

if (abs(k) > 1):

deltak = 1

while (abs(k) > abs(deltak)):

deltak = round ((deltak + (k-1)/10),7)

print(deltak)

b = [[deltak,0,0,0],[0,deltak,0,0],[0,0,deltak,0],[0,0,0,1]]

koordinatkubus = matmult(tempkoordinat,b)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

glutPostRedisplay()

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

else:

while (deltak > k):

deltak = round ((deltak - (1-k)/10),7)

print(deltak)

b = [[deltak,0,0,0],[0,deltak,0,0],[0,0,deltak,0],[0,0,0,1]]

koordinatkubus = matmult(tempkoordinat,b)

newtime = time.time()

glutPostRedisplay()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

return

def rotate():

global koordinatkubus

global matrix

global msk

if(msk == 2):

degree = float(input('degree :'))

a = float(input('a :'))

b = float(input('b :'))

c = [[1,0,0],[0,1,0],[-a,-b,1]]

matrix = matmult(matrix,c)

degree = degree \* math.pi / 180

c = [[math.cos(degree),math.sin(degree),0],[-1\*math.sin(degree),math.cos(degree),0],[0,0,1]]

matrix = matmult(matrix,c)

c = [[1,0,0],[0,1,0],[a,b,1]]

matrix = matmult(matrix,c)

elif(msk == 3):

sumbu = input('sumbu :')

degree = float(input('degree :'))

degree = math.radians(degree)

print(degree)

deltadeg = 0

timetemp = time.time()

tempkoordinat = koordinatkubus

if(sumbu == 'x'):

while (abs(deltadeg) < abs(degree)):

print(deltadeg)

deltadeg = round ((deltadeg + (degree-0)/10),20)

b = [[1,0,0,0],[0,math.cos(deltadeg),-math.sin(deltadeg),0],[0,math.sin(deltadeg),math.cos(deltadeg),0],[0,0,0,1]]

koordinatkubus = matmult(tempkoordinat,b)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

glutPostRedisplay()

elif(sumbu == 'y'):

while (abs(deltadeg) < abs(degree)):

deltadeg = round ((deltadeg + (degree-0)/10),20)

b = [[math.cos(deltadeg),0,math.sin(deltadeg),0],[0,1,0,0],[-math.sin(deltadeg),0,math.cos(deltadeg),0],[0,0,0,1]]

koordinatkubus = matmult(tempkoordinat,b)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

glutPostRedisplay()

elif(sumbu == 'z'):

while (abs(deltadeg) < abs(degree)):

deltadeg = round ((deltadeg + (degree-0)/10),20)

b = [[math.cos(deltadeg),-math.sin(deltadeg),0,0],[math.sin(deltadeg),math.cos(deltadeg),0,0],[0,0,1,0],[0,0,0,1]]

koordinatkubus = matmult(tempkoordinat,b)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

glutPostRedisplay()

#print(koordinatkubus)

return

def reflect():

global koordinatkubus

global matrix

global msk

dinc = 0

if(msk == 2):

tempkoordinat = matrix

timetemp = time.time()

basis = str(input('basis :'))

if(basis == 'x'):

while (dinc < 2):

dinc = dinc + 0.01

b = [[1,0,0],[0,1-dinc,0],[0,0,1]]

matrix = matmult(tempkoordinat,b)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

glutPostRedisplay()

elif(basis == 'y'):

while (dinc < 2):

dinc = dinc + 0.01

b = [[1-dinc,0,0],[0,1,0],[0,0,1]]

matrix = matmult(tempkoordinat,b)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

glutPostRedisplay()

elif(basis =='y=x'):

b = [[0,1,0],[1,0,0],[0,0,1]]

matrix = matmult(matrix,b)

elif(basis == 'y=-x'):

while (dinc < 2):

dinc = dinc + 0.01

b = [[0,1-dinc,0],[1-dinc,0,0],[0,0,1]]

matrix = matmult(tempkoordinat,b)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

glutPostRedisplay()

elif(basis == 'a,b'):

a = float(input('a :'))

b = float(input('b :'))

c = [[1,0,0],[0,1,0],[-a,-b,1]]

matrix = matmult(matrix,c)

c = [[-1,0,0],[0,-1,0],[0,0,1]]

matrix = matmult(matrix,c)

c = [[1,0,0],[0,1,0],[a,b,1]]

matrix = matmult(matrix,c)

elif(msk == 3):

basis = str(input('basis :'))

tempkoordinat = koordinatkubus

timetemp = time.time()

if(basis == 'xy'):

while (dinc < 2):

dinc = dinc + 0.01

#b = [[-dinc,0,0,0],[0,-dinc,0,0],[0,0,dinc,0],[0,0,0,-dinc]]

b = [[1,0,0,0],[0,1,0,0],[0,0,1-dinc,0],[0,0,0,1]]

koordinatkubus = matmult(tempkoordinat,b)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

glutPostRedisplay()

elif(basis == 'yz'):

while (dinc < 2):

dinc = dinc + 0.01

b = [[1-dinc,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]]

koordinatkubus = matmult(tempkoordinat,b)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

glutPostRedisplay()

elif(basis == 'zx'):

while (dinc < 2):

dinc = dinc + 0.01

b = [[1,0,0,0],[0,1-dinc,0,0],[0,0,1,0],[0,0,0,1]]

koordinatkubus = matmult(tempkoordinat,b)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

glutPostRedisplay()

elif(basis == '0,0,0'):

while (dinc < 2):

dinc = dinc + 0.01

b = [[1-dinc,0,0,0],[0,1-dinc,0,0],[0,0,1-dinc,0],[0,0,0,1]]

koordinatkubus = matmult(tempkoordinat,b)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

glutPostRedisplay()

return

def shear():

global koordinatkubus

global matrix

global msk

deltak = 0

if(msk == 2):

timetemp = time.time()

tempkoordinat = matrix

param = str(input('sumbu :'))

k = float(input('k :'))

if(param == 'x'):

while (abs(deltak) < abs(k)):

deltak = deltak + 0.02\*k

c = [[1,0,0],[deltak,1,0],[0,0,1]]

matrix = matmult(tempkoordinat,c)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

glutPostRedisplay()

elif(param == 'y'):

while (abs(deltak) < abs(k)):

deltak = deltak + 0.02\*k

c = [[1,deltak,0],[0,1,0],[0,0,1]]

matrix = matmult(tempkoordinat,c)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

glutPostRedisplay()

if(msk == 3):

param = str(input('sumbu :'))

k = float(input('k :'))

timetemp = time.time()

tempkoordinat = koordinatkubus

if(param == 'x'):

while (abs(deltak) < abs(k)):

deltak = deltak + 0.02\*k

c = [[1,0,0,0],[deltak,1,0,0],[deltak,0,1,0],[0,0,0,1]]

koordinatkubus = matmult(tempkoordinat,c)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

glutPostRedisplay()

elif(param == 'y'):

while (abs(deltak) < abs(k)):

deltak = deltak + 0.02\*k

c = [[1,deltak,0,0],[0,1,0,0],[0,deltak,1,0],[0,0,0,1]]

koordinatkubus = matmult(tempkoordinat,c)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

glutPostRedisplay()

elif(param == 'z'):

while (abs(deltak) < abs(k)):

deltak = deltak + 0.02\*k

c = [[1,0,deltak,0],[0,1,deltak,0],[0,0,1,0],[0,0,0,1]]

koordinatkubus = matmult(koordinatkubus,c)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

glutPostRedisplay()

#print(koordinatkubus)

return

def stretch():

global koordinatkubus

global matrix

global msk

deltak = 1

if(msk == 2):

timetemp = time.time()

tempkoordinat = matrix

param = str(input('sumbu :'))

k = float(input('k :'))

if(param == 'x'):

if (abs(k) > 1):

deltak = 1

while (abs(deltak) < abs(k)):

deltak = round ((deltak + (k-1)/10),7)

c = [[deltak,0,0],[0,1,0],[0,0,1]]

matrix = matmult(tempkoordinat,c)

glutPostRedisplay()

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

else:

deltak = 1

while (deltak > k):

deltak = round ((deltak - (1-k)/10),7)

c = [[deltak,0,0],[0,1,0],[0,0,1]]

matrix = matmult(tempkoordinat,c)

glutPostRedisplay()

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

elif(param == 'y'):

if (abs(k) > 1):

deltak = 1

while (abs(deltak) < abs(k)):

deltak = round ((deltak + (k-1)/10),7)

c = [[1,0,0],[0,deltak,0],[0,0,1]]

matrix = matmult(tempkoordinat,c)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

glutPostRedisplay()

else:

deltak = 1

while (deltak > k):

deltak = round ((deltak - (1-k)/10),7)

c = [[1,0,0],[0,deltak,0],[0,0,1]]

matrix = matmult(tempkoordinat,c)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

glutPostRedisplay()

elif(msk == 3):

param = str(input('sumbu :'))

k = float(input('k :'))

timetemp = time.time()

tempkoordinat = koordinatkubus

if(param == 'x'):

if (abs(k) > 1):

deltak = 1

while (abs(deltak) < abs(k)):

deltak = round ((deltak + (k-1)/10),7)

c = [[deltak,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]]

koordinatkubus = matmult(tempkoordinat,c)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

glutPostRedisplay()

else:

deltak = 1

while (deltak > k):

deltak = round ((deltak - (1-k)/10),7)

c = [[deltak,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]]

koordinatkubus = matmult(tempkoordinat,c)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

glutPostRedisplay()

elif(param == 'y'):

if (abs(k) > 1):

deltak = 1

while (abs(deltak) < abs(k)):

deltak = round ((deltak + (k-1)/10),7)

c = [[1,0,0,0],[0,deltak,0,0],[0,0,1,0],[0,0,0,1]]

koordinatkubus = matmult(tempkoordinat,c)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

glutPostRedisplay()

else:

deltak = 1

while (deltak > k):

deltak = round ((deltak - (1-k)/10),7)

c = [[1,0,0,0],[0,deltak,0,0],[0,0,1,0],[0,0,0,1]]

koordinatkubus = matmult(tempkoordinat,c)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

glutPostRedisplay()

elif(param == 'z'):

if (abs(k) > 1):

deltak = 1

while (abs(deltak) < abs(k)):

deltak = round ((deltak + (k-1)/10),7)

c = [[1,0,0,0],[0,1,0,0],[0,0,deltak,0],[0,0,0,1]]

koordinatkubus = matmult(tempkoordinat,c)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

glutPostRedisplay()

else:

deltak = 1

while (deltak > k):

deltak = round ((deltak - (1-k)/10),7)

c = [[1,0,0,0],[0,1,0,0],[0,0,deltak,0],[0,0,0,1]]

koordinatkubus = matmult(tempkoordinat,c)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

glutPostRedisplay()

return

def custom():

global koordinatkubus

global matrix

da = 0

db = 0

dc = 0

dd = 0

de = 0

df = 0

dg = 0

dh = 0

di = 0

if (msk == 2):

a = float(input('a :'))

b = float(input('b :'))

c = float(input('c :'))

d = float(input('d :'))

timetemp = time.time()

tempkoordinat = matrix

while ((abs(da) < abs(a)) or (abs(db) < abs(b)) or (abs(dc) < abs(c)) or (abs(dd) < abs(d))):

if ((abs(da) < abs(a))):

da = da + 0.01\*a

if ((abs(db) < abs(b))):

db = db + 0.01\*b

if ((abs(dc) < abs(c))):

dc = dc + 0.01\*c

if ((abs(dd) < abs(d))):

dd = dd + 0.01\*d

e = [[da,dc,0],[db,dd,0],[0,0,1]]

matrix = matmult(tempkoordinat,e)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

glutPostRedisplay()

elif (msk == 3):

a = float(input('a :'))

b = float(input('b :'))

c = float(input('c :'))

d = float(input('d :'))

e = float(input('e :'))

f = float(input('f :'))

g = float(input('g :'))

h = float(input('h :'))

i = float(input('i :'))

timetemp = time.time()

tempkoordinat = koordinatkubus

while ((abs(da) < abs(a)) or (abs(db) < abs(b)) or (abs(dc) < abs(c)) or (abs(dd) < abs(d)) or (abs(de) < abs(e)) or (abs(df) < abs(f)) or (abs(dg) < abs(g)) or (abs(dh) < abs(h)) or (abs(di) < abs(i))):

if ((abs(da) < abs(a))):

da = da + 0.01\*a

if ((abs(db) < abs(b))):

db = db + 0.01\*b

if ((abs(dc) < abs(c))):

dc = dc + 0.01\*c

if ((abs(dd) < abs(d))):

dd = dd + 0.01\*d

if ((abs(de) < abs(e))):

de = de + 0.01\*e

if ((abs(df) < abs(f))):

df = df + 0.01\*f

if ((abs(dg) < abs(g))):

dg = dg + 0.01\*g

if ((abs(dh) < abs(h))):

dh = dh + 0.01\*h

if ((abs(di) < abs(i))):

di = di + 0.01\*i

j = [[da,dd,dg,0],[db,de,dh,0],[dc,df,di,0],[0,0,0,1]]

koordinatkubus = matmult(tempkoordinat,j)

newtime = time.time()

sleeptime = ((1000/60) - (newtime - timetemp))/1000.0

if sleeptime > 0:

time.sleep(sleeptime)

timetemp = newtime

glutPostRedisplay()

#print(koordinatkubus)

return

def reset():

global koordinatkubus

global matrix

#global firstmatrix

if (msk == 2):

matrix = deepcopy(firstmatrix)

elif (msk == 3):

koordinatkubus = deepcopy(firstkoordinatkubus)

return

def multiple():

global koordinatkubus

i = int(input('Banyak perintah :'))

for i in range(i):

print(i+1)

print(' :')

command = input('')

if(command == 'translate'):

translate()

glutPostRedisplay()

elif(command == 'dilate'):

dilate()

glutPostRedisplay()

elif(command == 'rotate'):

rotate()

glutPostRedisplay()

elif(command == 'reflect'):

reflect()

glutPostRedisplay()

elif(command == 'shear'):

shear()

glutPostRedisplay()

elif(command == 'stretch'):

stretch()

glutPostRedisplay()

def matmult(a,b):

global width,height

zip\_b = zip(\*b)

zip\_b = list(zip\_b)

return [[sum(ele\_a\*ele\_b for ele\_a, ele\_b in zip(row\_a, col\_b))

for col\_b in zip\_b] for row\_a in a]

def LogicDraw() :

global dx,dy,dz

glMatrixMode(GL\_PROJECTION)

glLoadIdentity()

glOrtho(-dim\*asp,+dim\*asp, -dim,+dim, -dim,+dim)

glTranslate( -movex , -movey , -movez )

glMatrixMode(GL\_MODELVIEW)

glLoadIdentity()

return

def DrawCube3D () :

#Kode Untuk Membuat Kubus

#Membuat Grafik Khusus 3D

glBegin(GL\_QUADS)

global koordinatkubus

#koordinatkubus = [[1.0, 1.0, -1.0],[-1.0, 1.0, -1.0],[-1.0, 1.0, 1.0],[1.0, 1.0, 1.0]]

#Menyebutkan titik sudut kubus pada setiap sisi, beserta warna pada setiap sisi

#Sisi atas kubus, warna hijau

glColor3f(0.0, 1.0, 0.0)

glVertex3f(koordinatkubus[0][0], koordinatkubus[0][1], koordinatkubus[0][2])

glVertex3f(koordinatkubus[1][0], koordinatkubus[1][1], koordinatkubus[1][2])

glVertex3f(koordinatkubus[2][0], koordinatkubus[2][1], koordinatkubus[2][2])

glVertex3f(koordinatkubus[3][0], koordinatkubus[3][1], koordinatkubus[3][2])

#Sisi bawah kubus, warna kuning

glColor3f(1.0, 0.5, 0.0)

glVertex3f(koordinatkubus[4][0], koordinatkubus[4][1], koordinatkubus[4][2])

glVertex3f(koordinatkubus[5][0], koordinatkubus[5][1], koordinatkubus[5][2])

glVertex3f(koordinatkubus[6][0], koordinatkubus[6][1], koordinatkubus[6][2])

glVertex3f(koordinatkubus[7][0], koordinatkubus[7][1], koordinatkubus[7][2])

#Sisi depan kubus

glColor3f(1.0, 0.0, 0.0)

glVertex3f(koordinatkubus[8][0], koordinatkubus[8][1], koordinatkubus[8][2])

glVertex3f(koordinatkubus[9][0], koordinatkubus[9][1], koordinatkubus[9][2])

glVertex3f(koordinatkubus[10][0], koordinatkubus[10][1], koordinatkubus[10][2])

glVertex3f(koordinatkubus[11][0], koordinatkubus[11][1], koordinatkubus[11][2])

#Sisi belakang kubus

glColor3f(1.0, 1.0, 0.0)

glVertex3f(koordinatkubus[12][0], koordinatkubus[12][1], koordinatkubus[12][2])

glVertex3f(koordinatkubus[13][0], koordinatkubus[13][1], koordinatkubus[13][2])

glVertex3f(koordinatkubus[14][0], koordinatkubus[14][1], koordinatkubus[14][2])

glVertex3f(koordinatkubus[15][0], koordinatkubus[15][1], koordinatkubus[15][2])

#Sisi kiri kubus

glColor3f(0.0, 0.0, 1.0)

glVertex3f(koordinatkubus[16][0], koordinatkubus[16][1], koordinatkubus[16][2])

glVertex3f(koordinatkubus[17][0], koordinatkubus[17][1], koordinatkubus[17][2])

glVertex3f(koordinatkubus[18][0], koordinatkubus[18][1], koordinatkubus[18][2])

glVertex3f(koordinatkubus[19][0], koordinatkubus[19][1], koordinatkubus[19][2])

#Sisi kanan kubus

glColor3f(1.0, 0.0, 1.0)

glVertex3f(koordinatkubus[20][0], koordinatkubus[20][1], koordinatkubus[20][2])

glVertex3f(koordinatkubus[21][0], koordinatkubus[21][1], koordinatkubus[21][2])

glVertex3f(koordinatkubus[22][0], koordinatkubus[22][1], koordinatkubus[22][2])

glVertex3f(koordinatkubus[23][0], koordinatkubus[23][1], koordinatkubus[23][2])

glEnd()

return

def DrawAxis () :

glBegin(GL\_LINES)

glColor3f(0.0,1.0,1.0)

glVertex3d(0.0,0.0,0.0)

glVertex3d(-1\*length,0.0,0.0)

glVertex3d(length,0.0,0.0)

glVertex3d(0,0,0)

glVertex3d(0,length,0)

glVertex3d(0,-1\*length,0)

glVertex3d(0,0,0)

glVertex3d(0.0,0.0,length)

glVertex3d(0,0,0)

glVertex3d(0.0,0.0,-1.0\*length)

glEnd()

return

def DrawGuide () :

glBegin(GL\_LINES)

glColor4f(0.5, 0.5, 0.5, 0.3) #mengurangi opacity :)

#Mari kita gambar guide lines, sejajar sumbu x

i = -length

while (i <= length):

glVertex3d(0.0,0.0,i)

glVertex3d(length,0.0,i)

i = i + 50

i = -length

while (i<= length):

glVertex3d(0.0,0.0,i)

glVertex3d(-1\*length,0.0,i)

i = i + 50

#Mari kita gambar guide lines, sejajar sumbu z

i = -length

while (i <= length):

glVertex3d(i,0.0,0.0)

glVertex3d(i,0.0,length)

i = i + 50

i = -length

while (i <= length):

glVertex3d(i,0.0,0.0)

glVertex3d(i,0.0,-1\*length)

i = i + 50

glEnd()

def PrintWindow (x, y ,z, text) :

glColor3f(1,1,1)

glRasterPos3f(x,y,z)

for i in range (0,len(text)) :

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12,ord(text[i]))

def PrintWindow2D (x, y, text) :

glColor3f(1,1,1)

glRasterPos2f(x,y)

for i in range (0,len(text)) :

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12,ord(text[i]))

def PrintInfoGuide (x, y ,z, text) :

glColor4f(1, 1, 1, 0.3);

glRasterPos3f(x,y,z)

for i in range (0,len(text)) :

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_10,ord(text[i]))

def DrawInfo () :

global length

PrintWindow(koordinatkubus[3][0],koordinatkubus[3][1],koordinatkubus[3][2],"x = \n" + str((koordinatkubus[3][0])) +",\n y = " + str((koordinatkubus[3][1])) + ",\n z = " + str((koordinatkubus[3][2])))

PrintWindow(length+10,0.0,0.0,"X")

PrintWindow(0.0,length+10,0.0,"Y")

PrintWindow(0.0,0.0,length+10,"Z")

i = -500

while (i <= length):

PrintInfoGuide(i,0,0,str(i))

i = i + 50

return

def DrawInfo2D() :

global length

PrintWindow2D(length+50,0,"X")

PrintWindow2D(0,length+50,"Y")

PrintWindow2D(matrix[0][0],matrix[0][1],"x =" + str(round(matrix[0][0],2)) +",y = " + str(round(matrix[0][1],2)) )

i = -500

while (i <= length):

PrintWindow2D(i,0,str(i))

i = i + 50

def display() :

glEnable(GL\_BLEND)

glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA)

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

glEnable(GL\_DEPTH\_TEST)

glLoadIdentity()

glRotatef(ph,1,0,0);

glRotatef(th,0,1,0);

DrawCube3D()

DrawInfo()

DrawAxis()

DrawGuide()

LogicDraw()

glFlush()

glutSwapBuffers()

return

def reshape (widthtemp , heighttemp) : #Menjaga aspect ratio dari "world"

global width,height

widthtemp = width

heighttemp = height

asp = widthtemp / heighttemp

glViewport(0,0, (widthtemp), (heighttemp))

LogicDraw()

def keyboardSpecial(key, x, y):

global th

global ph

if (key == GLUT\_KEY\_RIGHT) :

th = th + 5

elif (key == GLUT\_KEY\_LEFT) :

th = th - 5

elif (key == GLUT\_KEY\_UP) :

ph = ph + 5

elif (key == GLUT\_KEY\_DOWN) :

ph = ph - 5

th = th % 360

ph = ph % 360

LogicDraw()

glutPostRedisplay()

return

def keyboardKey (bkey , x , y) :

global fov

global movex,movey,movez

global dim

global length

key = bkey.decode("utf-8")

if (key == 's') :

movey = movey - 5

elif (key == 'w') :

movey = movey + 5

elif (key == 'd') :

movex = movex + 5

elif (key == 'a') :

movex = movex - 5

elif (key == 'z') :

if (dim > 10):

dim = dim - 10

else:

dim = dim + 10

elif (key == 'x') :

dim = dim +10

elif (key == 't') :

glutLeaveMainLoop()

LogicDraw()

glutPostRedisplay()

def Draw3DWorld () :

global command

glutInit(sys.argv)

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

glutInitWindowSize(width, height)

glutCreateWindow("3D SYSTEM BETA")

glutDisplayFunc(display)

glutReshapeFunc(reshape)

glutSpecialFunc(keyboardSpecial)

glutKeyboardFunc(keyboardKey)

glutMainLoop()

if(command =='quit'):

#sys.exit

glutLeaveMainLoop()

return

def init():

glClearColor(0.0,0.0,0.0,1.0)

gluOrtho2D(-dim\*asp,+dim\*asp, -dim,+dim)

#gluOrtho2D(-500.0,500.0,-500.0,500.0)

def Draw2DAxis() :

glClear(GL\_COLOR\_BUFFER\_BIT)

glColor3f(0.0,1.0,1.0)

glBegin(GL\_LINES)

glVertex2f(-1\*length,0.0)

glVertex2f(length,0.0)

glVertex2f(0.0,-1\*length)

glVertex2f(0.0,length)

glEnd()

glBegin (GL\_LINES)

glColor4f(0.5, 0.5, 0.5, 0.3)

i = -length

while (i <= length):

if (i != 0):

glVertex2f(i,-length)

glVertex2f(i,length)

glVertex2f(-length,i)

glVertex2f(length,i)

i = i + 50

glEnd()

glFlush()

def plotmatrix():

Draw2DAxis()

DrawInfo2D()

glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA)

glBegin(GL\_POLYGON)

glColor3f(1.0,0.0,0.0)

for i in range(0,len(matrix)-1):

glVertex2f(matrix[i][0],matrix[i][1])

glVertex2f(matrix[i+1][0],matrix[i+1][1])

glVertex2f(matrix[len(matrix)-1][0],matrix[len(matrix)-1][1])

glVertex2f(matrix[0][0],matrix[0][1])

glEnd()

glFlush()

def Draw2DWorld():

global command

glutInit(sys.argv)

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB)

glutInitWindowSize(width, height)

glutCreateWindow("2D SYSTEM BETA")

glutDisplayFunc(plotmatrix)

init()

glutReshapeFunc(reshape)

glutSpecialFunc(keyboardSpecial)

glutKeyboardFunc(keyboardKey)

glutMainLoop()

if(command =='quit'):

#sys.exit

glutLeaveMainLoop()

return

def main3():

Draw2DWorld()

display()

def main2():

Draw3DWorld()

display()

def main1() :

global dx,dy,dz

global koordinatkubus

global msk

global command

print("Selamat Datang di TUBES 2 ALGEO!")

msk = int(input("Silakan input apakah Anda mau 2D atau 3D!\n"))

if (msk == 3):

thread.start() #Window 3d World

#thread.join()

#thread2.start()

command = ''

#command = input('Silakan masukan perintah :\n')

while (command != 'quit'):

command = input('Command : ')

if(command == 'translate'):

translate()

#glutPostRedisplay()

elif(command == 'dilate'):

dilate()

glutPostRedisplay()

elif(command == 'rotate'):

rotate()

elif(command == 'reflect'):

reflect()

glutPostRedisplay()

elif(command == 'shear'):

shear()

glutPostRedisplay()

elif(command == 'stretch'):

stretch()

glutPostRedisplay()

elif(command == 'reset'):

reset()

glutPostRedisplay()

elif(command == 'multiple'):

multiple()

glutPostRedisplay()

elif(command == 'custom'):

custom()

glutPostRedisplay()

elif(command == 'quit'):

#sys.exit()

glutLeaveMainLoop()

break

elif (msk == 2):

MakeMatrix()

thread3.start() #window 2d World

command = ''

#command = input('Silakan masukan perintah :\n')

while (command != 'quit'):

command = input('Command : ')

if(command == 'translate'):

translate()

glutPostRedisplay()

elif(command == 'dilate'):

dilate()

glutPostRedisplay()

elif(command == 'rotate'):

rotate()

glutPostRedisplay()

elif(command == 'reflect'):

reflect()

glutPostRedisplay()

elif(command == 'shear'):

shear()

glutPostRedisplay()

elif(command == 'stretch'):

stretch()

glutPostRedisplay()

elif(command == 'reset'):

reset()

glutPostRedisplay()

elif(command == 'custom'):

custom()

glutPostRedisplay()

elif(command == 'multiple'):

multiple()

glutPostRedisplay()

elif(command == 'quit'):

#sys.exit()

glutLeaveMainLoop()

break

return

thread = threading.Thread(target = main2)

thread2 = threading.Thread(target = main1)

thread3 = threading.Thread(target = main3)

thread2.start()

**II. Pembagian Tugas**

Edward A.J. : OpenGL3D, Animasi, Laporan

Timothy : Matriks Transformasi (Code), Threading, Laporan

Louis C. : Matriks Transformasi (Teori), OpenGL2D, Laporan

BAB IV

EKSPERIMEN

**Eksekusi Program**

|  |  |
| --- | --- |
| Gambar 3D  Default Position |  |
| Gambar 3D  Dilate 20 dari gambar sebelumnya  Benda diperbesar sebanyak 20 kali lipat |  |
| Gambar 3D  Rotate X 45o dari gambar sebelumnya  Benda diputar sejauh 45o dengan sumbu putar x |  |
| Gambar 3D  Shear y 1.2 dari gambar sebelumnya  Benda di “shear” dengan sumbu y sebanyak 1.2 kali |  |
| Gambar 3D  Refleksi terhadap basis xy dari gambar sebelumnya |  |
| Gambar 3D  stretch x 2 dari gambar sebelumnya  Benda direnggangkan mengikuti sumbu x sebanyak 2 kali |  |
| Gambar 2D  Default Position  Titik (0,0),(50,0),(20,30) |  |
| Gambar 2D  translate 10 10 dari gambar sebelumnya  Benda di geser sebanyak 10 satuan ke kanan dan 10 satuan ke atas |  |
| Gambar 2D  reflect x dari gambar sebelumnya  Benda dicerminkan terhadap sumbu x |  |
| Gambar 2D  Stretch x 2 dari gambar sebelumnya  Benda direnggangkan sebanyak 2 kali sepanjang sumbu x |  |

BAB V

KESIMPULAN DAN SARAN

**I. Kesimpulan**

Dari percobaan di atas, maka dapat kami simpulkan bahwa transformasi matriks dapat diselesaikan dengan menggunakan perkalian matriks yang sesuai dengan transformasi yang akan dilakukan. Segala bentuk transformasi yang dilakukan dalam ruang R, akan menghasilkan bentuk baru yang masih merupakan anggota dari ruang R.

OpenGL dapat menggambarkan hasil perkalian matriks yang telah dilakukan dengan menggunakan visualisasi gambar bentuk, baik itu dalam 2 dimensi maupun 3 dimensi. Untuk menjalankan OpenGl dan perkalian matriks secara konstan dalam waktu yang bersamaan, diperlukan *threading*, dimana setiap *thread* ditugaskan untuk menjalankan masing-masing tugas yang telah diberikan selamaprogram masih berjalan.

**II. Saran**

Untuk pengembangan program selanjutnya, penulis menyarankan untuk mengembangkan lebih banyak transformasi yang dapat diterapkan pada bidang yang telah dibentuk. Selain itu, kami juga menyarankan agar bentuk 3 dimensi tidak terbatas hanya pada bentuk kubus saja. Program akan lebih baik jika pengguna dapat memasukkan sembarang titik dan program dapat membentuk model 3 dimensinya.

**III. Sumber Referensi**

*-* [*https://docs.python.org/2/library/threading.html*](https://docs.python.org/2/library/threading.html)

*-* [*https://stackoverflow.com/questions/2846653/how-to-use-threading-in-python*](https://stackoverflow.com/questions/2846653/how-to-use-threading-in-python)