

LAPORAN TUGAS BESAR II

IF2123 : ALJABAR GEOMETRI



Dibuat Oleh :
Kelompok Sabeb Le

Anggota :

Aditya Putra Santosa	13517013
Harry Rahmadi Munly	13517033
Leonardo	13517048

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

Daftar Isi :

Bab I. Deskripsi Permasalahan	2
Bab II. Teori Singkat	8
Translasi (<i>Translate</i>)	8
Rotasi (<i>Rotate</i>)	9
Dilatasi (<i>Dilate</i>)	10
Refleksi (<i>Reflect</i>)	10
Menggeser (<i>Shear</i>)	11
Meregang (<i>Stretch</i>)	12
Matriks menurut input pengguna (<i>Custom</i>)	12
Multi-perintah (<i>Multiple</i>)	13
OpenGL	13
Bab III. Implementasi Program	14
main.py	14
transformasi.py	14
ProcInput.py	17
config.py	18
render.py	19
bentuk.py	21
handler.py	21
animasi.py	22
Bab IV. Eksperimen	23
Transformasi di ruang 2D	23
Transformasi di ruang 3D	25
Input Salah	28
Bab V. Kesimpulan, Saran, dan Refleksi	28
Daftar Pustaka	29

Bab I. Deskripsi Permasalahan

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**.

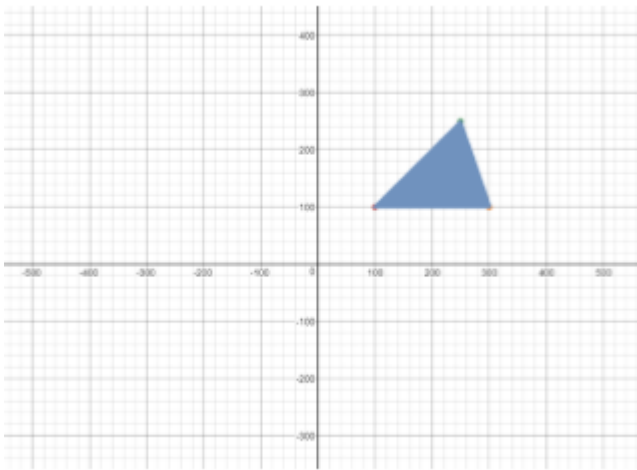
Berikutnya, program dapat menerima input yang didefinisikan pada tabel dibawah.

Input	Keterangan
translate <dx> <dy>	Melakukan translasi objek dengan menggeser nilai x sebesar dx dan menggeser nilai y sebesar dy .
dilate <k>	Melakukan dilatasi objek dengan faktor scaling k .
rotate <deg> <a> 	Melakukan rotasi objek secara berlawanan arah jarum jam sebesar deg derajat terhadap titik a,b .
reflect <param>	Melakukan pencerminan objek. Nilai <i>param</i> adalah salah satu dari nilainilai berikut: x , y , y=x , y=-x , atau (a,b) . Nilai (a,b) adalah titik untuk melakukan pencerminan terhadap.
shear <param> <k>	Melakukan operasi <i>shear</i> pada objek. Nilai <i>param</i> dapat berupa x (terhadap sumbu x) atau y (terhadap sumbu y). Nilai k adalah faktor <i>shear</i> .
stretch <param> <k>	Melakukan operasi <i>stretch</i> pada objek. Nilai <i>param</i> dapat berupa x

	(terhadap sumbu x) atau y (terhadap sumbu y). Nilai k adalah faktor <i>stretch</i> .
custom <a> <c> <d>	Melakukan transformasi linier pada objek dengan matriks transformasi sebagai berikut: $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$
multiple <n> ... // input 1 ... // input 2 // input n	Melakukan transformasi linier pada objek sebanyak n kali berurutan. Setiap baris input 1.. n dapat berupa <i>translate</i> , <i>rotate</i> , <i>shear</i> , dll tetapi bukan <i>multiple</i> , <i>reset</i> , <i>exit</i> .
reset	Mengembalikan objek pada kondisi awal objek didefinisikan.
exit	Keluar dari program.

Contoh I/O program :

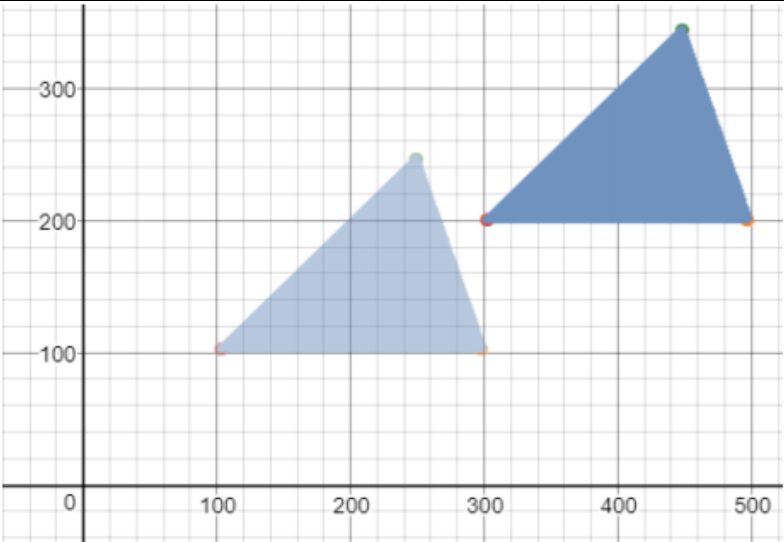
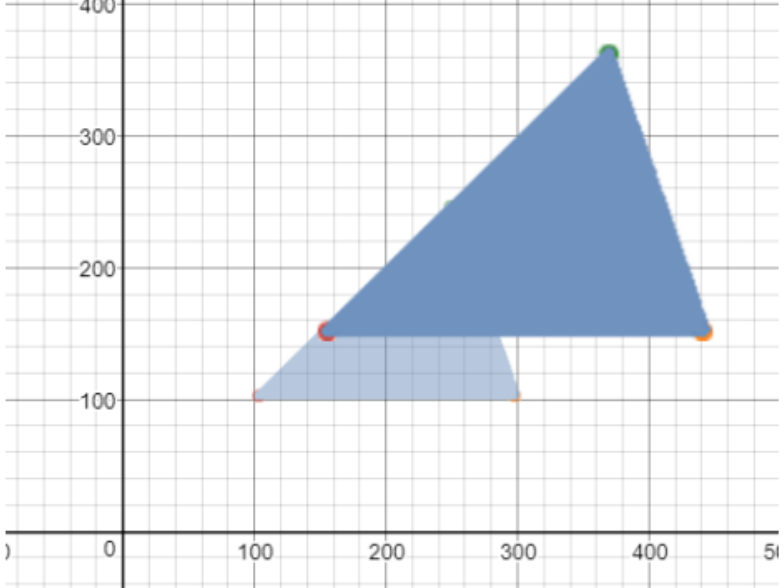
Saat program baru dimulai:

Input	Output
3 100,100 250,250 300,100	

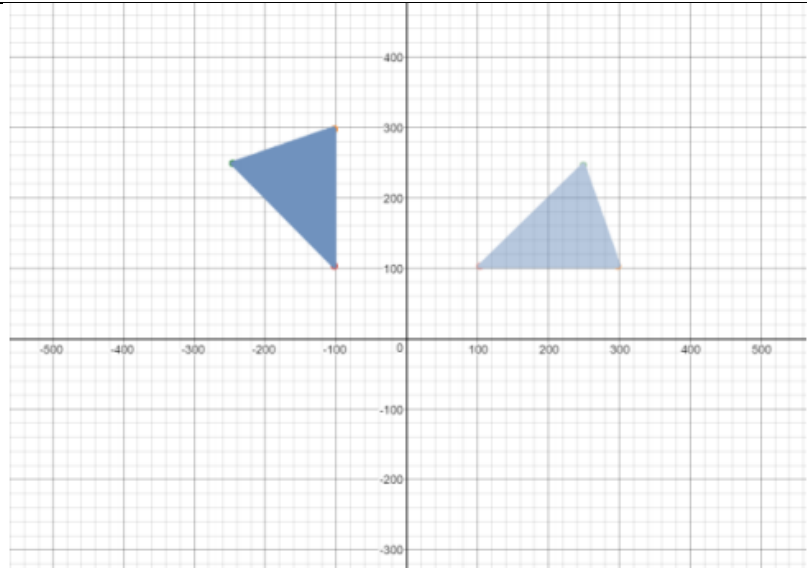
Perhatikan bahwa garis-garis tipis pada gambar diatas *tidak perlu diimplementasikan pada program*.

Saat program sudah membentuk objek dari input awal:

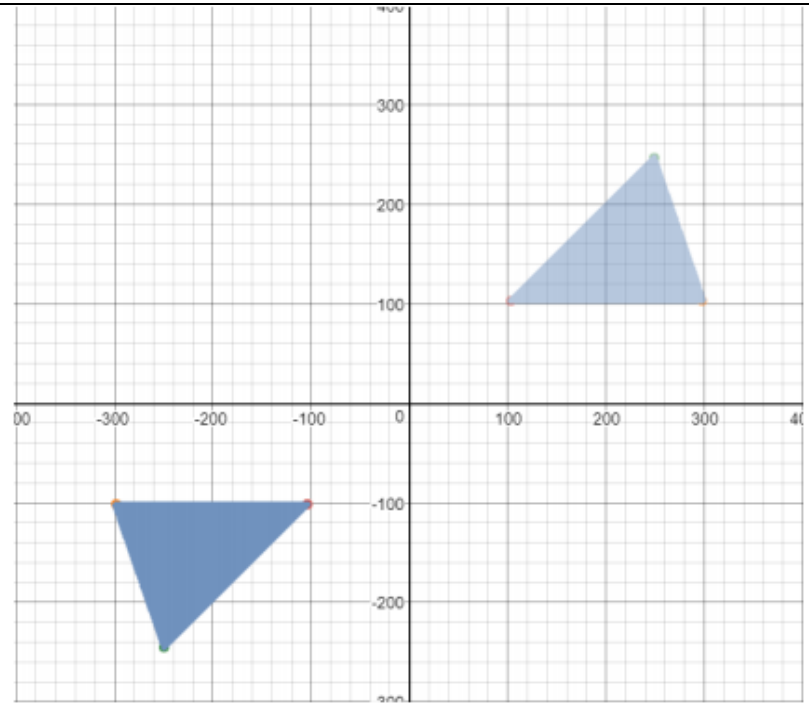
Catatan: Perhatikan bahwa gambar bidang yang transparan menunjukkan kondisi bidang *sebelum* input diberi, sedangkan bidang yang tidak transparan menunjukkan kondisi bidang *setelah* program mengeksekusi operasi dari input (bidang yang transparan *tidak ditampilkan* pada program).

Input	Output
translate 200 100	
dilate 1.5	

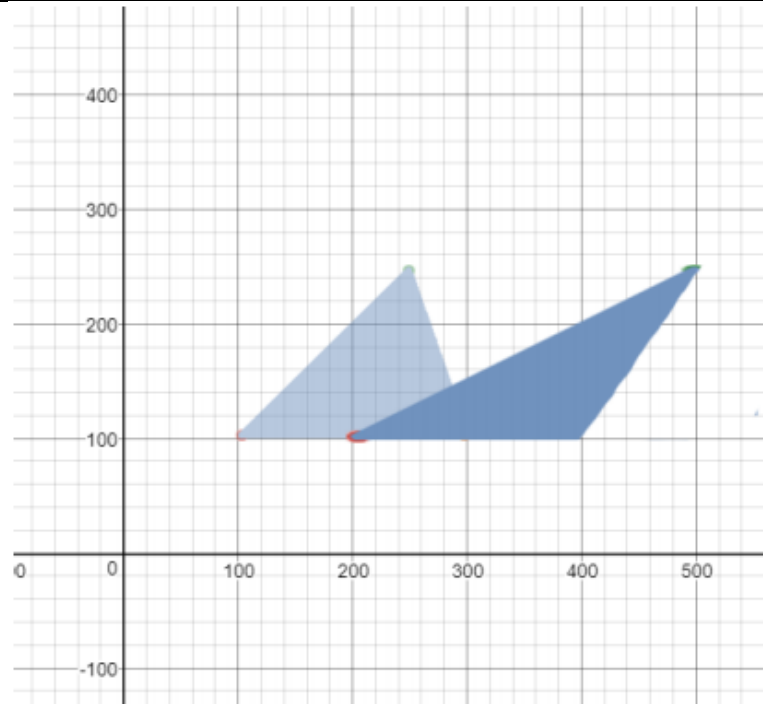
rotate 90 0 0



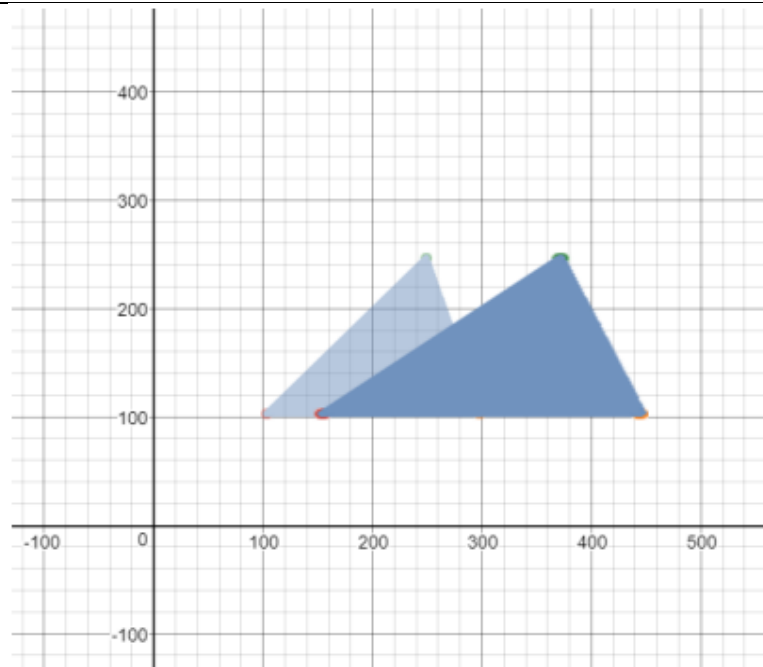
reflect (0,0)

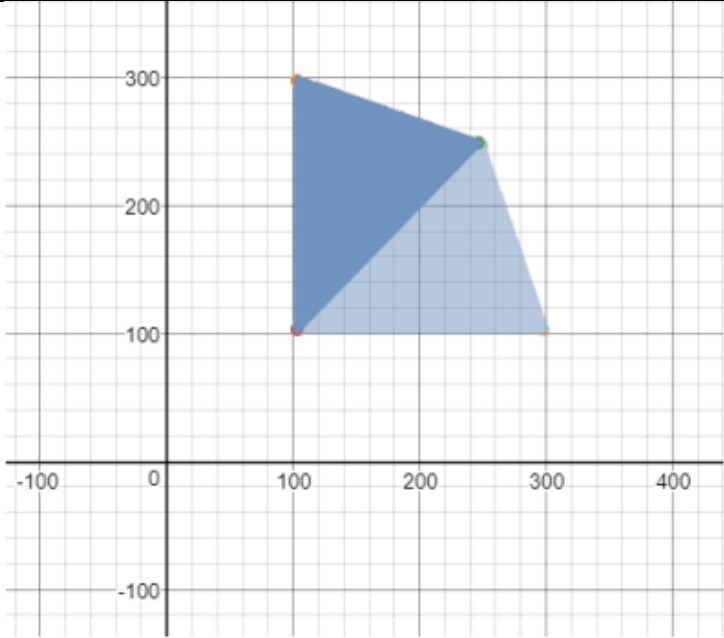



shear x 1



stretch x 1.5



custom 0 1 1 0	
reset	

Penjelasan singkat OpenGL API

Open Graphics Library (OpenGL) adalah API (*Application Programming Interface*) 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 GPU (*graphics processing unit*) untuk mencapai hasil *render* yang diakselerasi dengan *hardware*. Anda diharapkan untuk melakukan eksplorasi penggunaan *OpenGL*. Berikut adalah contoh kode program yang menggunakan library *OpenGL* :

Kode Program (khusus untuk objek 2D)	Keterangan
<pre>GLfloat triangleVertices[] = { 100, 100, 0, 300, 100, 0, 250, 250, 0 };</pre>	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.
<pre>GLFWwindow *window; window = glfwCreateWindow (600, 600, "MyWindowName", NULL, NULL);</pre>	Membuat sebuah window yang akan Anda gunakan untuk menampilkan output program
<pre>glEnableClientState(GL_VERTEX_ARRAY); glVertexPointer(3, GL_FLOAT, 0, triangleVertices); glDrawArrays(GL_POLYGON, 0, 3);</pre>	Menggambar sebuah poligon sesuai titik-titik yang sudah didefinisikan pada triangleVertices.
<pre>glDisableClientState(GL_VERTEX_ARRAY);</pre>	

Berikut adalah daftar pranala yang dapat membantu Anda untuk melakukan eksplorasi OpenGL:

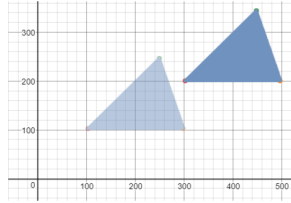
1. Tutorial OpenGL: <http://www.opengl-tutorial.org/>
2. Wiki: https://www.khronos.org/opengl/wiki/Getting_Started
3. Library: <https://www.opengl.org/sdk/libs/>
4. Wikipedia: <https://en.wikipedia.org/wiki/OpenGL>

Bab II. Teori Singkat

Pada penjelasan setiap fungsi yang dipakai, penjelasan akan diuraikan fungsi per fungsi. Perlu diberitahu bahwa setiap matriks yang kami gunakan merupakan matriks *general* agar transformasi dapat dilakukan di bidang 2 dimensi maupun 3 dimensi. Implementasi 2 dimensi yang kelompok kami gunakan merupakan implementasi 3 dimensi dengan $z = 0$.

1. Translasi (*Translate*)

Translasi merupakan transformasi perubahan objek dengan cara menggeser objek dari satu posisi ke posisi lainnya dengan jarak tertentu.



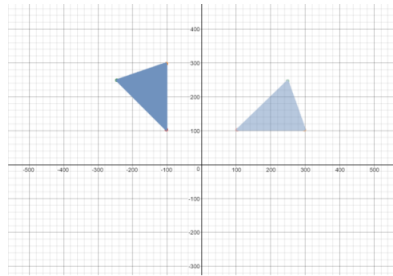
Pada umumnya, translasi yang dilakukan dengan matriks melakukan penjumlahan terhadap matriks yang sudah ada, tetapi untuk tugas besar kali ini, matriks yang kami pakai adalah matriks perkalian. Translasi titik-titik di ruang-2 dan ruang-3 memakai matriks tersebut. Matriks tersebut dapat digeneralisasi menjadi:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ dx & dy & dz & 1 \end{bmatrix}$$

Untuk transformasi 2 dimensi, dapat dilakukan transformasi dengan membuat $dz = 0$.

2. Rotasi (*Rotate*)

Rotasi atau perputaran merupakan perubahan kedudukan objek dengan cara diputar melalui pusat dan sudut tertentu. Besarnya rotasi dalam transformasi geometri sebesar α disepakati untuk arah yang berlawanan dengan arah jalan jarum jam. Jika arah perputaran rotasi suatu benda searah dengan jarum jam, maka sudut yang dibentuk adalah $-\alpha$.



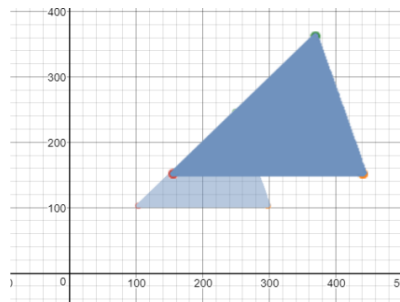
Adapula matriks transformasi yang dapat berlaku pada ruang 2 dimensi dan 3 dimensi. Matriks *general* tersebut adalah :

$$R(\alpha, a, b, c) = \begin{bmatrix} \cos(\alpha) + a^2(1 - \cos(\alpha)) & ab(1 - \cos(\alpha)) - c(\sin(\alpha)) & ac(1 - \cos(\alpha)) + b(\sin(\alpha)) & 0 \\ ab(1 - \cos(\alpha)) + c(\sin(\alpha)) & \cos(\alpha) + b^2(1 - \cos(\alpha)) & bc(1 - \cos(\alpha)) - a(\sin(\alpha)) & 0 \\ ac(1 - \cos(\alpha)) - b(\sin(\alpha)) & bc(1 - \cos(\alpha)) + a(\sin(\alpha)) & \cos(\alpha) + c^2(1 - \cos(\alpha)) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

jika sumbu putarnya merupakan titik (a, b, c) .

3. Dilatasi (*Dilate*)

Dilatasi disebut juga dengan perbesaran atau pengecilan suatu objek. Jika transformasi pada translasi, refleksi, dan rotasi hanya mengubah posisi benda, maka dilatasi melakukan transformasi geometri dengan merubah ukuran benda. Ukuran benda dapat menjadi lebih besar atau lebih kecil. Perubahan ini bergantung pada skala yang menjadi faktor pengalinya. Rumus dalam dilatasi ada dua, yang dibedakan berdasarkan pusatnya. Dilatasi tidak dapat dilakukan apabila parameter dibawah 0.



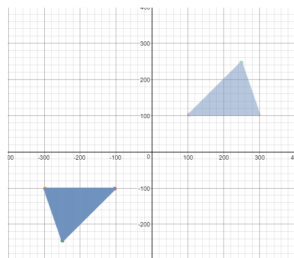
Pada algoritma transformasi dilatasi yang kami buat, kami juga memakai matriks. Matriks tersebut adalah :

$$\begin{bmatrix} k & 0 & 0 & 0 \\ 0 & k & 0 & 0 \\ 0 & 0 & k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

untuk dilatasi dengan skalar k . Pada penggunaan 2 dimensi, digunakan matriks serupa.

4. Refleksi (*Reflect*)

Refleksi adalah pencerminan yang memindahkan semua titik selayaknya dengan menggunakan cermin datar.



Pada implementasi program yang kelompok kami buat, kami menggunakan matriks transformasi terhadap garis x , y , z , $x=y$, $x=-y$, dan (a,b) dimana a dan b merepresentasikan poin pusat refleksi untuk implementasi 2 dimensi, dan garis xy , xz , dan yz untuk implementasi 3 dimensi.

Untuk 2 dimensi : $R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ $R_y = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

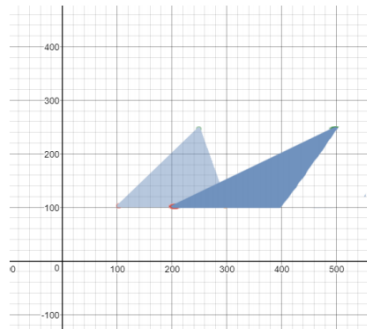
$$R_{(x=y)} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_{(x=-y)} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_{(a,b)} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 2a & 2b & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Untuk 3 dimensi : $R_{xy} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ $R_{xz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

$$R_{yz} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5. Menggeser (*Shear*)

Shear adalah pergeseran pada suatu sistem dengan terjadinya perubahan bentuk. Umumnya, *shear* pada grafika komputer digunakan untuk merubah sudut pandang berbeda untuk melihat suatu benda.



Pada implementasi program yang kelompok kami buat, pada implementasi 2 dimensi, transformasi *shear* dapat dilakukan terhadap sumbu x dan terhadap sumbu y sebesar konstanta skalar k dengan matriks :

$$Sh_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ k & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad Sh_y = \begin{bmatrix} 1 & k & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

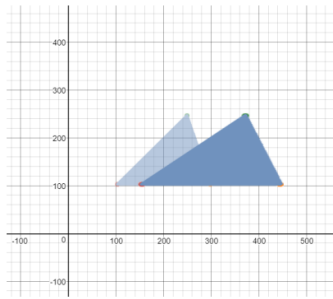
Pada implementasi 3 dimensi, transformasi *shear* dapat dilakukan dengan pendekatan xy , xz , yx , yz , zx , dan zy . Untuk setiap perhitungan, misalnya, apabila pendekatan yang digunakan

adalah pendekatan xy , parameter h_{xy} akan bernilai k dan sisanya bernilai 0, dan begitu seterusnya untuk pendekatan-pendekatan yang lain. Secara *general* matriks akan berbentuk :

$$S = \begin{bmatrix} 1 & h_{yx} & h_{zx} & 0 \\ h_{xy} & 1 & h_{zy} & 0 \\ h_{xz} & h_{yz} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

6. Meregang (*Stretch*)

Peregangan atau *stretch* adalah transformasi linier yang memperbesar atau memperkecil objek yang sama ke segala arahnya. Parameter *stretch* selalu lebih dari 0.



Pada implementasi program kelompok kami, mirip seperti *shear*, pada implementasi 2 dimensi, transformasi *stretch* dapat dilakukan terhadap sumbu x dan terhadap sumbu y sebesar konstanta skalar k dengan matriks :

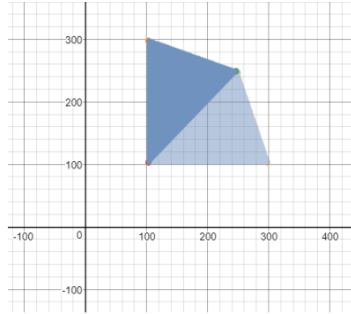
$$St_x = \begin{bmatrix} k & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad St_y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & k & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Pada implementasi 3 dimensi, transformasi *shear* dapat dilakukan terhadap sumbu x , sumbu y , dan sumbu z sebesar konstanta skalar k dengan matriks :

$$St_x = \begin{bmatrix} k & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad St_y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & k & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad St_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

7. Matriks menurut input pengguna (*Custom*)

Perintah *custom* diimplementasikan dengan masukan 4 buah angka untuk 2 dimensi dan 9 angka untuk 3 dimensi.



Pada perintah ini, 4/9 masukan yang ada setelah kata ‘*custom*’ adalah matriks transformasi terhadap bidang yang ada.

Untuk implementasi 2 dimensi, *custom a b c d* akan menghasilkan matriks

$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ dan mengalikannya dengan titik-titik bidang yang sudah ada.

Untuk implementasi 3 dimensi, *custom a b c d e f g h i* akan menghasilkan matriks

$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$ dan mengalikannya dengan titik-titik bidang yang sudah ada.

8. Multi-perintah (*Multiple*)

Perintah *multiple* adalah perintah untuk menjalankan berbagai jenis transformasi terhadap bidang awal. Perintah yang dapat dilakukan adalah semua perintah kecuali *multiple*, *reset*, dan *exit*.

Pada implementasi program kelompok kami, *exit* dapat digunakan dalam *multiple* sebagai perintah untuk keluar dari *multiple* itu sendiri, perintah ‘*exitmultiple*’ juga melakukan hal yang sama. Apabila pengguna melakukan *exitmultiple*, semua transformasi yang dilakukan dalam *multiple* akan dibatalkan.

Perlu diuraikan bahwa pada implementasi program kami, animasi transformasi terjadi saat program telah keluar dari *multiple* itu sendiri. Animasi akan dilakukan satu-per-satu sampai seluruh transformasi dilakukan.

9. OpenGL

OpenGL adalah salah satu *library* di bahasa pemrograman (pada kasus ini, bahasa Python). *Open Graphics Library (OpenGL)* adalah API (*Application Programming Interface*) 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 GPU (*graphics processing unit*) untuk mencapai hasil *render* yang diakselerasi dengan *hardware*.

Bab III. Implementasi Program

Program utama yang di-*compile* dan di-*run* adalah program *main.py*. program *main.py* adalah program yang membuat *window GUI* untuk menampilkan bidang pada ruang untuk ditransformasi. Pada saat program dimulai, Pengguna akan diminta masukan untuk melakukan manipulasi pada ruang 2 dimensi atau 3 dimensi. *Window GUI* yang dibuat bergantung pada input pengguna tersebut. Berikut adalah program *main.py*.

```
from render import draw
from bentuk import objek
import handler
from OpenGL.GLUT import glutInit, glutInitDisplayMode, glutInitWindowSize, glutCreateWindow
from OpenGL.GLUT import glutInitWindowPosition, glutKeyboardFunc, glutDisplayFunc, glutIdleFunc, glutMainLoop
from OpenGL.GLUT import GLUT_RGBA, GLUT_DOUBLE, GLUT_ALPHA, GLUT_DEPTH
from config import config
from transformasi import *
from ProcInput import ProcInput

import threading

if(__name__ == "__main__"):
    stis3D = input("Apakah anda ingin melakukan transformasi 3D?(Y = Yes, N = No) ")
    is3D = (stis3D == "Y") or (stis3D == "y")
    config.initAwal(is3D)
    #MultiThreading
    procInput = threading.Thread(target=ProcInput,args=(config.is3D,))
    procInput.start()
    #GUI
    glutInit()
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_ALPHA | GLUT_DEPTH)
    glutInitWindowSize(config.width, config.height)
    glutInitWindowPosition(0, 0)
    window = glutCreateWindow("Tubes Algeo 2")
    #Event Handler & Draw function
    glutKeyboardFunc(handler.on_keyPressed)
    glutDisplayFunc(draw)
    glutIdleFunc(draw)
    glutMainLoop()
```

Pada implementasi yang kelompok kami buat, transformasi yang kami lakukan menggunakan dasar perkalian matriks. Perkalian matriks dilakukan dengan implementasi matriks baris dikalikan dengan matriks transformasi. Contoh :

Titik $\langle a, b, c \rangle$ dikalikan dengan matriks $\begin{bmatrix} d & e & f \\ g & h & i \\ j & k & l \end{bmatrix}$ untuk menghasilkan titik baru berupa

$\langle (ad + bg + cj), (ae + bh + ck), (af + bi + cl) \rangle$. Matriks-matriks yang dipakai merupakan matriks keluaran dari fungsi-fungsi yang ada pada file *transformasi.py*. Berikut adalah program *transformasi.py*.

```

import numpy as np
import math

def translate(dx, dy, dz=0):
    return np.mat([[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [dx, dy, dz, 1]])

def rotasi(deg, a=0, b=0, c=0): # degree berlawanan arah jarum jam terhadap titik(a,b)
    # normalisasi vektor(a,b,c)
    panjang = np.sqrt(a*a+b*b+c*c)
    if(panjang != 0):
        a /= panjang
        b /= panjang
        c /= panjang
    # normalisasi sudut
    deg = np.deg2rad(deg)
    cosDeg = math.cos(deg)
    sinDeg = math.sin(deg)
    baris1 = [cosDeg+a*(1-cosDeg), a*b*(1-cosDeg)-c * sinDeg,
              a*c*(1-cosDeg)+b*sinDeg, 0]
    baris2 = [b*a*(1-cosDeg)+c*sinDeg, cosDeg+b*b * (1-cosDeg),
              b*c*(1-cosDeg)-a*sinDeg, 0]
    baris3 = [c*a*(1-cosDeg)-b*sinDeg, c*b*(1-cosDeg) + a*sinDeg,
              cosDeg+c*c*(1-cosDeg), 0]
    baris4 = [0, 0, 0, 1]
    return np.mat([baris1, baris2, baris3, baris4])
    # return np.mat([math.cos(deg), math.sin(deg), 0],
    #               [-1*math.sin(deg), math.cos(deg), 0],
    #               [-1*a*math.cos(deg)+b*math.sin(deg)+a, -1*a*math.sin(deg)-b*math.cos(deg)+a, 1])

# asumsi axis Z positif arah keluar layar, negatif masuk ke layar
# rotasi 3D dilakukan ini selalu counter-clockwise

```

```

def rotasiX(deg):
    # 1 0 0
    # 0 cos sin
    # 0 -sin cos
    # 0 0 1
    deg = np.deg2rad(deg)
    return np.mat([[1, 0, 0, 0],
                  [0, math.cos(deg), math.sin(deg), 0],
                  [0, -1*math.sin(deg), math.cos(deg), 0],
                  [0, 0, 0, 1]])

def rotasiY(deg):
    # cos 0 sin
    # 0 1 0
    # -sin 0 cos
    # 0 0 1
    deg = np.deg2rad(deg)
    return np.mat([[math.cos(deg), 0, math.sin(deg), 0],
                  [0, 1, 0, 0],
                  [-1*math.sin(deg), 0, math.cos(deg), 0],
                  [0, 0, 0, 1]])

def rotasiZ(deg):
    # cos sin 0
    # -sin cos 0
    # 0 0 1
    # 0 0 1
    deg = np.deg2rad(deg)
    return np.mat([[math.cos(deg), math.sin(deg), 0, 0],
                  [-1*math.sin(deg), math.cos(deg), 0, 0],
                  [0, 0, 1, 0],
                  [0, 0, 0, 1]])

```

```

def dilate(k):
    return np.mat([[k, 0, 0, 0],
                  [0, k, 0, 0],
                  [0, 0, k, 0],
                  [0, 0, 0, 1]])

def reflect(a, b=0): # a bisa sebagai string atau titik
    if (a == 'x'):
        return np.mat([[1, 0, 0, 0],
                        [0, -1, 0, 0],
                        [0, 0, 1, 0],
                        [0, 0, 0, 1]])
    elif (a == 'y'):
        return np.mat([[1, 0, 0, 0],
                        [0, 1, 0, 0],
                        [0, 0, 1, 0],
                        [0, 0, 0, 1]])
    elif (a == 'yex'):
        return np.mat([[0, 1, 0, 0],
                        [1, 0, 0, 0],
                        [0, 0, 1, 0],
                        [0, 0, 0, 1]])
    elif (a == 'yexx'):
        return np.mat([[0, -1, 0, 0],
                        [-1, 0, 0, 0],
                        [0, 0, 1, 0],
                        [0, 0, 0, 1]])
    else: # point (a,b)
        return np.mat([[1, 0, 0, 0],
                        [0, -1, 0, 0],
                        [2*a, 2*b, 1, 0],
                        [0, 0, 0, 1]])

```



```

def reflect3D(s):
    if (s == 'xz'):
        return np.mat([[1, 0, 0, 0],
                        [0, -1, 0, 0],
                        [0, 0, 1, 0],
                        [0, 0, 0, 1]])

    elif (s == 'yz'):
        return np.mat([[-1, 0, 0, 0],
                        [0, 1, 0, 0],
                        [0, 0, 1, 0],
                        [0, 0, 0, 1]])

    elif (s == 'xy'):
        return np.mat([[1, 0, 0, 0],
                        [0, 1, 0, 0],
                        [0, 0, -1, 0],
                        [0, 0, 0, 1]])

def shear(param, k):
    if(param == 'x'):
        return np.mat([[1, 0, 0, 0],
                        [k, 1, 0, 0],
                        [0, 0, 1, 0],
                        [0, 0, 0, 1]])

    else: # y
        return np.mat([[1, k, 0, 0],
                        [0, 1, 0, 0],
                        [0, 0, 1, 0],
                        [0, 0, 0, 1]])

def shear3D(param, k):
    hXY, hXZ, hYX, hYZ, hZX, hZY = [0 for i in range(6)]
    if(param == 'xy'):

```

```

        if(param == 'xy'):
            hXY = k
        elif(param == 'xz'):
            hXZ = k
        elif(param == 'yx'):
            hYX = k
        elif(param == 'yz'):
            hYZ = k
        elif(param == 'zx'):
            hZX = k
        elif(param == 'zy'):
            hZY = k
        return np.mat([[1, hYX, hZX, 0],
                        [hXY, 1, hZY, 0],
                        [hXZ, hYZ, 1, 0],
                        [0, 0, 0, 1]])

def stretch(param, k):
    if(param == 'x'):
        return np.mat([[k, 0, 0, 0],
                        [0, 1, 0, 0],
                        [0, 0, 1, 0],
                        [0, 0, 0, 1]])

    else: # y
        return np.mat([[1, 0, 0, 0],
                        [0, k, 0, 0],
                        [0, 0, 1, 0],
                        [0, 0, 0, 1]])

def stretch3D(param, k):
    if(param == 'x'): # horizontal
        return np.mat([[k, 0, 0, 0],

```

```

                        [0, 1, 0, 0],
                        [0, 0, 1, 0],
                        [0, 0, 0, 1]])
    elif (param == 'y'): # vertical
        return np.mat([[1, 0, 0, 0],
                        [0, k, 0, 0],
                        [0, 0, 1, 0],
                        [0, 0, 0, 1]])

    elif (param == 'z'):
        return np.mat([[1, 0, 0, 0],
                        [0, 1, 0, 0],
                        [0, 0, k, 0],
                        [0, 0, 0, 1]])

def custom(a, b, c, d):
    return np.mat([[a, b, 0, 0],
                    [c, d, 0, 0],
                    [0, 0, 1, 0],
                    [0, 0, 0, 1]])

def custom3D(a, b, c, d, e, f, g, h, i):
    return np.mat([[a, b, c, 0],
                    [d, e, f, 0],
                    [g, h, i, 0],
                    [0, 0, 0, 1]])

def transformVertex(titik, matriks):
    return np.mat(titik) * np.mat(matriks)

```

```

def transformPolygon(listOfVertex, matriks):
    newList = []
    for v in listOfVertex:
        newList.append(transformVertex(v, matriks))
    return newList

```

Untuk menerima input sesuai dengan spesifikasi tugas besar yang ada, kami memiliki 1 file sendiri yang berisi prosedur untuk menerima input. Pada *ProcInput.py*, terdapat 2 buah prosedur utama, prosedur yang menerima *single input* dan *multiple input*. Berikut adalah program *ProcInput.py*.

```

from config import config
from transformasi import *
from OpenGL.GLUT import glutLeaveMainLoop
maxKeyFrame = config.maxKeyFrame

def SingleInput(s, is3D):
    if (s[0] == "translate"):
        if (len(s) != 3 and not(is3D)):
            if (len(s) == 4):
                print("Input dx dy dz hanya bisa untuk 3D")
            else:
                print("Input salah!")
        elif (len(s) != 4 and is3D):
            if (len(s) == 3):
                print("Input dx dy hanya bisa untuk 2D")
            else:
                print("Input salah!")
        else:
            dx, dy = [float(s[i]) for i in range(1, 3)]
            if (is3D):
                dz = float(s[3])
                print("translasi sebanyak (" + str(dx) + ", " + str(dy) + ", " + str(dz) + ")")
                config.objTest.animator.pushAnimasi([translate(dx/maxKeyFrame, dy/maxKeyFrame, dz/maxKeyFrame), False])
            else:
                print("translasi sebanyak (" + str(dx) + ", " + str(dy) + ")")
                config.objTest.animator.pushAnimasi([translate(dx/maxKeyFrame, dy/maxKeyFrame, 0), False])
        # fungsi prosedur translate
    elif (s[0] == "rotate"):
        if (len(s) != 4 and not(is3D)):
            if (len(s) == 5):
                print("Input degree <x, y, z> hanya bisa untuk 3D")
            else:
                print("Input salah!")
        else:
            degree, dx, dy = [float(s[i]) for i in range(1, 4)]
            if (is3D):
                dz = float(s[4])
                print("rotasi sebesar " + str(degree) + " derajat dengan sumbu putar vektor (" + str(dx) + ", " + str(dy) + ", " + str(dz) + ")")
                config.objTest.animator.pushAnimasi([rotasi(-degree/maxKeyFrame, dx, dy, dz), False])
            else:
                print("rotasi sebesar " + str(degree) + " derajat dari pusat (" + str(dx) + ", " + str(dy) + ")")
                config.objTest.animator.pushAnimasi([translate(-dx, -dy) * rotasi(-degree/maxKeyFrame, 0, 0, 1) * translate(dx, dy), False])
        # fungsi prosedur rotate
    elif (s[0] == "dilate"):
        if (len(s) != 2):
            print("Input salah!")
        else:
            ratio = float(s[1])
            if (ratio > 0):
                print("dilatasi dengan rasio " + str(ratio) + ":1")
                config.objTest.animator.pushAnimasi([dilate(ratio*(1/maxKeyFrame)), False])
            else:
                print("Input salah! (rasio harus > 0)")
    elif (s[0] == "reflect"):
        if (len(s) != 2 and len(s) != 3):
            print("Input salah!")
        else:
            param = s[1]
            if (len(s) == 3):
                # Refleksi terhadap titik (a,b)
                param = int(param)
                param2 = int(s[2])
                print("refleksi berdasarkan titik (" + str(s[1]) + ", " + str(s[2]) + ")")
                config.objTest.animator.pushAnimasi([reflect(param, param2, True)])
            else:
                print("refleksi berdasarkan garis " + param)
                if (is3D):
                    config.objTest.animator.pushAnimasi([reflect3D(param), True])
                else:
                    config.objTest.animator.pushAnimasi([reflect(param), True])
        # fungsi prosedur reflect
    elif (s[0] == "shear"):
        if (len(s) != 3):
            print("Input salah!")
        else:
            param = s[1]
            k = float(s[2])
            print("shear dgn parameter " + param + " dengan konstanta " + str(k))
            if (config.is3D):
                config.objTest.animator.pushAnimasi([shear3D(param, k/maxKeyFrame), False])
            else:
                config.objTest.animator.pushAnimasi([shear(param, k/maxKeyFrame), False])
    elif (s[0] == "stretch"):
        if (len(s) != 3):
            print("Input salah!")
        else:
            param = str(s[1])
            k = float(s[2])
            if (k > 0):
                print("stretch dgn parameter " + param + " dengan konstanta " + str(k))
                if (config.is3D):
                    config.objTest.animator.pushAnimasi([stretch3D(param, k*(1/maxKeyFrame)), False])

```

```

else:
    config.objTest.Animator.pushAnimasi([stretch(param,k**(1/maxKeyFrame)),False])
else:
    print("Input salah! konstanta harus >= 0")
elif (s[0] == "custom"):
    if (len(s) != 5 and not(is30)):
        if (len(s) == 10):
            print("Input Matrix 3x3 hanya bisa untuk 30")
        else:
            print("Input salah!")
    elif (len(s) != 10 and is30):
        if (len(s) == 5):
            print("Input Matrix 2x2 hanya bisa untuk 20")
        else:
            print("Input salah!")
else:
    a, b, c, d = [float(s[i]) for i in range(1, 5)]
    if (is30):
        e, f, g, h, i = [float(s[i]) for i in range(5, 10)]
        print("transformasi custom dengan matrix :")
        print(a, b, c)
        print(d, e, f)
        print(g, h, i)
        config.objTest.Animator.pushAnimasi([custom3D(a,b,c,d,e,f,g,h,i),True])
    else:
        print("transformasi custom dengan matrix :")
        print(a, b)
        print(c, d)
        config.objTest.Animator.pushAnimasi([custom(a,b,c,d),True])
# fungsi prosedur custom
elif (s[0] == "reset"):
    config.curMinX, config.curMaxX, config.curMinY, config.curMaxY, config.curMinZ, config.curMaxZ = - \
    10, 10, -10, 10, -10, 10
    config.reset()

```

```

else:
    print("Input salah! Coba Lagi")
# endif
# end SingleInput procedure

def ProcInput(is30):
    if(not(is30)):
        #Baca koordinat 2d
        banyakKoordinat = int(input("Banyak koordinat 2D : "))
        listVertexInput = []
        for i in range(banyakKoordinat):
            inp = input("Koordinat "+str(i)+" : ").split(" ")
            listVertexInput.append(np.array([float(inp[0]),float(inp[1]),0,1]))
        config.initAwal(is30,listVertexInput,False)
    s = input("$ ").split(" ")
    while (s[0] != "exit"):
        if (s[0] == "multiple"):
            if (len(s) != 2):
                print("Input salah!")
            else:
                listCommand = []
                lanjut = True
                i = 0
                while(i < int(s[1]) and lanjut):
                    print("Perintah ke-"+str(i+1))
                    s2 = input("$ ----- ").split(" ")
                    if ((s2[0] != "exit") and (s2[0] != "exitmultiple")):
                        if(s2[0] != "reset"):
                            listCommand.append(s2)
                            i += 1
                        else:
                            print("Tidak bisa melakukan reset didalan multiple")

```

```

else:
    lanjut = False
# endif
# endforLoop
if ((s2[0] != "exit") and (s2[0] != "exitmultiple")):
    for command in listCommand:
        SingleInput(command,is30)
    print("Keluar dari perintah multiple")
# endif
else:
    SingleInput(s, is30)
# endif
s = input("$ ").split(" ")
# endwhile
config.jalan = False
glutLeaveMainLoop()
# end ProcInput procedure

```

Karena variabel dan konstanta yang dipakai cukup banyak, kami memiliki 1 program sendiri yang dikhususkan untuk menyimpan variabel-variabel tersebut agar dapat di-pass ke file .py lainnya. Berikut adalah program *config.py*

```
maxKey = 30
import numpy as np
import bentuk
from random import randint
from transformasi import rotasi, rotasiY, dilate
# Setting program
class config:
    # Static Variable
    # ukuran GUI
    width, height = 1000, 1000
    # Program Jalan
    jalan = True
    # nilai koordinat minimal X,Y,Z
    minX, maxX = -int(width/2), int(width/2)
    minY, maxY = minX, maxX
    minZ, maxZ = minX, maxX
    # 2D / 3D
    is3D = True
    # Nilai awal orthografi
    curMinX, curMaxX, curMinY, curMaxY, curMinZ, curMaxZ = -10, 10, -10, 10, -10, 10
    # Nilai camera
    camX2D, camY2D, camZ2D = 0.0, 0.0, 5.0
    camX3D, camY3D, camZ3D = 5.0, 5.0, 5.0
    # Vector Camera
    vecCam3D = np.mat([[camX3D, camY3D, camZ3D, 1]])
    # Tegaklurus = Proyeksi Vector Cam lalu diputar 90 derajat counterclockwise
    tegaklurusCam3D = np.mat([[camX3D, 0, camZ3D, 1]]) * rotasiY(90)
    # Nilai warna grid
    xy, xz, yz = [[255 for i in range(3)] for i in range(3)] * [randint(0, 255)/255 for color in range(3)] for i in range(3)]
    # Ukuran grid
    banyakGrid = 40
    # zoom
    navX, navY, navZ, navZoom = 1, 1, 1, 1.1
    navPutar = 5

def main():
    maxKeyframe = maxKey
    # default object
    objTest = bentuk.objek(is3D)
    listOFVertex = []
    default = True
    def initAwal(is3D, listOFVertex=[], default=True):
        config.is3D = is3D
        config.listOFVertex = listOFVertex
        config.default = default
        config.objTest = bentuk.objek(is3D, listOFVertex, default)
    def reset():
        config.vecCam3D = np.mat([[config.camX3D, config.camY3D, config.camZ3D, 1]])
        config.tegaklurusCam3D = np.mat([[config.camX3D, 0, config.camZ3D, 1]]) * rotasiY(90)
        config.objTest = bentuk.objek(config.is3D, config.listOFVertex, config.default)
    def putarCamY(derajat):
        # memutar vector kamera 3D dengan sumbu Y positif sebagai pusat
        config.vecCam3D *= rotasiY(derajat)
        config.tegaklurusCam3D *= rotasiY(derajat)
    def putarCam(derajat, vx, vy, vz):
        config.vecCam3D *= rotasi(derajat, vx, vy, vz)
    def zoomCam(k):
        config.vecCam3D *= dilate(k, config.is3D)
```

Karena bidang dispesifikasikan untuk digambar pada *window GUI*, kami memiliki 1 file tersendiri yang berisi fungsi-fungsi dan prosedur-prosedur yang digunakan untuk menggambar pada *window GUI* yang telah dibuka pada *main.py*. Berikut adalah program *render.py*.

```
from OpenGL.GL import *
from OpenGL.GLU import *
from OpenGL.GLUT import *
import numpy as np
from config import config
from transformasi import *
from ctypes import *

def refresh2d():
    glViewport(0, 0, config.width, config.height)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glOrtho(config.curMinX, config.curMaxX,
            config.curMinY, config.curMaxY, -10, 10)
    glMatrixMode(GL_MODELVIEW)
    gluLookAt(config.camX2D, config.camY2D, config.camZ2D, 0, 0, 0, 0, 1, 0)

def refresh3d():
    glViewport(0, 0, config.width, config.height)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glOrtho(config.curMinX, config.curMaxX,
            config.curMinY, config.curMaxY, -100, 100)
    glMatrixMode(GL_MODELVIEW)
    gluLookAt(config.vecCam3D.item(0), config.vecCam3D.item(
        1), config.vecCam3D.item(2), 0, 0, 0, 0, 1, 0)

def garis(startX, startY, startZ, finishX, finishY, finishZ):
    glBegin(GL_LINES)
    glVertex3f(startX, startY, startZ)
    glVertex3f(finishX, finishY, finishZ)
    glEnd()
```

```
def gambarSumbu(is3D):
    if(is3D):
        gllineWidth(4)
        # Sumbu Negatif
        glColor3f(1, 1, 1)
        garis(config.curMinX, 0, 0, 0, 0)
        garis(0, config.curMinY, 0, 0, 0)
        garis(0, 0, config.curMinZ, 0, 0)
        # Sumbu Positif
        glColor3f(1, 0, 0)
        garis(0, 0, 0, config.curMaxX, 0, 0)
        glColor3f(0, 1, 0)
        garis(0, 0, 0, 0, config.curMaxY, 0)
        glColor3f(0, 0, 1)
        garis(0, 0, 0, 0, 0, config.curMaxZ)
        gllineWidth(1)
    else:
        gllineWidth(4)
        glColor3f(1, 1, 1)
        garis(config.curMinX, 0, 0, config.curMaxX, 0, 0)
        garis(0, config.curMinY, 0, 0, config.curMaxY, 0)
        gllineWidth(1)

def gambarGrid(is3D):
    xy, xz, yz = config.xy, config.xz, config.yz
    if(is3D):
        dx = config.curMaxX-config.curMinX
        dy = config.curMaxY-config.curMinY
        dz = config.curMaxZ-config.curMinZ
        gridX, gridY, gridZ = np.ceil(dx/10), np.ceil(dy/10), np.ceil(dz/10)

        for i in range(6):
            # XY
```

```

...
glColor3f(xy[0],xy[1],xy[2])
glaris(config.curMinX, i*gridV, 0, config.curMaxX, i*gridV, 0)
glaris(config.curMinX, -i*gridV, 0, config.curMaxX, -i*gridV, 0)
...

# XZ
...
glColor3f(xz[0], xz[1], xz[2])
glaris(config.curMinX, 0, i*gridZ, config.curMaxX, 0, i*gridZ)
glaris(config.curMinX, 0, -i*gridZ, config.curMaxX, 0, -i*gridZ)
for i in range(6):
    xV
    ...
    glColor3f(xy[0],xy[1],xy[2])
    glaris(i*gridX, config.curMinY, 0, i*gridX, config.curMaxY, 0)
    glaris(-i*gridX, config.curMinY, 0, -i*gridX, config.curMaxY, 0)
    ...

# YZ
...
glColor3f(yz[0],yz[1],yz[2])
glaris(0, config.curMinY, i*gridZ, 0, config.curMaxY, i*gridZ)
glaris(0, config.curMinY, -i*gridZ, 0, config.curMaxY, -i*gridZ)
...
for i in range(6):
    # XZ
    glColor3f(xz[0], xz[1], xz[2])
    glaris(i*gridX, 0, config.curMinZ, i*gridX, 0, config.curMaxZ)
    glaris(-i*gridX, 0, config.curMinZ, -i*gridX, 0, config.curMaxZ)

    # YZ
    ...
    glColor3f(yz[0],yz[1],yz[2])
    glaris(0, i*gridY, config.curMinZ, 0, i*gridY, config.curMaxZ)

```

```
refresh2d()

config.objTest.gambar()
gambarGrid(config,1830)
gambarSumbu(config,1830)
#gambar_koordinat
gambarText(0,0,"Koordinat Vektor :")
y = 12
for i in config.objTest.listVertex:
    teks = "Koordinat ke-"+str(int(y/12))+": " +str(bulat(1.item(0),3))+", "+str(bulat(1.item(3),3))+", "+str(bulat(1.item(2),3))+","
    gambarText(0,y,teks)
    y += 12
# Sampai sini
glutSwapBuffers()
```

Meskipun untuk menggambar pada *window GUI* difungsikan di *render.py*, bentuk-bentuk bidang yang lain seperti *polygon* dan warna objek pada gambar diatur pada file tersendiri. Berikut adalah program *bentuk.py*.

```
import numpy as np
from animasi import animasi
import OpenGL.GL as gl
from random import randint

def gambarPolygon(listOfVertex, is3D, warnaKubus=[], listOfTriangle=[]):
    gl.glColor3f(warnaKubus[0][0], warnaKubus[0][1], warnaKubus[0][2])
    if(is3D):
        # Handle untuk 3D
        index = 0
        for triangle in listOfTriangle:
            r,g,b = warnaKubus[index][0], warnaKubus[index][1], warnaKubus[index][2]
            gl.glColor3f(r,g,b)
            gl.glBegin(gl.GL_POLYGON)
            for v in triangle:
                gl.glVertex3f(listOfVertex[v].item(0), listOfVertex[v].item(1), listOfVertex[v].item(2))
            gl.glEnd()
            index += 1
        else:
            # Handle untuk 2D
            gl.glBegin(gl.GL_POLYGON)
            for v in listOfVertex:
                gl.glVertex2f(v.item(0), v.item(1))
            gl.glEnd()

class objek:
    # Default Value
    listVertex2D = [np.mat([-1, -1, 0, 1]), np.mat([1, -1, 0, 1]),
                    np.mat([1, 1, 0, 1]), np.mat([-1, 1, 0, 1])]
    listVertex3D = [np.mat([-1, -1, 1, 1]), np.mat([1, -1, 1, 1]), np.mat([1, -1, -1, 1]), np.mat([-1, -1, -1, 1]),
                    np.mat([-1, 1, -1, 1]), np.mat([1, 1, -1, 1]), np.mat([1, 1, 1, 1]), np.mat([-1, 1, 1, 1])]
    listSegitiga3D = [(0,1,2), (0,2,3), (0,1,6), (0,5,6), (1,2,7), (1,6,7), (2,7,4), (2,3,4), (0,3,4), (0,5,4), (4,5,6), (4,6,7)]

    def __init__(self, is3D=False, listVertex=[], default=True):
        self.is3D = is3D
        if(default):
            if(is3D):
                self.listVertex = objek.listVertex3D
            else:
                self.listVertex = objek.listVertex2D
        else:
            self.listVertex = listVertex
        self.animasi = animasi()
        #Generate warna
        self.warnaKubus = []
        for i in range(len(self.listSegitiga3D)):
            r,g,b = randint(0,255)/255, randint(0,255)/255, randint(0,255)/255
            temp = [r,g,b]
            self.warnaKubus.append(temp)

    def gambar(self):
        self.listVertex = self.animasi.animate(self.listVertex)
        gambarPolygon(self.listVertex, self.is3D, self.warnaKubus, self.listSegitiga3D)
```

Pada implementasi program kami, saat *window GUI* dijalankan, kamera dapat digerakkan ke kanan(d), ke kiri(a), ke atas(w), dan ke bawah(s). (untuk implementasi 2 dimensi, dapat melakukan *zoom out(q)* dan *zoom in(e)*). Berikut adalah program *handler.py*.

```
from config import config
from transformasi import *

def on_keyPressed(key, mouseX, mouseY):

    if(key == 'd'):
        if(config.is3D):
            config.putarCamV(-config.navPutar)
        else:
            config.curMaxX += config.navX
            config.curMinX -= config.navX
            if(config.curMaxX > config.maxX):
                config.curMaxX -= (config.curMaxX - config.maxX)
            config.curMaxX = config.maxX
    elif(key == 'a'):
        if(config.is3D):
            config.putarCamV(config.navPutar)
        else:
            config.curMaxX -= config.navX
            config.curMinX += config.navX
            if(config.curMinX < config.minX):
                config.curMaxX += (config.minX - config.curMinX)
            config.curMinX = config.minX
    elif(key == 'w'):
        if(config.is3D):
            config.putarCam(config.navPutar, config.tegakLurusCam3D.item(0), config.tegakLurusCam3D.item(1), config.tegakLurusCam3D.item(2))
        else:
            config.curMaxY += config.navY
            config.curMinY -= config.navY
            if(config.curMaxY > config.maxY):
                config.curMinY -= (config.curMaxY - config.maxY)
            config.curMaxY = config.maxY
    elif(key == 's'):
        if(config.is3D):
```

```

config.putarCam(-config.navPutar,config.tegakLurusCam3D.item(0),config.tegakLurusCam3D.item(1),config.tegakLurusCam3D.item(2))
else:
    config.curMaxY -= config.navY
    config.curMinY -= config.navY
    if(config.curMinY < config.minY):
        config.curMaxY -= (config.minY-config.curMinY)
        config.curMinY = config.minY
elif(key == 'q'):
    if(config.is3D):
        pass
    else:
        # Zoom in
        config.curMinX += config.navZoom
        config.curMaxX -= config.navZoom
        config.curMinY += config.navZoom
        config.curMaxY -= config.navZoom
elif(key == 'e'):
    if(config.is3D):
        pass
    else:
        # Zoom out
        config.curMinX -= config.navZoom
        config.curMaxX += config.navZoom
        config.curMinY -= config.navZoom
        config.curMaxY += config.navZoom
elif(key == '0'):
    # Translasi ke atas
    config.objTest.animator.startAnimasi(translate(0, 5/config.maxKeyframe,0))
elif(key == '2'):
    # Translasi ke atas
    config.objTest.animator.startAnimasi(translate(0, -5/config.maxKeyframe,0))
elif(key == '4'):
    # Translasi ke atas
    config.objTest.animator.startAnimasi(translate(-5/config.maxKeyframe, 0,0))

elif(key == '6'):
    # Translasi ke atas
    config.objTest.animator.startAnimasi(translate(5/config.maxKeyframe, 0,0))

```

Kelompok kami mengerjakan spesifikasi bonus dalam tugas besar kami. Dalam melakukan transformasi geometri, program kami melakukan animasi dari bentuk dan posisi awal ke bentuk dan posisi terakhir. Untuk *multiple n*, animasi akan dilakukan apabila seluruh *n* perintah telah dilakukan dan program mengeluarkan ke layar “Keluar dari multiple”. Berikut adalah program *animasi.py*.

```

from transformasi import transformPolygon
from config import maxKey
from queue import Queue

class animasi:

    def __init__(self,isAnimasi=False,keyframe=0,maxKeyframe=maxKey,matrixAnimasi=0):
        self.isAnimasi = isAnimasi
        self.keyframe = keyframe
        self.maxKeyframe = maxKeyframe
        self.matrixAnimasi = matrixAnimasi
        self.queueAnimasi = Queue()

    def setMatrix(self,matrixAnimasi):
        self.matrixAnimasi = matrixAnimasi

    def animate(self,polygon):
        if(self.isAnimasi):
            if(self.keyframe >= self.maxKeyframe):
                if(not(self.isAnimasiEmpty())):
                    #Ada animasi lagi yang harus di animasikan
                    tupleQueue = self.popAnimasi()
                    self.startAnimasi(tupleQueue[0],tupleQueue[1])
                else:
                    #Tidak ada
                    self.keyframe = 0
                    self.isAnimasi = False
                    return polygon
            else:
                self.keyframe += 1
                return transformPolygon(polygon,self.matrixAnimasi)
        else:
            if(not(self.isAnimasiEmpty())):
                #Ada animasi lagi yang harus di animasikan
                tupleQueue = self.popAnimasi()
                self.startAnimasi(tupleQueue[0],tupleQueue[1])
            return polygon

    def startAnimasi(self,matrixAnimasi,langsung=False):
        #Cara pakai, test = animasi.startAnimasi(translate(10,2),test) <-Contoh
        self.isAnimasi = True
        self.setMatrix(matrixAnimasi)
        if(langsung):
            self.keyframe = self.maxKeyframe-1
        else:
            self.keyframe = 0
    def pushAnimasi(self,matrixAnimasi):
        self.queueAnimasi.put(matrixAnimasi)
    def popAnimasi(self):
        return self.queueAnimasi.get()
    def isAnimasiEmpty(self):
        return self.queueAnimasi.empty()

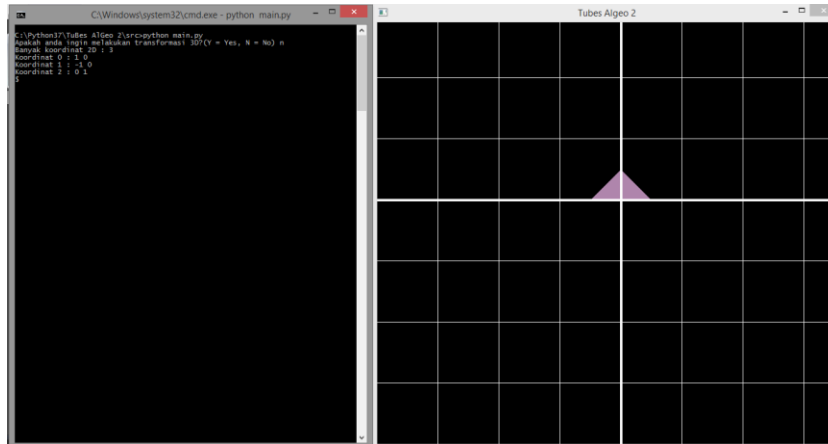
```

Bab IV. Eksperimen

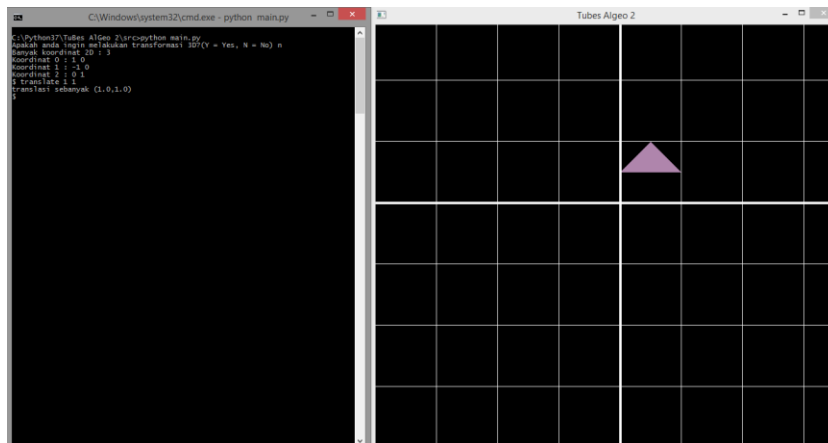
Berikut akan ditampilkan tangkapan-tangkapan layar berupa contoh-contoh penggunaan program kelompok kami. Akan dipisah 2 bagian besar, bagian transformasi di ruang 2 dimensi, dan transformasi di ruang 3 dimensi. (Note : animasi pada program tidak dapat ditunjukkan pada laporan)

Transformasi di ruang 2D :

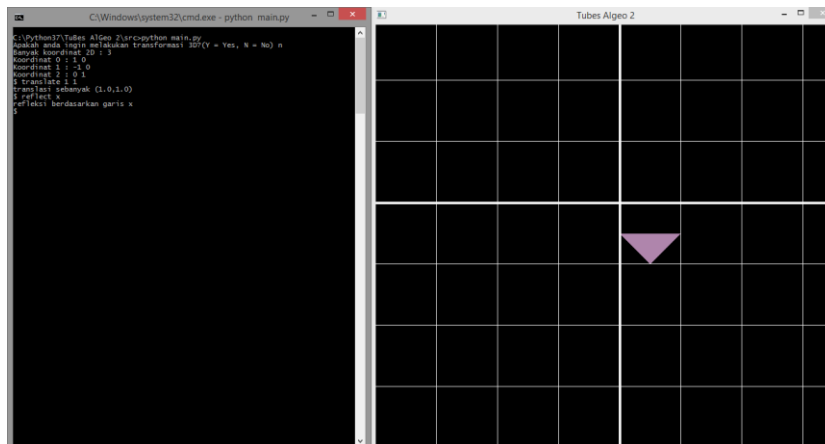
1. Memasukkan input (3 titik untuk membuat segitiga)



2. translate 1 1



3. reflect x



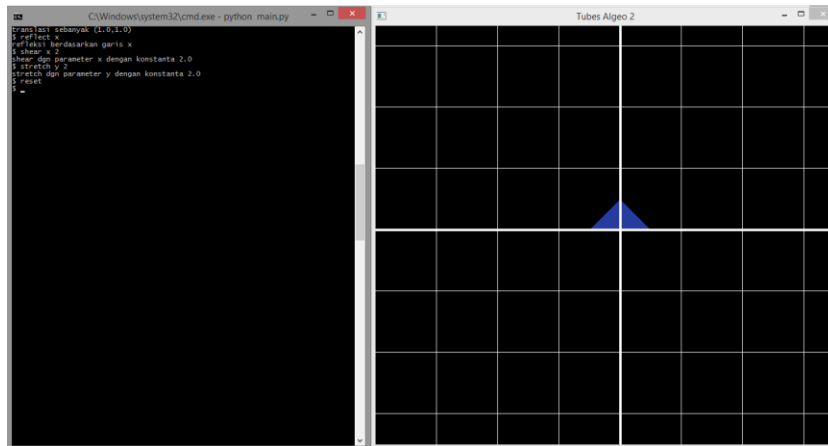
4. shear x 2



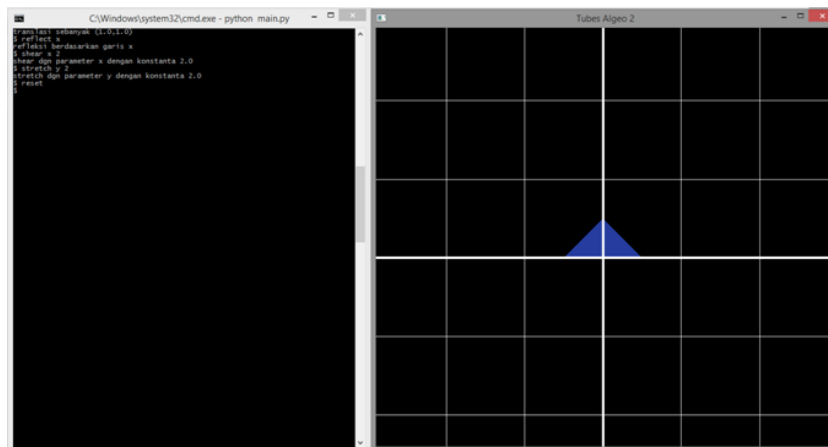
5. stretch y 2



6. reset

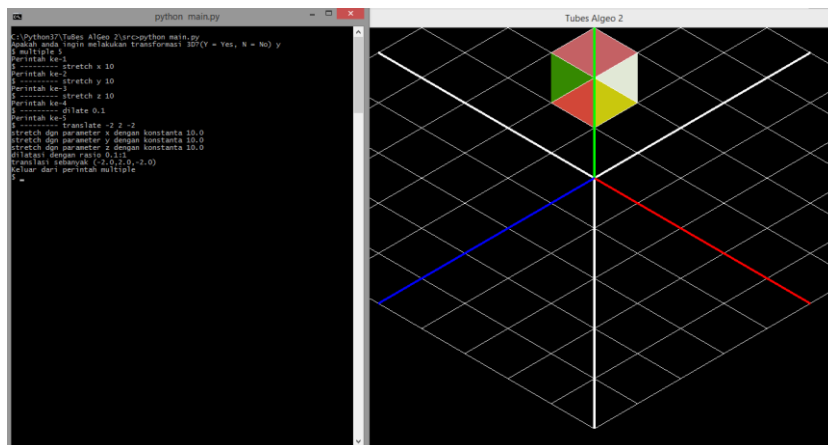


7. *zoom in* kamera

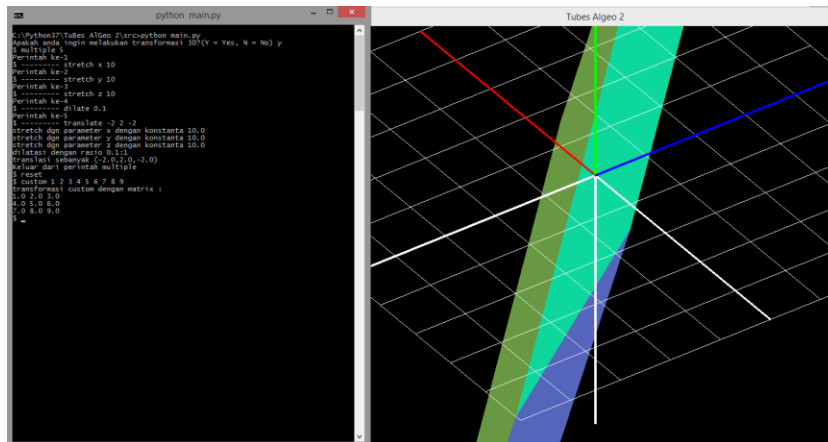


Transformasi di ruang 3D : (window terlalu besar sehingga ditangkap di bagian pentingnya)

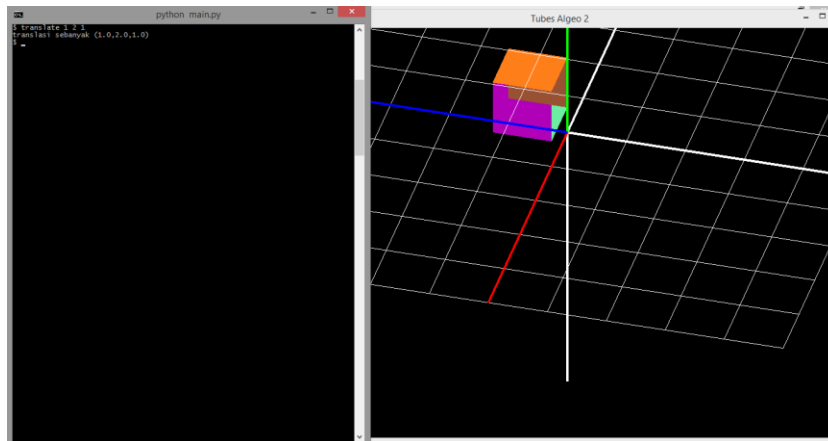
1. multiple 5



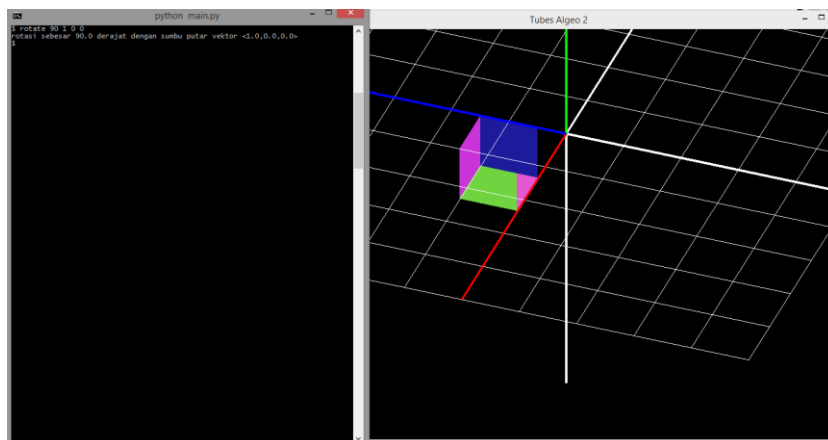
2. custom 1 2 3 4 5 6 7 8 9 (ditambah reset dan rotasi kamera)



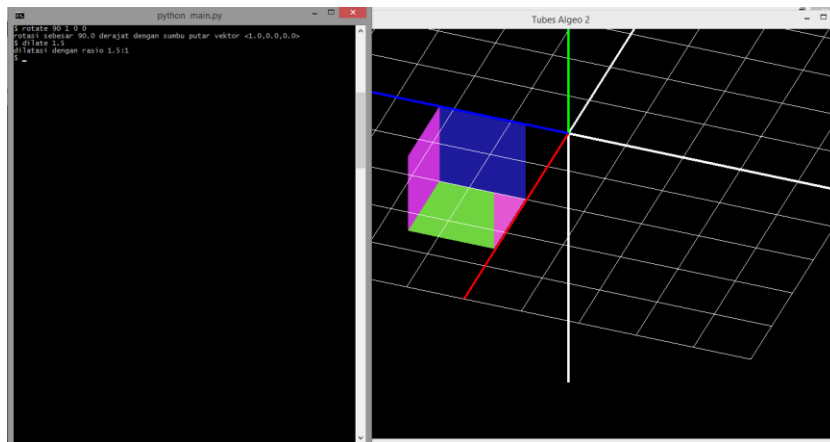
3. translate 1 2 1 (ditambah reset dan rotasi kamera)



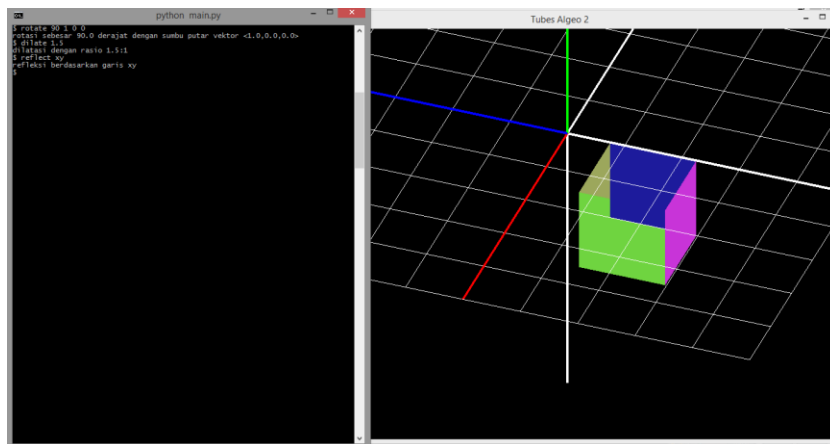
4. rotate 90 1 0 0



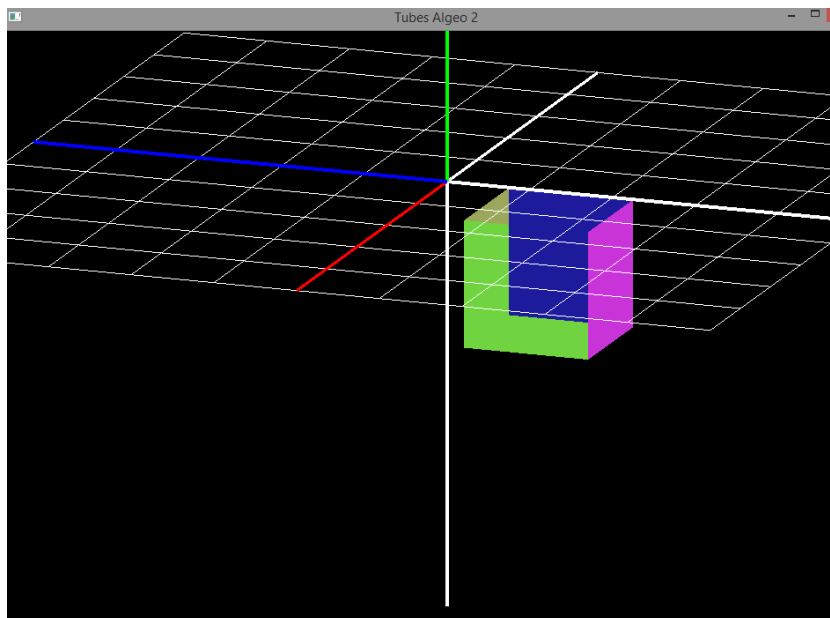
5. dilate 1.5



6. reflect xy



7. memutar kamera keatas



Dan berikut adalah contoh pengecekan apabila input benar (3 input teratas) dibandingkan dengan apabila input salah (6 pesan kesalahan di bawah) :



```
python main.py
$ rotate 90 1 0 0
rotasi sebesar 90.0 derajat dengan sumbu putar vektor <1.0,0.0,0.0>
$ dilate 1.5
dilatasi dengan rasio 1.5:1
$ reflect xy
refleksi berdasarkan garis xy
$ algeo
Input salah! Coba Lagi
$ translate 1 1
Input dx dy hanya bisa untuk 2D
$ multiple
Input salah!
$ multiple 4
Perintah ke-1
$ ----- reset
Tidak bisa melakukan reset didalam multiple
Perintah ke-1
$ ----- exitmultiple
Keluar dari perintah multiple
$ dilate -1
Input salah! (rasio harus > 0)
$ stretch -1
Input salah!
$
```

Bab V. Kesimpulan, saran, dan refleksi

Kesimpulan :

1. Program lulus kompilasi dan berjalan sesuai ekspektasi.
2. Seluruh spesifikasi (termasuk spesifikasi bonus) terpenuhi.

Saran :

1. Pengadaan implementasi algoritma pengakaran matriks agar program dapat melakukan animasi pada fungsi *custom* dan juga *reflect*.

Refleksi :

1. Lebih rapi dan sopan dalam berkomentar.
2. Memperjelas program dengan nama-nama variabel yang lebih menjelaskan.
3. Memperbanyak kerja kelompok bersama agar setiap informasi yang didapat dapat di-floor ke seluruh anggota dan kinerja kelompok jadi lebih lancar.
4. Jangan pernah menggunakan warna transparan untuk pewarnaan bidang menggunakan OpenGL. Karena adanya *bug* di OpenGL yang membuat warnat tidak bisa diconvert sama sekali.

Daftar Pustaka

1. <http://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2018-2019/Tubes2-Algeo-2018.pdf>
2. <https://idschool.net/sma/rumus-pada-transformasi-geometri-translasi-refleksi-rotasi-dan-dilatasi/>
3. https://en.wikipedia.org/wiki/Rotation_matrix
4. <https://blog.ruangguru.com/pengertian-dan-jenis-jenis-transformasi-geometri>
5. <http://www.technologyuk.net/mathematics/geometry/transformations.shtml>
6. https://www.khronos.org/opengl/wiki/Getting_Started