

LAPORAN TUGAS BESAR II IF2123 ALJABAR GEOMETRI

Simulasi Transformasi Linier pada Bidang 2D Dengan Menggunakan OpenGL API



Dibuat oleh

Adylan Roaffa Ilmy 13516016

Haifa Fadhila Ilma 13516076

INSTITUT TEKNOLOGI BANDUNG

2017

BAB I

DESKRIPSI MASALAH

1.1 Deskripsi Umum Tugas Besar

Pada tugas kali ini, mahasiswa diminta membuat program yang mensimulasikan transformasi linier untuk melakukan operasi translasi, refleksi, dilatasi, rotasi, dan sebagainya pada sebuah bidang 2D. Bidang dibuat dengan mendefinisikan sekumpulan titik sudut lalu membuat bidang dari titik-titik tersebut. 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. 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. Berikutnya, program dapat menerima input yang didefinisikan pada tabel dibawah.

1.2 Deskripsi Umum Masukan dan Keterangan

- a. Translasi
format masukan = `translate <dx> <dy>`
Melakukan translasi objek dengan menggeser nilai x sebesar dx dan menggeser nilai y sebesar dy.
- b. Dilatasi
format masukan = `dilate <k>`
Melakukan dilatasi objek dengan faktor scaling k.
- c. Rotasi
format masukan = `rotate <deg> <a> `
Melakukan rotasi objek secara berlawanan arah jarum jam sebesar deg derajat terhadap titik a,b
- d. Refleksi
format masukan = `reflect <param>`
Melakukan pencerminan objek. Nilai param adalah salah satu dari nilai-nilai berikut: x, y, y=x, y=-x, atau (a,b). Nilai (a,b) adalah titik untuk melakukan pencerminan terhadap.
- e. *Shear* (Menggeser)
format masukan = `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*.

f. *Stretch* (Meregangkan)

format masukan = `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.

g. *Custom*

format masukan = `custom <a> <c> <d>`

Melakukan transformasi linier pada objek dengan matriks transformasi A sebagai berikut:

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

h. *Multiple Commands*

format masukan = `multiple <n> ... // input 1 ... // input 2 ... // 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.

i. *Reset*

format masukan = `reset`

Mengembalikan objek pada kondisi awal objek didefinisikan.

j. *Exit*

format masukan = `exit`

Keluar dari program.

BAB II

TEORI SINGKAT

2.1 Transformasi Linier

Terdapat fungsi bernilai vektor dari sebuah peubah vektor. Yakni, fungsi yang berbentuk $w = F(v)$, dimana baik peubah bebas v maupun peubah tak-bebas w adalah vektor. Transformasi linier merupakan salah satu kelompok khusus fungsi vektor.

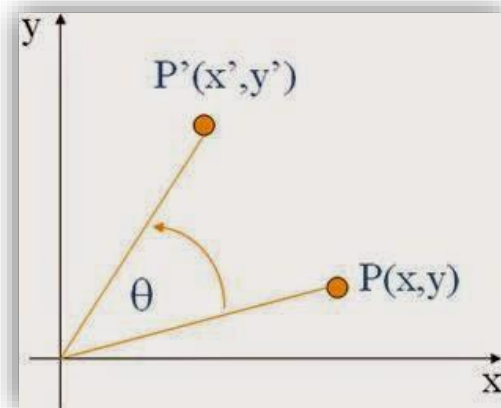
Tranformasi linier merupakan dasar dalam telaah aljabar yang berbentuk fungsi. Transformasi linier yang dimaksud adalah perpindahan dari satu ruang yang biasanya dinamakan dengan domain atau daerah asal ke ruang lain yang dinamakan kodomain atau daerah hasil.

1. Definisi 1: Suatu pemetaan f dari ruang vektor V ke ruang vektor W adalah aturan perkawanan sedemikian sehingga setiap vektor $v \in V$ dikawankan dengan vektor tunggal $w \in W$. Kita mengatakan bahwa f memetakan vektor v ke w , dan juga f memetakan ruang V ke W .
2. Definisi-2. Kata-kata pemetaan, operator, dan transformasi bermakna sama dengan pemetaan. Pada transformasi $f: V \rightarrow W$, ruang V disebut domain dan W disebut kodomain untuk f . Jika $u \in V$, maka vektor $f(u) \in W$ disebut bayangan dari u oleh f .
3. Definisi-3. Misalkan V dan W adalah ruang-ruang vektor atas medan K . Suatu transformasi linier dari V ke W adalah pemetaan $f: V \rightarrow W$ sedemikian sehingga $f(u + v) = f(u) + f(v)$ dan $f(ku) = kf(u)$ untuk semua $u, v \in V$ dan semua skalar $k \in K$.

Jika $F: V \rightarrow W$ adalah sebuah fungsi dari ruang vektor V ke dalam ruang vektor W , maka F dinamakan transformasi linier jika :

$F(u+v) = F(u) + F(v)$ untuk semua vektor u dan v di V

- Transformasi linier yang bekerja pada ruang vektor yang sama, $T: V \rightarrow V$ disebut operator linear.
- Transformasi linier $T: V \rightarrow W$ dengan $T(u) = 0$ disebut transformasi nol.
- Transformasi linier $T: V \rightarrow W$ dengan $T(x) = Ax$ disebut transformasi matriks sedangkan A disebut matriks transformasi.



2.2 Matriks Transformasi

Untuk memindahkan suatu titik atau bangun pada bidang, dapat dilakukan dengan menggunakan transformasi. Transformasi geometri adalah bagian dari geometri yang membicarakan perubahan, baik perubahan letak maupun bentuk dan penyajiannya berdasarkan dengan gambar dan matriks.

Transformasi yang tidak mengubah ukuran dan bentuk disebut transformasi isometri, diantaranya translasi (pergeseran), refleksi (pencerminan), dan rotasi (putaran). Selain itu, transformasi yang tidak isometri adalah dilatasi (perkalian), shear (pergeseran), dan juga stretch (peregangan), karena ukuran bayangan dapat diperbesar, diperkecil, dan bahkan berubah bentuk.

2.2.1 Translasi

Translasi adalah bentuk transformasi untuk memindahkan suatu objek pada bidang datar dengan jarak dan arah tertentu. Panjang jarak dan arah pada translasi dinyatakan oleh vektor \overline{AB} atau pasangan berurutan $\begin{pmatrix} a \\ b \end{pmatrix}$ dengan a merupakan komponen translasi pada arah sumbu-x dan b merupakan komponen translasi pada arah sumbu-y.

Suatu translasi dari R^2 (ruang dimensi dua) ke R^2 didefinisikan oleh pemetaan:

$$T : R^2 \rightarrow R^2$$

Maka, jika titik $P(x,y)$ ditranslasikan oleh matriks transformasi $T = \begin{pmatrix} a \\ b \end{pmatrix}$, artinya titik $P(x,y)$ ditranslasikan sejauh a satuan sepanjang sumbu-x dan b satuan sepanjang sumbu-y dan diperoleh x' dan y' sehingga berlaku:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a \\ b \end{pmatrix}$$

2.2.2 Dilatasi

Dilatasi adalah bentuk transformasi geometri yang memperbesar atau memperkecil objek tanpa mengubah bentuk objek tersebut. Untuk melakukan dilatasi diperlukan factor pengali atau skala. Jika skala > 1 maka bentuk obyek diperbesar, sebaliknya jika skala < 1 maka objek diperkecil.

Persamaan dilatasi dengan pusat $O(0,0)$ dan k skala dinyatakan dalam bentuk $x' =$

kx dan $y' = ky$. atau, dengan matriks transformasi $T : \begin{pmatrix} k & 0 \\ 0 & k \end{pmatrix}$, persamaan matriksnya adalah:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} k & 0 \\ 0 & k \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

2.2.3 Rotasi

Rotasi adalah bentuk transformasi geometri untuk memindahkan obyek dengan cara pemutaran. Untuk melakukan rotasi diperlukan titik pusat, besar sudut dan arah

sudut rotasi. Arah putaran sudut positif berlawanan dengan jarum jam, sebaliknya untuk arah sudut yang negatif putaran searah dengan jarum jam.

Misalkan titik $P(x,y)$, diputar dengan titik pusat $O(0,0)$ dengan sudut putar sebesar q radian berlawanan arah jarum jam, untuk mendapatkan titik hasil rotasi yaitu titik

$P'(x',y')$. Matriks transformasinya adalah $T : \begin{pmatrix} \cos q & -\sin q \\ \sin q & \cos q \end{pmatrix}$, persamaan matriksnya adalah:

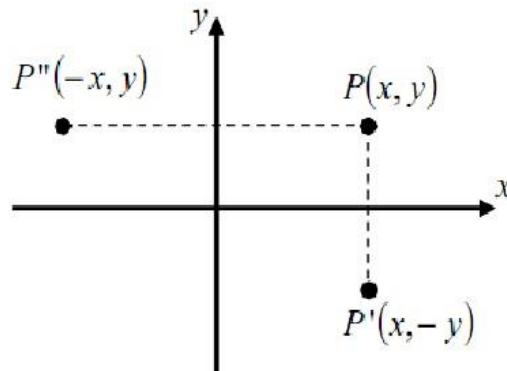
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos q & -\sin q \\ \sin q & \cos q \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Selain itu, jika titik $P(x,y)$ diputar dengan titik pusat $A(a,b)$ dengan sudut putar sebesar q radian berlawanan arah jarum jam, persamaannya ialah:

$$\begin{pmatrix} x' - a \\ y' - b \end{pmatrix} = \begin{pmatrix} \cos q & -\sin q \\ \sin q & \cos q \end{pmatrix} \begin{pmatrix} x - a \\ y - b \end{pmatrix}$$

2.2.4 Refleksi

Refleksi (pencerminan) adalah bentuk transformasi geometri yang memindahkan obyek menjadi bayangan seperti di depan cermin. Pencerminan titik terhadap sumbu cermin, jarak titik asal ke sumbu cermin sama dengan jarak titik bayangan ke sumbu cermin. Pada koordinat kartesius, titik $P(x,y)$, dicerminkan terhadap sumbu x dan sumbu y hasil dari pencerminan diperlihatkan dengan gambar dibawah ini:



Gambar 8.5.6 Pencerminan $P(x, y)$ terhadap sumbu koordinat

$$\begin{aligned} x' = x &\leftrightarrow x' = 1x + 0y & \begin{pmatrix} x' \\ y' \end{pmatrix} &= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \\ y' = -y &\leftrightarrow y' = 0x - 1y \end{aligned}$$

Pencerminan titik terhadap sumbu cermin, jarak titik asal ke sumbu cermin sama dengan jarak titik bayangan ke sumbu cermin. Pada koordinat kartesius, titik $P(x,y)$, dicerminkan terhadap sumbu x dan sumbu y hasil dari pencerminan diperlihatkan pada gambar. Titik $P(x,y)$, dicerminkan terhadap sumbu x menghasilkan $P'(x,-y)$, bentuk persamaan hasil pencerminan ini adalah: Dinyatakan dalam bentuk persamaan matriks:

No.	Transformasi	Matriks	Pemetaan
1	Pencerminan terhadap sumbu x	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	$(x,y) \rightarrow (x,-y)$
2	Pencerminan terhadap sumbu y	$\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$	$(x,y) \rightarrow (-x,y)$
3	Pencerminan terhadap titik asal O	$\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$	$(x,y) \rightarrow (-x,-y)$
4	Pencerminan terhadap garis $y = x$	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	$(x,y) \rightarrow (y,x)$
5	Pencerminan terhadap garis $y = -x$	$\begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}$	$(x,y) \rightarrow (-y,-x)$
6	Pencerminan terhadap garis $y = x \tan \alpha$	$\begin{pmatrix} \cos 2\alpha & \sin 2\alpha \\ \sin 2\alpha & -\cos 2\alpha \end{pmatrix}$	

Selain itu, terdapat pula pencerminan terhadap suatu titik (a,b) yang dinyatakan dengan persamaan berikut:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 2a \\ 2b \end{pmatrix} - \begin{pmatrix} x \\ y \end{pmatrix}$$

2.2.4 Shear (Penggesean)

Geseran. Sebuah geseran dalam arah x dengan faktor k adalah transformasi yang menggerakkan masing-masing titik (x, y) sejajar dengan sumbu x sebanyak ky menuju kedudukan yang baru $(x+ky, y)$. Dibawah transformasi seperti itu, titik-titik pada sumbu x tidak digerakkan karena $y=0$. Akan tetapi, sewaktu kita makin menjauh dari sumbu x , besar y bertambah, sehingga titik-titik yang lebih jauh dari sumbu x bergerak sejarak yang lebih besar dari titik-titik yang lebih dekat ke sumbu x tersebut.

Sebuah geseran dalam arah y dengan faktor k adalah transformasi yang menggerakkan masing-masing titik (x, y) sejajar dengan sumbu y sebanyak kx menuju kedudukan yang baru $(x, y+kx)$. Dibawah transformasi seperti itu, titik-titik pada sumbu y tetap diam dan titik-titik yang lebih jauh dari sumbu y bergerak sejarak yang lebih besar dari titik-titik yang lebih dekat ke sumbu y tersebut.

Dapat kita perhatikan bahwa geseran adalah transformasi linier. Jika $T: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ adalah geseran dalam arah x dengan faktor k , persamaan matriks untuk T adalah:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Demikian juga, persamaan matriks untuk geseran didalam arah y adalah:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ k & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

2.2.5 Stretch (Ekspansi/Kompresi)

Jika koordinat x dari masing-masing titik pada bidang dikalikan dengan konstanta k yang positif, maka efeknya adalah memperluas atau mengompresi masing-masing gambar bidang dalam arah x . Jika $0 < k < 1$, maka hasilnya adalah kompresi, dan jika $k > 1$, maka hasilnya adalah ekspansi.

Kita namakan transformasi seperti itu ekspansi (atau kompresi) dalam arah x dengan faktor k . Demikian juga, jika koordinat y dari masing-masing titik dikalikan dengan konstanta k positif, kita dapatkan sebuah ekspansi (atau kompresi) dalam arah y dengan faktor k . Dapat diperlihatkan bahwa ekspansi dan kompresi sepanjang sumbu-sumbu koordinat adalah transformasi linier.

Jika $T: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ adalah ekspansi atau kompresi dalam arah x dengan faktor k , persamaan matriks untuk T adalah:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} k & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Demikian juga, persamaan matriks untuk ekspansi dan kompresi didalam arah y adalah:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & k \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

2.3 OpenGL

2.3.1 Deskripsi Umum OpenGL

OpenGL adalah sebuah program aplikasi interface yang digunakan untuk mendefinisikan komputer grafis 2D dan 3D. Program lintas-platform API ini umumnya dianggap ketetapan standar dalam industri komputer dalam interaksi dengan komputer grafis 2D dan juga telah menjadi alat yang biasa untuk digunakan dengan grafis 3D. Singkatnya, Open Graphics Library, OpenGL menghilangkan kebutuhan untuk pemrogram untuk menulis ulang bagian grafis dari sistem operasi setiap kali sebuah bisnis akan diupgrade ke versi baru dari sistem.

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.

Membuat perintah dalam OpenGL dapat terjadi dalam dua cara yang berbeda. Pertama, adalah mungkin bagi programmer untuk membuat dan menyimpan daftar perintah yang dapat dieksekusi secara berulang. Ini adalah salah satu cara yang lebih rutin untuk program interface yang digunakan. Seiring dengan berkembangnya kelompok perintah yang kurang lebih permanen, maka memungkinkan untuk membuat dan menjalankan salah satu perintah dalam batas-batas waktu dari komputer grafis.

Seiring dengan kemampuan interface dari sistem operasi, OpenGL juga menyediakan beberapa built-in protokol yang mungkin berguna bagi pengguna akhir. Di antaranya fitur alat seperti alpha blending, pemetaan tekstur, dan efek atmosfer. Alat ini dapat berinteraksi dengan sistem operasi yang sedang digunakan.

2.3.2 Sintax Perintah OpenGL

Sintaks perintah OpenGL mengikuti aturan penulisan dari library dimana fungsi tersebut berasal, format penulisan fungsi OpenGL adalah :

<awalan library><perintah><optional jumlah argument><optional tipe argument>

Semua perintah OpenGL menggunakan awalan gl diikuti dengan huruf kapital pada setiap kata membentuk nama perintah (sebagai contoh glColor3f). Untuk mendefinisikan konstanta diawali dengan GL_, dengan menggunakan huruf kapital dan garis bawah untuk memisahkan kata (seperti GL_POLY_STIPPLE). Terkadang beberapa huruf dan angka ditambahkan pada akhir perintah (seperti 3f pada glVertex3f). Dalam hal ini angka 3 menunjukkan berapa banyak argumen yang harus ada pada perintah tersebut dan akhiran huruf f menunjukkan jenis datanya yaitu floating. Sebagai contoh pada dua perintah berikut ini :

```
glVertex3i(1,0,-2);
```

```
glVertex3f(1.0, 0.0, -2.0);
```

adalah sama yaitu meletakkan titik di layar pada koordinat $x = 1$, $y = 0$ dan $z = -2$, perbedaannya yaitu pada perintah pertama menspesifikasikan titik dengan tipe data integer 32-bit, sedangkan yang kedua dengan tipe data single precision floating point. Beberapa perintah OpenGL menambahkan perintah huruf akhir v yang menunjukkan bahwa perintah tersebut menggunakan pointer ke array/vektor. Di bawah ini contoh perbedaannya.

```
float color_array[]={1.0,0.0,0.0}
```

```
glColor3f (1.0,0.0,0.0);
```

```
glColor3fv(color_array);
```

2.3.3 Library yang Berhubungan dengan OpenGL

OpenGL menyediakan set perintah untuk menggambar dan semua penggambaran yang lebih tinggi tingkatnya harus dilakukan dengan mengambil fungsi dasar dari perintah ini. Maka dari itu dapat dibuat library itu sendiri di atas program OpenGL yang mempermudah pemrograman lebih lanjut. Fungsi asli dari OpenGL sendiri selalu diawali dengan gl yang terdapat pada library opengl32.dll dan file header gl.h. Sedangkan beberapa library yang telah ditulis untuk menyediakan fungsi-fungsi tambahan pada OpenGL adalah :

1. OpenGL Utility Library (GLU) yang didalamnya terdapat sejumlah rutin yang menggunakan level bawah dari perintah OpenGL. Rutin-rutin ini mempunyai awalan glu. Library ini digunakan sebagai bagian dari implementasi OpenGL.
2. OpenGL Extension untuk X-Windows yang menyediakan fungsi untuk menciptakan OpenGL context dan mengasosiasikannya dengan mesin yang menggunakan XWindows. Rutin-rutin ini mempunyai awalan glx.
3. Auxiliary atau aux library terdapat pada library glaux.lib dan file header glaux.h. Perintah yang akan digunakan selalu menggunakan awalan aux
4. OpenGL Utility Toolkit (GLUT) adalah toolkit untuk sistem windows yang ditulis oleh Mark Kilgard untuk menyembunyikan perintah API sistem windows yang kompleks.

BAB III

IMPLEMENTASI PROGRAM

3.1 Implementasi Program

Pada tugas besar kali ini, kami menggunakan pemrograman dengan Bahasa *Python* dan menggunakan Open Graphics Library (OpenGL) API dalam pengerjaannya. Selain itu, kami juga memanfaatkan modul NumPy, yaitu modul yang menyediakan objek-objek matematika yang memudahkan dalam melakukan perhitungan. Objek utama yang disediakan NumPy adalah array yang dapat berperan sebagai matriks. Array ini tidak sama dengan array biasa pada bahasa-bahasa pemrograman secara umum. Untuk itu, kami menggunakannya karena kami melakukan masukan beberapa titik untuk menggambar sebuah bidang dan melakukan berbagai transformasi matriks sebagai implementasi dari program.

3.1.1 Modul

```
import numpy as np
import time
import thread
import math
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
from copy import deepcopy
```

3.1.2 Fungsi dan Prosedur

A. Tampilan

def refresh2d(width, height):

Library OpenGL *defaultnya* dapat menampilkan gambar dalam tampilan 3D. Fungsi ini dipanggil saat pengguna ingin menampilkan gambar dalam 2D. Yang dilakukan adalah mengatur tampilan layar dan perspektif agar terlihat seperti 2D.

def idle_draw(points):

Pada Prosedur ini, pertama-tama kita melakukan *screen clearing* dan dan me-reset posisi. Selain itu kita juga me-reset warna background menjadi putih. Lalu, dipanggil beberapa fungsi lain seperti fungsi `draw_shape(points)`, `draw_line()`, `draw_grids()`.

def draw_shape(points):

Fungsi ini menggambar bentuk *polygon* (minimal 3 sisi) berdasarkan masukan kumpulan titik-titik yang ada pada array `points`.

def draw_line():

Prosedur ini menggambar 2 garis ditengah tampilan layar sebagai sumbu x dan y.

def draw_grids():

Prosedur ini menggambar garis-garis sejajar sumbu x dan y di sepanjang tampilan layar membentuk *grids*.

def input_vertices():

Prosedur ini menerima masukan berupa jumlah titik (sebagai identifikasi polygon-sisi-berapa yang akan dibuat) lalu koordinat tiap titik tersebut untuk kemudian dimasukkan ke array *current_vertices* untuk kemudian diproses pada fungsi-fungsi berikutnya.

def animate_transformation(current_vertices, func, *args):

Prosedur ini memungkinkan perubahan posisi/bentuk dari objek dianimasikan. Fungsi ini menggunakan algoritma looping dengan range selisih sudut ataupun selisih jarak beserta time delay nya.

def draw():

Prosedur ini memanggil prosedur *idle_draw()* dengan mengisi parameter *points* dengan *current_vertices*.

B. Fungsi-Fungsi Transformasi

def translate_vertices(current_vertices,dx,dy):

Fungsi ini melakukan translasi terhadap objek. Setiap titik absis ditambahkan dengan dx dan ordinat ditambahkan dy.

def dilate_vertices(vertices,k):

Fungsi ini melakukan dilatasi terhadap objek. Setiap titik dikalikan dengan faktor pengali k.

def reflectpoint_vertices(current_vertices, dx, dy):

Fungsi ini melakukan refleksi/pencerminan objek terhadap suatu titik dx dan dy.

def reflectline_vertices(current_vertices, line):

Fungsi ini melakukan refleksi/pencerminan objek terhadap berbagai garis. Garis yang dapat diproses disini adalah garis $y=x$, $y=-x$, sumbu x dan sumbu y. Setiap titik pada objek akan diproses berdasarkan matriks transformasi refleksi sesuai garis-garisnya.

def rotate_vertices(current_vertices, angle, pointA, pointB):

Fungsi ini melakukan rotasi objek dengan sudut tertentu (parameter angle) terhadap sebuah titik (pointA dan pointB). Setiap titik pada objek akan diproses berdasarkan matriks transformasi rotasi.

def shear_vertices(current_vertices, param, k):

Fungsi ini melakukan penggeseran dengan faktor k dan parameter berupa penggeseran terhadap sumbu x atau y. Setiap titik pada objek akan diproses berdasarkan matriks transformasi *shear*.

def stretch_vertices(current_vertices, param, k):

Fungsi ini melakukan *stretching object* dengan faktor k dan parameter terhadap sumbu x atau y. Setiap titik pada objek akan diproses berdasarkan matriks transformasi *stretch*. Jika param = x, maka absisnya yang akan dikalikan k, jika param = y, maka ordinatnya dikalikan k.

def custom_transform_vertices(current_vertices,command):

Fungsi ini mentransformasi tiap titik pada objek dengan matriks transformasi *custom* sesuai masukan pengguna.

def multiple_commands(current_vertices,n):

Fungsi ini akan menerima masukan berupa *command* untuk mentransformasi objek sebanyak n kali (selain command multiple, reset, dan exit), kemudian memasukkan command-command tersebut ke *command_list* untuk kemudian dijalankan secara berurutan setelah itu.

def reset_vertices(current_vertices):

Fungsi ini mengembalikan nilai *current_vertices* (titik-titik yang sekarang) menjadi *initial_vertices* (titik-titik awal saat masukan).

C. Pembacaan Command untuk Memanggil Fungsi Transformasi

```
def command_action(current_vertices,command):
```

Prosedur ini akan melakukan pemrosesan string yang ada pada parameter command. String pertama akan diproses untuk menentukan transformasi apa, atau proses apa yang akan dilakukan selanjutnya (misal translate, rotate, atau reset), lalu string setelahnya (jika ada) diproses sesuai kebutuhan parameter masing-masing fungsi, lalu didalamnya dilakukan pemanggilan fungsi `animate_transformation()` dengan parameter berupa `current_vertices` dan fungsi transformasi yang akan digunakan.

```
def get_command():
```

Prosedur ini melakukan pemanggilan prosedur `command_action()` dan memungkinkan prosedur tersebut dilakukan berulang kali, dan juga melakukan *exception/ error handling*.

D. Program Utama

```
def main():
```

```
    glutInit()
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_ALPHA |
GLUT_DEPTH)
    glutInitWindowSize(width, height)
    glutInitWindowPosition(500, 100)
    thread.start_new_thread(get_command, ())
    window = glutCreateWindow("Tugas Besar Aljabar Geometri -
2EZ4ANH")
    glutDisplayFunc(draw)
    glutIdleFunc(draw)
    glutMainLoop()
```

```
def start():
```

```
    while (not drawn):
        print "Enter the vertices"
        try:
            input_vertices()
        except ValueError as e:
            print "Input salah, mohon ulangi"
```

```
    main()
```

```
start()
```

3.2 Pembagian Tugas

Kami melakukan Tugas Besar II Aljabar Geometri kali ini dengan pembagian tugas pada fungsi-fungsi yang dibuat. Pembagian pengerjaan tugas dapat dilihat sebagai berikut:

1. Adylan Roaffa Ilmy (13516016)

- Fungsi draw_line()
- Fungsi input_vertices()
- Fungsi animate_transformation()
- Fungsi translate_vertices()
- Fungsi dilate_vertices()
- Fungsi custom_transform_vertices()
- Fungsi multiple_commands()
- Fungsi reset()
- Fungsi get_command()

2. Haifa Fadhila Ilma (13516076)

- Fungsi draw_grids()
- Fungsi reflectpoint_vertices()
- Fungsi reflectline_vertices()
- Fungsi rotate_vertices()
- Fungsi shear_vertices()
- Fungsi stretch_vertices()
- Membuat Laporan

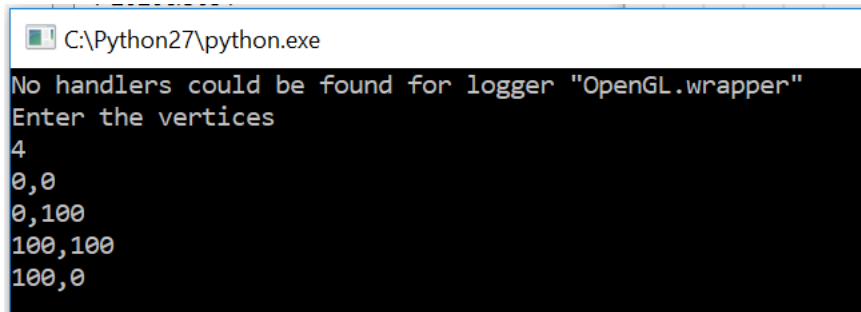
Selain itu, fungsi-fungsi lainnya seperti fungsi command_action() dibuat dan dilengkapi bersama, dan fungsi-fungsi lain seperti refresh2d(), draw(), draw_shape() dibuat berdasarkan sumber spesifikasi tugas besar dengan beberapa penyesuaian. Secara umum, presentase beban tugas yang kami kerjakan kurang lebih mencapai angka yang sama.

BAB IV

EKSPERIMEN

Berikut merupakan *screenshot* hasil eksperimen atau *test case* dari tiap-tiap transformasi objek yang dapat dilakukan oleh program kami:

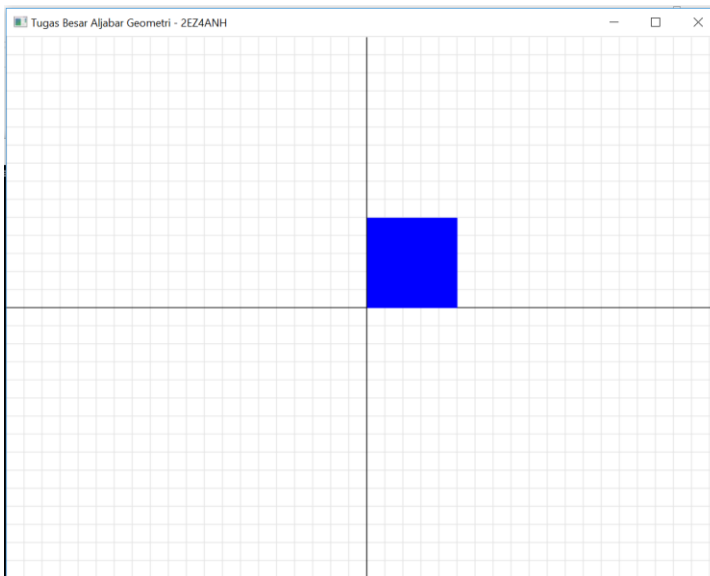
1. Menggunakan Bentuk Persegi



```
C:\Python27\python.exe
No handlers could be found for logger "OpenGL.wrapper"
Enter the vertices
4
0,0
0,100
100,100
100,0
```

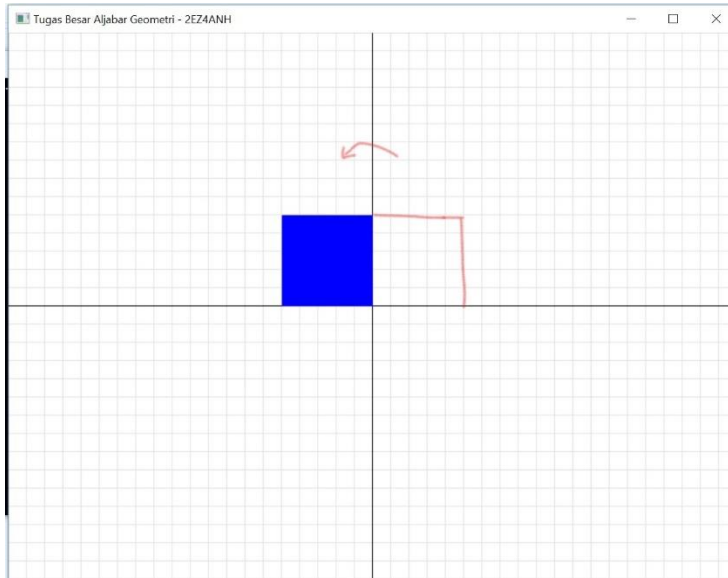
Berikut adalah command awal yang dimasukkan saat ingin membuat persegi dengan dimensi 100x100.

Dibawah ini adalah tampilan awal persegi tersebut:



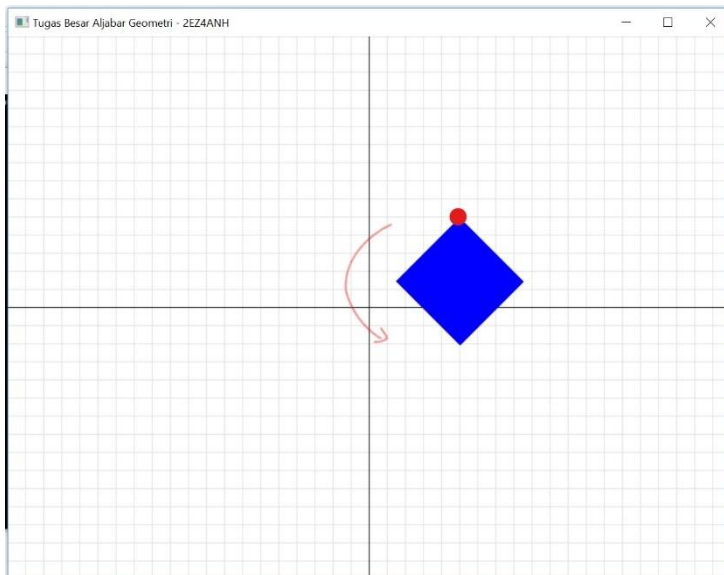
Setelah itu, kami memasukkan beberapa command untuk mentransformasi persegi tersebut. Berikut adalah command-command dan hasilnya:

a. reflect y



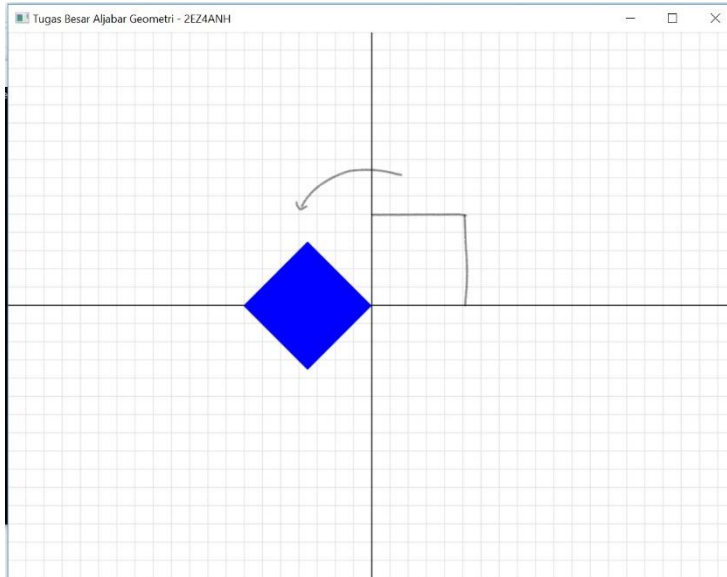
Command ini menyebabkan persegi di kondisi awal direfleksikan terhadap sumbu y seperti pada gambar diatas.

b. rotate 45 100 100



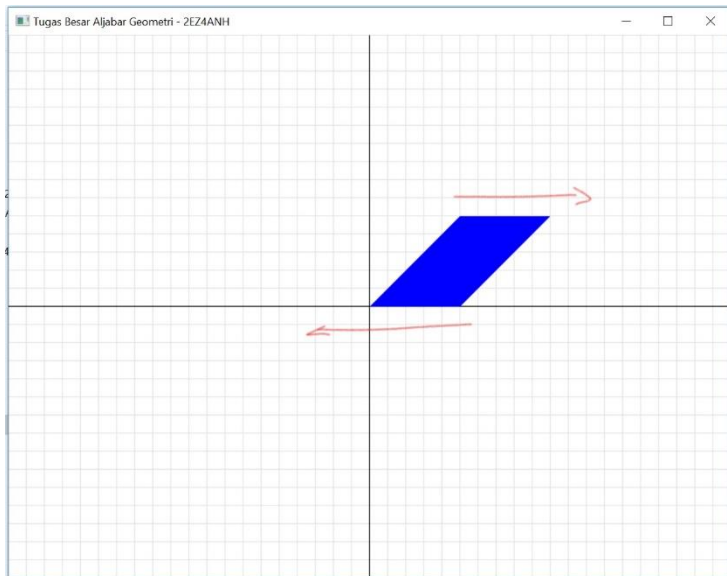
Command ini menyebabkan persegi pada kondisi awal dirotasi sebesar 45° berlawanan arah jarum jam dengan sumbu di titik $x=100$ dan $y=100$ (ditunjukkan pada titik merah pada gambar), yang hasil akhirnya ditunjukkan pada gambar diatas.

c. rotate 135 0 0



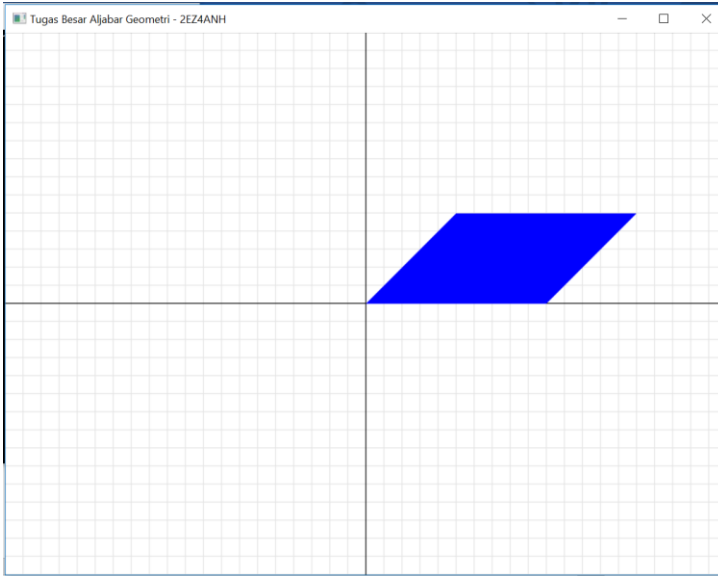
Command ini menyebabkan persegi pada kondisi awal dirotasi sebesar 135° berlawanan arah jarum jam dengan titik sumbu $x=0$ dan $y=0$ seperti ditunjukkan pada gambar.

d. shear x 1



Dapat dilihat pada gambar diatas, command ini membuat persegi pada kondisi awal mengalami pergeseran terhadap sumbu x dengan faktor k sebesar 1.

e. custom 2 0 1 1

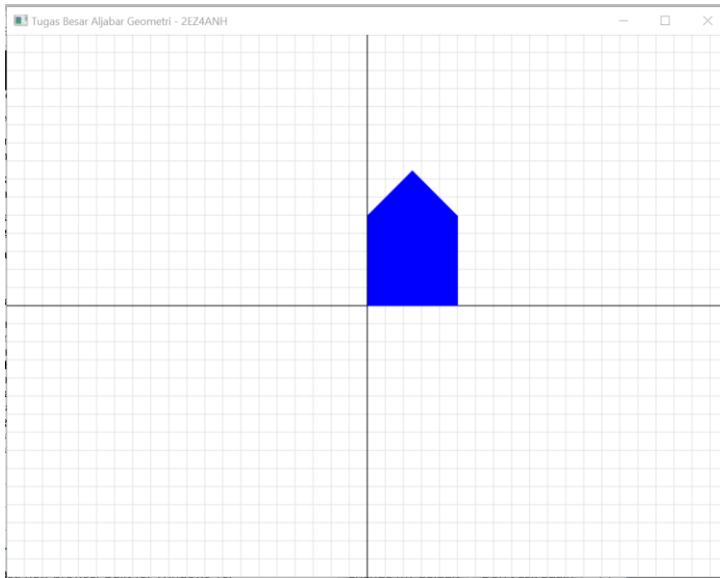


Command custom ini mentransformasi bentuk awal dengan matriks transformasi $= \begin{pmatrix} 2 & 0 \\ 1 & 1 \end{pmatrix}$.

2. Menggunakan Bentuk “Rumah” (Segi-5)

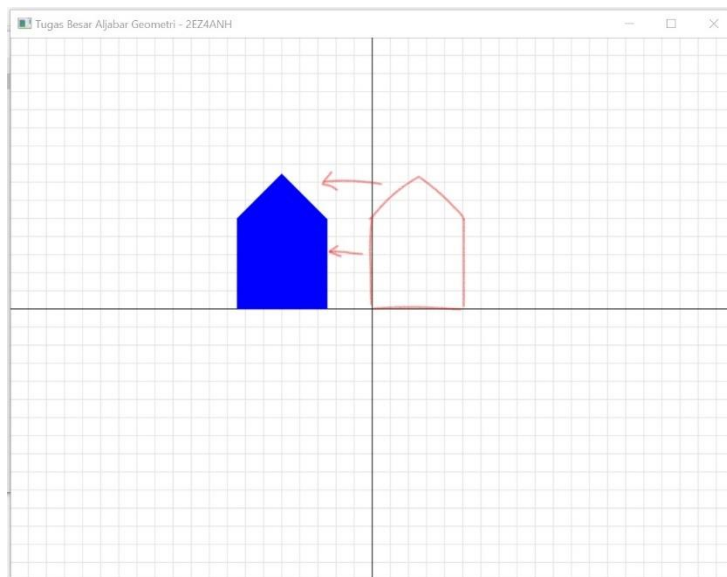
```
C:\Python27\python.exe
No handlers could be found for logger "OpenGL.wrapper"
Enter the vertices
5
0,0
100,0
100,100
50,150
0,100
```

Berikut adalah command awal untuk membuat objek 2 dimensi berbentuk seperti rumah atau segi-5 dengan titik-titik x dan y tertentu. Dibawah ini adalah tampilan awalnya:



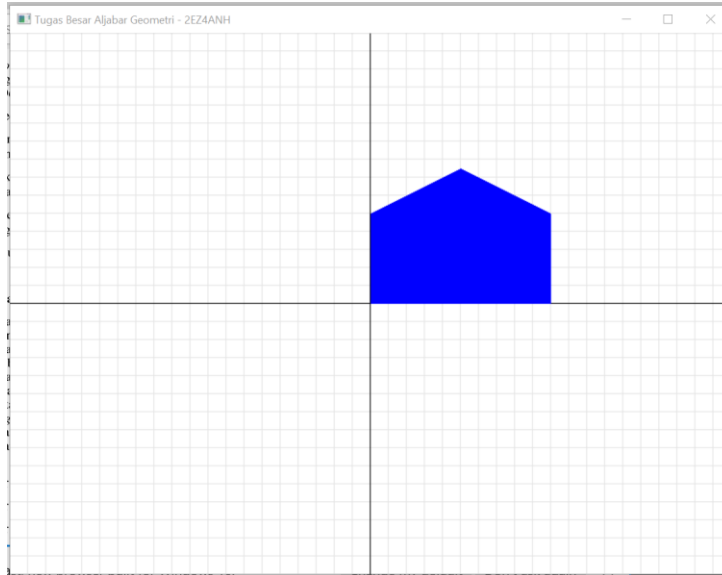
Setelah itu, kami memasukkan beberapa command untuk mentransformasi bangun datar tersebut. Berikut adalah command-command dan hasilnya:

a. translate -150 0



Gambar diatas menunjukkan transformasi translasi dengan parameter x sebesar -150 dan y sebesar 0 terhadap gambar awal. Artinya, pentagon ini ditranslasi ke arah sumbu x negatif (ke kiri) sebesar 150 satuan dan tidak ke arah sumbu y (karena $y=0$).

b. stretch x 2



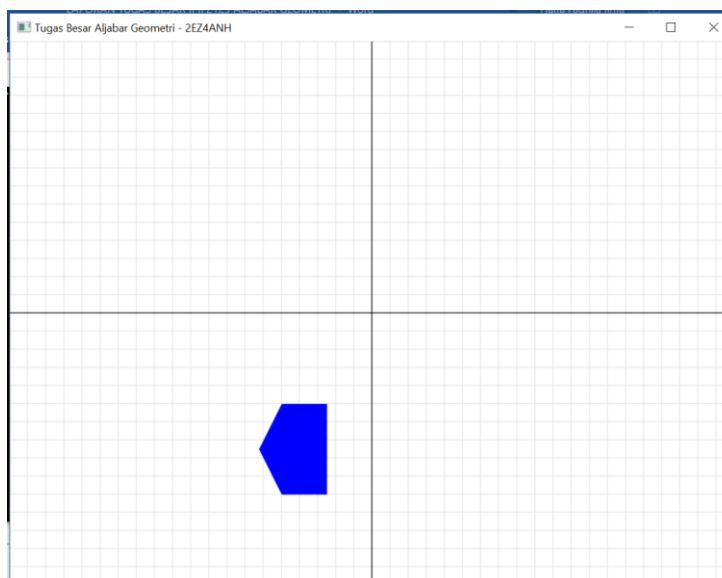
Command stretch ini menyebabkan bentuk awal di transformasi (diregangkan) sebesar factor skala $k = 2$, dan terhadap sumbu x . Jadi bentuk awal seperti dilebarkan secara horizontal (2 kali lebih lebar dari bentuk awal).

c. Multiple commands

```
multiple 3
translate 100 100
stretch y 0.5
reflect y=-x
```

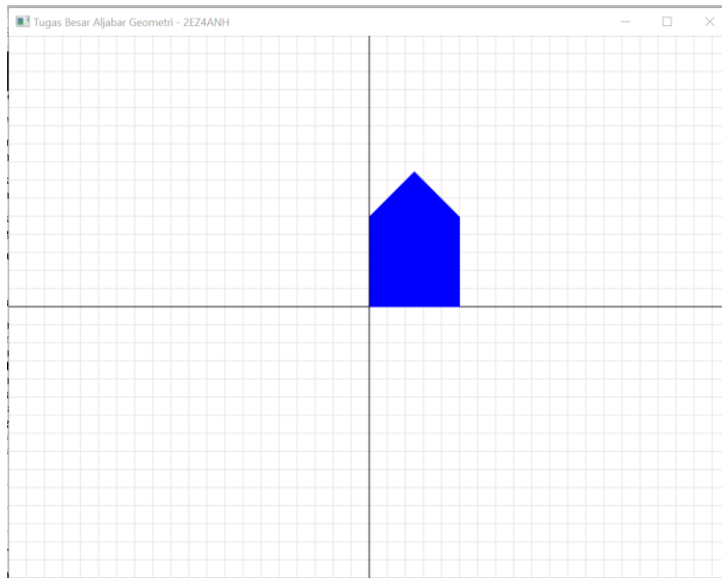
Gambar diatas menunjukkan masukan saat ingin melakukan beberapa transformasi sekaligus.

Berikut hasil dari rentetan command transformasi diatas:



Jadi, pertama-tama, poligon ditranslasi sejauh $x=100$ dan $y=100$, lalu polygon tersebut di regangkan terhadap sumbu y (vertikal) sebesar factor skala 0.5, yang artinya panjang polygon tersebut diperkecil (menjadi setengah ukuran awal). Lalu, polygon yang telah ditranslasi dan diperkecil tersebut direfleksikan terhadap garis $y=-x$, dan hasil akhirnya seperti yang ditampilkan pada gambar.

d. reset



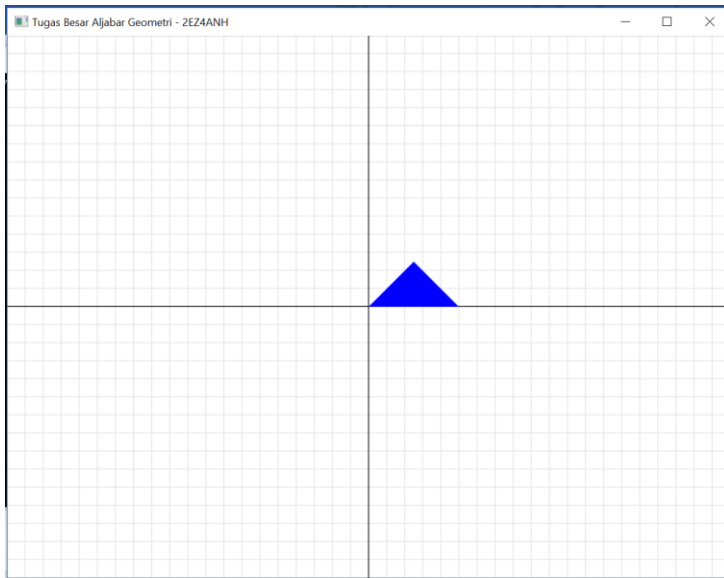
Command ini mengembalikan bentuk apapun hasil transformasi bentuk menjadi seperti semula.

3. Menggunakan Bentuk Segitiga

```
C:\Python27\python.exe
No handlers could be found for logger "OpenGL.wrapper"
Enter the vertices
3
0,0
100,0
50,50
```

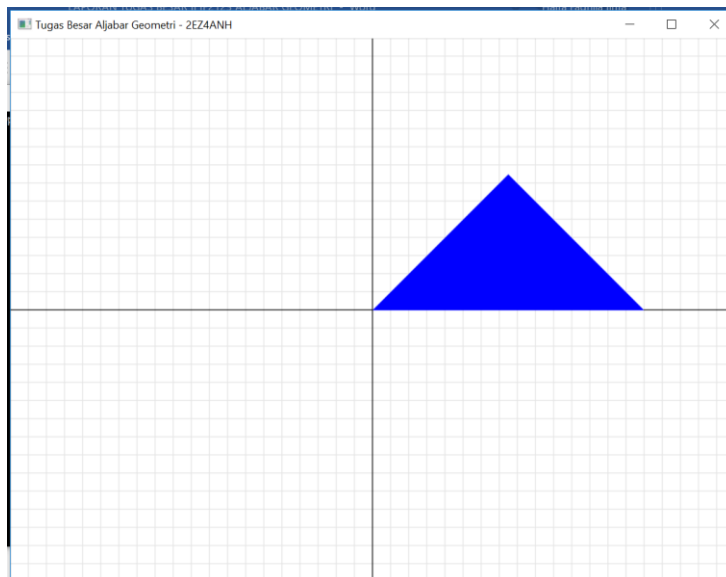
a. dilate 3

Berikut adalah command awal yang dimasukkan saat ingin membuat segitiga dengan alas 100 dan tinggi 50. Dibawah ini adalah tampilan awal segitiga tersebut:



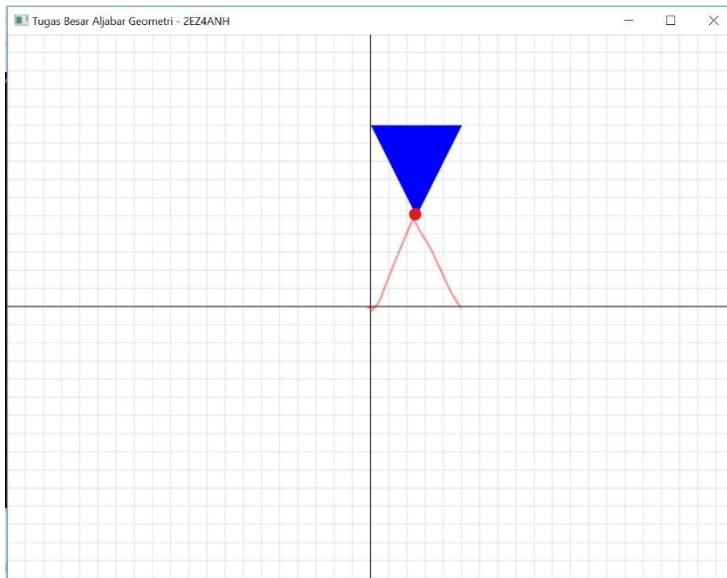
Setelah itu, kami memasukkan beberapa command untuk mentransformasi segitiga tersebut. Berikut adalah command-command dan hasilnya:

a. dilate 3



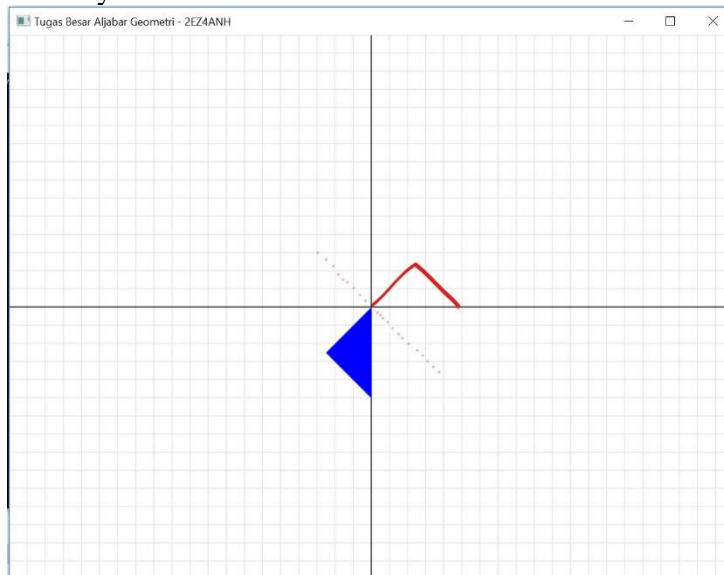
Command ini melakukan dilatasi (perbesaran atau perkalian) terhadap bentuk awal segitiga dengan factor pengali sebesar 3 (jadi, segitiga hasil dilatasi ini lebih besar 3 kali dari segitiga awal).

b. reflect (50,100)



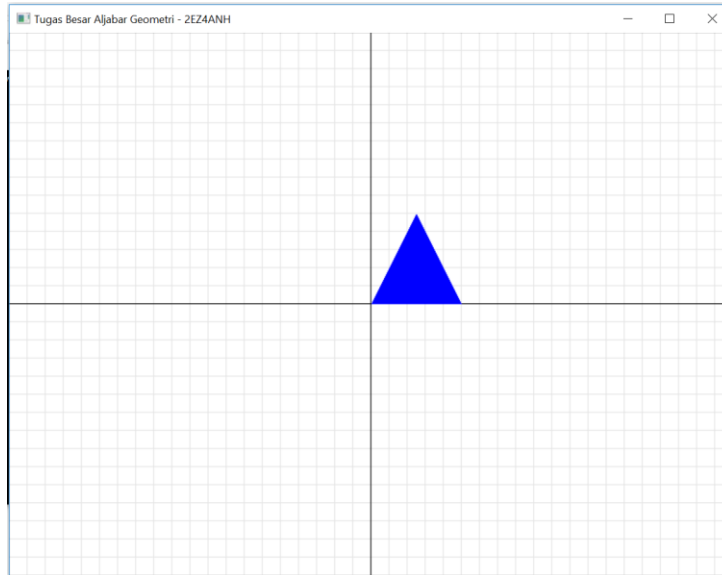
Command ini merefleksikan (mencerminkan) segitiga pada kondisi awal (ditunjukkan dengan garis-garis merah) terhadap titik $x=50$ dan $y=100$ yang ditunjukkan dengan titik merah, yang hasilnya dapat terlihat pada gambar diatas.

c. reflect $y=-x$



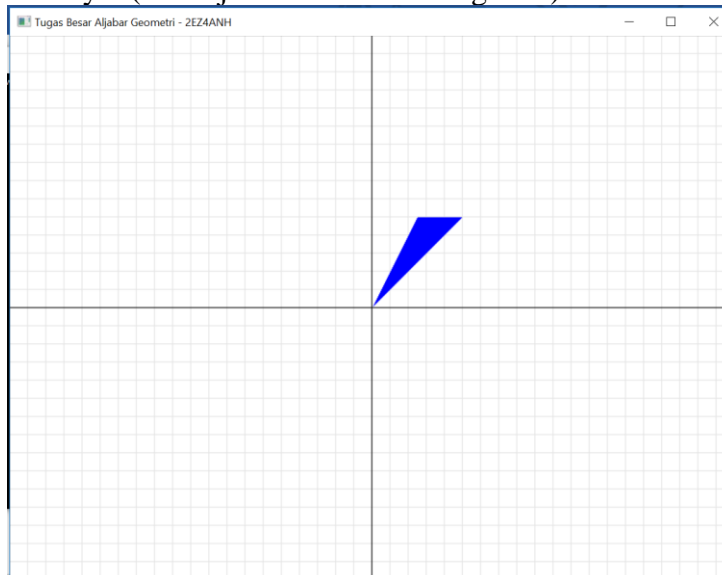
Command ini merefleksikan segitiga awal (ditunjukkan dengan garis merah tebal) terhadap garis $y=-x$ (ditunjukkan dengan titik-titik merah).

d. stretch y 2



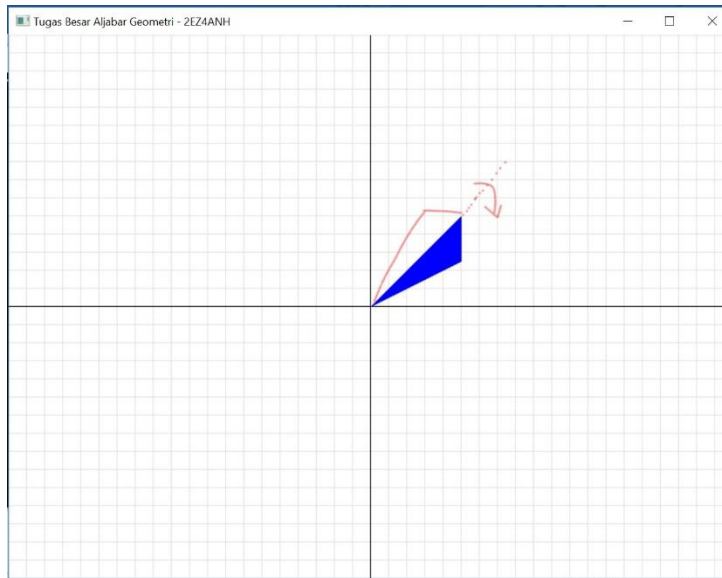
Command stretch ini menyebabkan bentuk awal segitiga di transformasi (diregangkan) sebesar factor skala $k = 2$, dan terhadap sumbu y . Jadi bentuk awal seperti dilebarkan secara vertikal (2 kali lebih lebar dari bentuk awal).

e. shear y 1 (melanjutkan command bagian d)



Setelah di stretch, segitiga tidak di reset dan dilanjutkan dengan melakukan shear (pergeseran) terhadap sumbu y dengan factor $k=1$.

f. reflect $y=x$ (melanjutkan command bagian d lalu e)



Melanjutkan command stretch dan shear pada bagian d dan e tanpa reset, lalu dilakukan refleksi bentuk tadi terhadap garis $y=x$ seperti ditunjukkan pada gambar.

BAB V

KESIMPULAN DAN SARAN

Dari seluruh pengerjaan tugas besar mengenai transformasi linier dan matriks transformasi yang menyertainya, lebih tepatnya simulasi transformasi linier pada bidang 2D dengan menggunakan OpenGL API, dapat disimpulkan bahwa setiap objek dalam bidang datar (atau lebih tepatnya bidang dua dimensi) dapat ditransformasi sedemikian rupa dengan berbagai jenis transformasi. Objek ini dapat berupa titik, garis, ataupun suatu bentuk poligon dengan berbagai sisi atau titik sudut sejumlah n (dimana n bilangan bulat lebih dari 2 untuk membentuk suatu poligon). Transformasi linier itu sendiri adalah perpindahan suatu objek dari satu ruang yang biasanya dinamakan dengan domain atau daerah asal ke ruang lain yang dinamakan kodomain atau daerah hasil.

Transformasi-transformasi yang dapat dilakukan program ini adalah translasi, dilatasi (perbesaran/perkalian), refleksi (pencerminan), rotasi sudut terhadap suatu titik, *shear* (pergeseran), *stretch* (peregangan), dan *custom transformation*. Selain itu juga dapat dilakukan transformasi dengan beberapa perintah sekaligus, dan juga perintah reset untuk mengembalikan ke semula.

Transformasi linier yang kami lakukan pada tugas ini adalah pada bidang 2 dimensi, untuk pengembangannya dapat dilakukan modifikasi program agar dapat melakukan transformasi pada bidang 3 dimensi. Selain itu, dengan mengerjakan tugas besar ini, kami menjadi lebih dimudahkan dalam mengerti materi Transformasi Linier pada mata kuliah Aljabar Geometri.

BAB VI

DAFTAR PUSTAKA

Tutorial menggunakan OpenGL dengan Bahasa Python:

<http://noobtuts.com/python/opengl-introduction>

Sumber teori singkat mengenai Transformasi Linier dan Matriks Transformasi:

<http://amriesagala.blogspot.co.id/2014/12/transformasi-linear.html>

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

http://file.upi.edu/Direktori/FPMIPA/PRODI_ILMU_KOMPUTER/HERI_SUTARNO/Aljabar_linier/BAB_7_TRANSFORMASI_LINEAR.pdf

<http://nonamirza03.blogspot.co.id/2014/12/transformasi-linier.html>

Sumber teori singkat mengenai OpenGL:

<http://agussale.com/penjelasan-mengenai-apa-itu-opengl>