

**LAPORAN TUGAS BESAR II**  
**IF2123 : ALJABAR GEOMETRI**  
**SIMULASI TRANSFORMASI LINIER PADA BIDANG 2D DAN 3D**  
**DENGAN MENGGUNAKAN *OPENGL API***



Dibuat Oleh :  
Kelompok Sabebe 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 :**

<b>Bab I. Deskripsi Permasalahan</b>	2
<b>Bab II. Teori Singkat</b>	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	14
<b>Bab III. Implementasi Program dan Pembagian Kerja</b>	14
main.py	14
transformasi.py	14
ProcInput.py	14
config.py	14
render.py	15
bentuk.py	15
handler.py	15
animasi.py	15
Pembagian Tugas	15
<b>Bab IV. Eksperimen</b>	15
Transformasi di ruang 2D	15
Transformasi di ruang 3D	15
Input Salah	18
<b>Bab V. Kesimpulan, Saran, dan Refleksi</b>	21
<b>Daftar Pustaka</b>	22

## 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.

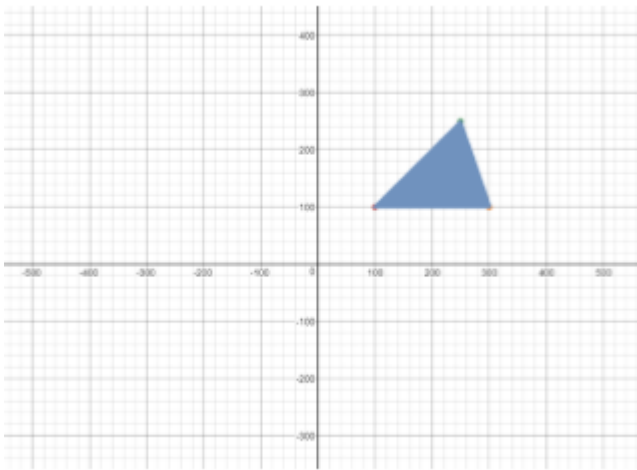
Berikutnya, program dapat menerima input yang didefinisikan pada tabel dibawah.

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> <b>	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: <b>x, y, y=x, y=-x</b> , atau <b>(a,b)</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> <b> <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 :

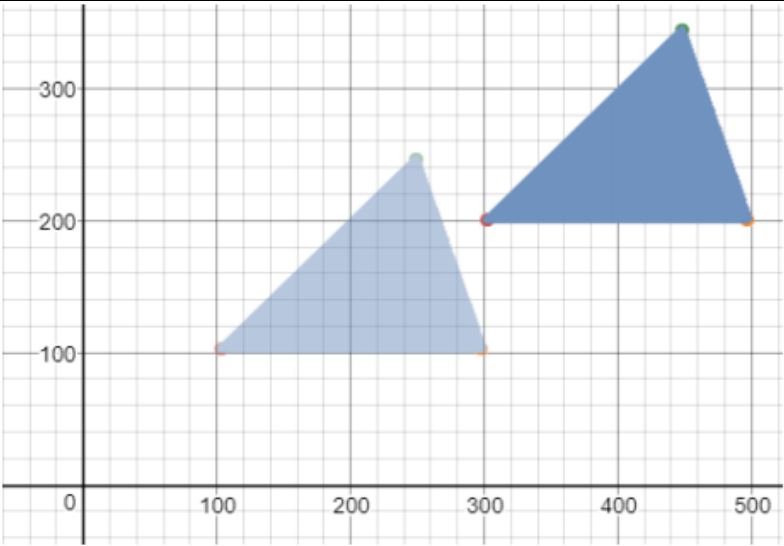
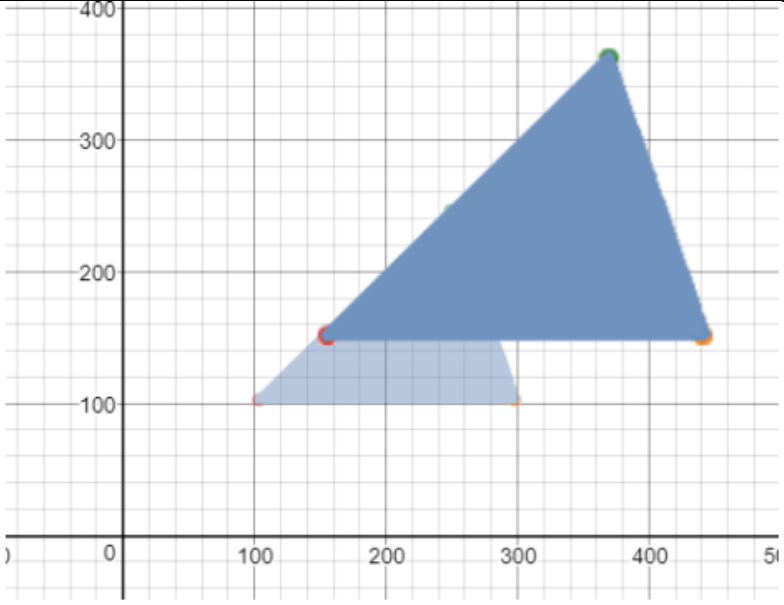
Saat program baru dimulai:

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

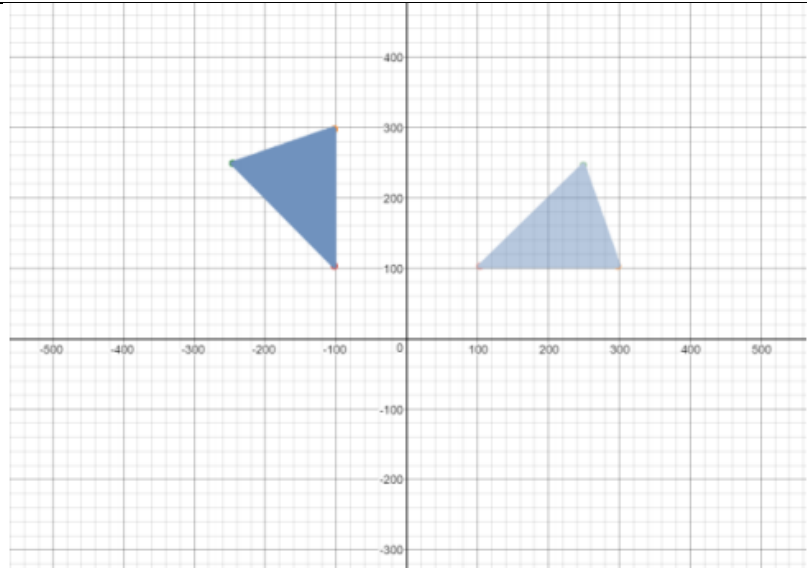
Perhatikan bahwa garis-garis tipis pada gambar diatas *tidak perlu diimplementasikan pada program*.

Saat program sudah membentuk objek dari input awal:

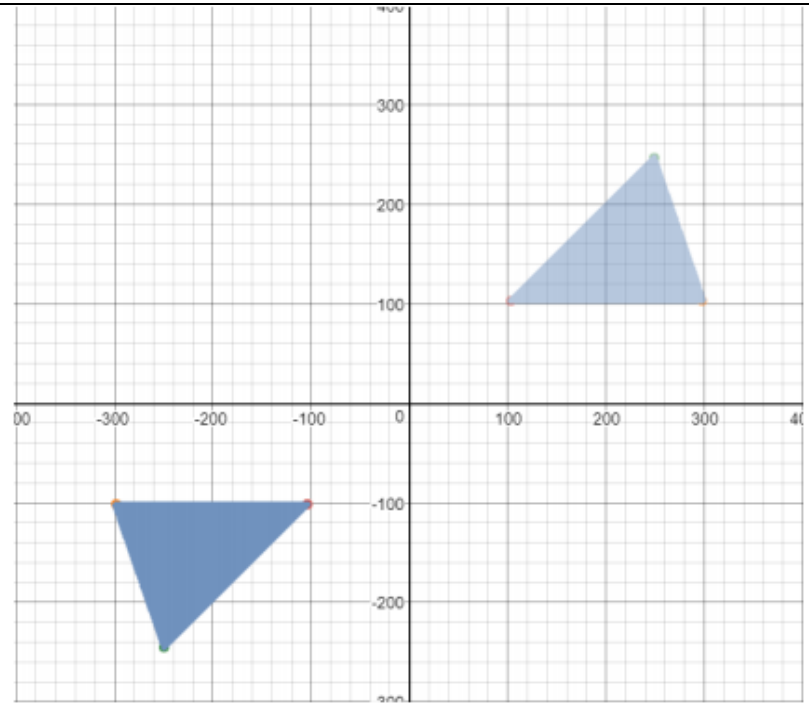
**Catatan:** Perhatikan bahwa gambar bidang yang transparan menunjukkan kondisi bidang *sebelum* input diberi, sedangkan bidang yang tidak transparan menunjukkan kondisi bidang *setelah* program mengeksekusi operasi dari input (bidang yang transparan *tidak ditampilkan* pada program).

Input	Output
translate 200 100	
dilate 1.5	

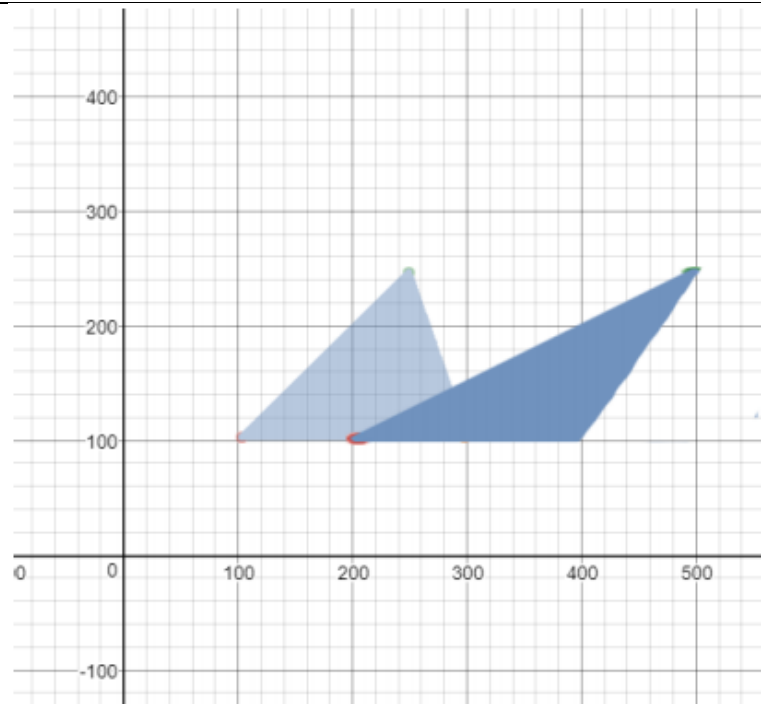
rotate 90 0 0



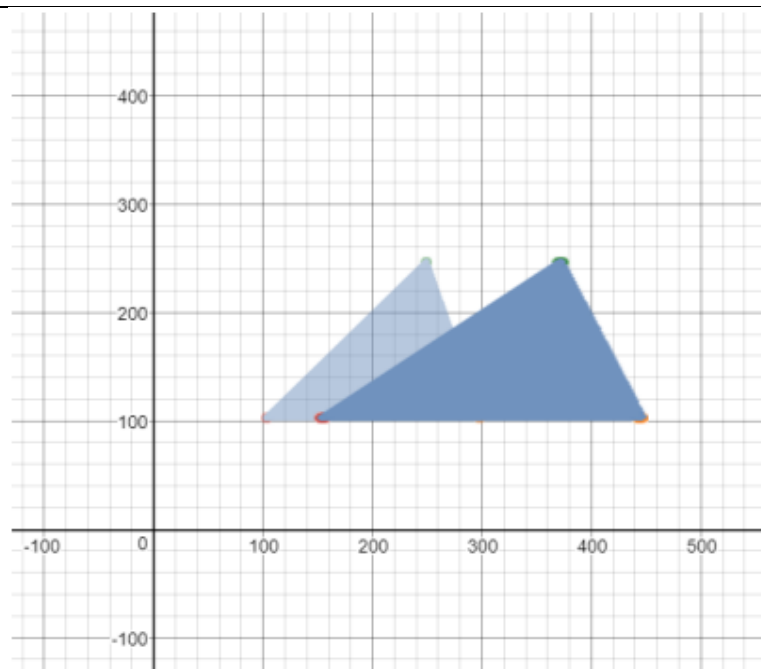
reflect (0,0)

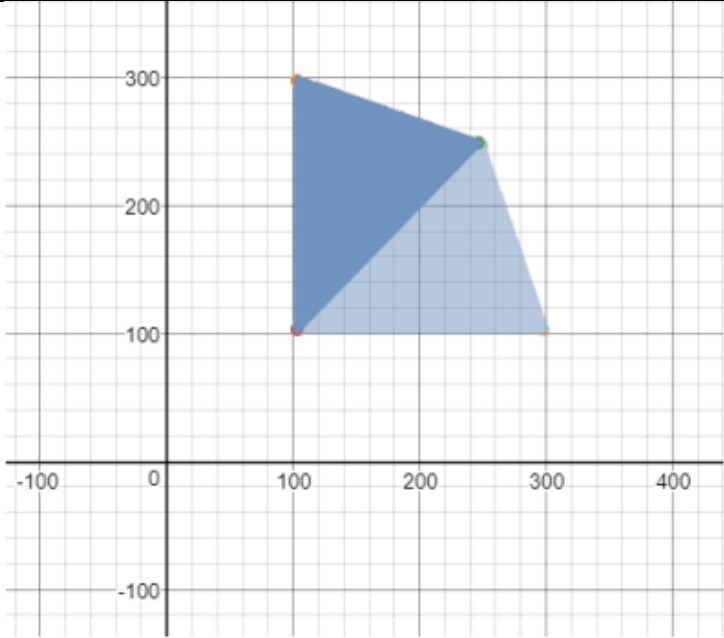



shear x 1



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](https://www.khronos.org/opengl/wiki/Getting_Started)
3. Library: <https://www.opengl.org/sdk/libs/>
4. Wikipedia: <https://en.wikipedia.org/wiki/OpenGL>

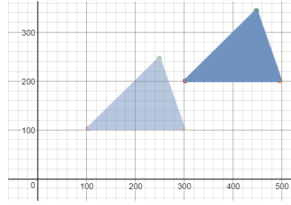
## Bab II. Teori Singkat

Transformasi Geometri adalah pemetaan titik-titik lama pembentuk suatu objek untuk mendapatkan titik-titik baru dari perkalian yang dilakukan terhadap matriks transformasi itu sendiri.

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.



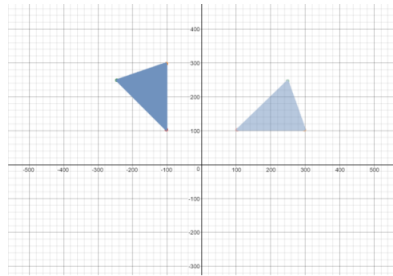
Pada umumnya, translasi yang dilakukan dengan matriks melakukan penjumlahan terhadap matriks yang sudah ada, tetapi untuk tugas besar kali ini, matriks yang kami pakai adalah matriks perkalian. Translasi titik-titik di ruang-2 dan ruang-3 memakai matriks tersebut. Matriks tersebut dapat digeneralisasi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



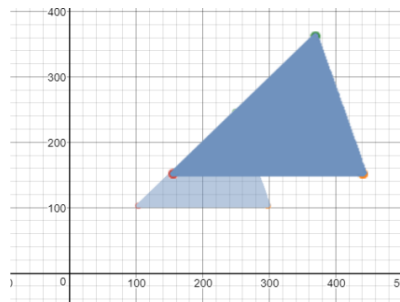
Adapula matriks transformasi yang dapat berlaku pada ruang 2 dimensi dan 3 dimensi. Matriks *general* tersebut adalah :

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

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.



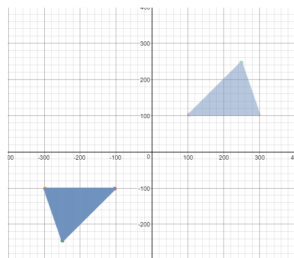
Pada algoritma transformasi dilatasi yang kami buat, kami juga memakai matriks. Matriks tersebut adalah :

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

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.

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

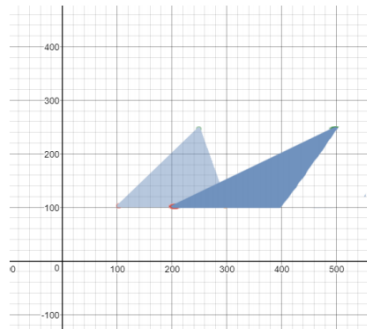
$$R_{(x=y)} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} R_{(x=-y)} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} R_{(a,b)} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 2a & 2b & 1 & 0 \\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

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

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

## 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 :

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

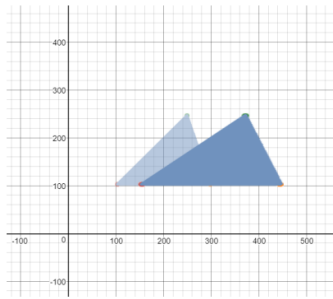
Pada implementasi 3 dimensi, transformasi *shear* dapat dilakukan dengan pendekatan  $xy$ ,  $xz$ ,  $yx$ ,  $yz$ ,  $zx$ , dan  $zy$ . Untuk setiap perhitungan, misalnya, apabila pendekatan yang digunakan

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 :

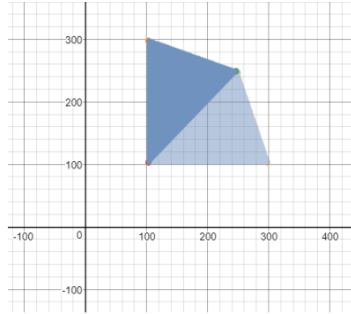
$$St_x = \begin{bmatrix} k & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad St_y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & k & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

$\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{bmatrix}$  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

$\begin{bmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$  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 transformasi 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 dan Pembagian Kerja

**main.py** : 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. Untuk dapat tetap menggerakkan kamera dan tetap bisa melakukan input ke *command window* selagi animasi berjalan dalam *GUI*, dipakai system *multithreading*, dalam hal ini, memakai 2 thread. Thread 1 untuk *command window*, dan Thread 2 untuk *window GUI*.

**transformasi.py** : 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 ini.

**ProcInput.py** : 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*.

**config.py** : 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.

**render.py** : 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*.

**bentuk.py** : 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.

**handler.py** : 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)).

**animasi.py** : 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”. Apabila multiple dihentikan di tengah, semua transformasi akan di-cancel dan animasi tidak dilakukan.

#### **Pembagian Tugas :**

Anggota	Tugas
Aditya Putra Santosa / 13517013	Pembuatan sistem utama program, GUI ,dan animasi objek.
Harry Rahmadi Munly /13517033	Pembuatan fungsi-fungsi transformasi objek 2D dan 3D.
Leonardo / 13517048	Pembuatan prosedur input, laporan, dan readme.

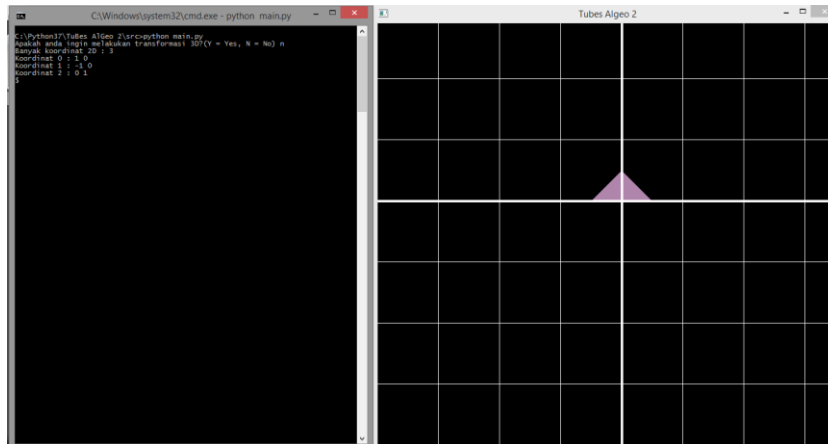


## Bab IV. Eksperimen

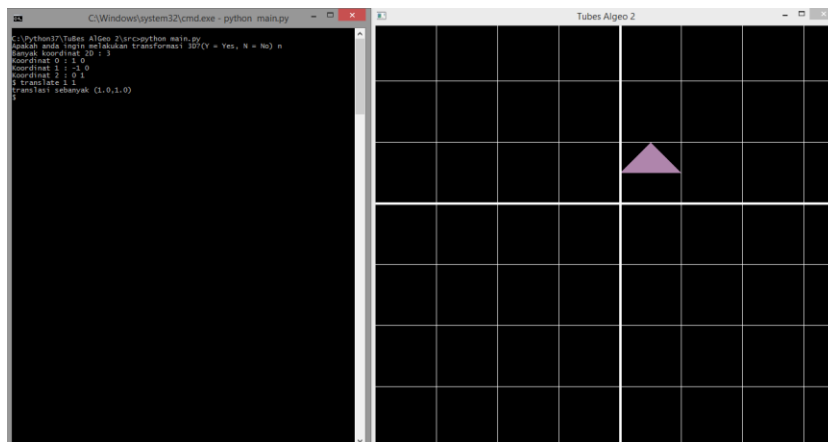
Berikut akan ditampilkan tangkapan-tangkapan layar berupa contoh-contoh penggunaan program kelompok kami. Akan dipisah 2 bagian besar, bagian transformasi di ruang 2 dimensi, dan transformasi di ruang 3 dimensi. (Note : animasi pada program tidak dapat ditunjukkan pada laporan)

Transformasi di ruang 2D :

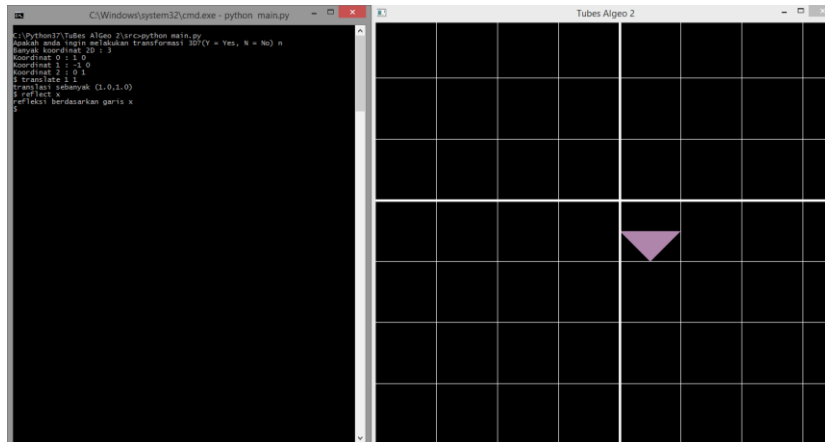
1. Memasukkan input (3 titik untuk membuat segitiga)



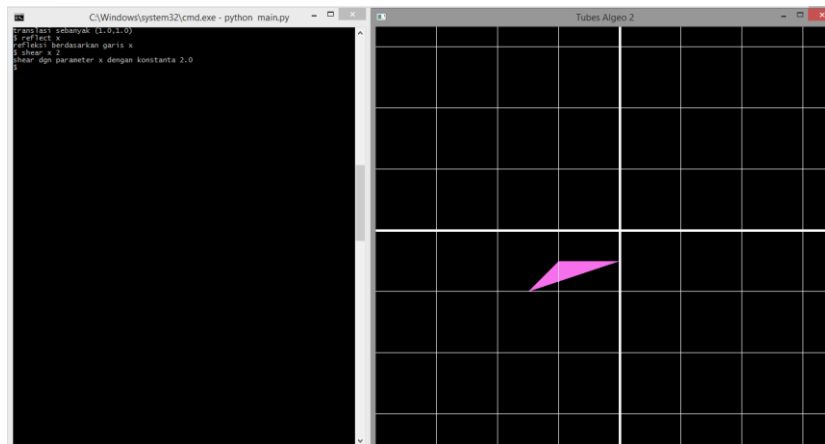
2. translate 1 1



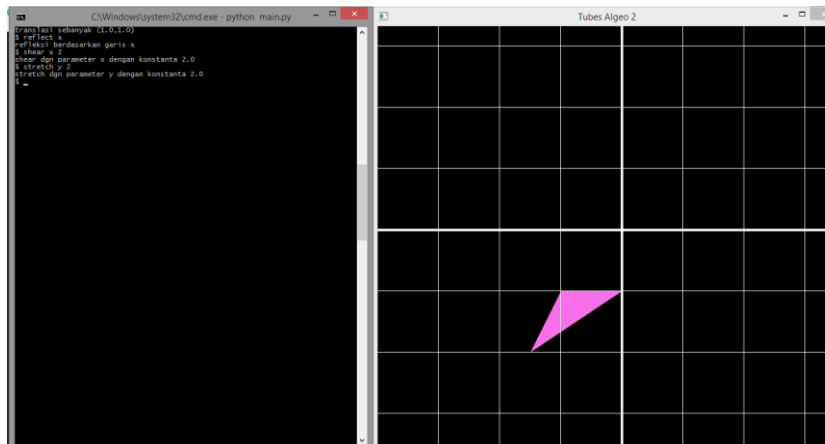
### 3. reflect x



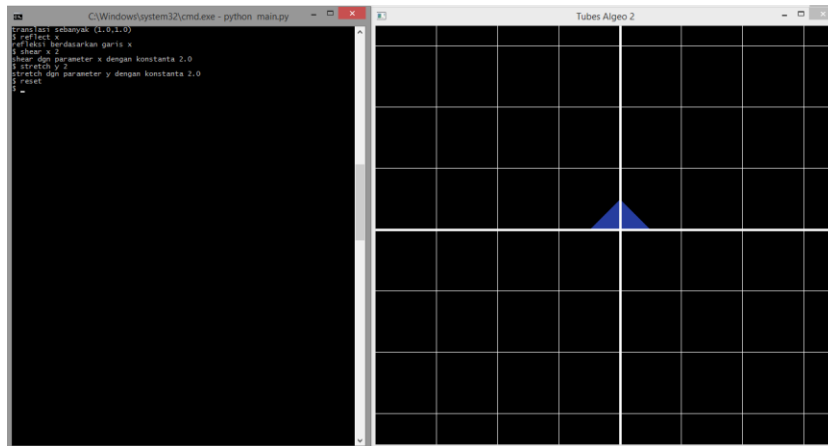
### 4. shear x 2



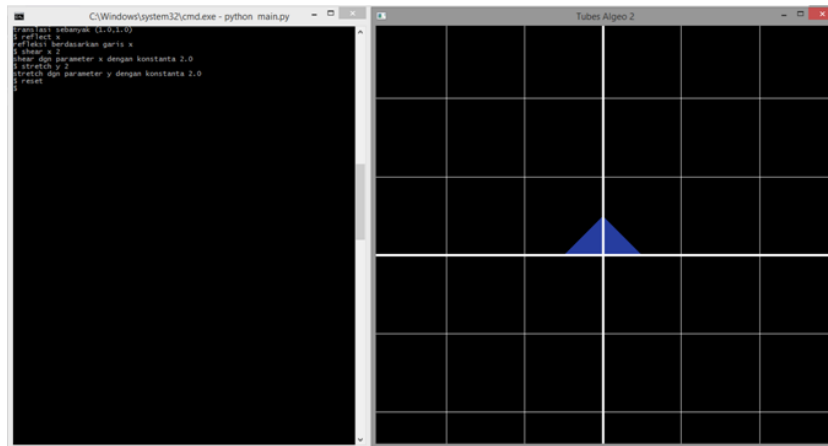
### 5. stretch y 2



## 6. reset

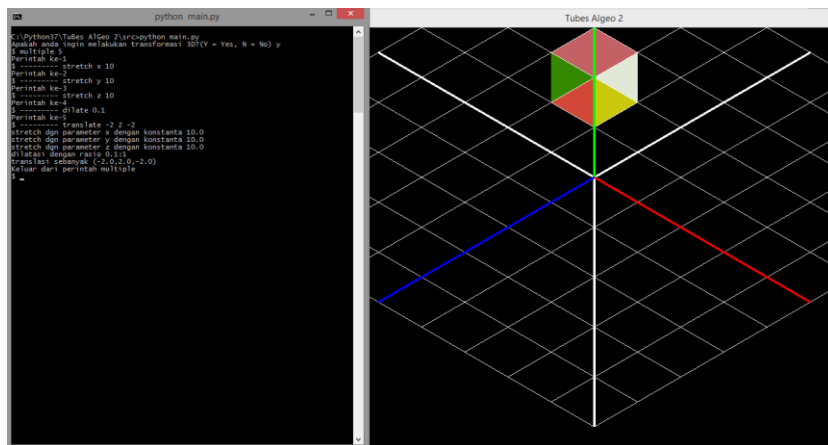


## 7. zoom in kamera

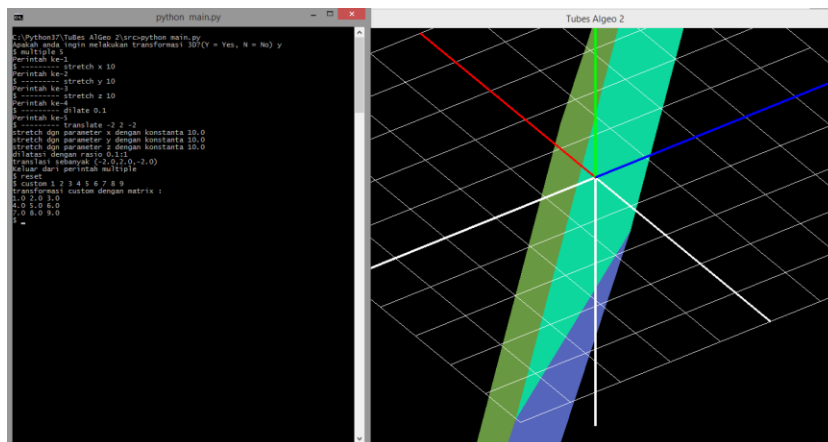


Transformasi di ruang 3D : (window terlalu besar sehingga ditangkap di bagian pentingnya)

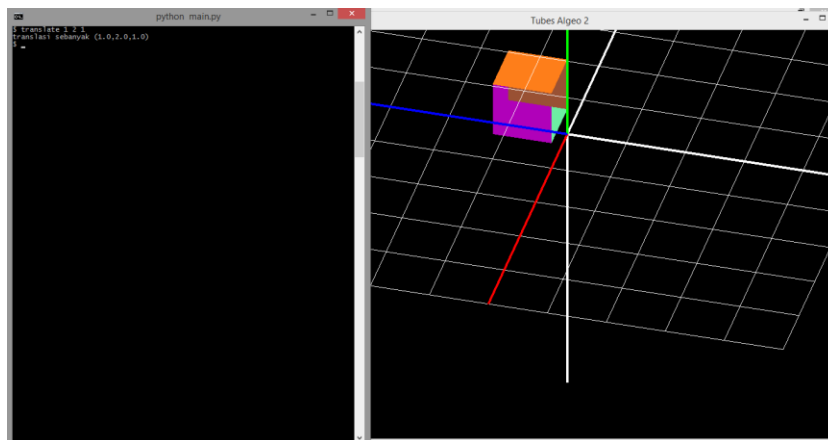
1. multiple 5



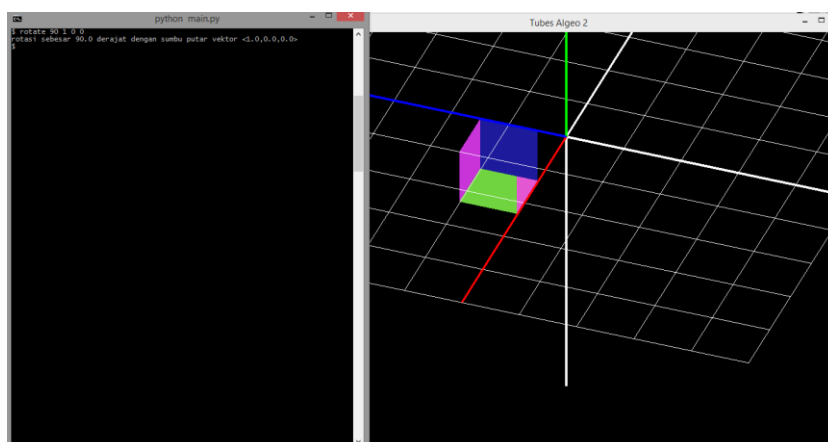
2. custom 1 2 3 4 5 6 7 8 9 (ditambah reset dan rotasi kamera)



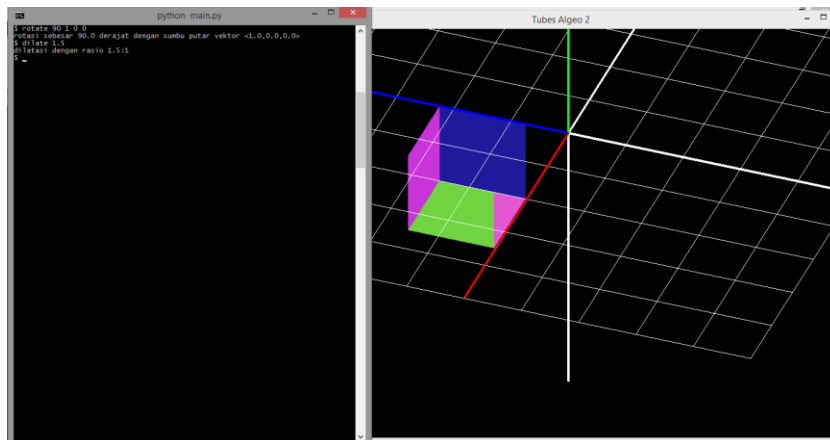
3. translate 1 2 1 (ditambah reset dan rotasi kamera)



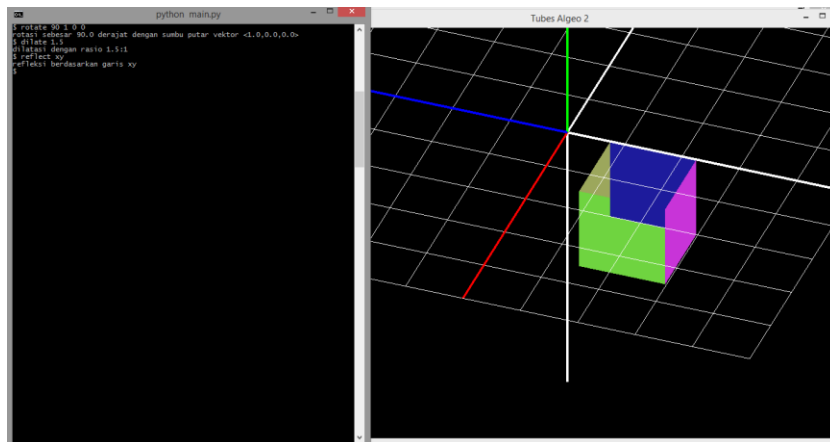
4. rotate 90 1 0 0



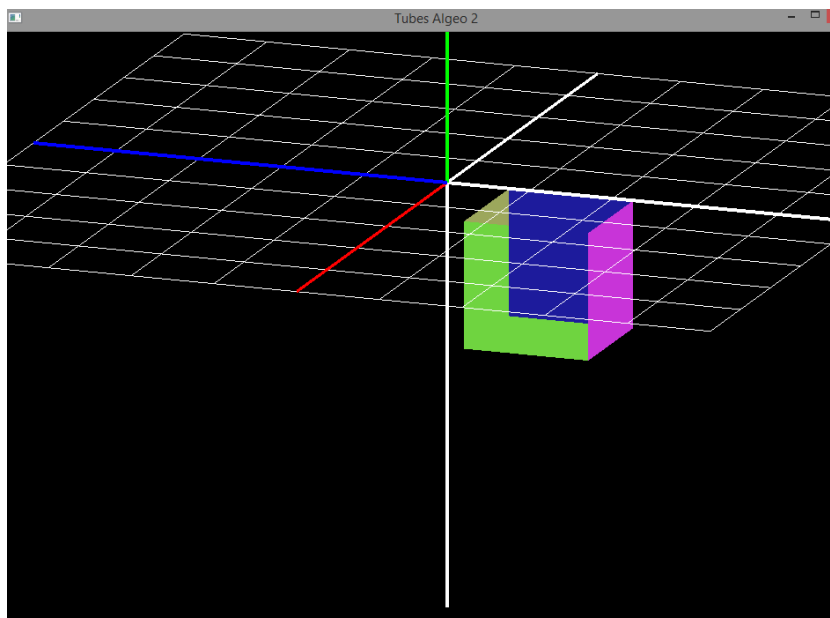
5. dilate 1.5



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*.
2. Membebaskan penggunaan bahasa implementasi, karena didapat banyak individu yang merasa lebih bisa dan lebih optimal saat menggunakan bahasa lain.

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. Tidak pernah lagi menggunakan warna transparan untuk pewarnaan bidang menggunakan OpenGL. Karena adanya *bug* di OpenGL yang membuat warna tidak bisa *diconvert* kembali 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://blog.ruangguru.com/pengertian-dan-jenis-jenis-transformasi-geometri>
4. <http://www.technologyuk.net/mathematics/geometry/transformations.shtml>
5. [https://www.khronos.org/opengl/wiki/Getting\\_Started](https://www.khronos.org/opengl/wiki/Getting_Started)