

Laporan Tugas Besar 2 IF2123 Aljabar Geometri Simulasi Transformasi Linier pada Bidang 2D dan 3D dengan Menggunakan OpenGL API



Disusun Oleh :

Muhammad Asyraf Desanto	13517027
Jeremy Arden Hartono	13517101
Kintan Sekar Adinda	13517102

**Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Bandung
2018**

DAFTAR ISI

DAFTAR ISI	2
BAB 1 DESKRIPSI MASALAH	3
BAB 2 TEORI SINGKAT	7
2.1 Transformasi Linear	7
2.2 Transformasi Matriks	9
2.3 OpenGL	13
BAB 3 IMPLEMENTASI PROGRAM	14
3.1 2d.py	14
3.2 3d.py	17
3.3 transformasi2d.py	19
3.4 transformasi3d.py	21
BAB 4 EKSPERIMEN	25
BAB 5 KESIMPULAN	35
DAFTAR PUSTAKA	36

BAB 1

DESKRIPSI MASALAH

Pada tugas kali ini, mahasiswa diminta membuat program yang mensimulasikan transformasi linier untuk melakukan operasi translasi, refleksi, dilatasi, rotasi, dan sebagainya pada sebuah objek 2D dan 3D. Objek dibuat dengan mendefinisikan sekumpulan titik sudut lalu membuat bidang 2D/3D dari titik-titik tersebut. Contoh objek 2D: segitiga, segiempat, polygon segi-n, lingkaran, rumah, gedung, mobil, komputer, lemari, dsb. Contoh objek 3D: kubus, pyramid, silinder, terompet, dll. Program akan memiliki dua buah window, window pertama (command prompt) berfungsi untuk menerima input dari user, sedangkan window kedua (GUI) berfungsi untuk menampilkan output berdasarkan input dari user. Kedua window ini muncul ketika user membuka file executable.

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

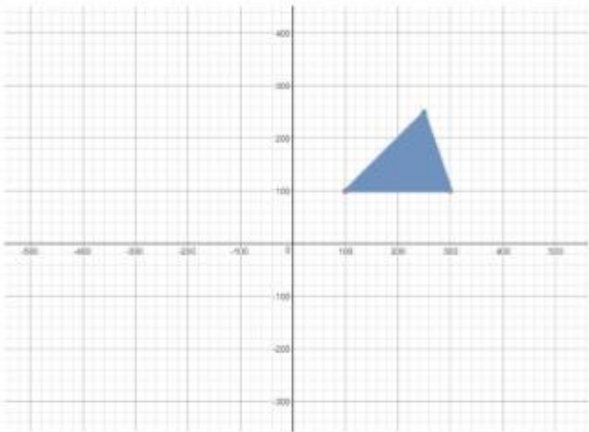
Input Keterangan

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 param 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 shear pada objek. Nilai param dapat berupa x (terhadap sumbu x) atau y (terhadap sumbu y). Nilai k adalah faktor shear.
stretch <param> <k>	Melakukan operasi stretch pada objek. Nilai param dapat berupa x (terhadap sumbu x) atau y (terhadap sumbu y). Nilai k adalah faktor stretch.
custom <a> <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 translate, rotate, shear, dll tetapi bukan multiple, reset, exit.

reset	Mengembalikan objek pada kondisi awal objek didefinisikan.
exit	Keluar dari program.

Contoh I/O program :

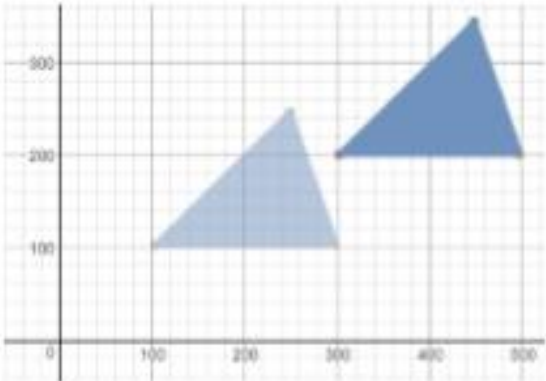
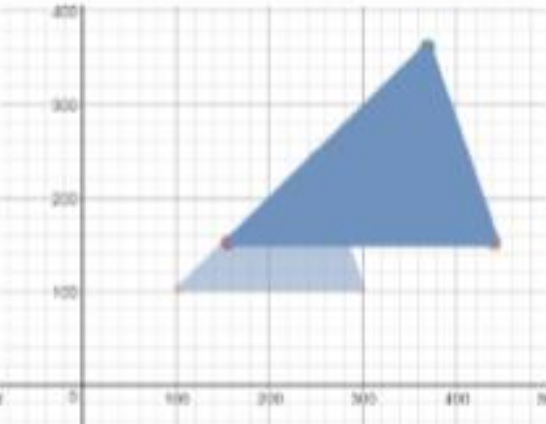
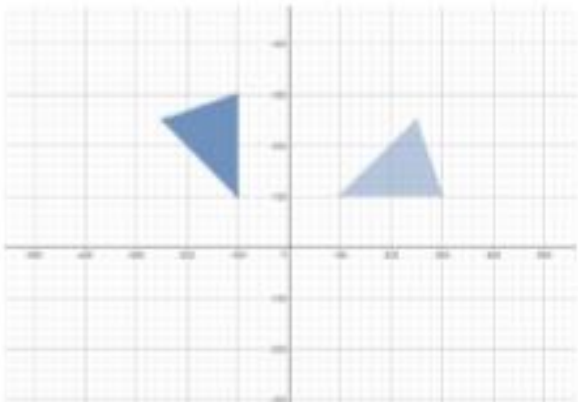
Saat program baru dimulai:

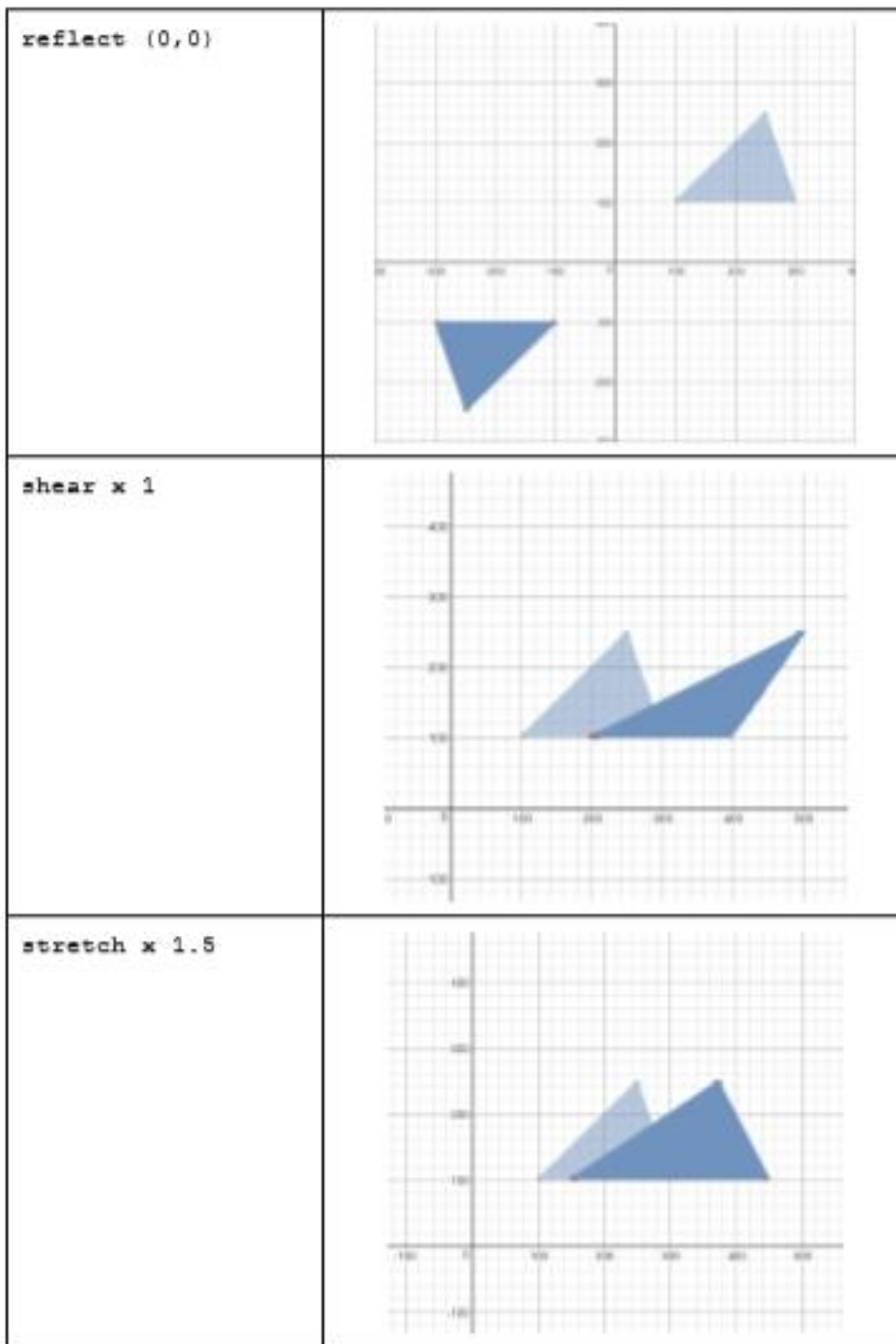
Input	Output
3 100,100 250,250 300,100	

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
<code>translate 200 100</code>	
<code>dilate 1.5</code>	
<code>rotate 90 0 0</code>	



BAB 2

TEORI SINGKAT

2.1 Transformasi Linear

2.1.1. Ide Transformasi Linear

Ketika suatu matriks A dikalikan dengan suatu vektor v , A mentransformasi v menjadi vektor Av , sehingga dihasilkan $T(v)=Av$. Transformasi T mengikuti ide yang sama seperti fungsi. Tujuan yang lebih jauh lagi adalah untuk melihat semua v secara bersamaan. Kita mentransformasikan seluruh ruang V ketika kita mengalikan setiap v dengan A .

Matriks A mentransformasikan v menjadi Av . Matriks A mentransformasikan w menjadi Aw . Lalu kita tahu yang terjadi adalah $u=v+w$. Bisa dipastikan bahwa $Au=Av+Aw$. Perkalian matriks $T(v)=Av$ menghasilkan **transformasi linear**:

Transformasi T menghasilkan $T(v)$ untuk setiap input vektor v dalam V . Transformasinya **linear** jika v dan w memenuhi:

$$(a) \ T(v+w)=T(v)+T(w) \quad (b) \ T(cv)=cT(v) \text{ untuk semua } c.$$

Jika input $v=0$, outputnya pasti $T(v)=0$. Kita kombinasikan (a) dan (b) menjadi satu:

Transformasi linear $T(cv+dw)$ **harus sama dengan** $cT(v)+dT(w)$.

Transformasi linear sangat terbatas. Misalnya transformasi T adalah menambahkan u_0 ke tiap vektor. Lalu $T(v)=v+u_0$ dan $T(w)=w+u_0$. Hal ini tidak linear karena jika $v+w$ ditransformasikan oleh T akan menghasilkan $v+w+u_0$. Hal tersebut tidak sama dengan $T(v)+T(w)$:

Transformasi tidak linear karena $v+w+u_0$ bukan $T(v)+T(w)=v+u_0+w+u_0$.

Pengecualian adalah ketika $u_0=0$. Transformasinya menjadi $T(v)=v$. ini adalah **transformasi identitas** (tidak ada yang berubah). Hal ini jelas-jelas linear. Dalam kasus ini input ruang V sama dengan output ruang W .

Garis ke Garis, Segitiga ke Segitiga

Gambar 7.1 menunjukkan garis dari v ke w dalam ruang input. Gambar tersebut juga menunjukkan garis dari $T(v)$ ke $T(w)$ dalam ruang output. **Suatu titik dalam ruang input menjadi suatu ke titik yang setara dalam ruang output**. Contohnya titik tengah $u = \frac{1}{2}v + \frac{1}{2}w$ menjadi titik tengah $T(u) = \frac{1}{2}T(v) + \frac{1}{2}T(w)$.

Gambar kedua mengubah dimensinya. Sekarang kita memiliki tiga sudut v_1, v_2, v_3 . Input-input tersebut menghasilkan tiga output $T(v_1), T(v_2), T(v_3)$. Segitiga input menjadi segitiga output. Titik-titik dalam ruang input menjadi titik-titik yang setara dalam ruang output (sepanjang sisi,

dan di antara sisi). Titik tengah $u = \frac{1}{3}(v_1 + v_2 + v_3)$ menjadi titik tengah $T(u) = \frac{1}{3}(T(v_1) + T(v_2) + T(v_3))$.

Aturan linearitas juga memenuhi kombinasi tiga vektor atau n vektor:

Linearitas $u = c_1v_1 + c_2v_2 + \dots + c_nv_n$ bertransformasi menjadi $T(u) = c_1 T(v_1) + c_2 T(v_2) + \dots + c_n T(v_n)$ (1)

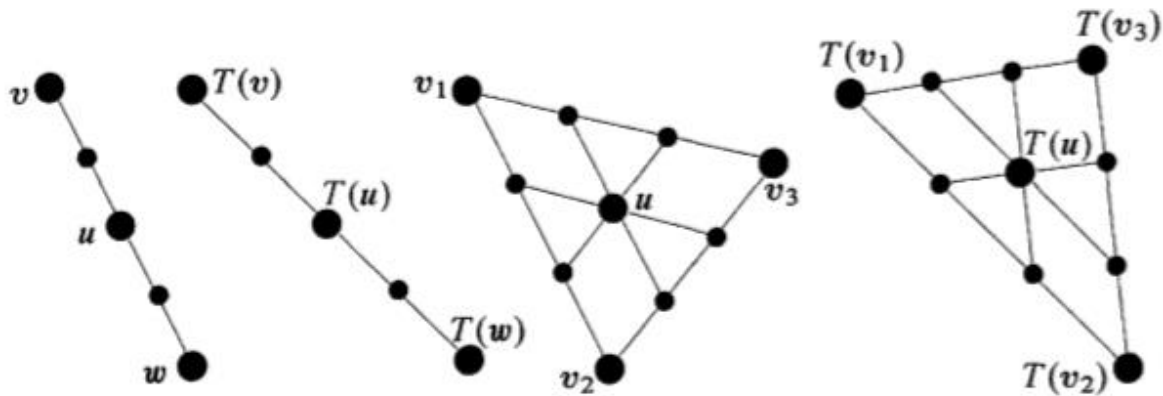


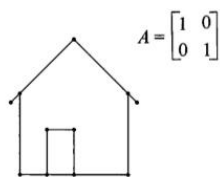
Figure 7.1: Lines to lines, equal spacing to equal spacing, $u = 0$ to $T(u) = 0$.

Transformasi Linear Bidang

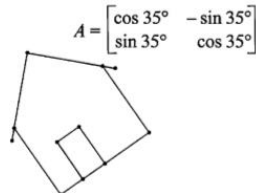
Ketika sebuah matriks A 2×2 dikalikan dengan semua vektor dalam \mathbb{R}^2 , kita bisa lihat contoh berikut. Gambar “rumah” memiliki sebelas titik sudut. Sebelas vektor v tersebut ditransformasikan menjadi sebelas vektor Av . Garis lurus di antara vektor-vektor v menjadi garis lurus di antara vektor-vektor Av . Mengaplikasikan A ke suatu gambar menghasilkan suatu gambar baru.

Berikut adalah contoh dari transformasi bidang oleh beberapa matriks. Kolom H adalah sebelas titik sudut dari gambar rumah. Sebelas titik pada matriks rumah H dikalikan oleh A untuk menghasilkan titik sudut AH dari rumah yang baru.

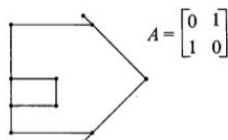
House matrix $H = \begin{bmatrix} -6 & -6 & -7 & 0 & 7 & 6 & 6 & -3 & -3 & 0 & 0 & -6 \\ -7 & 2 & 1 & 8 & 1 & 2 & -7 & -7 & -2 & -2 & -7 & -7 \end{bmatrix}$.



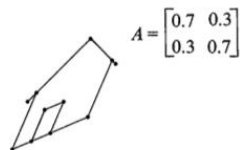
$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



$$A = \begin{bmatrix} \cos 35^\circ & -\sin 35^\circ \\ \sin 35^\circ & \cos 35^\circ \end{bmatrix}$$



$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$



$$A = \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix}$$

Figure 7.2: Linear transformations of a house drawn by `plot2d(A * H)`.

2.1.2. Matriks Transformasi Linear

Untuk vektor-vektor kolom biasa, input v ada dalam $V=R^n$ dan output $T(v)$ ada dalam $W=R^m$. Matriks A untuk transformasi T ini akan menjadi $m \times n$.

Kata kunci dari bagian ini

Jika diketahui $T(v_1), \dots, T(v_n)$ untuk vektor-vektor basis v_1, \dots, v_n .

Maka linearitas menghasilkan $T(v)$ untuk setiap input selain vektor v .

Karena Setiap v merupakan kombinasi $c_1v_1 + \dots + c_nv_n$ dari vektor-vektor basis v_i . Karena T adalah transformasi linear, $T(v)$ harus merupakan kombinasi yang sama $c_1T(v_1) + \dots + c_nT(v_n)$ dari yang diketahui output $T(v_i)$.

2.2 Transformasi Matriks

2.2.1 Matriks Refleksi

Salah satu dari operator matriks dasar pada R^2 dan R^3 adalah yang memetakan masing-masing titik menjadi gambar yang simetris terhadap suatu garis atau bidang; disebut matriks refleksi. Tabel 1 menunjukkan matriks standar untuk refleksi terhadap sumbu koordinat dalam R^2 , dan Tabel 2 menunjukkan matriks-matriks standar untuk refleksi terhadap bidang koordinat dalam R^3 . Pada masing-masing kasus, matriks standar didapat dari mencari gambar vektor-vektor basis, mengubah gambar-gambar tersebut menjadi vektor kolom, dan menggunakan vektor-vektor kolom secara berturut-turut tersebut sebagai matriks standar.

Table 1

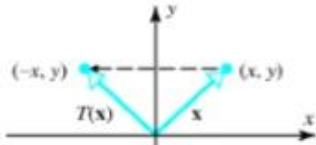
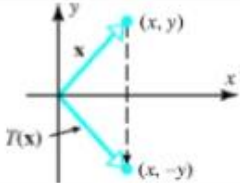
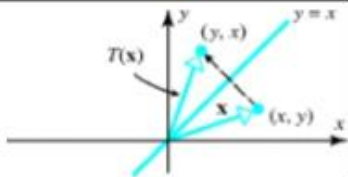
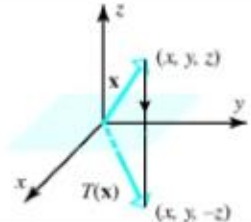
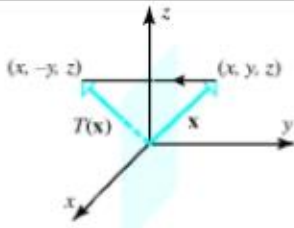
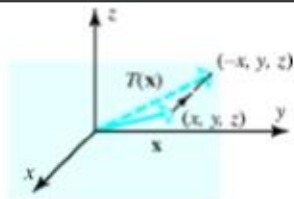
Operator	Illustration	Images of e_1 and e_2	Standard Matrix
Reflection about the y -axis $T(x, y) = (-x, y)$		$T(e_1) = T(1, 0) = (-1, 0)$ $T(e_2) = T(0, 1) = (0, 1)$	$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$
Reflection about the x -axis $T(x, y) = (x, -y)$		$T(e_1) = T(1, 0) = (1, 0)$ $T(e_2) = T(0, 1) = (0, -1)$	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Reflection about the line $y = x$ $T(x, y) = (y, x)$		$T(e_1) = T(1, 0) = (0, 1)$ $T(e_2) = T(0, 1) = (1, 0)$	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

Table 2

Operator	Illustration	e_1, e_2, e_3	Standard Matrix
Reflection about the xy -plane $T(x, y, z) = (x, y, -z)$		$T(e_1) = T(1, 0, 0) = (1, 0, 0)$ $T(e_2) = T(0, 1, 0) = (0, 1, 0)$ $T(e_3) = T(0, 0, 1) = (0, 0, -1)$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$
Reflection about the xz -plane $T(x, y, z) = (x, -y, z)$		$T(e_1) = T(1, 0, 0) = (1, 0, 0)$ $T(e_2) = T(0, 1, 0) = (0, -1, 0)$ $T(e_3) = T(0, 0, 1) = (0, 0, 1)$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Reflection about the yz -plane $T(x, y, z) = (-x, y, z)$		$T(e_1) = T(1, 0, 0) = (-1, 0, 0)$ $T(e_2) = T(0, 1, 0) = (0, 1, 0)$ $T(e_3) = T(0, 0, 1) = (0, 0, 1)$	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

2.2.2 Matriks Refleksi

Rotasi dalam R^2

Operator matriks pada R^2 dan R^3 yang memindahkan titik-titik sepanjang busur melingkar disebut matriks rotasi. Seperti yang diilustrasikan dalam Figure 4.9.3, vektor-vektor basis standar adalah

$$T(e_1) = T(1, 0) = (\cos \theta, \sin \theta) \text{ and } T(e_2) = T(0, 1) = (-\sin \theta, \cos \theta)$$

Jadi matriks standar untuk T adalah

$$\begin{bmatrix} T(e_1) & T(e_2) \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$

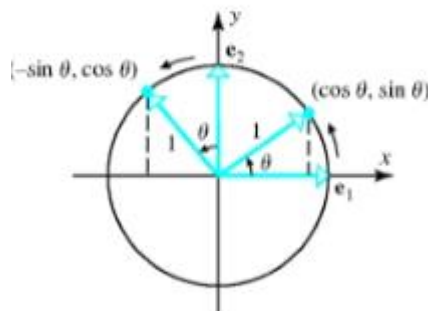


Figure 4.9.3

Agar dapat digunakan secara general kita akan menggunakan operator rotasi R_θ dan menyebut

$$R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

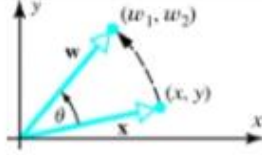
sebagai **matriks rotasi** untuk R^2 . Jika $x=(x,y)$ adalah vektor dalam R^2 , dan jika $w=(w_1,w_2)$ adalah hasil setelah dirotasi, maka hubungan $w=R_\theta x$ dapat ditulis dalam bentuk

$$w_1 = x\cos\theta - y\sin\theta$$

$$w_2 = x\sin\theta + y\cos\theta$$

Hal-hal ini disebut **persamaan rotasi** untuk R^2 dan dirangkum dalam Tabel 5.

Table 5

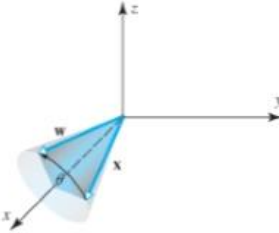
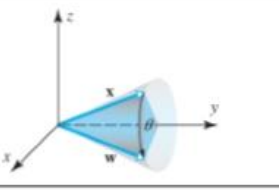
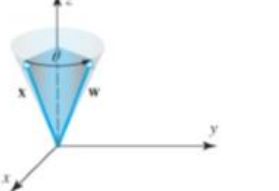
Operator	Illustration	Rotation Equations	Standard Matrix
Rotation through an angle θ		$w_1 = x\cos\theta - y\sin\theta$ $w_2 = x\sin\theta + y\cos\theta$	$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

Dalam bidang, sudut berlawanan arah jarum jam itu positif dan searah jarum jam sudutnya negative. Matriks rotasi searah jarum jam dengan $-\theta$ radian dapat didapat dengan mengganti θ dengan $-\theta$. Setelah disederhanakan akan menjadi

$$R_{-\theta} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

Operator matriks rotasi pada R^3 adalah operator matriks yang merotasi masing-masing vektor dalam R^3 terhadap sumbu rotasi yang melalui sudut θ . Dalam table 6 kita telah mendeskripsikan operator matriks rotasi pada R^3 yang sumbu rotasinya adalah sumbu koordinat positif.

Table 6

Operator	Illustration	Rotation Equations	Standard Matrix
Counterclockwise rotation about the positive x-axis through an angle θ		$w_1 = x$ $w_2 = y\cos\theta - z\sin\theta$ $w_3 = y\sin\theta + z\cos\theta$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$
Counterclockwise rotation about the positive y-axis through an angle θ		$w_1 = x\cos\theta + z\sin\theta$ $w_2 = y$ $w_3 = -x\sin\theta + z\cos\theta$	$\begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$
Counterclockwise rotation about the positive z-axis through an angle θ		$w_1 = x\cos\theta - y\sin\theta$ $w_2 = x\sin\theta + y\cos\theta$ $w_3 = z$	$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Untuk lebih lengkapnya, matriks standar untuk rotasi berlawanan arah jarum jam sebesar θ derajat dalam R^3 , dengan (a,b,c) adalah titik awal sebelum dirotasi, adalah

$$\begin{bmatrix} a^2(1 - \cos\theta) + \cos\theta & ab(1 - \cos\theta) - c\sin\theta & ac(1 - \cos\theta) + b\sin\theta \\ ab(1 - \cos\theta) + c\sin\theta & b^2(1 - \cos\theta) + \cos\theta & bc(1 - \cos\theta) - a\sin\theta \\ ac(1 - \cos\theta) - b\sin\theta & bc(1 - \cos\theta) + a\sin\theta & c^2(1 - \cos\theta) + \cos\theta \end{bmatrix}$$

2.2.3 Matriks Dilatasi

Jika k adalah scalar nonnegative, maka operator $T(x) = kx$ pada R^2 atau R^3 memiliki efek membesarkan atau mengecilkan panjang masing-masing vektor dengan factor k . Jika $0 \leq k < 1$ matriks operasi disebut kontraksi dengan factor k , dan jika $k > 1$ disebut dilatasi dengan factor k . Jika $k=1$ maka T adalah matriks identitas dan dapat dianggap kontraksi atau dilatasi. Table 7 dan Table 8 mengilustrasikan matriks operasi ini.

Table 7

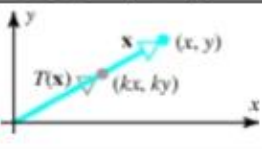
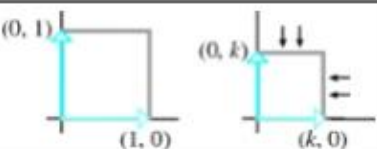
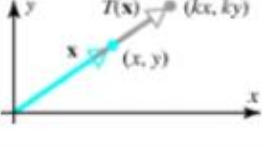
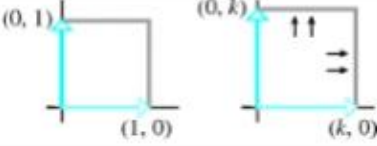
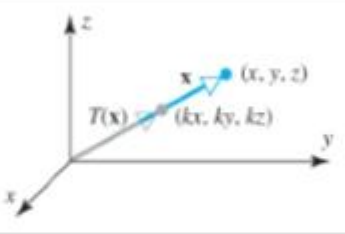
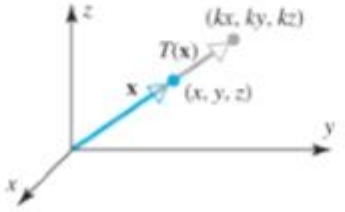
Operator	Illustration $T(x, y) = (kx, ky)$	Effect on the Standard Basis	Standard Matrix
Contraction with factor k on R^2 ($0 \leq k < 1$)			$\begin{bmatrix} k & 0 \\ 0 & k \end{bmatrix}$
Dilation with factor k on R^2 ($k > 1$)			

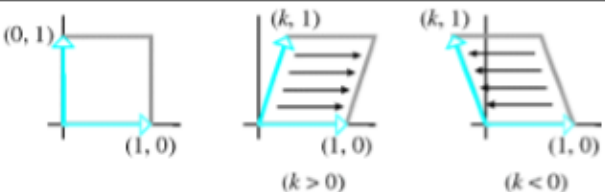
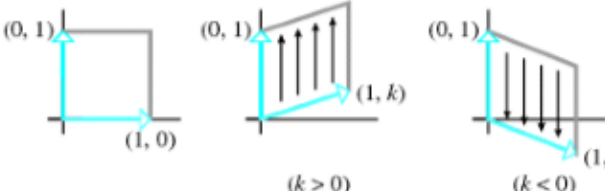
Table 8

Operator	Illustration $T(x, y, z) = (kx, ky, kz)$	Standard Matrix
Contraction with factor k on R^3 ($0 \leq k \leq 1$)		$\begin{bmatrix} k & 0 & 0 \\ 0 & k & 0 \\ 0 & 0 & k \end{bmatrix}$
Dilation with factor k on R^3 ($k \geq 1$)		

2.2.4 Matriks Shears

Matriks operator berbentuk $T(x,y)=(x+ky,y)$ disebut shear dalam arah x dengan faktor k , sementara $T(x,y)=(x,y+kx)$ disebut shear dalam arah y dengan faktor k . Tabel 10 menggambarkan informasi dasar tentang shear dalam R^2 .

Table 10

Operator	Effect on the Standard Basis	Standard Matrix
Shear of R^2 in the x -direction with factor k $T(x,y) = (x + ky, y)$	 <p>$(0, 1)$ $(1, 0)$ $(k, 1)$ $(1, 0)$ $(k > 0)$ $(k, 1)$ $(1, 0)$ $(k < 0)$</p>	$\begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix}$
Shear of R^2 in the y -direction with factor k $T(x,y) = (x, y + kx)$	 <p>$(0, 1)$ $(1, 0)$ $(0, 1)$ $(1, k)$ $(k > 0)$ $(0, 1)$ $(1, k)$ $(k < 0)$</p>	$\begin{bmatrix} 1 & 0 \\ k & 1 \end{bmatrix}$

2.3 OpenGL

OpenGL (Open Graphics Library) adalah spesifikasi standar yang mendefinisikan sebuah lintas-bahasa, lintas platform API untuk mengembangkan aplikasi yang menghasilkan grafis komputer 2D maupun 3D. Antarmuka terdiri dari lebih dari 250 panggilan fungsi yang berbeda yang dapat digunakan untuk menggambar tiga dimensi yang adegan-adegan kompleks dari bentuk-bentuk primitif sederhana. OpenGL dikembangkan oleh Silicon Graphics Inc (SGI) pada tahun 1992 dan secara luas digunakan dalam CAD, realitas maya, visualisasi ilmiah, visualisasi informasi, dan simulasi penerbangan. Hal ini juga digunakan dalam video game, di mana bersaing dengan Direct3D on Microsoft Windows platform (lihat vs OpenGL Direct3D). OpenGL dikelola oleh sebuah teknologi konsorsium nirlaba yaitu Khronos Group.

BAB 3

IMPLEMENTASI PROGRAM

Kami memiliki 4 source code, yaitu 2d.py, 3d.py, transformasi2d.py, dan transformasi3d.py. 2d.py berfungsi menerima input dari user untuk membentuk bidang, membentuk animasinya, dan berisi main program untuk menjalankan fungsi-fungsi transformasi khusus untuk 2 dimensi, sedangkan 3d.py memiliki fungsi yang sama namun khusus untuk 3 dimensi. transformasi2d.py berisi fungsi-fungsi transformasi khusus 2 dimensi, sedangkan transformasi3d.py berisi fungsi-fungsi yang sama tapi khusus 3 dimensi.

3.1 2d.py

Main Menu untuk 2 Dimensi

Fungsi ini berguna untuk menerima input dari user berupa bidang 2 dimensi.

```
def input2D():
    global N
    global matrix
    global oriMatrix

    Screen displays "Masukan banyak titik : "
    N is assigned by input from user in integer type.
    while (N less than or equal to 0):
        Screen displays "Masukan salah. Ulangi lagi : "
        N is assigned by input from user in integer type.
    matrix is matrix with N rows and 2 columns whose i as row index and j as column index.
    oriMatrix is also matrix with N rows and 2 columns whose i as row index and j as column index.
    for k as many as N:
        Screen displays "Masukan titik dalam format x,y : "
        point is assigned by input from user.
        point is two inputs from user which are x and y that is separated by ","
        x is in float type.
        y is in float type.
        matrix with row 0 and column k is assigned by x
        matrix with row 1 and column k is assigned by y
        oriMatrix with row 0 and column k is assigned by x
        oriMatrix with row 1 and column k is assigned by y
```

Fungsi berikut berguna untuk memasukkan command untuk 2 Dimensi, selain reset, list, dan exit, program lanjut ke animator.

```
def transformasi ():
    global N

    cmd = input(">> ")
    if cmd=="reset":
        tf2d.Reset2D(N,matrix,oriMatrix)
    elif cmd=="list":
        tf2d.List()
    elif cmd=="exit":
        exit()
    else:
```

```
    animator2d(cmd)
    plotPoint2d()
```

Animasi 2 Dimensi

Bagian program berikut berfungsi untuk membuat animasi 2 dimensi.

```
def animator2d (cmd):
    global matrix
    global step
    global N

    transform = cmd.split(" ")[0]
    param = cmd.split(" ",1)[1]

    if transform == "reflect":
        tf2d.Reflect2D(N,matrix,param)
    elif transform == "custom":
        a,b,c,d = param.split(" ")
        a = float(a)
        b = float(b)
        c = float(c)
        d = float(d)
        tf2d.Custom2D(N,matrix,a,b,c,d)
    elif transform == "multiple":
        n = int(param)
        listFungsi = []

        for i in range(n):
            Fungsi = input("- ")
            listFungsi.append(Fungsi)
        for f in listFungsi:
            animator2d(f)
            time.sleep(0.5)
    else :
        for i in range(step):
            if transform == "translate":
                dx,dy = param.split(" ")
                x = float(dx)/step
                y = float(dy)/step
                tf2d.Translate2D(N,matrix,x,y)
            elif transform == "dilate":
                k = pow(float(param),1/step)
                tf2d.Dilate2D(N,matrix,k)
            elif transform == "shear":
                sumbu = param.split(" ")[0]
                k = param.split(" ")[1]
                k = float(k)/step
                tf2d.Shear2D(N,matrix,sumbu,k)
            elif transform == "rotate":
                deg,x,y = param.split(" ")
                degr = float(deg)/step
                x = float(x)
                y = float(y)
                tf2d.Rotate2D(N,matrix,degr,x,y)
            elif transform == "stretch":
                axis,k = param.split(" ")
                k = pow(abs(float(k)),1/step)*(float(k)/abs(float(k)))
                tf2d.Stretch2D(N,matrix,axis,k)
```

```
time.sleep(0.01)
plotPoint2d()
```

Animasi Sumbu 2 Dimensi

Bagian berikut berfungsi untuk membuat animasi sumbu 2 dimensi.

```
def sumbu():
    glBegin(GL_LINES)
    glColor3f(1.0,0.0,0.0)
    glVertex2f(-500.0,0.0)
    glVertex2f(500.0,0.0)
    glVertex2f(0.0,500.0)
    glVertex2f(0.0,-500.0)
    glEnd()
```

Draw

```
def draw2d():
    glBegin(GL_POLYGON)
    glColor3f(1.0,0.0,0.0)
    for l in range(N):
        glVertex2f(matrix[0][l], matrix[1][l])
    glEnd()
```

Plot Point

```
def plotPoint2d():
    global N
    global matrix

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glLoadIdentity()
    gluOrtho2D(-500,500,-500,500)
    sumbu()
    draw2d()
    glutSwapBuffers()
```

Main Program untuk 2 Dimensi

Bagian berikut berfungsi sebagai main program 2 dimensi.

```
def main2d():
    global N
    global matrix
    global oriMatrix

    glutInit()
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_ALPHA | GLUT_DEPTH)
    glutInitWindowSize(500,500)
    glutInitWindowPosition(50,50)
    window = glutCreateWindow("2D") # create window with title
    input2D()
    glutDisplayFunc(plotPoint2d) # set draw function callback
    glutIdleFunc(transformasi) # draw all the time
    glutMainLoop()
```


3.2 3d.py

Bagian berikut berfungsi untuk menggambar Sumbu untuk 3 Dimensi

```
def draw_axis():
    glBegin(GL_LINES)
    glColor3f(1.0, 1.0, 1.0)    # White
    glVertex3f(-size, 0.0, 0.0)  # Sb X
    glVertex3f(size, 0.0, 0.0)   # Sb X
    glColor3f(1.0, 1.0, 1.0)    # White
    glVertex3f(0.0, -size, 0.0)  # Sb Y
    glVertex3f(0.0, size, 0.0)   # Sb Y
    glColor3f(1.0, 1.0, 1.0)    # White
    glVertex3f(0.0, 0.0, -size)  # Sb Z
    glVertex3f(0.0, 0.0, size)   # Sb Z
    glEnd()
```

Bagian berikut berfungsi untuk membentuk Kubus

```
def draw_cube():
    global matrix

    glBegin(GL_QUADS)
    # start drawing a rectangle
    # Top face (y = 0.5)
    # Define vertices in counter-clockwise (CCW) order with normal pointing out
    glColor3f(0.0, 1.0, 0.0)    # Green
    glVertex3f( matrix[0][0], matrix[0][1], matrix[0][2])
    glVertex3f( matrix[1][0], matrix[1][1], matrix[1][2])
    glVertex3f( matrix[2][0], matrix[2][1], matrix[2][2])
    glVertex3f( matrix[3][0], matrix[3][1], matrix[3][2])
    # Bottom face (y = -0.5)
    glColor3f(1.0, 0.5, 0.0)    # Orange
    glVertex3f( matrix[4][0], matrix[4][1], matrix[4][2])
    glVertex3f( matrix[5][0], matrix[5][1], matrix[5][2])
    glVertex3f( matrix[6][0], matrix[6][1], matrix[6][2])
    glVertex3f( matrix[7][0], matrix[7][1], matrix[7][2])
    # Front face (z = 0.5)
    glColor3f(1.0, 0.0, 0.0)    # Red
    glVertex3f( matrix[3][0], matrix[3][1], matrix[3][2])
    glVertex3f( matrix[2][0], matrix[2][1], matrix[2][2])
    glVertex3f( matrix[5][0], matrix[5][1], matrix[5][2])
    glVertex3f( matrix[4][0], matrix[4][1], matrix[4][2])
    # Back face (z = -0.5)
    glColor3f(1.0, 1.0, 0.0)    # Yellow
    glVertex3f( matrix[7][0], matrix[7][1], matrix[7][2])
    glVertex3f( matrix[6][0], matrix[6][1], matrix[6][2])
    glVertex3f( matrix[1][0], matrix[1][1], matrix[1][2])
    glVertex3f( matrix[0][0], matrix[0][1], matrix[0][2])
    # Left face (x = -0.5)
    glColor3f(0.0, 0.0, 1.0)    # Blue
    glVertex3f( matrix[2][0], matrix[2][1], matrix[2][2])
    glVertex3f( matrix[1][0], matrix[1][1], matrix[1][2])
    glVertex3f( matrix[6][0], matrix[6][1], matrix[6][2])
    glVertex3f( matrix[5][0], matrix[5][1], matrix[5][2])
    # Right face (x = 0.5)
    glColor3f(1.0, 0.0, 1.0)    # Magenta
    glVertex3f( matrix[0][0], matrix[0][1], matrix[0][2])
    glVertex3f( matrix[3][0], matrix[3][1], matrix[3][2])
```

```
glVertex3f( matrix[4][0], matrix[4][1], matrix[4][2])
glVertex3f( matrix[7][0], matrix[7][1], matrix[7][2])
glEnd()
```

Transformasi 3 Dimensi

Merupakan bagian yang menerima command dan juga berisi fungsi untuk menggeser-geser point of view.

```
def transformasi():
    global matrix
    global viewX
    global viewY
    global viewZ
    cmd = input(">>> ")
    if cmd=="reset":
        for i in range(8):
            for j in range(3):
                if j == 0:
                    if i == 1 or i == 2 or i == 5 or i == 6:
                        matrix[i][j] = -50.0
                    else:
                        matrix[i][j] = 50.0
                elif j == 1:
                    if i >= 4:
                        matrix[i][j] = -50.0
                    else:
                        matrix[i][j] = 50.0
                else:
                    if i == 0 or i == 1 or i == 6 or i == 7:
                        matrix[i][j] = -50.0
                    else:
                        matrix[i][j] = 50.0
    elif cmd=="exit":
        exit()
    elif cmd == "w":
        viewZ += 50
    elif cmd == "s":
        viewZ += -50
    elif cmd == "a":
        viewX += 50
    elif cmd == "d":
        viewX += -50
    else:
        animator3d(cmd)
    plotPoint3d()
```

Animasi 3 Dimensi

Bagian berikut berfungsi sebagai animasi 3 dimensi.

```
def animator3d (cmd):
    global matrix
    global step

    transform = cmd.split(" ")[0]
    param = cmd.split(" ",1)[1]
```

```

if transform == "reflect":
    tf3d.Reflection3D(matrix,param)
elif transform == "custom":
    a,b,c,d,e,f,g,h,i = param.split(" ")
    a = float(a)
    b = float(b)
    c = float(c)
    d = float(d)
    e = float(e)
    f = float(f)
    g = float(g)
    h = float(h)
    i = float(i)
    tf3d.Custom3D(matrix,a,b,c,d,e,f,g,h,i)
elif transform == "multiple":
    n = int(param)
    listFungsi = []

    for i in range(n):
        Fungsi = input("- ")
        listFungsi.append(Fungsi)
    for F in listFungsi:
        animator3d(F)
        time.sleep(0.5)
else :
    for i in range(step):
        if transform == "translate":
            dx,dy,dz = param.split(" ")
            x = float(dx)/step
            y = float(dy)/step
            z = float(dz)/step
            tf3d.Translate3D(matrix,x,y,z)
        elif transform == "dilate":
            k = pow(float(param),1/step)
            tf3d.Dilate3D(matrix,k)
        elif transform == "shear":
            sumbu,k = param.split(" ")
            k = float(k)/step
            tf3d.Shear3D(matrix,sumbu,k)
        elif transform == "rotate":
            axis,deg = param.split(" ")
            deg = float(deg)/step
            tf3d.Rotation3D(matrix,axis,deg)
        elif transform == "stretch":
            axis,k = param.split(" ")
            k = pow(abs(float(k)),1/step)*(float(k)/abs(float(k)))
            tf3d.Stretch3D(matrix,axis,k)
        time.sleep(0.01)
    plotPoint3d()

```

Plot Point 3 Dimensi

```

def plotPoint3d():
    time
    global matrix
    global viewX
    global viewY
    global viewZ
    # ondraw is called all the

```

```

glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT) # clear the screen
glLoadIdentity() # reset position
glOrtho(-1000,1000,-1000,1000,-2000,2000)
gluLookAt(viewX, viewY, viewZ, -50.0, -75.0, -100.0, 0.0,1.0,0.0)
draw_cube()
draw_axis()
glutSwapBuffers() # important for double buffering

```

Main Program untuk 3 Dimensi

```

def main3d ():
    global viewX
    global viewY
    global viewZ

    glutInit() # initialize glut
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_ALPHA | GLUT_DEPTH)
    glutInitWindowSize(width, height) # set window size
    glutInitWindowPosition(0, 0) # set window position
    window = glutCreateWindow("3D") # create window with title
    viewX = 0;
    viewY = 0;
    viewZ = 0;
    glEnable(GL_DEPTH_TEST) # remove unseen faces
    MatrixInit()
    glutDisplayFunc(plotPoint3d) # set draw function callback
    glutIdleFunc(transformasi) # draw all the time
    glutMainLoop()

```

3.3 transformasi2d.py

Translasi 2 Dimensi

Berfungsi untuk menggeser objek sejauh dx dan dy sesuai dengan input user.

```

def Translate2D(N, matrix, dx, dy):
    for i in range(N):
        matrix[0][i] = matrix[0][i] + dx
        matrix[1][i] = matrix[1][i] + dy

```

Dilatasi 2 Dimensi

Berfungsi untuk memperbesar objek sebesar k sesuai dengan input user.

```

def Dilate2D(N, matrix, k):
    for i in range(N):
        matrix[0][i] = matrix[0][i] * k
        matrix[1][i] = matrix[1][i] * k

```

Rotasi 2 Dimensi

Berfungsi untuk merotasi sebuah objek sebesar deg derajat terhadap titik x,y.

```
def Rotate2D(N, matrix, deg, x, y):
    rad = float(deg*0.0174533)
    for i in range(N):
        temp1 = matrix[0][i]
        temp2 = matrix[1][i]
        matrix[0][i] = ((math.cos(rad) * (temp1 - x)) -
                        (math.sin(rad) * (temp2 - y))) + x
        matrix[1][i] = ((math.sin(rad) * (temp1 - x)) +
                        (math.cos(rad) * (temp2 - y))) + y
```

Refleksi 2 Dimensi

Berfungsi untuk menghasilkan pencerminan dari sebuah objek terhadap garis yang diminta.

```
def Reflect2D(N, matrix, param):
    if param=="x":
        for i in range(N) :
            temp = matrix[1][i]
            matrix[1][i] = -temp
    elif param=="y":
        for i in range(N) :
            temp = matrix[0][i]
            matrix[0][i] = -temp
    elif param=="x=y" or param=="y=x":
        for i in range(N) :
            temp = matrix[0][i]
            matrix[0][i] = matrix[1][i]
            matrix[1][i] = temp
    elif param=="y=-x" or param=="-y=x" or param=="x=-y" or param=="-x=y":
        for i in range(N) :
            temp = matrix[0][i]
            matrix[0][i] = -matrix[1][i]
            matrix[1][i] = -temp
    else:
        a,b = param[1:][::-1].split(",")
        a = float(a)
        b = float(b)
        for i in range(N) :
            temp0 = matrix[0][i]
            temp1 = matrix[1][i]
            matrix[0][i] = 2*a - temp0
            matrix[1][i] = 2*b - temp1
```

Shear 2 Dimensi

```
def Shear2D(N,matrix,param, k):
    if (param == "x"):
        for i in range(N):
            matrix[0][i] = matrix[0][i] + k * matrix[1][i]
    elif (param == "y"):
        for i in range(N):
            matrix[1][i] = matrix[0][i] * k + matrix[1][i]
```

Stretch 2 Dimensi

```
def Stretch2D(N,matrix, param, k):
```

```

if (param == "x"):
    for i in range(N):
        matrix[0][i] = matrix[0][i] * k
elif (param == "y"):
    for i in range(N):
        matrix[1][i] = matrix[1][i] * k

```

Reset 2 Dimensi

Berfungsi mengembalikan objek ke bentuk asalnya.

```

def Reset2D(N,matrix,oriMatrix):
    for i in range(N):
        matrix[0][i] = oriMatrix[0][i]
        matrix[1][i] = oriMatrix[1][i]

```

Custom 2 Dimensi

Berfungsi untuk mentransformasikan suatu objek sesuai dengan matriks yang diinput user.

```

def Custom2D(N, matrix, a, b, c, d):
    for i in range(N):
        x = matrix[0][i]
        y = matrix[1][i]
        matrix[0][i] = x*a + y*b
        matrix[1][i] = x*c + y*d

```

3.4 transformasi3d.py

Translasi 3 Dimensi

```

def Translate3D(matrix,dx,dy,dz):
    for i in range(8):
        matrix[i][0] += dx
        matrix[i][1] += dy
        matrix[i][2] += dz

```

Dilatasi 3 Dimensi

```

def Dilate3D(matrix,k):
    for i in range(8):
        for j in range(3):
            matrix[i][j] *= k

```

Shear 3 Dimensi

```

def Shear3D(matrix,param,k):
    if param == "x":
        for i in range(8):
            matrix[i][0] += (k*matrix[i][1] + k*matrix[i][2])
    elif param == "y":
        for i in range(8):
            matrix[i][1] += (k*matrix[i][0] + k*matrix[i][2])
    elif param == "z":

```

```

for i in range(8):
    matrix[i][2] += (k*matrix[i][0] + k*matrix[i][1])

```

Rotasi 3 Dimensi

```

def Rotation3D(matrix,axis,deg):
    rad = deg/180*3.14
    if axis == "x":
        for i in range(8):
            a=matrix[i][0]
            b=matrix[i][1]
            c=matrix[i][2]
            matrix[i][1]=math.cos(rad)*b-math.sin(rad)*c
            matrix[i][2]=math.sin(rad)*b+math.cos(rad)*c
    elif axis == "y":
        for i in range(8):
            a=matrix[i][0]
            b=matrix[i][1]
            c=matrix[i][2]
            matrix[i][0]=math.cos(rad)*a+math.sin(rad)*c
            matrix[i][2]=-math.sin(rad)*a+math.cos(rad)*c
    elif axis == "z":
        for i in range(8):
            a=matrix[i][0]
            b=matrix[i][1]
            c=matrix[i][2]
            matrix[i][0]=math.cos(rad)*a-math.sin(rad)*b
            matrix[i][1]=math.sin(rad)*a+math.cos(rad)*b

```

Refleksi 3 Dimensi

```

def Reflection3D(matrix,plane):
    if plane == "xy" or plane == "yx":
        for i in range(8):
            matrix[i][2]*=(-1)
    elif plane == "yz" or plane == "zy":
        for i in range(8):
            matrix[i][0]*=(-1)
    elif plane == "xz" or plane == "zx":
        for i in range(8):
            matrix[i][1]*=(-1)

```

Custom 3 Dimensi

```

def Custom3D(matrix,a,b,c,d,e,f,g,h,i):
    for i in range(8):
        x=matrix[i][0]
        y=matrix[i][1]
        z=matrix[i][2]
        matrix[i][0]= x*a + y*b + z*c
        matrix[i][1]= x*d + y*e + z*f
        matrix[i][2]= x*g + y*h + z*i

```

Stretch 3 Dimensi

```
def Stretch3D(matrix,axis,k):
    if axis=='x':
        for i in range(8):
            matrix[i][0]=matrix[i][0]*k
    elif axis=='y':
        for i in range(8):
            matrix[i][1]=matrix[i][1]*k
    elif axis=='z':
        for i in range(8):
            matrix[i][2]=matrix[i][2]*k
```

Pembagian tugas:

1. Muhammad Raihan Asyraf Desanto/13517027

- fungsi transformasi 3 dimensi selain yang dibuat Kintan
- animasi 2 dimensi dan 3 dimensi

2. Jeremy Arden Hartono/13517101

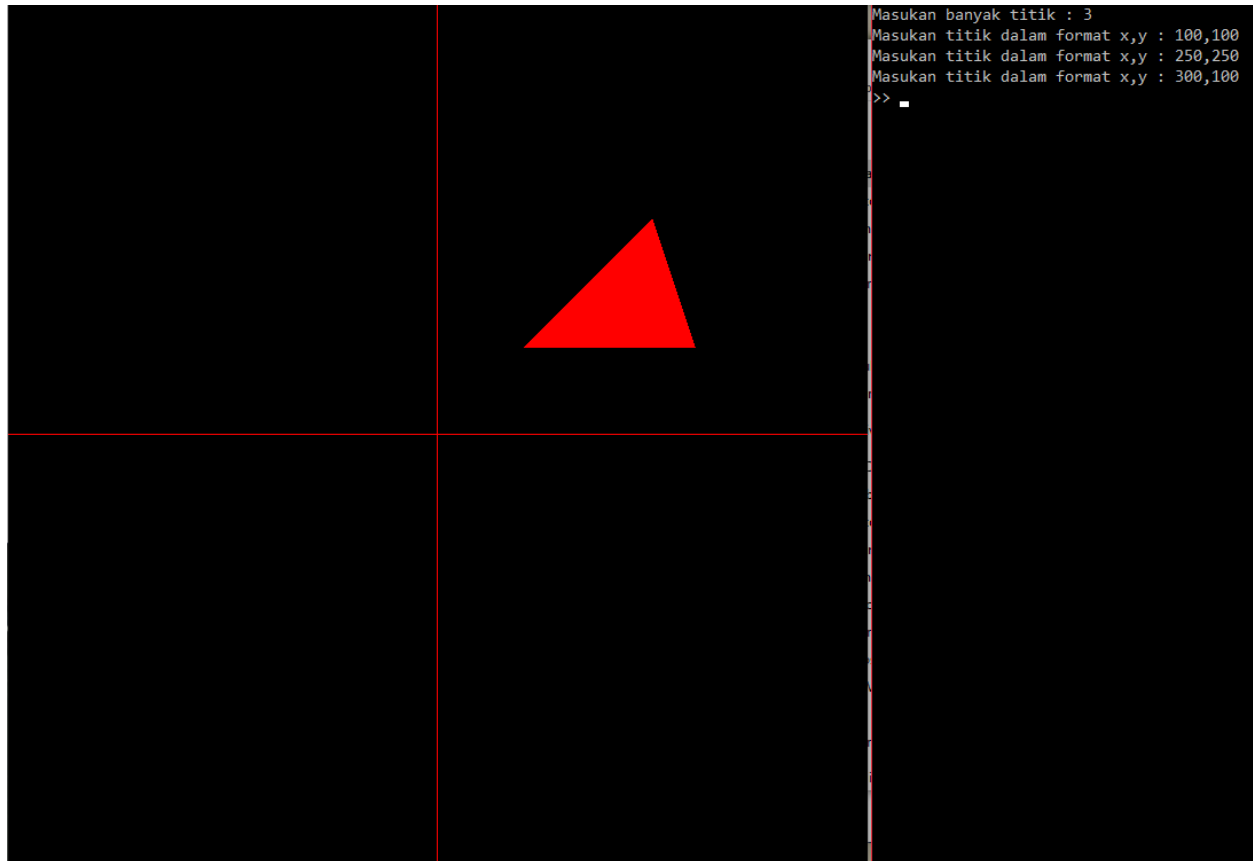
- seluruh fungsi transformasi 2 dimensi

3. Kintan Sekar Adinda/13517102

- fungsi transformasi rotasi, refleksi, custom, dan stretch 3 dimensi
- laporan bab 1,2,3

BAB 4 EKSPERIMEN

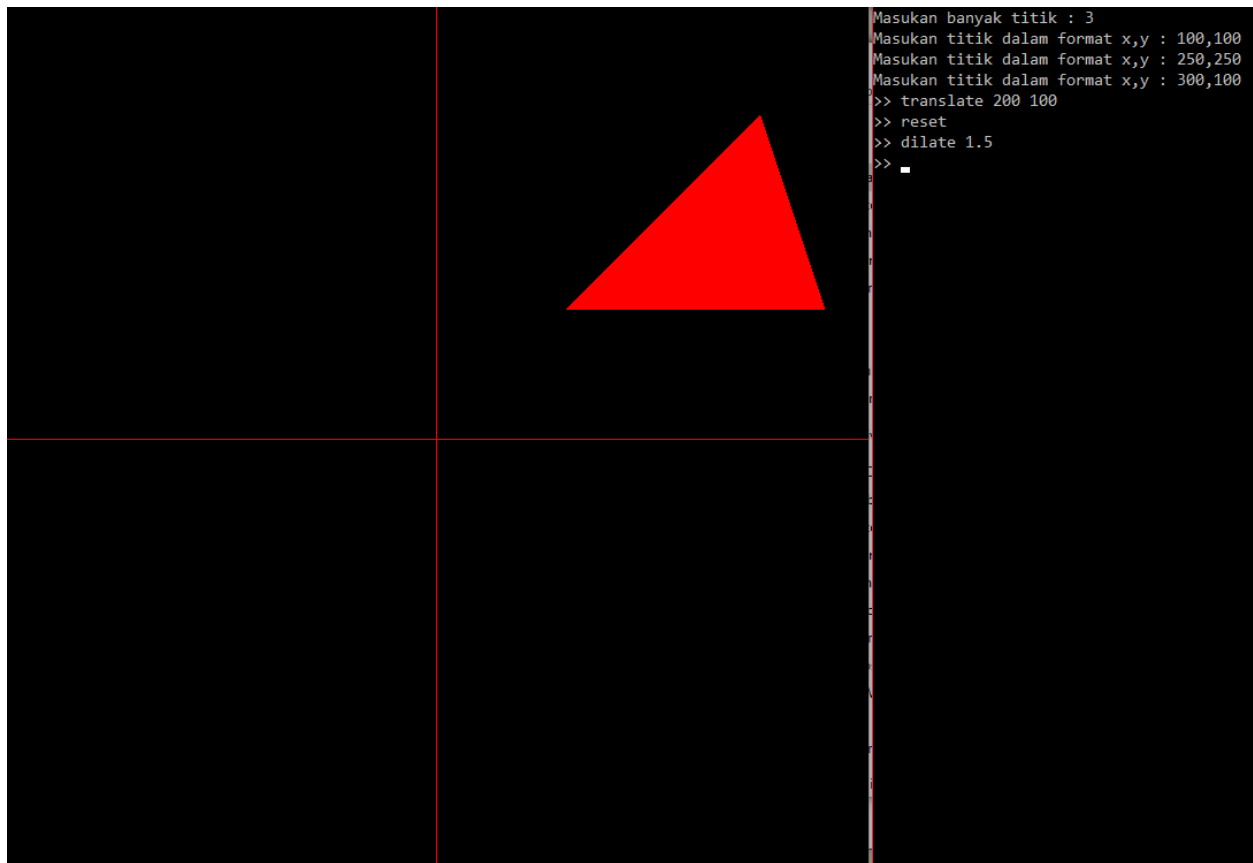
Membuat Bangun 2 Dimensi



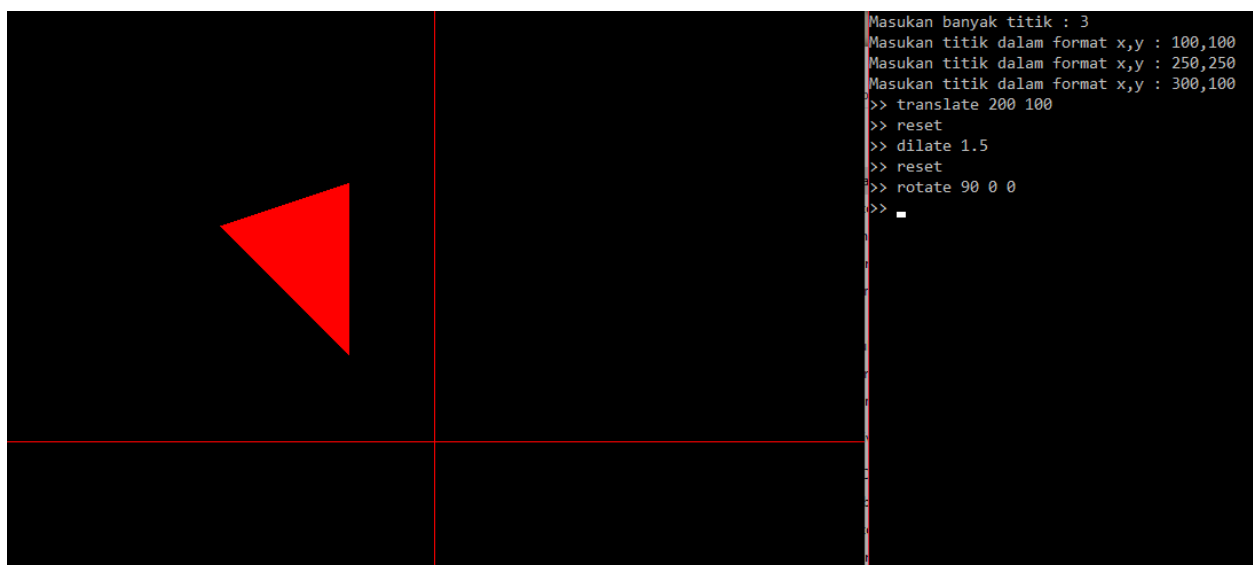
Translasi 2 Dimensi



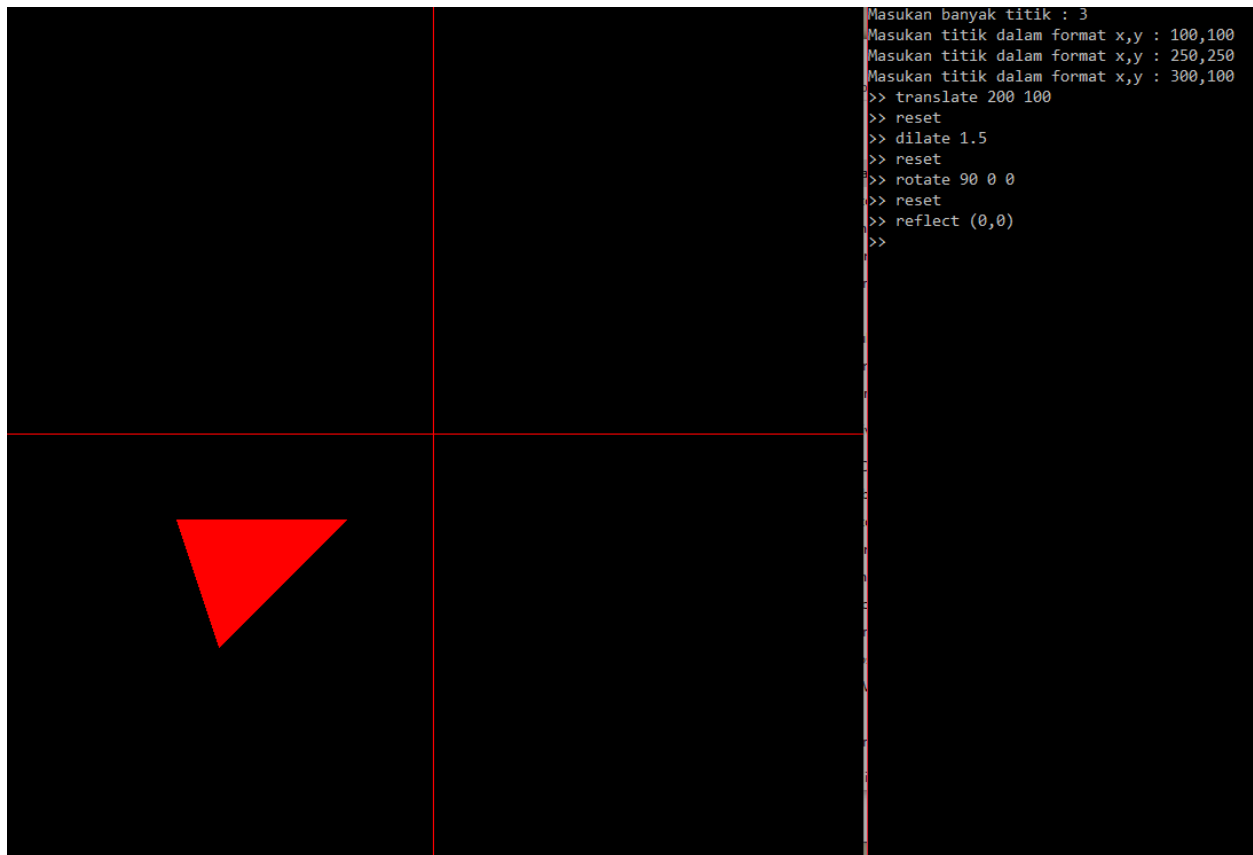
Dilatasi 2 Dimensi



Rotasi 2 Dimensi



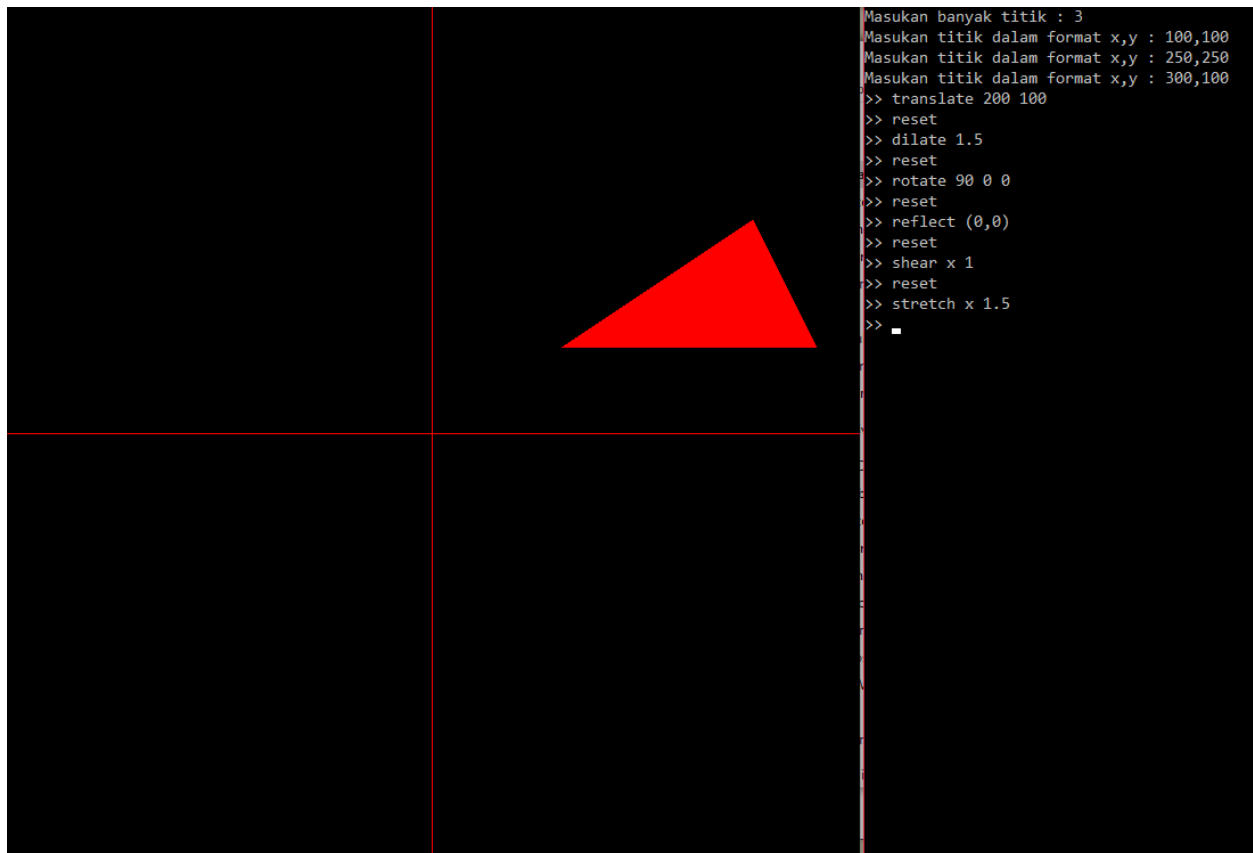
Refleksi 2 Dimensi



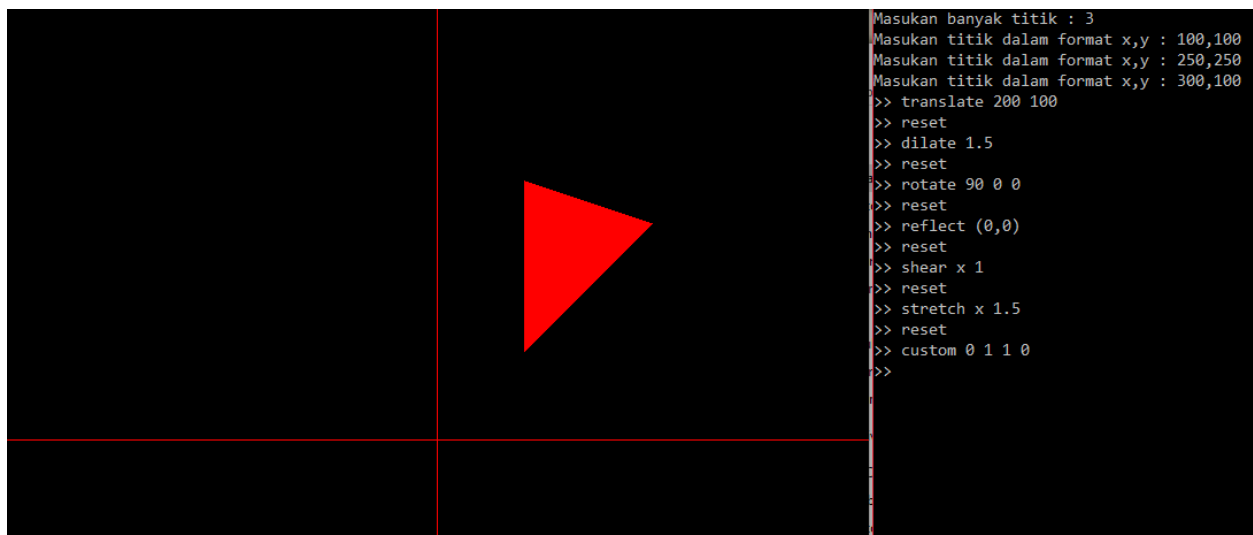
Shear 2 Dimensi



Stretch 2 Dimensi



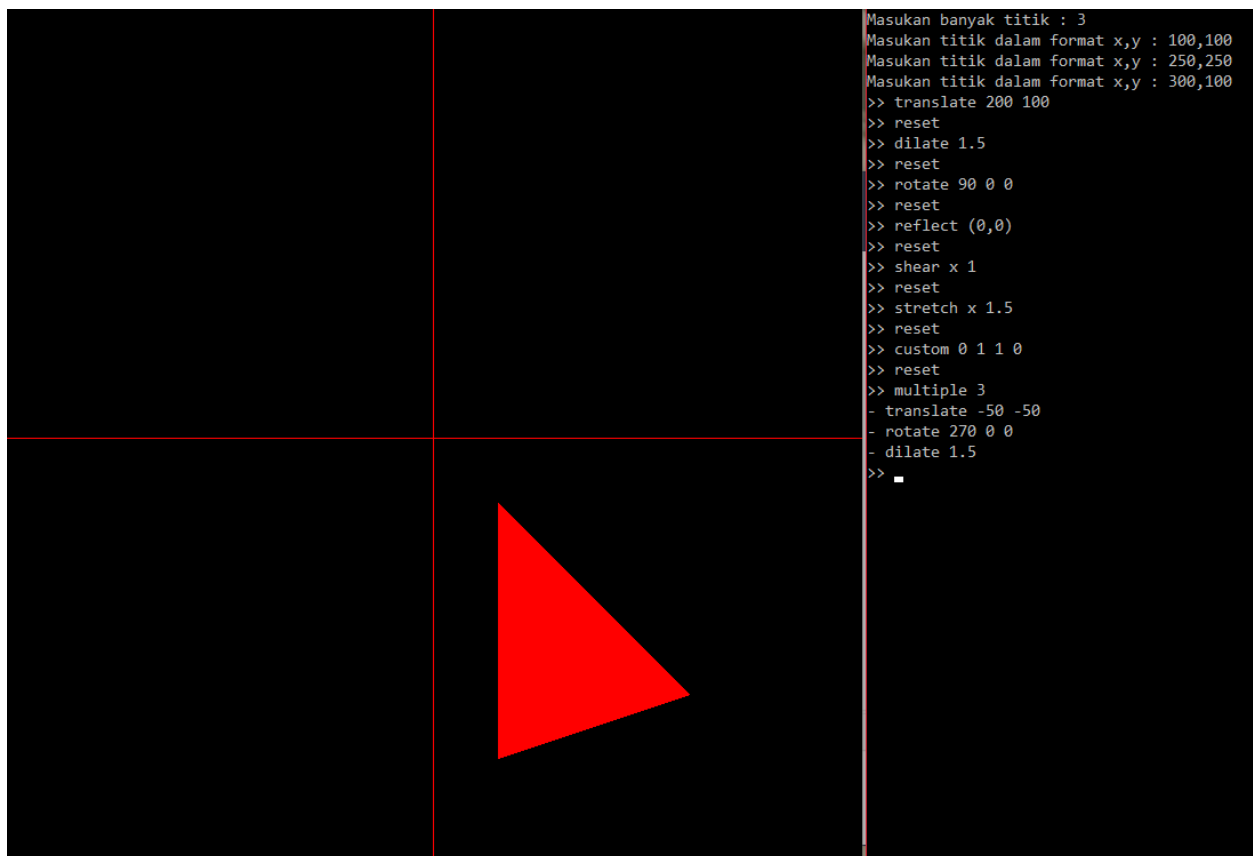
Custom 2 Dimensi



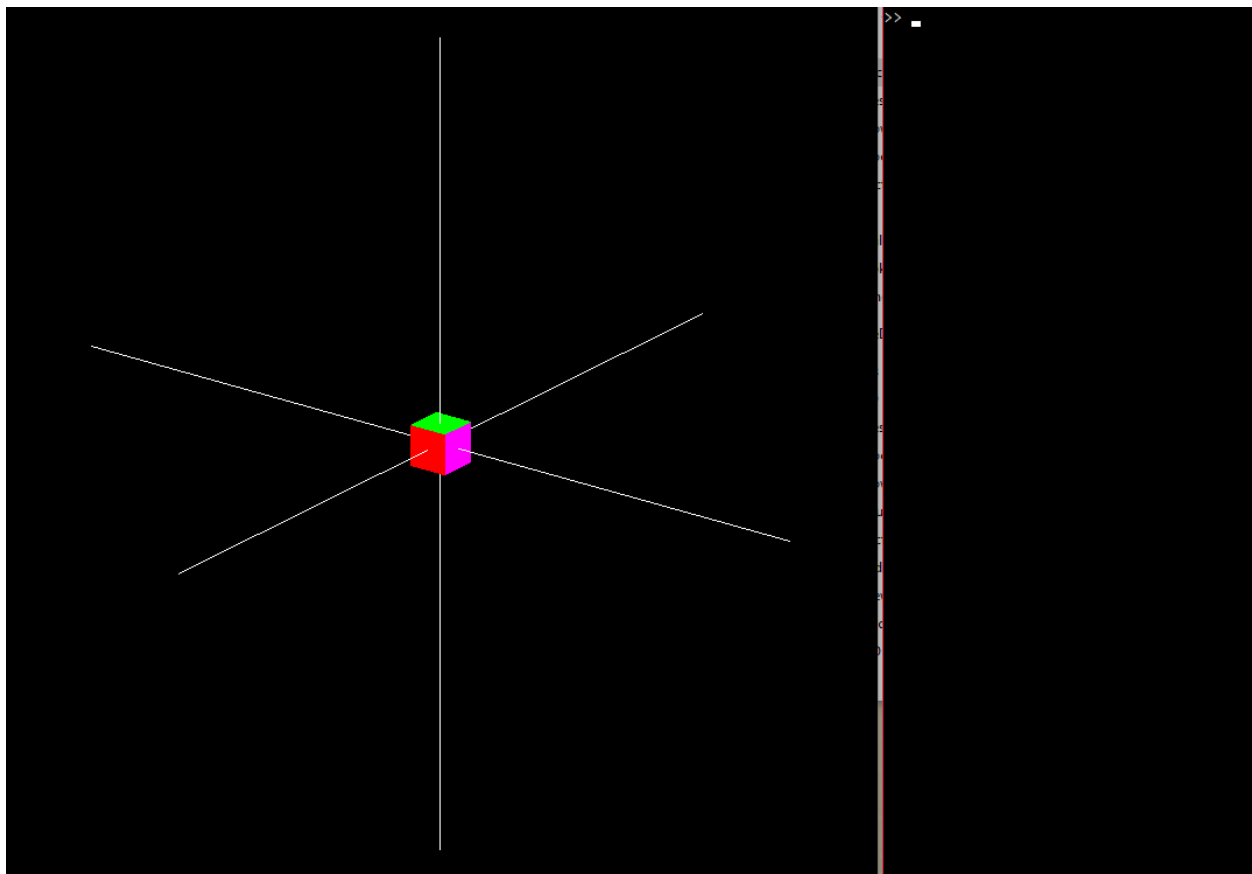
Reset 2 Dimensi



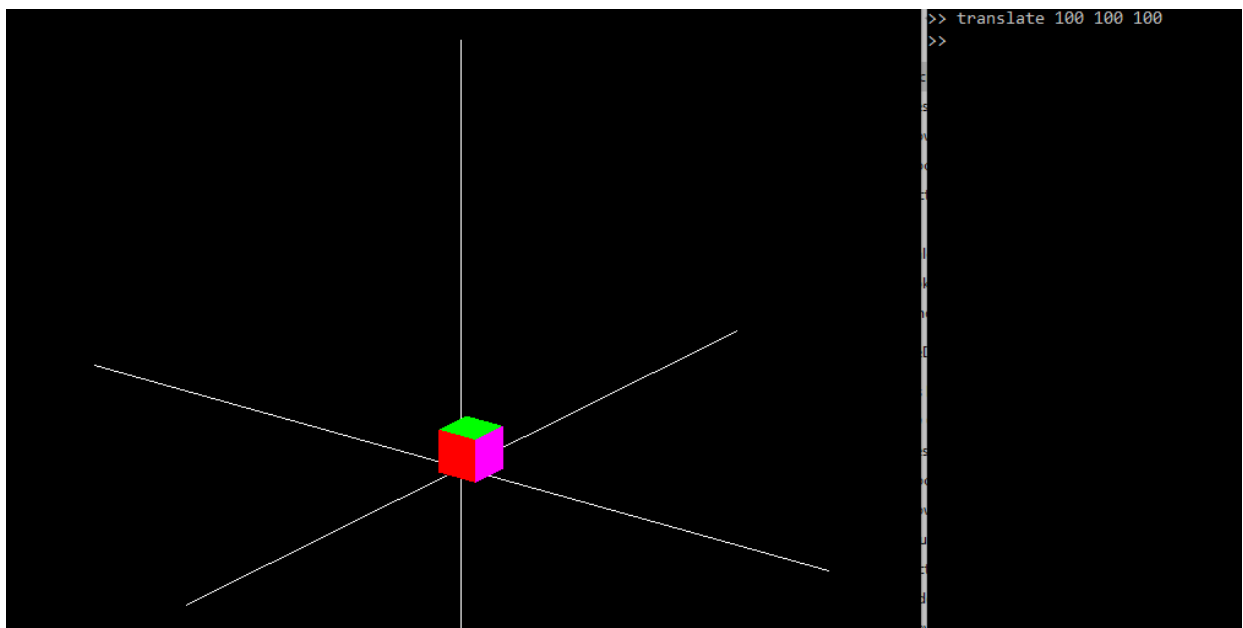
Multiple 2 Dimensi



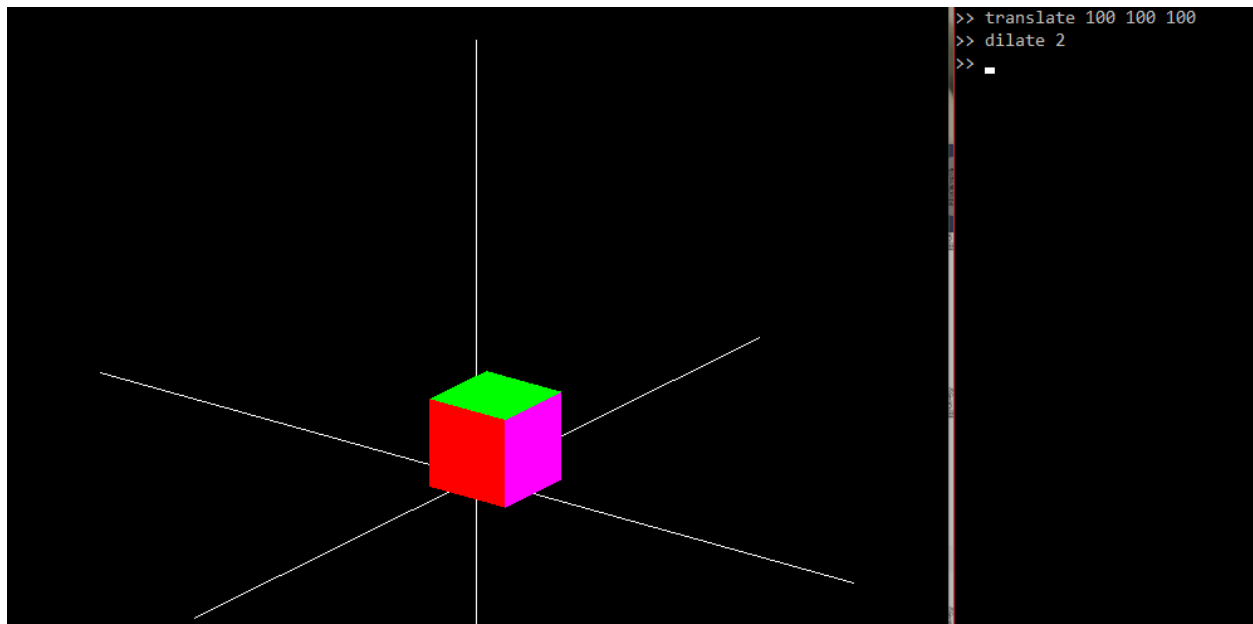
Animasi Kubus Sebelum Ditransformasi



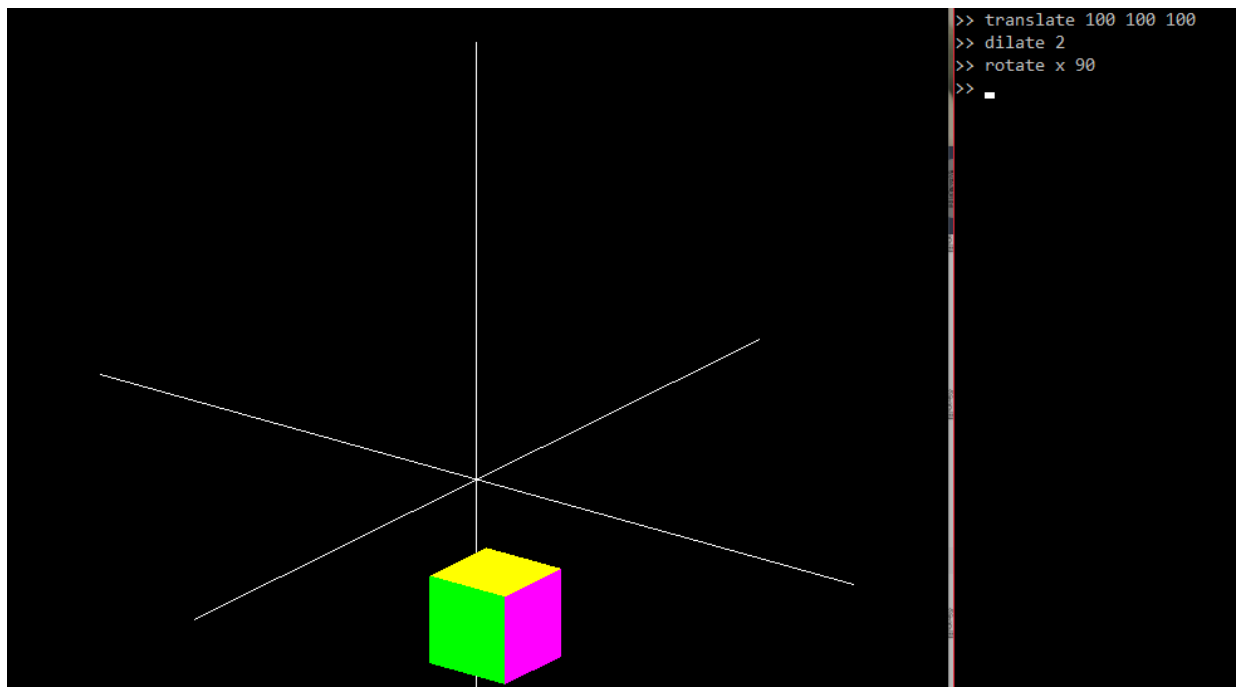
Translasi 3 Dimensi



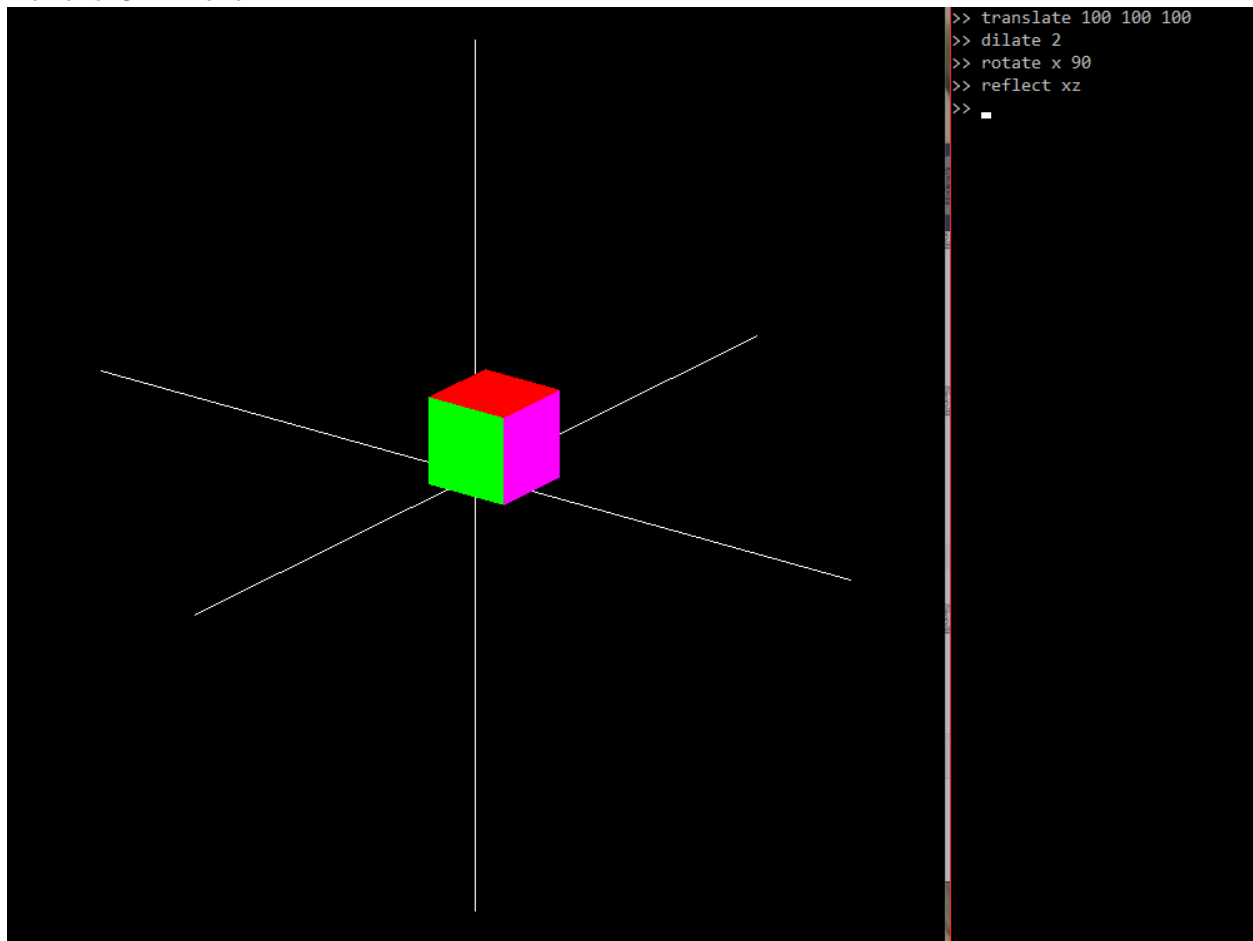
Dilatasi 3 Dimensi



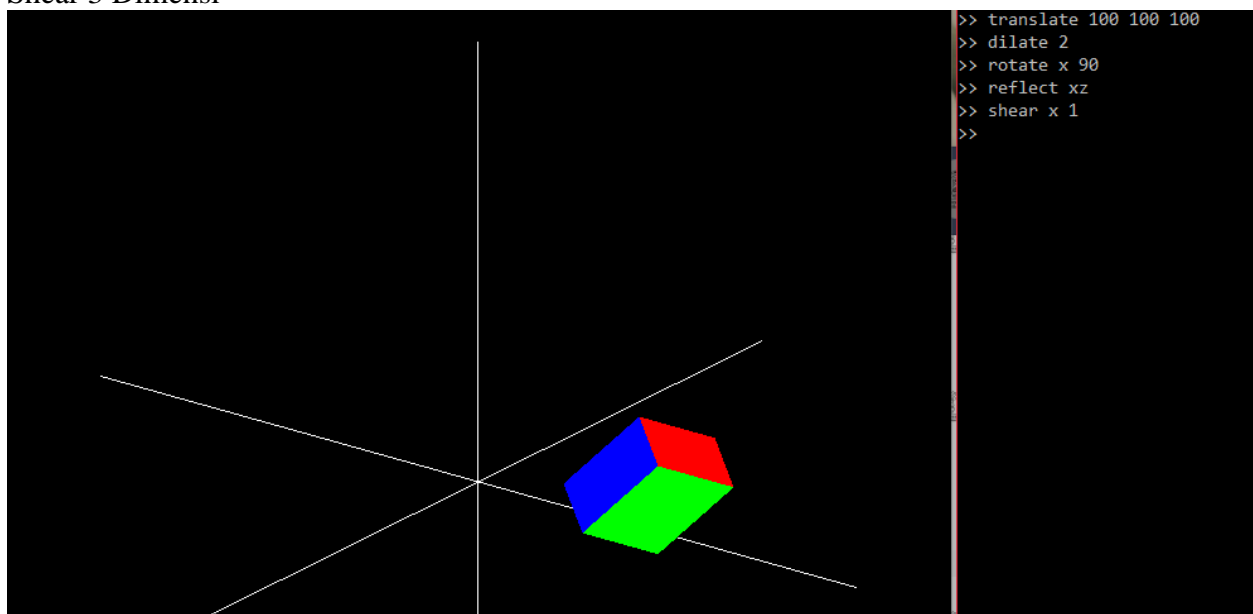
Rotasi 3 Dimensi



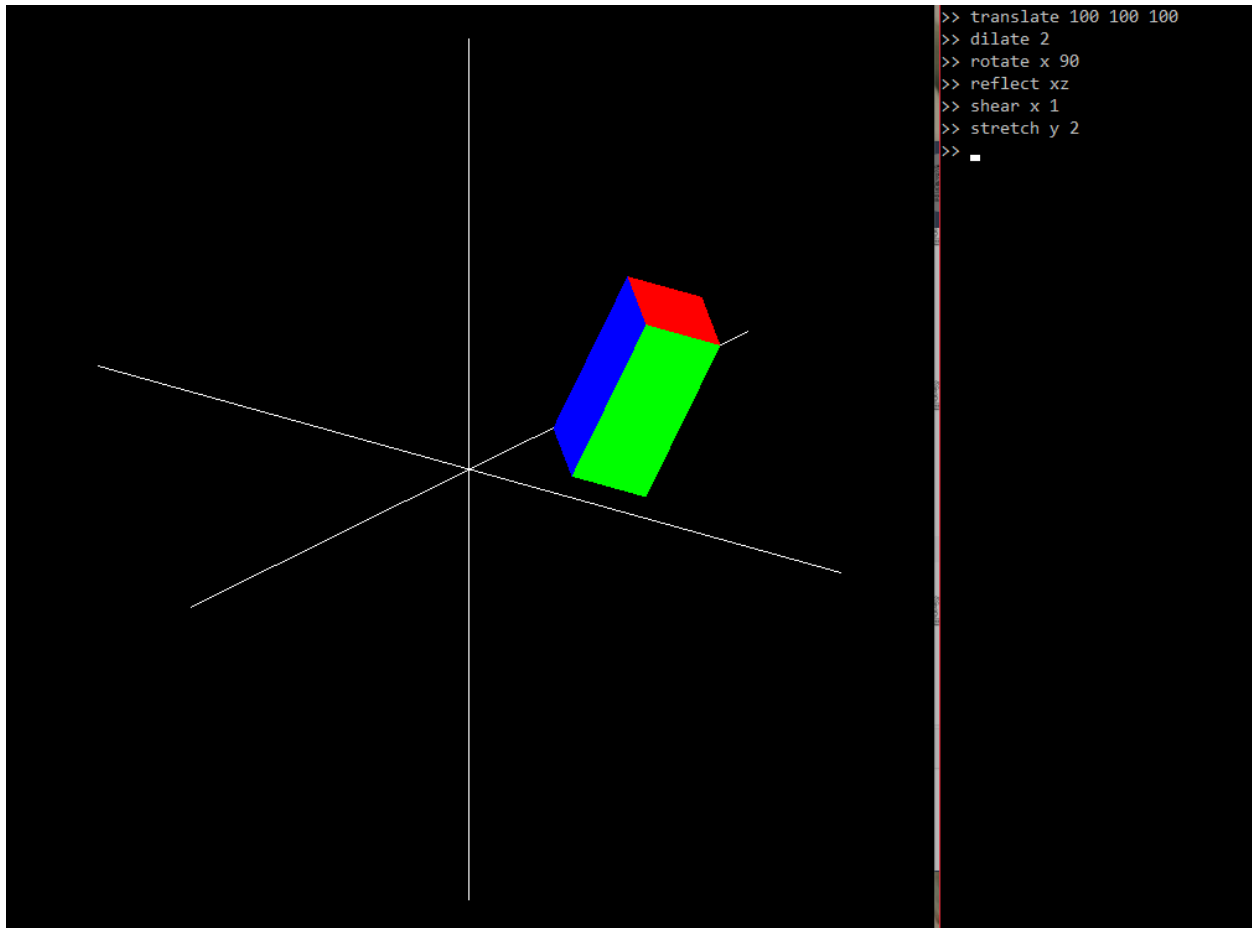
Refleksi 3 Dimensi



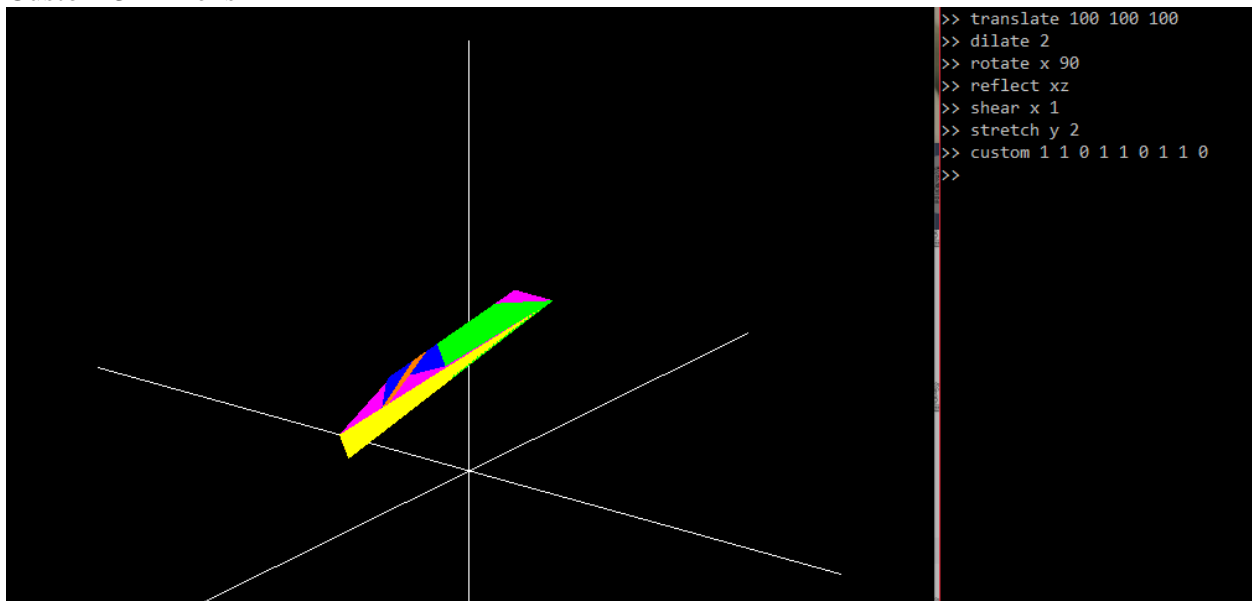
Shear 3 Dimensi



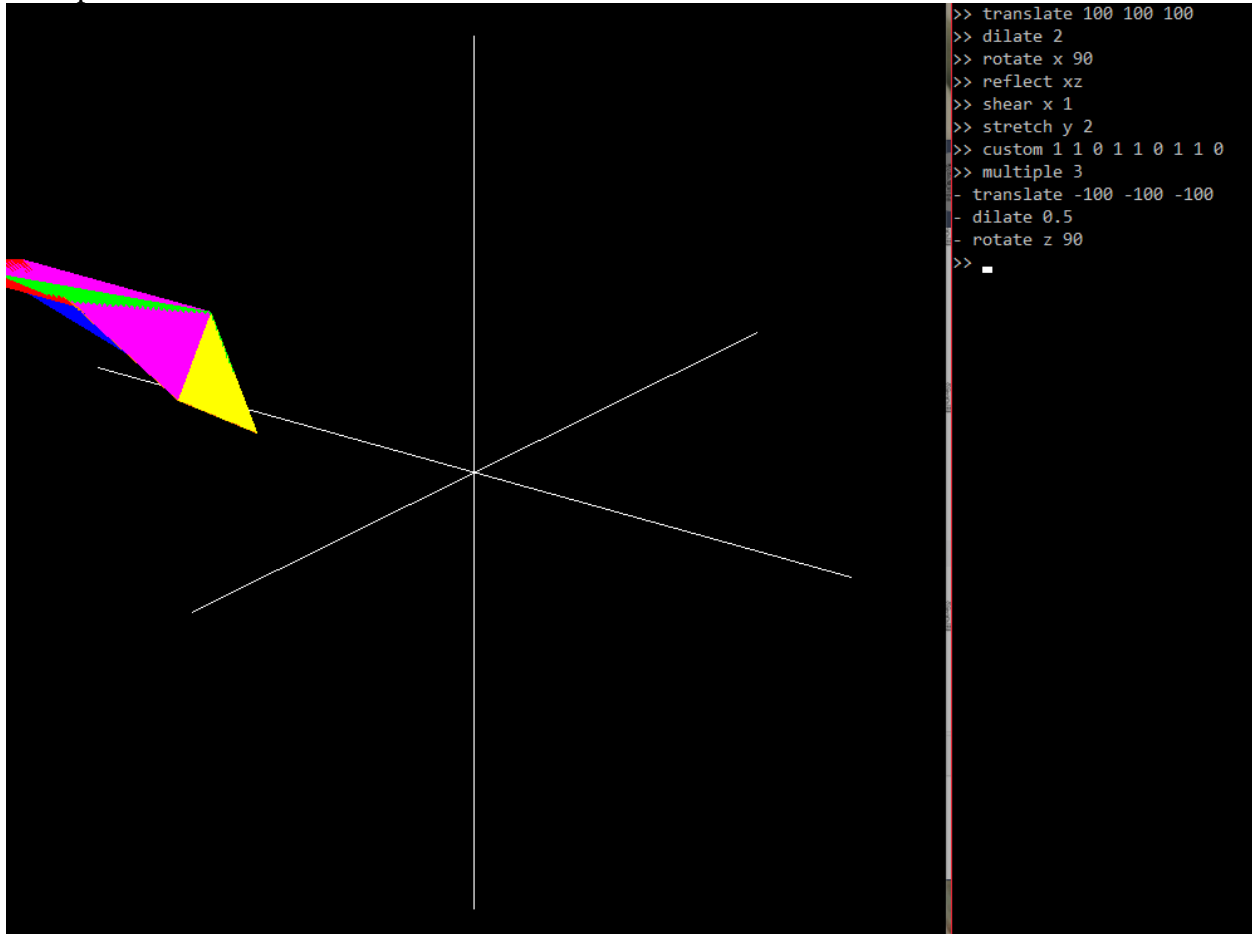
Stretch 3 Dimensi



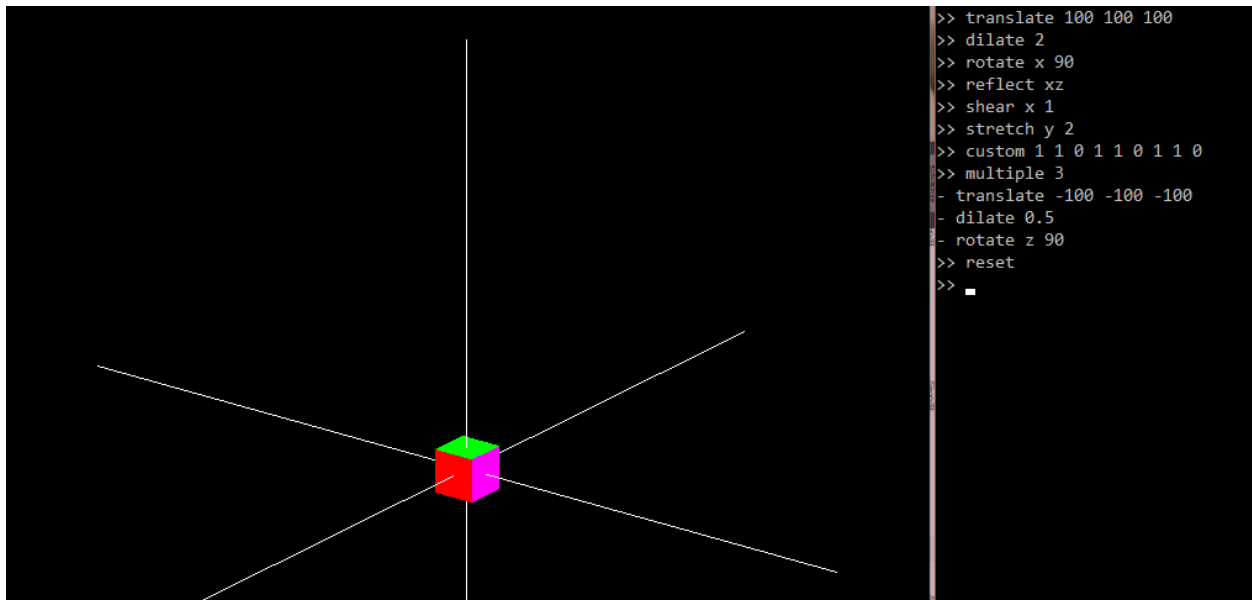
Custom 3 Dimensi



Multiple 3 Dimensi



Reset 3 Dimensi



BAB 5

KESIMPULAN, SARAN, DAN REFLEKSI

5.1 Kesimpulan

Program yang kami buat dapat menerima masukan berupa objek 2D maupun 3D melalui *keyboard* pada *window command prompt*, sedangkan *output* berdasarkan masukan dari *user* akan ditampilkan ke layer pada *window GUI*. Program ini dibuat dengan menggunakan *Bahasa Python* serta menggunakan *OpenGL API*. Program ini dapat mensimulasikan transformasi linier pada objek 2D maupun 3D dimensi berdasarkan perintah dari *user* sekaligus menampilkan animasi gerakannya.

5.2 Saran

Disarankan bagi pembuat program untuk lebih mendalami dan memahami kegunaan *library* pada *OpenGL API*. Dengan melakukan hal tersebut diharapkan dapat mengurangi halangan yang bersifat teknis dalam pembuatan program. Selain itu, hal tersebut dapat membantu pembuatan program yang efektif dan efisien dengan implementasi *OpenGL API* pada bahasa pemrograman *Python*. Dalam pengerjaan tugas, perlu dibuatnya *timeline* pengerjaan agar mendapatkan hasil akhir yang maksimal.

5.3 Refleksi

Tugas besar ini sangat menggugah wawasan pembuat program dalam pembuatan program grafika dan animasi pada komputer, serta dapat menambah wawasan bagi kami sebagai pembuat program tentang *OpenGL API* yang diimplementasikan dalam bahasa *Python*. Kami menyadari bahwa program yang kami buat belum rapih sepenuhnya. Oleh karena itu, kedepannya kami perlu berkoordinasi lebih baik lagi dalam hal pembagian tugas. Tugas besar ini menyadarkan pula betapa berharganya waktu, semakin dekat dengan *deadline* semakin berharga waktu yang kami miliki untuk menyelesaikan tugas ini sebaik mungkin.

DAFTAR PUSTAKA

Strang, Gilbert. (2009). Introduction to Linear Algebra, Fourth Edition.

Anton, Howard & Chris Rorres. (2005). Aljabar Linier Elementer edisi 8.