

# **Simulasi Transformasi Linier pada Bidang 2D dan 3D dengan Menggunakan OpenGL API**

## **LAPORAN TUGAS BESAR II ALJABAR GEOMETRI IF2123**

**Diajukan untuk memenuhi tugas mata kuliah Aljabar Geometri  
pada Semester I tahun Akademik 2018 – 2019**

**oleh**

**T. Andra Oksidian Tafly / 13517020**

**Andrian Cedric / 13517065**

**Jan Meyer Saragih / 13517131**



**Program Studi Teknik Informatika  
Sekolah Teknik Elektro dan Informatika  
Institut Teknologi Bandung**

# 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, poligon 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 di bawah.

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 $param$ 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.
shear <param> <k>	Melakukan operasi <i>shear</i> pada objek. Nilai $param$ 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 $param$ 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:

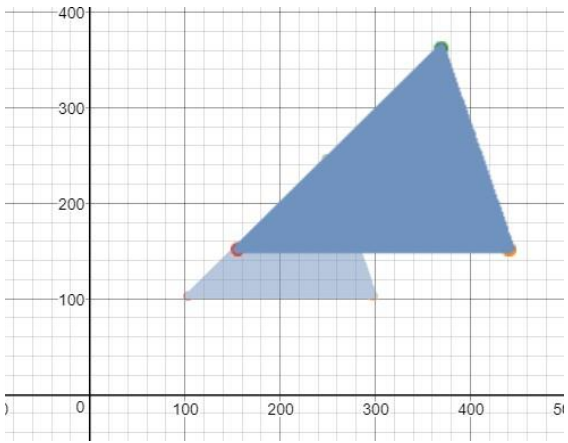
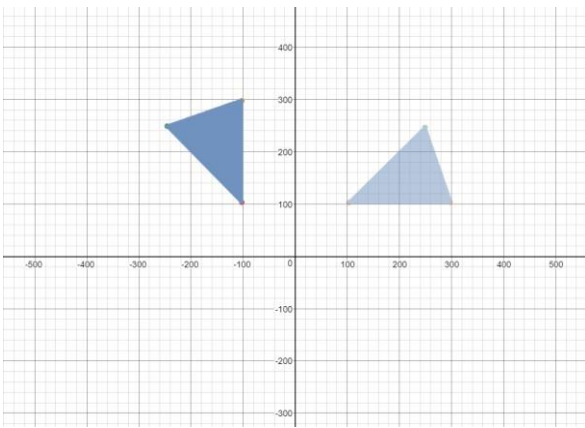
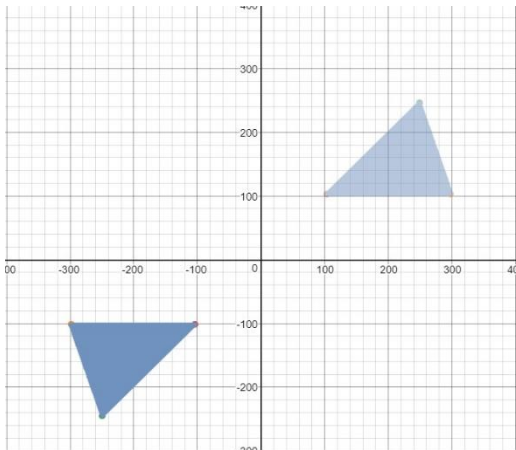
Input	Output
3 100,100 250,250 300,100	

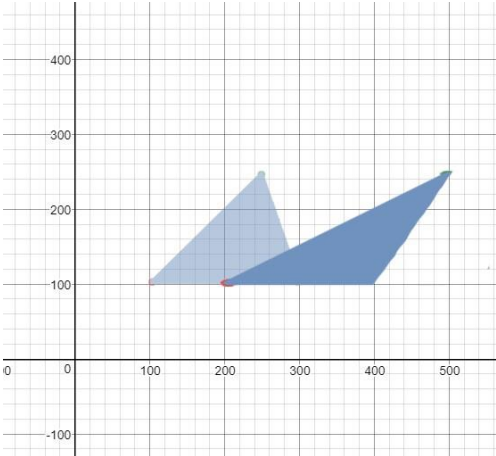
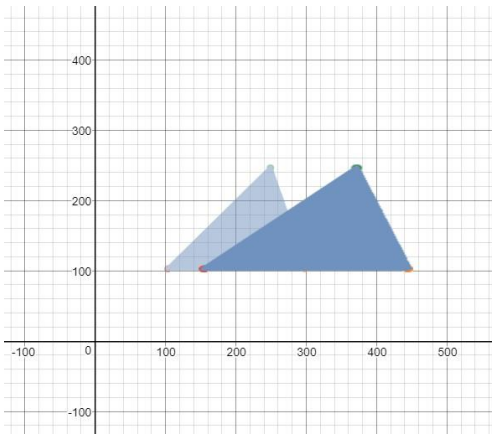
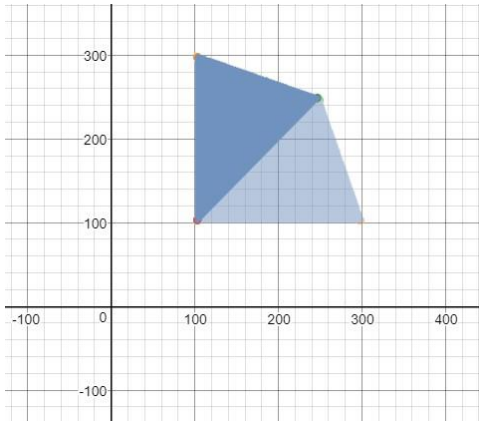
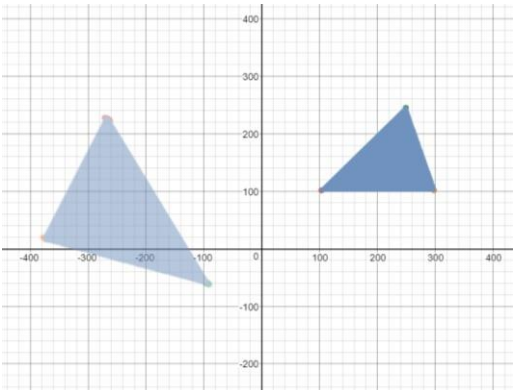
Perhatikan bahwa garis-garis tipis pada gambar diatas *tidak perlu diimplementasikan* pada program.

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	

<p>dilate 1.5</p>	
<p>rotate 90 0 0</p>	
<p>reflect (0,0)</p>	

<p>shear x 1</p>	
<p>stretch x 1.5</p>	
<p>custom 0 1 1 0</p>	
<p>reset</p>	

## BAB 2

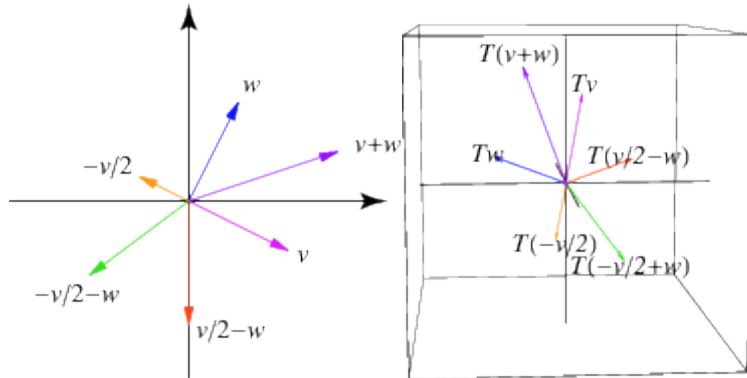
### LANDASAN TEORI

#### 2.1. Transformasi Linier

Misalkan, ada dua buah vektor ruang,  $V$  dan  $W$ , dengan pemetaan  $T: V \rightarrow W$ , maka  $T$  dapat disebut sebagai transformasi linier dengan memenuhi dua syarat berikut :

1.  $T(\mathbf{v}_1 + \mathbf{v}_2) = T(\mathbf{v}_1) + T(\mathbf{v}_2)$  untuk setiap vektor  $\mathbf{v}_1$  dan  $\mathbf{v}_2$  di dalam  $V$  (sifat aditif);
2.  $T(\alpha \mathbf{v}) = \alpha T(\mathbf{v})$  untuk setiap skalar  $\alpha$  (sifat kehomogenan).

Secara umum, transformasi linier bisa dikatakan injektif atau surjektif, tetapi juga tidak. Ketika  $V$  dan  $W$  memiliki dimensi yang sama, ada kemungkinan bahwa  $T$  dapat diinverskan, sesuai dengan hukum  $TT^{-1} = I$ . Selain itu, dalam transformasi linier juga berlaku  $T(0) = 0$ .



Gambar 2.1.1, contoh visualisasi dari transformasi linier.

#### 2.2 Matriks Transformasi

Matriks transformasi adalah matriks yang dapat digunakan untuk mengubah matriks sesuai dengan yang diinginkan pengguna, misal merotasi maupun mendilasi sesuatu objek.

Cara mengubah penulisan transformasi dengan cara biasa ke dalam bentuk matriks :

Misalkan  $T$  adalah transformasi yang memetakan titik  $(x, y)$  ke titik  $(2x-y, x+3y)$

Maka dengan cara biasa penulisannya  $x, y \Rightarrow T(2x-y, x+3y)$

BEBERAPA MATRIKS TRANSFORMASI		
Jenis transformasi	Penulisan dengan menggunakan matriks	Matriks transformasi
$M_x$	$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
$M_y$	$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$	$\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$
$M_{x=y}$	$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
$M_{x=-y}$	$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$	$\begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}$
$M_{x=a}$	$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 2a \\ 0 \end{pmatrix}$	$\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$
$M_{y=b}$	$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 2b \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
$M_{y=mx+c}$ $\tan \alpha = m$	$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos 2\alpha & \sin 2\alpha \\ \sin 2\alpha & -\cos 2\alpha \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \frac{c}{m} \begin{pmatrix} \cos 2\alpha - 1 \\ \sin 2\alpha \end{pmatrix}$	$\begin{pmatrix} \cos 2\alpha & \sin 2\alpha \\ \sin 2\alpha & -\cos 2\alpha \end{pmatrix}$
$R_\alpha$	$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$	$\begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix}$

LOGARITMA

MATRIKS

PELUANG

PERSAMAAN KUADRAT

PLANIMETRI

POLINOM (SUKU BANYAK)

PROGRAM LINEAR

SISTEM PERSAMAAN LINEAR DAN KUADRAT

STATISTIK DATA BERKELOMPOK

STATISTIKA DATA TUNGGA

Transformasi

TRIGONOMETRI

TRIGONOMETRI 2

TRIGONOMETRI ATURAN SIN, ATURAN COS, DAN LUAS SEGITIGA

TURUNAN

$R_{90^\circ}$	$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$	$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$
$R_{180^\circ}$	$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$	$\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$
$R_{270^\circ}$	$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$	$\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$
$R_{(a,b),\alpha}$	$\begin{pmatrix} x' - a \\ y' - b \end{pmatrix} = \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix} \begin{pmatrix} x - a \\ y - b \end{pmatrix}$	$\begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix}$
$[(0, 0), k]$	$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} k & 0 \\ 0 & k \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$	$\begin{pmatrix} k & 0 \\ 0 & k \end{pmatrix}$
$[(a, b), k]$	$\begin{pmatrix} x' - a \\ y' - b \end{pmatrix} = \begin{pmatrix} k & 0 \\ 0 & k \end{pmatrix} \begin{pmatrix} x - a \\ y - b \end{pmatrix}$	$\begin{pmatrix} k & 0 \\ 0 & k \end{pmatrix}$

Di samping ini ada matriks transformasi yang mempunyai nama khusus yaitu:

a. Regangan  
Matriks transformasi:

regangan searah sumbu  $X$  dengan factor skala  $k$  adalah  $\begin{pmatrix} k & 0 \\ 0 & 1 \end{pmatrix}$

regangan searah sumbu  $Y$  dengan factor skala  $k$  adalah  $\begin{pmatrix} 1 & 0 \\ 0 & k \end{pmatrix}$

b. Gusuran  
Matriks transformasi :

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

Gambar 2.2.1 Beberapa pengopreasian matriks transformasi beserta kegunaannya.

## 2.3 OpenGL®.

OpenGL® adalah API grafik 2D dan 3D yang sudah diadposi sangat luas di dalam dunia industri dengan membawa ribuan aplikasi kepada berbagai *platform* komputer. Aplikasi ini juga independen terhadap sistem operasi. OpenGL memungkinkan para pengembang peranti lunak bagi komputer pribadi, komputer *workstation*, maupun peranti keras komputer super untuk menciptakan aplikasi grafik berperforma tinggi dan berpenampilan visual yang menarik bagi pasar, seperti CAD, penciptaan konten, energi, hiburan, pengembangan gim, manufaktur, medis, dan realitas maya.

## BAB 3

# IMPLEMENTASI PROGRAM

### 3.1 IMPLEMENTASI OPERASI MATRIKS

```
import
numpy
as np

# Represents float
def RepresentFloat(s):
    try:
        float(s)
        return True
    except ValueError:
        return False

# Create point
def createpoint2D(a, b):
    point = np.array([a, b])
    point = np.append(point, [0])
    return point
def createpoint3D(a, b, c):
    point = np.array([a, b, c])
    return point

# Input point
def inputpoint2D():
    instr = input('Enter a point (x,y): ')
    point = np.array([float(n) for n in instr.split(',')])
    point = np.append(point, [0])
    return point
def inputpoint3D():
    instr = input('Enter a point (x,y,z): ')
    point = np.array([float(n) for n in instr.split(',')])
    return point

# Translasi 2 dimensi
def Translate2(P, dx, dy):
    MTrans = np.array([dx,dy,0])
    P = np.add(MTrans, P)
    return(P)

# Translasi 3 dimensi
def Translate3(P, dx, dy, dz):
    MTrans = np.array([dx,dy,dz])
    P = np.add(MTrans, P)
    return(P)
```



```

# Dilatasi 2 dimensi
def Dilate2(P, k):
    MTrans = np.array([[k, 0, 0], [0, k, 0], [0, 0, 1]])
    P = np.dot(MTrans, P)
    return(P)

# Dilatasi 3 dimensi
def Dilate3(P, k):
    MTrans = np.array([[k, 0, 0], [0, k, 0], [0, 0, k]])
    P = np.dot(MTrans, P)
    return(P)

# Rotasi 2 dimensi
def Rotate2(P, sudut, P2):
    MTrans = np.array([[np.cos(np.deg2rad(sudut)), np.sin(np.deg2rad((-1)*sudut)), 0],
    [np.sin(np.deg2rad(sudut)), np.cos(np.deg2rad(sudut)), 0], [0, 0, 1]])
    P = np.add((np.dot(MTrans, np.subtract(P,P2))), P2)
    P = np.round(P, 2)
    return(P)

# Rotasi 3 dimensi
def Rotate3(P, sudut, P3, sumbu):
    P2 = np.array([P3[0],P3[1],P3[2]])
    a = np.cos(np.deg2rad(sudut))
    b = np.sin(np.deg2rad(sudut))
    # Rotasi terhadap sumbu
    if (sumbu == 'z') or (sumbu == 'Z'):
        MTrans = np.array([[a, b*(-1), 0], [b, a, 0], [0, 0, 1]])
        P = np.add((np.dot(MTrans, np.subtract(P,P2))), P2)
    elif (sumbu == 'x') or (sumbu == 'X'):
        MTrans = np.array([[1, 0, 0], [0, a, b*(-1)], [0, b, a]])
        P = np.add((np.dot(MTrans, np.subtract(P,P2))), P2)
    elif (sumbu == 'y') or (sumbu == 'Y'):
        MTrans = np.array([[a, 0, b], [0, 1, 0], [b*(-1), 0, a]])
        P = np.add((np.dot(MTrans, np.subtract(P,P2))), P2)
    # Rotasi terhadap garis
    # Sumbu dibuat dalam bentuk parametrik seperti (x,1,)
    else:
        print('Input tidak dikenal')

    P = np.round(P, 2)
    return (P)

# Refleksi 2 dimensi
def Reflect2(P, param):
    # Refleksi terhadap sumbu
    if (param == 'x') or (param == 'X'):
        MTrans = np.array([[1, 0, 0],[0, -1, 0], [0, 0, 1]])
        P = np.dot(MTrans, P)

```

```

        return(P)
    elif (param == 'y') or (param == 'Y'):
        MTrans = np.array([[ -1, 0, 0],[0, 1, 0], [0, 0, 1]])
        P = np.dot(MTrans, P)
        return(P)
    # Refleksi terhadap garis
    elif (param == 'y=x') or (param == 'Y=X'):
        MTrans = np.array([[0, 1, 0],[1, 0, 0], [0, 0, 1]])
        P = np.dot(MTrans, P)
        return(P)
    elif (param == 'y=-x') or (param == 'Y=-X'):
        MTrans = np.array([[0, -1, 0],[-1, 0, 0], [0, 0, 1]])
        P = np.dot(MTrans, P)
        return(P)
    # Refleksi terhadap titik
    else:
        param = param[1:-1]
        point = np.array([float(n) for n in param.split(',')])
        point = np.append(point, [0])
        P = np.subtract(np.dot(point, 2), P)
        return(P)

# Refleksi 3 dimensi
def Reflect3(P, param):
    # Refleksi terhadap bidang
    if (param == 'xz') or (param == 'XZ'):
        MTrans = np.array([[1, 0, 0],[0, -1, 0], [0, 0, 1]])
        P = np.dot(MTrans, P)
        return(P)
    elif (param == 'yz') or (param == 'YZ'):
        MTrans = np.array([[ -1, 0, 0],[0, 1, 0], [0, 0, 1]])
        P = np.dot(MTrans, P)
        return(P)
    elif (param == 'xy') or (param == 'XY'):
        MTrans = np.array([[1, 0, 0],[0, 1, 0], [0, 0, -1]])
        P = np.dot(MTrans, P)
        return(P)
    # Refleksi terhadap sumbu
    elif (param == 'x') or (param == 'X'):
        MTrans = np.array([[1, 0, 0],[0, -1, 0], [0, 0, -1]])
        P = np.dot(MTrans, P)
        return(P)
    elif (param == 'y') or (param == 'Y'):
        MTrans = np.array([[ -1, 0, 0],[0, 1, 0], [0, 0, -1]])
        P = np.dot(MTrans, P)
        return(P)
    elif (param == 'z') or (param == 'Z'):
        MTrans = np.array([[ -1, 0, 0],[0, -1, 0], [0, 0, 1]])
        P = np.dot(MTrans, P)

```

```

        return(P)
# Refleksi terhadap bidang lain
elif (param == 'y=x') or (param == 'Y=X'):
    MTrans = np.array([[0, 1, 0],[1, 0, 0], [0, 0, 1]])
    P = np.dot(MTrans, P)
    return(P)
elif (param == 'y=-x') or (param == 'Y=-X'):
    MTrans = np.array([[0, -1, 0],[-1, 0, 0], [0, 0, 1]])
    P = np.dot(MTrans, P)
    return(P)
elif (param == 'y=z') or (param == 'Y=Z'):
    MTrans = np.array([[1, 0, 0],[0, 0, 1], [0, 1, 0]])
    P = np.dot(MTrans, P)
    return(P)
elif (param == 'y=-z') or (param == 'Y=-Z'):
    MTrans = np.array([[1, 0, 0],[0, 0, -1], [0, -1, 0]])
    P = np.dot(MTrans, P)
    return(P)
elif (param == 'x=z') or (param == 'X=Z'):
    MTrans = np.array([[0, 0, 1],[0, 1, 0], [1, 0, 0]])
    P = np.dot(MTrans, P)
    return(P)
elif (param == 'x=-z') or (param == 'X=-Z'):
    MTrans = np.array([[0, 0, -1],[0, 1, 0], [-1, 0, 0]])
    P = np.dot(MTrans, P)
    return(P)
# Refleksi terhadap garis
elif (param == 'x=y=z') or (param == 'X=Y=Z'):
    x = (P[0]+P[1]+P[2])/3
    P2 = np.array([(x-P[0]), (x-P[1]), (x-P[2])])
    P = np.add(P, np.dot(P2, 2))
    return (P)
elif (param == 'x=y=-z') or (param == 'X=Y=-Z'):
    x = (P[0]+P[1]-P[2])/3
    P2 = np.array([(x-P[0]), (x-P[1]), (-x-P[2])])
    P = np.add(P, np.dot(P2, 2))
    return (P)
elif (param == 'x=-y=z') or (param == 'X=-Y=Z'):
    x = (P[0]-P[1]+P[2])/3
    P2 = np.array([(x-P[0]), (-x-P[1]), (x-P[2])])
    P = np.add(P, np.dot(P2, 2))
    return (P)
elif (param == '-x=y=z') or (param == '-X=Y=Z'):
    x = (-P[0]+P[1]+P[2])/3
    P2 = np.array([(x-P[0]), (x-P[1]), (x-P[2])])
    P = np.add(P, np.dot(P2, 2))
    return (P)
# Refleksi terhadap titik
else:

```

```

        param = param[1:-1]
        point = np.array([float(n) for n in param.split(',')])
        P = np.subtract(np.dot(point, 2), P)
        return(P)

# Shear 2 dimensi
def Shear2(P, sumbu, k):
    if (sumbu == 'x') or (sumbu == 'X'):
        MTrans = np.array([[1, k, 0], [0, 1, 0], [0, 0, 1]])
        P = np.dot(MTrans, P)
        return (P)
    elif (sumbu == 'y') or (sumbu == 'Y'):
        MTrans = np.array([[1, 0, 0], [k, 1, 0], [0, 0, 1]])
        P = np.dot(MTrans, P)
        return (P)
    else:
        print('Input tidak diterima')
        return (P)

# Shear 3 dimensi
def Shear3(P, sumbu, k1, k2):
    if (sumbu == 'x') or (sumbu == 'X'):
        MTrans = np.array([[1, k1, k2], [0, 1, 0], [0, 0, 1]])
        P = np.dot(MTrans, P)
        return (P)
    elif (sumbu == 'y') or (sumbu == 'Y'):
        MTrans = np.array([[1, 0, 0], [k1, 1, k2], [0, 0, 1]])
        P = np.dot(MTrans, P)
        return (P)
    elif (sumbu == 'z') or (sumbu == 'Z'):
        MTrans = np.array([[1, 0, 0], [0, 1, 0], [k1, k2, 1]])
        P = np.dot(MTrans, P)
        return (P)
    else:
        print('Input tidak diterima')
        return (P)

# Stretch 2 dimensi
def Stretch2(P, sumbu, k):
    if (sumbu == 'x') or (sumbu == 'X'):
        MTrans = np.array([[k, 0, 0], [0, 1, 0], [0, 0, 1]])
        P = np.dot(MTrans, P)
        return (P)
    elif (sumbu == 'y') or (sumbu == 'Y'):
        MTrans = np.array([[1, 0, 0], [0, k, 0], [0, 0, 1]])
        P = np.dot(MTrans, P)
        return (P)
    else:
        print('Input tidak diterima')

```

```

        return (P)

# Stretch 3 dimensi
def Stretch3(P, sumbu, k):
    if (sumbu == 'x') or (sumbu == 'X'):
        MTrans = np.array([[k, 0, 0], [0, 1, 0], [0, 0, 1]])
        P = np.dot(MTrans, P)
        return (P)
    elif (sumbu == 'y') or (sumbu == 'Y'):
        MTrans = np.array([[1, 0, 0], [0, k, 0], [0, 0, 1]])
        P = np.dot(MTrans, P)
        return (P)
    elif (sumbu == 'z') or (sumbu == 'Z'):
        MTrans = np.array([[1, 0, 0], [0, 1, 0], [0, 0, k]])
        P = np.dot(MTrans, P)
        return (P)
    else:
        print('Input tidak diterima')
        return (P)

# Custom 2 dimensi
def Custom2(P, a, b, c, d):
    MTrans = np.array([[a, b, 0], [c, d, 0], [0, 0, 1]])
    P = np.dot(MTrans, P)
    return (P)

# Custom 3 dimensi
def Custom3(P, a, b, c, d, e, f, g, h, i):
    MTrans = np.array([[a, b, c], [d, e, f], [g, h, i]])
    P = np.dot(MTrans, P)
    return (P)

```

### 3.2 IMPLEMENTASI PROGRAM 2D

```

import
numpy
as np

from operasimatriks import *

import pygame
from pygame.locals import *

from OpenGL.GL import *
from OpenGL.GLU import *

objects=[]

class obj:
    def __init__(self,verticies,overticies,edges,shape,color):

```

```

        self.v=vertices
        self.o=overticies
        self.e=edges
        self.s=shape
        self.c=color

    idx=0

    color=0

    verticies = []

    overticies=[]

    edges = []

    psq = [(1,1,0),(1,-1,0),(-1,-1,0),(-1,1,0)]

    ptri = [(1,0,0),(0,1,0),(-1,0,0)]

    shape=0

    colors = (
        (1,0,0),
        (0,1,0),
        (0,0,1),
        (0,1,1),
        (1,1,0),
        (1,0,1),
        (1,1,1),
        (0,0,0)
    )

    clock = pygame.time.Clock()

    def gamecontrol():
        for event in pygame.event.get():
            if event.type == KEYDOWN and event.key == K_ESCAPE:
                pygame.quit()
                quit()
            if event.type == KEYDOWN and event.key == K_RIGHT:
                glTranslatef(1,0.0,0.0)
            if event.type == KEYDOWN and event.key == K_LEFT:
                glTranslatef(-1,0.0,0)
            if event.type == KEYDOWN and event.key == K_UP:
                glTranslatef(0.0,1,0)
            if event.type == KEYDOWN and event.key == K_DOWN:
                glTranslatef(0.0,-1,0)
            if event.type == KEYDOWN and event.key == K_m:

```

```

        glTranslatef(0.0,0,1)
        if event.type == KEYDOWN and event.key == K_n:
            glTranslatef(0.0,0,-1)
    pygame.display.flip()
    clock.tick(60)

def addobject():
    global verticies
    global edges
    global objects
    global shape
    global idx
    global color
    verticies=[]
    edges=[]
    shape=input('Pilih bentuk:\n1. Polygon\n2. Segitiga\n3. Segiempat\n4. Lingkaran\n')
    shape=int(shape)
    if (shape==1):
        setPolygon()
    elif (shape==2):
        setTriangle()
    elif (shape==3):
        setSquare()
    elif (shape==4):
        setCircle()
    else:
        print('Input salah')
    ob=obj(verticies,overticies,edges,shape,color)
    objects.append(ob)
    idx=len(objects)-1

def sisi():
    global edges
    x=0
    y=1
    for s in verticies:
        z=(x,y)
        edges.append(z)
        x+=1
        y+=1
        y=(y%(len(verticies)))

def highlight():
    glBegin(GL_LINES)
    for edge in edges:
        for vertex in edge:
            glColor3fv((1,1,1))
            glVertex3fv(verticies[vertex])
    glEnd()

```

```

def setSquare():
    global verticies
    global color
    x=input('Masukkan warna: \n 1. Merah\n 2. Hijau\n 3. Biru\n 4. Cyan\n 5. Kuning\n
6. Magenta\n 7. Putih \n 8. Hitam\n')
    while(x<'1' or x>'8'):
        x=input('Salah input\n')
    color=int(x)-1
    preset = input('Use Preset?(Y/N)')
    if (preset=='Y'):
        verticies=psq
        sisi()
        Square()
    elif (preset=='N'):
        N=4
        print('Masukkan 4 titik (x,y)')
        while (N > 0):
            p=inputpoint2d()
            verticies.append(p)
            N-=1
        sisi()
        Square()
    else:
        print('Wrong input')
    global overties
    overties=verticies

def Square():
    glBegin(GL_QUADS)
    for vertex in verticies:
        glColor3fv(colors[color])
        glVertex3fv(vertex)
    glEnd()

def setPolygon():
    global verticies
    global color
    x=input('Masukkan warna: \n 1. Merah\n 2. Hijau\n 3. Biru\n 4. Cyan\n 5. Kuning\n
6. Magenta\n 7. Putih \n 8. Hitam\n')
    while(x<'1' or x>'8'):
        x=input('Salah input\n')
    color=int(x)-1
    N=input('Masukkan jumlah sudut (Sudut-sudut yang dimasukkan harus berurutan)\n')
    N=int(N)
    if (N<=1):
        N=input('Jumlah sudut harus diatas 1\n')
    while (N > 0):

```



```

        p=inputpoint2d()
        vertices.append(p)
        N-=1
    sisi()
    Polygon()
    global overties
    overties=vertices

def Polygon():
    glBegin(GL_POLYGON)
    for vertex in vertices:
        glColor3fv(colors[color])
        glVertex3fv(vertex)
    glEnd()

def setCircle():
    global vertices
    global color
    x=input('Masukkan warna: \n 1. Merah\n 2. Hijau\n 3. Biru\n 4. Cyan\n 5. Kuning\n
6. Magenta\n 7. Putih \n 8. Hitam\n')
    while(x<'1' or x>'8'):
        x=input('Salah input\n')
    color=int(x)-1
    r=input('Masukkan radius\n')
    r=float(r)
    sudut=360
    while (sudut>0):
        p=(np.cos(np.deg2rad(sudut))*r,np.sin(np.deg2rad(sudut))*r,0)
        vertices.append(p)
        sudut-=1
    sisi()
    Circle()
    global overties
    overties=vertices

def Circle():
    glBegin(GL_POLYGON)
    for vertex in vertices:
        glColor3fv(colors[color])
        glVertex3fv(vertex)
    glEnd()

def setTriangle():
    global vertices
    global color
    x=input('Masukkan warna: \n 1. Merah\n 2. Hijau\n 3. Biru\n 4. Cyan\n 5. Kuning\n
6. Magenta\n 7. Putih \n 8. Hitam\n')
    while(x<'1' or x>'8'):
        x=input('Salah input\n')

```

```

color=int(x)-1
preset = input('Use Preset?(Y/N)')
if (preset=='Y'):
    verticies=ptri
    sisi()
    Triangle()
elif (preset=='N'):
    N=3
    print('Masukkan 3 titik (x,y)')
    while (N>0):
        p=inputpoint2d()
        verticies.append(p)
        N-=1
    sisi()
    Triangle()
else:
    print('Wrong input')
global overties
overties=verticies

def Triangle():
    glBegin(GL_TRIANGLES)
    for vertex in verticies:
        glColor3fv(colors[color])
        glVertex3fv(vertex)
    glEnd()

def Cartesius():
    glBegin(GL_LINES)
    glColor3fv((1,1,1))
    glVertex3fv((0,-500,0))
    glVertex3fv((0,500,0))
    glVertex3fv((500,0,0))
    glVertex3fv((-500,0,0))
    glEnd()

def inputpoint2d():
    valid=False
    TruList=[]
    while (not(valid)) or (not(len(TruList) == 2)):
        TruList=[]
        instr = input('Enter a point (x,y): ')
        inList = [n for n in instr.split(',')]
        for n in inList:
            valid=RepresentFloat(n)
            if (not(valid)):
                print('Input Salah!\n')
                break
        else:

```

```

        TruList.append(float(n))
    if (not(len(TruList) == 2)):
        print('Input harus x,y\n')
    TruList.append(0)
    return tuple(TruList)

def refresh():
    global verticies
    global edges
    global shape
    global color
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    for ob in objects:
        verticies=ob.v
        edges=ob.e
        shape=ob.s
        color=ob.c
        if (shape==1):
            Polygon()
        elif (shape==2):
            Triangle()
        elif (shape==3):
            Square()
        elif (shape==4):
            Circle()
        if(ob==objects[idx]):
            highlight()
    verticies=objects[idx].v
    edges=objects[idx].e
    shape=objects[idx].s
    color=objects[idx].c
    Cartesius()
    pygame.display.flip()
    clock.tick(60)

def animate(nv):
    global verticies
    global objects
    a=np.array(nv)
    b=np.array(verticies)
    delta=np.divide(np.subtract(a,b),100)
    N=100
    while (N>0):
        verticies=np.add(verticies,delta)
        objects[idx].v=verticies
        refresh()
        gamecontrol()
        pygame.time.wait(10)
        N-=1

```

```

def animaterotate(sudut,P2):
    global verticies
    global objects
    delta=sudut/100
    N=100
    while (N>0):
        nverticies=[]
        for vertex in verticies:
            vx=createpoint2D(vertex[0],vertex[1])
            v=Rotate2(list(vx),delta,P2)
            x=v.tolist()
            nverticies.append(x)
        verticies=nverticies
        objects[idx].v=verticies
        refresh()
        gamecontrol()
        pygame.time.wait(50)
        N-=1

def operate(opr):
    valid=False
    if (opr[0]=='translate'):
        if (len(opr)==3):
            if(RepresentFloat(opr[1]) and RepresentFloat(opr[2])):
                dx=float(opr[1])
                dy=float(opr[2])
                nverticies=[]
                for vertex in verticies:
                    vx=createpoint3D(vertex[0],vertex[1],vertex[2])
                    v=Translate2(list(vx),dx,dy)
                    x=v.tolist()
                    nverticies.append(x)
                animate(nverticies)
                valid=True
            elif (opr[0]=='dilate'):
                if(len(opr)==2):
                    if(RepresentFloat(opr[1])):
                        k=float(opr[1])
                        nverticies=[]
                        for vertex in verticies:
                            vx=createpoint3D(vertex[0],vertex[1],vertex[2])
                            v=Dilate2(list(vx),k)
                            x=v.tolist()
                            nverticies.append(x)
                        animate(nverticies)
                        valid=True
            elif (opr[0]=='rotate'):
                if(len(opr)==4):

```

```

        if(RepresentFloat(opr[1]) and RepresentFloat(opr[2]) and
RepresentFloat(opr[3])):
            sudut=float(opr[1])
            P1=(float(opr[2]),float(opr[3]),0)
            P2=createpoint3D(P1[0],P1[1],P1[2])
            animaterotate(sudut,P2)
            valid=True
    elif(opr[0]=='reflect'):
        if(len(opr)==2):
            if(True):
                param=input('Masukkan parameter:\n')
                nverticies=[]
                for vertex in verticies:
                    vx=createpoint3D(vertex[0],vertex[1],vertex[2])
                    v=Reflect2(list(vx),param)
                    x=v.tolist()
                    nverticies.append(x)
                animate(nverticies)
                valid=True
    elif(opr[0]=='shear'):
        if(len(opr)==3):
            if(RepresentFloat(opr[2])):
                sumbu=opr[1]
                k=float(opr[2])
                nverticies=[]
                for vertex in verticies:
                    vx=createpoint3D(vertex[0],vertex[1],vertex[2])
                    v=Shear2(list(vx),sumbu,k)
                    x=v.tolist()
                    nverticies.append(x)
                animate(nverticies)
                valid=True
    elif(opr[0]=='stretch'):
        if(len(opr)==3):
            if(RepresentFloat(opr[2])):
                sumbu=opr[1]
                k=float(opr[2])
                nverticies=[]
                for vertex in verticies:
                    vx=createpoint3D(vertex[0],vertex[1],vertex[2])
                    v=Stretch2(list(vx),sumbu,k)
                    x=v.tolist()
                    nverticies.append(x)
                animate(nverticies)
                valid=True
    elif(opr[0]=='custom'):
        if(len(opr)==5):
            if(RepresentFloat(opr[1]) and RepresentFloat(opr[2]) and
RepresentFloat(opr[3]) and RepresentFloat(opr[4])):

```

```

a=float(opr[1])
b=float(opr[2])
c=float(opr[3])
d=float(opr[4])
nverticies=[]
for vertex in verticies:
    vx=createpoint3D(vertex[0],vertex[1],vertex[2])
    v=Custom2(list(vx),a,b,c,d)
    x=v.tolist()
    nverticies.append(x)
animate(nverticies)
valid=True
elif(opr[0]=='multi'):
    if(len(opr)==2):
        if(RepresentFloat(opr[1])):
            n = opr[1]
            print('Multi ', n, ' kali START')
            nverticies=[]
            mverticies=verticies
            while (n>0):
                print ('Masukkan operasi:\n1. Translasi\n2. Dilatasi\n3. Rotasi\n4.
Refleksi\n5. Shear\n6. Stretch\n7. Custom\n')
                opsx = input('Masukkan operan')
                opr =[n for n in opsx.split(' ')]
                if (opr[0]=='translate'):
                    if (len(opr)==3):
                        if(RepresentFloat(opr[1]) and RepresentFloat(2)):
                            dx=float(opr[1])
                            dy=float(opr[2])
                            nverticies=[]
                            for vertex in verticies:
                                vx=createpoint3D(vertex[0],vertex[1],vertex[2])
                                v=Translate2(list(vx),dx,dy)
                                x=v.tolist()
                                nverticies.append(x)
                            animate(nverticies)
                            valid=True
                    elif (opr[0]=='dilate'):
                        if(len(opr)==2):
                            if(RepresentFloat(opr[1])):
                                k=float(opr[1])
                                nverticies=[]
                                for vertex in verticies:
                                    vx=createpoint3D(vertex[0],vertex[1],vertex[2])
                                    v=Dilate2(list(vx),k)
                                    x=v.tolist()
                                    nverticies.append(x)
                                animate(nverticies)
                                valid=True

```

```

elif (opr[0]=='rotate'):
    if(len(opr)==4):
        if(RepresentFloat(opr[1]) and RepresentFloat(opr[2]) and
RepresentFloat(opr[3])):
            sudut=float(opr[1])
            P1=(float(opr[2]),float(opr[3]),0)
            P2=createpoint3D(P1[0],P1[1],P1[2])
            animaterotate(sudut,P2)
            valid=True
elif(opr[0]=='reflect'):
    if(len(opr)==2):
        if(True):
            param=input('Masukkan parameter:\n')
            nverticies=[]
            for vertex in verticies:
                vx=createpoint3D(vertex[0],vertex[1],vertex[2])
                v=Reflect2(list(vx),param)
                x=v.tolist()
                nverticies.append(x)
            animate(nverticies)
            valid=True
elif(opr[0]=='shear'):
    if(len(opr)==3):
        if(RepresentFloat(opr[2])):
            sumbu=opr[1]
            k=float(opr[2])
            nverticies=[]
            for vertex in verticies:
                vx=createpoint3D(vertex[0],vertex[1],vertex[2])
                v=Shear2(list(vx),sumbu,k)
                x=v.tolist()
                nverticies.append(x)
            animate(nverticies)
            valid=True
elif(opr[0]=='stretch'):
    if(len(opr)==3):
        if(RepresentFloat(opr[2])):
            sumbu=opr[1]
            k=float(opr[2])
            nverticies=[]
            for vertex in verticies:
                vx=createpoint3D(vertex[0],vertex[1],vertex[2])
                v=Stretch2(list(vx),sumbu,k)
                x=v.tolist()
                nverticies.append(x)
            animate(nverticies)
            valid=True
elif(opr[0]=='custom'):
    if(len(opr)==5):

```

```

        if(RepresentFloat(opr[1]) and RepresentFloat(opr[2]) and
RepresentFloat(opr[3]) and RepresentFloat(opr[4])):
            a=float(opr[1])
            b=float(opr[2])
            c=float(opr[3])
            d=float(opr[4])
            nverticies=[]
            for vertex in verticies:
                vx=createpoint3D(vertex[0],vertex[1],vertex[2])
                v=Custom2(list(vx),a,b,c,d)
                x=v.tolist()
                nverticies.append(x)
            animate(nverticies)
            valid=True
            if(valid==False):
                print('Input salah')
            n -= 1
            animate(mverticies)
            valid=True
        elif(opr[0]=='reset'):
            nverticies=overticies
            animate(nverticies)
            valid=True
        if(valid==False):
            print('Input salah')

def main2():
    pygame.init()

    global objects
    global verticies
    global edges
    global shape
    global idx
    global color

    display = (800,600)

    pygame.display.set_mode(display, DOUBLEBUF|OPENGL)

    addobject()
    Cartesius()

    pygame.display.set_caption('Algeo Tubes Edition')
    pygame.display.set_mode(display, DOUBLEBUF|OPENGL)
    gluPerspective(45, (display[0]/display[1]), 0.1, 50.0)
    glTranslatef(0.0,0.0, -5)
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

```



```

    op='Masukkan operasi:\n1. Translasi\n2. Dilatasi\n3. Rotasi\n4. Refleksi\n5.
    Shear\n6. Stretch\n7. Custom\n8. Multi\n9. Reset\n'
    print('Menunggu input dari window pygame...\n r=operasi\n c=cycle\n a=addobject\n
    m/n=zoom in/zoom out\n')
    while True:
        for event in pygame.event.get():
            if event.type == KEYDOWN and event.key == K_ESCAPE:
                pygame.quit()
                quit()
            if event.type == KEYDOWN and event.key == K_r:
                opxs=input(op)
                opxs=[n for n in opxs.split(' ')]
                operate(opxs)
                print('Menunggu input dari window pygame...\n r=operasi\n c=cycle\n
a=addobject\n')
            if event.type == KEYDOWN and event.key == K_a:
                addobject()
                print('Menunggu input dari window pygame...\n r=operasi\n c=cycle\n
a=addobject\n')
            if event.type == KEYDOWN and event.key == K_c:
                idx+=1
                idx=idx%len(objects)
                color=objects[idx].c
                verticies=objects[idx].v
                edges=objects[idx].e
                shape=objects[idx].s
            if event.type == KEYDOWN and event.key == K_RIGHT:
                glTranslatef(0.2,0.0,0.0)
            if event.type == KEYDOWN and event.key == K_LEFT:
                glTranslatef(-0.2,0.0,0)
            if event.type == KEYDOWN and event.key == K_UP:
                glTranslatef(0.0,0.2,0)
            if event.type == KEYDOWN and event.key == K_DOWN:
                glTranslatef(0.0,-0.2,0)
            if event.type == KEYDOWN and event.key == K_m:
                glTranslatef(0.0,0,0.2)
            if event.type == KEYDOWN and event.key == K_n:
                glTranslatef(0.0,0,-0.2)
        refresh()

main2()

```

### 3.3 IMPLEMENTASI PROGRAM 3D

```

import
numpy
as np

from operasimatriks import *

```

```

import pygame
from pygame.locals import *

from OpenGL.GL import *
from OpenGL.GLU import *

clock = pygame.time.Clock()

suduts = [
    (1, -1, -1),
    (1, 1, -1),
    (-1, 1, -1),
    (-1, -1, -1),
    (1, -1, 1),
    (1, 1, 1),
    (-1, -1, 1),
    (-1, 1, 1)
]

verticies = []
overticies = []

edges = [
    (0,1),
    (0,3),
    (0,4),
    (2,1),
    (2,3),
    (2,7),
    (6,3),
    (6,4),
    (6,7),
    (5,1),
    (5,4),
    (5,7)
]

colors = [
    (1,1,0),
    (0,1,0),
    (0,0,1),
    (0,1,0),
    (1,1,1),
    (0,1,1),
    (1,1,0),
    (0,1,0),
    (0,0,1),
    (0,1,0),
    (1,1,1),

```

```

        (0,1,1),
    ]

surfaces = [
    (0,1,2,3),
    (3,2,7,6),
    (6,7,5,4),
    (4,5,1,0),
    (1,5,7,2),
    (4,0,3,6),
]

def gamecontrol():
    for event in pygame.event.get():
        if event.type == KEYDOWN and event.key == K_ESCAPE:
            pygame.quit()
            quit()
        if event.type == KEYDOWN and event.key == K_RIGHT:
            glTranslatef(1,0.0,0.0)
        if event.type == KEYDOWN and event.key == K_LEFT:
            glTranslatef(-1,0.0,0)
        if event.type == KEYDOWN and event.key == K_UP:
            glTranslatef(0.0,1,0)
        if event.type == KEYDOWN and event.key == K_DOWN:
            glTranslatef(0.0,-1,0)
        if event.type == KEYDOWN and event.key == K_m:
            glTranslatef(0.0,0,1)
        if event.type == KEYDOWN and event.key == K_n:
            glTranslatef(0.0,0,-1)
    pygame.display.flip()
    clock.tick(60)

def setCube():
    global verticies
    preset = input('Use Preset?(Y/N)')
    if (preset=='Y'):
        verticies=suduts
        Cube()

    elif (preset=='N'):
        N=8
        print('Masukkan 8 titik (x,y,z)')
        while (N > 0):
            p=inputpoint3d()
            verticies.append(p)
            N-=1

```

```

        Cube()
    else:
        print('Wrong input')
    global overties
    overties =verticies

def Cube():
    glBegin(GL_QUADS)
    for surface in surfaces:
        x = 0
        for vertex in surface:
            x+=1
            glColor3fv(colors[x])
            glVertex3fv(verticies[vertex])
    glEnd()

    glBegin(GL_LINES)
    for edge in edges:
        for vertex in edge:
            glVertex3fv(verticies[vertex])
    glEnd()

def Cartesius():
    glBegin(GL_LINES)
    glColor3fv((1,0,1))
    glVertex3fv((0,-500,0))
    glVertex3fv((0,500,0))
    glColor3fv((1,0,0))
    glVertex3fv((500,0,0))
    glVertex3fv((-500,0,0))
    glColor3fv((0,0,1))
    glVertex3fv((0,0,500))
    glVertex3fv((0,0,-500))
    glEnd()

def animate(nv):
    global verticies
    a=np.array(nv)
    b=np.array(verticies)
    delta=np.divide(np.subtract(a,b),100)

    N=100
    while (N>0):
        verticies=np.add(verticies,delta)
        refresh()
        gamecontrol()
        pygame.time.wait(50)
        N-=1

```

```

def animaterotate(sudut,P2,sumbu):
    global verticies
    delta=sudut/100
    N=100
    while (N>0):
        nverticies=[]
        for vertex in verticies:
            vx=createpoint3D(vertex[0],vertex[1],vertex[2])
            v=Rotate3(list(vx),delta,P2,sumbu)
            x=v.tolist()
            nverticies.append(x)
        verticies=nverticies
        refresh()
        gamecontrol()
        pygame.time.wait(50)
        N-=1

def operate(opr):
    print(opr)
    valid = False
    if (opr[0]=='translate'): #translasi
        if(len(opr) == 4):
            if (RepresentFloat(opr[1]) and RepresentFloat(opr[2]) and
RepresentFloat(opr[3])):
                dx = float(opr[1])
                dy = float(opr[2])
                dz = float(opr[3])
                nverticies=[]
                for vertex in verticies:
                    vx=createpoint3D(vertex[0],vertex[1],vertex[2])
                    v=Translate3(list(vx),dx,dy,dz)
                    x=v.tolist()
                    nverticies.append(x)
                animate(nverticies)
                valid = True

    elif (opr[0]=='dilate'): #dilasi
        if(len(opr) == 2):
            if(RepresentFloat(opr[1])):
                k = float(opr[1])
                nverticies=[]
                for vertex in verticies:
                    vx=createpoint3D(vertex[0],vertex[1],vertex[2])
                    v=Dilate3(list(vx),k)
                    x=v.tolist()
                    nverticies.append(x)

```

```

        animate(nverticies)
        valid = True

    elif (opr[0] == 'rotate'): #rotasi
        if(len(opr) == 6):
            if(RepresentFloat(opr[1]) and RepresentFloat(opr[2]) and
RepresentFloat(opr[3]) and RepresentFloat(opr[4])):
                sumbu=opr[5]
                sudut= float(opr[1])
                P1 = (float(opr[2]), float(opr[3]), float(opr[4]))
                P2=createpoint3D(P1[0],P1[1],P1[2])

                animaterotate(sudut,P2,sumbu)
                valid = True

    elif(opr[0] == 'reflect'): #refleksi
        if(len(opr) == 2):
            if(True):
                param=input('Masukkan parameter:\n')
                nverticies=[]
                for vertex in verticies:
                    vx=createpoint3D(vertex[0],vertex[1],vertex[2])
                    v=Reflect3(list(vx),param)
                    x=v.tolist()
                    nverticies.append(x)
                animate(nverticies)
                valid = True

    elif(opr[0]=='shear'): #shear
        if(len(opr) == 4):
            if(RepresentFloat(opr[2]) and RepresentFloat(opr[3])):
                sumbu = opr[1]
                k1 = float(opr[2])
                k2 = float(opr[3])
                nverticies=[]
                for vertex in verticies:
                    vx=createpoint3D(vertex[0],vertex[1],vertex[2])
                    v=Shear3(list(vx),sumbu,k1,k2)
                    x=v.tolist()
                    nverticies.append(x)
                animate(nverticies)
                valid = True

    elif(opr[0]=='stretch'): #stretch
        if(len(opr)==3):
            if(RepresentFloat(opr[2])):
                sumbu=opr[1]

```

```

        k=float(opr[2])
        nverticies=[]
        for vertex in verticies:
            vx=createpoint3D(vertex[0],vertex[1],vertex[2])
            v=Stretch2(list(vx),sumbu,k)
            x=v.tolist()
            nverticies.append(x)
        animate(nverticies)
        valid=True

    elif(opr[0]=='custom'): #custom
        if(len(opr) == 10):
            if(RepresentFloat(opr[1]) and RepresentFloat(opr[2]) and
RepresentFloat(opr[3]) and RepresentFloat(opr[4]) and RepresentFloat(opr[5]) and
RepresentFloat(opr[6]) and RepresentFloat(opr[7]) and RepresentFloat(opr[8]) and
RepresentFloat(opr[9])):
                a=float(opr[1])
                b=float(opr[2])
                c=float(opr[3])
                d=float(opr[4])
                e=float(opr[5])
                f=float(opr[6])
                g=float(opr[7])
                h=float(opr[8])
                i=float(opr[9])
                nverticies=[]
                for vertex in verticies:
                    vx=createpoint3D(vertex[0],vertex[1],vertex[2])
                    v=Custom3(list(vx),a,b,c,d,e,f,g,h,i)
                    x=v.tolist()
                    nverticies.append(x)
                animate(nverticies)
                valid = True

    elif(opr[0]=='multi'): #multiple
        if(len(opr) == 2):
            if(RepresentFloat(opr[1])):
                n = opr[1]
                print('Multi ', n, ' kali START')
                nverticies=[]
                mverticies=verticies
                while (n>0):
                    print ('Masukkan operasi:\n1. Translasi\n2. Dilatasi\n3. Rotasi\n4.
Refleksi\n5. Shear\n6. Stretch\n7. Multi\n')
                    oprx = input('Masukkan operan')
                    opr = [n for n in oprx.split(' ')]
                    if (opr[0]=='translate'): #translasi
                        if(len(opr) == 4):

```

```

        if (RepresentFloat(opr[1]) and RepresentFloat(opr[2]) and
RepresentFloat(opr[3])):

            dx = float(opr[1])
            dy = float(opr[2])
            dz = float(opr[3])
            nverticies=[]
            for vertex in verticies:
                vx=createpoint3D(vertex[0],vertex[1],vertex[2])
                v=Translate3(list(vx),dx,dy,dz)
                x=v.tolist()
                nverticies.append(x)
            animate(nverticies)
            valid = True

elif (opr[0]=='dilate'): #dilasi
    if(len(opr) == 2):
        if(RepresentFloat(opr[1])):
            k = float(opr[1])
            nverticies=[]
            for vertex in verticies:
                vx=createpoint3D(vertex[0],vertex[1],vertex[2])
                v=Dilate3(list(vx),k)
                x=v.tolist()
                nverticies.append(x)
            animate(nverticies)
            valid = True

elif (opr[0] == 'rotate'): #rotasi
    if(len(opr) == 6):
        if(RepresentFloat(opr[1]) and RepresentFloat(opr[2]) and
RepresentFloat(opr[3]) and RepresentFloat(opr[4])):
            sumbu=opr[5]
            sudut= float(opr[1])
            P1 = (float(opr[2]), float(opr[3]), float(opr[4]))
            P2=createpoint3D(P1[0],P1[1],P1[2])

            animaterotate(sudut,P2,sumbu)
            valid = True

elif(opr[0] == 'reflect'): #refleksi
    if(len(opr) == 2):
        if(True):
            param=input('Masukkan parameter:\n')
            nverticies=[]
            for vertex in verticies:

```



```

        vx=createpoint3D(vertex[0],vertex[1],vertex[2])
        v=Reflect3(list(vx),param)
        x=v.tolist()
        nverticies.append(x)
        animate(nverticies)
        valid = True

elif(opr[0]=='shear'): #shear
    if(len(opr) == 4):
        if(RepresentFloat(opr[2]) and RepresentFloat(opr[3])):
            sumbu = opr[1]
            k1 = float(opr[2])
            k2 = float(opr[3])
            nverticies=[]
            for vertex in verticies:
                vx=createpoint3D(vertex[0],vertex[1],vertex[2])
                v=Shear3(list(vx),sumbu,k1,k2)
                x=v.tolist()
                nverticies.append(x)
            animate(nverticies)
            valid = True

elif(opr[0]=='stretch'): #stretch
    if(len(opr)==3):
        if(RepresentFloat(opr[2])):
            sumbu=opr[1]
            k=float(opr[2])
            nverticies=[]
            for vertex in verticies:
                vx=createpoint3D(vertex[0],vertex[1],vertex[2])
                v=Stretch2(list(vx),sumbu,k)
                x=v.tolist()
                nverticies.append(x)
            animate(nverticies)
            valid=True

elif(opr[0]=='custom'): #custom
    if(len(opr) == 10):
        if(RepresentFloat(opr[1]) and RepresentFloat(opr[2]) and
RepresentFloat(opr[3]) and RepresentFloat(opr[4]) and RepresentFloat(opr[5]) and
RepresentFloat(opr[6]) and RepresentFloat(opr[7]) and RepresentFloat(opr[8]) and
RepresentFloat(opr[9])):
            a=float(opr[1])
            b=float(opr[2])
            c=float(opr[3])
            d=float(opr[4])
            e=float(opr[5])

```

```

        f=float(opr[6])
        g=float(opr[7])
        h=float(opr[8])
        i=float(opr[9])
        nverticies=[]
        for vertex in verticies:
            vx=createpoint3D(vertex[0],vertex[1],vertex[2])
            v=Custom3(list(vx),a,b,c,d,e,f,g,h,i)
            x=v.tolist()
            nverticies.append(x)
        animate(nverticies)
        valid = True
    if(valid == False):
        print('Input salah')
        n -= 1
        animate(mverticies)
        valid = True
elif(opr[0] == 'reset'):
    nverticies = oververticies
    animate(nverticies)
    valid = True
if (valid == False):
    print('Input salah')

def inputpoint3d():
    valid = False
    TruList=[]
    while(not(valid)) or (not(len(TruList) == 3)):
        TruList=[]
        instr = input('Enter a point (x,y,z): ')
        inList = [float(n) for n in instr.split(',')]
        for n in inList:
            valid = RepresentFloat(n)
            if (not(valid)):
                print('Input Salah!\n')
                break
            else:
                TruList.append(float(n))
        if (not(len(inList) == 3)) :
            print('Input harus (x,y,z)\n')
    TruList.append(0)
    return tuple(TruList)

def refresh():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    Cartesius()
    Cube()

```

```

pygame.display.flip()
clock.tick(60)

def main3():
    pygame.init()

    display = (800,600)

    pygame.display.set_mode(display, DOUBLEBUF|OPENGL)

    Cartesius()
    setCube()

    pygame.display.set_caption('Algeo Yeah!!!')
    pygame.display.set_mode(display, DOUBLEBUF|OPENGL)
    gluPerspective(45, (display[0]/display[1]), 0.1, 50.0)
    glTranslatef(0.0,0.0, -5)
    gluLookAt(3.0, 3.0, 5.0, 0.0, 0.0, 0.0, 0.0, 5.0, 0.0)
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)

    op='Masukkan operasi:\n1. Translasi\n2. Dilatasi\n3. Rotasi\n4. Refleksi\n5.
    Shear\n6. Stretch\n7. Custom\n8.Multi \n9.Reset\n'
    print('Menunggu input dari window pygame...\n r=operasi\n m/n=zoom in/zoom out\n')
    while True:
        for event in pygame.event.get():
            if event.type == KEYDOWN and event.key == K_ESCAPE:
                pygame.quit()
                quit()
            if event.type == KEYDOWN and event.key == K_r:
                operasi=input(op)
                operasi = [n for n in operasi.split(' ')]
                operate(operasi)
                print('Menunggu input dari window pygame...\n r=operasi\n m/n=zoom
in/zoom out\n')
            if event.type == KEYDOWN and event.key == K_RIGHT:
                glTranslatef(1,0.0,0.0)
            if event.type == KEYDOWN and event.key == K_LEFT:
                glTranslatef(-1,0.0,0)
            if event.type == KEYDOWN and event.key == K_UP:
                glTranslatef(0.0,1,0)
            if event.type == KEYDOWN and event.key == K_DOWN:
                glTranslatef(0.0,-1,0)
            if event.type == KEYDOWN and event.key == K_m:
                glTranslatef(0.0,0,1)
            if event.type == KEYDOWN and event.key == K_n:
                glTranslatef(0.0,0,-1)

        refresh()

```

### 3.4 PEMBAGIAN TUGAS

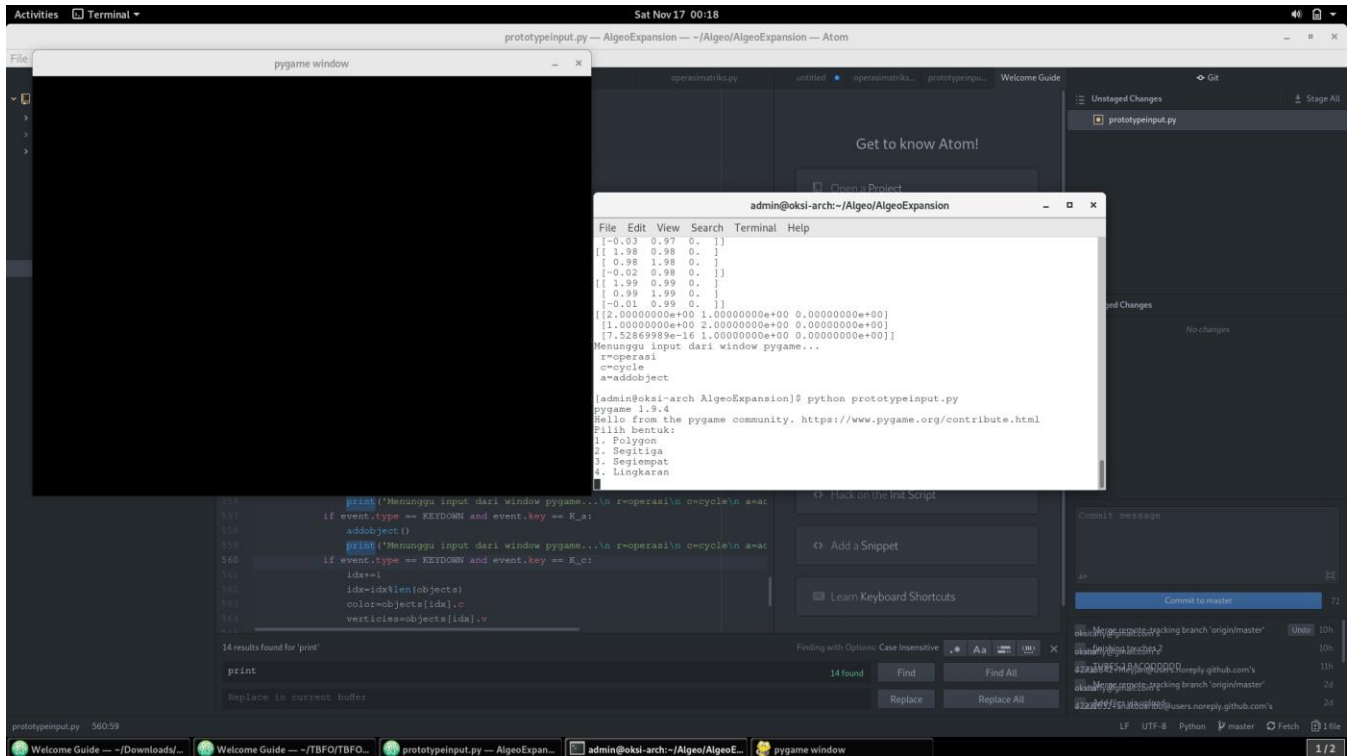
Dalam pengerjaan tugas besar ini, kami melakukan pembagian tugas dengan proporsi yang dijabarkan di bawah ini :

- |                            |  |
|----------------------------|--|
| 1. T. Andra Oksidian Tafly | : Membuat program grafik 2D dan laporan            |
| 2. Andrian Cedric          | : Membuat program grafik 3D dan laporan            |
| 3. Jan Meyer Saragih       | : Membuat program transformasi matriks dan laporan |

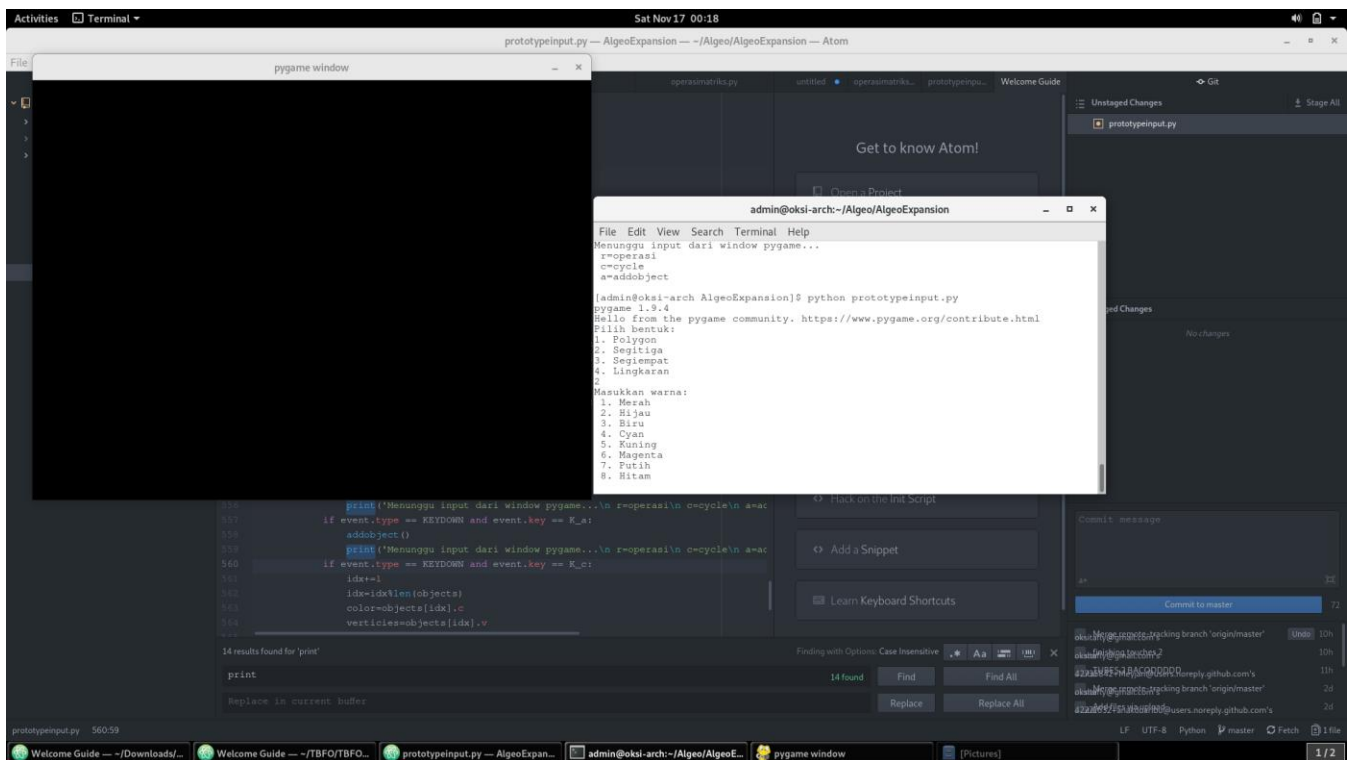
# BAB 4

## EKSPERIMEN

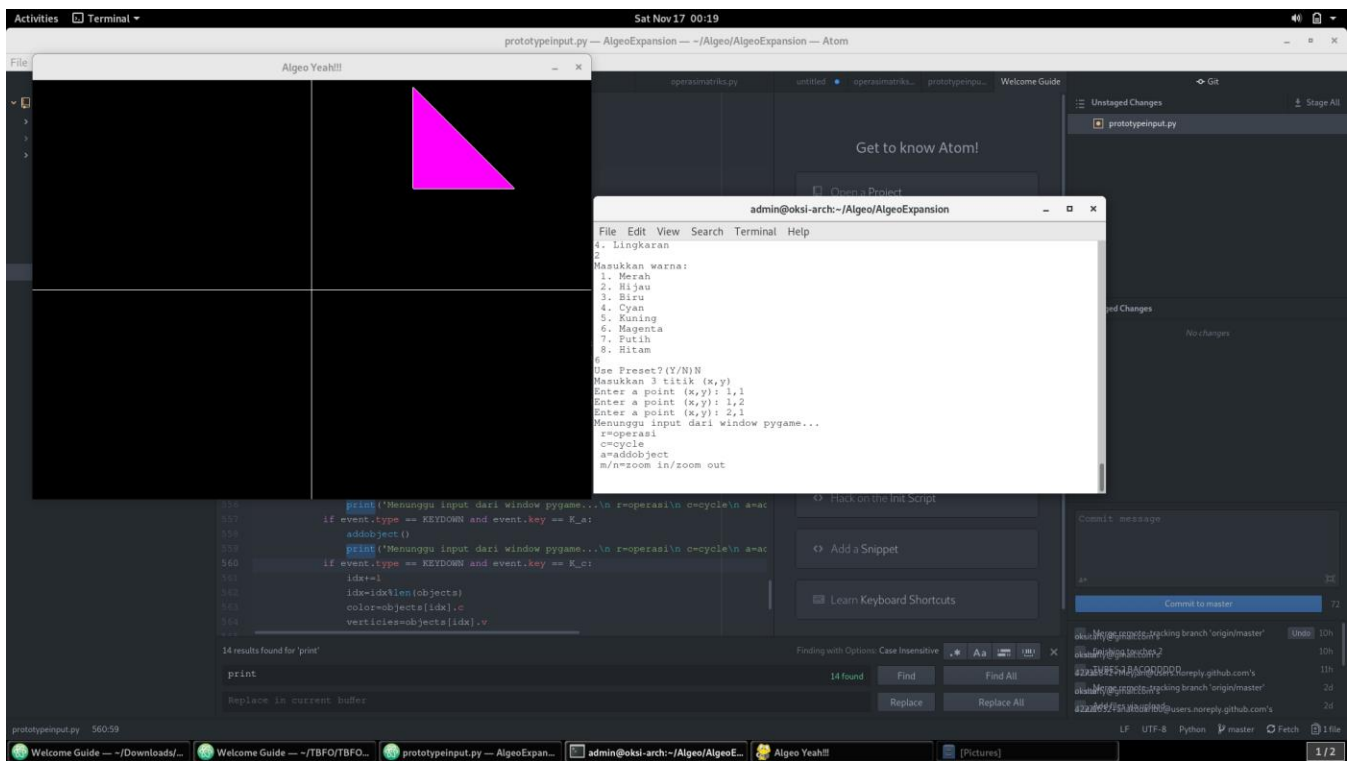
### 4.1 HASIL SCREENSHOT OPERASI 2D



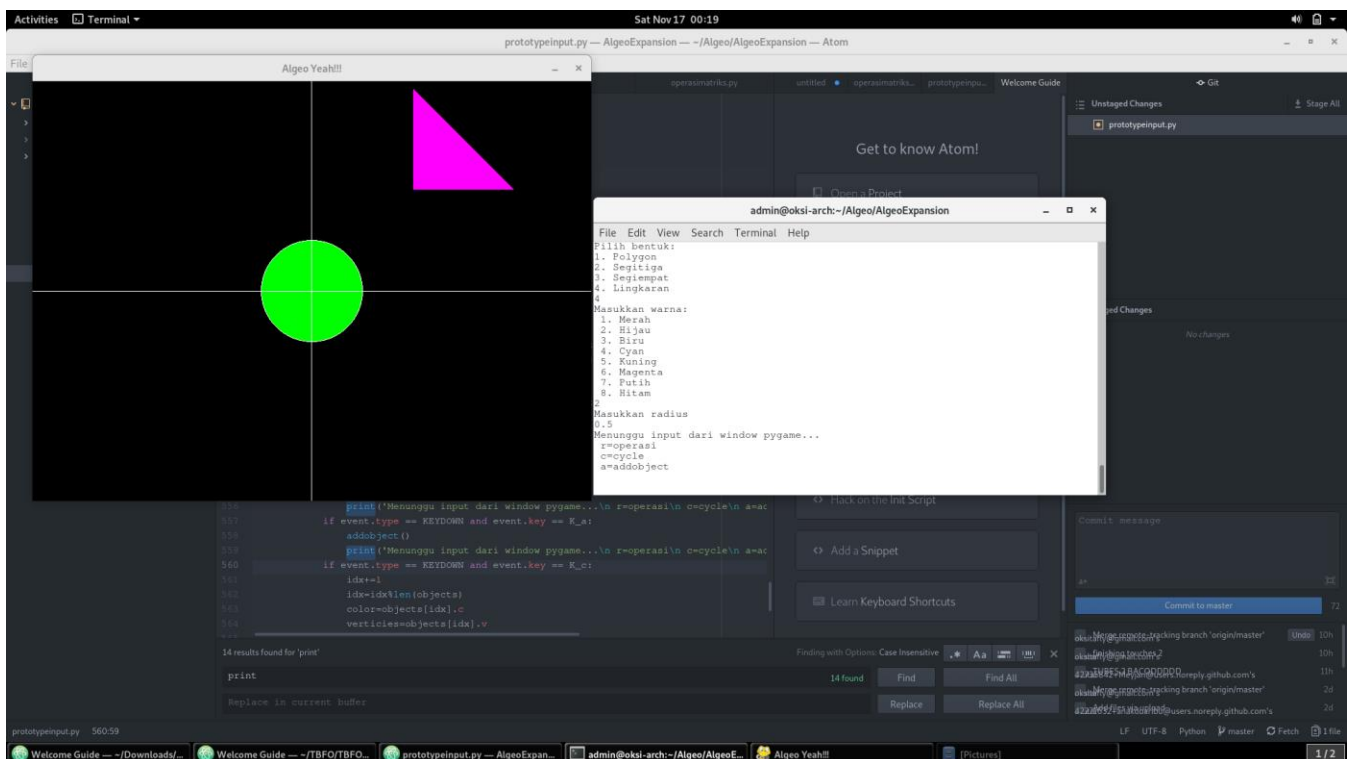
Gambar 4.1.1 Tampilan utama aplikasi



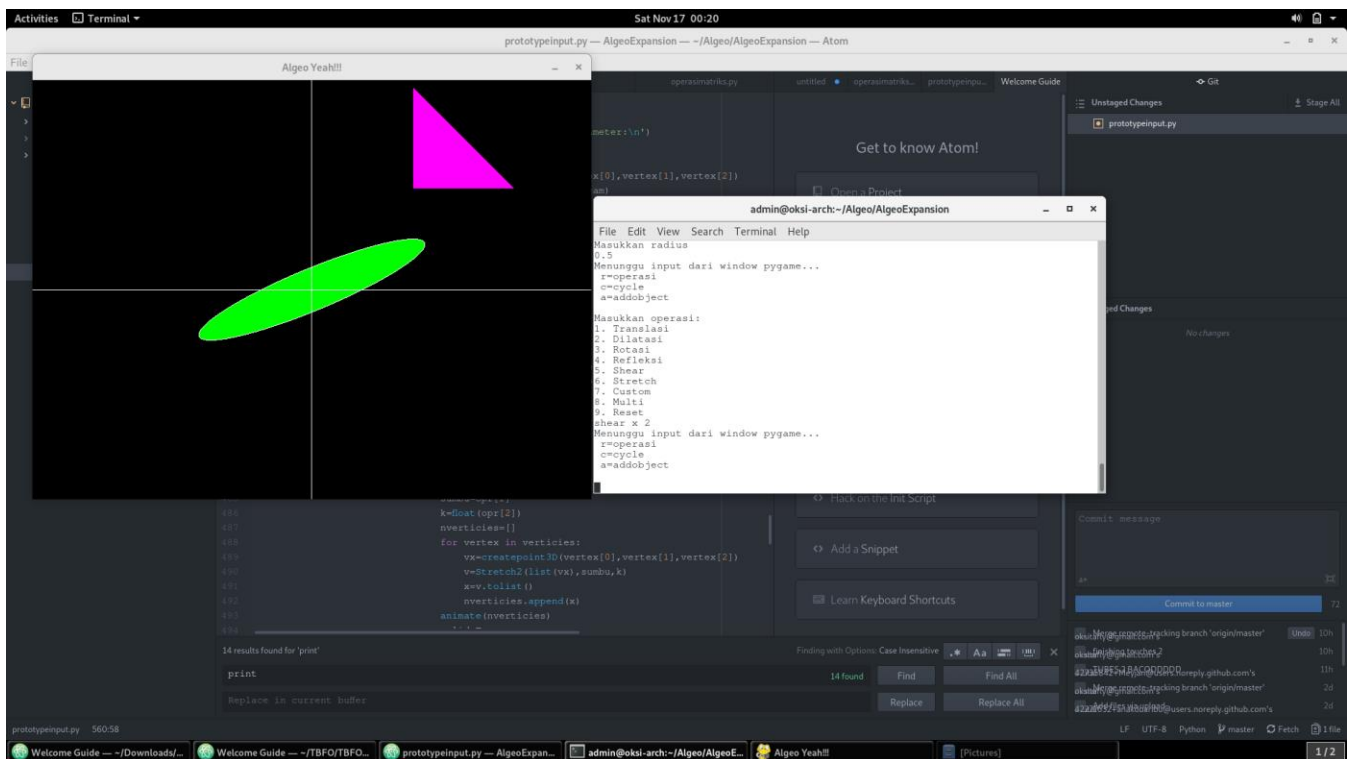
Gambar 4.1.2 Memilih bentuk dan warna



Gambar 4.1.3 Terbentuk segitiga 2D secara *custom*

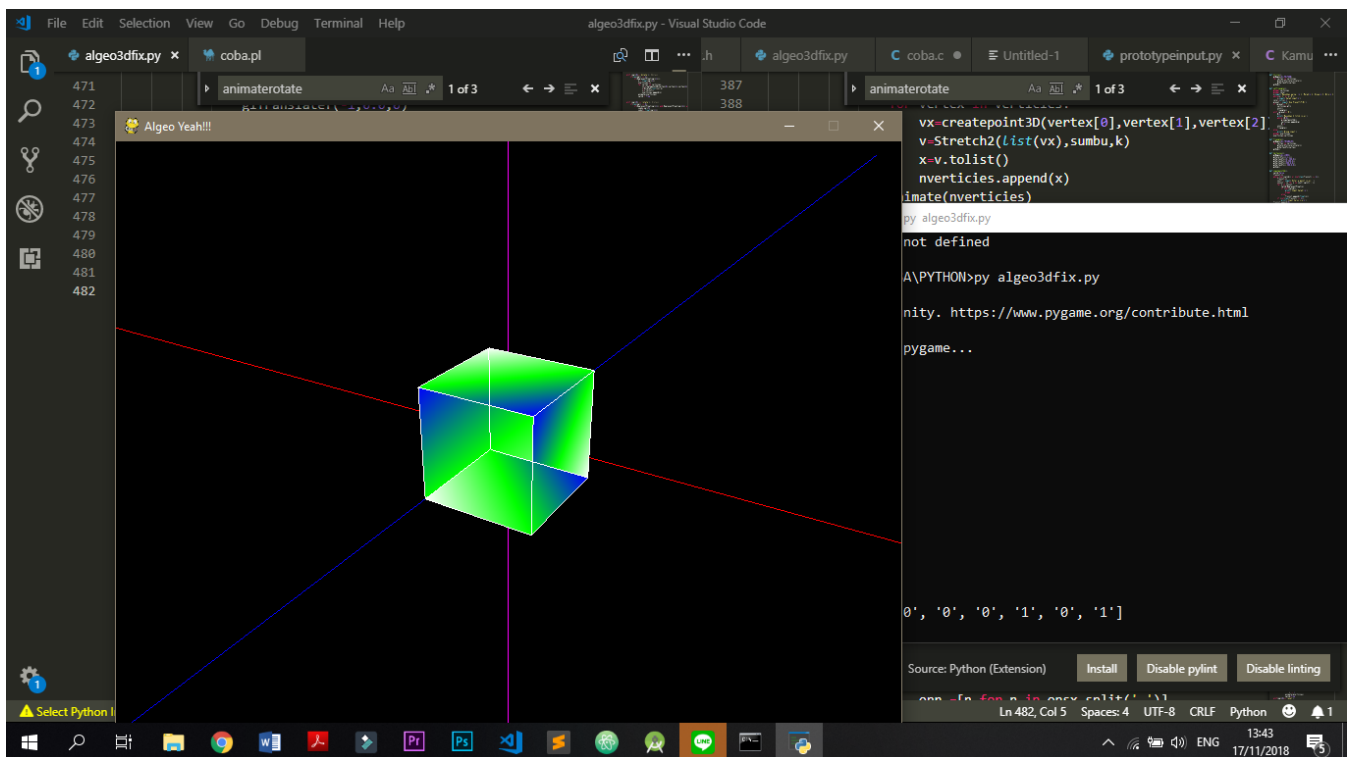


Gambar 4.1.4 Terbentuk lingkaran 2D

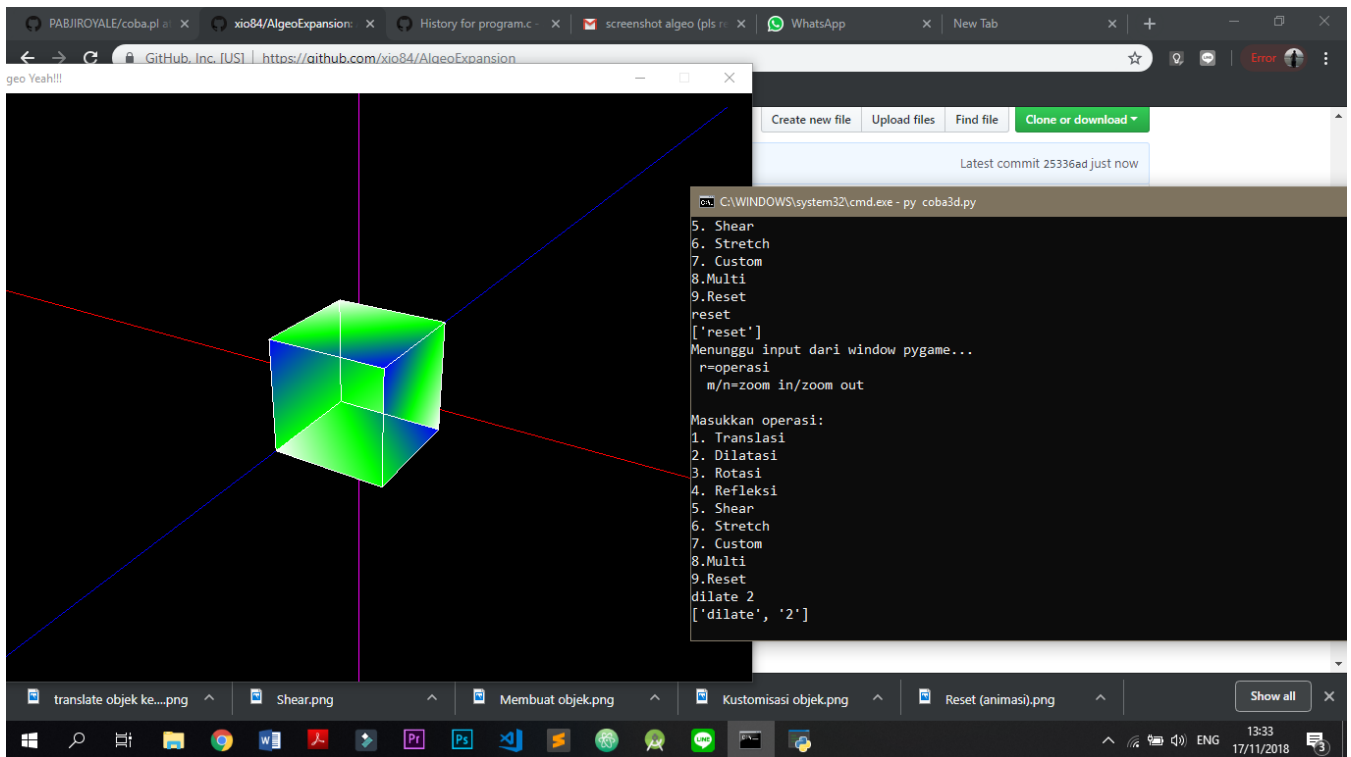


Gambar 4.1.5 Melakukan shear terhadap benda lingkaran

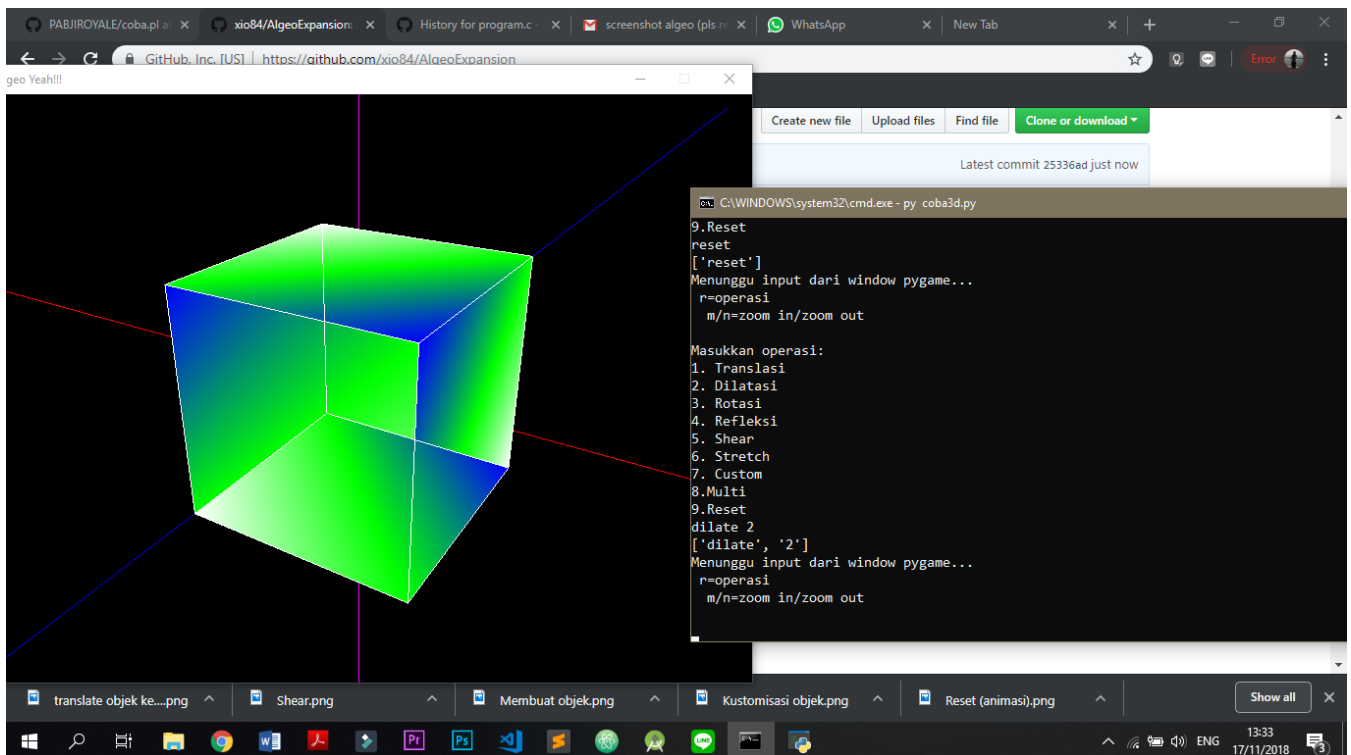
## 4.2 HASIL SCREENSHOT OPERASI 3D



Gambar 4.2.1 Tampilan utama program 3D

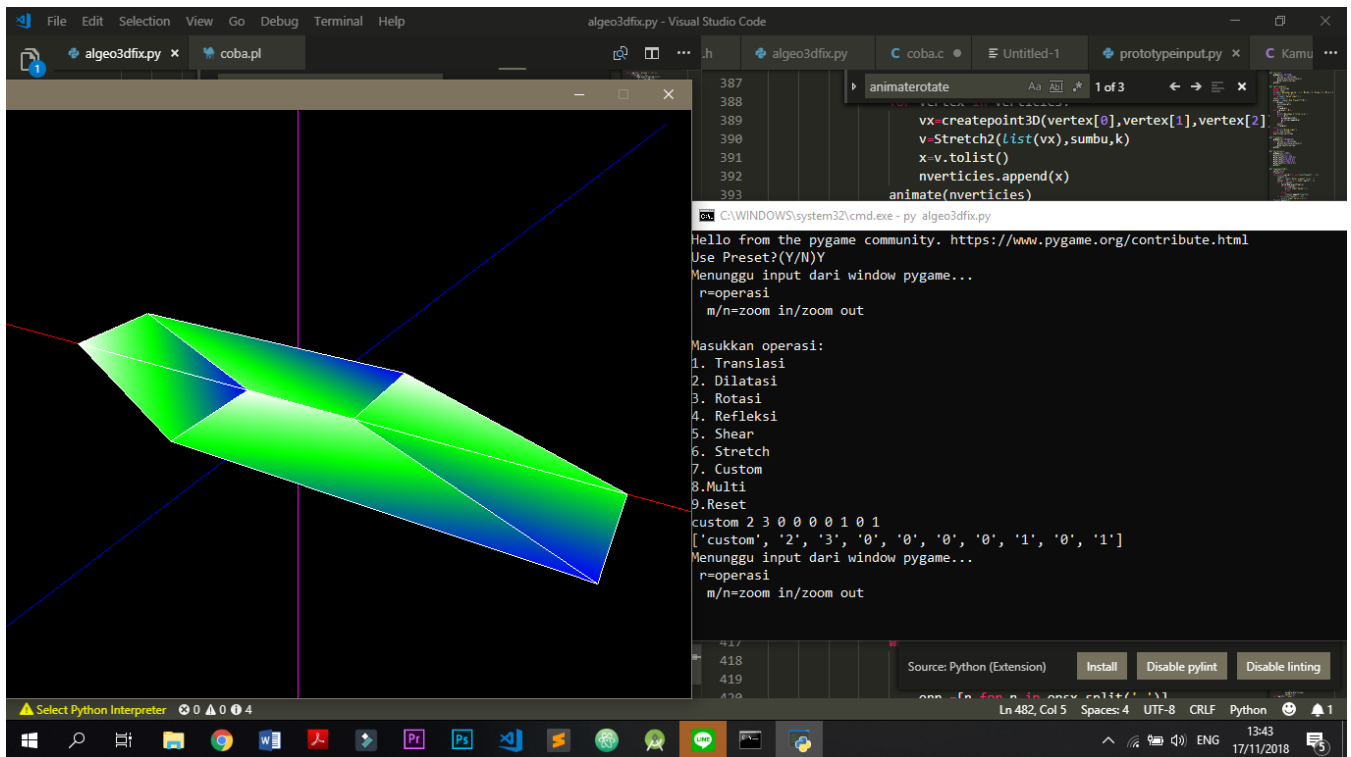


Gambar 4.2.2 Tampilan utama program 3D

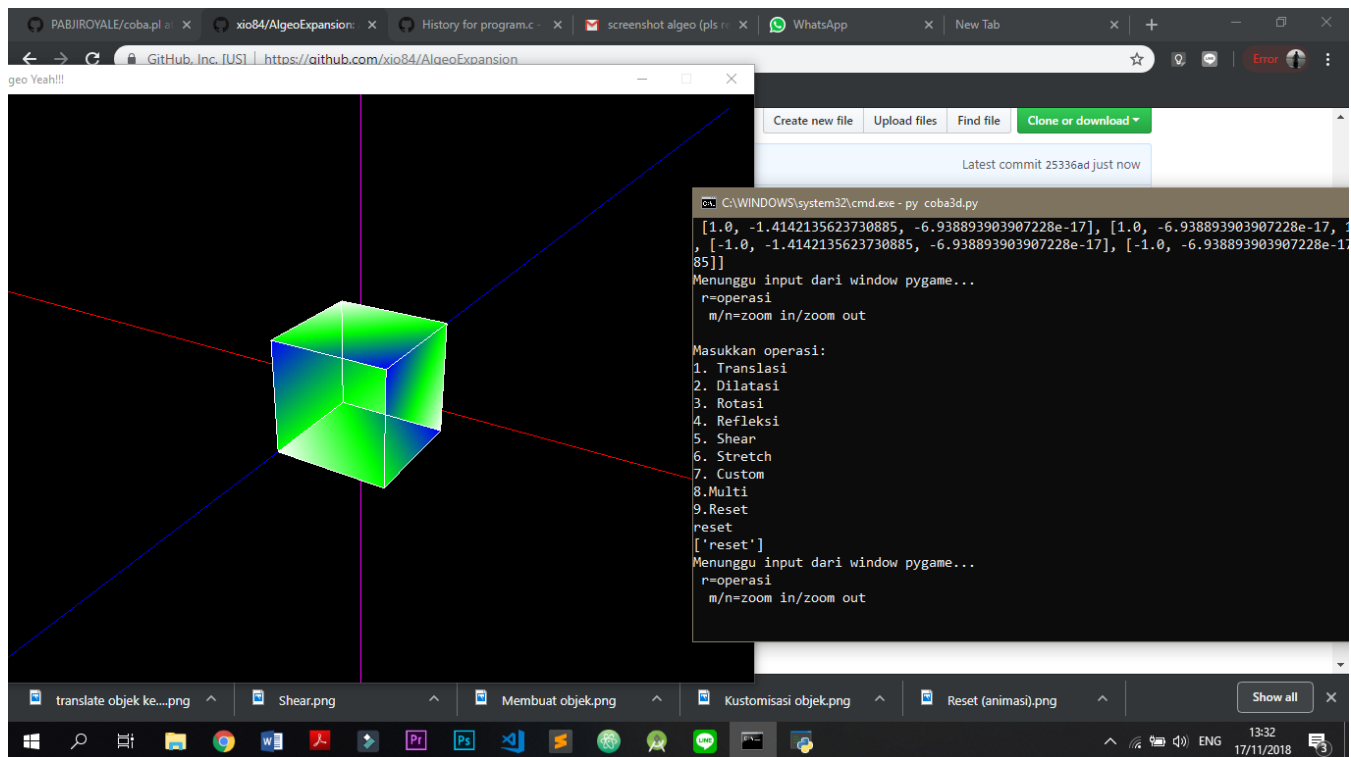


Gambar 4.2.3 Dilasi kubus sebesar 2 kali





Gambar 4.2.4 Melakukan operasi *custom*



Gambar 4.2.5 Melakukan operasi *reset*

## BAB 5

### PENUTUP

Dari Tugas Besar I IF2123 Aljabar Geometri mengenai *Simulasi Transformasi Linier pada Bidang 2D dan 3D dengan Menggunakan OpenGL API*, kami dapat menyimpulkan bahwasanya transformasi linier adalah salah satu kakas yang sangat berguna dan kuat untuk perkembangan ilmu pengetahuan dan juga teknologi. Teknologi informasi dan pengolahan grafik, seperti penyuntingan foto, adalah contoh dari bidang-bidang yang sangat bergantung pada transformasi linier, baik pada bidang 2D maupun 3D.

Melalui program kami, kami dapat melihat bahwa program kami seakan-akan melakukan simulasi terhadap aplikasi-aplikasi penyunting foto, seperti *Adobe Photoshop®*, saat kita mencoba untuk menyunting foto yang kita sematkan dalam aplikasi tersebut. Saat kita ingin *me-resize* sebuah foto, tentu transformasi linier dipakai dengan metode dilasi (*dilate*) untuk membesarkan atau mengecilkan setiap piksel pada foto. Di kesempatan lain, saat kita ingin merotasi foto, aplikasi juga akan melakukan transformasi linier dengan metode rotasi (*rotate*) kepada setiap piksel foto sesuai dengan kemiringan derajat yang kita inginkan. Transformasi-transformasi demikian dapat kami saksikan dari visualisasi yang diberikan oleh *OpenGL* terhadap matriks dari benda 2D maupun 3D.

Di sisi lain, kami dapat menyimpulkan bahwa tugas ini sudah amat baik untuk menggambarkan transformasi linier melalui simulasi yang menggunakan *OpenGL*. Saran dari kami untuk pelaksanaan Tugas Besar II IF2123 Aljabar Geometri adalah pemberian tugas yang bisa diberi lebih awal agar waktu pengumpulan tugas besar ini tidak tumpang tindih dengan tugas lainnya serta adanya komunikasi yang baik antarasisten agar tidak terulang lagi peristiwa ketika adanya dua jawaban berbeda dari asisten yang berbeda ketika mahasiswa menanyakan pertanyaan yang sama.

Selain itu, dari tugas besar ini juga kami mendapatkan banyak hal, seperti penerapan dari transformasi linier ini pada bidang Informatika, khususnya pada pengolahan grafik. Kemudian, kami juga mengerti bahwa dunia Informatika tidak bisa kami ketahui sepenuhnya jikalau kami tidak menyelami sendiri dan menjelajah sendiri ribuan bahkan jutaan informasi yang tersimpan di dalamnya dunia Informatika. Misal, dalam melakukan tugas besar ini, kami diminta untuk mengerjakannya dalam bahasa pemrograman Python - sebuah bahasa yang tentunya baru kami pelajari saat tugas ini. Kami melihat bagaimana sebuah bahasa yang sederhana pengoperasiannya, tetapi sangat powerful dan tentunya kami harus terbiasa dengan bahasa tersebut. Selain itu, kami juga dapat belajar betapa bergunanya bahasa Python dalam berbagai proses pemrograman yang kompleks.

## DAFTAR REFERENSI

1. *Linear Transformation*. <http://mathworld.wolfram.com/LinearTransformation.html> Diakses pada 12 November 2018.
2. *Transformasi Linier (Pemetaan Linier), Slide Kuliah Aljabar Linier Fakultas Informatika Telkom University*.  
[http://cdndata.telkomuniversity.ac.id/pjj/15161/MUG1E3/MZI/COURSE\\_MATERIAL/z1449047645b310e12a05737e6737fc80b4153923c0.pdf](http://cdndata.telkomuniversity.ac.id/pjj/15161/MUG1E3/MZI/COURSE_MATERIAL/z1449047645b310e12a05737e6737fc80b4153923c0.pdf) Diakses pada 12 November 2018.
3. *Ringkasan Materi Matriks Transformasi*. <http://www.aksiomaid.com/Matematika/Ringkasan-Materi/0130010600000000/Transformasi/Matriks-Transformasi> Diakses pada 12 November 2018
- 4.