

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

LAPORAN TUGAS BESAR II IF2123

ALJABAR GEOMETRI



Disusun oleh:

Isa Mujahid Darussalam 13517002

Ahmad Mutawalli 13517026

Adyaksa Wisanggeni 13517091

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

2018

KATA PENGANTAR

Puji syukur saya panjatkan kehadiran Tuhan Yang Maha Esa yang atas segala rahmat dan berkat-Nya, laporan tugas besar 2 ini dapat diselesaikan.

Terima kasih kepada Bapak/ Ibu dosen pengajar dan asisten dosen yang telah membimbing dan memberikan ilmu mengenai transformasi linier sehingga aplikasi dan laporan ini dapat diselesaikan.

Laporan ini dibuat sebagai salah satu syarat pengumpulan tugas besar 2 dari mata kuliah IF 2123 Aljabar Geometri selain pembuatan aplikasi yang berbasis Python.

Aplikasi yang dibuat pada tugas besar 2 ini adalah sebuah program yang dapat melakukan transformasi linier baik pada bidang dua dimensi maupun tiga dimensi menggunakan *OpenGL API*. Untuk bidang dua dimensi masukan ditentukan oleh pengguna, sedangkan untuk tiga dimensi sudah dalam *hardcode* program.

Akhir kata, semoga laporan ini dapat bermanfaat tidak hanya bagi penulis namun juga bagi pembaca dan dapat menambah wawasan mengenai transformasi linier.

DAFTAR ISI

KATA PENGANTAR	1
DAFTAR ISI	2
PENDAHULUAN	3
1.1 Deskripsi Masalah	3
TEORI SINGKAT	4
2.1 Transformasi Linier	4
2.2 Matriks Transformasi	5
2.3 OpenGL	6
IMPLEMENTASI PROGRAM	7
3.1 Garis Besar Program	7
3.2 Transformation.py	7
3.3 main.py	14
3.4 Pembagian tugas	22
EKSPERIMEN	24
KESIMPULAN DAN SARAN	32
5.1 Kesimpulan	32
5.2 Saran	32
5.3 Refleksi	32

BAB I

PENDAHULUAN

1.1 Deskripsi Masalah

Pada tugas kali ini, mahasiswa diminta membuat program yang mensimulasikan transformasi linier untuk melakukan operasi translasi, refleksi, dilatasi, rotasi, 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.

BAB II

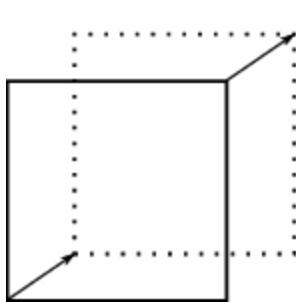
TEORI SINGKAT

2.1 Transformasi Linier

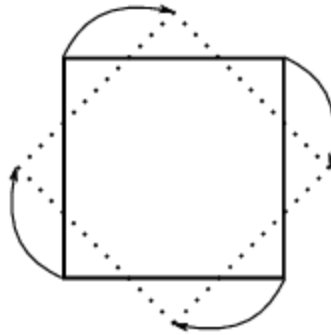
Jika $T : V \rightarrow W$ adalah sebuah fungsi dari ruang vektor V ke ruang vektor W , maka T dinamakan transformasi linier jika:

- (i) $T(u + v) = T(u) + T(v)$ untuk semua vektor u dan v di dalam V
- (ii) $T(ku) = kT(u)$ untuk semua vektor u di dalam V

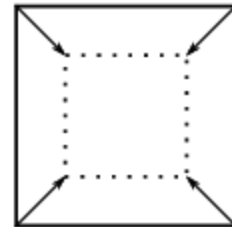
34



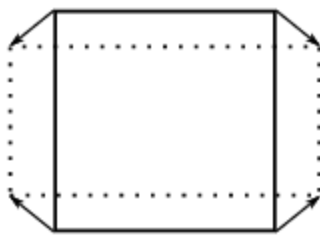
Translation



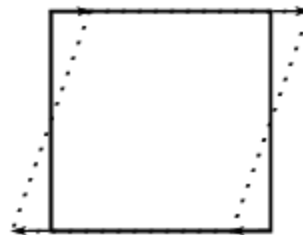
Rotation



Scaling



Stretch



Shearing

https://www.researchgate.net/profile/Antonio_Anjos/publication/223130071/figure/fig1/AS:394015594827782@1470952046671/Elementary-geometric-transforms-for-a-planar-surface-element-used-in-the-affine.png

2.2 Matriks Transformasi

Transformasi linier $T : \mathbb{R}^n \rightarrow \mathbb{R}^m$ dapat dinyatakan sebagai sebuah perkalian matriks
 $T(x) = A(x)$

Adapun untuk matriks transformasi linier 2D adalah sebagai berikut :

Transformasi	Rumus	Matriks
Identitas	$A(x, y) \xrightarrow{1} A'(x, y)$	$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$
Translasi	$A(x, y) \xrightarrow{\begin{pmatrix} p \\ q \end{pmatrix}} A'(x+p, y+q)$	$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} p \\ q \end{pmatrix}$
Refleksi terhadap sumbu-x	$A(x, y) \xrightarrow{sb.x} A'(x, -y)$	$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$
Refleksi terhadap sumbu-y	$A(x, y) \xrightarrow{sb.y} A'(-x, y)$	$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$
Refleksi terhadap garis $y=x$	$A(x, y) \xrightarrow{y=x} A'(y, x)$	$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$
Refleksi terhadap garis $y=-x$	$A(x, y) \xrightarrow{y=-x} A'(y, -x)$	$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$
Refleksi terhadap garis $x=k$	$A(x, y) \xrightarrow{x=k} A'(2k-x, y)$	
Refleksi terhadap garis $y=k$	$A(x, y) \xrightarrow{y=k} A'(x, 2k-y)$	
Refleksi terhadap titik (p, q)	$A(x, y) \xrightarrow{(p, q)} A'(x', y')$ Sama dengan rotasi pusat (p, q) sejauh 180°	$\begin{pmatrix} x'-p \\ y'-q \end{pmatrix} = \begin{pmatrix} \cos 180^\circ & -\sin 180^\circ \\ \sin 180^\circ & \cos 180^\circ \end{pmatrix} \begin{pmatrix} x-p \\ y-q \end{pmatrix}$

<http://1.bp.blogspot.com/-w7JQWmnPGfM/UqqsoX1jLwI/AAAAAAAAAHG8/sjISj2GMOCS/s1600/ssss.jpg>

Sedangkan untuk matriks transformasi linier 3D adalah sebagai berikut:

$$\begin{array}{ccc}
 \text{X-Rotation in 3D} & \text{Z-Rotation in 3D} & \text{Scale in 3D} \\
 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 \text{Y-Rotation in 3D} & \text{Translation in 3D} & \text{Matrix Multiplication} \\
 \begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & Tx \\ 0 & 1 & 0 & Ty \\ 0 & 0 & 1 & Tz \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ q \end{bmatrix}
 \end{array}
 \quad (4 \times 4) * (4 \times 1) = (4 \times 1)$$

<https://ilmuenterprise.files.wordpress.com/2014/12/capture48.jpg>

2.3 OpenGL

OpenGL (Open Graphics Library) adalah spesifikasi standar yang mendefinisikan sebuah 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 menggambarkan 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.

BAB III

IMPLEMENTASI PROGRAM

3.1 Garis Besar Program

Program ini menggunakan 2 file yang berbeda, yaitu main.py dan transformation.py. main.py berfungsi sebagai tempat interaksi dengan pengguna beserta menampilkan GUI kepada pengguna. Sementara transformation.py sebagai implementasi dari struktur data matriks beserta transformasi-transformasi yang dapat dilakukan pada matriks tersebut.

Cara kerja dari program ini adalah pengguna akan diminta untuk memasukkan masukkan sesuai dari menu yang ada. Input pertama pengguna adalah memilih dimensi dari transformasi yang diinginkan. Jika memilih 2 dimensi, maka pengguna dapat memasukkan titik-titik pada bidang yang ingin dibuat. Program akan secara otomatis membangun sebuah poligon dari titik yang telah diberikan. Program juga akan membuat sebuah matriks representasi dari titik-titik pada bangun tersebut

Setelah langkah tersebut, akan muncul layar baru yang berisi bangun yang ingin ditransformasikan. Pengguna dapat mengetikkan perintah-perintah untuk mentransformasikan bangun tersebut. Terdapat 9 perintah yang tersedia, yaitu translate, dilate, rotate, reflect, shear, stretch, custom, reset, dan exit.

Saat pengguna memasukkan perintah, program akan mengecek untuk inputan tersebut, matriks transformasi apa yang cocok digunakan. Setelah itu, ia akan mengalikan matriks representasi dari titik sekarang dengan matriks transformasi tersebut untuk membuat matriks baru. Semua proses ini disimpan ke dalam sebuah variabel sementara, dimana bangun awal akan berubah secara perlahan menuju bangun yang direpresentasikan oleh variabel sementara tersebut.

3.2 Transformation.py

```
class Matriks:

    def __init__(self, m = [[0]]):
        # Inisialisasi semua elemen matriks diisi 0
        self.M = np.array(m)
        self.brs = np.size(m,0)
        self.kol = np.size(m,1)

    def AddRow(self, m):
        self.M = np.append(self.M, m, axis = 0)
```



```
        self.brs += 1

def DelRow(self,idx):
    mat = Matriks(np.delete(self.M,idx,axis = 0))
    self.brs -= 1
    return mat

def AddColumn(self,m):
    self.M = np.append(self.M,m,axis = 1)
    self.kol += 1

def GetNBrsEff(self):
    return self.brs

def GetNKolEff(self):
    return self.kol

def GetElmt(self,brs,kol):
    return self.M[brs,kol]

def transpose(self):
    m, n = self.kol, self.brs
    mat = Matriks([[self.GetElmt(j,i) for j in range (n)] for i in
range (m)])
    return mat

def __str__(self):
    s = ""
    for i in range(self.brs):
        for j in range(self.kol):
            s += "%f" % self.GetElmt(i, j)
            if (j<self.kol-1):
                s += " "
            else:
                s += "\n"
    return s

def __add__(self, mat):
    # Penjumlahan matriks
    mhs1 = Matriks([[0]*self.GetNKolEff() for x in
range(self.GetNBrsEff())])
    mhs1.M = self.M + mat.M
```

```
        return mhs1

    def __sub__(self, mat):
        # Pengurangann matriks
        mhs1 = Matriks([[0]*self.GetNKolEff() for x in
range(self.GetNBrsEff())])
        mhs1.M = self.M - mat.M
        return mhs1

    def __mul__(self, m1):
        # Mengali matriks ini dengan matriks lain tanpa mengubah value
keduanya
        # Input selalu benar : neff kolom Matriks ini = neff baris
matriks lainnya
        mat = Matriks(np.matmul(self.M, m1.M))
        return mat
```

Class Matriks berfungsi untuk memberikan fungsionalitas pada tipe data Matriks yang akan digunakan untuk merepresentasikan titik dan matriks transformasi

```
class Object():

    initVertices = Matriks()
    vertices = Matriks()
    edges = Matriks()

    # Object
    def __init__(self, vertices, edges):
        self.vertices = vertices
        self.initVertices = vertices
        self.edges = edges

    def reset(self):
        self.vertices = self.initVertices
```

Class Object berfungsi sebagai parent dari object2D dan object3D dimana class ini memberikan inisialisasi kepada kedua class tersebut

```
class Object2D(Object):
    def translate(self, dx, dy):
        #Melakukan translasi tiap point sejauh dx,dy
```

```
        mhs1 = Matriks([[self.vertices.GetElmt(i,j) for j in
range(self.vertices.GetNKolEff())] for i in
range(self.vertices.GetNBrEff())])
        mhs1.AddRow([[1]*mhs1.GetNKolEff()])
        mtrans = Matriks([[1,0,dx],[0,1,dy],[0,0,1]])
        mhs1 = mtrans*mhs1
        self.vertices = mhs1.DelRow(2)

def dilate(self, k):
    #Melakukan dilatasi tiap point sebesar k
    mtrans = Matriks([[k,0],[0,k]])
    self.vertices = mtrans*self.vertices

def rotate(self, deg, a, b):
    #Melakukan rotasi tiap titik dengan deg derajat dengan titik
pusat (a,b)
    angle = radians(deg)
    mtrans =
Matriks([[cos(angle), -1*sin(angle)], [sin(angle), cos(angle)]])
    mpusat =
Matriks([[a/10]*self.vertices.GetNKolEff(), [b/10]*self.vertices.GetNKolEff(
)])
    self.vertices = mtrans*(self.vertices - mpusat) + mpusat

def reflect(self, param):
    #Melakukan refleksi dengan parameter
    if (param == "y"):
        mtrans = Matriks([[-1,0],[0,1]])
        self.vertices = mtrans*self.vertices
    elif (param == "x"):
        mtrans = Matriks([[1,0],[0,-1]])
        self.vertices = mtrans*self.vertices
    elif (param == "y=x"):
        mtrans = Matriks([[0,1],[1,0]])
        self.vertices = mtrans*self.vertices
    elif (param == "y=-x"):
        mtrans = Matriks([[0,-1],[-1,0]])
        self.vertices = mtrans*self.vertices
    else : # pencerminan (a,b) = rotasi 180 derajat terhadap titik
a,b
        p = param.split(',')
        p[0] = p[0].replace(' ','') # menghapus spasi apabila ada
```

```
        p[0] = p[0].replace('(', "") # menghapus kurung apabila
ada
        p[1] = p[1].replace(' ', "") # menghapus spasi apabila ada
        p[1] = p[1].replace(')', "") # menghapus kurung apabila
ada

        a = float(p[0])
        b = float(p[1])
        mtrans = Matriks([[ -1, 0], [0, -1]])
        mpusat =
Matriks([[a/10]*self.vertices.GetNKolEff(), [b/10]*self.vertices.GetNKolEff(
)])

        self.vertices = mtrans*(self.vertices-mpusat)+mpusat

def shear(self, param, k):
    #Melakukan operasi shear pada objek
    if (param == "x"):
        mtrans = Matriks([[1, k], [0, 1]])
    elif (param == "y"): #y
        mtrans = Matriks([[1, 0], [k, 1]])
    self.vertices = mtrans*self.vertices

def stretch(self, param, k):
    #Melakukan operasi stretch pada objek
    if (param == "x"):
        mtrans = Matriks([[1, 0], [0, k]])
    elif (param == "y"): #y
        mtrans = Matriks([[k, 0], [0, 1]])
    self.vertices = mtrans*self.vertices

def custom(self, a, b, c, d):
    #Melakukan transformasi dengan matriks [[a,b][c,d]]
    mtrans = Matriks([[a, b], [c, d]])
    self.vertices = mtrans * self.vertices

def multiple(self, n):
    #Melakukan n buah transformasi
    return

def output(self):
    glColor3f(random.random(), random.random(), random.random())
    glBegin(GL_POLYGON)
    for edge in self.edges:
```

```
        for vertex in edge:
            glVertex2fv(self.vertices.transpose().M[vertex])
    glEnd()
```

Class Object2D berisi fungsionalitas-fungsionalitas seperti transformasi objek 2 dimensi dan cara menampilkan objek tersebut pada OpenGL

```
class Object3D(Object):
    def translate(self, dx, dy, dz):
        mhs1 = Matriks([[self.vertices.GetElmt(i,j) for j in
range(self.vertices.GetNKolEff())] for i in
range(self.vertices.GetNBrEff())])
        mhs1.AddRow([[1]*mhs1.GetNKolEff()])
        mtrans = Matriks([[1,0,0,dx],[0,1,0,dy],[0,0,1,dz],[0,0,0,1]])
        mhs1 = mtrans*mhs1
        self.vertices = mhs1.DelRow(3)

    def dilate(self, k):
        mtrans = Matriks([[k,0,0],[0,k,0],[0,0,k]])
        self.vertices = mtrans*self.vertices

    def rotate(self, deg, a, b, c):
        # rotasi terhadap sumbu x apabila a = 1
        angle = radians(deg)
        if (a == 1):
            mtrans =
Matriks([[1,0,0],[0,cos(angle),-1*sin(angle)],[0,sin(angle),cos(angle)]])
        elif (b == 1):
            mtrans =
Matriks([[cos(angle),0,sin(angle)],[0,1,0],[-1*sin(angle),0,cos(angle)]])
        elif (c == 1):
            mtrans = Matriks([[cos(angle),
-1*sin(angle),0],[sin(angle), cos(angle),0],[0,0,1]])
        self.vertices = mtrans*self.vertices

    def reflect(self, param):
        if (param == "xy"):
            mtrans = Matriks([[1,0,0],[0,1,0],[0,0,-1]])
        elif (param == "xz"):
            mtrans = Matriks([[1,0,0],[0,-1,0],[0,0,1]])
```

```
elif (param == "yz"):
    mtrans = Matriks([[-1,0,0],[0,1,0],[0,0,1]])
elif (param == "o"):
    mtrans = Matriks([[-1,0,0],[0,-1,0],[0,0,-1]])
self.vertices = mtrans*self.vertices

def shear(self, param, k1 =0 ,k2=0):
    # Bentuk matriks transformasinya : [[1 sh(yx) sh
    (zx)], [sh(xy),1,sh(zy)], [sh(xz),sh(yz),1]]
    if (param == "x"):
        mtrans = Matriks([[1,0,0],[k1,1,0],[k2,0,1]])
    elif (param == "y"):
        mtrans = Matriks([[1,k1,0],[0,1,0],[0,k2,1]])
    elif (param == "z"):
        mtrans = Matriks([[1,0,k1],[0,1,k2],[0,0,1]])
    self.vertices = mtrans * self.vertices

def stretch(self, param, k1=1,k2=1):
    if (param == "x"):
        mtrans = Matriks([[1,0,0],[0,k1,0],[0,0,k2]])
    elif (param == "y"):
        mtrans = Matriks([[k1,0,0],[0,1,0],[0,0,k2]])
    elif (param == "z"):
        mtrans = Matriks([[k1,0,0],[0,k2,0],[0,0,1]])
    self.vertices = mtrans * self.vertices

def custom(self, a, b, c, d, e, f, g, h, i):
    mtrans = Matriks([[a,b,c],[d,e,f],[g,h,i]])
    self.vertices = mtrans * self.vertices

def multiple(self, n):
    #for x in range(n)
    #...
    return

def output(self):
    glColor3f(1.0,1.0,1.0)
    glBegin(GL_POLYGON)
    for edge in self.edges:
        for vertex in edge:
```

```
glVertex3fv(self.vertices.transpose().M[vertex])  
glEnd()
```

Class Object2D berisi fungsionalitas-fungsionalitas seperti transformasi objek 3 dimensi dan cara menampilkan objek tersebut pada OpenGL

3.3 main.py

```
def displayObject():  
    #Menampilkan object saat ini  
    shape.output()  
  
def displayCartesian():  
    #Menampilkan sumbu x, y, dan z  
    #X Axis  
    glPushAttrib(GL_ENABLE_BIT)  
    glLineStipple(1, 0xAAAA)  
    glEnable(GL_LINE_STIPPLE)  
    glColor3f(1.0, 0, 0)  
    glBegin(GL_LINES)  
    glVertex3fv((-500,0,0))  
    glVertex3fv((0,0,0))  
    glEnd()  
    glPopAttrib()  
  
    glBegin(GL_LINES)  
    glVertex3fv((0,0,0))  
    glVertex3fv((500,0,0))  
    glEnd()  
  
    #Y Axis  
    glPushAttrib(GL_ENABLE_BIT)  
    glLineStipple(1, 0xAAAA)  
    glEnable(GL_LINE_STIPPLE)  
    glColor3f(0, 1.0, 0)  
    glBegin(GL_LINES)  
    glVertex3fv((0,-500,0))  
    glVertex3fv((0,0,0))  
    glEnd()  
    glPopAttrib()
```

```
glBegin(GL_LINES)
glVertex3fv((0,0,0))
glVertex3fv((0,500,0))
glEnd()

#Z Axis
glPushAttrib(GL_ENABLE_BIT)
glLineStipple(1, 0xAAAA)
glEnable(GL_LINE_STIPPLE)
glColor3f(0, 0, 1.0)
glBegin(GL_LINES)
glVertex3fv((0,0,-500))
glVertex3fv((0,0,0))
glEnd()
glPopAttrib()

glBegin(GL_LINES)
glVertex3fv((0,0,0))
glVertex3fv((0,0,500))
glEnd()
```

Fungsi-fungsi tersebut berfungsi untuk menampilkan objek dan sumbu kartesius pada OpenGL

```
def change():
    global is3D
    #Mengubah nilai titik sesuai yang diinginkan secara minimal
    if(q):
        q[0][-1] -= 1
        if(q[0][0] == "rotate"):
            if is3D:
                shape.rotate(q[0][1],q[0][2],q[0][3],q[0][4])
            else:
                shape.rotate(q[0][1],q[0][2],q[0][3])
        else:
            shape.vertices += q[0][0]
    if(q[0][-1] == 0):
        q.pop(0)
```


Fungsi change berfungsi untuk mengubah bangun pada OpenGL sedikit demi sedikit setiap kali OpenGL memanggil fungsi display

```
def input2D():
    #Meminta input untuk objek 2D
    global shape
    N = int(input("Masukkan nilai N\n"))
    if(N > 0):
        print("Masukkan " + str(N) + " buah titik 2 dimensi")
        valid = False
        while(not valid):
            try:
                x, y = map(float, input().split())
                valid = True
            except:
                print("Harap memasukkan 2 buah angka dipisahkan
oleh spasi")
        vertices = Matriks([[x/10],[y/10]])
        edges = []
        for i in range(N-1):
            valid = False
            while(not valid):
                try:
                    x, y = map(float, input().split())
                    valid = True
                except:
                    print("Harap memasukkan 2 buah angka
dipisahkan oleh spasi")
            vertices.AddColumn([[x/10],[y/10]])
            edges.insert(len(edges),[i,(i+1)%N])
        shape = Object2D(vertices, edges)
    else:
        shape = Object2D(Matriks([[0],[0]]),[])

def input3D():
    #Meminta input untuk objek 3D
    global shape, vertices3d, edges3d
    shape = Object3D(vertices3d, edges3d)

def inputDimensionChoice():
    #Meminta input dimensi yang diinginkan
    global is2D,is3D
```

```
x = int(input("Keluarkan 2 jika ingin 2 dimensi, dan 3 jika ingin 3D\n"))
while(x != 2 and x != 3):
    print("Masukkan salah")
    x = int(input("Keluarkan 2 jika ingin 2 dimensi, dan 3 jika ingin 3D\n"))
if(x == 2):
    is2D = True
else:
    is3D = True
```

Fungsi-fungsi tersebut berfungsi untuk membuat bangun berdasarkan inputan pengguna maupun default dari program

```
def processCommand(command):
    #Menjalankan command yang telah diberikan
    global q,is3D,isEnd
    if(q):
        print("Sedang terjadi transformasi, silakan tunggu transformasi selesai")
        return
    parsedCommand = command.split(' ')
    func = parsedCommand[0].lower()
    iteration = 500
    try:
        if is3D:
            temp = Object3D(shape.vertices,shape.edges)
        else:
            temp = Object2D(shape.vertices,shape.edges)
        temp.initVertices = shape.initVertices
        if(func == "translate"):
            dx = float(parsedCommand[1])/10
            dy = float(parsedCommand[2])/10
            if is3D:
                dz = float(parsedCommand[3])/10
                temp.translate(dx,dy,dz)
            else:
                temp.translate(dx,dy)
        elif(func == "dilate"):
            k = float(parsedCommand[1])
            temp.dilate(k)
```

```
elif(func == "rotate"):
    degree = float(parsedCommand[1])
    a = float(parsedCommand[2])
    b = float(parsedCommand[3])
    if is3D:
        c = float(parsedCommand[4])
        q.append(["rotate",
degree/iteration,a,b,c,iteration])
    else:
        q.append(["rotate",
degree/iteration,a,b,iteration])
elif(func == "reflect"):
    param = parsedCommand[1]
    temp.reflect(param)
elif(func == "shear"):
    param = parsedCommand[1]
    k = float(parsedCommand[2])
    if is3D:
        k2 = float(parsedCommand[3])
        temp.shear(param,k,k2)
    else:
        temp.shear(param,k)
elif(func == "stretch"):
    param = parsedCommand[1]
    k = float(parsedCommand[2])
    if is3D:
        k2 = float(parsedCommand[3])
        temp.stretch(param,k,k2)
    else:
        temp.stretch(param,k)
elif(func == "custom"):
    a = float(parsedCommand[1])
    b = float(parsedCommand[2])
    c = float(parsedCommand[3])
    d = float(parsedCommand[4])
    if is3D:
        e = float(parsedCommand[5])
        f = float(parsedCommand[6])
        g = float(parsedCommand[7])
        h = float(parsedCommand[8])
        i = float(parsedCommand[9])
        temp.custom(a,b,c,d,e,f,g,h,i)
```

```
        else:
            temp.custom(a,b,c,d)
    elif(func == "help"):
        commandList()
        return
    elif(func == "reset"):
        temp.reset()
    elif(func == "exit"):
        glutLeaveMainLoop()
        isEnd = True
        return
    else:
        print("Command tidak valid, silakan ulangi")
        return

    if(func != "rotate"):
        temp.vertices -= shape.vertices
        temp.vertices.M /= iteration
        q.append([temp.vertices,iteration])
except:
    print("Terdapat parameter yang salah, silakan ulangi")
```

Fungsi processCommand berfungsi untuk memproses perintah yang diberikan pengguna. Jika perintah tersebut valid, maka transformasi tersebut akan disimpan di variabel q dimana akan diproses pada variable change setiap menampilkan display

```
def keyPressed(key, x, y):
    #Menampilkan output pada terminal saat OpenGL telah dijalankan
    global currentCommand
    if(ord(key) == 13): #Newline
        print(end='\n',flush=True)
        processCommand(currentCommand)
        currentCommand = ""
        if(not isEnd):
            transformationInput()
        return
    elif(ord(key) == 8): #Backspace
        key = '\b \b'
        print('\b \b', end='', flush=True)
        currentCommand = currentCommand[:-1]
    else:
```

```
        currentCommand += key.decode('utf-8')
        print(key.decode('utf-8'), end='', flush=True)

def specialKey(key, x, y):
    #Special key untuk menggerakkan kamera
    global eyeX, eyeY, eyeZ, moveX, moveY, moveZ
    if(key == GLUT_KEY_UP):
        eyeY += moveY
    elif(key == GLUT_KEY_DOWN):
        eyeY -= moveY
    elif(key == GLUT_KEY_LEFT):
        eyeX -= moveX
    elif(key == GLUT_KEY_RIGHT):
        eyeX += moveX
    elif(key == GLUT_KEY_F1):
        glRotate(5.0, 1.0, 0.0, 1.0)
    elif(key == GLUT_KEY_F2):
        glRotate(-5.0, 1.0, 0.0, 1.0)
```

Fungsi-fungsi tersebut berfungsi untuk menerima masukkan input dari keyboard saat OpenGL sedang berjalan. Fungsi KeyPressed berfungsi sebagai penerima alfabet normal untuk menerima perintah transformasi yang akan dijalankan. Sementara fungsi specialKey berfungsi untuk mengubah pandangan kamera pada OpenGL

```
def display():
    #Menampilkan objek dan sumbu kartesius
    global eyeX, eyeY, eyeZ, is2D
    change()
    glClearColor(0.0, 0.0, 0.0, 0.0)
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glMatrixMode(GL_MODELVIEW)

    displayCartesian()
    gluLookAt(eyeX, eyeY, eyeZ, eyeX, eyeY, -1.0, 0.0, 1.0, 0.0);
    displayObject()
    gluLookAt(0, 0, 0, 0, 0, -1.0, 0.0, 1.0, 0.0);
    gluLookAt(eyeX, eyeY, eyeZ, eyeX, eyeY, -1.0, 0.0, 1.0, 0.0);
    displayObject()
    gluLookAt(0, 0, 0, 0, 0, -1.0, 0.0, 1.0, 0.0);

    eyeX = 0
```

```
eyeY = 0
eyeZ = 0
glutSwapBuffers()
glutPostRedisplay()
return
```

Fungsi display berfungsi untuk mengupdate GUI OpenGL berdasarkan keadaan program saat ini.

```
def transformationInput():
    #Menampilkan tampilan saat menerima input
    print("Masukkan transformasi yang diinginkan")
    print("Ketik help untuk melihat command yang ada")
    print(">>> ", end='', flush=True)

def commandList():
    #Menampilkan command yang dapat dilakukan
    if(is2D):
        print("translate <dx> <dy>: Melakukan translasi objek dengan
menggeser nilai x sebesar dx dan menggeser nilai y sebesar dy.")
        print("dilate <k>: Melakukan dilatasi objek dengan faktor
scaling k.")
        print("rotate <deg> <a> <b>: Melakukan rotasi objek secara
berlawanan arah jarum jam sebesar deg derajat terhadap titik a,b")
        print("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.")
        print("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.")
        print("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.")
        print("Melakukan transformasi linier pada objek dengan matriks
transformasi [[a,b],[c,d]]")
        print("reset: Mengembalikan objek pada kondisi awal objek
didefinisikan.")
        print("exit: Keluar dari program.")
    else:
        print("translate <dx> <dy> <dz>: Melakukan translasi objek
dengan menggeser nilai x sebesar dx, menggeser nilai y sebesar dy, dan
```

```
menggeser nilai z sebesar dz.")
    print("dilate <k>: Melakukan dilatasi objek dengan faktor
scaling k.")
    print("rotate <deg> <a> <b> <c>: Melakukan rotasi objek
terhadap sumbu x y atau z dengan parameter a = 1 jika terhadap sumbu x, b =
1 jika terhadap sumbu y, dan c = 1 jika terhadap sumbu z")
    print("reflect <param>: Melakukan pencerminan objek. Nilai
param adalah salah satu dari nilai-nilai berikut: xy, xz, yz, atau titik
origin (0,0,0).")
    print("shear <param> <k1> <k2>: Melakukan operasi shear pada
objek. Nilai param dapat berupa x (terhadap sumbu x), y (terhadap sumbu y),
atau z (terhadap sumbu z). Nilai k1 dan k2 adalah faktor shear terhadap
sumbu bukan param.")
    print("stretch <param> <k1> <k2>: Melakukan operasi stretch
pada objek. Nilai param dapat berupa x (terhadap sumbu x), y (terhadap
sumbu y), atau z (terhadap sumbu z). Nilai k1 dan k2 adalah faktor stretch
terhadap sumbu bukan param.")
    print("Melakukan transformasi linier pada objek dengan matriks
transformasi [[a,b,c],[d,e,f],[g,h,i]]")
    print("reset: Mengembalikan objek pada kondisi awal objek
didefinisikan.")
    print("exit: Keluar dari program.")
```

Fungsi `commandList` berfungsi untuk menampilkan list command yang diterima program, sementara `transformationInput` untuk menjelaskan kepada pengguna untuk memberikan suatu input

```
def openGLDisplay():
    #Menampilkan tampilan openGL
    glutInit(sys.argv)
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH)
    glutInitWindowSize(600,600)
    glutInitWindowPosition(0,0)
    glutCreateWindow(window_name)
    glutDisplayFunc(display)
    glutKeyboardFunc(keyPressed)
    transformationInput()
    glutSpecialFunc(specialKey)
    glutMainLoop()
    return
```

Fungsi `openGLDisplay` berfungsi untuk menginisialisasi dan memulai loop pada OpenGL

3.4 Pembagian tugas

Pada tugas besar ini, pembagian tugasnya adalah sebagai berikut:

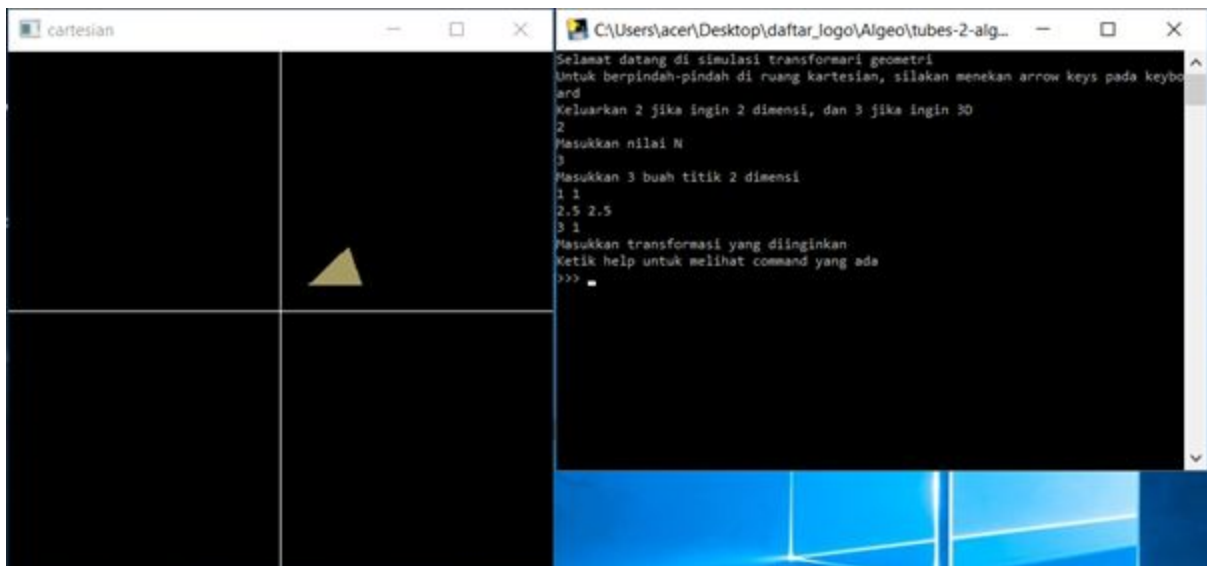
- Isa Mujahid Darussalam: Transformasi Linier
- Ahmad Mutawalli: Menampilkan objek secara 3 dimensi
- Adyaksa Wisanggeni: Menampilkan objek secara 2 dimensi beserta program utama

BAB IV

EKSPERIMEN

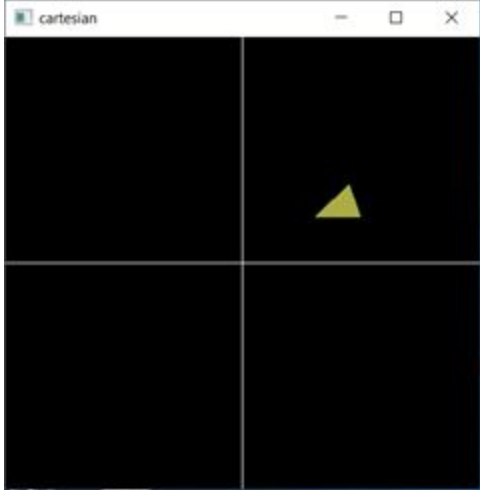
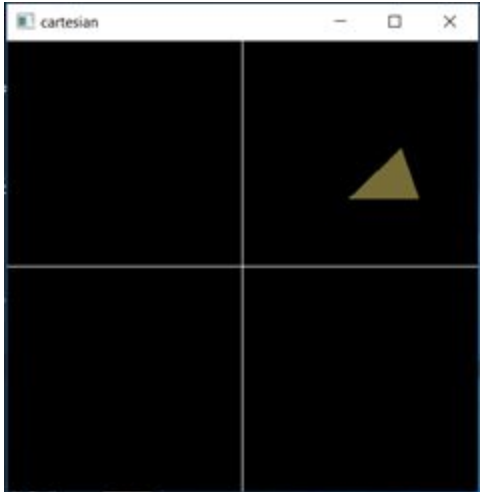
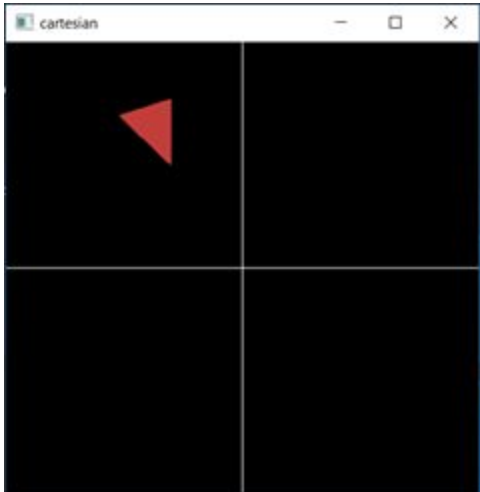
Hasil eksekusi 1 :

Pada eksperimen pertama, pengguna memberikan masukan tiga buah titik dalam bidang dua dimensi yaitu titik (1,1), (2.5,2.5), dan (3,1). Program menampilkan segitiga di layar sesuai titik yang diinput. Akan dilakukan eksperimen translasi, dilatasi, rotasi, dan refleksi terhadap segitiga tersebut.

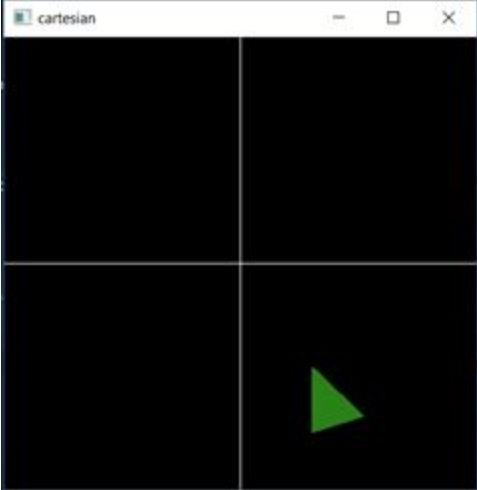


Input	Output	Analisis
-------	--------	----------

Tugas Besar II IF2123 Aljabar Geometri
Simulasi Transformasi Linier

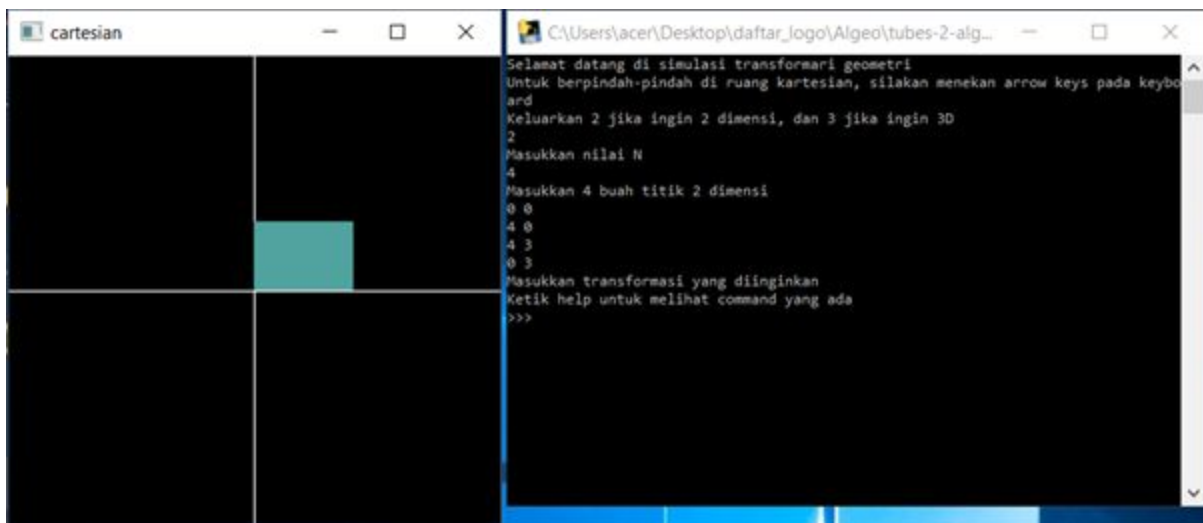
<code>translate 2 1</code>		Melakukan operasi translasi terhadap segitiga. Segitiga digeser sejauh 2 terhadap x dan sejauh 1 terhadap y.
<code>dilate 1.5</code>		Setelah ditranslasi, dilakukan dilatasi sebesar faktor skala 1.5. Dilatasi dilakukan dengan mengalikan matriks transformasi dengan matriks yang dibentuk oleh segitiga.
<code>rotate 90 0 0</code>		Melakukan operasi rotasi terhadap segitiga dengan sudut 90 derajat terhadap pusat (0,0).

Tugas Besar II IF2123 Aljabar Geometri
Simulasi Transformasi Linier

<p>reflect (0,0)</p>		<p>Melakukan operasi refleksi atau pencerminan terhadap segitiga dengan titik pusat yaitu (0,0).</p>
----------------------	-----------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------



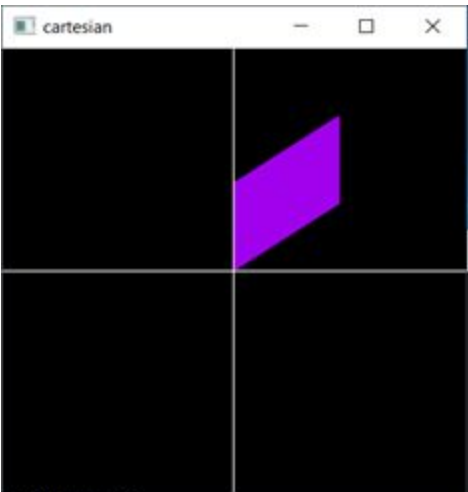
Hasil Eksekusi 2 :

Pada eksperimen yang kedua, pengguna memberi masukan empat buah input yaitu (0,0), (4,0), (4,3), dan (0,3) sehingga membentuk segi empat. Akan dilakukan operasi *shear*, *stretch*, *custom*, dan *reset* pada segiempat tersebut.




Input	Output	Analisis
-------	--------	----------

Tugas Besar II IF2123 Aljabar Geometri
Simulasi Transformasi Linier

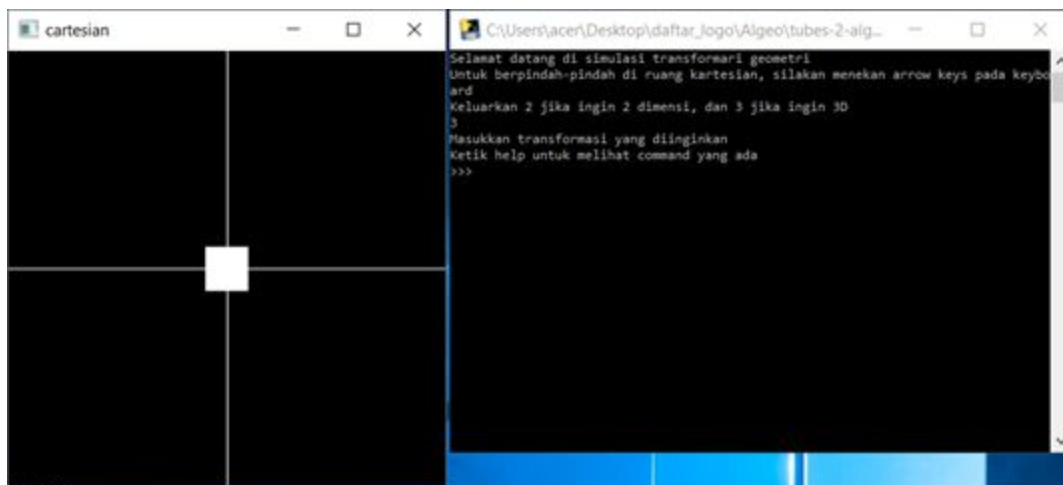
<p>shear x 1</p>		<p>Melakukan operasi <i>shear</i> terhadap sumbu x dengan faktor <i>shear</i> sebesar 1. Hasil dari operasi <i>shear</i> di samping didapat dengan mengali matriks transformasi dengan matriks yang dibentuk oleh segiempat.</p>
<p>stretch x 1.5</p>		<p>Melakukan operasi <i>stretch</i> terhadap sumbu x dengan faktor <i>stretch</i> sebesar 1,5. Hasil dari operasi <i>stretch</i> di samping didapat dengan mengali matriks transformasi dengan matriks yang dibentuk oleh segiempat.</p>
<p>custom 0 1 1 0</p>		<p>Melakukan transformasi terhadap segiempat dengan mengalikan matriks dari input pengguna yaitu dengan matriks yang dibentuk oleh segiempat.</p>

Tugas Besar II IF2123 Aljabar Geometri
Simulasi Transformasi Linier

reset		<p>Mengembalikan bentuk segiempat seperti keadaan semula yaitu terletak pada titik $(0,0)$, $(4,0)$, $(4,3)$, dan $(0,3)$.</p>
-------	-----------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------

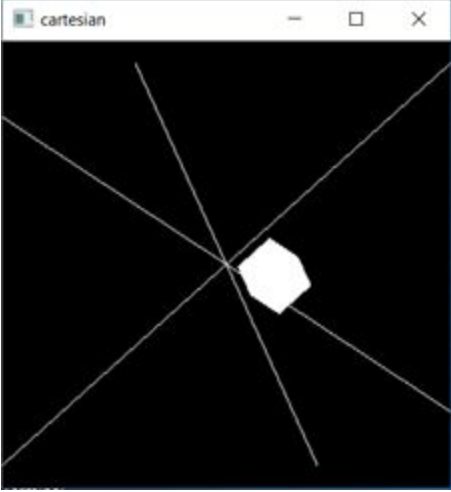


Hasil Eksekusi 3 :

Pada eksperimen ketiga, pengguna mencoba objek dalam bentuk tiga dimensi. Objek telah di-*hardcode* dalam program dan berbentuk kubus. Akan dilakukan operasi translasi, dilatasi, rotasi, refleksi, *shear*, dan *stretch* terhadap kubus.

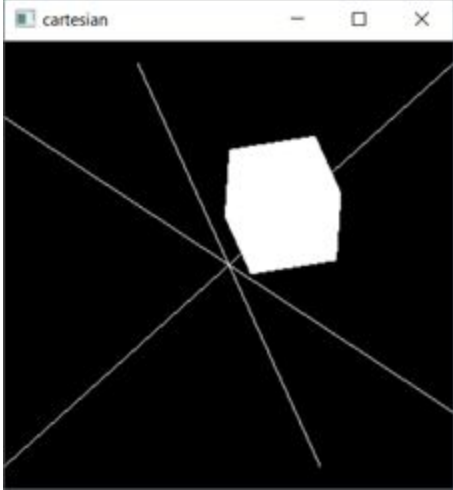
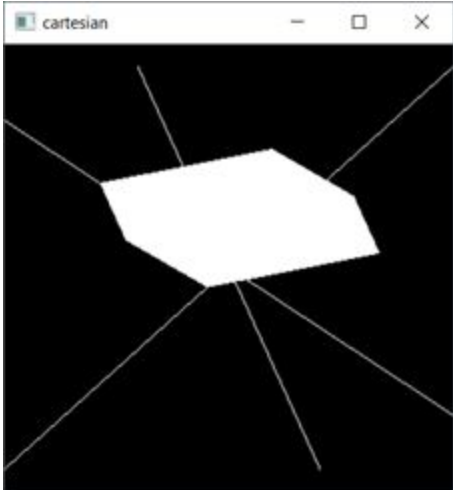



Input	Output	Analisis
-------	--------	----------

Tugas Besar II IF2123 Aljabar Geometri
Simulasi Transformasi Linier

<pre>translate 0.5 2 1</pre>		<p>Melakukan operasi translasi terhadap kubus. kubus digeser sejauh 0.5 terhadap x, sejauh 2 terhadap y, dan sejauh 1 terhadap z.</p>
<pre>dilate 2</pre>		<p>Melakukan operasi dilatasi dengan faktor skala 2. Dilatasi dilakukan dengan mengalikan matriks transformasi dengan matriks yang dibentuk oleh kubus.</p>
<pre>rotate 45 1 0 0</pre>		<p>Melakukan operasi rotasi terhadap kubus dengan sudut 45 derajat terhadap sumbu x.</p>

Tugas Besar II IF2123 Aljabar Geometri
Simulasi Transformasi Linier

<pre>reflect xz</pre>		Melakukan operasi refleksi atau pencerminan terhadap kubus terhadap bidang yang dibentuk sumbu x dan z.
<pre>shear y 1 1.5</pre>		Melakukan operasi <i>shear</i> terhadap sumbu y dengan faktor <i>shear</i> sebesar 1 terhadap sumbu x dan faktor <i>shear</i> sebesar 1.5 terhadap sumbu z.
<pre>stretch z 2 1</pre>		Melakukan operasi <i>stretch</i> terhadap sumbu z dengan faktor <i>stretch</i> sebesar 2 terhadap sumbu x dan faktor <i>stretch</i> sebesar 1 terhadap sumbu y.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Program simulasi transformasi linier ini dapat mensimulasikan operasi-operasi umum pada bidang 2D dan 3D. Dengan bantuan dari library OpenGL, program ini dapat menampilkan bentuk Graphical User Interface (GUI) dari simulasi tersebut. Untuk 2D, pengguna dapat memberikan titik-titik dalam bidang yang diinginkan dan program akan membuat sebuah polygon dari titik-titik tersebut. Sementara untuk 3D, program akan menampilkan sebuah kubus yang dapat ditransformasi sesuai keinginan pengguna.

5.2 Saran

Untuk program, mungkin dapat dikembangkan lebih baik terutama pada bagian 3D, dimana pengguna dapat memberikan bangun yang ingin ditransformasikan dengan memberikan titik dan garis dari bidang tersebut. Selain itu, program diharapkan dapat dibuat lebih modular agar pemrogram yang ingin mengembangkannya dapat mengubah-ubah tanpa mengubah fungsionalitas program sebelumnya.

5.3 Refleksi

Tugas ini sangat bermanfaat bagi kami semua. Hal ini dikarenakan kita diperkenalkan dengan bahasa baru berupa Python dan mendalami materi Aljabar Geometri terutama pada bagian transformasi linier. Diharapkan dari pengalaman yang didapatkan ini akan bermanfaat kedepannya untuk mengerjakan tugas-tugas lainnya dan mengembangkan kemampuan tiap individu agar dapat berkontribusi lebih kepada masyarakat di depannya.

Daftar Referensi

<http://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2018-2019/algeo18-19.htm>

<https://id.wikipedia.org/wiki/OpenGL>