LAPORAN
PROJEK PHYTON KUBUS 3D DAN PERSEGI 2D

Dosen Pengampu:

Teuku Riski Noviandi S.KOM .,M.KOM

Nama : Ferdinan Mahrus

NIM : 24146086

Prodi : Sistem Informasi

Mata Kuliah : Komputer Grafik

Code Python :

```
import pygame
from pygame.locals import *
from OpenGL.GL import *
from OpenGL.GLU import *

# ================= KUBUS 3D =================
cube_vertices = [
   (-1,-1,-1),(1,-1,-1),(1,1,-1),(-1,1,-1),
   (-1,-1,1),(1,-1,1),(1,1,1),(-1,1,1)
]

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

cube_pos = [-2,0,-6]
cube_rot_x = 0
cube_rot_y = 0
cube_scale = 1

def draw_cube():
   glEnable(GL_BLEND)
   glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)

   glColor4f(0.6, 0.6, 0.6, 0.5)  # abu transparan
   glBegin(GL_LINES)
   for edge in cube_edges:
```

```
        for v in edge:
            glVertex3fv(cube_vertices[v])
    glEnd()

    glDisable(GL_BLEND)

# ================= PERSEGI 2D =================
square_vertices = [
    (0,0,0),(2,0,0),(2,2,0),(0,2,0)
]

def apply_reflection(vertices, axis=None):
    result = []
    for x,y,z in vertices:
        if axis == 'x':
            result.append((x,-y,z))
        elif axis == 'y':
            result.append((-x,y,z))
        else:
            result.append((x,y,z))
    return result

def apply_shearing(vertices, shear_x=0):
    result = []
    for x,y,z in vertices:
        result.append((x + shear_x*y, y, z))
    return result

sq_pos = [2,0]
sq_rot = 0
sq_scale = 1
shear_x = 0
reflect_axis = None

def draw_square():
    v = apply_reflection(square_vertices, reflect_axis)
    v = apply_shearing(v, shear_x)

    glColor3f(0.0, 0.8, 0.0)  # HIJAU
    glBegin(GL_QUADS)
    for p in v:
        glVertex3fv(p)
    glEnd()

# ================= MAIN =================
pygame.init()
display = (800,600)
pygame.display.set_mode(display, DOUBLEBUF | OPENGL)
pygame.display.set_caption("Transformasi 2D & 3D")

clock = pygame.time.Clock()
```

```
while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            quit()
        elif event.type == KEYDOWN:
            if event.key == K_1: shear_x = 1
            if event.key == K_2: shear_x = 0
            if event.key == K_3: reflect_axis = 'x'
            if event.key == K_4: reflect_axis = 'y'
            if event.key == K_5: reflect_axis = None

    keys = pygame.key.get_pressed()

    # === KUBUS ===
    if keys[K_LEFT]: cube_pos[0] -= 0.1
    if keys[K_RIGHT]: cube_pos[0] += 0.1
    if keys[K_UP]: cube_pos[1] += 0.1
    if keys[K_DOWN]: cube_pos[1] -= 0.1
    if keys[K_w]: cube_pos[2] += 0.1
    if keys[K_s]: cube_pos[2] -= 0.1
    if keys[K_a]: cube_rot_y -= 5
    if keys[K_d]: cube_rot_y += 5
    if keys[K_q]: cube_rot_x -= 5
    if keys[K_e]: cube_rot_x += 5
    if keys[K_z]: cube_scale += 0.05
    if keys[K_x]: cube_scale = max(0.1, cube_scale - 0.05)

    # === PERSEGI ===
    if keys[K_i]: sq_pos[1] += 0.1
    if keys[K_k]: sq_pos[1] -= 0.1
    if keys[K_j]: sq_pos[0] -= 0.1
    if keys[K_l]: sq_pos[0] += 0.1
    if keys[K_u]: sq_rot += 5
    if keys[K_o]: sq_rot -= 5
    if keys[K_n]: sq_scale += 0.05
    if keys[K_m]: sq_scale = max(0.1, sq_scale - 0.05)

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glEnable(GL_DEPTH_TEST)

    # === KUBUS 3D ===
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    gluPerspective(45, display[0]/display[1], 0.1, 50)

    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()
    glTranslatef(*cube_pos)
    glRotatef(cube_rot_x,1,0,0)
```

```
glRotatef(cube_rot_y,0,1,0)
glScalef(cube_scale,cube_scale,cube_scale)
draw_cube()

# === PERSEGI 2D ===
glMatrixMode(GL_PROJECTION)
glLoadIdentity()
gluOrtho2D(-4,6,-4,4)

glMatrixMode(GL_MODELVIEW)
glLoadIdentity()
glTranslatef(sq_pos[0],sq_pos[1],0)
glRotatef(sq_rot,0,0,1)
glScalef(sq_scale,sq_scale,1)
draw_square()

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

A. PENJELASAN DATAIL CODE :

1. Import Library

```
import pygame
from pygame.locals import
from OpenGL.GL import
from OpenGL.GLU import
import math
```

Kode di atas digunakan untuk mengimpor pustaka yang diperlukan dalam pembuatan program grafika.

- pygame digunakan untuk membuat jendela aplikasi dan menangani input keyboard.
- pygame.locals berisi konstanta seperti DOUBLEBUF, OPENGL, dan tombol keyboard.
- OpenGL.GL berisi fungsi OpenGL untuk menggambar objek dan melakukan transformasi.
- OpenGL.GLU digunakan untuk mengatur kamera dan perspektif.
- math digunakan untuk perhitungan matematika, terutama pada proses rotasi dan shearing.

2. Inisialisasi Pygame dan OpenGL

```
pygame.init()
display = (800, 600)
pygame.display.set_mode(display, DOUBLEBUF | OPENGL)
pygame.display.set_caption("Transformasi Objek 2D dan 3D")
```

Bagian ini berfungsi untuk:

- Menginisialisasi modul Pygame.
- Membuat jendela dengan ukuran 800×600 piksel.
- Mengaktifkan mode OpenGL dan double bu>ering agar tampilan lebih halus.
- Memberikan judul pada jendela program.

3. Pengaturan Kamera dan Perspektif

```
gluPerspective(45, (display[0]/display[1]), 0.1, 50.0)
glTranslatef(0.0, 0.0, -10)
```

Penjelasan:

- gluPerspective digunakan untuk mengatur sudut pandang kamera dengan sudut 45 derajat.
- Rasio aspek disesuaikan dengan ukuran layar.
- Nilai 0.1 dan 50.0 merupakan jarak near dan far clipping plane.
- glTranslatef digunakan untuk memindahkan kamera ke belakang agar objek terlihat.

4. Pendefinisian Objek 3D (Kubus)

```
vertices = (
  (1, -1, -1),
  (1, 1, -1),
  (-1, 1, -1),
  (-1, -1, -1),
  (1, -1, 1),
```

```
    (1, 1, 1),
    (-1, -1, 1),
    (-1, 1, 1)
)
```

Kode ini mendefinisikan titik-titik (vertex) pembentuk kubus 3D.
Setiap titik memiliki koordinat (x, y, z) dalam ruang tiga dimensi.

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

Variabel edges berisi pasangan indeks vertex yang dihubungkan untuk membentuk sisi-sisi kubus.

```
def draw_cube():
    glBegin(GL_LINES)
    for edge in edges:
        for vertex in edge:
            glVertex3fv(vertices[vertex])
    glEnd()
```

Fungsi draw_cube() digunakan untuk menggambar kubus 3D:

- glBegin(GL_LINES) menandakan bahwa objek digambar menggunakan garis.
- glVertex3fv menggambar titik berdasarkan data vertex.
- glEnd() menandai akhir proses penggambaran.

5. Pendefinisian Objek 2D (Persegi)

```
square = [
    [0.5, 0.5],
    [-0.5, 0.5],
```

```
    [-0.5, -0.5],
    [0.5, -0.5]
]
```

Kode ini mendefinisikan sebuah objek persegi 2D menggunakan empat titik koordinat (x, y).
Objek ini akan digunakan untuk menerapkan transformasi 2D seperti translasi, rotasi, skala, shearing, dan refleksi.

6. Transformasi Translasi Objek 2D

```
def translate(points, tx, ty):
    return [[x + tx, y + ty] for x, y in points]
```

Fungsi ini memindahkan posisi objek:

- tx adalah perpindahan sumbu X
- ty adalah perpindahan sumbu Y
- Setiap titik akan ditambahkan nilai translasi

7. Transformasi Rotasi Objek 2D

```
def rotate(points, angle):
    rad = math.radians(angle)
    cos_a = math.cos(rad)
    sin_a = math.sin(rad)
    return [[x cos_a – y sin_a, x sin_a + y cos_a] for x, y in points]
```

Penjelasan:

- Sudut rotasi diubah dari derajat ke radian.
- Digunakan rumus rotasi matriks 2D.
- Setiap titik diputar terhadap titik pusat (0,0).

8. Transformasi Skala Objek 2D

```
def scale(points, sx, sy):
```

```
    return [[x sx, y sy] for x, y in points]
```

Fungsi ini memperbesar atau memperkecil objek:

- sx adalah faktor skala sumbu X
- sy adalah faktor skala sumbu Y

9. Transformasi Shearing Objek 2D

```
def shear_x(points, k):
    return [[x + k y, y] for x, y in points]
```

Shearing dilakukan dengan mengubah nilai X berdasarkan nilai Y.
Transformasi ini menyebabkan objek menjadi miring ke samping.

10. Transformasi Refleksi Objek 2D

```
def reflect_x(points):
    return [[x, -y] for x, y in points]
```

Refleksi sumbu X dilakukan dengan mengubah tanda koordinat Y.
Objek akan dicerminkan terhadap sumbu X.

11.
```
def draw_square(points):
    glBegin(GL_QUADS)
    for x, y in points:
        glVertex2f(x, y)
    glEnd()
```

Fungsi ini menggambar persegi 2D:

- GL_QUADS digunakan untuk menggambar bangun segi empat.
- Setiap titik diberikan ke OpenGL menggunakan glVertex2f.

ALUR PROGRAM :

1. Inisialisasi Awal Program

Program dimulai dengan proses inisialisasi library Pygame dan OpenGL.
Tahap ini bertujuan untuk menyiapkan lingkungan grafika sebelum objek digambar.

Langkah-langkah awal yang dilakukan program:

1. Menginisialisasi Pygame menggunakan pygame.init()
2. Membuat jendela tampilan dengan mode OpenGL
3. Mengatur kamera dan sistem koordinat
4. Menyiapkan objek 2D dan 3D yang akan ditampilkan

Tahap ini hanya dilakukan satu kali di awal program.

2. Pengaturan Sistem Koordinat

Program menggunakan dua sistem koordinat:

- Koordinat 3D untuk objek kubus
- Koordinat 2D untuk objek persegi

Pengaturan ini memungkinkan kedua objek ditampilkan dalam satu jendela tanpa saling bertabrakan.

Objek 3D menggunakan perspektif kamera (gluPerspective), sedangkan objek 2D menggunakan koordinat datar (x, y).

3. Main Loop (Perulangan Utama Program)

while True:

Perulangan utama ini berfungsi agar program terus berjalan selama jendela belum ditutup.
Seluruh proses input, transformasi, dan rendering terjadi di dalam loop ini.

a. Deteksi dan Penanganan Event

```
for event in pygame.event.get():
  if event.type == pygame.QUIT:
    pygame.quit()
    quit()
```

Bagian ini berfungsi untuk:

- Mendeteksi aksi pengguna
- Menutup program dengan aman saat jendela ditutup

b. Input Keyboard

Program membaca input dari keyboard untuk melakukan transformasi objek.

Contoh alur:

- Tombol tertentu ditekan
- Program memproses transformasi yang sesuai
- Objek diperbarui dan digambar ulang

Input ini memungkinkan pengguna berinteraksi langsung dengan objek.

4. Proses Transformasi Objek

Berdasarkan input pengguna, program menerapkan transformasi sebagai berikut:

a. Translasi

Objek 2D dipindahkan ke kanan, kiri, atas, atau bawah.

b. Rotasi

Objek diputar berdasarkan sudut tertentu.

c. Skala

Ukuran objek diperbesar atau diperkecil.

d. Shearing

Objek dimiringkan pada sumbu X atau Y.

e. Refleksi

Objek dicerminkan terhadap sumbu tertentu (sumbu X atau Y).

Transformasi dilakukan sebelum proses penggambaran, sehingga perubahan langsung terlihat di layar.

5. Proses Rendering (Penggambaran Objek)

glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

Perintah ini digunakan untuk:

- Membersihkan layar dari frame sebelumnya
- Mencegah objek bertumpuk secara visual

a. Menggambar Objek 3D

Kubus 3D digambar terlebih dahulu menggunakan fungsi draw_cube().

- Objek 3D ditampilkan dalam bentuk wireframe
- Menggunakan sistem koordinat 3 dimensi
- Dapat mengalami transformasi rotasi

b. Menggambar Objek 2D

Objek 2D digambar setelah objek 3D.

- Menggunakan fungsi draw_square()
- Transformasi 2D diterapkan sebelum penggambaran
- Warna dan bentuk tetap konsisten

6. Update Tampilan

pygame.display.flip()
pygame.time.wait(10)

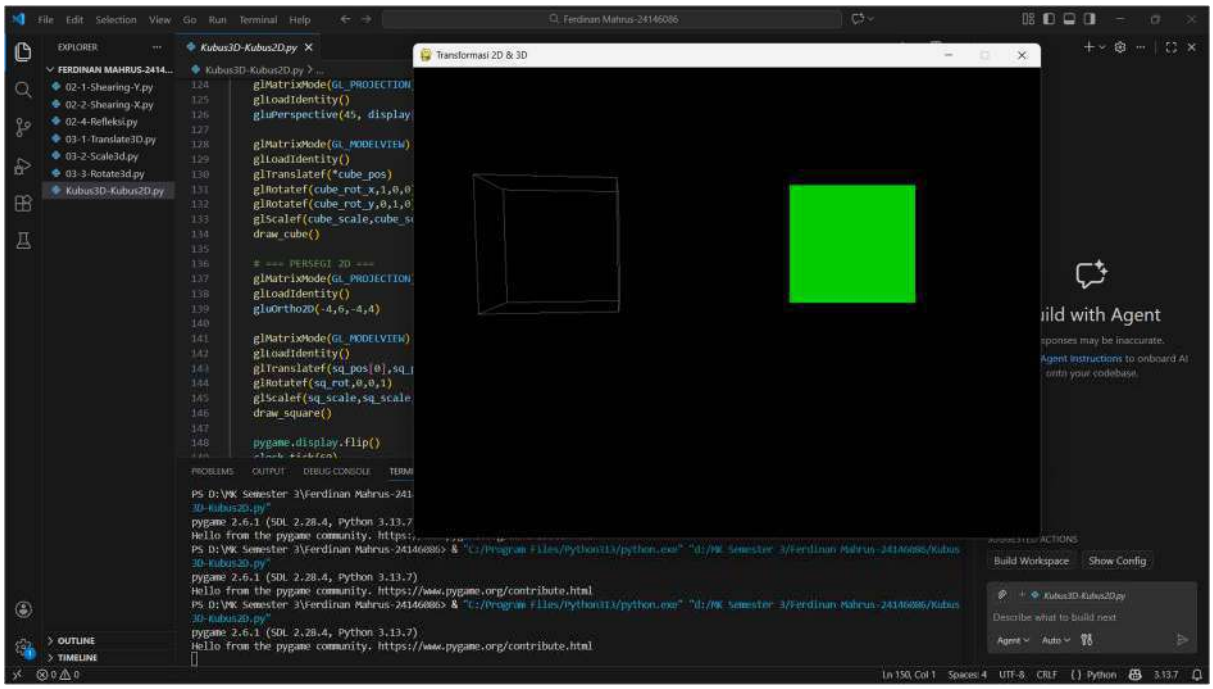
- pygame.display.flip() menampilkan hasil gambar ke layar
- pygame.time.wait(10) memberikan jeda waktu agar program tidak berjalan terlalu cepat

7. Akhir Program

Program akan terus berjalan sampai pengguna menutup jendela.
Saat jendela ditutup:

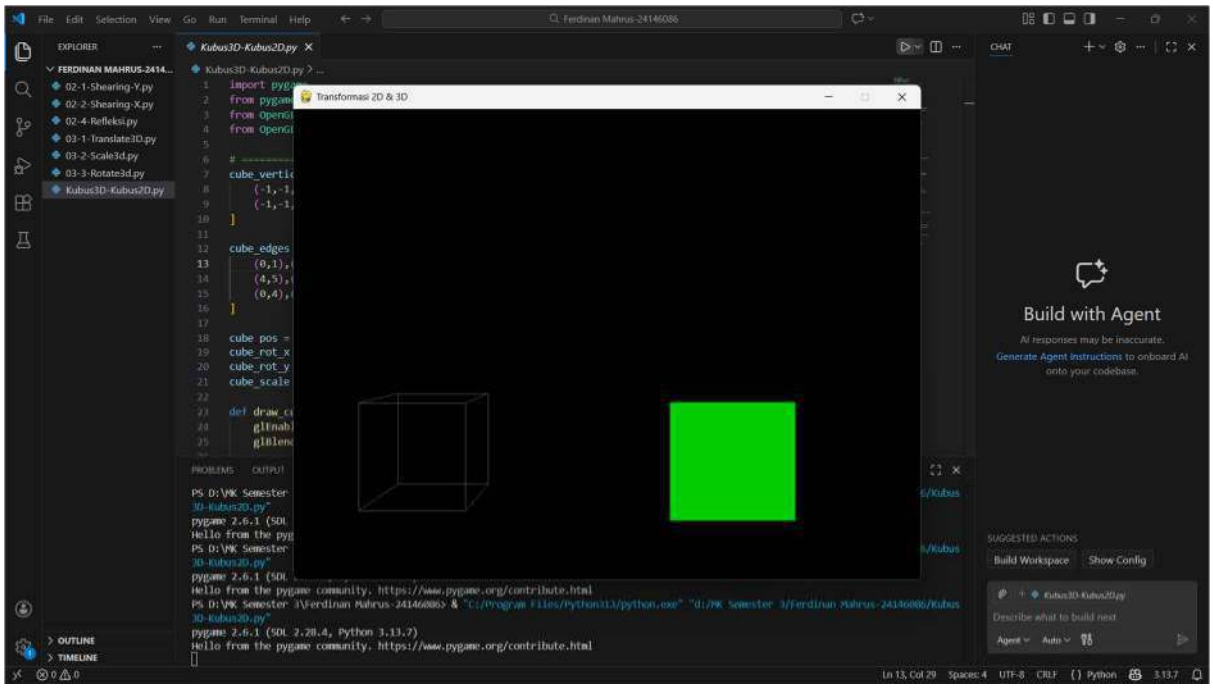
- Loop utama berhenti
- Pygame dimatikan
- Program keluar dengan aman

TAMPILAN AWAL :

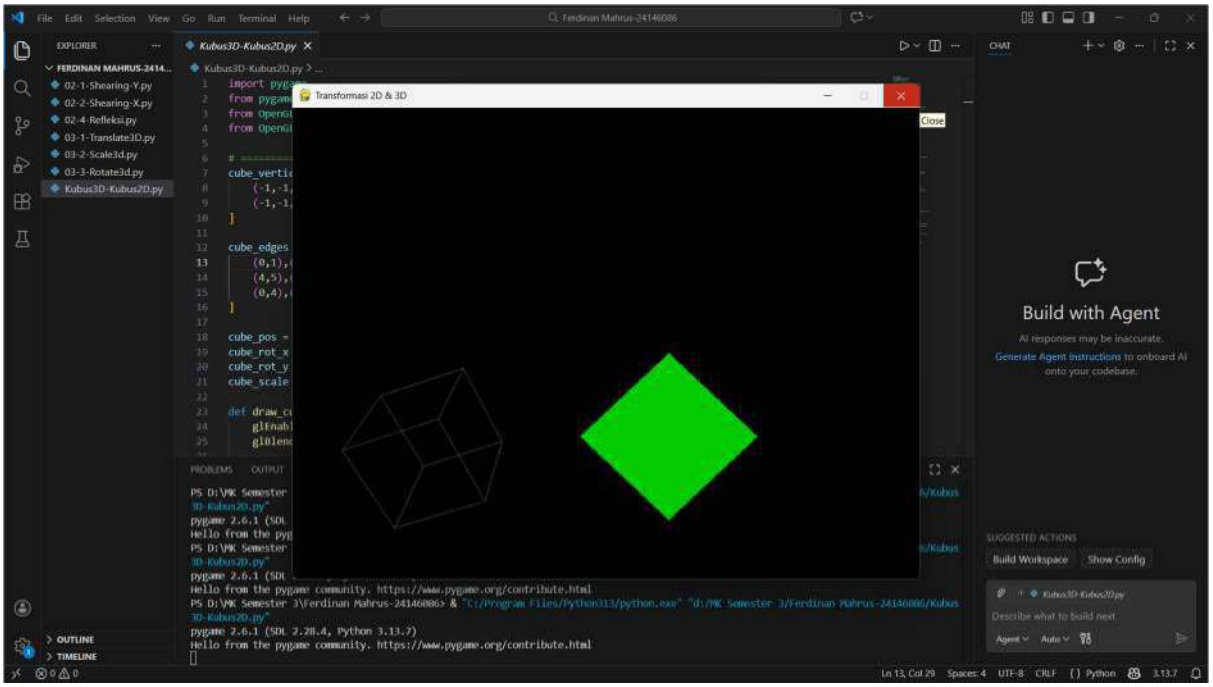


B. OUT PUT DARI PROGRAM :

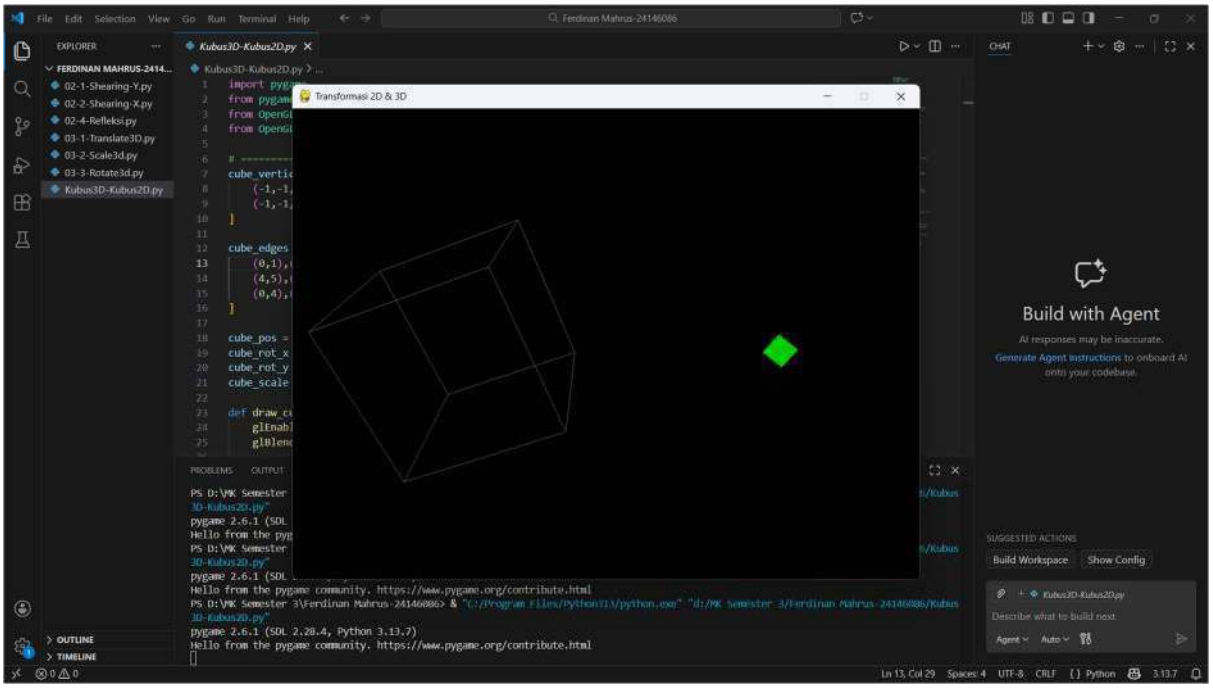
TRANSLASI X, Y, Z KUBUS 3D, TRANSLASI PERSEGI 2D



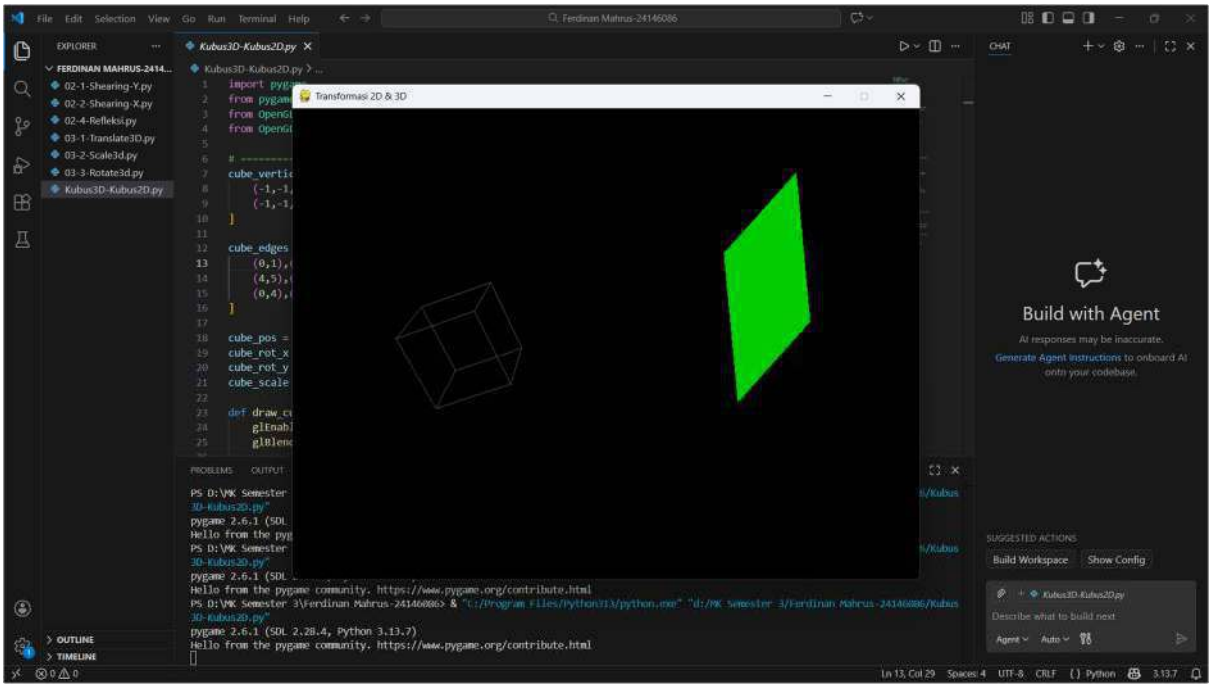
ROTASI X, Y KUBUS 3D, DAN ROTASI PERSEGI 2D



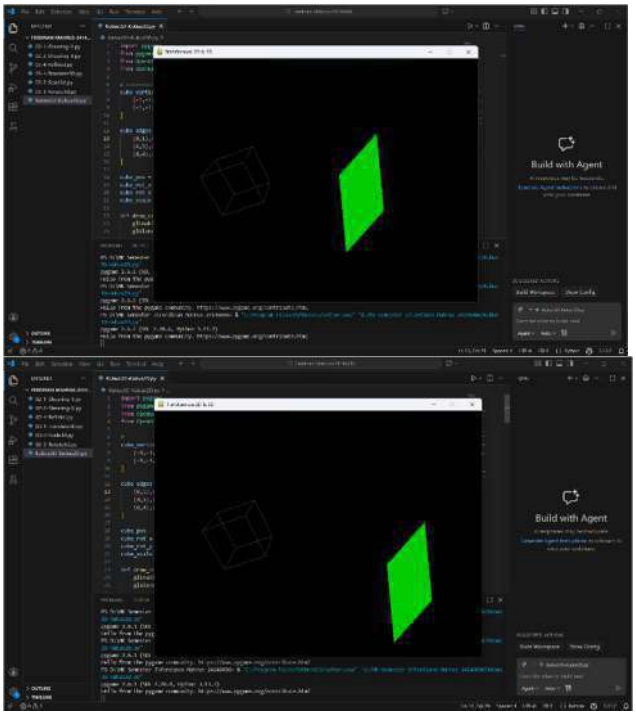
SKALA +/- KUBUS 3D, DAN SKALA PESEGI 2D



SHEARING X PADA PERSEGI 2D



REFLEKSI X, Y PERSEGI 2D



C. CARA MENJALANKAN PROGRAM :

1. Control Translasi Kubus 3D

Kontrol Keyboard Translasi

Tombol	Fungsi:
←	Geser ke kiri (sumbu X −)
→	Geser ke kanan (sumbu X +)
↑	Geser ke atas (sumbu Y +)
↓	Geser ke bawah (sumbu Y −)
W	Maju ke depan (sumbu Z +)
S	Mundur ke belakang (sumbu Z −)

Control Translasi Persegi 2D

Control Keyboard Translasi

Tombol	Fungsi:
I	Geser ke atas
K	Geser ke bawah
J	Geser ke kiri
L	Geser ke kanan

2. Control Rotasi Kubus 3D

Control Keyboard Rotasi

Tombol	Fungsi:
A	Rotasi sumbu Y (ke kiri)
D	Rotasi sumbu Y (ke kanan)
Q	Rotasi sumbu X (ke atas)
E	Rotasi sumbu X (ke bawah)

Control Rotasi Persegi 2D

Control Keyboard Rotasi

Tombol Fungsi:

U Rotasi searah jarum jam

O Rotasi berlawanan arah jarum jam

3. Control Skala Kubus 3D

Control Keyboard Skala

Tombol Fungsi:

Z Memperbesar kubus

X Memperkecil kubus

Control Skala Persegi 2D

Control Keyboard Skala

Tombol Fungsi:

N Memperbesar persegi

M Memperkecil persegi

4. Control Shearing Persegi 2D

Control Keyboard Shearing

Shearing digunakan untuk memiringkan objek pada sumbu tertentu.

Tombol Fungsi:

1 Aktifkan shearing sumbu X

2 Nonaktifkan shearing

5. Control Refleksi Persegi 2D

Control Keyboard Refleksi

Tombol	Fungsi
3	Refleksi terhadap sumbuh X
4	Refleksi terhadap sumbuh Y
5	Kembali kebentuk awal

D. Link GitHub

https://github.com/ferdinanmahrus07-source/Transformasi-3D-2D.git