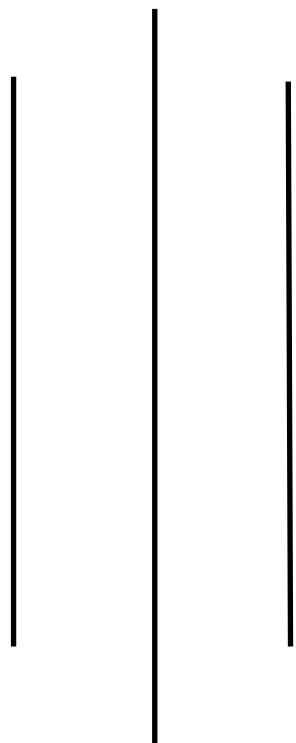


# LAPORAN UAS KOMPUTER GRAFIK

Dosen Pengampuh :

Teuku Risky Noviandi, S.KOM., M.KOM



NAMA : Rosa Juliana

NIM : 24146052

MK : Komputer Grafik

PROGRAM STUDI SISTEM INFORMASI  
FAKULTAS TEKNIK  
UNIVERSITAS ABULYATAMA

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(1.0, 0.5, 0.0) # ORANGE
    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()
glOrtho2D(-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)

```

#### PENJELASAN CODE :

##### 1. Pendahuluan

Program ini dibuat menggunakan bahasa pemrograman Python dengan bantuan library Pygame dan PyOpenGL. Tujuan dari program ini adalah untuk menampilkan serta

menerapkan transformasi geometri pada objek 3D (kubus) dan 2D (persegi) dalam satu jendela aplikasi.

Transformasi yang diterapkan meliputi:

- Translasi
- Rotasi
- Skala
- Shearing
- Refleksi

Pengendalian objek dilakukan melalui input keyboard.

## 2. Library yang Digunakan

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

Penjelasan:

- pygame → menangani window, event, dan input keyboard.
- OpenGL.GL → menyediakan fungsi dasar OpenGL untuk menggambar objek.
- OpenGL.GLU → menyediakan fungsi utilitas seperti perspektif kamera (gluPerspective) dan proyeksi ortogonal (gluOrtho2D).

## 3. Objek 3D: Kubus

### 3.1 Definisi Titik dan Sisi Kubus

```
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)
]
```

- Kubus terdiri dari 8 titik (vertex) dalam ruang 3 dimensi (x, y, z).

```
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\_edges berfungsi menghubungkan titik-titik untuk membentuk kerangka kubus.

### 3.2 Transformasi Kubus

```
cube_pos = [-2,0,-6]
```

```
cube_rot_x = 0
```

```
cube_rot_y = 0
```

```
cube_scale = 1
```

- Translasi: cube\_pos
- Rotasi: cube\_rot\_x, cube\_rot\_y
- Skala: cube\_scale

### 3.3 Fungsi Menggambar Kubus

```
def draw_cube():
    glEnable(GL_BLEND)
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)
    glColor4f(0.6, 0.6, 0.6, 0.5)
    • Kubus digambar sebagai kerangka (GL_LINES).
    • Warna abu-abu transparan menggunakan alpha blending.
```

## 4. Objek 2D: Persegi

### 4.1 Definisi Titik Persegi

```
square_vertices = [
    (0,0,0),(2,0,0),(2,2,0),(0,2,0)
]
```

- Persegi terdiri dari 4 titik dalam bidang 2D ( $z = 0$ ).

### 4.2 Transformasi Persegi

#### a. Refleksi

```
def apply_reflection(vertices, axis=None):
```

- Refleksi terhadap:
  - Sumbu X → y dibalik
  - Sumbu Y → x dibalik

#### b. Shearing

```
def apply_shearing(vertices, shear_x=0):
```

- Menggeser koordinat x berdasarkan nilai y.
- Digunakan untuk efek miring (shear).

#### c. Parameter Transformasi Persegi

```
sq_pos = [2,0]
```

```
sq_rot = 0
```

```
sq_scale = 1
```

```
shear_x = 0
```

```
reflect_axis = None
```

- Translasi, rotasi, skala, shearing, dan refleksi disimpan dalam variabel terpisah.

### 4.3 Fungsi Menggambar Persegi

```
glColor3f(1.0, 0.5, 0.0)
```

- Persegi diberi warna ORANGE.
- Digambar menggunakan GL\_QUADS.

### 5. Inisialisasi Window dan Loop Program

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

- Membuat window OpenGL dengan double buffering agar animasi halus.

```
while True:
```

- Loop utama program:
  - Mendeteksi event keyboard
  - Mengupdate transformasi
  - Menggambar ulang objek

## 6. Sistem Proyeksi

- Kubus 3D menggunakan:

```
gluPerspective(45, aspect_ratio, 0.1, 50)
```

- Persegi 2D menggunakan:

gluOrtho2D(-4,6,-4,4)

Ini menyebabkan objek 3D dan 2D berada pada sistem proyeksi yang berbeda.

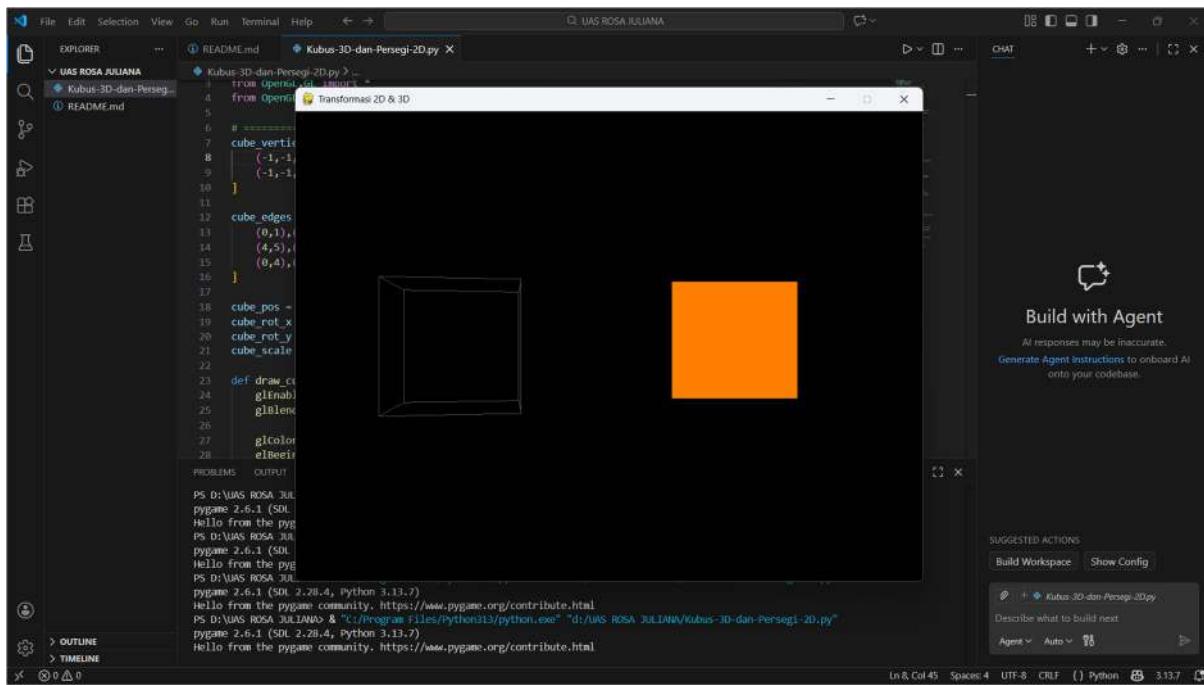
## 7. Kesimpulan

Program ini berhasil menampilkan objek 3D dan 2D serta menerapkan berbagai transformasi geometri secara interaktif menggunakan keyboard. Dengan memanfaatkan OpenGL dan Pygame, pengguna dapat memahami konsep transformasi grafika komputer secara visual dan praktis.

## TAMPILAN HASIL KODE :

## TRANSLASI, ROTASI, SKALA, SHEARING, DAN REFLEKSI

## TAMPILAN AWAL :



## TAMPILAN AKHIR :

The screenshot shows a Python development environment with several windows open. The main window displays a Python script named `Kubus-3D-dan-Persegi-2D.py`. The code uses Pygame to draw a 3D cube and a 2D parallelogram. The 3D cube is drawn with black lines on a white background. The 2D parallelogram is filled with orange. The environment includes a sidebar with project files like `README.md`, `Kubus-3D-dan-Persegi-2D.py`, and `Transformasi 2D & 3D`. A terminal window at the bottom shows the command-line output of running the script. A floating window titled "Build with Agent" provides AI integration options.

```
File Edit Selection View Go Run Terminal Help ← → MAS ROSA JULIANA EXPLORER README.md Kubus-3D-dan-Persegi-2D.py README.md Kubus-3D-dan-Persegi-2D.py from OpenGL.GL import * from OpenGL.GLU import * # cube vertices 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 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 = [0,0,0] cube_rot_x = 0 cube_rot_y = 0 cube_scale = 1 def draw_cube(): glTranslatef(cube_pos[0], cube_pos[1], cube_pos[2]) glRotatef(cube_rot_x, 1, 0, 0) glRotatef(cube_rot_y, 0, 1, 0) glBegin(GL_LINES) glColor3f(0,0,0) for edge in cube_edges: glVertex3fv(cube_vertices[edge[0]]) glVertex3fv(cube_vertices[edge[1]]) glEnd() glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA) glColor3f(1,0,0) glScalef(cube_scale, cube_scale, cube_scale) glBegin(GL_QUADS) elipsis
```

```
PS D:\MAS ROSA JULIANA> pygame 2.6.1 (SDL 2.28.4, Python 3.13.7)
Hello from the pygame community, https://www.pygame.org/contribute.html
PS D:\MAS ROSA JULIANA> & "C:\Program Files\Python311\python.exe" "d:\MAS ROSA JULIANA\Kubus-3D-dan-Persegi-2D.py"
pygame 2.6.1 (SDL 2.28.4, Python 3.13.7)
Hello from the pygame community. https://www.pygame.org/contribute.html
```

Build with Agent  
AI responses may be inaccurate.  
Generate Agent Instructions to onboard AI onto your codebase.

SUGGESTED ACTIONS  
Build Workspace Show Config

Kubus-3D-dan-Persegi-2D.py  
Describe what to build next  
Agent Auto

## CARA MENJALANKAN :

# Kontrol Keyboard

## Kontrol Kubus (3D)

## Tombol Fungsi

← → ↑ ↓ Translasi X & Y

## W/S Translasi Z

A/D Botasi su

O/E Botasi sumbu X

7 / 8 Skala besser / ke

Kontrol Persegi (2D)

Tombol Fungsi

I J K L    Translasi

U / O    Rotasi

N / M    Skala

1 / 2    Shearing aktif / nonaktif

3 / 4    Refleksi sumbu X / Y

5        Reset refleksi

Dapat di jalankan menggunakan control keyboard 3D dan 2D yang tertera di atas

LINK GITHUB :

<https://github.com/RosaJuliana/Kubus3D-2D-RosaJuliana.git>