**MODUL 5: 3D Vertex**

**Latihan 5\_1: Kubus 3D**

import pygame

from pygame.locals import \*

from OpenGL.GL import \*

from OpenGL.GLU import \*

import numpy as np

import math

**def** plotfunc():

    squares = [

        [[-1,-1,1], [1,-1,1], [1,1,1], [-1,1,1]],

        [[1,-1,-1], [1,1,-1], [-1,1,-1],[-1,-1,-1]],

        [[-1,1,-1], [-1,1,1], [1,1,1],[1,1,-1]],

        [[-1,-1,-1], [1,-1,-1], [1,-1,1], [-1,-1,1]],

        [[1,-1,-1], [1,1,-1], [1,1,1],[1,-1,1]],

        [[-1,-1,-1], [-1,-1,1], [-1,1,1],[-1,1,-1]],

    ]

    glClear (GL\_COLOR\_BUFFER\_BIT)

    glColor3f (0.0, 0.0, 0.0)

    glPointSize (1.0)

    for square in squares:

        glColor3f (1, 1, 0)

        glBegin(GL\_POLYGON)

        for point in square:

            glVertex3f (point[0], point[1], point[2])

        glEnd()

        glColor3f (0, 0, 0)

        glBegin (GL\_LINE\_LOOP)

        for point in square:

            glVertex3f (point [0], point [1], point [2])

        glEnd()

    glFlush ()

**def** main():

    pygame.init()

    display = (400,400)

    pygame.display.set\_caption('Nurul Izza Farhana 0102523729')

    pygame.display.set\_mode(display, DOUBLEBUF|OPENGL)

    gluPerspective(45, (display[0]/display[1]), 0.1, 50.0)

    glTranslatef(0.0,0.0, -5)

    glClearColor(1.0, 1.0, 1.0, 1.0)

    while True:

        for event in pygame.event.get():

            if event.type == pygame.QUIT:

                pygame.quit()

                quit()

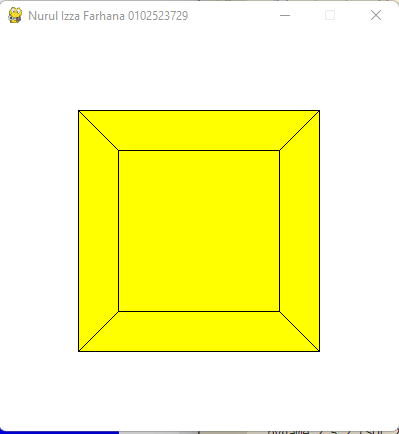
        glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT)

        plotfunc()

        pygame.display.flip()

        pygame.time.wait(10)

main()

****

**Latihan3\_2: Bangunan\_Sederhana**

**1. Gantilah objek kubus 3D dengan objek 3D menggunakan fungsi yang telah**

**disediakan oleh GLUT:**

import sys

import pygame

from screeninfo import get\_monitors

from OpenGL.GL import \*

from OpenGL.GLU import \*

from OpenGL.GLUT import \*

from pygame.locals import \*

**def** solidTeapot():

    glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)

    glutSolidTeapot(1.0)

    glutSwapBuffers()

**def** wireTeapot():

    glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)

    glutWireTeapot(1.0)

    glutSwapBuffers()

**def** solidCube():

    glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)

    glutSolidCube(1.0)

    glutSwapBuffers()

**def** wireCube():

    glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)

    glutWireCube(1.0)

    glutSwapBuffers()

**def** reshape(width, height):

    glViewport(0, 0, width, height)

    glMatrixMode(GL\_PROJECTION)

    glLoadIdentity()

    gluPerspective(45.0, width / height, 0.1, 100.0)

    glMatrixMode(GL\_MODELVIEW)

    glLoadIdentity()

    gluLookAt(0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0)

**def** get\_display\_size():

    primary\_monitor = get\_monitors()[0]

    width = primary\_monitor.width

    height = primary\_monitor.height

    return [width, height]

**def** get\_window\_size(scale=1):

    display\_size = get\_display\_size()

    size = round(display\_size[0]\*scale)

    if round(display\_size[1]\*scale) > size :

        size = round(display\_size[1]\*scale)

    return size

**def** main():

    glutInit(sys.argv)

    glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH)

    scale = 0.4

    size = get\_window\_size(scale)

    glutInitWindowSize(size,size)

    glutCreateWindow(**b**"Hana")

    glEnable(GL\_DEPTH\_TEST)

    glutDisplayFunc(solidCube)

    glutDisplayFunc(wireCube)

    glutDisplayFunc(wireTeapot)

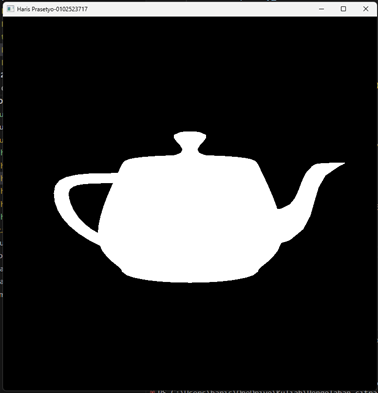
    glutDisplayFunc(solidTeapot)

    glutReshapeFunc(reshape)

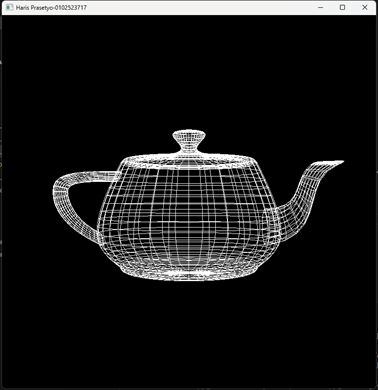
    glutMainLoop()

main()

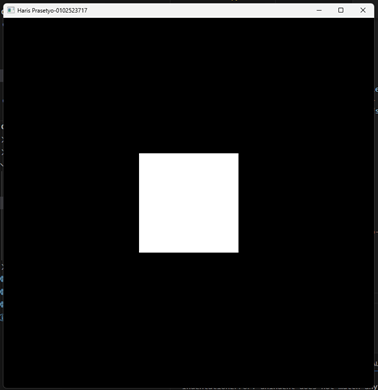
1. **SolidTeapot**

****

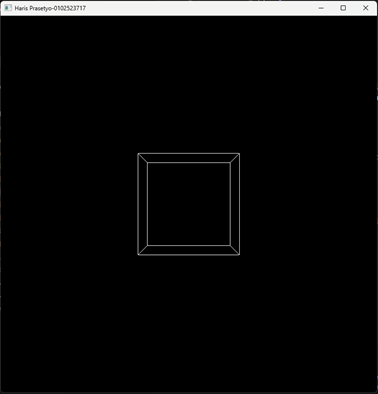
1. **WireTeapot**

****

1. **SolidCube**

****

1. **WireCube**

****

**2. Kesimpulan dari latihan pada Modul 6**

Dapat membuat object dengan menggunakan fungsi yang disediakan oleh module GLUT.

**TUGAS**

Buatlah sebuah bangunan yang menggunakan dua atau lebih bangun dasar yang ada

(contoh: balok, limas, atau prisma)

from screeninfo import get\_monitors

from OpenGL.GL import \*

from OpenGL.GLUT import \*

from OpenGL.GLU import \*

**def** draw\_cube(x, y, z, size):

    glPushMatrix()

    glTranslatef(x, y, z)

    glutSolidCube(size)

    glPopMatrix()

**def** draw\_sphere(x, y, z, radius, slices=30, stacks=30):

    glPushMatrix()

    glTranslatef(x, y, z)

    glutSolidSphere(radius, slices, stacks)

    glPopMatrix()

**def** display():

    glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)

    glLoadIdentity()

    gluLookAt(5, 5, 5, 0, 0, 0, 0, 1, 0)

    glColor3f(1.0, 0.0, 0.0)  *# warna merah*

    draw\_cube(-1, 0, 0, 1)

    glColor3f(0.0, 0.0, 1.0)  *# warna biru*

    draw\_cube(1, 0, 0, 1)

    glColor3f(0.0, 1.0, 0.0)  *# warna hijau*

    draw\_sphere(0,0,0, 0.5)

    glutSwapBuffers()

**def** reshape(width, height):

    glViewport(0, 0, width, height)

    glMatrixMode(GL\_PROJECTION)

    glLoadIdentity()

    gluPerspective(45, (width / height), 0.1, 100.0)

    glMatrixMode(GL\_MODELVIEW)

**def** get\_display\_size():

    primary\_monitor = get\_monitors()[0]

    width = primary\_monitor.width

    height = primary\_monitor.height

    return [width, height]

**def** get\_window\_size(scale=1):

    display\_size = get\_display\_size()

    size = round(display\_size[0]\*scale)

    if round(display\_size[1]\*scale) > size :

        size = round(display\_size[1]\*scale)

    return size

**def** main():

    scale = 0.4

    size = get\_window\_size(scale)

    glutInitWindowSize(size,size)

    glutInit(sys.argv)

    glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH)

    glutCreateWindow(**b**"Hana")

    glEnable(GL\_DEPTH\_TEST)

    glutDisplayFunc(display)

    glutReshapeFunc(reshape)

    glutMainLoop()

main()

