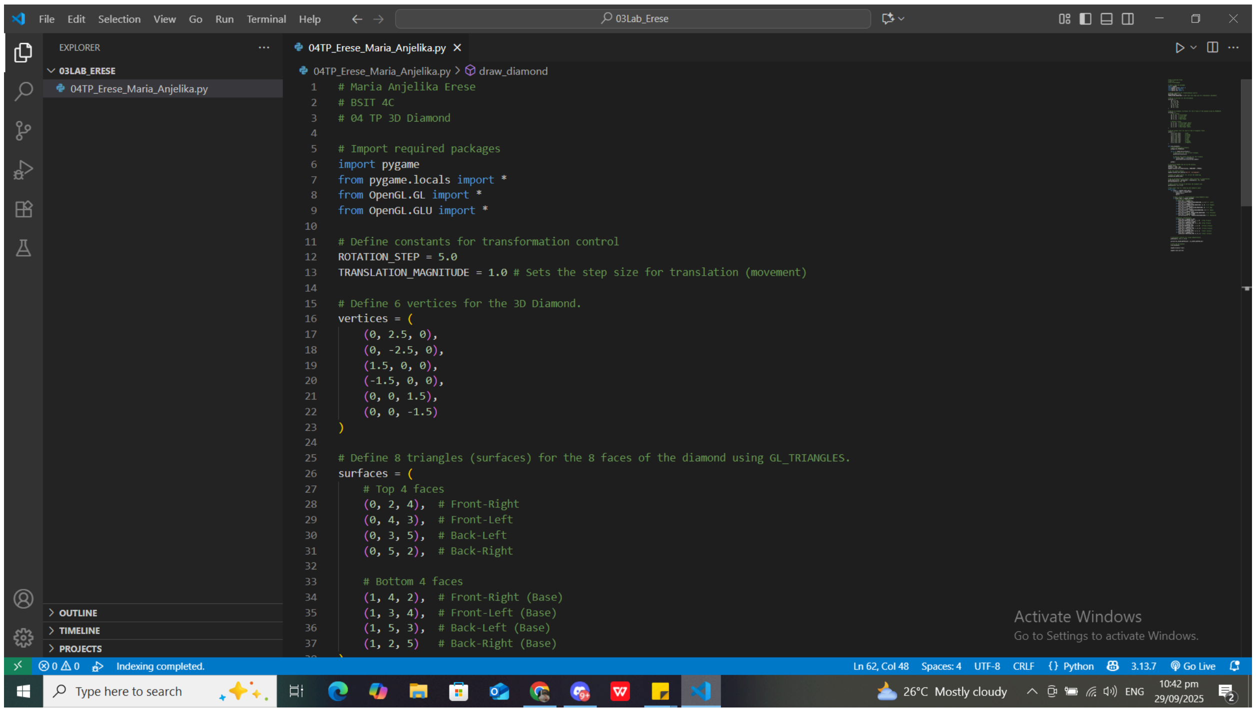
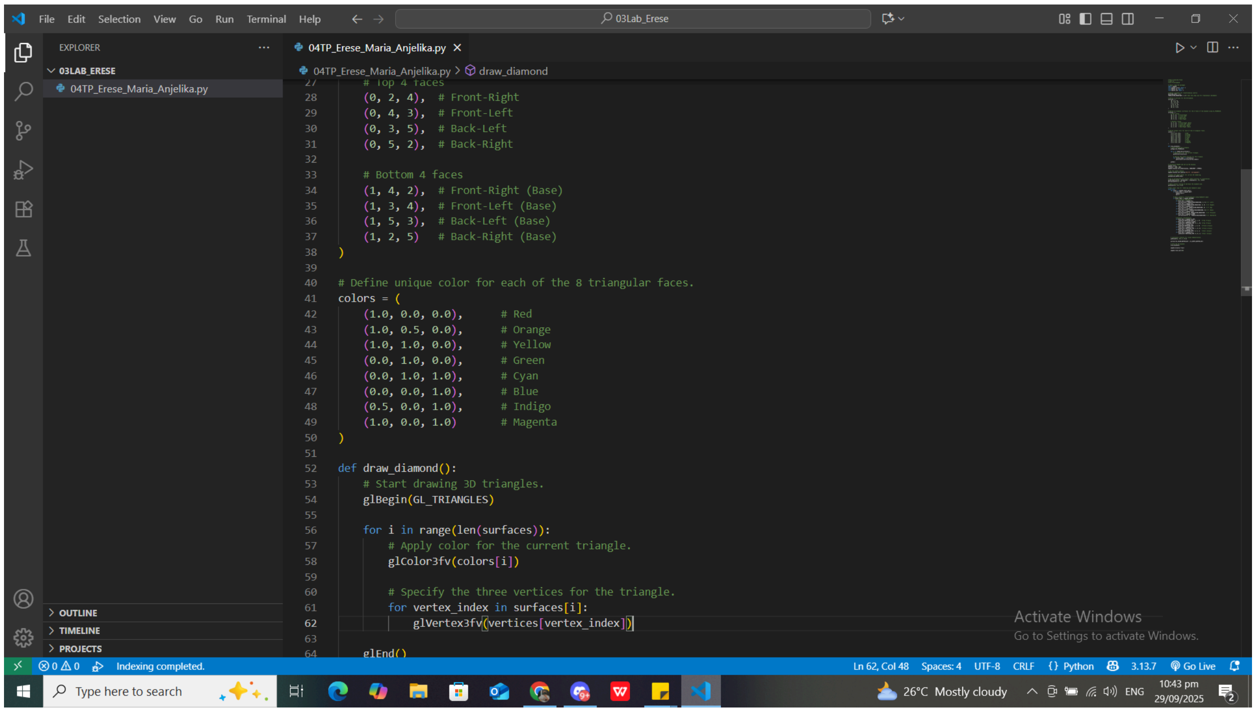
Maria Anjelika Erese
BSIT 4C
04 Task Performance

Instructions:

Draw a 3D diamond using GL_TRIANGLES. Then, apply geometric transformations to it using your keyboard. Demonstrate your output to your instructor and/or upload your .py code.

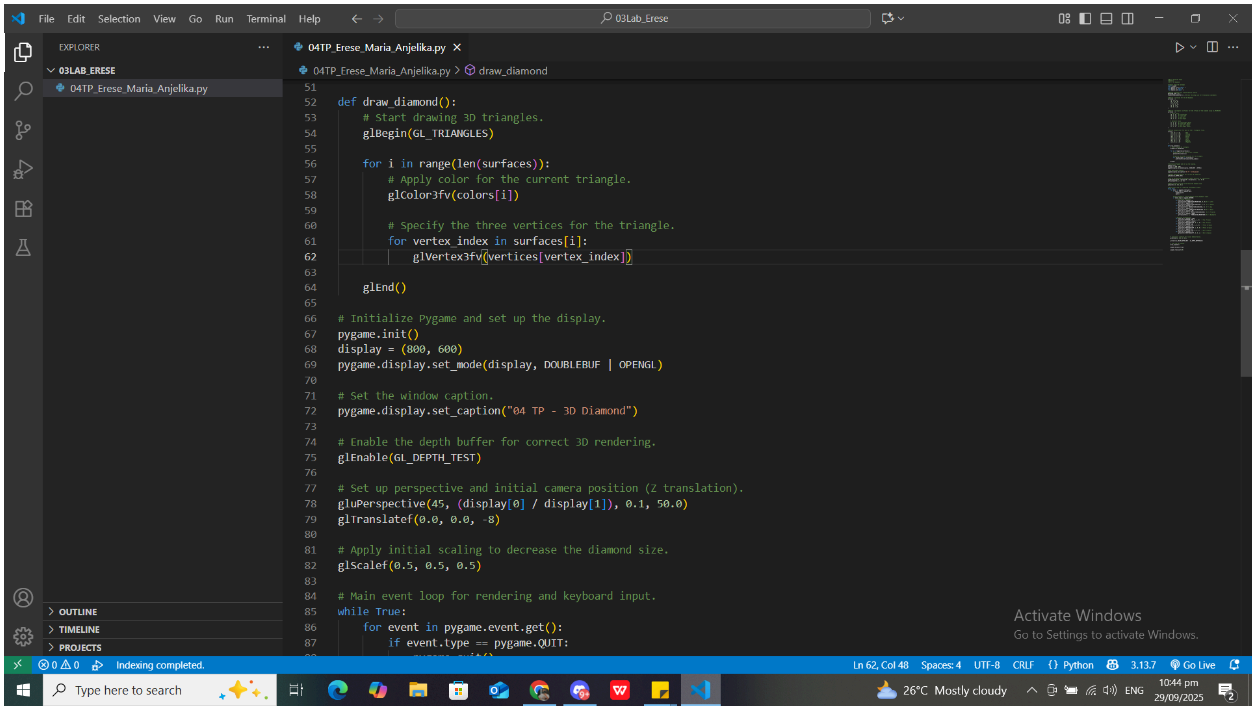


```python
# Maria Anjelika Erese
# BSIT 4C
# 04 TP 3D Diamond

# Import required packages
import pygame
from pygame.locals import *
from OpenGL.GL import *
from OpenGL.GLU import *

# Define constants for transformation control
ROTATION_STEP = 5.0
TRANSLATION_MAGNITUDE = 1.0 # Sets the step size for translation (movement)

# Define 6 vertices for the 3D Diamond.
vertices = (
    (0, 2.5, 0),
    (0, -2.5, 0),
    (1.5, 0, 0),
    (-1.5, 0, 0),
    (0, 0, 1.5),
    (0, 0, -1.5)
)

# Define 8 triangles (surfaces) for the 8 faces of the diamond using GL_TRIANGLES.
surfaces = (
    # Top 4 faces
    (0, 2, 4),  # Front-Right
    (0, 4, 3),  # Front-Left
    (0, 3, 5),  # Back-Left
    (0, 5, 2),  # Back-Right

    # Bottom 4 faces
    (1, 4, 2),  # Front-Right (Base)
    (1, 3, 4),  # Front-Left (Base)
    (1, 5, 3),  # Back-Left (Base)
    (1, 2, 5)   # Back-Right (Base)
```

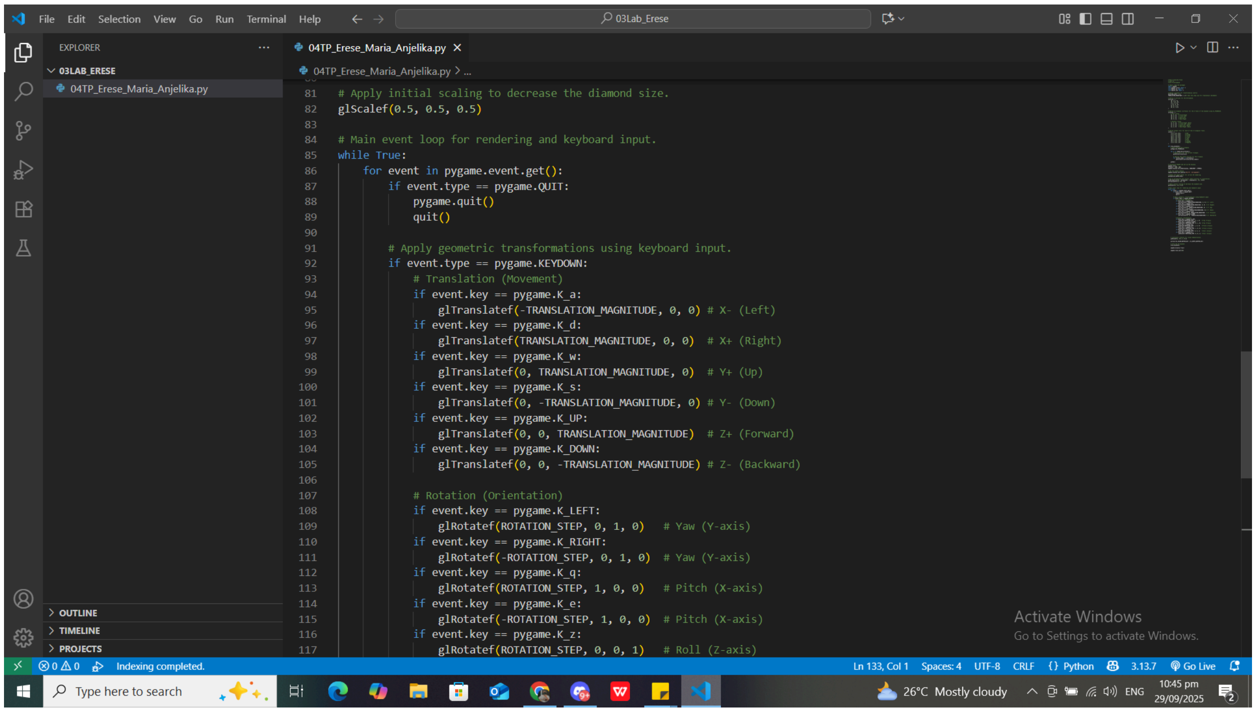


```python
    # Top 4 faces
    (0, 2, 4),  # Front-Right
    (0, 4, 3),  # Front-Left
    (0, 3, 5),  # Back-Left
    (0, 5, 2),  # Back-Right

    # Bottom 4 faces
    (1, 4, 2),  # Front-Right (Base)
    (1, 3, 4),  # Front-Left (Base)
    (1, 5, 3),  # Back-Left (Base)
    (1, 2, 5)   # Back-Right (Base)
```

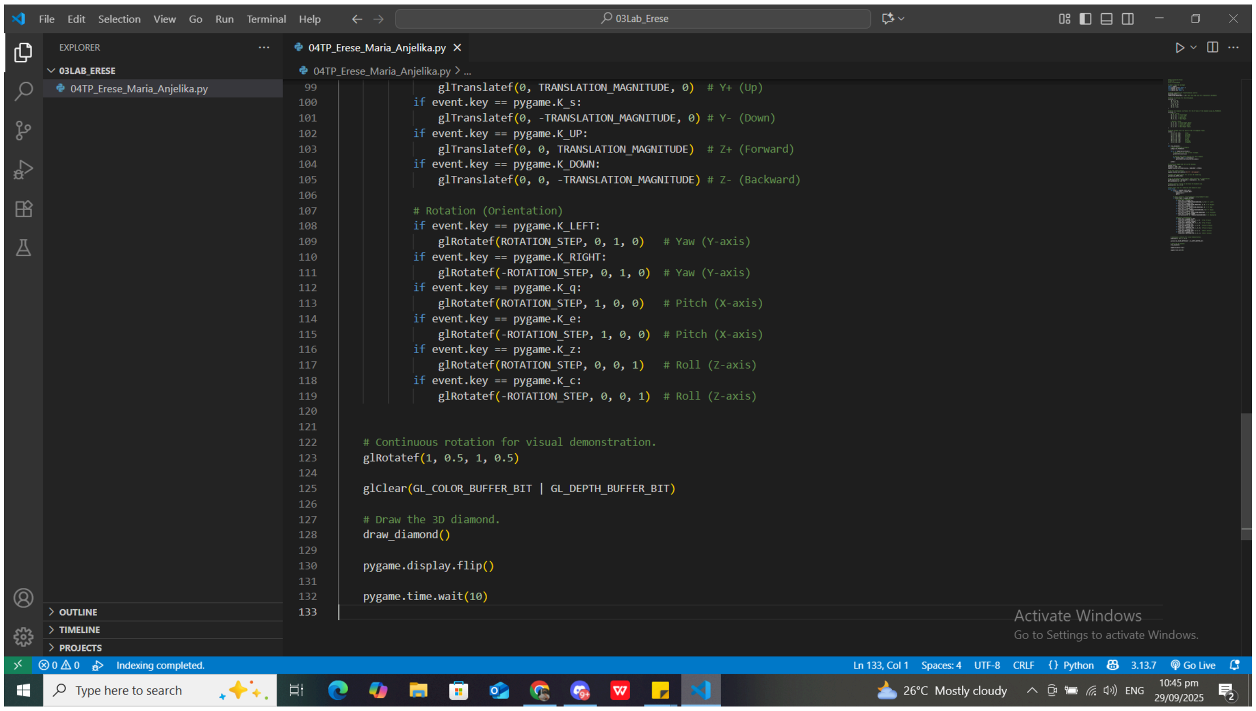
```python
)

# Define unique color for each of the 8 triangular faces.
colors = (
    (1.0, 0.0, 0.0),       # Red
    (1.0, 0.5, 0.0),       # Orange
    (1.0, 1.0, 0.0),       # Yellow
    (0.0, 1.0, 0.0),       # Green
    (0.0, 1.0, 1.0),       # Cyan
    (0.0, 0.0, 1.0),       # Blue
    (0.5, 0.0, 1.0),       # Indigo
    (1.0, 0.0, 1.0)        # Magenta
)

def draw_diamond():
    # Start drawing 3D triangles.
    glBegin(GL_TRIANGLES)

    for i in range(len(surfaces)):
        # Apply color for the current triangle.
        glColor3fv(colors[i])

        # Specify the three vertices for the triangle.
        for vertex_index in surfaces[i]:
            glVertex3fv(vertices[vertex_index])

    glEnd()
```


03Lab_Erese
EXPLORER
03LAB_ERESE
04TP_Erese_Maria_Anjelika.py
04TP_Erese_Maria_Anjelika.py > draw_diamond
Activate Windows
Go to Settings to activate Windows.
Indexing completed.
Ln 62, Col 48 Spaces: 4 UTF-8 CRLF {} Python 3.13.7 Go Live
Type here to search
26°C Mostly cloudy ENG 10:44 pm 29/09/2025

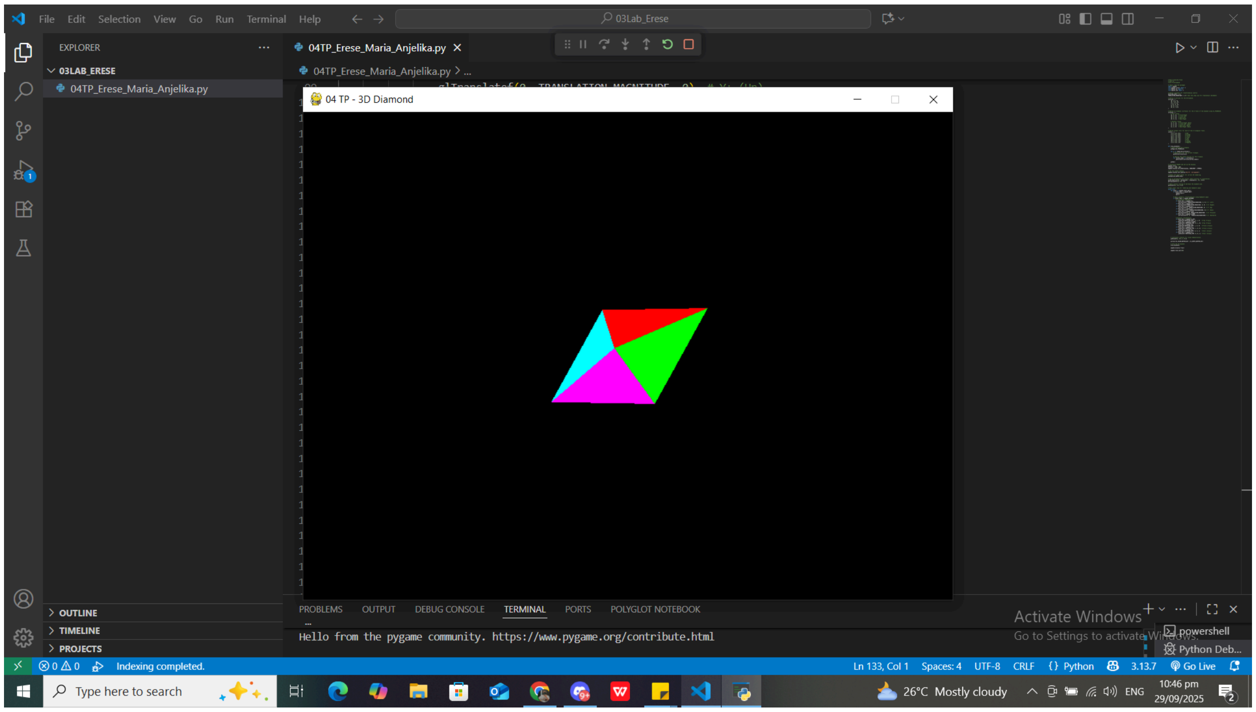
```python

def draw_diamond():
    # Start drawing 3D triangles.
    glBegin(GL_TRIANGLES)

    for i in range(len(surfaces)):
        # Apply color for the current triangle.
        glColor3fv(colors[i])

        # Specify the three vertices for the triangle.
        for vertex_index in surfaces[i]:
            glVertex3fv(vertices[vertex_index])

    glEnd()

# Initialize Pygame and set up the display.
pygame.init()
display = (800, 600)
pygame.display.set_mode(display, DOUBLEBUF | OPENGL)

# Set the window caption.
pygame.display.set_caption("04 TP - 3D Diamond")

# Enable the depth buffer for correct 3D rendering.
glEnable(GL_DEPTH_TEST)

# Set up perspective and initial camera position (Z translation).
gluPerspective(45, (display[0] / display[1]), 0.1, 50.0)
glTranslatef(0.0, 0.0, -8)

# Apply initial scaling to decrease the diamond size.
glScalef(0.5, 0.5, 0.5)

# Main event loop for rendering and keyboard input.
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
```


03Lab_Erese
EXPLORER
03LAB_ERESE
04TP_Erese_Maria_Anjelika.py
04TP_Erese_Maria_Anjelika.py > ...
Activate Windows
Go to Settings to activate Windows.
Indexing completed.
Ln 133, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.7 Go Live
Type here to search
26°C Mostly cloudy ENG 10:45 pm 29/09/2025

```python
# Apply initial scaling to decrease the diamond size.
glScalef(0.5, 0.5, 0.5)

# Main event loop for rendering and keyboard input.
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            quit()

        # Apply geometric transformations using keyboard input.
        if event.type == pygame.KEYDOWN:
            # Translation (Movement)
            if event.key == pygame.K_a:
                glTranslatef(-TRANSLATION_MAGNITUDE, 0, 0) # X- (Left)
            if event.key == pygame.K_d:
                glTranslatef(TRANSLATION_MAGNITUDE, 0, 0)  # X+ (Right)
            if event.key == pygame.K_w:
                glTranslatef(0, TRANSLATION_MAGNITUDE, 0)  # Y+ (Up)
            if event.key == pygame.K_s:
                glTranslatef(0, -TRANSLATION_MAGNITUDE, 0) # Y- (Down)
            if event.key == pygame.K_UP:
                glTranslatef(0, 0, TRANSLATION_MAGNITUDE)  # Z+ (Forward)
            if event.key == pygame.K_DOWN:
                glTranslatef(0, 0, -TRANSLATION_MAGNITUDE) # Z- (Backward)

            # Rotation (Orientation)
            if event.key == pygame.K_LEFT:
                glRotatef(ROTATION_STEP, 0, 1, 0)   # Yaw (Y-axis)
            if event.key == pygame.K_RIGHT:
                glRotatef(-ROTATION_STEP, 0, 1, 0)  # Yaw (Y-axis)
            if event.key == pygame.K_q:
                glRotatef(ROTATION_STEP, 1, 0, 0)   # Pitch (X-axis)
            if event.key == pygame.K_e:
                glRotatef(-ROTATION_STEP, 1, 0, 0)  # Pitch (X-axis)
            if event.key == pygame.K_z:
                glRotatef(ROTATION_STEP, 0, 0, 1)   # Roll (Z-axis)
```



```python
                    glTranslatef(0, TRANSLATION_MAGNITUDE, 0)  # Y+ (Up)
                if event.key == pygame.K_s:
                    glTranslatef(0, -TRANSLATION_MAGNITUDE, 0) # Y- (Down)
                if event.key == pygame.K_UP:
                    glTranslatef(0, 0, TRANSLATION_MAGNITUDE)  # Z+ (Forward)
                if event.key == pygame.K_DOWN:
                    glTranslatef(0, 0, -TRANSLATION_MAGNITUDE) # Z- (Backward)

                # Rotation (Orientation)
                if event.key == pygame.K_LEFT:
                    glRotatef(ROTATION_STEP, 0, 1, 0)   # Yaw (Y-axis)
                if event.key == pygame.K_RIGHT:
                    glRotatef(-ROTATION_STEP, 0, 1, 0)  # Yaw (Y-axis)
                if event.key == pygame.K_q:
                    glRotatef(ROTATION_STEP, 1, 0, 0)   # Pitch (X-axis)
                if event.key == pygame.K_e:
                    glRotatef(-ROTATION_STEP, 1, 0, 0)  # Pitch (X-axis)
                if event.key == pygame.K_z:
                    glRotatef(ROTATION_STEP, 0, 0, 1)   # Roll (Z-axis)
                if event.key == pygame.K_c:
                    glRotatef(-ROTATION_STEP, 0, 0, 1)  # Roll (Z-axis)


        # Continuous rotation for visual demonstration.
        glRotatef(1, 0.5, 1, 0.5)

        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

        # Draw the 3D diamond.
        draw_diamond()

        pygame.display.flip()

        pygame.time.wait(10)
```



```
Hello from the pygame community. https://www.pygame.org/contribute.html
```