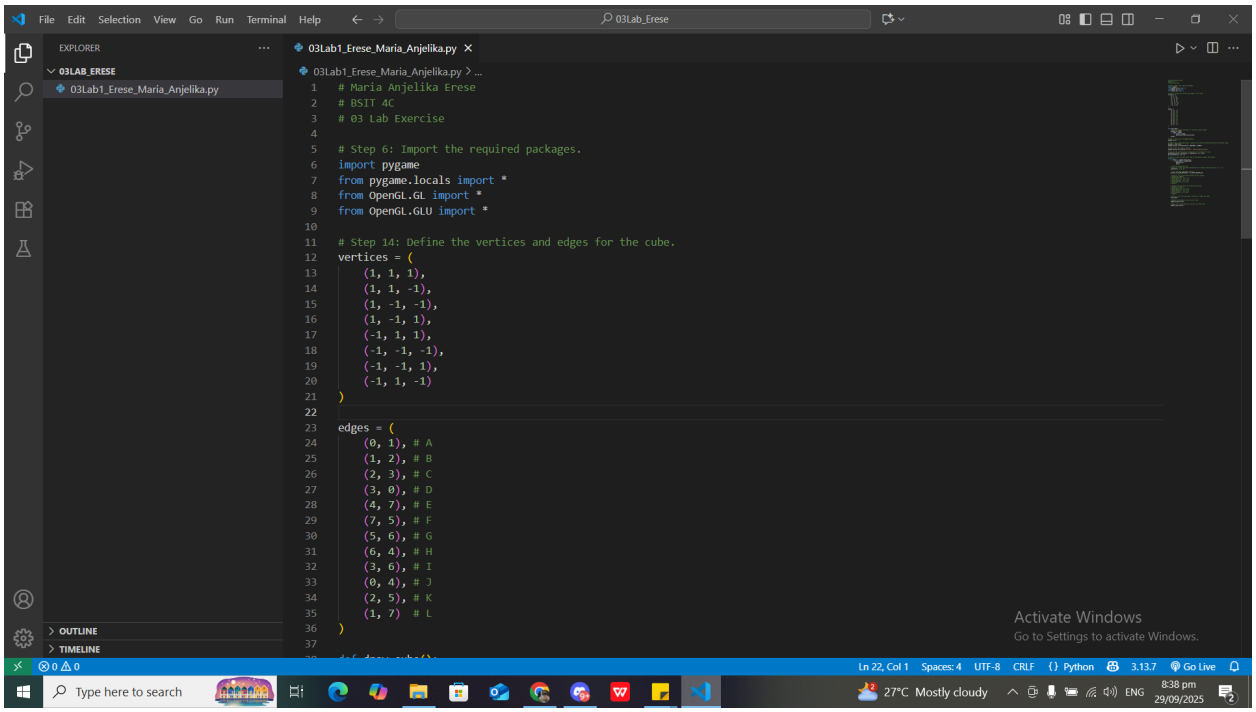
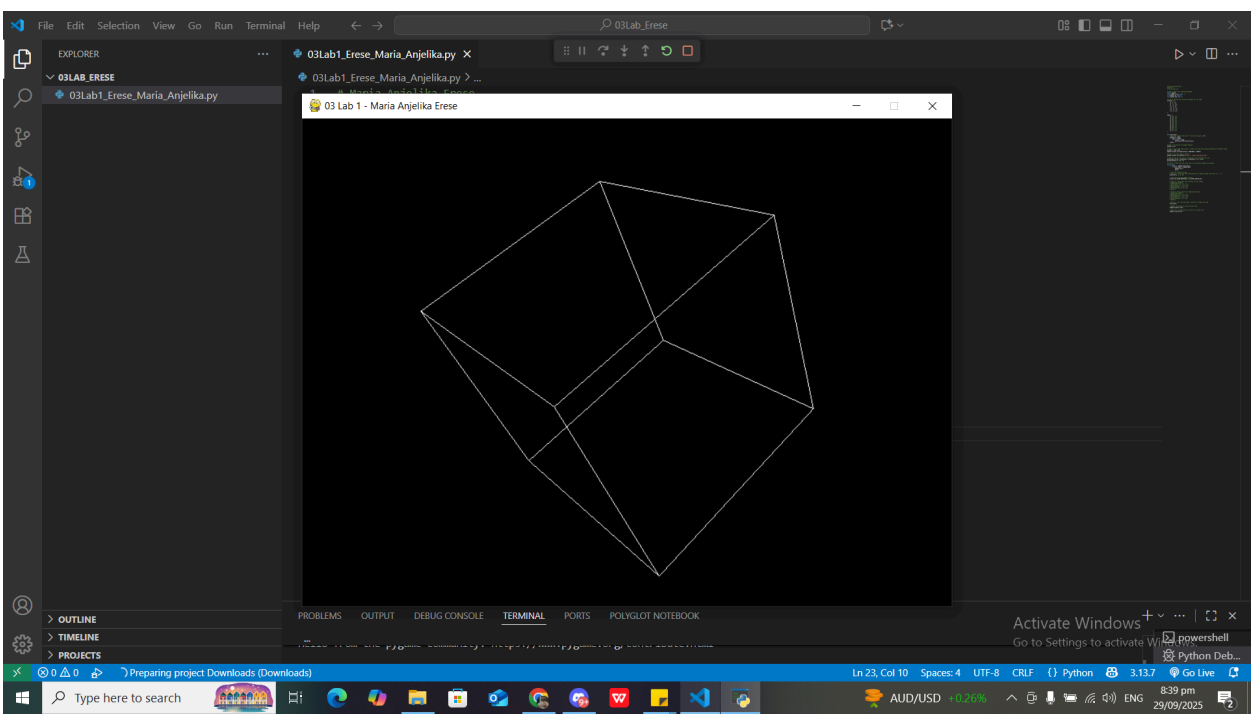
Maria Anjelika Erese
BSIT 4C
04 Laboratory Exercise
Procedure:

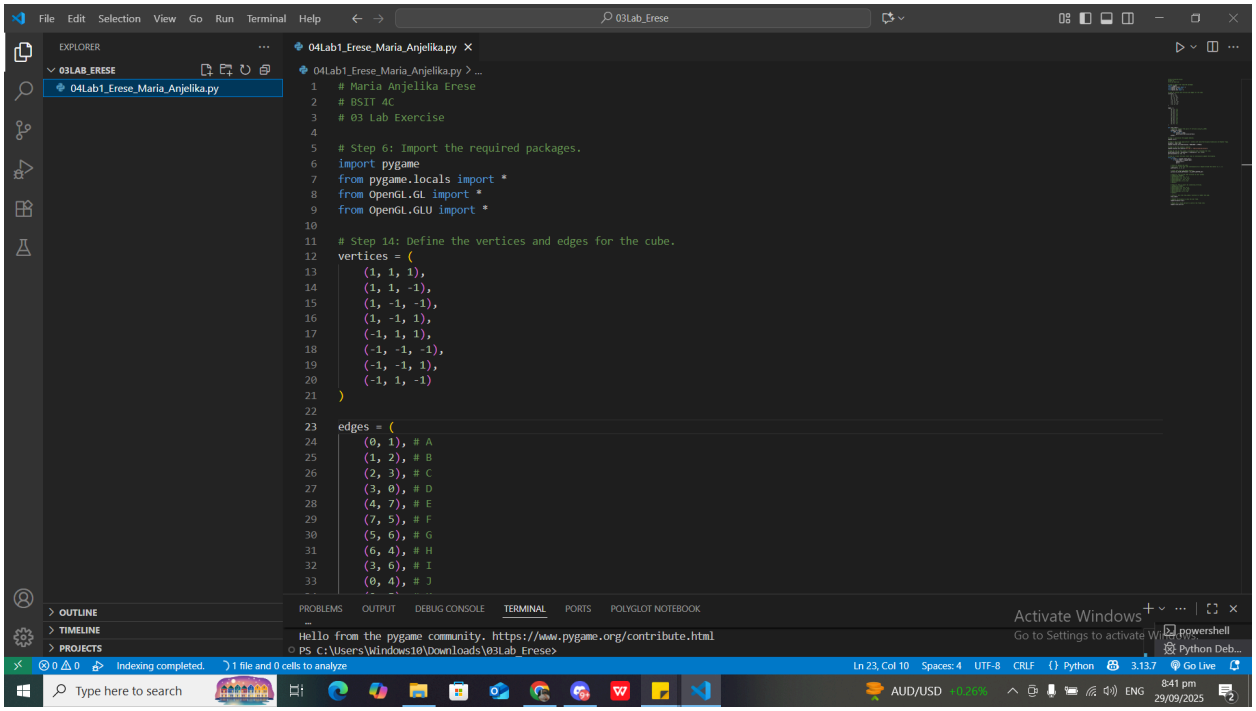
1. Launch IDLE and open your Python module from 03 Laboratory Exercise 1.



2. Save a copy of your module using a different filename, then run it by clicking Run > Run Module or by pressing F5.



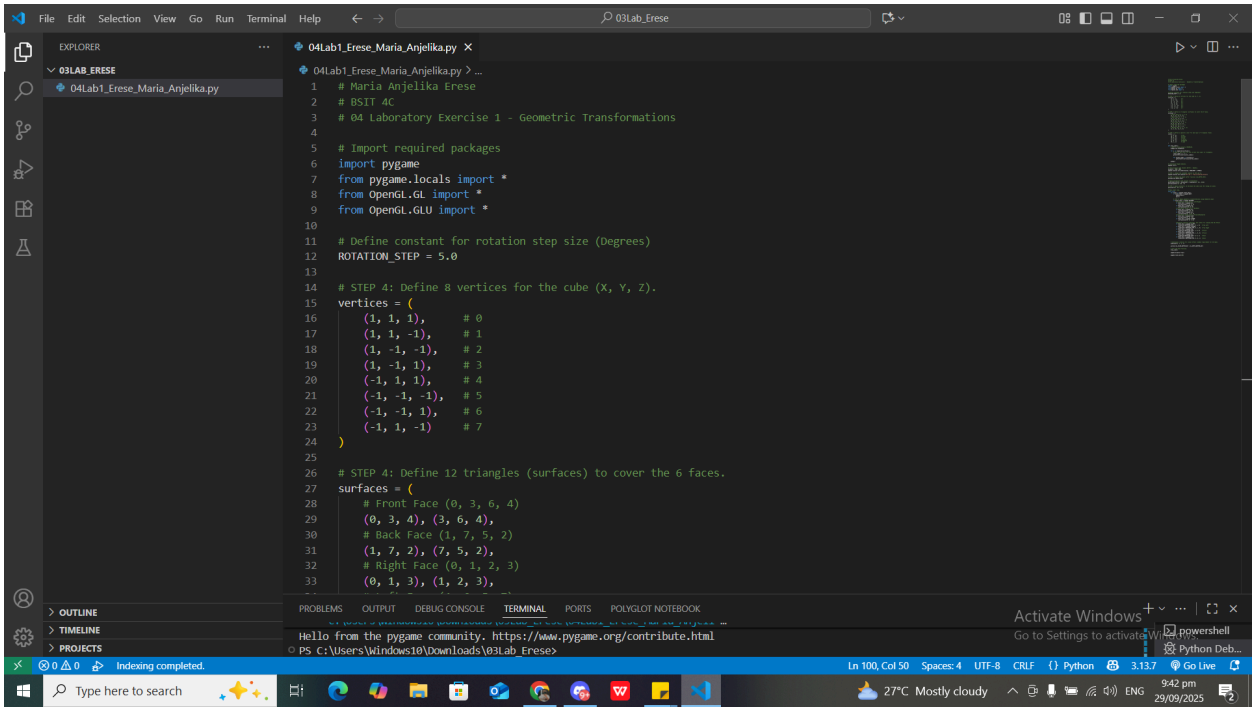
3. Change the window's caption to "04 Lab 1".



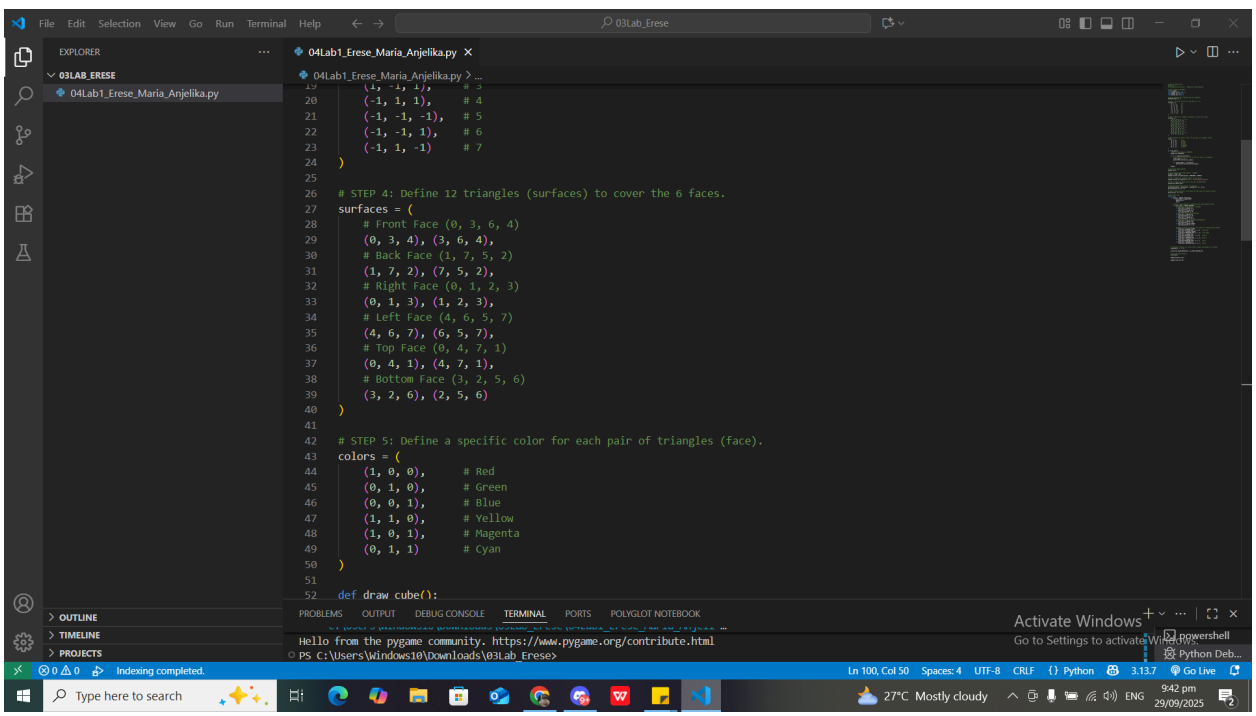
4. Redraw your cube using 12 triangles. In your draw_cube() method, replace GL_LINES with GL_TRIANGLES, then edit the glVertex3f() statements based on the following table and figure. For every set of three (3) vertices, you can form a triangle. For example, three (3) glVertex3f() statements can form a triangle, as illustrated in the sample output below.
5. Set a specific color for each pair of triangles using glColor3f(). The glColor3f() function has three (3) parameters for setting the RGB value. For example, glColor3f(0,1,0) draws in full-intensity green. Each glColor3f() statement is placed before every pair of triangles.
6. The glScalef() function multiplies the current matrix by a general scaling matrix. It has three (3) parameters for specifying the scale factors along the x, y, and z axes, respectively. Decrease the size of your cube using this function.
7. To enable the function of updating the depth buffer, insert this statement before the gluPerspective statement: glEnable(GL_DEPTH_TEST)
8. Apply geometric transformations to your cube using your keyboard. In the sample code below, the cube moves to the left if you press A.

```
if event.type == pygame.KEYDOWN:
 if event.key == pygame.K_a:
 glTranslatef(-1,0,0)
```

All the screenshots of my code:



```python
# Maria Anjelika Erese
# BSIT 4C
# 04 Laboratory Exercise 1 - Geometric Transformations

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

# Define constant for rotation step size (Degrees)
ROTATION_STEP = 5.0

# STEP 4: Define 8 vertices for the cube (X, Y, Z).
vertices = (
    (1, 1, 1),      # 0
    (1, 1, -1),     # 1
    (1, -1, -1),    # 2
    (1, -1, 1),     # 3
    (-1, 1, 1),     # 4
    (-1, -1, -1),   # 5
    (-1, -1, 1),    # 6
    (-1, 1, -1)     # 7
)

# STEP 4: Define 12 triangles (surfaces) to cover the 6 faces.
surfaces = (
    # Front Face (0, 3, 6, 4)
    (0, 3, 4), (3, 6, 4),
    # Back Face (1, 7, 5, 2)
    (1, 7, 2), (7, 5, 2),
    # Right Face (0, 1, 2, 3)
    (0, 1, 3), (1, 2, 3),
```



```python
    (-1, 1, 1),     # 4
    (-1, -1, -1),   # 5
    (-1, -1, 1),    # 6
    (-1, 1, -1)     # 7
)

# STEP 4: Define 12 triangles (surfaces) to cover the 6 faces.
surfaces = (
    # Front Face (0, 3, 6, 4)
    (0, 3, 4), (3, 6, 4),
    # Back Face (1, 7, 5, 2)
    (1, 7, 2), (7, 5, 2),
    # Right Face (0, 1, 2, 3)
    (0, 1, 3), (1, 2, 3),
    # Left Face (4, 6, 5, 7)
    (4, 6, 7), (6, 5, 7),
    # Top Face (0, 4, 7, 1)
    (0, 4, 1), (4, 7, 1),
    # Bottom Face (3, 2, 5, 6)
    (3, 2, 6), (2, 5, 6)
)

# STEP 5: Define a specific color for each pair of triangles (face).
colors = (
    (1, 0, 0),      # Red
    (0, 1, 0),      # Green
    (0, 0, 1),      # Blue
    (1, 1, 0),      # Yellow
    (1, 0, 1),      # Magenta
    (0, 1, 1)       # Cyan
)

def draw_cube():
```

```python
    (1, 0, 1),      # Magenta
    (0, 1, 1)       # Cyan
)

def draw_cube():
    # STEP 4: Draw using GL_TRIANGLES.
    glBegin(GL_TRIANGLES)

    for i in range(len(surfaces)):
        # STEP 5: Set color for the current face (pair of triangles).
        color_index = i // 2
        glColor3fv(colors[color_index])

        for vertex_index in surfaces[i]:
            glVertex3fv(vertices[vertex_index])

    glEnd()

# Initialize Pygame modules.
pygame.init()

# Setup display mode (Double Buffer | OpenGL).
display = (800, 600)
pygame.display.set_mode(display, DOUBLEBUF | OPENGL)

# STEP 3: Change the window's caption to "04 Lab 1".
pygame.display.set_caption("04 Lab 1 - Maria Anjelika Erese")

# STEP 7: Enable the depth buffer function (GL_DEPTH_TEST).
glEnable(GL_DEPTH_TEST)

# Setup perspective and initial Z translation.
gluPerspective(45, (display[0] / display[1]), 0.1, 50.0)
glTranslatef(0.0, 0.0, -5)
```

```
Hello from the pygame community. https://www.pygame.org/contribute.html
PS C:\Users\Windows10\Downloads\03Lab_Erese>
```

```python
glEnable(GL_DEPTH_TEST)

# Setup perspective and initial Z translation.
gluPerspective(45, (display[0] / display[1]), 0.1, 50.0)
glTranslatef(0.0, 0.0, -5)

# STEP 6: Apply glScalef() to decrease the cube size (0.5 along all axes).
glScalef(0.5, 0.5, 0.5)

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

        # STEP 8: Apply geometric transformations using keyboard input.
        if event.type == pygame.KEYDOWN:
            # Translation along X-axis (Left/Right)
            if event.key == pygame.K_a:
                glTranslatef(-1.0, 0, 0)
            if event.key == pygame.K_d:
                glTranslatef(1.0, 0, 0)
            # Translation along Y-axis (Up/Down)
            if event.key == pygame.K_w:
                glTranslatef(0, 1.0, 0)
            if event.key == pygame.K_s:
                glTranslatef(0, -1.0, 0)
            # Translation along Z-axis (Forward/Backward)
            if event.key == pygame.K_UP:
                glTranslatef(0, 0, 1.0)
            if event.key == pygame.K_DOWN:
                glTranslatef(0, 0, -1.0)
```

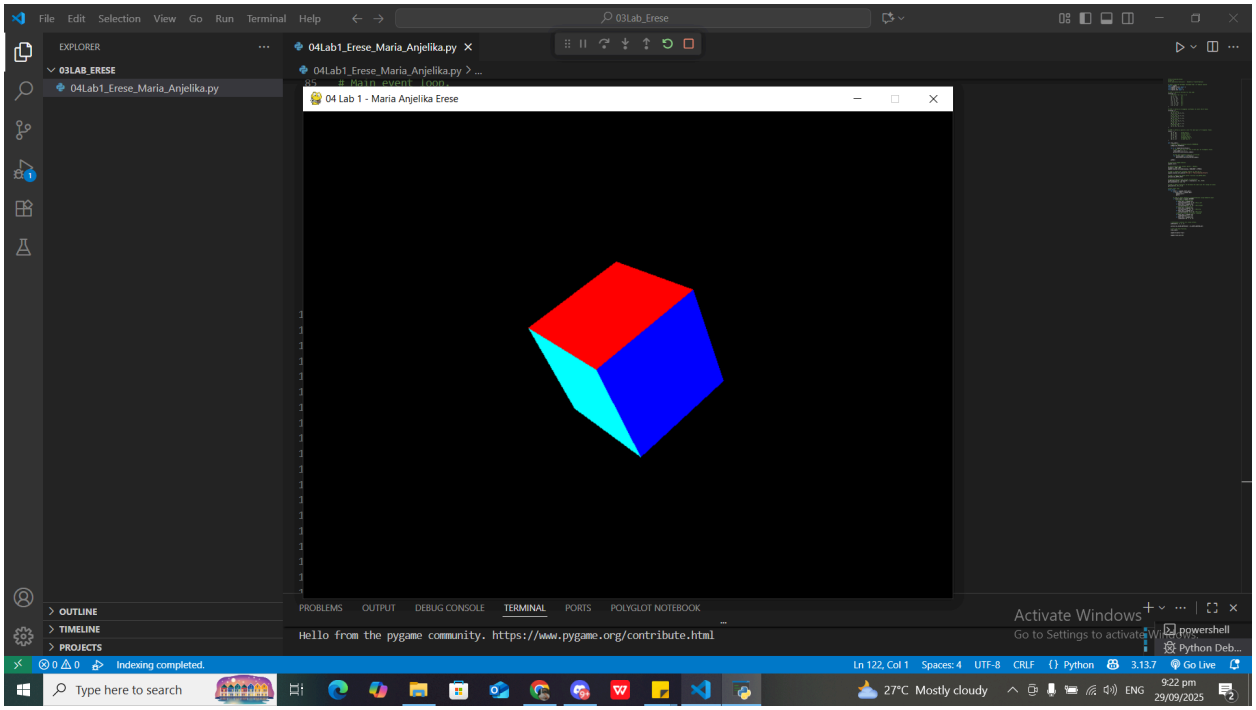
```
Hello from the pygame community. https://www.pygame.org/contribute.html
PS C:\Users\Windows10\Downloads\03Lab_Erese>
```

```python
            if event.key == pygame.K_a:
                glTranslatef(-1.0, 0, 0)
            if event.key == pygame.K_d:
                glTranslatef(1.0, 0, 0)
            # Translation along Y-axis (Up/Down)
            if event.key == pygame.K_w:
                glTranslatef(0, 1.0, 0)
            if event.key == pygame.K_s:
                glTranslatef(0, -1.0, 0)
            # Translation along Z-axis (Forward/Backward)
            if event.key == pygame.K_UP:
                glTranslatef(0, 0, 1.0)
            if event.key == pygame.K_DOWN:
                glTranslatef(0, 0, -1.0)

            # Rotation controls (optional, but useful for viewing the 3D effect)
            if event.key == pygame.K_LEFT:
                glRotatef(ROTATION_STEP, 0, 1, 0)   # Yaw Left
            if event.key == pygame.K_RIGHT:
                glRotatef(-ROTATION_STEP, 0, 1, 0)  # Yaw Right
            if event.key == pygame.K_q:
                glRotatef(ROTATION_STEP, 1, 0, 0)   # Pitch
            if event.key == pygame.K_e:
                glRotatef(-ROTATION_STEP, 1, 0, 0)  # Pitch
            if event.key == pygame.K_z:
                glRotatef(ROTATION_STEP, 0, 0, 1)   # Roll
            if event.key == pygame.K_c:
                glRotatef(-ROTATION_STEP, 0, 0, 1)  # Roll


    # Continuous rotation for visual effect (common requirement for 3D labs).
    glRotatef(1, 1, 1, 1)

```

```
Hello from the pygame community. https://www.pygame.org/contribute.html
PS C:\Users\Windows10\Downloads\03Lab_Erese>
```

```python
            if event.key == pygame.K_LEFT:
                glRotatef(ROTATION_STEP, 0, 1, 0)   # Yaw Left
            if event.key == pygame.K_RIGHT:
                glRotatef(-ROTATION_STEP, 0, 1, 0)  # Yaw Right
            if event.key == pygame.K_q:
                glRotatef(ROTATION_STEP, 1, 0, 0)   # Pitch
            if event.key == pygame.K_e:
                glRotatef(-ROTATION_STEP, 1, 0, 0)  # Pitch
            if event.key == pygame.K_z:
                glRotatef(ROTATION_STEP, 0, 0, 1)   # Roll
            if event.key == pygame.K_c:
                glRotatef(-ROTATION_STEP, 0, 0, 1)  # Roll


    # Continuous rotation for visual effect (common requirement for 3D labs).
    glRotatef(1, 1, 1, 1)

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

    # Call the draw function.
    draw_cube()

    pygame.display.flip()

    pygame.time.wait(10)

```

```
Hello from the pygame community. https://www.pygame.org/contribute.html
PS C:\Users\Windows10\Downloads\03Lab_Erese>
```



9. Show/submit your code and output to your instructor.
10. Save a copy of your code for future use.