Computer Graphics Lab 6

Camera View

Setting Viewing Transformations

1. Download and run program main.py:
   https://drive.google.com/drive/folders/15iDEjFYIoTlkSG2l4GyGFPO7PYjLckss?usp=drive_link
   TA Checking

2. Create a Python file named “Camera.py” that defines a Camera class with the following functionalities:
   2.1) **Initialization:**
   - The camera should have attributes to store its position (eye), orientation (up, right, forward), and a point it's looking at (look).
   - Initialize the camera at the origin (0, 0, 0) looking down the positive z-axis.

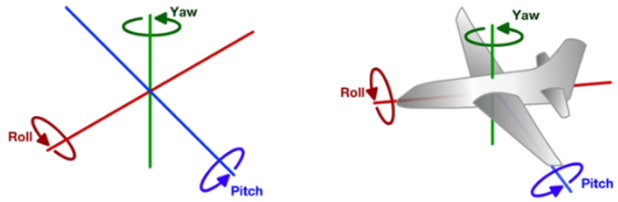
   2.2) **Movement:**
   - Implement methods to move the camera forward, backward, left, and right by arrow key press.
   - These movements should be relative to the camera's current orientation.

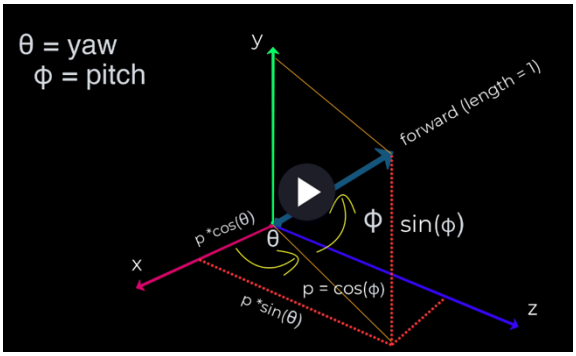
   2.3) **View Transformation:**
   - Use the gluLookAt() function from OpenGL.GLU to define the viewing transformation based on the camera's position (eye), the point it's looking at (look), and the up vector (up).

   2.4) **Integration with main.py:**
   - mport your Camera class into main.py.
   - Create an instance of the camera.
   - Apply the camera's view transformation in the display() function before drawing any objects.

   TA Checking

3. **Enhancing Camera Controls with Mouse-Based Yaw and Pitch**





You've implemented a basic camera class that allows for movement using keyboard controls. Now, let's make the camera even more interactive by adding mouse-controlled yaw and pitch. This will enable users to look around the 3D scene more naturally.

Modify your existing camera.py to include the following functionalities:

1. **Mouse Input:** Capture mouse movement using pygame.mouse.get_pos() and calculate the change in mouse position since the last frame.
2. **Yaw and Pitch:**
    - Implement yaw (horizontal rotation) and pitch (vertical rotation) based on the mouse movement.
    - **Hint:** You can use the change in mouse x-position to control yaw and the change in mouse y-position to control pitch.
    - **Note:** Be mindful of potential issues like gimbal lock when implementing pitch. You might want to limit the pitch to avoid looking straight up or down.
3. **Camera Orientation:** Update the camera's orientation vectors (forward, right, up) based on the calculated yaw and pitch.
    - **Hint:** You might need to use trigonometric functions (sine, cosine) to calculate the new orientation vectors.
4. **View Transformation:** Ensure that the gluLookAt() function in your update() method correctly uses the updated camera orientation vectors.

TA Checking