Cusson Laohapatanawong 66010988

1) What does PyGame do in this project, and what does PyOpenGL do? Why do we need Pillow (PIL) here instead of loading the image directly with OpenGL?

- PyGame manages the application framework, including window creation, event loop handling for inputs like orbiting, and frame timing. PyOpenGL provides the Python bindings for the rendering pipeline, enabling the script to communicate with the GPU via fixed-function calls like glBegin() and glEnd(). Pillow (PIL) is necessary because OpenGL cannot natively decode image formats like PNG or JPG; it is used in load_texture() to convert images into raw pixel bytes for GPU upload.

2) Explain the difference between GL_PROJECTION and GL_MODELVIEW. Where does gluPerspective() belong, and why?

- The GL_PROJECTION matrix defines the camera's lens properties and viewing volume, while GL_MODELVIEW handles the placement of objects and the camera in the world. gluPerspective() belongs in the GL_PROJECTION state because it mathematically establishes the perspective frustum used to project 3D coordinates onto a 2D screen. In the code, set_projection() initializes this lens, and apply_camera() modifies the MODELVIEW matrix to position the eye relative to the target.

3) In load_texture(), explain the purpose of: glTexImage2D, GL_TEXTURE_WRAP_S/T = GL_REPEAT, and glTexParameteri(... GL_LINEAR). Why do we flip the image vertically (Image.FLIP_TOP_BOTTOM)?

- In load_texture(), glTexImage2D transfers decoded pixel data to the GPU, while GL_TEXTURE_WRAP_S/T = GL_REPEAT allows textures to tile when UV coordinates exceed 1.0. glTexParameteri with GL_LINEAR ensures smooth filtering during scaling to prevent pixelation. We flip the image using Image.FLIP_TOP_BOTTOM because OpenGL expects the texture origin (0,0) at the bottom-left, whereas standard image formats store data from the top-left down.

4) What does GL_LIGHT0 contribute to the scene? Explain why we call glLightfv(GL_LIGHT0, GL_POSITION, light_pos) every frame (not only once).

- GL_LIGHT0 contributes the primary illumination, enabling ambient, diffuse, and specular reflections on the geometry. We call glLightfv(GL_LIGHT0, GL_POSITION, light_pos) every frame because light positions are transformed by the current MODELVIEW matrix; updating it ensures the light moves correctly relative to the camera as the user orbits. This allows specular highlights to shift realistically based on the viewing angle.

5) Why does enabling GL_CULL_FACE often cause a single-quad floor to disappear at some camera angles? How does the provided code prevent that?

- Enabling GL_CULL_FACE often causes a single-quad floor to disappear because it hides polygons based on their vertex winding order to optimize performance. If the camera views the floor from an angle where the winding appears "back-facing," OpenGL culls the quad. The provided code prevents this by calling glDisable(GL_CULL_FACE) within the drawing functions, forcing the quad to render regardless of the viewing angle.

6) Explain why transparent objects are typically drawn after opaque objects. What does glDepthMask(GL_FALSE) do, and why is it used for the transparent sphere?

- Transparent objects are drawn after opaque ones so they can blend their colors with the background already rendered in the frame. glDepthMask(GL_FALSE) is used for the transparent sphere to stop it from writing to the depth buffer, ensuring it doesn't "block" other objects behind it. Once the sphere is drawn, glDepthMask(GL_TRUE) is called to restore normal depth-writing behavior for the rest of the scene.