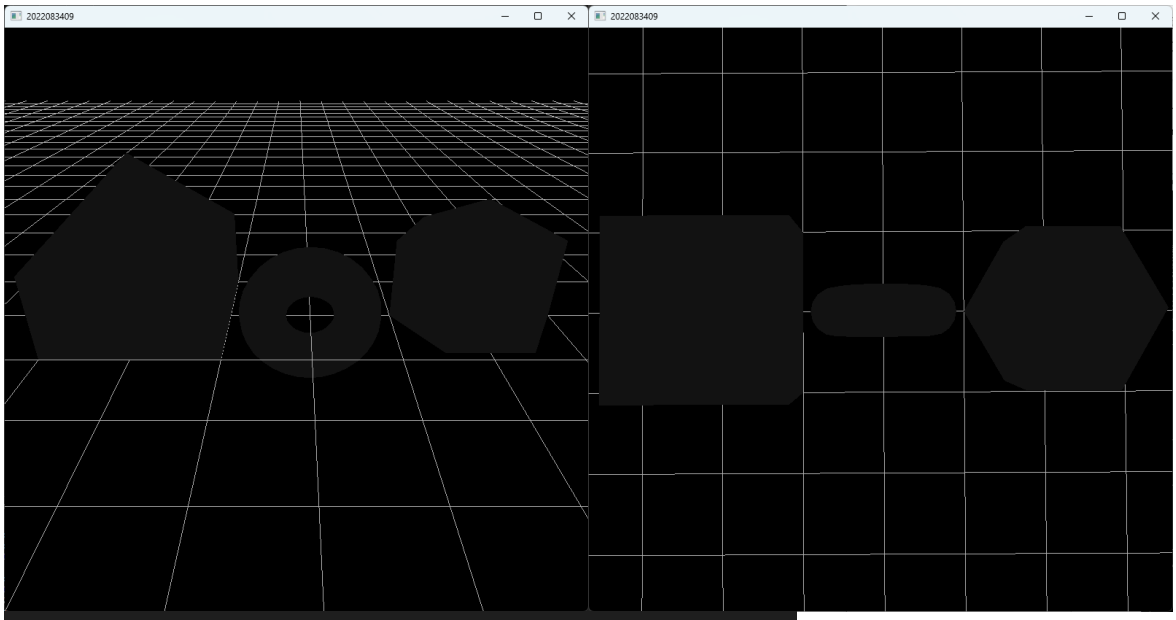
Computer Graphics Project2 Report – 2022083409 김승관

Implementation

1. objloader.py
   A. load(self, path)
      .obj 파일을 열어 vertex, normals, face 정보를 파싱합니다. 각 줄의 접두사(v, vn, f)를 기반으로 데이터를 분리하여 내부 리스트(positions, normals, faces)에 저장합니다. 이후 triangulation 단계에서 사용할 원본 데이터를 준비합니다.
   B. get_or_add_vertex(self, v_idx, n_idx)
      하나의 정점 정보를 (위치 인덱스, 법선 인덱스)로 받아 해당 정점이 이미 등록되어 있는지 확인한 뒤, 없으면 새롭게 vertices, normals 리스트에 추가하고 인덱스를 부여합니다. 이 과정을 통해 정점 중복을 방지하고, 효율적인 glDrawElements() 호출이 가능하게 합니다.
   C. triangulate(self)
      OBJ 포맷에서 다각형 면(f 요소)을 삼각형(face)으로 분해합니다. 삼각형은 그대로 유지하고, 4개 이상의 다각형은 "triangle fan" 알고리즘으로 쪼개서 indices 리스트에 저장합니다.
   D. printf_info(self)
      load된 .obj 파일의 파일 이름, 전체 face 개수, triangle, quad, N-gon의 개수를 출력합니다.
2. main.py
   A. drop_callback(window, paths)
      GLFW에서 제공하는 파일 드래그 앤 드롭 이벤트를 처리하는 함수입니다. 사용자가 .obj 파일을 드롭하면, OBJLoader를 통해 파일을 읽고 새로운 Mesh 인스턴스를 생성하여 meshes 리스트에 추가합니다. 각 mesh는 이전 mesh보다 x축 방향으로 +2 유닛만큼 이동된 위치에 배치됩니다.
   B. class Mesh
      i. __init__(self, vertices, normal, indices)
         주어진 정점/법선/인덱스 정보를 기반으로 interleaved vertex array를 구성합니다.
         VAO, VBO, EBO를 생성하고 GPU에 데이터를 업로드합니다.
      ii. draw(self, shader, VP)
         Mesh 인스턴스를 실제로 화면에 렌더링하는 함수입니다. 모델 변환행렬(M)과 뷰-투영행렬(VP)을 조합해 MVP를 계산하고, 셰이더에 전달합니다. 또한 material_color, view_pos 등의 uniform 변수도 설정합니다.

이후 VAO를 바인딩하고, glDrawElements()를 통해 실제 그리기 명령을 실행합니다. Phong 조명 효과를 구현하기 위해 법선과 뷰 벡터를 포함한 계산이 fragment shader에서 이뤄지도록 설정됩니다.




```
(cg-course) C:\Users\kimse\University Classes\Computer Graphics\project2>python main.py
OBJ file name: house.obj
Total Faces: 10
Triangles: 4, Quads: 6, N-gons: 0
OBJ file name: donut.obj
Total Faces: 1024
Triangles: 1024, Quads: 0, N-gons: 0
OBJ file name: ngon_pyramid.obj
Total Faces: 19
Triangles: 18, Quads: 0, N-gons: 1
```