컴퓨터 그래픽스 프로젝트2. OpenGL 뷰어

컴퓨터소프트웨어학과 2021027329 박현우

0. 설계

코드가 길고 향후 해당 프로젝트를 활용하기에 모듈화, main.py 이외에 의존성 최소화하도록 설계

main.py - 각 .py 임포트하고 메인루프에서 전체적인 흐름 담당

camera.py - 기존 그대로

input_callback.py - 모듈 간 의존성 줄이기 위해 클래스 구조로 변경

vao.py - 기존 프로젝트 코드에 prepare_vao_mesh 메소드만 추가 구현, EBO 방식

shader.py - lab-lighting의 6-all-components-phong-avgnorm 재활용

obj_loader.py - obj 파일 load 및 parsing, triangulate나 vertex의 average값을 구하는 등 필요한 과정 수행 후 최종적으로 vao에 들어갈 vertex array와 index array return

1. 구현한 requirement

0. 기존 프로젝트 1에서 구현했던 카메라 동작 기능이나 기본적인 grid는 그대로사용했습니다.

1. Open an obj file by drag-and-drop to your obj viewer window

클래스로 구현된 input_callback.py에서 drop_callback이 호출되면 obj_loader.py의 load_obj 메소드를 호출합니다. 이때 여러개의 obj 파일을 한번에 드래그 앤 드롭할 수 있으니 for문에 따라 처리하게 했습니다.

2.. Read the obj file and display the mesh using only vertex positions, vertex normals, faces information

obj_loader.py의 load_obj와 parse_obj에서 이를 처리합니다. 단순히 read하는 걸 넘어서 display 하기 위해선 triangulate이나 averaging을 통한 vertex normal을 구하는 과정이 필요했습니다. 각각 triangulate_face()와 average_vertex_normal()에서 수행했습니다.

3. Print out the following information of the obj file to stdout (terminal) each time an obj file is loaded

load_obj()에서 이를 수행합니다.

4. Support for Opening Multiple OBJ Files:

main.py의 메인루프에서 이렇게 얻어진 mesh 정보들을 바탕으로 vao를 최초에 한번 그리고 이후 매 프레임 렌더링합니다.

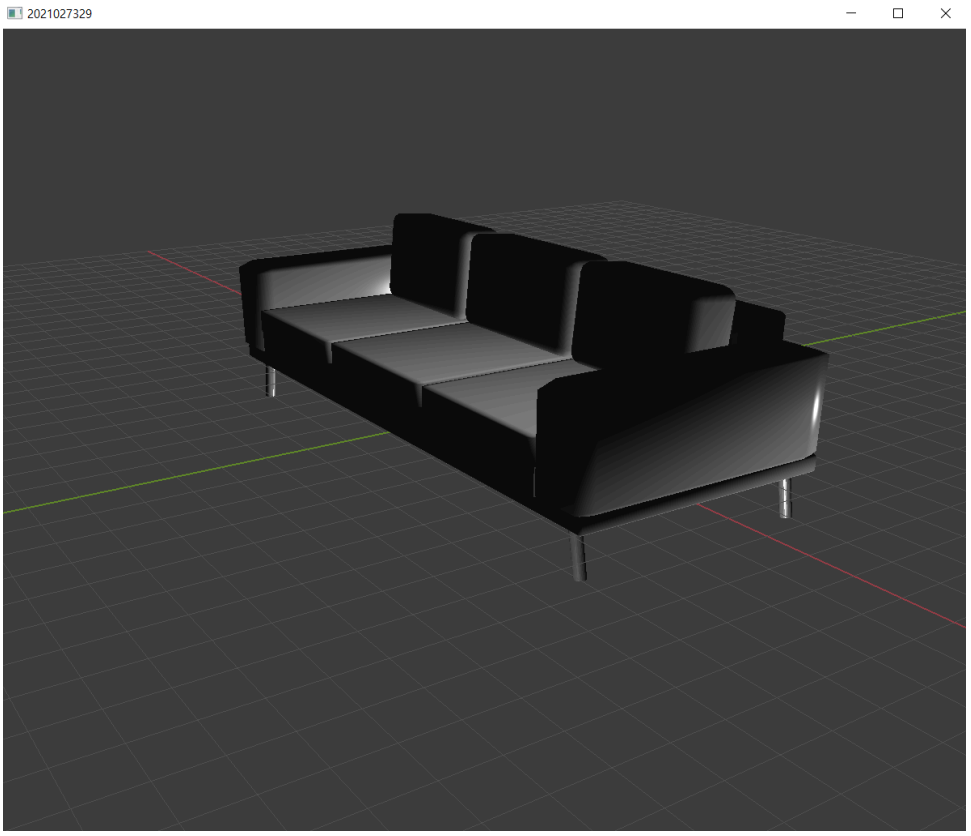
5. Support for Meshes with Polygons of Varying Vertex Counts:

triangulate을 통해 이를 구현했습니다.

6. Lighting & Etc

phong Illumination and phong shading 을 구현하기 위해 기존의 lab 코드를 재활용하는 식으로 구현했습니다.

2. 프로그램 실행 스크린샷



인터넷에서 구한 소파 파일 (블렌더 형식)을 블렌더에서 obj 파일로 변환

크기가 너무 커서 크기를 0.03배, y축에 대해 180’ 회전 한번 한 상태 (이 두개는 행렬로 직접 수행)