2019049716 안재국

Requirements

A. camera와 grid plane은 project1 코드를 그대로 가져왔습니다

B. drop_callback을 통해 윈도우에 파일을 끌었을 때 렌더링이 될 수 있도록 했습니다.

파일에서 v, vn, f로 시작하는 line만 읽어서 parsing을 했습니다. F로 시작하는 line에서 읽은 index 정보를 이용해서 vertices list를 채워줬습니다. 이때 렌더링할 vertex의 수를 세어서 전역 변수에 저장합니다.

Vertices list는 prepare_vao.py의 prepare_vao_obj 함수의 인자로 넘겨주었습니다.

Prepare_vao_obj 함수는 vertices list를 인자로 받아서, glm array로 type casting을 해준 후 vao를 만들어 리턴해 줍니다. callback함수는 이 vao를 받아서 전역 변수에 저장합니다.

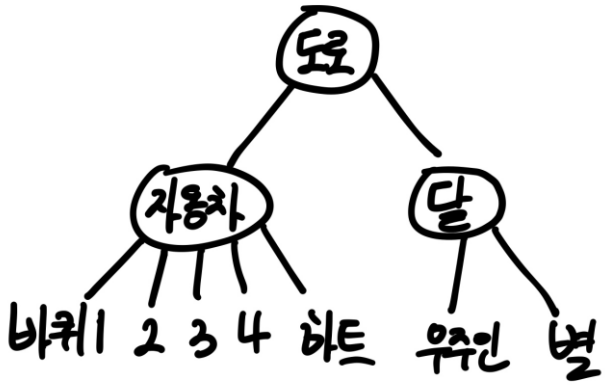
메인 함수에서는 이 전역 변수들을 이용해서 렌더링을 합니다.

단, single mesh mode가 True일 때만 해당 obj 파일을 렌더링합니다.

C. single mesh mode에서 'H' key를 누르면 animating hierarchical model rendering mode로 전환합니다.

최소 3개의 obj 파일을 다운받아야 한다고 했지만, 저는 8개의 파일을 import해서 계층구조로 연결했습니다.

Tree는 3 level tree가 되어야 하고, 리프 노드를 제외한 노드들은 최소 2개의 자식을 가져야 한다고 명시되어 있습니다.



저는 이런 식으로 계층 구조를 설정하고, 각 노드는 부모 노드에 대해 transform을 할 수 있도록 해주었습니다.

D. Lighting & Etc

Fragment shader로 phong shader를 사용해서 렌더링 했습니다.

이때 여러 개의 광원을 설정하라는 명세에 따라 기존의 광원에 새로운 광원을 하나 더 추가했습니다.

Ambient, Diffuse, Specular 값을 광원마다 각각 결정해서 물체의 색상 값으로 결정해주었습니다.

'Z' key를 눌렀을 때, wire 모드를 수행하기 위해서 while문 내에 다음과 같은 코드를 작성했습니다.

```
        # render in "wireframe mode"
        if wire_mode == True:
            glPolygonMode(GL_FRONT_AND_BACK, GL_LINE)
        else: glPolygonMode(GL_FRONT_AND_BACK, GL_FILL)
```

GL_LINE으로 mode를 설정하는 경우엔 wireframe mode로 렌더링 됩니다.

다시 'Z' key를 누르면 원래대로 돌아가야 하기 때문에 GL_FILL mode로 전환합니다.

Youtube 주소: https://youtube.com/shorts/s5nfuQRNkEM