컴퓨터 그래픽스

2019038513 박윤배

1. OBJLoader

두 가지 모드 모두에서 파일들을 불러오기 위해서 OBJLoader라는 클래스를 새로 정의 하였습니다. 해당 클래스에서는 포지션, 노말, 인덱스, face의 숫자 등등을 담는 배열들이 있습니다. 또한 파일의 경로를 인자로 받아 해당 파일을 파싱하여 배열에 저장해주는 함수(load_model), 파싱한 정보들을 가지고 vao를 만들어주는 함수가 있습니다(prepare_vao_obj).

먼저 load_model에서는 먼저 파일의 시작이 v혹은 vn일때 해당 값들을 파싱하여 순서대로 각각 배열에 넣어줍니다. 이는 버텍스, 노말의 실제 값들을 순서에 맞추어 배열에 넣어주는 작업입니다. 이후 다시 파일을 읽어 시작이 f인 경우 각각의 값들을 파싱하여 인덱스 배열에 넣어주게 됩니다. 이 때 우리가 필요한 값들은 v와 vn뿐 이기 때문에 각각 첫번째와 세번째 요소만을 담습니다. 이 때 값의 개수가 3개가 넘는 경우는 삼각형이 아닌 다각형 이기 때문에 해당 경우에 맞게 삼각형으로 나누어 배열에 추가해줍니다. (12345 -> 123, 134, 145)

다음으로 prepare_vao_obj에서는 제일 먼저 인덱스로 저장되어 있던 값들을 실제 값으로 바꾸어 주고 vertex1-norm1-vertex2-norm2-… 이 순서대로 오도록 새로운 배열을 만들어 줍니다. 이후 np의 사이즈 함수를 사용하기 위해 해당 배열을 float32의 형식을 가진 np.array로 변환해줍니다. 이후 location0은 벌텍스, location1은 노말로 오도록 VAO를 구성하고 반환해 줍니다.

2. Node

계층을 설정해주기 위해 이전에 실습에서 배운 Node class를 사용해 주었습니다. 이를 통해 부모의 움직임에 따라 자식들도 움직일 수 있게 하였습니다.

3. Key_Callback

H버튼을 누르면 file_Mode라는 전역변수가 0이 되도록 합니다. 해당 값이 1이면 single_mesh 모드로 작동하고 0이면 계층 모델이 랜더링 되도록 하였습니다. 해당 기능은 메인 함수의 반복문 안에 조건문으로 구현하였습니다. 값이 0 이면 미리 불러둔 계층 모델을 그리도록 하였고 1이라면 불러온 단일 파일을 랜더링 하도록 하였습니다.

4. File_callback

파일을 화면에 드래그&드랍을 하게 되면 해당 파일의 주소를 반환하게 되는데 이때 이 주소에 해당하는 파일을 OBJLoader를 통해 열고 vao를 생성하도록 하였습니다. 이때 불러온 오브젝트와 vao는 전역변수인 obj_object, obj_vao에 저장됩니다.

5. Draw_node

해당 함수는 계층 구조로 이루어진 물체를 그릴 때 같은 함수를 여러번 사용하여야 하는데 이때 중복적인 코드 작성을 방지하기 위해 만든 함수입니다. MVP에 M을 노드의 로컬 움직임, 부모의 움직임(월드) 을 곱한 값으로 사용한다. 이를 통해 부모의 움직임이 반영된 MVP를 유니폼으로 넘기게 된다. 또한 유니폼으로 M, color_vec, view_pos를 쉐이더로 넘긴다. 이후 받은 버텍스의 수를 이용해 glDrawArray를 이용해 폴리곤들을 그린다.

6. Main()

먼저 dropcallback의 함수를 등록해준다.

각각의 오브젝트들도 미리 로드해 두는데 단일모드에서 사용될 파일은 아직 정해지지 않았으므로 객체만 생성해두고 계층 모드에 사용될 오브젝트들은 미리 생성하고 미리 vao까지 만들어 둔다. 이후 계층모델링의 계층을 설정해주기 위해 Node를 설정해준다. 계층 정보는 밑의 그림자료로 설명하겠습니다.

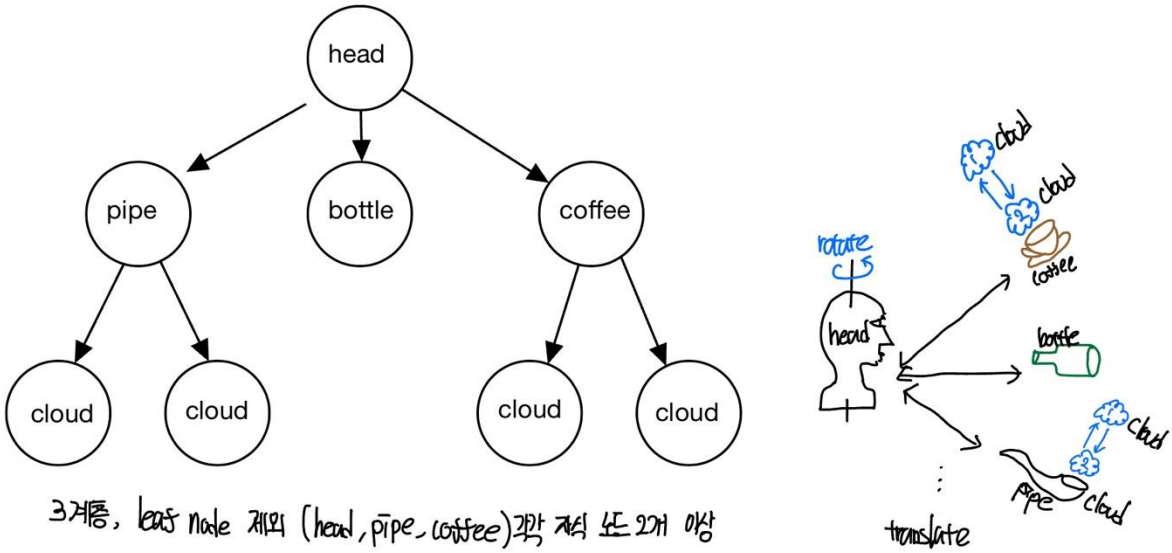
랜더링하는 반복문 안에서는 먼저 프레임은 기본적으로 그려주고 solid_mode 값을 확인하여 1이라면 solid로 0이라면 wireframe으로 그려준다. (glPolugonMode 함수를 사용)
다음으론 file_mode의 값을 확인 한 후 1이라면 단일 모드로 아니라면 미리 불러둔 계층 모델들을 랜더링한다. 이때 시간에 따른 움직임을 반영해주기 위해 node의 set_transform 함수를 이용해 원하는 움직임을 입력해준다. 마지막으로는 가장 위에 있는 head의 update_tree_global_transform을 실행시켜 각각 계층의 움직임을 재귀적으로 leaf node까지 적용시킨다.

7. 빛

쉐이더에서 Phong shading, Phong illumination을 이용해 빛을 구현 했습니다. 광원의 위치는 각각 (20,20,20), (-20,-20,-20) 입니다.

주의사항 : 카메라의 시작부분이 (0,0,0)이기 때문에 뒤로 많이 줌 아웃을 해야합니다.

8. 계층 구조



9. 링크

https://youtube.com/shorts/HIbKKuYOwJo?feature=share