**Class Assignment 2 Report**

2017030446 한상엽

Single Mesh Rendering Mode

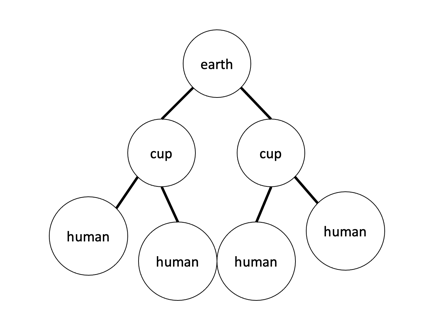
glfwSetDropCallback 을 이용했다. drop\_callback함수 내부에서 handle\_dropped\_file 함수를 호출하고 인자로 절대경로를 넘겨주었다. handle\_dropped\_file함수에서는 vertex, index, vertex normal, vertex normal index를 각각 저장할 numpy 배열을 전역변수로 가져와 초기화 시켰다. 면이 몇 개의 점으로 되어있는지 확인할 변수도 초기화 시켰다. 이후 경로를 통해 파일을 열고 string split을 이용해서 경로의 마지막에 있는 파일명을 출력해준다. 파일의 내용을 한 줄씩 읽은 뒤 공백으로 split해서 각각의 요소들을 적절한 곳에 저장했다. 각 면이 몇 개의 점으로 구성되어 있는지 각각 세어주었다. 이후 그 정보를 전부 출력해주고 읽기 편하게 모든 배열을 3열로 맞췄다. 좌표를 담은 배열의 경우 모든 원소의 크기를 np.float32 타입으로 맞춰서 4바이트의 크기를 갖게 했다. 초기 normal 은 파일에서 주어진 대로 그려야하기 때문에 narr 에 파일에서 읽은 값을 넣어주고 make\_farr 함수를 통해 farr 에는 normal vector 한 줄, 해당하는 vertex의 좌표값 한 줄, 이렇게 번갈아 가면서 넣어 나중에 glDrawArrays로 한 번에 그려준다. make\_nnarr 함수는 extra credit에서 Forced Smooth Shading을 구현하기 위함이다.

텍스트이(가) 표시된 사진

자동 생성된 설명 1) 출력 결과물

Animating Hierarchical Model Redering Mode

프로그램 실행 중에 “H”를 누르게 되면 Hierarchical Model Rendering Mode가 활성화된다. 최초로 h가 눌렸을 때만 init\_models함수가 호출된다. earth.obj, cup.obj, human.obj 이렇게 세 파일을 직접 경로를 구해 handle\_dropped\_file 함수의 인자로 넘긴다. 경로를 구할 때는 os.getcwd()로 현재 파일의 위치를 구하고 그 뒤에 os.path.join으로 현재 파일의 위치에 파일 명을 붙여 경로를 구했다. 그 이후 각 파일의 farr을 eartharr, cuparr, humanarr에 저장해준다. 이후 render에서 “H”가 눌린 경우에만 각 node의 transformation을 설정해준다. 각 오브젝트의 크기를 glScalef로 설정하고 farr을 그려야할 오브젝트의 배열로 설정해준 뒤 drawObject함수를 호출해서 그려주었다. 각 노드는 DFS로 서치하고 서치할 때 glPushMatrix로 그 노드의 transformation을 저장했다. 해당 노드를 다 탐색하면 glPopMatrix를 호출하여 그 노드의 transformation을 지웠다. 이런 방식으로 아래의 구조를 만들었다. 처음에 “H”를 누를 때 init\_models가 호출되면서 blocking된다. MacBook Pro 13inch 2020년형(Intel) 기준 2분정도 걸렸다. 이 이유는 extra credit에서 설명하겠다.

검은색, 어두운이(가) 표시된 사진

자동 생성된 설명검은색, 어두운이(가) 표시된 사진

자동 생성된 설명 Hierarchical models of objects / earth.obj / human.obj / cup.obj

drag and drop video

<https://youtu.be/cnsbh8c5i8k>

hierarchical model video

<https://youtube.com/shorts/HPTxwoUrEJ8?feature=share>

Light Source and Switching Polygon Mode

광원의 개수는 총 4개이며 GL\_LIGHT0부터 GL\_LIGHT3까지 사용했다. 각각 흰색, 노란색, 보라색, 청록색이다. 위치는 순서대로 (3,4,5), (-3,4,5), (-3,4,-5), (3,4,-5)이다. 사방에서 빛이 비춰진다. 모든 광원은 점광원이다.

프로그램 실행중에 “Z”를 누르면 기본인 solid mode에서 wireframe 모드로 바뀌게 된다. glPolygonMode의 인자를 GL\_LINE과 GL\_FILL을 번갈아가면서 넣어서 구현했다.

Extra Credit

* Changing Shading

Forced smooth Shading을 구현하기 위해 각 vertex의 normal을 새로 계산해주었다. 먼저 해당 면에 해당하는 점의 좌표로 두 벡터를 만든 뒤 둘을 외적해준다. 외적한 결과가 해당 면의 normal vector이므로 모든 면의 normal vector를 구해준다. 이후 모든 vertex에 대해서 인접한 면의 normal vector를 전부 더해 평균내야한다. nnarr는 각 점의 인접한 면의 normal vector의 합을 저장한다. 이후 그 크기로 나누어 단위 벡터로 만들면 각 점의 normal vector를 계산한 결과가 된다.

handle\_dropped\_file에서 make\_nnarr을 호출해 계산해주기 때문에 크기가 큰 오브젝트 파일은 make\_nnarr에서 시간을 많이 잡아먹게 된다. index 값을 저장하는 배열에서 각 값을 읽어 그 값을 다시 index로 찾아가 계산을 해줘야하기 때문에 메모리 접근이 많이 일어나 여기서 시간이 제일 많이 소모된다. 하지만 오래 걸리는 것이지 작동을 안하는 것은 아니다.

* Triangulation Algorithm

면을 이루는 점의 개수가 4개가 넘어가게 되면 모든 면을 GL\_TRIANGLES로 그릴 수 없다. 그래서 전부 삼각형으로 만들어주는 알고리즘이 필요하다. 파일에서 면에 대한 정보를 읽을 때 사각형 이상이면 기준 점을 첫번째로 나오는 그 점으로 잡고 두 점씩 순서대로 삼각형으로 만들었다. 예를 들어 “f 0 1 2 3 4” 이면 [0 1 2] [0 2 3] [0 3 4] 이런 식으로 만들었다. 이후 이 정보를 그대로 iarr(face index array)와 vniarr(vertex normal index array)에 넣어줬다.