컴퓨터그래픽스 과제4 레포트

19011493 김준우

1. 접근 방법/아이디어

1) .obj에서 데이터 읽어오기

- ifstream을 이용해 .obj 파일을 처음부터 끝까지 한 번 읽고,
- v로 시작하는 행의 개수와 f로 시작하는 행의 개수를 카운팅
- 이후 v행 개수만큼 크기의 verticeArray, normalArrayPhong 배열과
  - verticeArray 배열은 모든 정점들의 좌표를 저장하기 위해 사용
  - normalArrayPhong은 이후 phong shading에서 정점이 속해있는 평면들의 법선 벡터를 누적하기 위해 사용
- f행 개수의 3배 크기의 normalArrayFlat, Vertices 배열 동적 할당
  - normalArrayFlat은 이후 flat shading에서 각 정점들이 속한 평면의 법선 벡터를 기억하기 위해 사용
  - Vertices는 정점들의 position, color, normal 등을 저장하기 위해 사용
- 이후 .obj 파일을 처음부터 다시 읽어 Vertices에 정점 좌표 저장

2) 오브젝트 중심점 찾아 이동 / scaling 하기

- 오브젝트 중심점
  - .obj에서 데이터를 읽어오면서 정점 좌표 X Y Z들의 최솟값과 최댓값을 각각 저장 (총 6개)
  - XYZ의 최솟값, 최댓값들의 중간 지점 좌표를 오브젝트의 중심점으로 취급, 그에 맞게 이동
- scaling
  - XYZ의 최댓값에서 최솟값을 빼 X축 방향, Y축 방향, Z축 방향의 폭, 너비, 높이 계산
  - 폭, 너비, 높이 중 최댓값을 오브젝트의 크기로 취급
  - XYZ좌표 -1부터 1까지의 좌표를 볼 수 있는 OpenGL 환경에 맞게 scaling

3) flat shading/phong shading 구현하기

- flat shading
  - flat shading에서 정점들의 normal vector는 곧 속해 있는 평면의 법선 벡터
    - 평면 위 두 벡터의 외적을 이용한 계산값을 normalArrayFlat에 저장해 두고 사용
- phong shading
  - phong shading에서 정점들의 normal vector는 속해 있는 모든 평면들의 법선 벡터의 평균
    - 한 정점이 속한 평면들의 모든 법선 벡터를 normalArrayPhong에 누적하고 마지막에 normalize하여 사용

4) 고려청자 색깔 입히기

- 

Figure 1 참고 이미지: 고려 청자

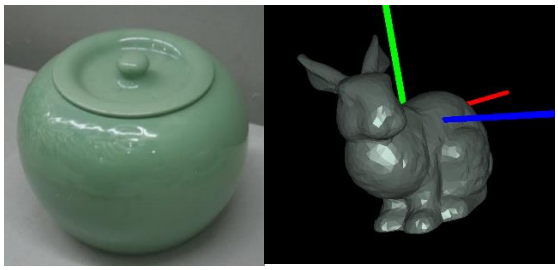
  - 기존의 조각 같은 .obj들의 개성을 살리기 위해 고려청자 특유의 비색을 넣어 좀 더 세련되고 우아한 느낌의 색감을 부여하고자 함 (vec4(0.79, 0.89, 0.84, 1))
  - 결과물은 마치 오래 묵어 조금 때 탄 옥색 요강같이 되었지만 그런대로 고급스러워 만족
- 

Figure 2 참고 이미지: 요강

2. 마주친 문제/해결방안

1) phong shading 법선 벡터 계산 시간 복잡도 최소화 O(n^2) -> O(n)
phong shading을 위해서는 정점을 둘러싼 모든 평면들의 법선 벡터를 더해야 했는데, 이를 .obj의 f행을 돌며 누적했다가는 이중 반복문이 되어 O(n^2)의 시간 복잡도를 가지게 된다.

   A. Vertices에 position 값을 저장하는 O(n) 반복문에서 누적 역시 한 번에 하여 이후 O(n^2)의 반복문을 다시 도는 일이 없도록 하였다.

2) flat shading <-> phong shading 전환 최적화
기존의 각 정점들의 normal vector를 입력하는 작업은 init()에서 이루어졌다.
하지만 init() 중 .obj 파일을 다시 읽어오고, 계산하는 등의 작업을 매번 shading을 전환한 할 때마다 반복할 필요는 없었다.

   A. 처음 한 번 init()이 호출되었을 때 flat shading, phong shading에 필요한 normal vector들을 모두 계산해두고, normal vector을 대입하는 부분은 init()로부터 따로 분리하여 shading을 전환할 때마다 그 부분만 다시 호출하였다.

3) 회전축 전환 과정에서의 어색함
마우스 클릭으로 오브젝트의 회전 방향을 바꿀 때 기존의 각도를 유지하고 회전 방향만 변하는 것이 아닌, 기존의 각도를 잃고 회전 방향이 바뀌었다.

   A. 매 프레임 오브젝트의 각도에다 g_Time이 곱하여진 수식을 단순 대입했기 때문에 생긴 일이었다. 수식에서 g_Time을 제거하고 매 프레임 일정한 크기를 곱하여 회전하게 함으로써 도중에 회전 방향이 바뀌더라도 누적된 각도가 유지되도록 하였다.