Computer Graphics Assignment 3

2019-18499 김준혁

1. 구현 내용

과제 2와 중복되는 내용이 있으며, 유지된 내용은 수정 없이 두었습니다.

primitives.py

CustomGroup을 상속받은 PointGroup, LineGroup, TriangleGroup, class와 TriangleGroup을 상속받은 PhongTriangleGroup, PhongWithTextureTriangleGroup, PhongWithTextureWithNormalTriangleGroup을 추가하였습니다. 각 그룹에 대해 shading 방법에 맞는 shader_program을 생성하여 이용하도록 해주었습니다.

- PointGroup : obj 파일의 점들의 집합 그룹
- LineGroup : obj 파일의 선들의 집합 그룹
- TriangleGroup : obj 파일의 면을 삼각형으로 표시한 것의 그룹
- PhongTriangleGroup : Phong illumination을 이용하여 표현하는 삼각형 그룹
- PhongWithTextureTriangleGroup : Phong에 텍스쳐를 같이 이용한 삼각형 그룹
- PhongWithTextureWithNormalTriangleGroup : Phong, 텍스쳐, Normal Mapping을 이용한 삼각형 그룹

PointSet, LineSet, TriangleSet 도형 클래스를 추가하였습니다. 이 중 TriangleSet은 obj의 각 점의 위치, 삼각형 구성, 색, normal, 텍스쳐 좌표, TBN을 구하기 위한 인접한 점들의 3차원 위치, 텍스쳐 위치 등 shader.py를 위한 변수들을 미리 구합니다.

- PointSet, LineSet은 obj 파일의 내용을 그대로 표현하는 데에 사용됩니다.
- TriangleSet은 obj 파일의 내용을 그대로 표현하기도 하며, rendering, shading을 위한 변수를 미리 구하여 가지고 있습니다.

main.py

obj 파일을 불러오는 load_object 함수를 추가하였습니다. 이 함수에서 미리 각 도형의 꼭짓점, 선, 3차원 위치의 삼각형 꼭짓점(v) list, 삼각형 텍스쳐 꼭짓점(vt) list, 삼각형 Normal 꼭짓점(vn) list, 시작 꼭짓점 인덱스를 반환합니다.

add_pointSet, add_lineSet, add_triangleSet을 함수를 추가하였습니다. 이들은 primitives.py에서 정의한 각 도형들을 추가해주는 함수입니다. add_triangleSet에서는 텍스쳐 파일(BaseColor, AO, Normal, Roughness, Specular) 이름을 도형 추가 시에 모두 전달합니다.

렌더링 방식 번호를 받아 해당하는 그룹 클래스를 반환하는 group_int_to_group 함수를 추가하였습니다. 렌더링 방식 번호는 바로 아래 문단에 표시하였습니다.

프로그램을 실행시킨 이후, 콘솔창을 통해 불러올 obj 파일, 텍스쳐 파일(순서대로 BaseColor, AO, Normal, Roughness, Specular)의 위치를 입력 받고, 마지막으로 렌더링 방식을 설정합니다(1 : wireframe, 2 : Phong w/o texture, 3 : Phong w texture, 4 : Phong w texture & normal mapping). 그러면 모델이 프로그램 화면으로 나타나게 됩니다.

render.py

camera_move 함수를 추가하여 키보드 및 마우스로 시점을 바꾸어주는 기능을 추가하였습니다.
이에 필요한 변수를 __init__에서 초기화하였으며, update 함수를 통해 호출됩니다.
불러온 obj 파일의 위치 또한 저장합니다.

update 함수에 Phong Shading을 이용하는 경우에 대해 빛의 위치, 색깔, intensity를 추가하여 shading에 이용하도록 하였습니다.

add_shape 함수에 Shading 방식에 따라 shader.py에 필요한 변수를 넣어서 도형을 추가하도록 수정하였습니다.

control.py

__init__에 컨트롤에 필요한 변수를 추가하였습니다.
- selected_point : 마우스 왼쪽 클릭으로 선택된 점을 저장합니다.
- mouse_move_camera : 마우스 오른쪽 클릭으로 카메라를 움직이는 상태를 저장합니다.
- dragging : 마우스 드래그 상태를 저장합니다.

on_key_press, on_key_release, on_mouse_press, on_mouse_release, on_mouse_drag 에 키를 추가하였습니다.

- W : 카메라 시점 기준 앞으로 이동
- S : 카메라 시점 기준 뒤로 이동
- A : 카메라 시점 기준 왼쪽으로 이동
- D : 카메라 시점 기준 오른쪽으로 이동
- LShift : 카메라 시점 기준 아래로 이동
- Space : 카메라 시점 기준 위로 이동
- 마우스 오른쪽 클릭, 드래그 : 카메라 시점(카메라 위치는 고정) 타겟 변화

(과제 2에서 만들고 유지했지만, 검증하지 않았으며, 문제가 남아있는 상태입니다.) exit_work, save_object 함수를 추가하여 프로그램에서 esc를 누르면 종료하며, 저장할 지를 확인하고 어떤 위치에 저장할 지를 콘솔창에 입력하여 각 도형을 모두 저장하도록 하였습니다. default는 불러온 파일과 동일합니다.

shader.py

Phong shading w/o textures 에 사용되는 vertex_source_phong, fragment_source_phong 변수를 추가했습니다. k_a(ambientStrength) = (0.1, 0.1, 0.1), k_d(diffuseStrength) = (0.5, 0.5, 0.5), k_s(specularStrength) = (0.8, 0.8, 0.8), n(shininess) = 8로 하여 구현하였습니다. vec3 대신에 float로도 쉽게 계산할 수 있는 변수의 경우, vec3 대신 float로 (해당 shading에서 k_a = 0.1, k_d = 0.5, k_s = 0.8) 표현하여 계산하였습니다.

Phong shading w/ textures 에 사용되는 vertex_source_phong_texture, fragment_source_phong_texture 변수를 추가했습니다. Normal mapping 텍스쳐를 제외한 나머지 4개의 텍스쳐 파일을 이용하여 k_a, k_d, k_s, n을 구성하며, n = 1 / a * x(shininess 파일 기준) + b 에서 a = 0.1, b = 0.1을 적용하였습니다.

Phong shading w/ textures & normal mapping 에 사용되는 vertex_source_phong_texture_normal, fragment_source_phong_texture_normal 변수를 Phong shading w/ textures에서 추가적으로 Normal Mapping 텍스쳐를 받아오고, primitives.py에 보이듯이, 인접한 두 점의 위치와 UV 값을 가져와 TBN 행렬을 구하고, normal map과 곱하여 normal을 구하고, 이를 이용하여 계산을 합니다.

2. 실행 방법

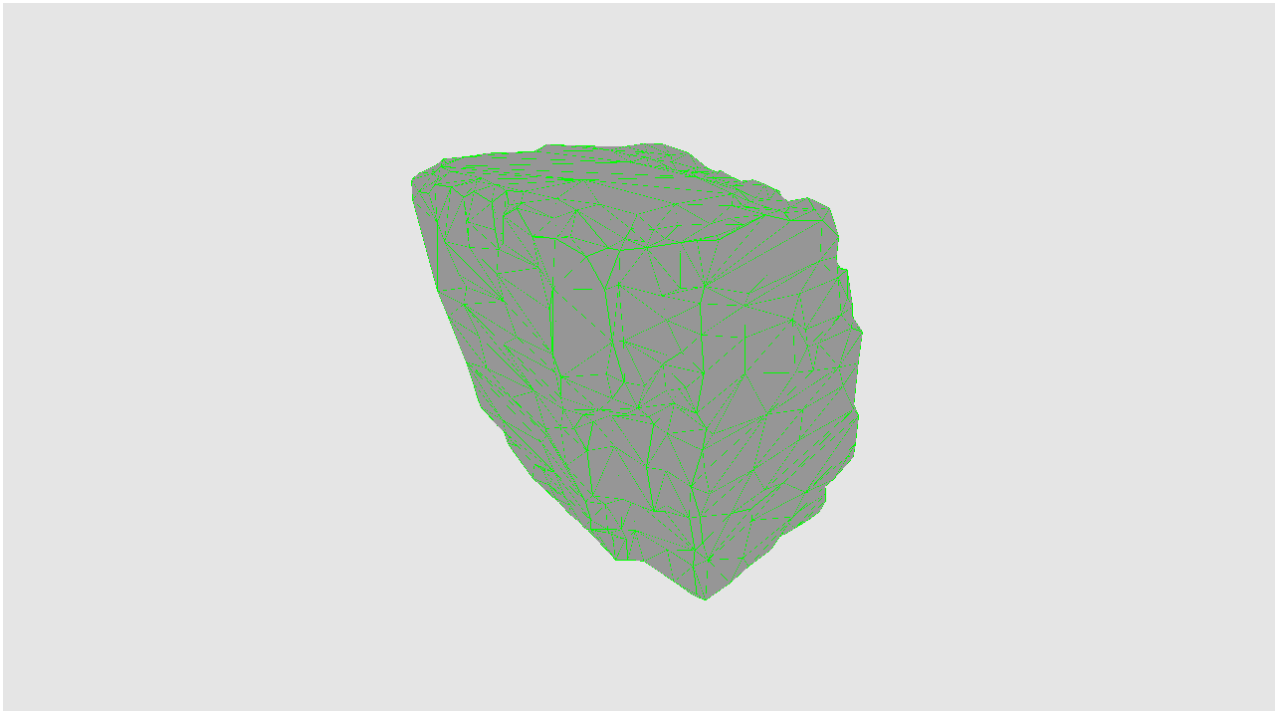
Base code와 같이 python3 main.py로 실행되며, 표시할 obj와 텍스쳐 파일(순서대로 BaseColor, AO, Normal, Roughness, Specular)의 위치를 콘솔창에 지정해주고, 렌더링 번호 (1 : wireframe, 2 : Phong w/o texture, 3 : Phong w texture, 4 : Phong w texture & normal mapping)를 지정해주어야 프로그램이 정상적으로 작동합니다 (default는 Free_rock의 obj, jpg 위치로 지정되어 있습니다)..

프로그램이 obj 파일을 불러온 이후에는 아래의 키를 이용해 편집이 가능합니다.
(구현 내용에 서술한 것과 동일합니다.)

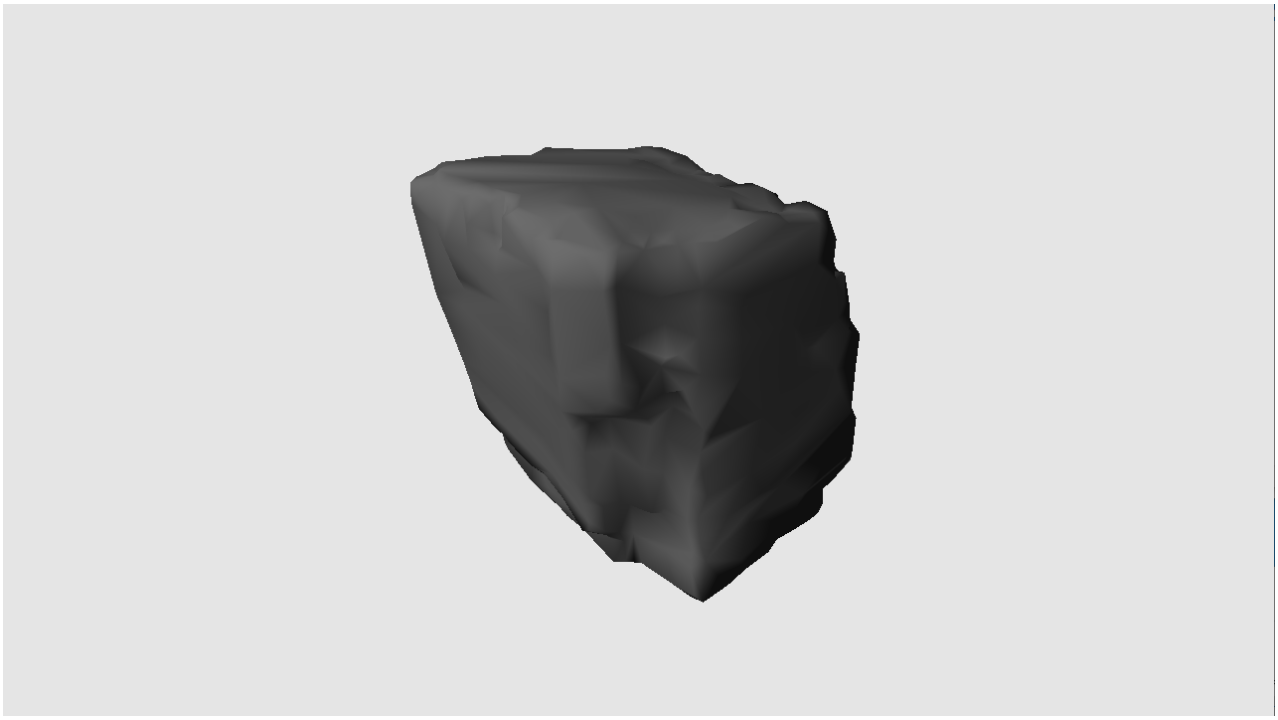
- W : 카메라 시점 기준 앞으로 이동
- S : 카메라 시점 기준 뒤로 이동
- A : 카메라 시점 기준 왼쪽으로 이동
- D : 카메라 시점 기준 오른쪽으로 이동
- LShift : 카메라 시점 기준 아래로 이동
- Space : 카메라 시점 기준 위로 이동
- 마우스 오른쪽 클릭, 드래그 : 카메라 시점(카메라 위치는 고정) 타겟 변화

(저장 기능은 과제2에서 만들어진 것을 유지는 했으나, 과제 3에 맞게 수정하지는 않아 정상적으로 동작하지 않습니다.) ESC 키를 눌러 종료를 할 수 있으며, 저장할 지 여부를 콘솔창에서 묻습니다. 오른쪽 위 X를 눌러 종료하거나 저장 여부에서 n을 입력할 경우 저장하지 않습니다. 저장 여부는 y / n 으로 입력받으며, y를 입력하여 저장할 경우에 저장 위치를 입력받습니다. default는 불러온 obj 파일과 동일합니다. 저장이 완료되면 안정적으로 프로그램을 종료합니다.

3. 실행 결과



1. Wireframe (Basic.PNG)



2. Phong shading w/o Texture (Step1.PNG)

3. Phong shading w/ Texture (Step2.PNG)

4. Phong shading w/ Texture & Normal Mapping (Step3.PNG)

Model for Artistic score

T-34-85_1.PNG

T-34-85_2.PNG

T-34-85_3.PNG

https://www.turbosquid.com/3d-models/3d-model-of-tank/899695

위의 링크로 들어가면 무료 파일인 것을 확인할 수 있습니다.

세계 2차 대전 소련군 중형전차 T-34-85 모델입니다.

파일 위치는 "T-34-85 file input.txt"에 저장하여, 이를 복사해서 이용하면 테스트 실행하기에 편할 것입니다.

4. 참고 사이트

pyglet 공식 문서 - https://pyglet.readthedocs.io/en/latest/