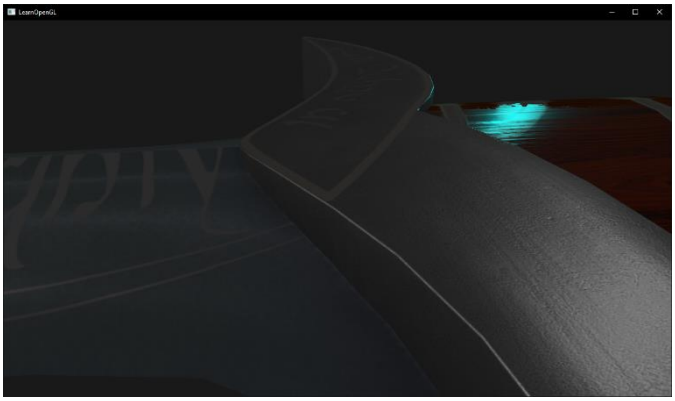
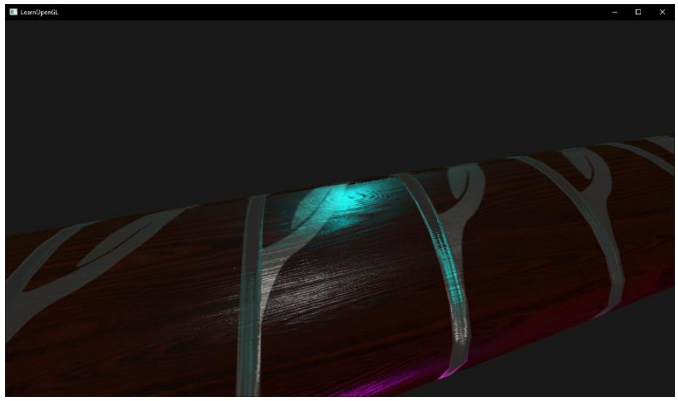
Computer Graphics Project3

제출자: 20193494 문서형

검 모델을 다운로드 받아서 Physics Based Lighting 을 적용시켜 보았습니다. 이 모델을 선정한 이유는 무료로 구할 수 있는 low polygon 모델 중 물리 텍스처가 가장 다양하게 제공되는 모델이었기 때문입니다.





brushed metal 의 광택표현, 나무 손잡이의 굴곡이 매우 현실적으로 표현된 모습을 확인했습니다.

흰색이 아닌 cyan, magenta 색의 빛도 현실적인 색으로 reflect, diffuse 되는 모습을 확인할 수 있었습니다.

구현방식

```
stbi_set_flip_vertically_on_load(false);
Model swordModel(FileSystem::getPath("../resources/sword/sword.obj"));
meshes = swordModel.meshes;
```

우선 `model_load` 의 코드와 같이 `assimp` 라이브러리를 사용해 model 을 로드합니다. 그리고 사용할 텍스쳐를 로드해줍니다.

```
unsigned int albedo =
        loadTexture(FileSystem::getPath("../resources/textures/pbr/sword/diffuse
        .jpg").c_str());
unsigned int normal =
        loadTexture(FileSystem::getPath("../resources/textures/pbr/sword/normal.
        png").c_str());
unsigned int metallic =
        loadTexture(FileSystem::getPath("../resources/textures/pbr/sword/specula
        r.png").c_str());
unsigned int roughness =
        loadTexture(FileSystem::getPath("../resources/textures/pbr/sword/roughne
        ss.png").c_str());
unsigned int ao =

        loadTexture(FileSystem::getPath("../resources/textures/pbr/sword/ao.jpg"
        ).c_str());
```

다음은 메인 렌더링 루프에 모델의 사이즈와 위치를 결정할 model matrix 를 matrix transform 을 이용해 생성해주었습니다.

```
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(0, -10, -10));
model = glm::scale(model, glm::vec3(0.5f));
shader.setMat4("model", model);
```

pbr_light 자체는 주어진 쉐이더에 구현이 되어 있으므로 이를 이용했습니다. 하지만 `model_load` 와 같이 `Model.Draw(Shader shader)` 함수를 사용하면 rendering 이 이상하게 됩니다. 버퍼에 들어가 있지 않고 Draw 함수를 통해 그려지는 object 의 경우 shader 가 제대로 적용이 되지 않기 때문으로 보입니다. 이를 해결하기 위해 `pbr_light` 에서 rusted iron 구를 render 한 것처럼 각 vertex 의 position, texture coordinate 등을 Model 에서 추출해 Buffer 에 넣어주었습니다. 이 내용은 `void renderMesh(Mesh mesh, int i)`에 구현되어 있습니다.

`renderMesh` 함수:

`Model` 의 `meshes` 멤버안에 있는 `Mesh` 와 번호를 argument 로 하여 호출하면 버퍼에 모델이 없는 경우 버퍼에 로드 후, 버퍼에 있는 object 를 그려줍니다.

1. 각 Mesh 번호에 해당하는 좌표를 `renderSphere` 와 같은 방식으로 vector 에 저장한 후 버퍼에 로드합니다.
2. 버퍼에 로드된 메쉬를 그려줍니다.

```
glBindVertexArray(vaos[i]);
glDrawElements(GL_TRIANGLES, indexlist[i].size(), GL_UNSIGNED_INT, 0);
```

여기서 `glDrawElemnts` 의 parmeter, `glElumMode` 를 `GL_TRIANGLE_STRIP` 에서 `GL_TRIANGLES` 로 바꾸어 주었습니다. `Mesh` 의 `Indices` 배열을 살펴본 결과 strip 형태가 아니라 개별의 삼각형 index 가 나열되어 있는 형태였기 때문입니다.