Lighting / Lighting maps

-지금까지 사용한 material은 하나의 오브젝트에 하나의 material을 적용시킴

-실제의 물체, 오브젝트들은 여러가지 material을 가질 수 있음

Diffuse maps

-빛이 존재하는 scene에서 texture 이미지가 오브젝트의 모든 diffuse 컬러를 나타냄

-따라서 texture를 diffuse map이라고 부름

-fragment shader source

|  |
| --- |
| struct Material {  sampler2D diffuse;  vec3 specular;  float shininess;  };  ...  in vec2 TexCoords; |

-ambient컬러도 대부분 diffuse 컬러와 동일해서 삭제함

(그대로 사용해도 무방하나, 각 fragment 마다 다른 ambient값을 얻기 위해서 ambient값을 위한 다른 텍스처를 사용해야함)

-텍스처 좌표도 추가

|  |
| --- |
| vec3 diffuse = light.diffuse \* diff \* vec3(texture(material.diffuse, TexCoords)); |

-ambient컬러를 diffuse컬러와 동일하게 설정

|  |
| --- |
| vec3 ambient = light.ambient \* vec3(texture(material.diffuse, TexCoords)); |

-vertex shader soucre

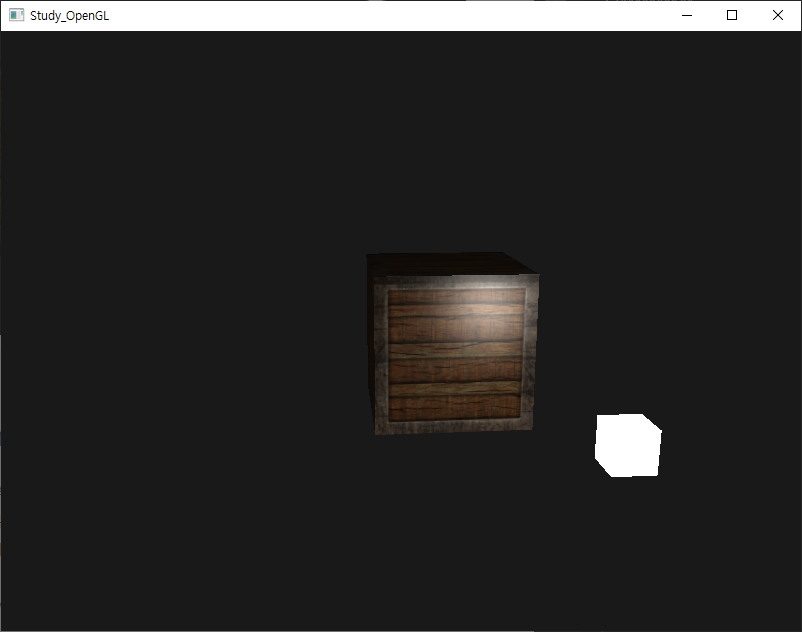
|  |
| --- |
| #version 460 core  layout (location = 0) in vec3 aPos;  layout (location = 1) in vec3 aNormal;  layout (location = 2) in vec2 aTexCoords;  ...  out vec2 TexCoords;  void main()  {  ...  TexCoords = aTexCoords;  } |

-텍스처 불러오기(함수로 따로 만들어서 진행)

|  |
| --- |
| unsigned int diffuseMap = loadTexture("textures/container2.png"); |
| unsigned int loadTexture(char const \* path)  {  unsigned int textureID;  glGenTextures(1, &textureID);    int width, height, nrComponents;  unsigned char \*data = stbi\_load(path, &width, &height, &nrComponents, 0);  if (data)  {  GLenum format;  if (nrComponents == 1)  format = GL\_RED;  else if (nrComponents == 3)  format = GL\_RGB;  else if (nrComponents == 4)  format = GL\_RGBA;  glBindTexture(GL\_TEXTURE\_2D, textureID);  glTexImage2D(GL\_TEXTURE\_2D, 0, format,  width, height, 0, format, GL\_UNSIGNED\_BYTE, data);  glGenerateMipmap(GL\_TEXTURE\_2D);  glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT);  glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT);  glTexParameteri(GL\_TEXTURE\_2D,  GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR\_MIPMAP\_LINEAR);  glTexParameteri(GL\_TEXTURE\_2D,  GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);  stbi\_image\_free(data);  }  else  {  std::cout << "Texture failed to load at path: " << path << std::endl;  stbi\_image\_free(data);  }  return textureID;  } |

-렌더링 루프에 텍스처 바인딩

|  |
| --- |
| lightingShader.setInt("material.diffuse", 0);  ...  glActiveTexture(GL\_TEXTURE0);  glBindTexture(GL\_TEXTURE\_2D, diffuseMap); |



Specular maps

-현실에서는 나무는 하이라이트가 없고, 금속은 하이라이트가 있음

-구현을 위해 specular을 vec3(0.0f)으로 설정하면 하이라이트가 없어지나, 금속부분에도 영향을 미침

-이를 해결하기 위해 Specular map을 사용

-Specular Map은 흑백사진이며, 이는 0에 가까울수록 검은색, Specular도 0에 가까워짐, 즉 0은 빛나지 않고, 1은 빛나는 모습을 보여주게 됨

Sampling Specular Maps

-diffuse map 코드와 비슷함

-텍스쳐를 로드

|  |
| --- |
|  |

-텍스쳐를 바인딩

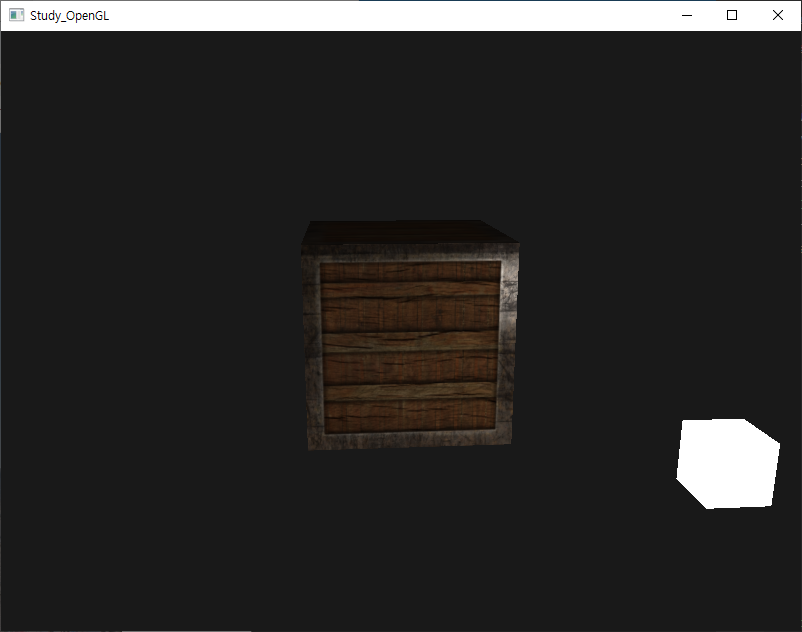
|  |
| --- |
| lightingShader.setInt("material.specular", 1);  ...  glActiveTexture(GL\_TEXTURE1);  glBindTexture(GL\_TEXTURE\_2D, specularMap); |

-Fragment shader / Material 속성 변경

|  |
| --- |
| struct Material {  sampler2D diffuse;  sampler2D specular;  float shininess;  }; |

-specular map을 샘플링

|  |
| --- |
| vec3 ambient = light.ambient \* vec3(texture(material.diffuse, TexCoords));  vec3 diffuse = light.diffuse \* diff \* vec3(texture(material.diffuse, TexCoords));  vec3 specular = light.specular \* spec \* vec3(texture(material.specular, TexCoords));  FragColor = vec4(ambient + diffuse + specular, 1.0); |



Emission Map

-게임에서 사용되는 오브젝트에서 특정 부분이 항상 빛나는 것을 표현할 때 사용

ex)

-texture 추가

|  |
| --- |
| unsigned int emissionMap = loadTexture("textures/matrix.jpg"); |

-텍스쳐 바인딩

|  |
| --- |
| lightingShader.setInt("material.emission", 2);  ...  glActiveTexture(GL\_TEXTURE2);  glBindTexture(GL\_TEXTURE\_2D, emissionMap); |

-fragment shader 변경

|  |
| --- |
| struct Material{  sampler2D diffuse;  sampler2D specular;  sampler2D emission;  float shininess;  }; |
| // emission  vec3 show = step(vec3(1.0f),  vec3(1.0) - vec3(texture(material.specular, TexCoords)));  vec3 emission = vec3(texture(material.emission, TexCoords)) \* show;  vec3 result = ambient + diffuse + specular + emission; |

-step : 0, 1의 결과를 return하는 함수

<https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/step.xhtml>

-결과

