Lighting / Multiple Light

-지금까지 배운 빛

-Phone Shading

-Materials

-Lighting Maps

-Light Caster

-하나 이상의 광원을 scene에 사용할려면 계산을 GLSL의 function에 캡슐화 해야함

-main에 넣으면 너무 복잡한 형태가 됨

-캡슐화된 후의 main코드의 형태(대략적인 구조)

|  |
| --- |
| out vec4 FragColor;    void main()  {  // 출력 컬러 값을 정의  vec3 output = vec3(0.0);  // Directional light의 기여를 출력에 더함  output += someFunctionToCalculateDirectionalLight();  // 모든 point light들에도 동일하게 수행  for(int i = 0; i < nr\_of\_point\_lights; i++)  output += someFunctionToCalculatePointLight();  // 다른 light들도 더함(spotlight 같은 것들)  output += someFunctionToCalculateSpotLight();    FragColor = vec4(output, 1.0);  } |

-Directional Light 변수 추가

|  |
| --- |
| struct DirLight {  vec3 direction;    vec3 ambient;  vec3 diffuse;  vec3 specular;  };  uniform DirLight dirLight; |

-dirLight를 전달할 함수

|  |
| --- |
| vec3 CalcDirLight(DirLight light, vec3 normal, vec3 viewDir); |

-함수

|  |
| --- |
| vec3 CalcDirLight(DirLight light, vec3 normal, vec3 viewDir)  {  vec3 lightDir = normalize(-light.direction);  // diffuse shading  float diff = max(dot(normal, lightDir), 0.0);  // specular shading  vec3 reflectDir = reflect(-lightDir, normal);  float spec = pow(max(dot(viewDir, reflectDir), 0.0), material.shininess);  // 결과들을 결합  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));  return (ambient + diffuse + specular);  } |

Point Light

-PointLight 변수추가

|  |
| --- |
| struct PointLight {  vec3 position;    float constant;  float linear;  float quadratic;  vec3 ambient;  vec3 diffuse;  vec3 specular;  };  #define NR\_POINT\_LIGHTS 4  uniform PointLight pointLights[NR\_POINT\_LIGHTS]; |

-함수 추가

|  |
| --- |
| vec3 CalcPointLight(PointLight light, vec3 normal, vec3 fragPos, vec3 viewDir); |

-함수

|  |
| --- |
| vec3 CalcPointLight(PointLight light, vec3 normal, vec3 fragPos, vec3 viewDir)  {  vec3 lightDir = normalize(light.position - fragPos);  // diffuse shading  float diff = max(dot(normal, lightDir), 0.0);  // specular shading  vec3 reflectDir = reflect(-lightDir, normal);  float spec = pow(max(dot(viewDir, reflectDir), 0.0), material.shininess);  // attenuation  float distance = length(light.position - fragPos);  float attenuation = 1.0 / (light.constant + light.linear \* distance +  light.quadratic \* (distance \* distance));  // 결과들을 결합  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));  ambient \*= attenuation;  diffuse \*= attenuation;  specular \*= attenuation;  return (ambient + diffuse + specular);  } |

Putting it all together

-main함수

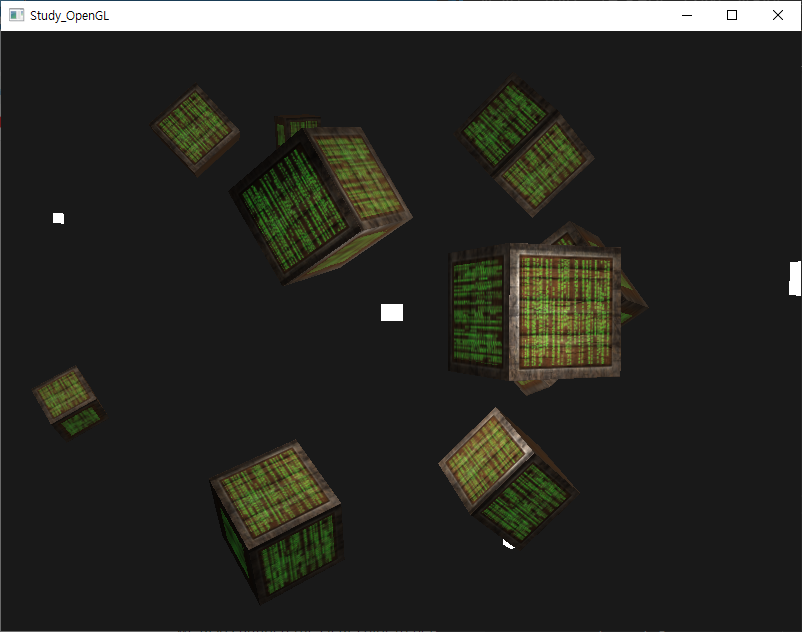
|  |
| --- |
| void main()  {  // 속성들  vec3 norm = normalize(Normal);  vec3 viewDir = normalize(viewPos - FragPos);  // 1 단계: Directional lighting  vec3 result = CalcDirLight(dirLight, norm, viewDir);  // 2 단계: Point lights  for(int i = 0; i < NR\_POINT\_LIGHTS; i++)  result += CalcPointLight(pointLights[i], norm, FragPos, viewDir);  // 3 단계: Spot light  //result += CalcSpotLight(spotLight, norm, FragPos, viewDir);    FragColor = vec4(result, 1.0);  } |

-SpotLight는 각자 구현해보기!

-4개의 PointLight위치

|  |
| --- |
| glm::vec3 pointLightPositions[] = {  glm::vec3( 0.7f, 0.2f, 2.0f),  glm::vec3( 2.3f, -3.3f, -4.0f),  glm::vec3(-4.0f, 2.0f, -12.0f),  glm::vec3( 0.0f, 0.0f, -3.0f)  }; |

-emmision을 추가한 상태



-SpotLight를 켜고 끌 수 있게 해보자

-Fragment shader에서 result+= CalcSpotLight가 특정 버튼을 눌렀을 때, 작동하거나 하지 않게 하면 될 것 같다.

-fragment shader

|  |
| --- |
| [...]  uniform bool isOnSpotLight;  [...]  if(isOnSpotLight)  result += CalcSpotLight(spotLight, norm, FragPos, viewDir); |

-전역변수 추가

|  |
| --- |
| //SpotLight 켜고 끄기  bool isOnSpotLight = true;  bool isFkeyPressed = false; |

-랜더링 함수내부

|  |
| --- |
| //spotlight 켜고 끄기  if(glfwGetKey(window, GLFW\_KEY\_F) == GLFW\_PRESS && !isFkeyPressed && !isOnSpotLight)  {  isFkeyPressed = true;  }  if(glfwGetKey(window, GLFW\_KEY\_F) == GLFW\_RELEASE && isFkeyPressed && !isOnSpotLight)  {  isOnSpotLight = true;  isFkeyPressed = false;  }  if(glfwGetKey(window, GLFW\_KEY\_F) == GLFW\_PRESS && !isFkeyPressed && isOnSpotLight)  {  isFkeyPressed = true;  }  if(glfwGetKey(window, GLFW\_KEY\_F) == GLFW\_RELEASE && isFkeyPressed && isOnSpotLight)  {  isOnSpotLight = false;  isFkeyPressed = false;  }  [...]  lightingShader.setBool("isOnSpotLight", isOnSpotLight); |

-bool 을 2개 쓴 이유

F키를 눌렀을 때, isFkeyPressed를 바꾸는 형태로 코딩하면

F키의 입력을 유지하고 있을 때, 빠르게 켜고 꺼지는 오류가 발행할 수도 있기에

bool을 F키의 입력상태, SpotLight의 on/off상태를 만들어서 오류가 발행하지 않게 제작

