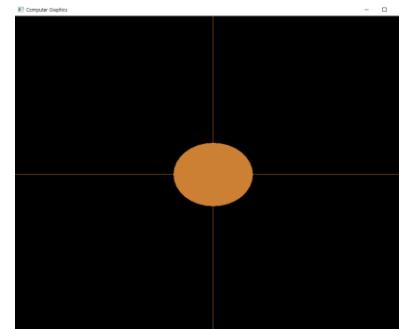


openGL 조명

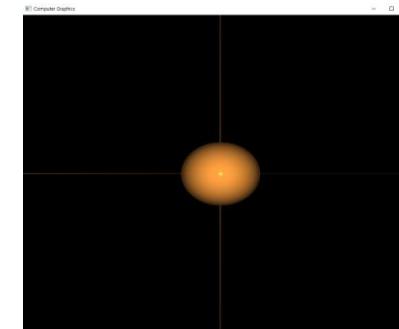
2025년 2학기

조명

- 광원은 색(color)과 강도(intensity)를 가지고 있다.
 - 기본 색은 백색
 - 우리가 보는 객체의 색은 빛의 색 (light color)과 객체의 재질의 색 (material color)의 합성으로 정해진다.
 - 빛의 색: 조명의 RGB 값
 - 재질의 색: 객체의 표면에서 RGB 값
 - 예)
 - 하얀색 조명과 빨간색 재질: $(1.0, 1.0, 1.0) * (1.0, 0.0, 0.0) \rightarrow (1.0, 0.0, 0.0)$ 객체의 색은 빨간색
 - 빨간색 조명과 하얀색 재질: $(1.0, 0.0, 0.0) * (1.0, 1.0, 1.0) \rightarrow (1.0, 0.0, 0.0)$ 객체의 색은 빨간색
 - 빨간색 조명과 파란색 재질: $(1.0, 0.0, 0.0) * (0.0, 0.0, 1.0) \rightarrow (0.0, 0.0, 0.0)$ 객체의 색은 검정색
 - 노란색 조명과 청록색 재질: $(1.0, 1.0, 0.0) * (0.0, 1.0, 1.0) \rightarrow (0.0, 1.0, 0.0)$ 객체의 색은 초록색
 - $(0.8, 0.5, 0.2)$ 재질의 색과 백색 조명 $(1.0, 1.0, 1.0)$ 인 경우



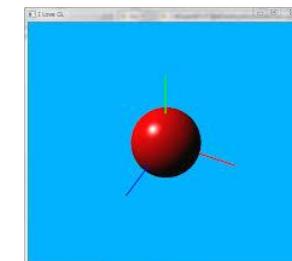
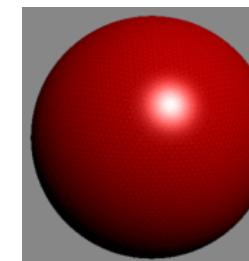
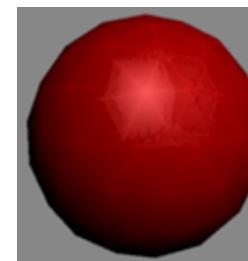
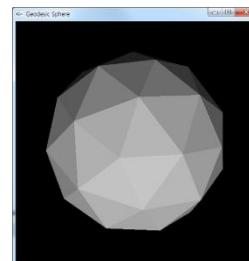
재질의 색: $(0.8, 0.5, 0.2)$



조명의 색: $(1.0, 1.0, 1.0)$

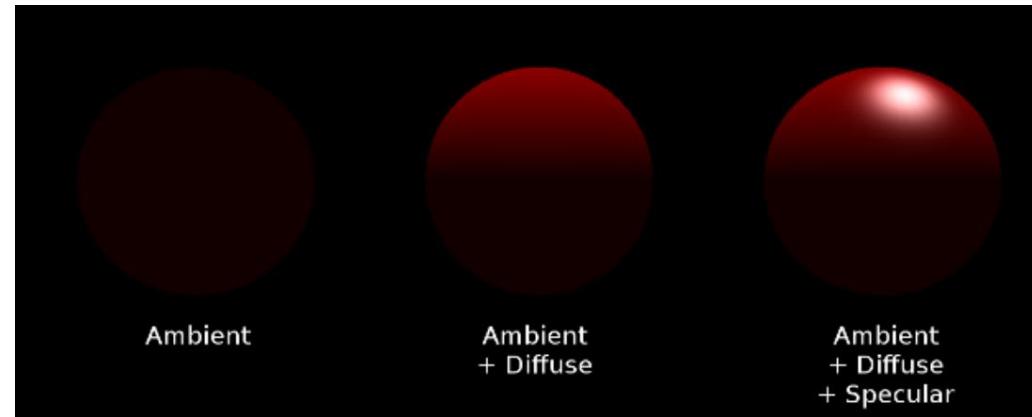
조명

- OpenGL에서 빛 (조명)은
 - OpenGL의 조명은 **광원 (Lights)**과 **재질 (Materials)**, 그리고 **면의 법선벡터 (Normal)**에 의해서 결정된다.
 - **광원**: 위치 (position), 색 (color)
 - 광원의 강도는 색(Red, Green, Blue)의 강도에 의해 결정
 - 광원은 객체와 같이 변환 행렬에 의해 움직일 수 있다.
 - **재질**: 표면이 빛을 어떻게 반사하는지 나타냄
 - 기본적으로 표면의 색, 반사계수
 - **법선 벡터**: 표면에 수직을 이루는 법선벡터는 표면의 방향
 - 법선 벡터는 단위벡터이어야 한다.
 - 법선 벡터는 버텍스의 속성으로 전달할 수 있다.
 - 조명 효과 설정
 - 색과 법선 벡터, 반사 계수: 응용 프로그램에서 설정
 - 버텍스 세이더 또는 프래그먼트 세이더에서 객체의 색과 법선 벡터, 반사 계수 등을 사용하여 조명 효과 설정



Phong Lighting Model

- 풍 라이팅 모델: 가장 일반적인 라이팅 모델로 주변 조명, 산란반사 조명, 거울반사 조명 등 세 가지 요소로 조명 모델 설정
 - 주변 조명 (ambient light)
 - 배경 조명, 전역 조명
 - 특정 방향을 갖지 않고 모든 방향으로 고루 비춰지는 조명
 - 산란 반사 조명 (diffuse light)
 - 방향과 위치를 가지고 있는 조명으로 빛이 비치는 물체의 면이 밝아진다.
 - 빛을 받는 표면은 그렇지 않은 부분에 비해 밝게 보인다.
 - 거울 반사 조명 (specular light)
 - 특정한 방향으로 들어와서 한 방향으로 완전히 반사되는 조명
 - 반짝이는 표면을 모델링할 때 사용됨



주변 조명 (Ambient light)

- 주변 조명 (전역 조명)

- 간접적으로 들어오는 빛을 나타내는 조명으로 특정 방향을 갖지 않음
- 객체의 위치나 방향과 관계없이 일정한 밝기의 빛이 고르게 퍼져있다고 정함
- 주변 조명 효과는 조명색과 주변조명 계수, 객체의 색으로 결정함
 - 조명의 색상과 주변조명 계수를 곱해서 주변 조명값으로 정한다.
 - 또는 특정 상수값으로 설정한다.

- $I = K_a \cdot I_a,$ $0 \leq K_a \leq 1$

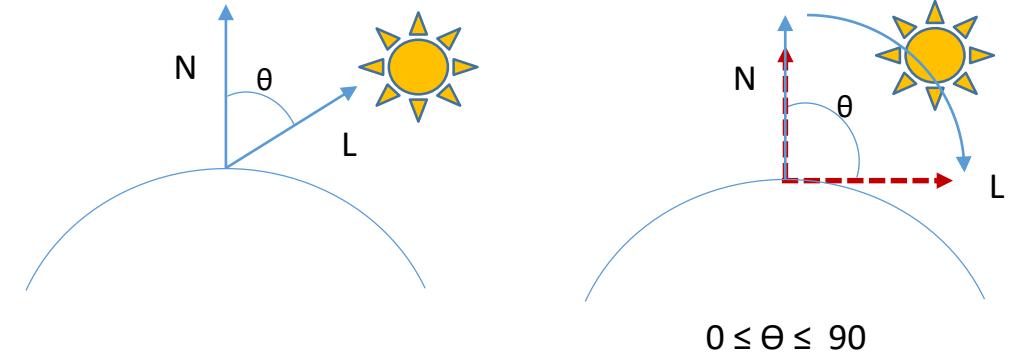
- $I_a:$ 광원의 색
 - $K_a:$ 주변조명 계수 (물체의 표면이 입사된 빛을 반사하는 정도로 표면을 이루는 고유 물질에 따라 다르다)

산란반사조명 (Diffuse light)

- 산란반사조명
 - 특정 방향으로 진행하다가 표면에 닿으면 모든 방향으로 동일하게 반사된다.
 - 관찰자의 위치와 무관하며 가장 일반적인 조명으로 조명을 받는 부분은 안 받는 부분보다 환하게 보인다.
 - 객체의 표면과 광원 과의 각도에 따라 조명 값이 달라진다.
- 산란반사 조명값:

$$I = \frac{I_p}{d} \cdot K_d \cdot \cos\theta = \frac{I_p}{d} \cdot K_d \cdot (N \cdot L)$$

- I_p : 광원의 색
- K_d : 표면의 산란반사 계수
- N : 표면의 법선 벡터 (단위벡터)
- L : 광원의 방향 벡터 (단위벡터)
- θ : N 과 L 사이의 각도
- d : 표면에서 광원까지의 거리

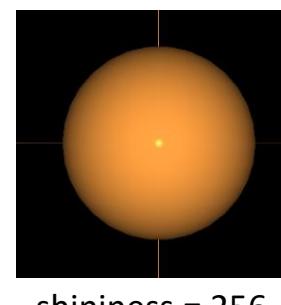


거울반사조명 (Specular light)

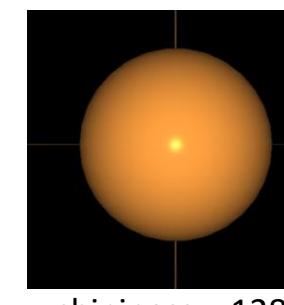
- 거울반사조명 (정반사 조명)
 - 반짝이는 하이라이트를 생성한다.
 - 관찰자가 빛의 입사각과 거의 같은 반사각 부근에 위치할 경우 입사된 빛의 전부를 인식하며 하이라이트가 생긴다.
 - 거울 반사 조명은 재질의 shininess (광택 계수) 정도를 추가: shininess 가 높으면 작은 면적의 하이라이트가 생성된다.
- 거울반사조명값:

$$I = \frac{I_p}{d} \cdot K_s \cdot \cos^n \Phi = \frac{I_p}{d} \cdot K_s \cdot (V \cdot R)^n$$

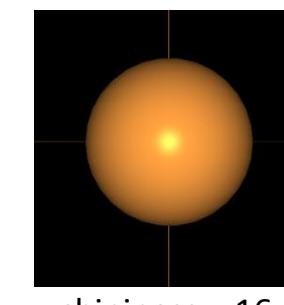
- I_p : 광원의 색
- K_s : 표면 물질의 거울반사계수
- V : 관찰자에 대한 벡터
- R : 빛의 반사 방향 벡터
- n (shininess 광택 계수): 표면의 광택 정도에 따라 정해지는 값
- d : 표면에서 광원까지의 거리



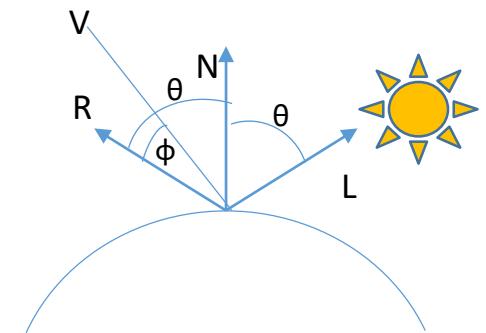
shininess = 256



shininess = 128



shininess = 16

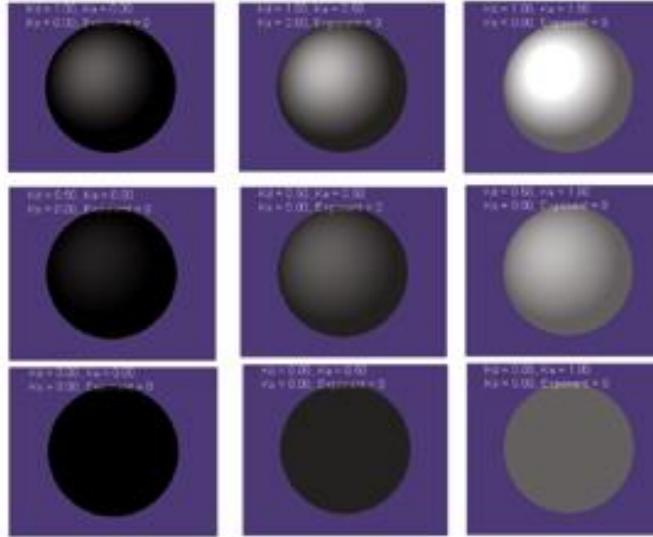


N: normal vector
L: 조명을 가리키는 벡터
R: 빛의 반사 방향
V: 관찰자 위치
 ϕ : V와 R 사이의 각도

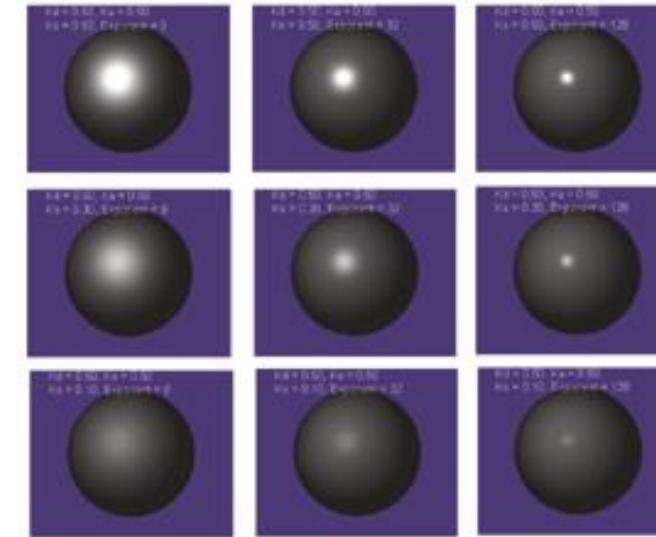
최종 조명 효과

- 표면에 조명 효과를 모델링할 수 있는 공식

$$\bullet I = K_a \cdot I_a + \frac{I_p}{d} \cdot [K_d \cdot (N \cdot L) + K_s \cdot (V \cdot R)^n]$$



산란반사



산란반사 + 거울반사

셰이딩 (Shading)

- 셰이딩 (Shading):
 - 3차원 객체의 표면을 처리할 때 조명이 위치와 물체의 방향, 색, 밝기에 따라 객체에 음영을 주는 것
 - Flat shading (플랫 셰이딩)
 - 하나의 다각형 전체를 동일한 색으로 적용
 - Gouraud shading (고라우드 셰이딩)
 - 버텍스 간 결과 **색상들을 보간하여 셰이딩에 사용**
 - 전체 라이팅 공식을 **버텍스 세이더에서 구현**
 - 각 프래그먼트의 최종 색상은 프래그먼트 세이더로 전달
 - 각 프래그먼트 색상은 프래그먼트 세이더로 전달되기 전에 보간
 - 프래그먼트 세이더는 단순히 입력 색상을 프레임버퍼에 쓰는 역할
 - 스페큘러 하이라이트 부분에 별 모양의 패턴
 - 삼각형 간의 불일치, 즉 색상 공간 내에서 선형적으로 보간되기 때문
 - Phong shading (퐁 셰이딩)
 - 버텍스간에 색상값을 보간하는 것이 아니라 **버텍스 간의 표면 법선벡터를 보간**
 - 그 결과 법선값을 사용하여 버텍스가 아닌 **프래그먼트 세이더에서 픽셀에 대해 전체 라이팅 계산을 수행**
 - 고라우드 셰이딩 같은 별 모양이 없고 훨씬 좋은 결과물
 - 프래그먼트 세이더에서 많은 작업을 수행한다.



퐁 셰이딩: 1) 주변조명 (Ambient light) 적용

- 프래그먼트 세이더에 조명값을 적용한다.

```
//--- 프래그먼트 세이더
#version 330 core
out vec4 FragColor;

uniform vec3 objectColor;           //--- 응용 프로그램에서 설정한 객체의 색상
uniform vec3 lightColor;           //--- 응용 프로그램에서 설정한 조명 색상

void main ()
{
    float ambientLight = 0.5;        //--- 주변 조명 계수: 0.0 ≤ ambientLight ≤ 1.0
    vec3 ambient = ambientLight * lightColor; //--- 주변 조명값

    vec3 result = ambient * objectColor; //--- 객체의 색과 주변조명값을 곱하여 최종 객체 색상 설정

    FragColor = vec4 (result, 1.0);
}
```

퐁 셰이딩: 1) 주변조명 (Ambient light) 적용

- 응용 프로그램

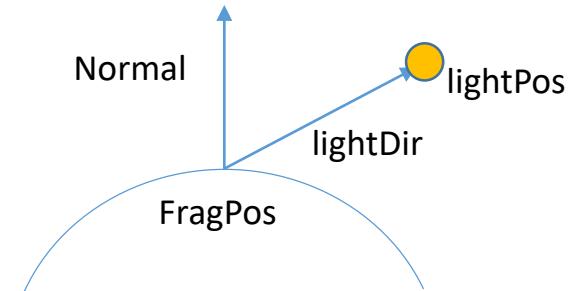
```
void InitBuffer ()  
{  
    unsigned int VBO, VAO;  
  
    glGenVertexArrays(1, &VAO);  
    glGenBuffers(1, &VBO);  
  
    glBindVertexArray(VAO);  
    glBindBuffer(GL_ARRAY_BUFFER, VBO);  
    glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices, GL_STATIC_DRAW);  
  
    glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float), (void*)0);           //--- 위치 속성  
    glEnableVertexAttribArray(0);  
    glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float), (void*)(3 * sizeof(float))); //--- 노말 속성  
    glEnableVertexAttribArray(1);  
  
    glUseProgram(shaderProgram);  
    int lightColorLocation = glGetUniformLocation(shaderProgram, "lightColor");           //--- lightColor 값 전달: (1.0, 1.0, 1.0) 백색  
    glUniform3f(lightColorLocation, 1.0, 1.0, 1.0);  
    int objColorLocation = glGetUniformLocation(shaderProgram, "objectColor");           //--- object Color값 전달: (1.0, 0.5, 0.3)의 색  
    glUniform3f(objColorLocation, 1.0, 0.5, 0.3);  
}
```

퐁 셰이딩: 2) 주변조명 + 산란반사조명 (Diffuse light) 적용

- 산란반사 조명 설정

- 응용 프로그램

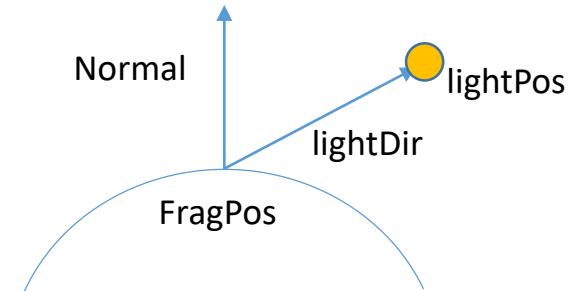
```
float vertices[] = {  
    //--- 버텍스 속성: 좌표값(FragPos), 노말값 (Normal)  
    -0.5f, -0.5f, -0.5f, 0.0f, 0.0f, -1.0f, 0.5f, -0.5f, -0.5f, 0.0f, 0.0f, -1.0f,  
    0.5f, 0.5f, -0.5f, 0.0f, 0.0f, -1.0f, -0.5f, -0.5f, 0.5f, 0.0f, 0.0f, 1.0f,  
    0.5f, 0.5f, 0.5f, 0.0f, 0.0f, 1.0f, -0.5f, 0.5f, 0.5f, 0.0f, 0.0f, 1.0f,  
    -0.5f, 0.5f, 0.5f, -1.0f, 0.0f, 0.0f, -0.5f, -0.5f, -0.5f, -1.0f, 0.0f, 0.0f,  
    -0.5f, -0.5f, -0.5f, -1.0f, 0.0f, 0.0f, 0.5f, 0.5f, 0.5f, 1.0f, 0.0f, 0.0f,  
    0.5f, 0.5f, 0.5f, 1.0f, 0.0f, 0.0f, 0.5f, -0.5f, -0.5f, 1.0f, 0.0f, 0.0f,  
    0.5f, -0.5f, -0.5f, -1.0f, 0.0f, 0.0f, 0.5f, -0.5f, 0.5f, 1.0f, 0.0f, 0.0f,  
    -0.5f, -0.5f, -0.5f, 0.0f, -1.0f, 0.0f, 0.5f, -0.5f, 0.5f, 0.0f, -1.0f, 0.0f,  
    0.5f, 0.5f, -0.5f, 0.0f, 1.0f, 0.0f, 0.5f, 0.5f, 0.5f, 0.0f, 1.0f, 0.0f,  
    0.5f, 0.5f, 0.5f, 0.0f, 1.0f, 0.0f, -0.5f, 0.5f, 0.5f, 0.0f, 1.0f, 0.0f};
```



퐁 셰이딩: 2) 주변조명 + 산란반사조명 (Diffuse light) 적용

- 응용 프로그램

```
void InitBuffer ()  
{  
    unsigned int VBO, VAO;  
  
    glGenVertexArrays(1, &VAO);  
    glGenBuffers(1, &VBO);  
  
    glBindVertexArray(VAO);  
    glBindBuffer(GL_ARRAY_BUFFER, VBO);  
    glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices, GL_STATIC_DRAW);  
  
    glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float), (void*)0); //--- 위치 속성  
    glEnableVertexAttribArray(0);  
    glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float), (void*)(3 * sizeof(float))); //--- 노말 속성  
    glEnableVertexAttribArray(1);  
  
    glUseProgram(shaderProgram);  
    unsigned int lightPosLocation = glGetUniformLocation(shaderProgram, "lightPos"); //--- lightPos 값 전달: (0.0, 0.0, 5.0);  
    glUniform3f(lightPosLocation, 0.0, 0.0, 5.0);  
    unsigned int lightColorLocation = glGetUniformLocation(shaderProgram, "lightColor"); //--- lightColor 값 전달: (1.0, 1.0, 1.0) 백색  
    glUniform3f(lightColorLocation, 1.0, 1.0, 1.0);  
    unsigned int objColorLocation = glGetUniformLocation(shaderProgram, "objectColor"); //--- object Color값 전달: (1.0, 0.5, 0.3)의 색  
    glUniform3f(objColorLocation, 1.0, 0.5, 0.3);  
}
```



//--- 위치 속성

//--- 노말 속성

//--- lightPos 값 전달: (0.0, 0.0, 5.0);

//--- lightColor 값 전달: (1.0, 1.0, 1.0) 백색

//--- object Color값 전달: (1.0, 0.5, 0.3)의 색

퐁 셰이딩: 2) 주변조명 + 산란반사조명 (Diffuse light) 적용

- 버텍스 세이더

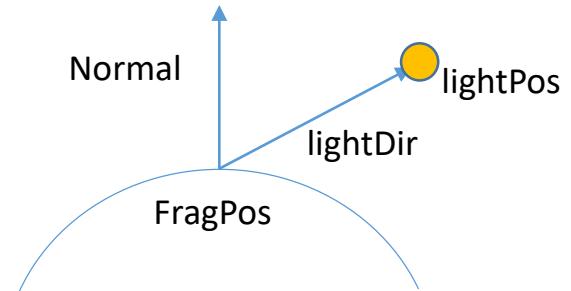
```
#version 330 core
layout (location = 0) in vec3 vPos;
layout (location = 1) in vec3 vNormal;

out vec3 FragPos;           //--- 객체의 위치값을 프래그먼트 세이더로 보낸다.
out vec3 Normal;            //--- 노멀값을 프래그먼트 세이더로 보낸다.

uniform mat4 model;         //--- 모델링 변환값
uniform mat4 view;          //--- 뷰잉 변환값
uniform mat4 projection;    //--- 투영 변환값

void main()
{
    gl_Position = projection * view * model * vec4(vPos, 1.0);

    FragPos = vec3(model * vec4(vPos, 1.0));   //--- 객체에 대한 조명 계산을 프래그먼트 세이더에서 한다.
                                                //--- 따라서 월드공간에 있는 버텍스 값을 프래그먼트 세이더로 보낸다.
    Normal = vNormal;                         //--- 노멀값을 프래그먼트 세이더로 보낸다.
}
```



퐁 셰이딩: 2) 주변조명 + 산란반사조명 (Diffuse light) 적용

- 프래그먼트 세이더

```
#version 330 core
in vec4 FragPos;
in vec3 Normal;

out vec4 FragColor;

uniform vec3 lightPos;
uniform vec3 lightColor;
uniform vec3 objectColor;

void main ()
{
    vec3 ambientLight = 0.3;
    vec3 ambient = ambientLight * lightColor;

    vec3 normalVector = normalize (Normal);
    vec3 lightDir = normalize (lightPos – FragPos);
    float diffuseLight = max (dot (norm, lightDir), 0.0);
    float diffuse = diffuseLight * lightColor;

    vec3 result = (ambient + diffuse) * objectColor;

    FragColor = vec4 (result, 1.0);
}
```

//--- 위치값
//--- 버텍스 세이더에서 받은 노멀값

//--- 최종 객체의 색 저장

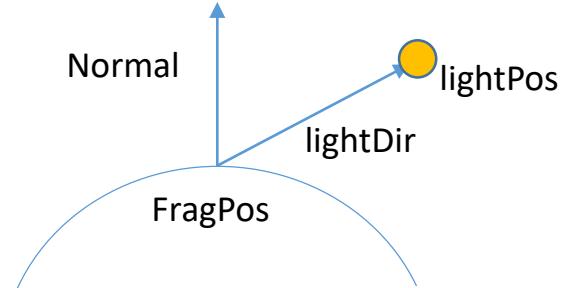
//--- 조명의 위치
//--- 조명의 색
//--- 객체의 색

//--- 주변 조명 계수
//--- 주변 조명 값

//--- 노말값을 정규화한다.
//--- 표면과 조명의 위치로 조명의 방향을 결정한다.
//--- N과 L의 내적 값으로 강도 조절 (음의 값을 가질 수 없게 한다.)
//--- 산란반사조명값=산란반사값*조명색상값

//--- 최종 조명 설정된 픽셀 색상=(주변조명+산란반사조명)*객체 색상

//--- 픽셀 색을 출력



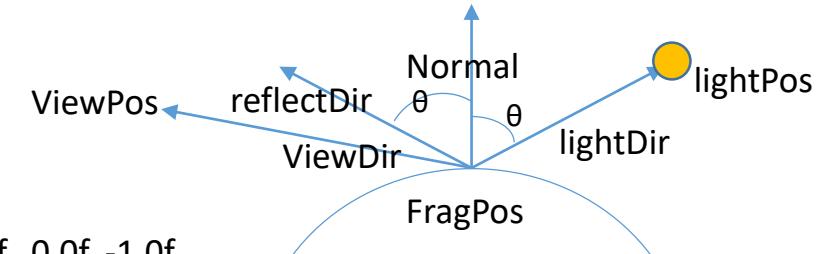
퐁 셰이딩: 3) 주변, 산란반사 + 거울반사 조명 (Specular light) 적용

- 거울반사 조명 설정

- 응용 프로그램

```
float vertices[] = { //--- 버텍스 속성: 좌표값(FragPos), 노말값 (Normal)
```

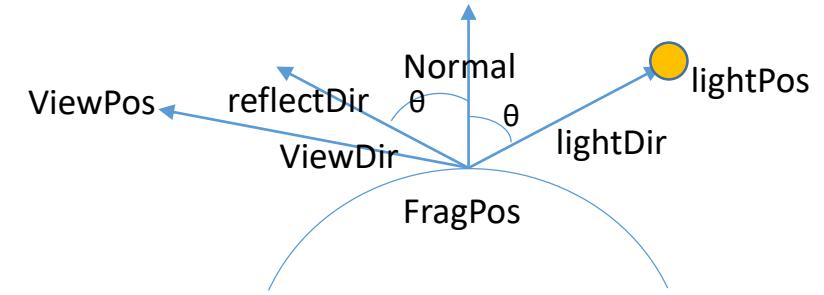
-0.5f, -0.5f, -0.5f, 0.0f, 0.0f, -1.0f, 0.5f, 0.5f, -0.5f, 0.0f, 0.0f, -1.0f,	0.5f, -0.5f, -0.5f, 0.0f, 0.0f, -1.0f, -0.5f, 0.5f, -0.5f, 0.0f, 0.0f, -1.0f,	0.5f, 0.5f, -0.5f, 0.0f, 0.0f, -1.0f, -0.5f, -0.5f, -0.5f, 0.0f, 0.0f, -1.0f,
-0.5f, -0.5f, 0.5f, 0.0f, 0.0f, 1.0f, 0.5f, 0.5f, 0.5f, 0.0f, 0.0f, 1.0f,	0.5f, -0.5f, 0.5f, 0.0f, 0.0f, 1.0f, -0.5f, 0.5f, 0.5f, 0.0f, 0.0f, 1.0f,	0.5f, 0.5f, 0.5f, 0.0f, 0.0f, 1.0f, -0.5f, -0.5f, 0.5f, 0.0f, 0.0f, 1.0f,
-0.5f, 0.5f, 0.5f, -1.0f, 0.0f, 0.0f, -0.5f, -0.5f, -0.5f, -1.0f, 0.0f, 0.0f,	-0.5f, 0.5f, -0.5f, -1.0f, 0.0f, 0.0f, -0.5f, -0.5f, 0.5f, -1.0f, 0.0f, 0.0f,	-0.5f, -0.5f, -0.5f, -1.0f, 0.0f, 0.0f, -0.5f, 0.5f, 0.5f, -1.0f, 0.0f, 0.0f,
0.5f, 0.5f, 0.5f, 1.0f, 0.0f, 0.0f, 0.5f, -0.5f, -0.5f, 1.0f, 0.0f, 0.0f,	0.5f, 0.5f, -0.5f, 1.0f, 0.0f, 0.0f, 0.5f, -0.5f, 0.5f, 1.0f, 0.0f, 0.0f,	0.5f, -0.5f, -0.5f, 1.0f, 0.0f, 0.0f, 0.5f, 0.5f, 0.5f, 1.0f, 0.0f, 0.0f,
-0.5f, -0.5f, -0.5f, 0.0f, -1.0f, 0.0f, 0.5f, -0.5f, 0.5f, 0.0f, -1.0f, 0.0f,	0.5f, -0.5f, -0.5f, 0.0f, -1.0f, 0.0f, -0.5f, -0.5f, 0.5f, 0.0f, -1.0f, 0.0f,	0.5f, -0.5f, 0.5f, 0.0f, -1.0f, 0.0f, -0.5f, -0.5f, -0.5f, 0.0f, -1.0f, 0.0f,
-0.5f, 0.5f, -0.5f, 0.0f, 1.0f, 0.0f, 0.5f, 0.5f, 0.5f, 0.0f, 1.0f, 0.0f,	0.5f, 0.5f, -0.5f, 0.0f, 1.0f, 0.0f, -0.5f, 0.5f, 0.5f, 0.0f, 1.0f, 0.0f,	0.5f, 0.5f, 0.5f, 0.0f, 1.0f, 0.0f, -0.5f, 0.5f, -0.5f, 0.0f, 1.0f, 0.0f,



퐁 셰이딩: 3) 주변, 산란반사 + 거울반사 조명 (Specular light) 적용

- 응용 프로그램

```
void InitBuffer ()  
{  
    unsigned int VBO, VAO;  
  
    glGenVertexArrays(1, &VAO);  
    glGenBuffers(1, &VBO);  
  
    glBindVertexArray(VAO);  
    glBindBuffer(GL_ARRAY_BUFFER, VBO);  
    glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices, GL_STATIC_DRAW);  
  
    glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float), (void*)0);           //--- 위치 속성  
    glEnableVertexAttribArray(0);  
    glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float), (void*)(3 * sizeof(float))); //--- 노말 속성  
    glEnableVertexAttribArray(1);  
  
    glUseProgram(shaderProgram);  
    unsigned int lightPosLocation = glGetUniformLocation(shaderProgram, "lightPos");          //--- lightPos 값 전달: (0.0, 0.0, 5.0);  
    glUniform3f(lightPosLocation, 0.0, 0.0, 5.0);  
    unsigned int lightColorLocation = glGetUniformLocation(shaderProgram, "lightColor");          //--- lightColor 값 전달: (1.0, 1.0, 1.0) 백색  
    glUniform3f(lightColorLocation, 1.0, 1.0, 1.0);  
    unsigned int objColorLocation = glGetUniformLocation(shaderProgram, "objectColor");          //--- object Color값 전달: (1.0, 0.5, 0.3)의 색  
    glUniform3f(objColorLocation, 1.0, 0.5, 0.3);  
    unsigned int viewPosLocation = glGetUniformLocation(s_program, "viewPos");                  //--- viewPos 값 전달: 카메라 위치  
    glUniform3f(viewPosLocation, cameraPos.x, cameraPos.y, cameraPos.z);  
}
```



퐁 셰이딩: 3) 주변, 산란반사 + 거울반사 조명 (Specular light) 적용

- 버텍스 세이더

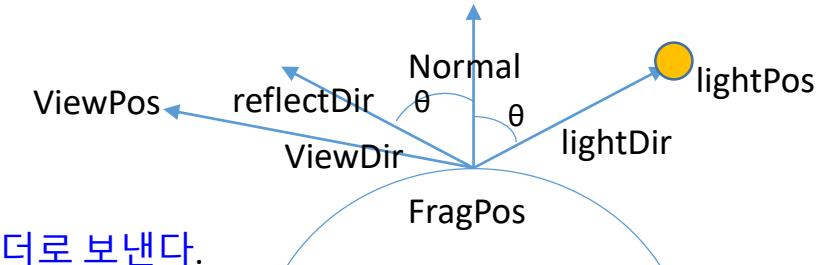
```
#version 330 core

layout (location = 0) in vec3 vPos;
layout (location = 1) in vec3 vNormal;

out vec3 FragPos; //--- 객체의 위치값을 프래그먼트 세이더로 보낸다.
out vec3 Normal; //--- 노멀값을 프래그먼트 세이더로 보낸다.

uniform mat4 model; //--- 모델링 변환값
uniform mat4 view; //--- 뷰잉 변환값
uniform mat4 projection; //--- 투영 변환값

void main()
{
    gl_Position = projection * view * model * vec4(vPos, 1.0);
    FragPos = vec3(model * vec4(vPos, 1.0));
    Normal = vNormal;
}
```



퐁 셰이딩: 3) 주변, 산란반사 + 거울반사 조명 (Specular light) 적용

- 프래그먼트 세이더

```
#version 330 core
in vec4 FragPos;           //--- 노멀값을 계산하기 위해 객체의 위치값을 버텍스 세이더에서 받아온다.
in vec3 Normal;
out vec4 FragColor

void main ()
{
    vec3 ambientLight = 0.3;
    vec3 ambient = ambientLight * lightColor;

    vec3 normalVector = normalize (Normal);
    vec3 lightDir = normalize (lightPos – FragPos);
    float diffuseLight = max (dot (norm, lightDir), 0.0);
    float diffuse = diffuseLight * lightColor;

    int shininess = 128;
    vec3 viewDir = normalize (viewPos – FragPos);
    vec3 reflectDir = reflect (-lightDir, normVector);
    float specularLight = max (dot (viewDir, reflectDir), 0.0);
    specularLight = pow(specularLight, shininess);
    vec3 specular = specularLight * lightColor;

    vec3 result = (ambient + diffuse + specular) * objectColor; //--- 최종 조명 설정된 픽셀 색상: (주변+산란반사+거울반사조명)*객체 색상
    FragColor = vec4 (result, 1.0);                                // --- 픽셀 색을 출력
}
```

//--- 주변 조명 세기
//--- 주변 조명 값

//--- 표면과 조명의 위치로 조명의 방향을 결정한다.
//--- N과 L의 내적 값으로 강도 조절: 음수 방지
//--- 산란 반사 조명값: 산란반사값 * 조명색상값

//--- 광택 계수
//--- 관찰자의 방향
//--- 반사 방향: reflect 함수 - 입사 벡터의 반사 방향 계산
//--- V와 R의 내적값으로 강도 조절: 음수 방지
//--- shininess 승을 해주어 하이라이트를 만들어준다.
//--- 거울 반사 조명값: 거울반사값 * 조명색상값

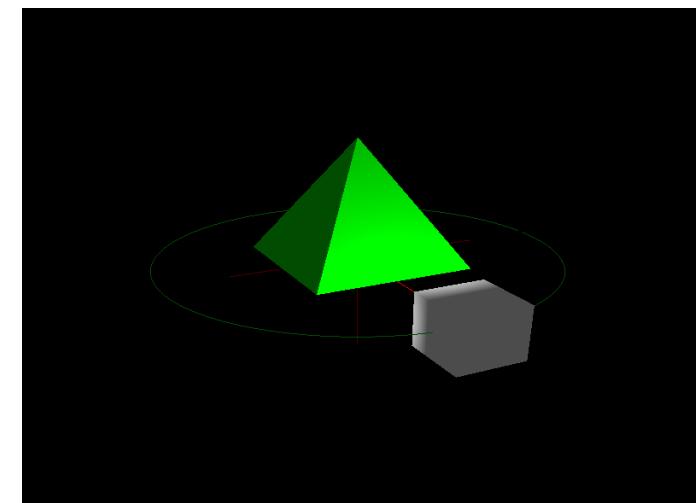
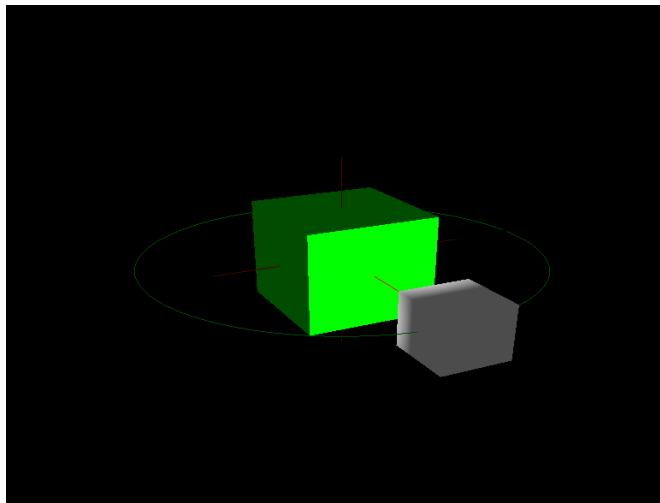
실습 25

- 조명 넣기

- 육면체, 사각뿔 그리기 실습 (실습 16 또는 17에서 한 개)을 변형하여 조명을 적용한다.
- 조명을 객체의 앞 쪽에 놓는다.
 - 조명 위치에 작은 도형을 그려 표시한다.

- 키보드 명령:

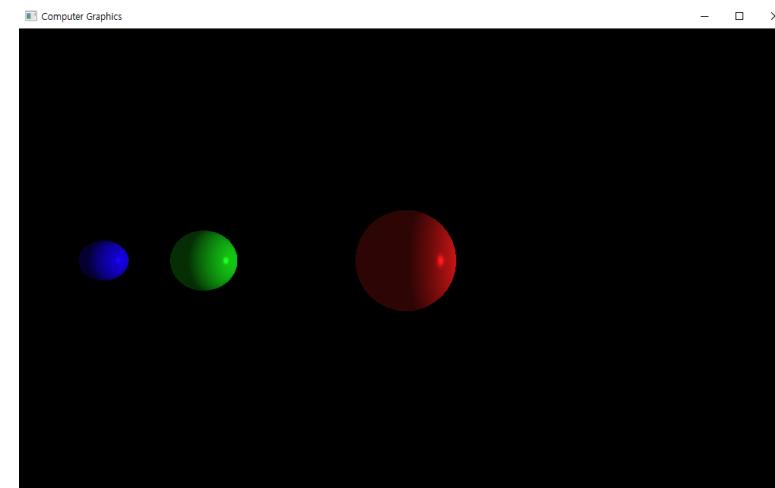
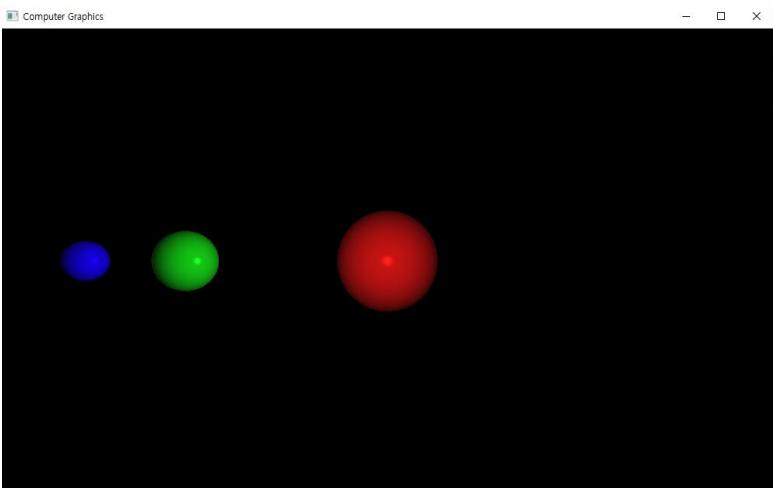
- n: 육면체/사각뿔 그리기
- m: 조명 켜기/끄기
- y: 객체를 y축에 대하여 회전 (제자리에서 자전)
- r: 조명을 객체의 중심 y축에 대하여 양/음 방향으로 공전시키기
 - 조명의 공전 궤도를 원으로 그린다.
- z/Z: 조명을 객체에 가깝게/멀게 이동하기
 - 조명의 위치가 바뀌게 되면 공전 궤도의 크기도 같이 바뀐다.
- q: 프로그램 종료



실습 26

- 조명 넣기

- 다면체 (구)에 조명을 적용한다. (실습 19 이용할 수 있다.)
 - 한 개의 구는 화면 중앙에, 2개의 구는 다른 위치에 놓여있다.
 - 조명이 화면 앞쪽에 놓여있다. 조명의 위치에 작은 객체를 그려 조명의 위치를 확인할 수 있게 한다.
 - 키보드 명령:
 - c: 조명 색을 다른 색으로 바꿔도록 한다. 3종류의 다른 색을 적용해본다.
 - r/R: 조명의 위치를 중심의 구의 y축에 대하여 양/음 방향으로 회전한다.
 - q: 프로그램 종료



이번 주에는

- 조명 모델
 - 주변 조명
 - 산란반사 조명
 - 거울반사 조명
- 다음에는
 - 조명 추가 실습