KWANGWOON UNIVERSITY

컴 퓨 터 그 래 픽 스

HW02 Lighting with Bunny

담당 교수님 : 최민규 교수님

제 출 날 짜 : 2018. 11. 22.

학 과 : 컴퓨터소프트웨어학과

학 번 : 2014707040

이 름 : 유 진 혁

구현 방법 및 결과 화면

(1). Draw the bunny model

- 구현 방법

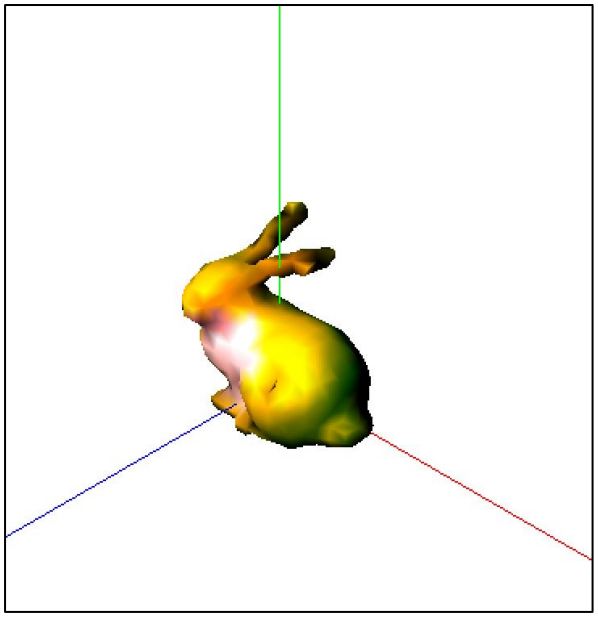
```
ifstream ifs("bunny_origin.txt", ifstream::binary); // Read a file 'bunny_origin.txt'
if (ifs.is_open())
{
    int points, triangles;
    ifs >> points;  // The number of points
    ifs >> triangles;   // The number of triangles
    p.resize(points);
    n.resize(points);
    cnt.resize(points);
    t.resize(3 * triangles);
    // The coordinates of the points are input
    for (int i = 0; i < points; i++)
    {
        ifs >> p[i].x;
        ifs >> p[i].y;
        ifs >> p[i].z;
    }
    // The points of the triangle are input
    for (int i = 0; i < 3 * triangles; i++)
    {
        ifs >> t[i];
        t[i]--;
    }
```

bunny_origin.txt 파일을 읽어 bunny 모델의 각 점들과 삼각형의 점 index를 vector 배열 p, t에 저장하였다.

```
// Draw the Stanford Bunny
void DrawBunny()
{
    glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
    glBegin(GL_TRIANGLES);
    {
        for (size_t i = 0; i < t.size(); i++)
        {
            glNormal3f(n[t[i]].x, n[t[i]].y, n[t[i]].z);
            glVertex3f(p[t[i]].x, p[t[i]].y, p[t[i]].z);
        }
    }
    glEnd();
}
```

배열 p에 들어있는 각 점의 좌표와 GL_TRIANGLES를 이용하여 bunny의 면을 그렸다. vertex를 그리기 전에 glNormal3f 함수로 점의 법선 벡터를 등록시켰다.

- 결과 화면



(2). Draw the vertex normal vectors

- 구현 방법

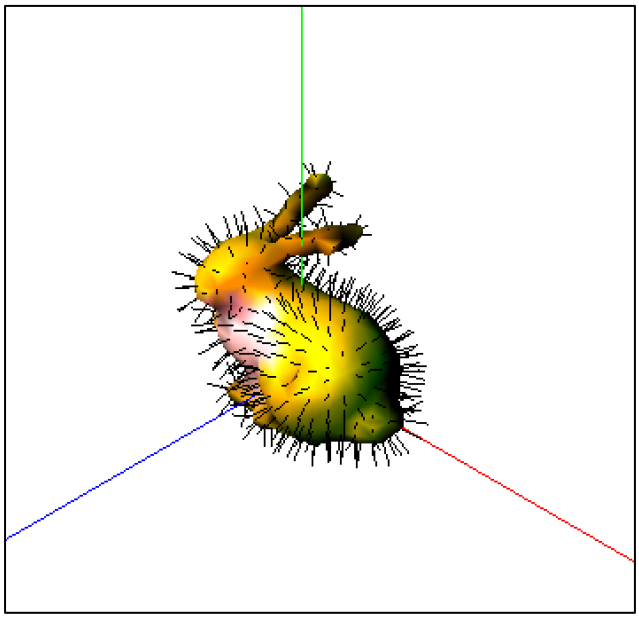
```
for (int i = 0; i < 3 * triangles; i += 3)
{
    // Find and assign normal vectors of planes
    Vector v1, v2, r;
    v1 = p[t[i]].CreateVector(p[t[i + 1]]);
    v2 = p[t[i]].CreateVector(p[t[i + 2]]);
    r = v1 * v2;
    n[t[i]] += r;
    n[t[i + 1]] += r;
    n[t[i + 2]] += r;
    // Assign the number of triangles connected to the vertex
    cnt[t[i]]++;
    cnt[t[i + 1]]++;
    cnt[t[i + 2]]++;
}
// Computing normal vectors of points
for (int i = 0; i < points; i++)
    n[i] /= cnt[i];
```

한 면을 이루는 삼각형의 한 점에서 다른 두 점으로 향하는 벡터를 각각 구하고 이 두 벡터를 외적하여 면의 법선 벡터 r을 구하였다. 그리고 이 r의 값을 세 점의 법선 벡터를 담는 배열 n에 더해주고, 한 점에 연결된 삼각형 면의 수를 파악하기 위해 해당 정보를 담는 배열 cnt의 값을 증가시켰다. 이 과정을 모든 면에 대해 실행한 후, 배열 n에 저장된 값에 배열 cnt에 저장된 값을 나눠 점의 법선 벡터를 구하였다.

```
// Draw normal vectors of points
void DrawVertexNormal()
{
    glColor3f(0, 0, 0); // black
    glBegin(GL_LINES);
    {
        for (size_t i = 0; i < p.size(); i++)
        {
            glVertex3f(p[i].x, p[i].y, p[i].z);
            glVertex3f((p[i] + (n[i] / 10)).x, (p[i] + (n[i] / 10)).y, (p[i] + (n[i] / 10)).z);
        }
    }
    glEnd();
}
```

점의 법선 벡터를 시각적으로 표현하기 위해, GL_LINES를 이용하여 bunny 모델의 한 점에서 그 점의 법선 벡터만큼 이동한 점을 잇는 선을 그렸다. 선이 너무 길게 그려지지 않도록 실제 법선 벡터의 길이의 1/10배로 그려지게 하였다.

- 결과 화면



(3), (4), (5). Make directional light rotate them, and draw their geometry

- 구현 방법

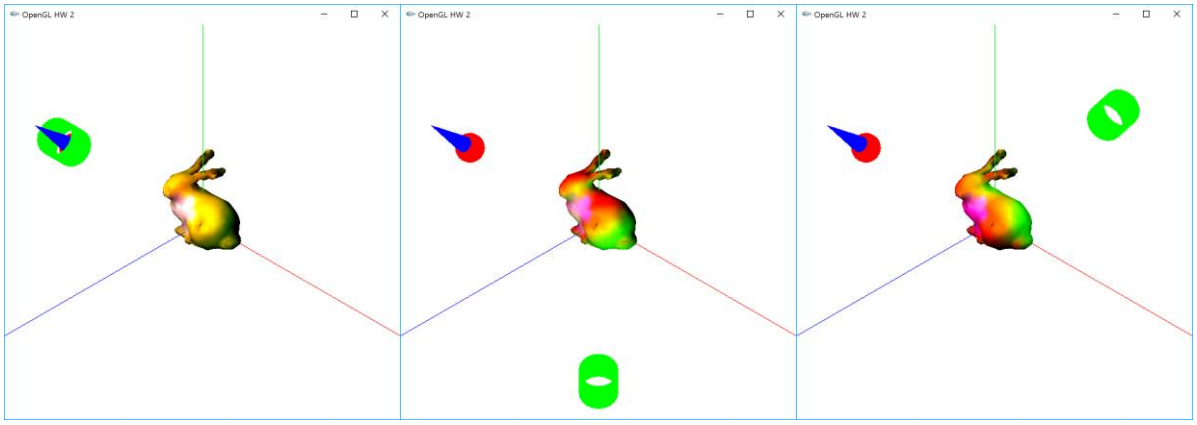
```
// Directional light
glPushMatrix();
glRotatef(thetaD, 1, 1, 1); // Rotate light
GLfloat positionD[4] = { -0.224745f, 1.61237f, 1.61237f, 0 };
GLfloat ambientD[4] = { 0, 0, 0, 1 };
GLfloat diffuseD[4] = { 0, 1, 0, 1 };
GLfloat specularD[4] = { 0, 1, 0, 1 };  // The green light is directional light

glLightfv(GL_LIGHT1, GL_POSITION, positionD);
glLightfv(GL_LIGHT1, GL_AMBIENT, ambientD);
glLightfv(GL_LIGHT1, GL_DIFFUSE, diffuseD);
glLightfv(GL_LIGHT1, GL_SPECULAR, specularD);
glPopMatrix();
//
```

Directional light는 GL_LIGHT1에 구현하였다. 먼저, glEnable(GL_LIGHTING)와 glEnable(GL_LIGHT1)로 조명을 활성화하였다. 조명의 방향을 지정하기 전에 (1, 1, 1) 중심으로 회전시켜야 하므로 glPushMatrix로 현재 좌표계를 저장하였다. glRotatef로 회전 변환 행렬을 등록시키고 조명의 위치와 광원 정보를 설정, 등록하였다. 방향만 갖는 Directional light기 때문에 GL_POSITION의 마지막 요소인 w 값을 0으로 설정하였고, 다른 조명들과 구분하기 위해 광원의 색을 초록색으로 설정하였다. 설정 후 원래 좌표계로 돌아가기 위해 glPopMatrix로 등록된 행렬을 제거하였다.

```
glPushMatrix();
glColor3f(0, 1, 0); // Green Directional light
GLUquadric* cylinder = gluNewQuadric(); // Create a cylinder
glRotatef(thetaD, 1, 1, 1); // Rotate the cylinder
glTranslatef(-0.224745f, 1.61237f, 1.61237f);   // Translate the cylinder
glRotatef(-45, 1, 0, 0);    //  Rotate to look at the bunny
gluCylinder(cylinder, 0.2f, 0.2f, 0.4f, 20, 20);    // Draw the cylinder
glPopMatrix();
```

Directional light의 광원 방향을 기하적으로 표현하기 위해 cylinder를 그렸다. 우선, 현재 좌표계를 glPushMatrix로 저장하고 그리는 색을 초록색을 설정하였다. cylinder를 생성한 뒤, 이 cylinder가 directional light 방향대로 회전하도록 회전 변환과 이동 변환을 해주었다. 또, cylinder가 bunny를 바라보면서 회전해야 하므로 -45도만큼 회전시키는 변환도 해주었다. cylinder를 그리고, glPopMarix로 등록한 변환을 해제하였다.

- 결과 화면



초록색 cylinder가 향하는 방향으로 초록색 directional light가 비춘다.

(6), (7), (8). Make point light, rotate them, and draw their geometry

- 구현 방법

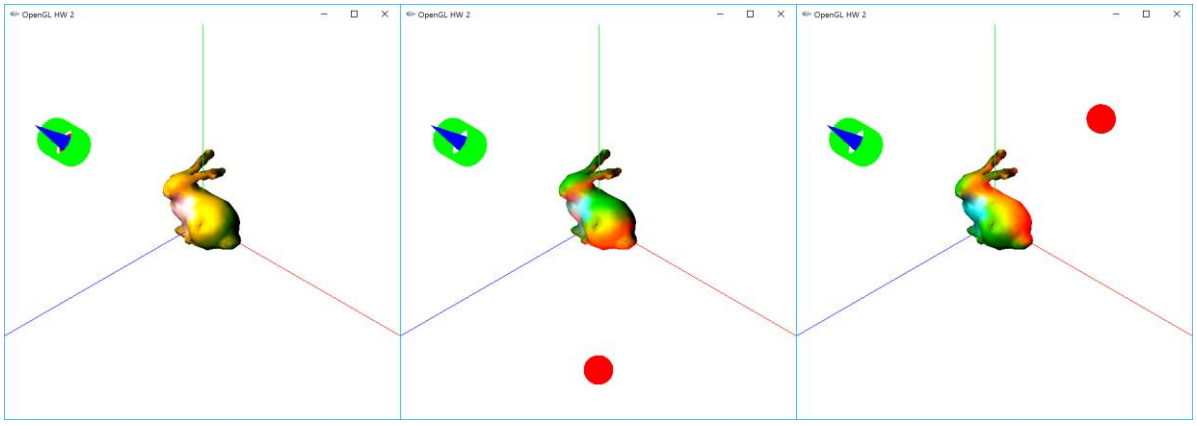
```
// Point light
glPushMatrix();
glRotatef(thetaP, 1, 1, 1); // Rotate light
GLfloat positionP[4] = { -0.224745f, 1.61237f, 1.61237f, 1 };
GLfloat ambientP[4] = { 0, 0, 0, 1 };
GLfloat diffuseP[4] = { 1, 0, 0, 1 };
GLfloat specularP[4] = { 1, 0, 0, 1 };  // The red light is point light

glLightfv(GL_LIGHT0, GL_POSITION, positionP);
glLightfv(GL_LIGHT0, GL_AMBIENT, ambientP);
glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuseP);
glLightfv(GL_LIGHT0, GL_SPECULAR, specularP);

glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, 0.2);
glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, 0.1);
glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, 0.05);
glPopMatrix();
//
```

Point light는 GL_LIGHT0에 구현하였다. 먼저, 조명을 활성화하고 glPushMatrix로 현재 좌표계를 저장하였다. glRotatef로 회전 변환 행렬을 등록시키고 조명의 위치와 광원 정보를 설정, 등록하였다. Point light은 색을 빨간색으로 하였고, Directional light와는 달리 광원의 위치를 가져야 하므로 GL_POSITION의 마지막 요소 w를 1로 설정하였다. 또, 광원과의 거리에 따라 빛의 밝기도 약해져야 하므로 감쇠 설정도 해주었다. 설정 후 원래 좌표계로 돌아가기 위해 glPopMatrix로 등록된 행렬을 제거하였다.

```
// Draw the geometry of lights
glPushMatrix();
glColor3f(1, 0, 0); // Red point light
GLUquadric* sphere = gluNewQuadric(); // Create a sphere
glRotatef(thetaP, 1, 1, 1); // Rotate the sphere
glTranslatef(-0.224745f, 1.61237f, 1.61237f);   // Translate the sphere
gluSphere(sphere, 0.15f, 20, 20);  // Draw the sphere
glPopMatrix();
```

Point light의 광원 위치를 기하적으로 표현하기 위해 sphere를 그렸다. 우선, 현재 좌표계를 glPushMatrix로 저장하고 그리는 색을 빨간색을 설정하였다. sphere를 생성한 뒤, 이 sphere가 point light 위치에 맞게 움직이도록 회전 변환과 이동 변환을 해주었다. sphere를 그리고, glPopMarix로 등록한 변환을 해제하였다.

- 결과 화면



빨간색 sphere의 위치에서 point light가 비춘다. 첫 번째 그림을 보면, point light와 directional light가 같은 표면을 비출 때 point light의 감쇠효과로 인해 directional light가 비추는 영역이 더 넓은 것을 볼 수 있다.

(9), (10), (11). Make spot light, rotate them, and draw their geometry

- 구현 방법

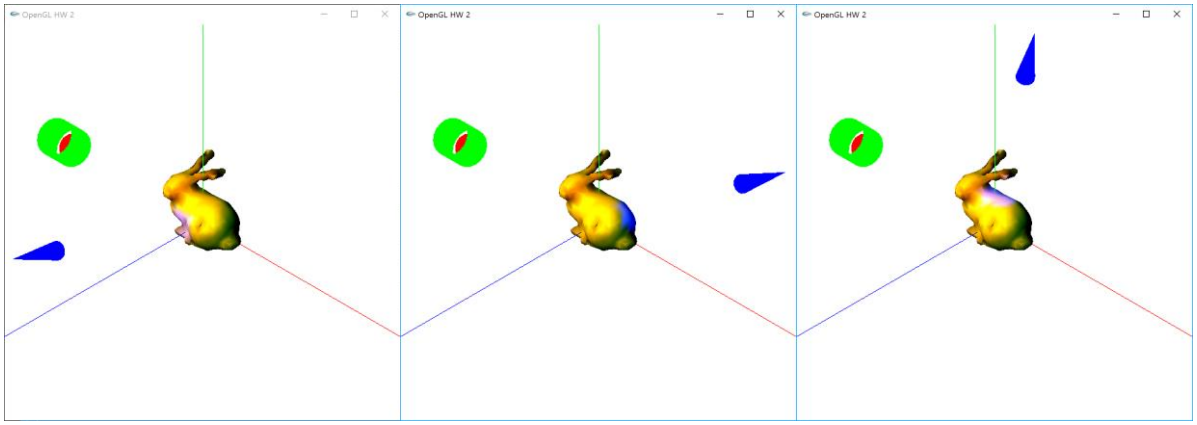
```
// Spot light
glPushMatrix();
glRotatef(thetaS, 1, 1, 1); // Rotate light
GLfloat positionS[4] = { -0.224745f, 1.61237f, 1.61237f, 1 };
GLfloat ambientS[4] = { 0, 0, 0, 1 };
GLfloat diffuseS[4] = { 0, 0, spotShine, 1 };
GLfloat specularS[4] = { 0, 0, spotShine, 1 };  // The blue light is spot light

glLightfv(GL_LIGHT2, GL_POSITION, positionS);
glLightfv(GL_LIGHT2, GL_AMBIENT, ambientS);
glLightfv(GL_LIGHT2, GL_DIFFUSE, diffuseS);
glLightfv(GL_LIGHT2, GL_SPECULAR, specularS);

GLfloat spot_direction[3] = { 0.224745f, -1.61237f, -1.61237f };
glLightfv(GL_LIGHT2, GL_SPOT_DIRECTION, spot_direction);
glLightf(GL_LIGHT2, GL_SPOT_CUTOFF, spotRange);
glLightf(GL_LIGHT2, GL_SPOT_EXPONENT, 1.0);
glPopMatrix();
```

Spot light는 GL_LIGHT2에 구현하였다. 먼저, 조명을 활성화하고 glPushMatrix로 현재 좌표계를 저장하였다. glRotatef로 회전 변환 행렬을 등록시키고 조명의 위치와 광원 정보를 설정, 등록하였다. 광원의 색은 파란색으로 하였다. Spot light는 광원의 위치와 방향을 전부 가져야 하므로 point light처럼 광원 위치를 지정해주고 GL_SPOT_DIRECTION으로 spot light가 비추는 방향을 지정해주었다. GL_SPOT_CUTOFF로 빛이 비추는 범위를, GL_SPOT_EXPONENT로 경계를 설정한 후 원래 좌표계로 돌아가기 위해 glPopMatrix로 등록된 행렬을 제거하였다.

```
glPushMatrix();
glColor3f(0, 0, 1); // Blue spot light
GLUquadric* cylinderZeroTop = gluNewQuadric();  // Create a cylinder with zero top
glRotatef(thetaS, 1, 1, 1); // Rotate the cylinder
glTranslatef(-0.224745f, 1.61237f, 1.61237f);   // Translate the cylinder
glRotatef(-45, 1, 0, 0);    // Rotate to look at the bunny
gluCylinder(cylinderZeroTop, 0.1f, 0, 0.8f, 20, 20);    // Draw the cylinder
glPopMatrix();
```

Spot light의 광원의 위치와 방향을 기하적으로 표현하기 위해 윗면의 반지름이 0인 cylinder를 그렸다. 우선, 현재 좌표계를 glPushMatrix로 저장하고 그리는 색을 파란색을 설정하였다. cylinder를 생성한 뒤, 이 cylinder가 Spot light의 위치에 맞게 움직이도록 회전 변환과 이동 변환을 해주었다. 또, cylinder의 아랫면이 bunny를 바라보면서 회전해야 하므로 -45도만큼 회전시키는 변환도 해주었다. 윗면의 반지름 길이를 0으로 하여 cylinder를 그리고, glPopMarix로 등록한 변환을 해제하였다.

- 결과 화면



윗면이 0인 파란색 cylinder의 위치에서 cylinder가 향하는 방향으로 파란색 spot light가 비춘다. 초기 cut-off 범위는 5도이고, 빛의 밝기는 1.0이다.

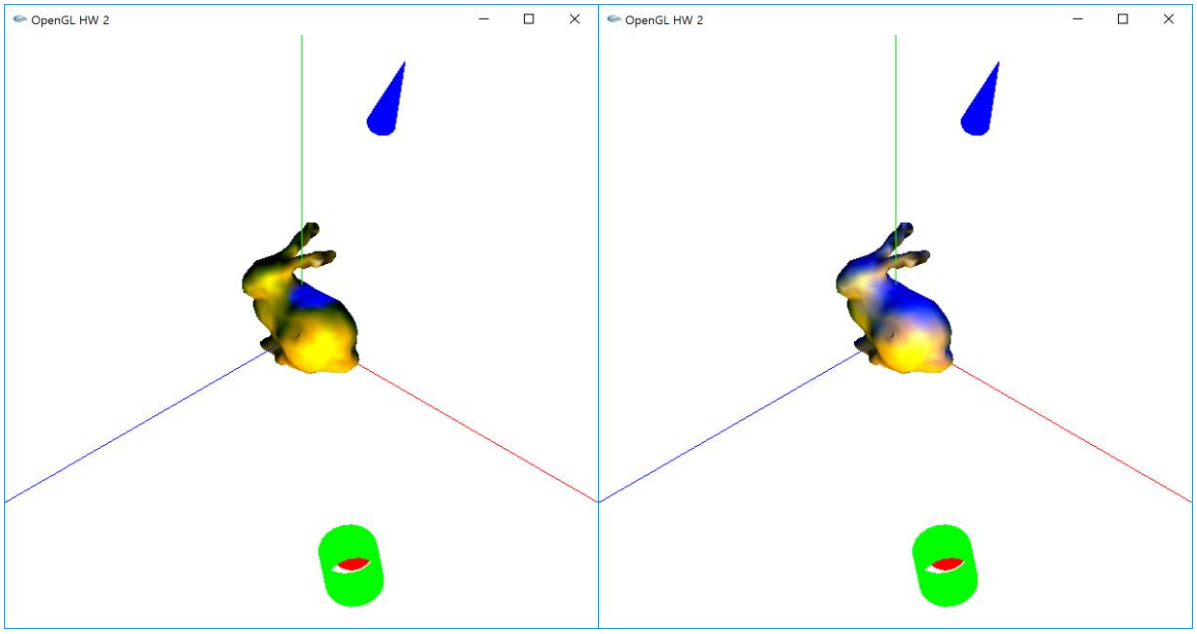
(12), (13). Time-varying cut-off angle of the spot light

- 구현 방법

```
if (toggleSpotCutOff)
{
    if (spotRange >= 25 || spotRange <= 5)
        signCutOff *= -1;
    spotRange += signCutOff * 0.111f;
}
```

1/90초마다 실행되는 timer 함수 코드의 일부이다. Spot light의 cut-off 각도를 5도에서 25도 사이에서 변화시켰다. GL_SPOT_CUTOFF에 설정되는 변수 spotRange의 증가와 감소가 번갈아가며 일어난다.

- 결과 화면



(14). Time-varying shininess coefficient of the spot light

- 구현 방법

```
if (toggleSpotShininess)
{
    if (spotShine <= 0.0f || spotShine >= 1.0f)
        signShininess *= -1;
    spotShine += signShininess * 0.006f;
}
```

1/90초마다 실행되는 timer 함수 코드의 일부이다. Spot light의 diffuse 값과 specular 값을 0과 1 사이에서 조절했다. 변수 spotShine은 GL_DIFFUSE와 GL_SPECULAR의 Blue 값을 대체하는 변수이고 증가와 감소가 번갈아가며 일어난다.

- 결과 화면