컴퓨터그래픽스 HW_2

소프트웨어학부

2015726056

조동희

self-scoring table

	1	2	3	4	5	6	Total
Score	1/1	1/1	1/1	1/1	1/1	1/1	6/6

1. Draw a torus model

과제1에서와 같이 18 x 36 벡터 배열에 점들을 저장하고 그렸다. 이때 빛의 다양한 표현을 잘 확인하기 위해 빛의 회전 경로를 고려하여 시점을 설정했다. Phong Model은 diffuse(난반사), Specular(하이라이트), ambient(간접광)을 뜻하며 각각의 인자에 따라 성질이 달라진다. 그리고 phong 모델은 아래의 식을 만족한다.

$$I = k_a I_a + k_d I_l (N \cdot L) + k_s I_l (R \cdot V)^n$$

torus의 Material을 다음과 같이 설정해주었다.

```
void setupWhiteShinyMaterial()
{
    // Material
    GLfloat mat_ambient[4]{0.1, 0.1, 0.1, 1 };
    GLfloat mat_diffuse[4]{ 1, 1, 1, 1 };
    GLfloat mat_specular[4] = { 1, 1, 1, 1 };
    GLfloat mat_shininess = 100;
    glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, mat_ambient);
    glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, mat_diffuse);
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, mat_specular);
    glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, mat_shininess);
}
```

```
glEnable(GL_NORMALIZE);
```

특정 광원을 설정하기 위해서는 법선벡터와 단위법선 벡터를 계산할 수 있어야 한다. OpenGL에서는 GL_NORMALIZE를 통해 자동적으로 단위법선벡터를 구할 수 있도록 도와준다.

2. Draw the vertex normal vectors

vertex의 법선 벡터는 다음과 같다. 4개로 둘러싸인 사각형의 법선벡터의 평균을 vertex의 법선벡터로 정의했다.

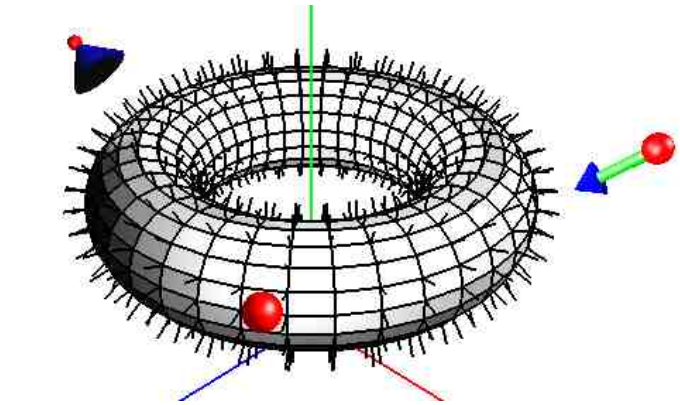
```
void drawNormalVector() {
    glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
    for (int i = 0; i < numHorizon; i++) {
        for (int j = 0; j < numVertical; j++) {
            vec3 p1(p[i][(j + 1) % 18].x - p[i][j].x, p[i][(j + 1) % 18].y - p[i][j].y, p[i][(j + 1) % 18].z - p[i][j].z
            vec3 p2(p[(i + 1) % 36][j].x - p[i][j].x, p[(i + 1) % 36][j].y - p[i][j].y, p[(i + 1) % 36][j].z - p[i][j].z
            vec3 normalVector1= cross(p2, p1);
            ////////////////////////////////////////////////////////
            vec3 p3(p[(i + 1) % 36][(j + 1) % 18].x - p[i][(j + 1) % 18].x,
                    p[(i + 1) % 36][(j + 1) % 18].y - p[i][(j + 1) % 18].y,
                    p[(i + 1) % 36][(j + 1) % 18].z - p[i][(j + 1) % 18].z);

            vec3 p4(p[i][(j + 2) % 18].x - p[i][(j + 1) % 18].x,
                    p[i][(j + 2) % 18].y - p[i][(j + 1) % 18].y,
                    p[i][(j + 2) % 18].z - p[i][(j + 1) % 18].z);
            vec3 normalVector2 = cross(p2, p1);
            ////////////////////////////////////////////////////////
            vec3 p5(p[(i + 2) % 36][(j + 1) % 18].x - p[(i + 1) % 36][(j + 1) % 18].x,
                    p[(i + 2) % 36][(j + 1) % 18].y - p[(i + 1) % 36][(j + 1) % 18].y,
                    p[(i + 2) % 36][(j + 1) % 18].z - p[(i + 1) % 36][(j + 1) % 18].z);

            vec3 p6(p[(i + 1) % 36][(j + 2) % 18].x - p[(i + 1) % 36][(j + 1) % 18].x,
                    p[(i + 1) % 36][(j + 2) % 18].y - p[(i + 1) % 36][(j + 1) % 18].y,
                    p[(i + 1) % 36][(j + 2) % 18].z - p[(i + 1) % 36][(j + 1) % 18].z);
            vec3 normalVector3 = cross(p2, p1);
            ////////////////////////////////////////////////////////
            vec3 p7(p[(i + 2) % 36][j].x - p[(i + 1) % 36][j].x,
                    p[(i + 2) % 36][j].y - p[(i + 1) % 36][j].y,
                    p[(i + 2) % 36][j].z - p[(i + 1) % 36][j].z);

            vec3 p8(p[(i + 1) % 36][(j + 1) % 18].x - p[(i + 1) % 36][j].x,
                    p[(i + 1) % 36][(j + 1) % 18].y - p[(i + 1) % 36][j].y,
                    p[(i + 1) % 36][(j + 1) % 18].z - p[(i + 1) % 36][j].z);
            vec3 normalVector4 = cross(p2, p1);
            ////////////////////////////////////////////////////////
            vec3 normalVector = (normalVector1 + normalVector2 + normalVector3 + normalVector4);
            vec3 middleVector = p[(i + 1) % 36][(j + 1) % 18];
            glBegin(GL_LINES);
            glColor3f(0.0f, 0.0f, 0.0f);
            glVertex3f(middleVector.x, middleVector.y, middleVector.z);
            glVertex3f(
                2*normalVector.x + middleVector.x,
                2*normalVector.y + middleVector.y,
                2*normalVector.z + middleVector.z
            );
        }
```

이는 각각 사각형을 구성하는 벡터의 외적의 평균을 구하는 코드이다. 이를 해당 vertex와의 차로 도형에 표현했다.

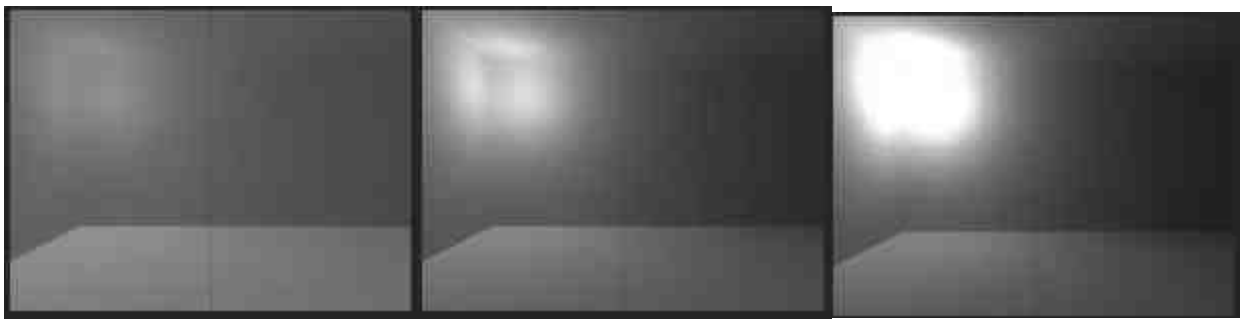


3. Point light, Directional light, Spot light

point light는 하나의 점에서부터 빛 에너지가 시작되고, 점 광원에서 가장 중요한 것은 위치와 방향, 그리고 뿜어져 나오는 빛의 색이다. 이 3가지 요소의 영향을 받아 객체로 직접 빛을 쏘는 것이다. directional light는 방향 광원으로 방향만 존재한다.

```
void setupLight(const vec4& p, int i)
{
    GLfloat ambient[4]{ 0.1, 0.1, 0.1, 1 };
    GLfloat diffuse[4] = { 1.0, 1.0, 1.0, 1 };
    GLfloat specular[4] = { 1.0, 1.0, 1.0, 1 };
    glLightfv(GL_LIGHT0 + i, GL_AMBIENT, ambient);
    glLightfv(GL_LIGHT0 + i, GL_DIFFUSE, diffuse);
    glLightfv(GL_LIGHT0 + i, GL_SPECULAR, specular);
    glLightfv(GL_LIGHT0 + i, GL_POSITION, value_ptr(p));
    // Attenuation for the point light
    if (i == 0 && attenuation)
    {
        glLightf(GL_LIGHT0 + i, GL_CONSTANT_ATTENUATION, 1.0);
        glLightf(GL_LIGHT0 + i, GL_LINEAR_ATTENUATION, 0.1);
        glLightf(GL_LIGHT0 + i, GL_QUADRATIC_ATTENUATION, 0.05);
    }
    else { // Default value
        glLightf(GL_LIGHT0 + i, GL_CONSTANT_ATTENUATION, 1.0);
        glLightf(GL_LIGHT0 + i, GL_LINEAR_ATTENUATION, 0.0);
        glLightf(GL_LIGHT0 + i, GL_QUADRATIC_ATTENUATION, 0.0);
    }
    if (i == 2) // Spot light
    {
        vec3 spotDirection = -vec3(p);
        glLightfv(GL_LIGHT0 + i, GL_SPOT_DIRECTION, value_ptr(spotDirection));
        glLightf(GL_LIGHT0 + i, GL_SPOT_CUTOFF, cutoffValue ); // (0, 90]
        glLightf(GL_LIGHT0 + i, GL_SPOT_EXPONENT, exponentValue); // [0, 128]
    }
    else { // Point and distant light.
          // 180 to turn off cutoff when it was used as a spot light.
        glLightf(GL_LIGHT0 + i, GL_SPOT_CUTOFF, 180); // uniform light distribution
    }
}
```

GL_CONSTANT_ATTENUATION, GL_LINEAR_ATTENUATION, GL_QUADRATIC_ATTENUATION



위와 같이 지속적인 감쇠와 선형적 감쇠 그리고 이차적인 감쇠를 나타낸다. setupLight를 통해 point, spot, distant light를 설정한다.

```
void animate() {
    frame += 1;
    if (rotationLight)
    {
        for (int i = 0; i < 3; i++)
            if (lightOn[i]) thetaLight[i] += 4 / period; // degree
        if (lightOn[2] && cutoff)
            cutoffNorm += radians(4.0 / period) / M_PI;
        cutoffValue = cutoffMax * (acos(cos(cutoffNorm * M_PI)) / M_PI);
    }
}
```

Space를 통해 회전을 시작하면 rotationLight이 활성화되며 cutoff를 각도에 맞춰 설정하고 render를 통해 그리게 된다.

```
void computeRotation(const vec3& a, const vec3& b, float& theta, vec3& axis)
{
    axis = cross(a, b);
    float sinTheta = length(axis);
    float cosTheta = dot(a, b);
    theta = atan2(sinTheta, cosTheta) * 180 / M_PI;
}
```

이러한 빛의 종류들은 computeRotation을 통해 회전한다.

```
vec4 lightP[3];
for (int i = 0; i < 3; i++)
{
    // Just turn off the i-th light, if not lit
    if (!lightOn[i]) { glDisable(GL_LIGHT0 + i); continue; }
    // Turn on the i-th light
    glEnable(GL_LIGHT0 + i);
    // Dealing with the distant light
    lightP[i] = lightInitialP;
    if (i == 1) lightP[i].w = 0;
    mat4 R = rotate(mat4(1.0), radians(thetaLight[i]), axis);
    // Lights rotate around the center of the world coordinate system
    lightP[i] = R * lightP[i];
    // Set up the i-th light
    setupLight(lightP[i], i);
}
```

이렇게 설정된 빛들은 렌더링 과정에서 위와 같은 방식으로 그려진다.

6. Time-varying shiniess coefficient in specular reflection

cutoff를 활성화하며 cutoffNorm을 조정하고 cutoffValue를 계산하여 시간에 따라 spot light를 변화시킨다.

```
void drawSpotLight(const vec3& p, float cutoff)
{

    glPushMatrix();
    glTranslatef(p.x, p.y, p.z);
    float theta;
    vec3 axis;

    computeRotation(vec3(0, 0, 1), vec3(0, 0, 0) - vec3(p), theta, axis);
    glRotatef(theta, axis.x, axis.y, axis.z);
    // Color

    setupColoredMaterial(vec3(0, 0, 1));
    // tan(cutoff) = r/h
    float h = 0.15;
    float r = h * tan(radians(cutoff));
    drawCone(r, h, 16, 5);
    // Color
    setupColoredMaterial(vec3(1, 0, 0));
    // Apex
    float apexRadius = 0.06 * (0.5 + exponentValue / 128.0);
    drawSphere(apexRadius, 16, 16);
    glPopMatrix();
}
```

```
case GLFW_KEY_T: cutoff = !cutoff; break;
```