POLITECHNIKA WROCŁAWSKA

Instytut Informatyki, Automatyki i Robotyki

Zakład Systemów Komputerowych

Wprowadzenie do grafiki komputerowej

Kurs: INE4234L

Sprawozdanie z ćwiczenia nr 5

TEMAT ĆWICZENIA OpenGL - oświetlanie scen 3-D

Wykonał:	**Bartłomiej Sawicki**
Termin:	**PN/P 7.30-10.30**
Data wykonania ćwiczenia:	**6.12.2021r.**
Data oddania sprawozdania:	**13.12.2021r.**
Ocena:	

Uwagi prowadzącego:

1 Wstęp teoretyczny

Dodanie oświetlenia na scenie odbywa się przez zdefiniowanie zachowania oświetlanego obiektu. Istnieją różne modele zachowywania się oświetlanych obiektów, ale na potrzeby zadania laboratoryjnego wybrałem model Phonga.

1.1 Model Phonga

Model Phonga - model oświetlenia stosowany w grafice komputerowej, służący do modelowania odbić zwierciadlanych od obiektów. Model Phonga opisuje światło odbite przy pomocy trzech składowych:

- światło otoczenia (ang. *specular*)
- światło rozproszone (ang. *diffuse*)
- światło odbite (ang. *ambient*)

Dla każdego obiektu należy obliczyć wektory normalne do powierzchni aby system oświetlenia mógł wykonać obliczenia dotycznące intensywności światła.

1.2 Wektory normalne

Wyznaczenie wektorów normalnych do powierzchni jajka odbyło się przy pomocy poniższych wzorów.

$$x_u = \frac{\partial x(u,v)}{\partial u} = \left(-450u^4 + 900u^3 - 810u^2 + 360u - 45\right) \cdot \cos(\pi v)$$

$$x_v = \frac{\partial x(u,v)}{\partial v} = \pi \cdot \left(90u^5 - 225u^4 + 270u^3 - 180u^2 + 45u\right) \cdot \sin(\pi v)$$

$$y_u = \frac{\partial y(u,v)}{\partial u} = 640u^3 - 960u^2 + 320u$$

$$y_v = \frac{\partial y(u,v)}{\partial v} = 0$$

$$z_u = \frac{\partial z(u,v)}{\partial u} = \left(-450u^4 + 900u^3 - 810u^2 + 360u - 45\right) \cdot \sin(\pi v)$$

$$z_v = \frac{\partial z(u,v)}{\partial v} = -\pi \cdot \left(90u^5 - 225u^4 + 270u^3 - 180u^2 + 45u\right) \cdot \cos(\pi v)$$

Po wykonaniu obliczeń składowych wektora normalnego należało dokonać normalizacji tego wektora. Normalizacja polegała na podzieleniu wszystkich wartości składowych wektora przez długość wektora. W efekcie wektor normalny ma długość równą 1.

1.3 Materiały

Aby zmienić wygląd wyświetlanego obiektu można wykorzytsać materiały. Wybrany materiał pokrywa powierzchnię figury dzięki czemu obiekt może reagować w inny sposób na światło. Materiał jest opisany przy pomocy trzech składowych z modelu Phonga oraz połysku.

2 Realizacja zadania 1.

Modyfikacja programu obracające się jajko przez:

1. Wprowadzenie na scenę jednego źródła światła

```
/*************************************************************************************/
// Definicja źródła światła

GLfloat light_position[] = { 0.0, 0.0, 10.0, 1.0 };
// położenie źródła

GLfloat light_ambient[] = { 1.0, 1.0, 0.0, 1.0 };
// składowe intensywności świecenia źródła światła otoczenia
// Ia = [Iar,Iag,Iab]

GLfloat light_diffuse[] = { 1.0, 1.0, 0.0, 1.0 };
// składowe intensywności świecenia źródła światła powodującego
// odbicie dyfuzyjne Id = [Idr,Idg,Idb]

GLfloat light_specular[] = { 1.0, 1.0, 0.0, 1.0 };
// składowe intensywności świecenia źródła światła powodującego
// odbicie kierunkowe Is = [Isr,Isg,Isb]

GLfloat att_constant = { 1.0 };
// składowa stała ds dla modelu zmian oświetlenia w funkcji
// odległości od źródła

GLfloat att_linear = { 0.05 };
// składowa liniowa dl dla modelu zmian oświetlenia w funkcji
// odległości od źródła

GLfloat att_quadratic = { 0.001 };
// składowa kwadratowa dq dla modelu zmian oświetlenia w funkcji
// odległości od źródła
```

```
glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
glLightfv(GL_LIGHT0, GL_POSITION, light_position);

glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, att_constant);
glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, att_linear);
glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, att_quadratic);
```

2. Zdefiniowane materiału z jakiego wykonane jest jajko

```
/*****************************************************************************/
// Definicja materiału z jakiego zrobiony jest obiekt

GLfloat mat_ambient[] = { 0.1, 0.1, 0.1, 1.0 };
// współczynniki ka =[kar,kag,kab] dla światła otoczenia

GLfloat mat_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
// współczynniki kd =[kdr,kdg,kdb] światła rozproszonego

GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
// współczynniki ks =[ksr,ksg,ksb] dla światła odbitego

GLfloat mat_shininess = { 20.0 };
// współczynnik n opisujący połysk powierzchni
```

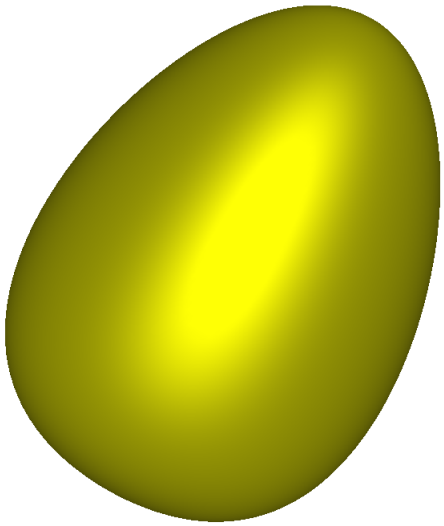
3. Oblczenie wektorów normalnych

```
float xu, xv, yu, yv, zu, zv;
xu = (-450 * pow(u, 4) + 900 * pow(u, 3) - 810 * pow(u, 2) + 360 * u - 45) * cos(pi * v);
xv = pi * (90 * pow(u, 5) - 255 * pow(u, 4) + 270 * pow(u, 3) - 180 * pow(u, 2) + 45 * u) * sin(pi * v);
yu = 640 * pow(u, 3) - 960 * pow(u, 2) + 320 * u;
yv = 0;
zu = (-450 * pow(u, 4) + 900 * pow(u, 3) - 810 * pow(u, 2) + 360 * u - 45) * sin(pi * v);
zv = -pi * (90 * pow(u, 5) - 255 * pow(u, 4) + 270 * pow(u, 3) - 180 * pow(u, 2) + 45 * u) * cos(pi * v);

float x, y, z;
x = yu * zv - zu * yv;
y = zu * xv - xu * zv;
z = xu * yv - yu * xv;

float vectorLen;
vectorLen = sqrt(pow(x, 2) + pow(y, 2) + pow(z, 2));

if (vectorLen != 0)
{
    x /= vectorLen;
    y /= vectorLen;
    z /= vectorLen;
}
//zmiana zwrotu wektora normalnego
if (i < (n / 2) - (n / 16))
{
    x *= -1;
    y *= -1;
    z *= -1;
}
normal[i][j][0] = x;
normal[i][j][1] = y;
normal[i][j][2] = z;
```

4. Dodanie dla poszczególnych wierzchołków modelu jajka informacji o wektorach normalnych

```
glBegin(GL_TRIANGLES);
for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        if (j + 1 < n)
        {
            glNormal3fv(normal[i][j]);
            glVertex3fv(array[i][j]);
            glNormal3fv(normal[(i + 1) % n][j]);
            glVertex3fv(array[(i + 1) % n][j]);
            glNormal3fv(normal[(i + 1) % n][j + 1]);
            glVertex3fv(array[(i + 1) % n][j + 1]);
        }
        else
        {
            //laczenie trojkatow w osi poziomej
            glNormal3fv(normal[i][j]);
            glVertex3fv(array[i][j]);
            glNormal3fv(normal[(i + 1) % n][j]);
            glVertex3fv(array[(i + 1) % n][j]);
            glNormal3fv(normal[(n - (i + 1)) % n][0]);
            glVertex3fv(array[(n - (i + 1)) % n][0]);
        }
        if (j + 1 < n)
        {
            glNormal3fv(normal[i][j]);
            glVertex3fv(array[i][j]);
            glNormal3fv(normal[i][j + 1]);
            glVertex3fv(array[i][j + 1]);
            glNormal3fv(normal[(i + 1) % n][j + 1]);
            glVertex3fv(array[(i + 1) % n][j + 1]);
        }
        else
        {
            //laczenie trojkatow w osi poziomej
            glNormal3fv(normal[i][j]);
            glVertex3fv(array[i][j]);
            glNormal3fv(normal[(n - i) % n][0]);
            glVertex3fv(array[(n - i) % n][0]);
            glNormal3fv(normal[(n - (i + 1)) % n][0]);
            glVertex3fv(array[(n - (i + 1)) % n][0]);
        }
    }
}
glEnd();
```

Efekt działania programu



3 Realizacja zadania 2.

Zadanie polegało na napisaniu programu, pozwalającego na oświetlanie modelu jajka przy pomocy dwóch źródeł barwnego światła i umożliwieniu manipulowania położeniem źródeł przy pomocy myszy.

1. Opis materiału i źródeł światła

```
GLfloat mat_ambient[] = { 0.1, 0.1, 0.1, 1.0 };
// współczynniki ka =[kar,kag,kab] dla światła otoczenia

GLfloat mat_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
// współczynniki kd =[kdr,kdg,kdb] światła rozproszonego

GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
// współczynniki ks =[ksr,ksg,ksb] dla światła odbitego

GLfloat mat_shininess = { 20.0 };
// współczynnik n opisujący połysk powierzchni

/****************************************************************************/
// Definicja źródła światła

GLfloat light_position[] = { light1[0], light1[1], light1[2], 1.0 };
// położenie źródła


GLfloat light_ambient1[] = { 1, 0.0, 0.0, 1.0 };
GLfloat light_ambient2[] = { 0.0, 0.0, 1, 1.0 };
// składowe intensywności świecenia źródła światła otoczenia
// Ia = [Iar,Iag,Iab]

GLfloat light_diffuse1[] = { 1, 0, 0, 1 };
GLfloat light_diffuse2[] = { 0, 0, 1, 1 };
// składowe intensywności świecenia źródła światła powodującego
// odbicie dyfuzyjne Id = [Idr,Idg,Idb]

GLfloat light_specular[] = { 1.0, 1.0, 1.0, 1 };
// składowe intensywności świecenia źródła światła powodującego
// odbicie kierunkowe Is = [Isr,Isg,Isb]

GLfloat att_constant = { 1.0 };
// składowa stała ds dla modelu zmian oświetlenia w funkcji
// odległości od źródła

GLfloat att_linear = { 0.001 };
// składowa liniowa dl dla modelu zmian oświetlenia w funkcji
// odległości od źródła

GLfloat att_quadratic = { 0.001 };
// składowa kwadratowa dq dla modelu zmian oświetlenia w funkcji
// odległości od źródła
```

2. Dodanie źródeł światła

```
glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient1);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse1);
glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
glLightfv(GL_LIGHT0, GL_POSITION, light_position);

glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, att_constant);
glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, att_linear);
glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, att_quadratic);


glLightfv(GL_LIGHT1, GL_AMBIENT, light_ambient2);
glLightfv(GL_LIGHT1, GL_DIFFUSE, light_diffuse2);
glLightfv(GL_LIGHT1, GL_SPECULAR, light_specular);
glLightfv(GL_LIGHT1, GL_POSITION, light_position);

glLightf(GL_LIGHT1, GL_CONSTANT_ATTENUATION, att_constant);
glLightf(GL_LIGHT1, GL_LINEAR_ATTENUATION, att_linear);
glLightf(GL_LIGHT1, GL_QUADRATIC_ATTENUATION, att_quadratic);
glEnable(GL_LIGHT0);     // włączenie źródła o numerze 0
glEnable(GL_LIGHT1);     // włączenie źródła o numerze 1
```

3. Poruszanie źródłami światła

```
if (status == 1)                    // jeśli lewy klawisz myszy wcięnięty
{
    theta1 += delta_x * pix2angle / 100;    // modyfikacja kąta obrotu o kat proporcjonalny
    // do różnicy położeń kursora myszy

    phi1 += delta_y * pix2angle / 100;

    if (phi1 > pi / 2)
    {
        phi1 = pi / 2;
    }
    if (phi1 < -pi / 2)
    {
        phi1 = -pi / 2;
    }
}
if (status == 2)
{
    theta2 += delta_x * pix2angle / 100;    // modyfikacja kąta obrotu o kat proporcjonalny
    // do różnicy położeń kursora myszy

    phi2 += delta_y * pix2angle / 100;

    if (phi2 > pi / 2)
    {
        phi2 = pi / 2;
    }
    if (phi2 < -pi / 2)
    {
        phi2 = -pi / 2;
    }
}

light1[0] = radius * cos(theta1) * cos(phi1);
light1[1] = radius * sin(phi1);
light1[2] = radius * sin(theta1) * cos(phi1);

light2[0] = radius * cos(theta2) * cos(phi2);
light2[1] = radius * sin(phi2);
light2[2] = radius * sin(theta2) * cos(phi2);
```

Efekt działania programu