

Grafika komputerowa

Autor:

Jacek Wieczorek (181043)

Prowadzący:

Dr inż. Tomasz Kapłon

Wydział Elektroniki

III rok

Pn TP 08.15 - 11.00

12 grudnia 2011

1 Cel laboratorium

Celem ćwiczenia jest ilustracja możliwości oświetlania obiektów na scenach 3 – D z wykorzystaniem biblioteki *OpenGL* z rozszerzeniem *GLUT*, sterowania oświetleniem oraz budowy własnego modelu oświetlenia.

2 Zadanie 1

Pierwsze zadanie polegało na oświetleniu jednym źródłem światła obracającego się jajka, stworzonego podczas laboratorium drugiego. W celu poprawnego oświetlenia jajka, korzystając z modelu Phong'a, należy wyliczyć wektor normalny z następujących zależności :

Powierzchnia jajka opisana została równaniem : Wektor normalny można

$$x(u, v) = (-90u^5 + 225u^4 - 270u^3 + 180u^2 - 45u)\cos(\pi v)$$

$$y(u, v) = 160u^4 - 320u^3 + 160u^2$$

$$z(u, v) = (-90u^5 + 225u^4 - 270u^3 + 180u^2 - 45u)\sin(\pi v)$$

$$0 \leq u \leq 1$$

$$0 \leq v \leq 1$$

znaleźć za pomocą następujących równań :

$$\begin{aligned} N(u, v) &= \left[\begin{vmatrix} y_u & z_u \\ y_v & z_v \end{vmatrix}, \begin{vmatrix} z_u & x_u \\ z_v & x_v \end{vmatrix}, \begin{vmatrix} x_u & y_u \\ x_v & y_v \end{vmatrix} \right] = \\ &= [y_u \cdot z_v - z_u y_v, \quad z_u \cdot x_v - x_u z_v, \quad x_u \cdot y_v - y_u x_v] \neq 0 \end{aligned}$$

Czyli :

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

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

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

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

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

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

2.1 Funkcje programu :

2.1.1 Obliczanie współrzędnych powierzchni jajka i wektora normalnego

```

1  for(int i=0; i< N; i++)
    {
        for(int j=0; j<N; j++)
        {
            u = (float) i / (N-1);
            v = (float) j / (N-1);
            tab[i][j][0] = (-90 * pow(u, 5.0 f) + 225 * pow(u, 4.0 f) - 270 * pow(
                (u, 3.0 f) + 180 * pow(u, 2.0 f) - 45 * u) * cos(PI * v) ;
            tab[i][j][1] = 160 * pow(u, 4.0 f) - 320 * pow(u, 3.0 f) + 160 * pow(u
                , 2.0 f) - 5.0;
            tab[i][j][2] = (-90 * pow(u, 5.0 f) + 225 * pow(u, 4.0 f) - 270 *
                pow(u, 3.0 f) + 180 * pow(u, 2.0 f) - 45 * u) * sin(PI * v) ;

11         xu = (-450 * pow(u, 4) + 900 * pow(u, 3) - 810 * pow(u, 2) + 360 * u -
            45) * cos(PI * v);
            yu = 640 * pow(u, 3) - 960 * pow(u, 2) + 320 * u;
            zu = (-450 * pow(u, 4) + 900 * pow(u, 3) - 810 * pow(u, 2) + 360 * u -
            45) * sin(PI * v);

```

```

xv = PI * (90*pow(u,5) - 225*pow(u,4) + 270*pow(u,3) - 180*
pow(u,2) + 45*u)*sin(PI * v);
yv = 0;
zv = -PI * (90*pow(u,5) - 225*pow(u,4) + 270*pow(u,3) -
180*pow(u,2) + 45*u)*cos(PI * v);

nor[i][j][0] = (yu * zv - zu * yv);
nor[i][j][1] = (zu * xv - xu * zv);
21 nor[i][j][2] = (xu * yv - yu * xv);

float len = sqrt(pow(nor[i][j][0],2) + pow(nor[i][j][1],2) +
pow(nor[i][j][2],2));
for(int k=0; k<3; k++)
    nor[i][j][k] /= len;
}

}

for(int i=N/2; i<N; i++)
31 for(int j=0; j<N; j++)
    for(int k=0; k<3; k++)
        nor[i][j][k] *= -1;

```

2.1.2 Ustawienie parametrów materiału i źródła światła

```

glClearColor(0.0f, 0.0f, 0.0f, 1.0f);

GLfloat mat_ambient[] = {1.0, 1.0, 1.0, 1.0};
GLfloat mat_diffuse[] = {0.8, 0.8, 0.8, 1.0};
GLfloat mat_specular[] = {0.8, 0.8, 0.8, 1.0};
7 GLfloat mat_shininess = {50.0};

GLfloat light0_ambient[] = {0.0, 0.0, 0.0, 1.0};
GLfloat light0_diffuse[] = {1.0, 1.0, 0.2, 1.0};
GLfloat light0_specular[] = {1.0, 1.0, 0.2, 1.0};

GLfloat att_constant = {1.0};
GLfloat att_linear = {0.05};
GLfloat att_quadratic = {0.001};

17 glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
glMaterialf(GL_FRONT, GL_SHININESS, mat_shininess);

glLightfv(GL_LIGHT0, GL_AMBIENT, light0_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light0_diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, light0_specular);
glLightfv(GL_LIGHT0, GL_POSITION, light_pos[0]);

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

glShadeModel(GL_SMOOTH);
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glEnable(GL_DEPTH_TEST);

```

2.1.3 Rysowanie jajka

```
void EggsTriangles () {
    glBegin (GL_TRIANGLES);
    for (int i=0; i<N-1; i++){
        for (int j=0; j<N-1; j++)
        {
            glBegin (GL_TRIANGLES);

7             glVertex3fv (nor [ i ] [ j ] );
              glVertex3fv (tab [ i ] [ j ] );

              glVertex3fv (nor [ i + 1 ] [ j ] );
              glVertex3fv (tab [ i + 1 ] [ j ] );

              glVertex3fv (nor [ i ] [ j + 1 ] );
              glVertex3fv (tab [ i ] [ j + 1 ] );
              glEnd ();

17             glBegin (GL_TRIANGLES);

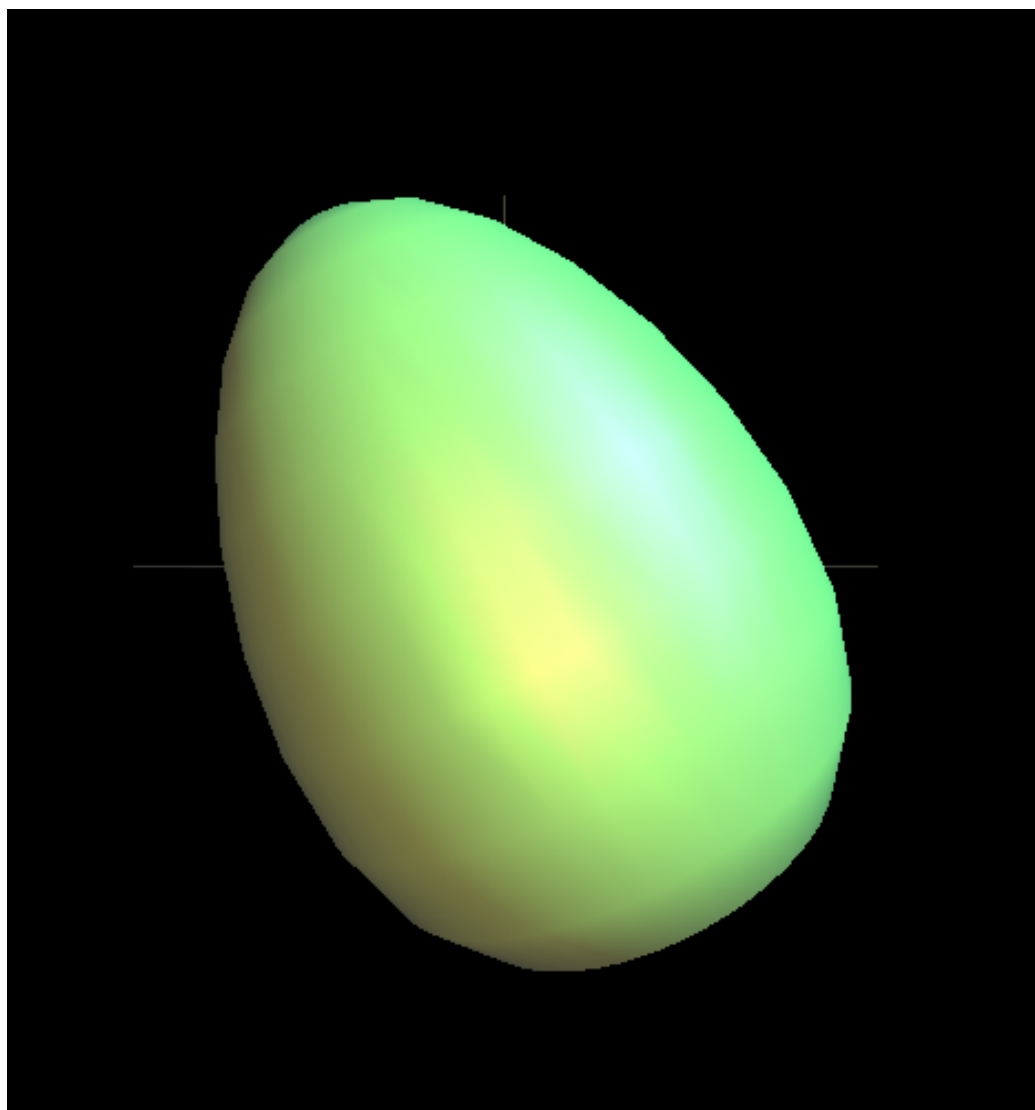
              glVertex3fv (nor [ i + 1 ] [ j + 1 ] );
              glVertex3fv (tab [ i + 1 ] [ j + 1 ] );

              glVertex3fv (nor [ i + 1 ] [ j ] );
              glVertex3fv (tab [ i + 1 ] [ j ] );

              glVertex3fv (nor [ i ] [ j + 1 ] );
              glVertex3fv (tab [ i ] [ j + 1 ] );
              glEnd ();

27             glEnd ();
        }
    }
    glEnd ();
}
```

2.2 Przykładowy wynik



3 Zadanie 2

Zadanie 2 polegało na dodaniu kolejnego źródła światła i możliwości sterowania nimi za pomocą myszki poprzez zmianę kątów azymutu i elewacji.

3.1 Kod programu

3.1.1 Dodanie drugiego źródła światła

```
GLfloat mat_ambient[] = {1.0, 1.0, 1.0, 1.0};
GLfloat mat_diffuse[] = {0.8, 0.8, 0.8, 1.0};
GLfloat mat_specular[] = {0.8, 0.8, 0.8, 1.0};
GLfloat mat_shininess = {50.0};

GLfloat light0_ambient[] = {0.0, 0.0, 0.0, 1.0};
GLfloat light0_diffuse[] = {1.0, 1.0, 0.2, 1.0};
8  GLfloat light0_specular[] = {1.0, 1.0, 0.2, 1.0};

GLfloat light1_ambient[] = {0.0, 0.0, 0.0, 1.0};
GLfloat light1_diffuse[] = {0.2, 0.9, 0.5, 1.0};
GLfloat light1_specular[] = {0.2, 0.9, 0.5, 1.0};

GLfloat att_constant = {1.0};
GLfloat att_linear = {0.05};
GLfloat att_quadratic = {0.001};

18  glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);

glLightfv(GL_LIGHT0, GL_AMBIENT, light0_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light0_diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, light0_specular);
glLightfv(GL_LIGHT0, GL_POSITION, light_pos[0]);

28  glLightfv(GL_LIGHT1, GL_AMBIENT, light1_ambient);
glLightfv(GL_LIGHT1, GL_DIFFUSE, light1_diffuse);
glLightfv(GL_LIGHT1, GL_SPECULAR, light1_specular);
glLightfv(GL_LIGHT1, GL_POSITION, light_pos[1]);

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

glLightf(GL_LIGHT1, GL_CONSTANT_ATTENUATION, att_constant);
38 glLightf(GL_LIGHT1, GL_LINEAR_ATTENUATION, att_linear);
glLightf(GL_LIGHT1, GL_QUADRATIC_ATTENUATION, att_quadratic);

glShadeModel(GL_SMOOTH);
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glEnable(GL_LIGHT1);
glEnable(GL_DEPTH_TEST);
```

3.1.2 Sterowanie oświetleniem

```
void Mouse(int btn, int state, int x, int y)
{
    if(btn==GLUTLEFTBUTTON && state == GLUTDOWN)
    {
5       x_pos_old = x;
        y_pos_old = y;
        status = 1;
    }
    else if(btn==GLUTRIGHTBUTTON && state == GLUTDOWN){
        x_pos_old = x;
        y_pos_old = y;
        status = 2;
    }
    else
15     status = 0;
}

void Motion( GLsizei x, GLsizei y )
{
    delta_x= x - x_pos_old;
    x_pos_old = x;
    delta_y = y - y_pos_old;
    y_pos_old = y;
25     glutPostRedisplay();
}

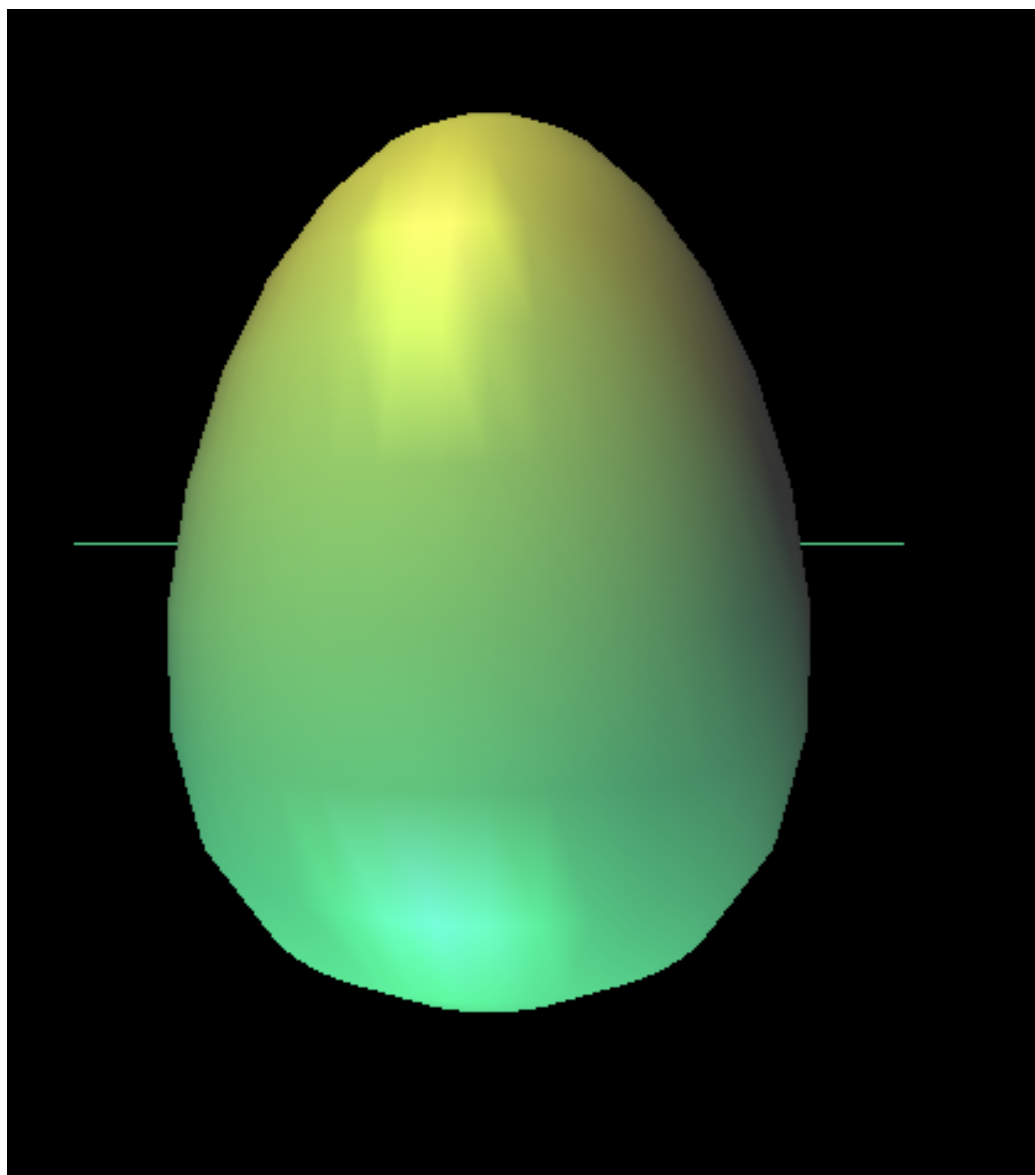
W RenderScene()

if(status == 1)
{
35     beta[0][0] += delta_x * pix2angle / 30.0;
    beta[0][1] += delta_y * pix2angle / 30.0;
}
else if(status ==2){
    beta[1][0] += delta_x * pix2angle / 30.0;
    beta[1][1] += delta_y * pix2angle / 30.0;
}

for(int b=0; b<2; b++){
    light_pos[b][0] = theta_zoom *cos(beta[b][0])*cos(beta[b][1]);
    light_pos[b][1] = theta_zoom *sin(beta[b][1]);
45     light_pos[b][2] = theta_zoom *sin(beta[b][0])*cos(beta[b][1]);
}

glLightfv(GL_LIGHT0, GL_POSITION, light_pos[0]);
glLightfv(GL_LIGHT1, GL_POSITION, light_pos[1]);
```


3.2 Przykładowy wynik



4 Wnioski

Oświetlanie obiektu w OpenGL za pomocą modelu Phong'a nie jest skomplikowanym zadaniem. Problemатyczne okazuje się wyliczenie wektorów normalnych powierzchni obiektu nie wchodzącego w skład biblioteki. Te utrudnienia nie występowały przy generowaniu oświetlenia dla czajniczka, będącego częścią biblioteki OpenGL i GLUT.