pwr**POLITECHNIKA WROCŁAWSKA**

**Instytut Informatyki, Automatyki i Robotyki**

**Zakład Systemów Komputerowych**

**Wprowadzenie do grafiki komputerowej**

**Kurs: INE4234L**

**Sprawozdanie z ćwiczenia nr 3**

**Open GL – oświetlanie scen 3D**

|  |  |
| --- | --- |
| **Wykonał:** | **Wojciech Wójcik, 235621** |
| **Termin:** | **PT/TP 8:00-11:00** |
| **Data wykonania ćwiczenia:** | **30.11.2018** |
| **Data oddania sprawozdania:** | **5.01.2019** |
| **Ocena:** |  |

|  |
| --- |
| **Uwagi prowadzącego:** |

1. **Cel ćwiczenia**

Celem ćwiczenia jest zapoznanie się z możliwościami oświetlenia obiektów w scenie 3D z wykorzystaniem OpenGL oraz GLUT. W ćwiczeniach została wykorzystana wiedza zdobyta we wcześniejszych ćwiczeniach. Zaprezentowano działanie wektorów normalnych oraz definicje parametrów światła oraz materiałów oświetlanego obiektu.

1. **Zadanie do wykonania**

Pierwsze zadanie są stosunkowo proste, należy odpowiednio zmodyfikować istniejący już program, tak aby dodać do sceny oświetlenie. Dotyczy to własnoręcznie wykonanego modelu jajka, którym można manipulować przy użyciu myszki.

Kolejnym zadaniem było dodanie dwóch źródeł światła do sceny z modelem jajka oraz umożliwienie manipulacji każdym z nich (jajko jest statyczne).

1. **Realizacja zadania**
   1. **Dodanie oświetlenia do sceny z modelem jajka.**

Zgodnie z instrukcją zostały dodane wartości w funkcji MyInit tak aby poprawnie zainicjować parametry światła.

void MyInit(void)

{

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

GLfloat mat\_ambient[] = { 1.0,1.0, 1.0, 1 };

GLfloat mat\_diffuse[] = { 1.0, 1.0, 1.0, 1 };

GLfloat mat\_specular[] = { 1.0, 1.0, 1.0, 1.0 };

GLfloat mat\_shininess = { 20.0 };

GLfloat light\_position[] = { 0.0, 0.0, 30.0, 1.0 };

GLfloat light\_ambient[] = { 1.0, 0.0, 0.0, 1.0 };

GLfloat light\_diffuse[] = { 1.0, 0.0, 0.0, 1.0 };

GLfloat light\_specular[] = { 1.0, 1.0, 0.0, 1.0 };

GLfloat att\_constant = { 1.0 };

GLfloat att\_linear = { (GLfloat) 0.05 };

GLfloat att\_quadratic = { (GLfloat) 0.001 };

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, 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);

glShadeModel(GL\_SMOOTH); // właczenie łagodnego cieniowania

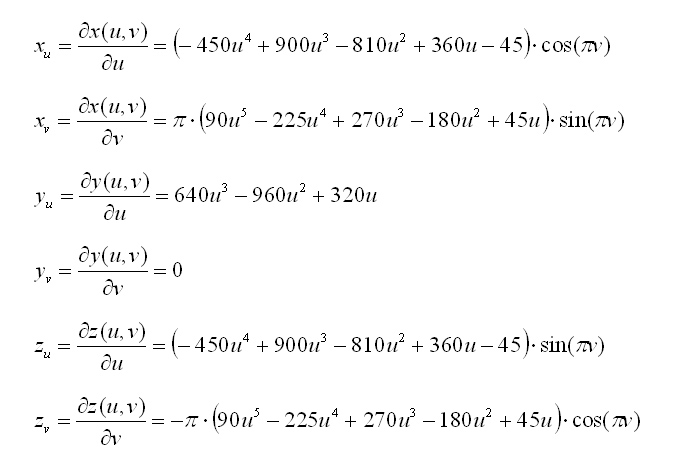
glEnable(GL\_LIGHTING); // właczenie systemu oświetlenia sceny

glEnable(GL\_LIGHT0); // włączenie źródła o numerze 0

glEnable(GL\_DEPTH\_TEST); // włączenie mechanizmu z-bufora

}

Kolejnym i ostatnim krokiem była edycja funkcji Egg, tak aby dodać do modelu informacje o jego wektorach normalnych, co pozwoli na poprawną iluminacje powierzchni. Zostało to wykonane zgodnie z wzorami dostarczonymi w instrukcji laboratoryjnej:



Do funkcji Egg została dodana zmienna vektorNorm, która jest trójwymiarową tablicą przetrzymującą informacje o wektorach normalnych. Po wyznaczeniu wektorów normalnych zostały one dodane do pętli odpowiadającej za rysowanie modelu jajka.

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - 1; j++) {

glColor3f(255.0f, 255.0f, 255.0f);

glBegin(GL\_TRIANGLES);

glNormal3fv(vektorNorm[i][j + 1]);

glVertex3fv(vectors3d[i][j + 1]);

glNormal3fv(vektorNorm[i + 1][j]);

glVertex3fv(vectors3d[i + 1][j]);

glNormal3fv(vektorNorm[i + 1][j + 1]);

glVertex3fv(vectors3d[i + 1][j + 1]);

glEnd();

glBegin(GL\_TRIANGLES);

glNormal3fv(vektorNorm[i][j]);

glVertex3fv(vectors3d[i][j]);

glNormal3fv(vektorNorm[i + 1][j]);

glVertex3fv(vectors3d[i + 1][j]);

glNormal3fv(vektorNorm[i][j + 1]);

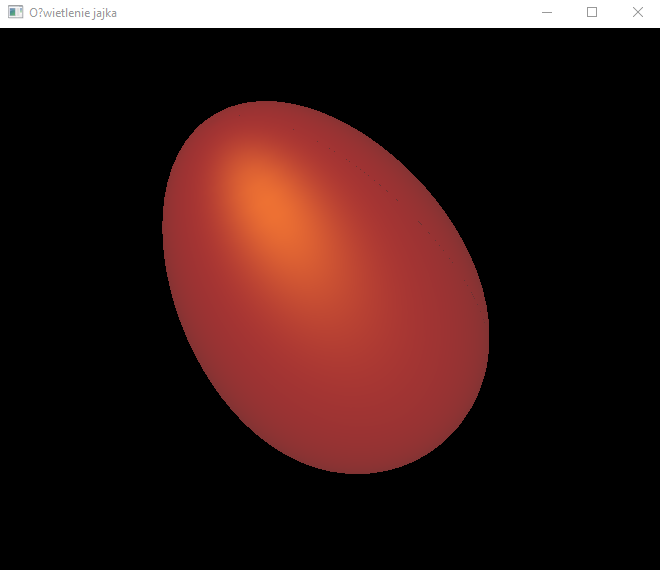
glVertex3fv(vectors3d[i][j + 1]);

glEnd();

}

}

Efekt końcowy prezentuje się następująco:



* 1. **Oświetlenie modelu jajka dwoma źródłami światła.**

Zgodnie z poprzednią instrukcją laboratoryjną została zaimplementowana mechanika obracania elementu wokół punktu, lecz tym razem dotyczyło to położenia 2 źródeł światła. Funkcja MyInit prezentuje wyliczanie położenia źródeł świateł.

void RenderScene(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

if (status == 1)

{

theta1 += delta1\_x \* pix2angle / 180 \* M\_PI;

theta1\_y += delta1\_y \* pix2angle / 180 \* M\_PI;

}

if (status == 2)

{

theta2 += delta2\_x \* pix2angle / 180 \* M\_PI;

theta2\_y += delta2\_y \* pix2angle / 180 \* M\_PI;

}

GLfloat up = 1.0;

GLfloat x1 = viewer[2] \* cos(-theta1) \* cos(-theta1\_y);

GLfloat y1 = viewer[2] \* sin(-theta1\_y);

GLfloat z1 = viewer[2] \* sin(-theta1) \* cos(-theta1\_y);

GLfloat x2 = viewer[2] \* cos(-theta2) \* cos(-theta2\_y);

GLfloat y2 = viewer[2] \* sin(-theta2\_y);

GLfloat z2 = viewer[2] \* sin(-theta2) \* cos(-theta2\_y);

GLfloat light\_position[] = { x1, y1, z1, 1.0 };

GLfloat light\_position2[] = { x2, y2, z2, 1.0 };

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);

glLightfv(GL\_LIGHT1, GL\_POSITION, light\_position2);

gluLookAt(0.0, 0.0, -15.0, 0.0, 0.0, 0.0, 0.0, up, 0.0);

Axes();

glColor3f(1.0f, 1.0f, 1.0f);

glTranslated(0.0, -5.0, 0.0);

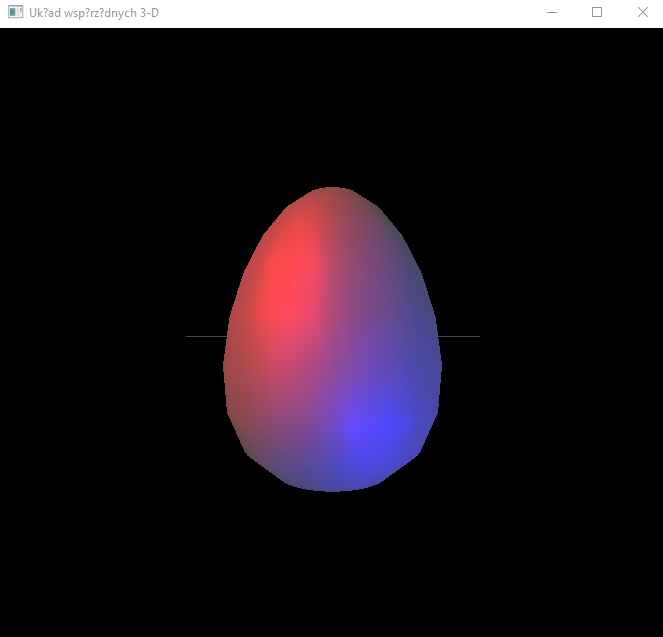
Egg();

glFlush();

glutSwapBuffers();

}

Również została zmodyfikowana odpowiednio funkcja odpowiedzialna za przechwytywanie informacji z myszki aby umożliwić manipulację źródłami świtała. Uzyskany efekt prezentuje się następująco:



1. **Wnioski**

Głównym problemem ćwiczenia było poprawne zaimplementowanie obliczenia wektorów normalnych, wymagane było przypomnienie jak dokładnie działa funkcja tworząca model jajka. Zostały zdobyte podstawowe informację o oświetlaniu sceny, materiałach oraz wyznaczaniu wektorów normalnych. Ponadto zostały udoskonalone umiejętności interakcji z użytkownikiem.