

Laboratorium grafiki komputerowej i animacji

Ćwiczenie V - Biblioteka OpenGL - oświetlenie sceny

Przygotowanie do ćwiczenia:

1. Zapoznać się ze zdefiniowanymi w OpenGL modelami światła i właściwościami materiałów.
2. Zapoznać się z zestawem komend OpenGL umożliwiającym zdefiniowanie źródeł światła i właściwości materiałów.

Przebieg ćwiczenia:

1. Założenia:
 - a. Celem prac na zajęciach laboratoryjnych jest zdefiniowanie parametrów oświetlenia i oświetlenie modelu manipulatora Puma (rysunek 1.1).
 - b. Wynikiem prac na dzisiejszych zajęciach ma być program zbliżony w działaniu do programu „**puma_swiatlo.exe**” dostarczonego do materiałów laboratoryjnych.
 - c. Realizacja ćwiczenia polega na uzupełnieniu kodu programu „**gl_template**” modyfikowanego na ostatnich zajęciach.
 - d. W realizacji prac wzorować się należy na rozwiązaniach przyjętych w programie „**shinyjet**” również dołączonym do materiałów laboratoryjnych.



Rys 1.1 Oświetlony model manipulatora Puma

2. Uwagi do sposobu realizacji celu dzisiejszych zajęć laboratoryjnych:
 - a. W pierwszej kolejności należy zdefiniować parametry źródła światła. Charakterystyki źródła światła dokonuje się zwykle jednokrotnie przed wyświetlaniem sceny (w przykładowych programach definicja oświetlenia znajduje się w funkcji „SetupRC()”). Szczegóły można znaleźć w materiałach wprowadzających do ćwiczenia. Poniżej zawarto przykładowy ciąg poleceń OpenGL ustalający oświetlenie na scenie:

```
// wartości składowych oświetlenia i koordynaty źródła światła:  
GLfloat ambientLight[] = { 0.3f, 0.3f, 0.3f, 1.0f };  
GLfloat diffuseLight[] = { 0.7f, 0.7f, 0.7f, 1.0f };  
GLfloat specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };  
GLfloat lightPos[] = { 0.0f, 150.0f, 150.0f, 1.0f };
```

```
GLfloat specref[] = { 1.0f, 1.0f, 1.0f, 1.0f };
```

```
glEnable(GL_DEPTH_TEST);      // usuwanie niewidocznych powierzchni
glFrontFace(GL_CCW);          // ustalenie "zewnątrznych" ścian
glEnable(GL_CULL_FACE);       // wyłączenie obliczania wewnętrznych ścian
```

```
// Uaktywnienie oświetlenia
glEnable(GL_LIGHTING);
```

```
// Ustawienie parametrów źródła światła 0
glLightfv(GL_LIGHT0, GL_AMBIENT, ambientLight);
glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuseLight);
glLightfv(GL_LIGHT0, GL_SPECULAR, specular);
glLightfv(GL_LIGHT0, GL_POSITION, lightPos);
glEnable(GL_LIGHT0);
```

```
// Uaktywnienie śledzenia kolorów
glEnable(GL_COLOR_MATERIAL);
```

```
// Materiały mają śledzić kolory ustalone poleceniem glColor
glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
```

```
// Od tej chwili wszystkie materiały będą miały pełną odbłaskowość
// z dużą jasnością
glMaterialfv(GL_FRONT, GL_SPECULAR, specref);
glMateriali(GL_FRONT, GL_SHININESS, 128);
```

```
// Normalne zostaną automatycznie normalizowane
glEnable(GL_NORMALIZE);
```

- b. Kolejnym etapem tworzenia efektu oświetlenia jest zdefiniowanie wektorów normalnych do ścian, które chcemy widzieć jako oświetlone. Normalną można zdefiniować dla każdego wierzchołka wielokąta. Ponieważ zakładamy, że pojedynczy wielokąt wyznacza płaszczyznę, wystarczy dla wszystkich wierzchołków wielokąta wyznaczyć jedną normalną. Wyjątek w naszym programie stanowią będą tworzące walców. Dla osiągnięcia efektu „wygładzenia” powierzchni bocznych walca wygodniej jest definiować normalne prostopadłe do krawędzi ciągu czworokątów przybliżających tworzącą walca.

- c. Przykładowy fragment programu definiujący normalne dla walca zawarto poniżej:

```
void walec(double h, double r)
```

```
{
    double angle,x,y;
```

```
    glBegin(GL_TRIANGLE_FAN);
    glNormal3d(0.0,0.0,-1.0);
    glVertex3d(0.0f, 0.0f, 0.0f);
    for(angle = 0.0f; angle <= (2.0f*GL_PI); angle += (GL_PI/8.0f))
    {
        x = r*sin(angle);
```

```

        y = r*cos(angle);

        glVertex2d(x, y);
    }
    glEnd();

    glBegin(GL_TRIANGLE_FAN);
    glNormal3d(0.0,0.0,1.0);
    glVertex3d(0.0, 0.0, h);
    for(angle = 0.0f; angle >= -(2.0f*GL_PI); angle -= (GL_PI/8.0f))
    {
        x = r*sin(angle);
        y = r*cos(angle);
        glVertex3d(x, y, h);
    }
    glEnd();

    glBegin(GL_QUAD_STRIP);

    for(angle = 0.0f; angle >= -(2.0f*GL_PI); angle -= (GL_PI/8.0f))
    {
        x = r*sin(angle);
        y = r*cos(angle);
        glNormal3d(sin(angle),cos(angle),0.0);
        glVertex3d(x, y, h);
        glVertex3d(x, y, 0);
    }
    glEnd();
}

```

- d. Dla dowolnego wielokąta, usytuowanego dowolnie w przestrzeni trudno jest poprawnie zdefiniować wektor prostopadły o długości 1. W tym celu uzupełniono zestaw funkcji programu glTemplate o funkcję calcNormal(float v[3][3], float out[3]). Dokładny opis działania funkcji zawarto w materiałach wprowadzających do ćwiczenia. Funkcja umożliwia zdefiniowanie normalnej do wielokąta na podstawie współrzędnych trzech wierzchołków danego wielokąta. Przykład wykorzystania funkcji umieszczono poniżej:

```

glBegin(GL_QUADS);
{
    float v[3][3]= {      {0.0f, -15.0f, 10.0f},
                          {-60.0f,-10.0f,10.0f},
                          {-60.0f,-10.0f,0.0f}
    };

    float norm[3];
    calcNormal(v,norm);
    glNormal3d(norm[0],norm[1],norm[2]);
    glVertex3d(0.0, -15.0, 10.0);
    glVertex3d(-60.0,-10.0,10.0);
    glVertex3d(-60.0,-10.0,0.0);
    glVertex3d(0.0, -15.0, 0.0);
}

```

```
}  
glEnd();
```

Zadania nadobowiązkowe:

1. Stosując rozwiązania zawarte w przykładowym programie „spot” należy zdefiniować reflektor w scenie z robotem.