OpenGL – model oświetlenia

- **Składowe światła OpenGL**
  - **Światło otaczające (ambient)**

    Nie pochodzi z żadnego określonego kierunku. Powoduje równomierne oświetlenie obiektów na wszystkich powierzchniach i wszystkich kierunkach.
  - **Światło rozproszone (diffuse)**

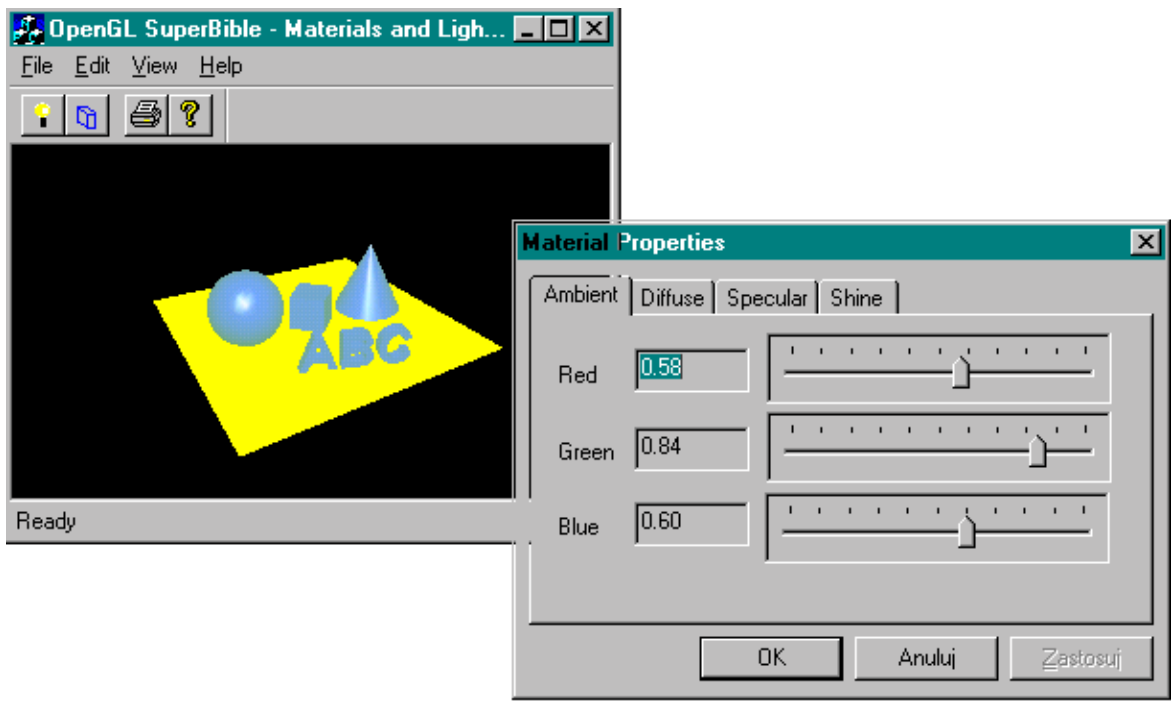
    Pochodzi z określonego kierunku, odbijane jest od powierzchni równomiernie. Powierzchnie intensywniej oświetlone są jaśniejsze od mniej oświetlonych.
  - **Światło odbłysków (kierunkowe) (specular)**

    Biegnie z określonego kierunku, odbijane jest w ściśle określonym kierunku.
- **Przykład – definiowanie źródła światła laserowego:**

	Czerwony (R)	Zielony (G)	Niebieski (B)	Alpha
Św. kierunkowe	0,99	0,0	0,0	1,0
Św. rozproszone	0,10	0,0	0,0	1,0
Św. otaczające	0,05	0,0	0,0	1,0

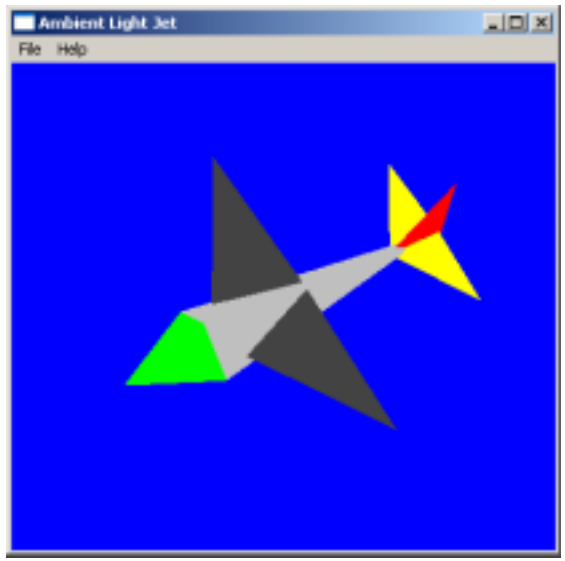
OpenGL – model materiału

- **W rzeczywistym świecie materiały posiadają swój własny kolor – zdolność odbijania większości fotonów o określonej długości fali i pochłaniania większości fotonów o innych długościach fali.**
- **Kolor obiektu na scenie jest wynikiem złożenia koloru oświetlenia i koloru materiału powiązanego z danym obiektem.**
- **Ilustracja problemu - program matlight:**

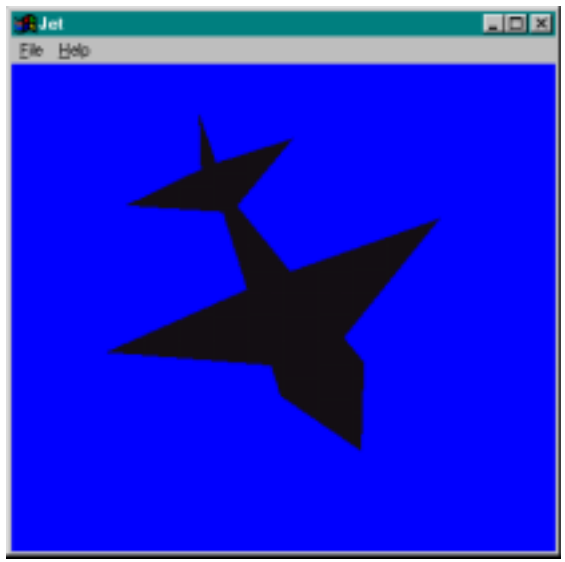


OpenGL – kreowanie oświetlenia sceny

Sytuacja wyjściowa –
scena bez włączonego oświetlenia:

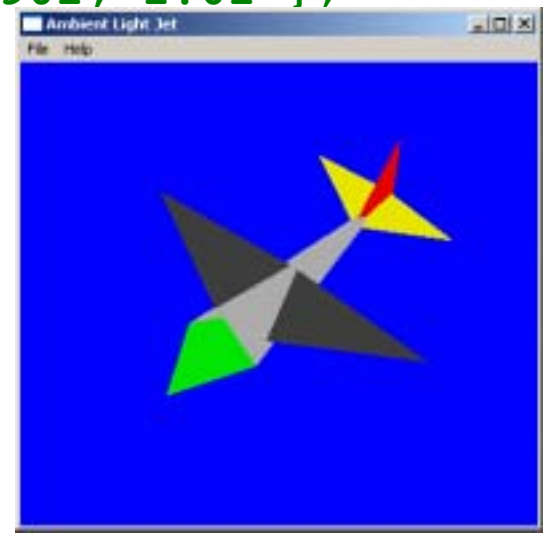



```
// Włączenie oświetlenia
{...
glEnable(GL_LIGHTING);
...}
```

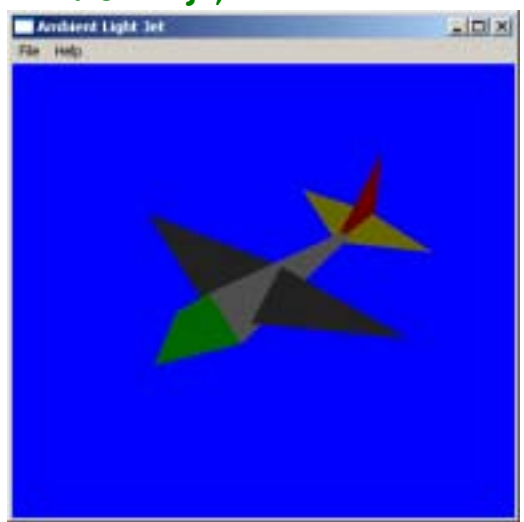


OpenGL – scena oparta na świetle otaczającym

```
{GLfloat ambientLight[] = { 0.90f, 0.90f, 0.90f, 1.0f };
 glEnable(GL_LIGHTING);
 glLightModelfv(     GL_LIGHT_MODEL_AMBIENT,
                     ambientLight
                );
 glEnable(GL_COLOR_MATERIAL);
 glColorMaterial(    GL_FRONT,
                     GL_AMBIENT_AND_DIFFUSE
                 );
...}
```

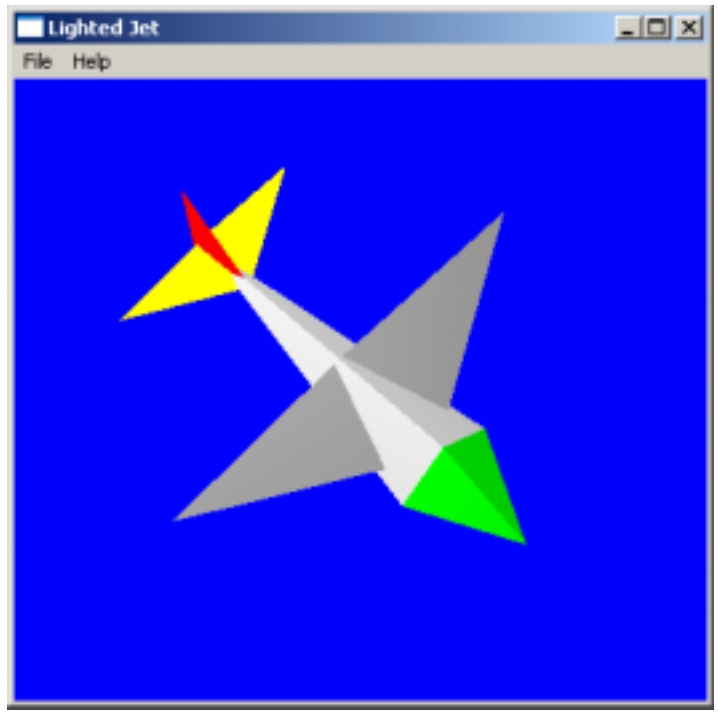



```
{GLfloat ambientLight[] = { 0.4f, 0.4f, 0.4f, 1.0f };
...
}
```



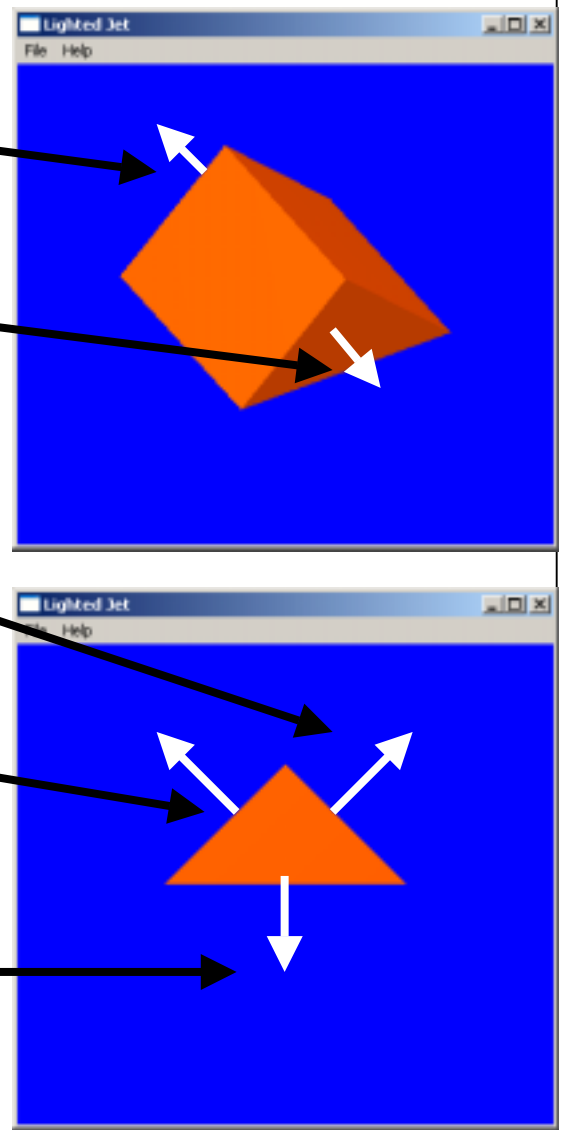
OpenGL – scena oparta na świetle rozproszonym

```
{GLfloat  ambientLight[] = { 0.5f, 0.5f, 0.5f, 1.0f };
 GLfloat  diffuseLight[] = { 0.7f, 0.7f, 0.7f, 1.0f };
 GLfloat       lightPos[] = { -50.f, 50.0f, 100.0f, 1.0f };

 glEnable(GL_LIGHTING);

 glLightfv(   GL_LIGHT0,
              GL_AMBIENT,
              ambientLight);
 glLightfv(   GL_LIGHT0,
              GL_DIFFUSE,
              diffuseLight);
 glLightfv(   GL_LIGHT0,
              GL_POSITION,
              lightPos);
 glEnable(GL_LIGHT0);

 glEnable(GL_COLOR_MATERIAL);
 glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
 ...
}
```



OpenGL – podstawowe definiowanie normalnych

```
glBegin(GL_TRIANGLES);
glNormal3d(0,0,-1);
glVertex3d(10,0,-10); glVertex3d(-10,0,-10);
glVertex3d(0,10,-10);
glNormal3d(0,0,1);
glVertex3d(10,0,10); glVertex3d(0,10,10);
glVertex3d(-10,0,10); glEnd();

glBegin(GL_QUADS);
glNormal3d(sqrt(2)/2,sqrt(2)/2,0);
glVertex3d(10,0,10);  glVertex3d(10,0,-10);
glVertex3d(0,10,-10); glVertex3d(0,10,10);

glNormal3d(-sqrt(2)/2,sqrt(2)/2,0);
glVertex3d(-10,0,10); glVertex3d(0,10,10);
glVertex3d(0,10,-10); glVertex3d(-10,0,-10);

glNormal3d(0,-1,0);
glVertex3d(10,0,10);  glVertex3d(-10,0,10);
glVertex3d(-10,0,-10);glVertex3d(10,0,-10);
glEnd();
```



OpenGL – zaawansowane definiowanie normalnych

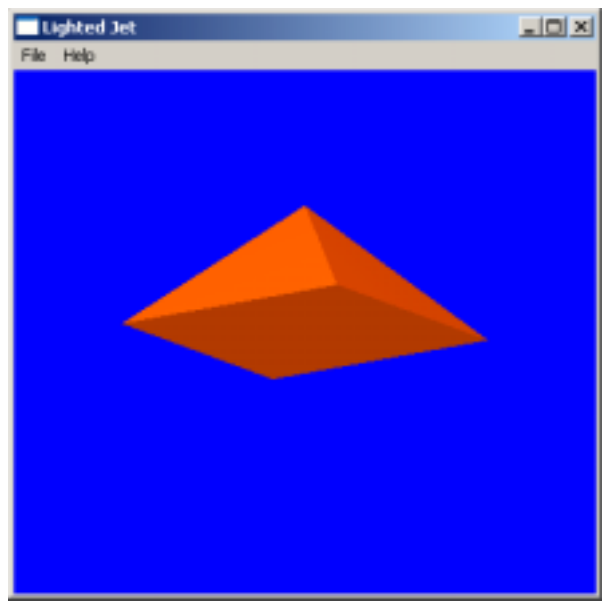
```
// Normalizacja wektora:
void ReduceToUnit(float vector[3])
    {
    float length;
    length = (float)sqrt((vector[0]*vector[0]) +
                         (vector[1]*vector[1]) +
                         (vector[2]*vector[2]));

    if(length == 0.0f)
        length = 1.0f;
    vector[0] /= length;
    vector[1] /= length;
    vector[2] /= length;
    }
```

OpenGL – zaawansowane definiowanie normalnych

```
// Obliczanie normalnych i normalizacja
void calcNormal(float v[3][3], float out[3])
    {
    float v1[3],v2[3];
    static const int x = 0;
    static const int y = 1;
    static const int z = 2;

    v1[x] = v[0][x] - v[1][x];
    v1[y] = v[0][y] - v[1][y];
    v1[z] = v[0][z] - v[1][z];

    v2[x] = v[1][x] - v[2][x];
    v2[y] = v[1][y] - v[2][y];
    v2[z] = v[1][z] - v[2][z];

    out[x] = v1[y]*v2[z] - v1[z]*v2[y];
    out[y] = v1[z]*v2[x] - v1[x]*v2[z];
    out[z] = v1[x]*v2[y] - v1[y]*v2[x];

    ReduceToUnit(out);      }
```

OpenGL – zaawansowane definiowanie normalnych

```
// Zastosowanie funkcji calcNormal
{
        float normal[3];
        float v[3][3] = {    { 10.0f,0.0f,10.0f },
                             { 0.0f,10.0f,0.0f },
                             { -10.0f,0.0f,10.0f },
                       };

        calcNormal(v,normal);
        glBegin(GL_TRIANGLES);
                glNormal3fv(normal);
                glVertex3fv(v[0]);
                glVertex3fv(v[1]);
                glVertex3fv(v[2]);
        glEnd();
    }
```



OpenGL – światło odbłysków

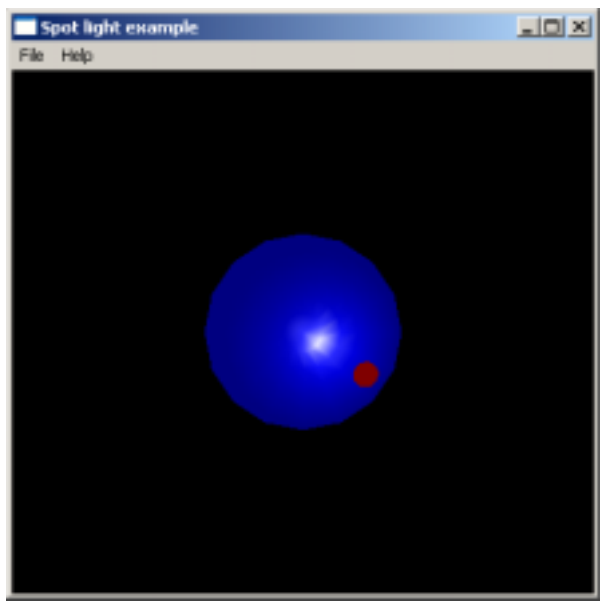
```
{
GLfloat  ambientLight[] = { 0.3f, 0.3f, 0.3f, 1.0f };
GLfloat  diffuseLight[] = { 0.7f, 0.7f, 0.7f, 1.0f };
GLfloat  specular[] = { 1.0f, 1.0f, 1.0f, 1.0f};
Glfloat  lightPos[] = { 0.0f, 150.0f, 150.0f, 1.0f };
GLfloat  specref[] =  { 1.0f, 1.0f, 1.0f, 1.0f };


glEnable(GL_LIGHTING);

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);

glEnable(GL_COLOR_MATERIAL);
glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);

glMaterialfv(GL_FRONT, GL_SPECULAR,specref);
glMateriali(GL_FRONT,GL_SHININESS,128);
}
```

OpenGL – reflektor – ustawienie parametrów źródła światła

```
{GLfloat  specular[] = { 1.0f, 1.0f, 1.0f, 1.0f};
 GLfloat  specref[] =  { 1.0f, 1.0f, 1.0f, 1.0f };
 GLfloat  ambientLight[] = { 0.5f, 0.5f, 0.5f, 1.0f};

 glEnable(GL_LIGHTING);
 glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientLight);

 glLightfv(GL_LIGHT0,GL_DIFFUSE,ambientLight);
 glLightfv(GL_LIGHT0,GL_SPECULAR,specular);
 glLightfv(GL_LIGHT0,GL_POSITION,lightPos);

 glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,60.0f);
 glLightf(GL_LIGHT0,GL_SPOT_EXPONENT,10.0f);

 glEnable(GL_LIGHT0);

 glEnable(GL_COLOR_MATERIAL);

 glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);

 glMaterialfv(GL_FRONT, GL_SPECULAR,specref);
 glMateriali(GL_FRONT, GL_SHININESS,50);}
```

OpenGL – reflektor – symulacja źródła światła

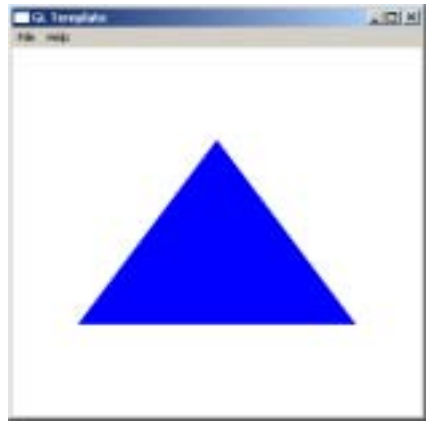
```
glRGB(0, 0, 255);
auxSolidSphere(30.0f);
glPushMatrix();
        glRotatef(yRot, 0.0f, 1.0f, 0.0f);
        glRotatef(xRot, 1.0f, 0.0f, 0.0f);

        glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
        glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);

        glRGB(255,0,0);
        glTranslatef(lightPos[0],lightPos[1],lightPos[2]);
        auxSolidCone(4.0f,6.0f);

        glPushAttrib(GL_LIGHTING_BIT);
                glDisable(GL_LIGHTING);
                glRGB(255,255,0);
                auxSolidSphere(3.0f);
        glPopAttrib();
glPopMatrix();
```



OpenGL- cieniowanie

- „płaskie” – kolor wielokąta, to kolor ostatniego wierzchołka:

```
{
    glShadeModel(GL_FLAT);
    glBegin(GL_TRIANGLES);
        glColor3d(1,0,0);    glVertex3d(0,20,0);
        glColor3d(0,1,0);    glVertex3d(-30,-20,0);
        glColor3d(0,0,1);    glVertex3d(30,-20,0);
    glEnd();
}
```



- „gładkie” – kolor każdego wierzchołka może być dobierany indywidualnie:

```
{
    glShadeModel(GL_SMOOTH);
    glBegin(GL_TRIANGLES);
        glColor3d(1,0,0);    glVertex3d(0,20,0);
        glColor3d(0,1,0);    glVertex3d(-30,-20,0);
        glColor3d(0,0,1);    glVertex3d(30,-20,0);
    glEnd();
}
```

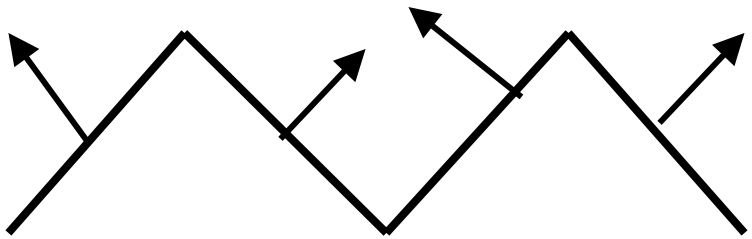
OpenGL- cieniowanie – zastosowanie w oświetleniu

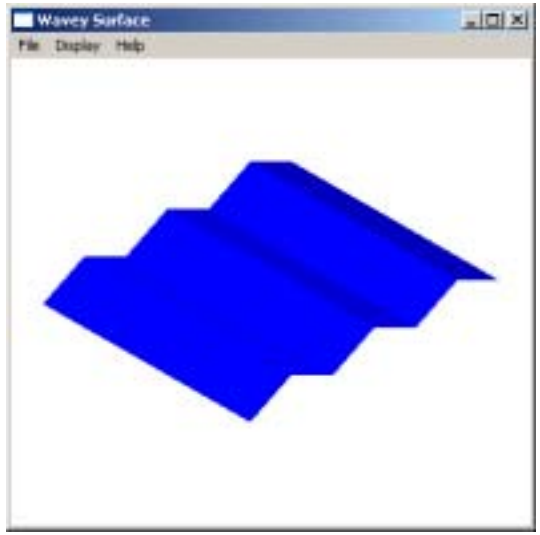
- **Symulowanie oświetlenia:**

```
{
    glShadeModel(GL_FLAT);
    glColor3d(1,0,0);   glVertex3d(0,20,0);
    glColor3d(1,0,0);   glVertex3d(-30,-20,0);
    glColor3d(0.3,0,0); glVertex3d(30,-20,0);
}
```

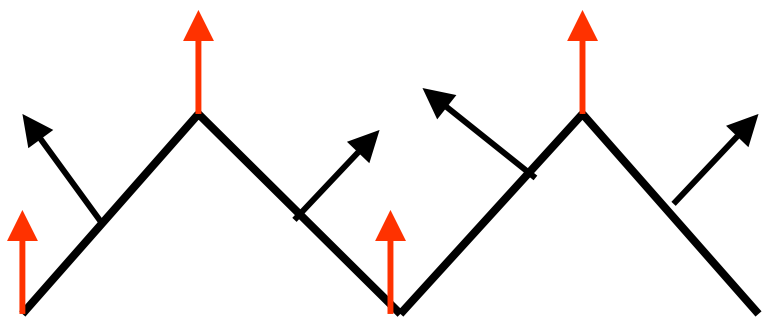
Uśrednianie normalnych

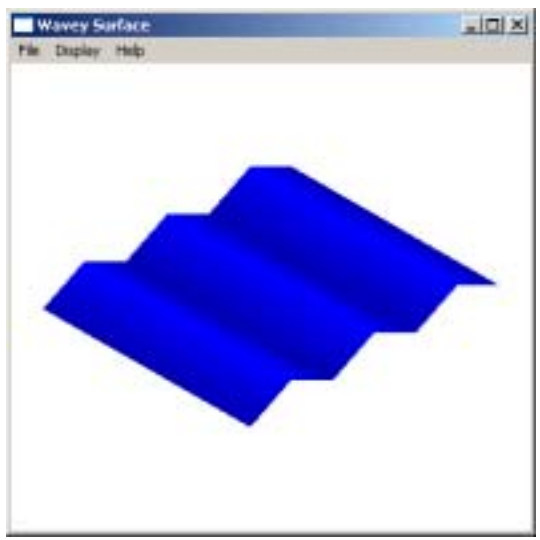
- **Typowe definiowanie normalnych:**





- **Efekt „wygładzenia krawędzi” przez uśrednienie normalnych:**

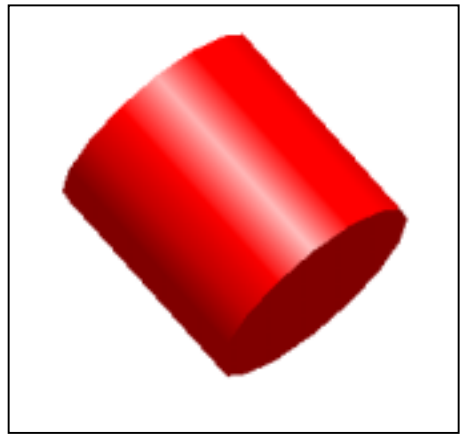




Przykład – oświetlony walec

```
void SetupRC(){ //definicja 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); FrontFace(GL_CCW);
glEnable(GL_CULL_FACE);  glEnable(GL_LIGHTING);
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);

glEnable(GL_COLOR_MATERIAL);
glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);

glMaterialfv(GL_FRONT, GL_SPECULAR,specref);
glMateriali(GL_FRONT,GL_SHININESS,128);
glShadeModel(GL_SMOOTH);
glClearColor(1.0f, 1.0f, 1.0f, 1.0f ); glColor3f(0.0,0.0,0.0);}
```

Przykład – oświetlony walec



```
void walec(double h, double r){
double angle,x,y, GL_PI = 3.1415;

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();}
```

OpenGL – zestawienie funkcji związanych z generacją oświetlenia

```
glEnable(GL_LIGHTING);          // glDisable(GL_LIGHTING);
glEnable(GL_COLOR_MATERIAL);    // glDisable(GL_COLOR_MATERIAL);
glEnable(GL_LIGHT0);            // glDisable(GL_LIGHT0);
glColorMaterial(GL_FRONT,           GL_AMBIENT,           );
//              GL_BACK,            GL_DIFFUSE,
//              GL_FRONT_AND_BACK,GL_SPECULAR,
//              GL_AMBIENT_AND_DIFFUSE,
glLightf[v](GL_LIGHT0,      GL_AMBIENT,          parametry);
//                          GL_DIFFUSE,
//                          GL_SPECULAR,
//                          GL_POSITION,
//                          GL_SPOT_DIRECTION,
//                          GL_SPOT_EXPONENT,
//                          GL_SPOT_CUTOFF,
//                          GL_CONSTANT_ATTENUATION,
//                          GL_LINEAR_ATTENUATION,
//                          GL_QUADRATIC_ATTENUATION,
glMaterialf[v]( GL_FRONT,           GL_AMBIENT,parametry);
//              GL_BACK,                    GL_DIFFUSE,
//              GL_FRONT_AND_BACK,          GL_SPECULAR,
//              GL_AMBIENT_AND_DIFFUSE,
glLightModel(GL_LIGHT_MODEL_AMBIENT,parametry);
```