

# 7. Модели освещения

# Составляющие освещения

Модель освещения (закрашивания) определяет, как свет от источника рассеивается по поверхности или отражается от нее.

Модели освещения:

1. Ахроматическая (монохромная) модель;
2. Цветовая модель.

В модели освещения, используемой в компьютерной графике, предполагается, что объекты сцены освещаются двумя типами источников света: точечным источником света и фоновым источником света.

# Составляющие освещения

При взаимодействии с объектом часть света поглощается и превращается в тепло; часть – отражается и часть – проникает внутрь.

Объект видим в том случае, когда часть света отражается и попадает в глаз наблюдателя.

Если весь падающий свет поглощается – то это абсолютно черное тело.

Если весь свет проходит сквозь объект, то объект видим только за счет рефракции.

# Составляющие освещения

Различают два типа отражения света:

1. Диффузное рассеивание – часть падающего на объект света слегка проникает внутрь поверхности и излучается обратно во всех направлениях равномерно. Рассеянный цвет сильно взаимодействует с материалом поверхности, поэтому его цвет зависит от природы материала, из которого сделана поверхность;
2. Зеркальное отражение – падающий свет прямо отражается от поверхности, не проникая вглубь. В первом приближении зеркально отраженный свет имеет тот же цвет, что и падающий. В более сложных моделях цвет зеркально отраженного света пробегает интервал бликов, что дает лучшее приближение металлических поверхностей.

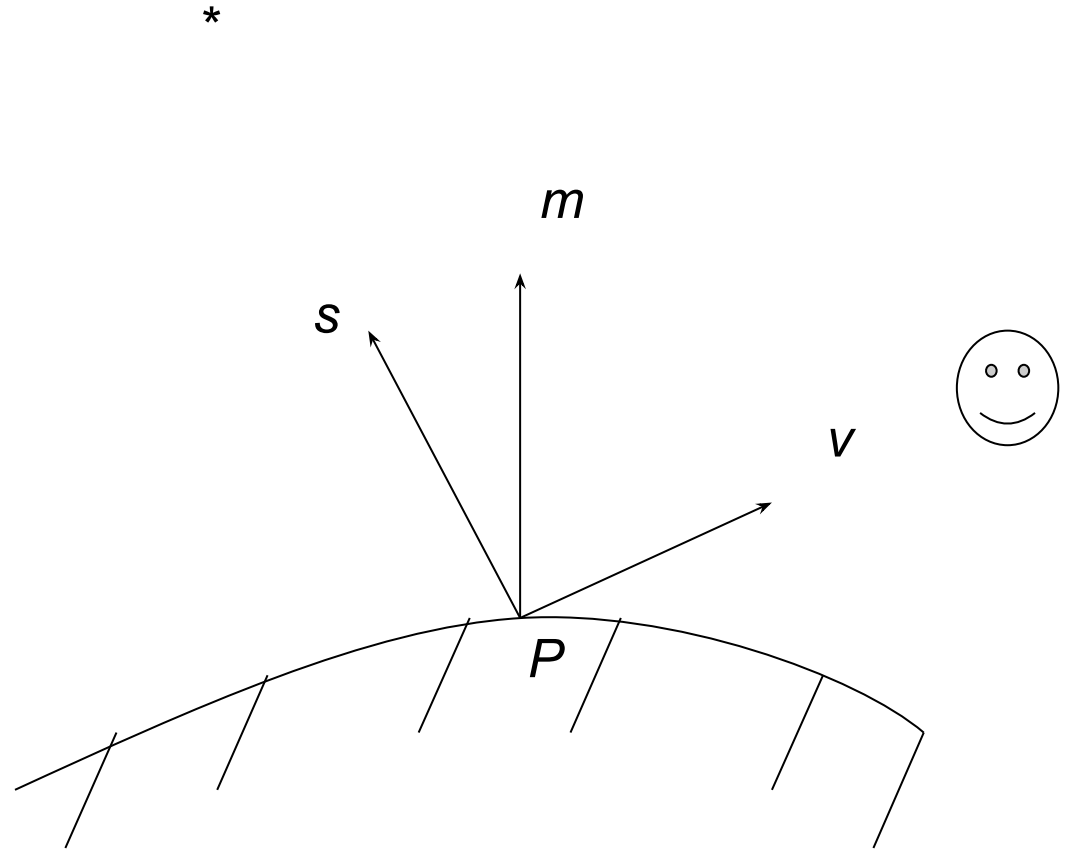
# Составляющие освещения

$P$  – точка на  
поверхности;

$m$  – нормаль к  
поверхности в  
точке  $P$ ;

$s$  – вектор,  
указывающий  
направление от  
точки  $P$  к  
источнику света;

$v$  – вектор,  
указывающий  
направление от  
точки  $P$  к глазу  
наблюдателя.



# Составляющие освещения

Каждая грань объекта имеет две стороны: видимую и невидимую.

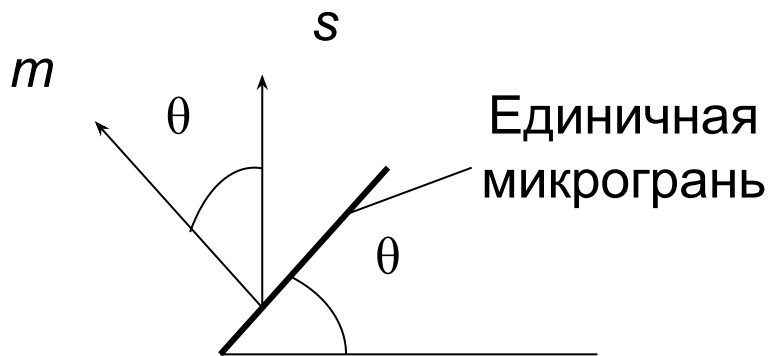
Для видимой стороны должно соблюдаться условие  $v \cdot m > 0$  ( $\cdot$  - операция скалярного умножения)

# Диффузная составляющая отраженного света

Пусть  $I_s$  — интенсивность источника. Тогда интенсивность отраженного света будет  $I_s \cos \Theta$ . Из всего отраженного доля  $\rho_d$  будет приходиться на диффузную составляющую ( $\rho_d$  — коэффициент диффузного отражения).

Тогда диффузная компонента отраженного света будет равна  $I_s \rho_d \cos \Theta$ . Поскольку диффузное отражение равномерно во всех направлениях, то интенсивность диффузной компоненты, попадающей в глаз наблюдателя, не зависит от вектора.

# Диффузная составляющая отраженного света



$$\cos \Theta = \frac{s \cdot m}{|s| \cdot |m|}$$

$$I_d = I_s \rho_d \frac{s \cdot m}{|s| \cdot |m|}$$

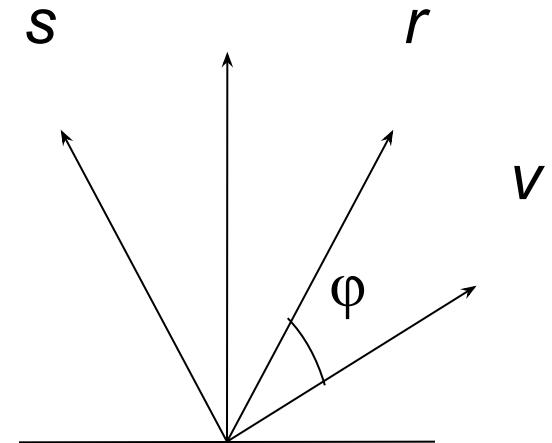
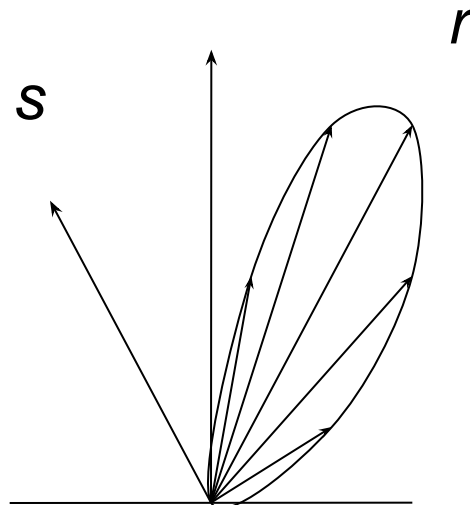
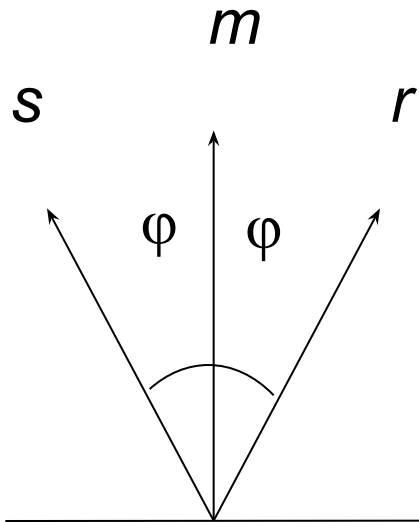
Закон Ламберта

$$I_d = I_s \rho_d \max \left( \frac{s \cdot m}{|s| \cdot |m|}, 0 \right)$$



# Зеркальная составляющая отраженного света

Если поверхность идеально зеркальная, то отражение осуществляется по правилу: угол падения равен углу отражения. Однако в реальности это не так.



# Зеркальная составляющая отраженного света

Модель Фонга

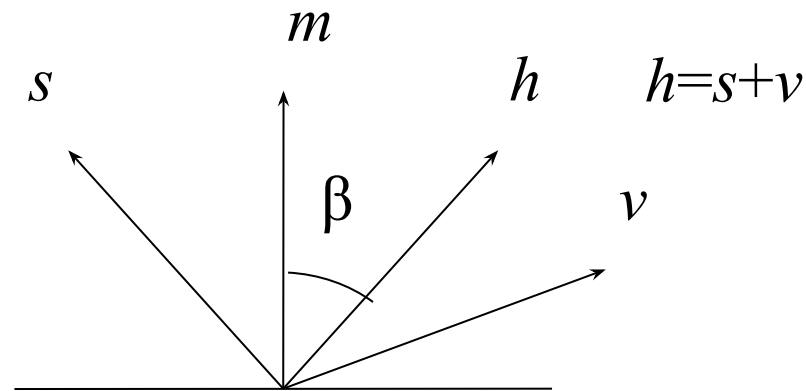
$$r = -s + 2 \frac{s \cdot m}{|m|^2} m \quad \cos \varphi = \frac{r \cdot v}{|r| \cdot |v|},$$
$$I_{sp} = I_s \rho_s \left( \frac{r \cdot v}{|r| \cdot |v|} \right)^\Phi$$

$I_{sp}$  – интенсивность зеркальной составляющей,  
попадающей в глаз наблюдателя;

$\rho_s$  – коэффициент зеркального отражения;

$\Phi$  – параметр, учитывающий неидеальную зеркальность  
( $\Phi = 1 \dots 200$ )

# Зеркальная составляющая отраженного света



$$I_{sp} = I_s \rho_s \max \left( \left( \frac{h \cdot m}{|h| \cdot |m|} \right)^\Phi, 0 \right)$$

# Фоновая составляющая отраженного света

Если использовать только диффузную и зеркальную компоненты, то реалистичность полученного изображения оказывается не всегда удовлетворительной. Грани, которые не освещаются, будут глубоко черными, тени резкими и глубокими.

Для смягчения этого эффекта добавляют третью компоненту света — фоновый свет, источник которого считается не расположенным ни в каком определенном месте, свет от него распространяется во всех направлениях одинаково.

Этот источник характеризуется интенсивностью  $I_a$ , а каждая грань коэффициентом фонового отражения  $\rho_a$ .

Следовательно, в глаз наблюдателя, расположенного в любом месте, попадает отраженная часть фонового источника  $I_a \rho_a$ .

# Монохромная модель освещения

$$I = I_s \rho_d \max\left(\frac{s \cdot m}{|s| \cdot |m|}, 0\right) + I_s \rho_s \max\left(\left(\frac{h \cdot m}{|h| \cdot |m|}\right)^\Phi, 0\right) + I_a \rho_a$$

$$I = I_{s1} \rho_d \max\left(\frac{s \cdot m}{|s| \cdot |m|}, 0\right) + I_{s2} \rho_s \max\left(\left(\frac{h \cdot m}{|h| \cdot |m|}\right)^\Phi, 0\right) + I_a \rho_a$$

В освещении участвуют три независимых источника – фоновый  $I_a$  и два точечных  $I_{s1}$ ,  $I_{s2}$ . Точечные независимы, но совмещены в пространстве. Один из них имеет интенсивность  $I_{s1}$  и является источником, создающим диффузную составляющую, другой имеет интенсивность  $I_{s2}$  и является источником, создающим зеркальную составляющую.

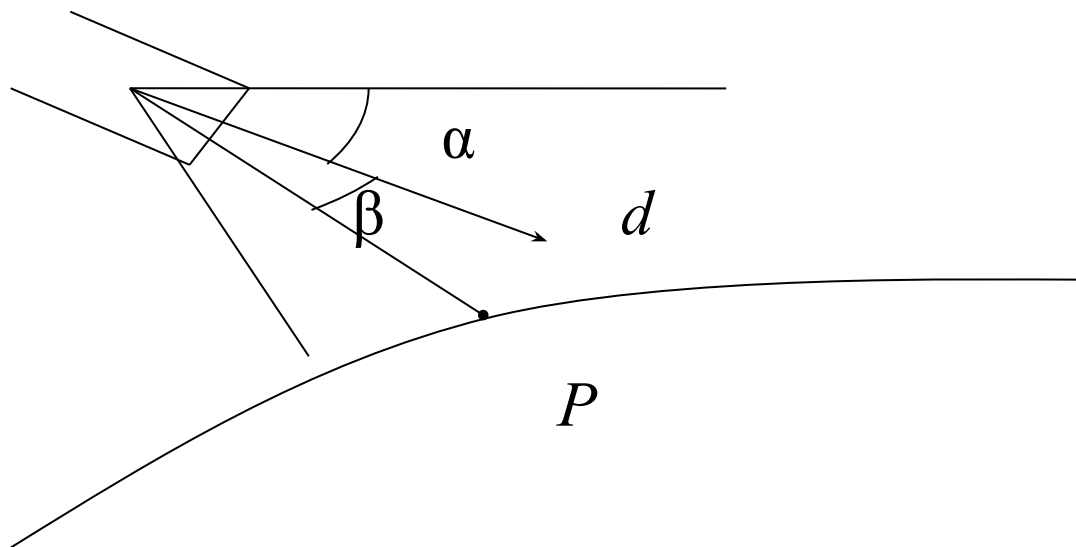
# Цветовая модель освещения

$R, G, B:$

$$I = I_{s1}\rho_d \max\left(\frac{s \cdot m}{|s| \cdot |m|}, 0\right) + I_{s2}\rho_s \max\left(\left(\frac{h \cdot m}{|h| \cdot |m|}\right)^\Phi, 0\right) + I_a\rho_a$$

Каждый фоновый, диффузный и зеркальный источник представляет собой три независимых источника света, излучающие свет красного, зеленого и синего цвета.

# Прожекторы



$$I = \cos^{\varepsilon} \beta$$

# Ослабление света с расстоянием

$$K_{ослаб} = \frac{1}{K_1 + K_2 D + K_3 D^2}$$

$K_1$  – коэффициент постоянного ослабления

$K_2$  – коэффициент линейного ослабления

$K_3$  – коэффициент квадратичного ослабления



# Параметры источника освещения

- Цвет фонового освещения (GL\_AMBIENT)
- Цвет рассеянного освещения (GL\_DIFFUSE)
- Цвет отраженного света (GL\_SPECULAR)
- Расположение (GL\_POSITION)
- Направление распространения света (GL\_SPOT\_DIRECTION)
- Концентрация светового луча (сфокусированность источника) (GL\_SPOT\_EXPONENT)
- Угол разброса световых лучей (GL\_SPOT\_CUTOFF)
- Коэффициент постоянного ослабления (GL\_CONSTANT\_ATTENUATION)
- Коэффициент линейного ослабления (GL\_LINEAR\_ATTENUATION)
- Коэффициент квадратичного ослабления (GL\_QUADRATIC\_ATTENUATION)

# Свойства материала

- Фоновый цвет материала (GL\_AMBIENT)
- Рассеянный цвет материала (GL\_DIFFUSE)
- Фоновый и рассеянный цвет материала (GL\_AMBIENT\_AND\_DIFFUSE)
- Отраженный цвет материала (GL\_SPECULAR)
- Коэффициент зеркального отражения (блеск) (GL\_SHININESS)
- Излучаемый цвет материала (GL\_EMISSION)
- Индексы фонового, рассеянного и отраженного цветов (GL\_COLOR\_INDEXES)

# Цвет вершины

$$\begin{aligned} \text{цвет вершины} = & \text{emission}_{\text{material}} + \text{ambient}_{\text{light}} \times \text{ambient}_{\text{material}} + \\ & + \sum_{i=1}^n \left( \frac{1}{k_{\text{constant}} + k_{\text{linear}} d + k_{\text{quadratic}} d^2} \right)_i \times \text{spotlight\_effect}_i \times [\text{ambient}_{\text{light}} \times \text{ambient}_{\text{material}} + \\ & + (\max\{L \cdot n, 0\}) \times \text{diffuse}_{\text{light}} \times \text{diffuse}_{\text{material}} + \\ & + (\max\{s \cdot n, 0\})^{\text{shininess}} \times \text{specular}_{\text{light}} \times \text{specular}_{\text{material}}]_i \end{aligned}$$

$\text{spotlight\_effect} = 1$ , если источник – не прожектор; 0, если источник – прожектор, но вершина лежит вне его конуса;  $(\max\{v \cdot d, 0\})^{\text{GL\_SPOT\_EXPONENT}}$ , если источник – прожектор и вершина лежит в его конусе,  $v$  – единичный вектор из прожектора в вершину,  $d$  – вектор ориентации прожектора.

$L$  – единичный вектор направления из вершины на источник;  $n$  – единичный вектор нормали;  $s$  – нормализованная сумма двух векторов: вектор направления из вершины на источник и вектор из вершины на точку наблюдения.

# Пример

```
#include <GL/glut.h>
GLfloat diffuseMaterial[4] = { 0.5, 0.5, 0.5, 1.0 };
void init(void)
{
    GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
    GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glShadeModel (GL_SMOOTH);
    glEnable(GL_DEPTH_TEST);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, diffuseMaterial);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glMaterialf(GL_FRONT, GL_SHININESS, 25.0);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glColorMaterial(GL_FRONT, GL_DIFFUSE);
    glEnable(GL_COLOR_MATERIAL);
}
```

# Пример

```
void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glutSolidSphere(1.0, 20, 16);
    glFlush ();
}

void reshape (int w, int h)
{
    glViewport (0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity();
    if (w <= h) glOrtho (-1.5, 1.5, -1.5*(GLfloat)h/(GLfloat)w,
        1.5*(GLfloat)h/(GLfloat)w, -10.0, 10.0);
    else
        glOrtho (-1.5*(GLfloat)w/(GLfloat)h,
            1.5*(GLfloat)w/(GLfloat)h, -1.5, 1.5, -10.0, 10.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}
```

# Пример

```
void mouse(int button, int state, int x, int y)
{
    switch (button) {
        case GLUT_LEFT_BUTTON:
            if (state == GLUT_DOWN) {
                diffuseMaterial[0] += 0.1;
                if (diffuseMaterial[0] > 1.0) diffuseMaterial[0] = 0.0;
                glColor4fv(diffuseMaterial);
                glutPostRedisplay();
            }
            break;
        case GLUT_MIDDLE_BUTTON:
            if (state == GLUT_DOWN) {
                diffuseMaterial[1] += 0.1;
                if (diffuseMaterial[1] > 1.0) diffuseMaterial[1] = 0.0;
                glColor4fv(diffuseMaterial);
                glutPostRedisplay();
            }
            break;
    }
}
```

# Пример

```
case GLUT_RIGHT_BUTTON:
    if (state == GLUT_DOWN) {
        diffuseMaterial[2] += 0.1;
        if (diffuseMaterial[2] > 1.0) diffuseMaterial[2] = 0.0;
        glColor4fv(diffuseMaterial);
        glutPostRedisplay();
    }
    break;
default: break;
}
}

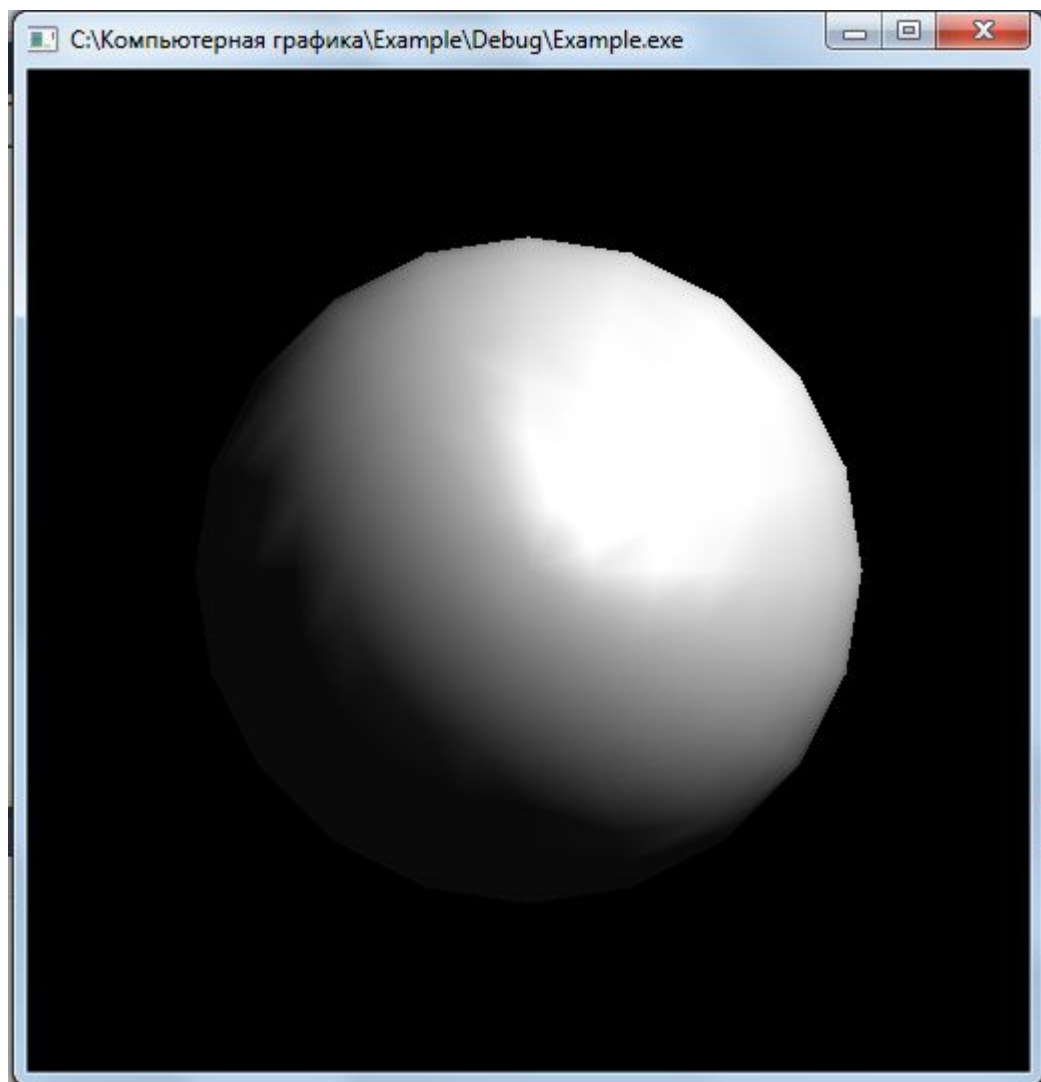
void keyboard(unsigned char key, int x, int y)
{
    switch (key) {
        case 27:
            exit(0);
            break;
    }
}
```

# Пример

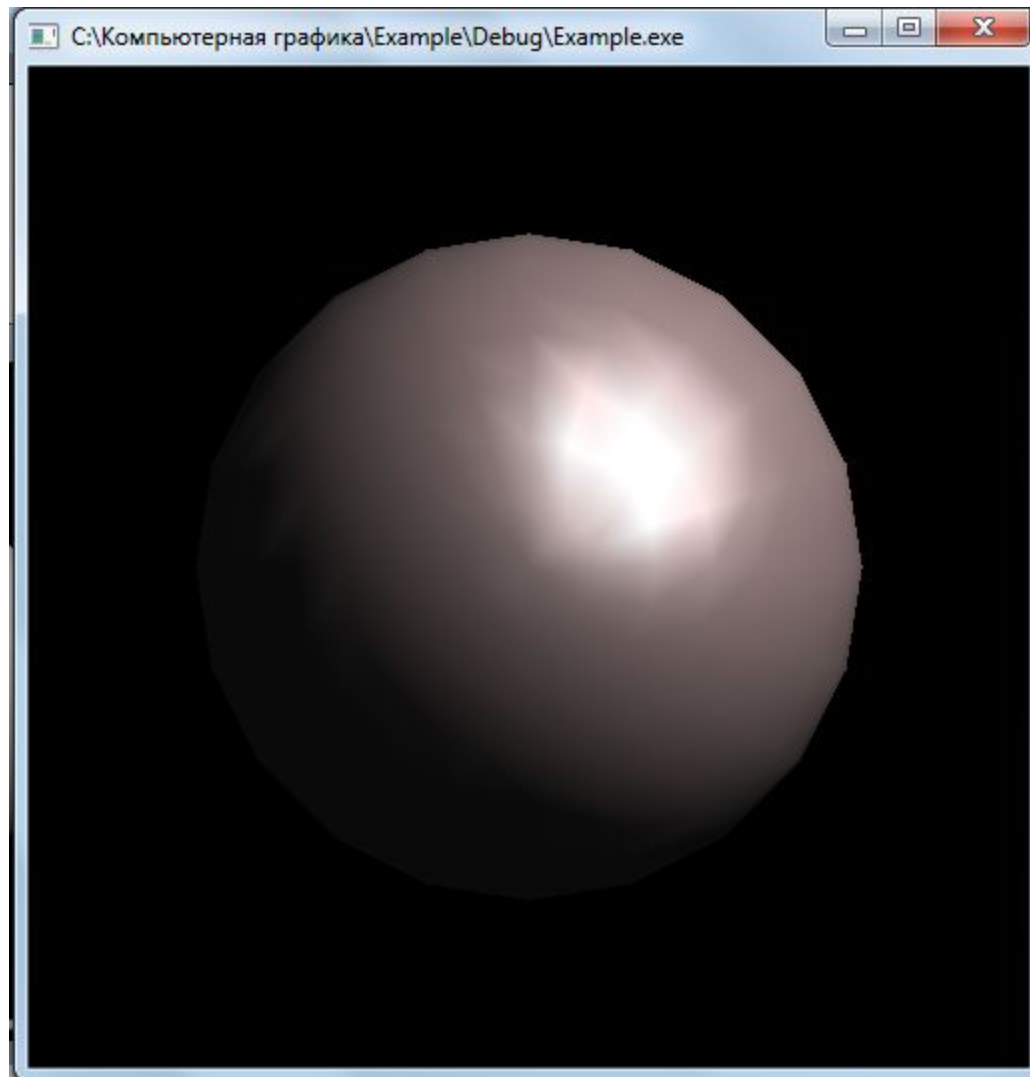
```
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize (500, 500);
    glutInitWindowPosition (100, 100);
    glutCreateWindow (argv[0]);
    init ();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMouseFunc(mouse);
    glutKeyboardFunc(keyboard);
    glutMainLoop();
    return 0;
}
```



# Пример



# Пример



# Пример

