ЛАБОРАТОРНАЯ РАБОТА №6
Освещение и материалы.

Цель работы: сделать освещение и задать материал по модели Фонга.

Задание: написать код для задания освещения, создать трехмерный объект с нормалями и материалом.

Освещение по Фонгу

Освещение по Фонгу включает в себя также и модель освещения Фонга, т.е. алгоритм расчёта освещения в заданной точке. Это локальная модель освещения, т.е. она учитывает только свойства заданной точки и источников освещения, игнорируя эффекты рассеивания, линзирования, отражения от соседних тел.

Затенение по Фонгу требует сравнительно мало ресурсов, но большинство оптических явлений игнорируются либо рассчитываются с грубым приближением.

Другие модели освещения могут лучше учитывать свойства материала (локальные модели Орена-Наяра, Кука-Торренса, анизотропные модели) или сложные оптические явления (глобальные модели), но ведут к росту накладных расходов.

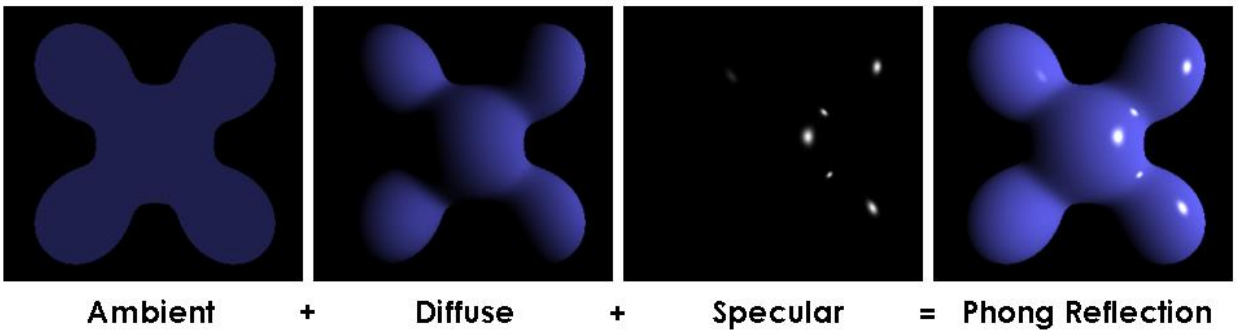

Рисунок 1 Создание освещения разными компонентами по Фонгу

Ambient

В модели Фонга окружающий свет - это тип рассеянного света, который отражает общее освещение сцены. Это свет, который рассеивается и отражается всеми поверхностями в сцене, и у него нет определенного направления или источника.

Чтобы реализовать окружающий свет в OpenGL с использованием модели Фонга, вам необходимо установить свойство ambient material для каждого объекта в сцене с помощью функции glMaterialfv(). Это свойство определяет, какая часть окружающего света отражается объектом.

Вам также необходимо установить параметры окружающего освещения с помощью функции glLightfv(). Эти параметры определяют цвет и интенсивность окружающего света в сцене.

Diffuse

В модели Фонга рассеянный свет представляет собой свет, который рассеивается и отражается поверхностью во всех направлениях одинаково. Этот тип освещения важен для создания восприятия текстуры поверхности и глубины сцены.

Чтобы реализовать рассеянный свет в OpenGL с использованием модели Фонга, вам необходимо установить свойство diffuse material для каждого объекта в сцене с помощью функции glMaterialfv(). Это свойство определяет, какая часть рассеянного света отражается объектом.

Вам также необходимо установить параметры рассеянного света с помощью функции glLightfv(). Эти параметры определяют цвет и интенсивность рассеянного света в сцене.

Specular

В модели Фонга зеркальный свет представляет собой свет, который отражается в определенном направлении блестящей поверхностью. Этот тип освещения важен для создания бликов и блестящих отражений на поверхностях в сцене.

Чтобы реализовать зеркальный свет в OpenGL с использованием модели Фонга, вам необходимо установить свойство зеркального материала для каждого объекта в сцене с помощью функции glMaterialfv(). Это свойство определяет, какая часть зеркального света отражается объектом.

Вам также необходимо установить параметры зеркального освещения с помощью функции glLightfv(). Эти параметры определяют цвет и интенсивность зеркального света в сцене, а также положение и ориентацию зрителя.

Normals

В модели освещения Фонга нормали важны для расчета направления отраженного света. Зеркальная составляющая модели Фонга зависит от угла

между направлением зрителя и направлением отраженного света, который вычисляется с использованием нормали к поверхности.

Без нормалей расчеты освещения были бы неточными, и полученное изображение выглядело бы плоским и нереалистичным. Нормали необходимы для создания реалистичных световых эффектов в OpenGL с использованием модели освещения Фонга.

Нормали задаются для каждой вершины многоугольника с помощью функции glNormal3f() или массива. Эти значения используются для интерполяции значений нормали по поверхности многоугольника, что важно для расчета освещенности в каждой точке поверхности.

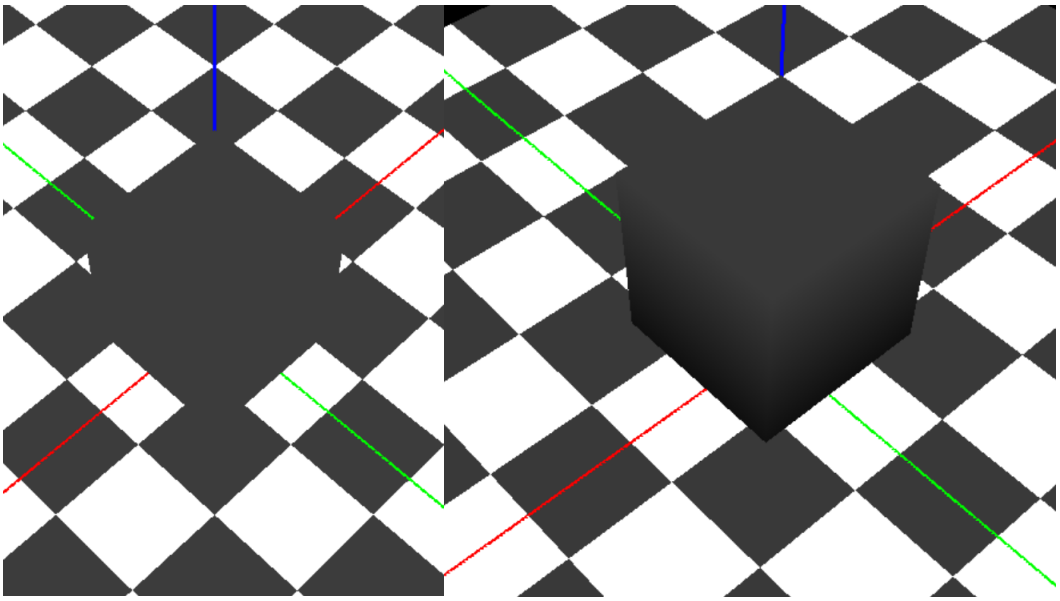

Рисунок 2 Разница в освещении с нормалями и без

Для включения работы освещения

Для корректной работы освещения требуется настроить свет и материал.

Настройка света:

```
void Init_Light()
{
   glEnable(GL_LIGHTING); //общее освещения для всего пространства
   glShadeModel(GL_SMOOTH);

   GLfloat light_position[] = { 0.0f, 0.0f, 5.0f, 1.0f }; //позиция источника
   GLfloat light_spot_direction[] = {0.0, 0.0, -1.0, 1.0}; // позиция цели
   GLfloat light_ambient[] = { 0.1f, 0.1f, 0.1f, 1.0f }; //параметры
   GLfloat light_diffuse[] = { 1.0f, 1.0f, 1.0f, 1.0f }; //параметры
   GLfloat light_specular[] = { 0.2f, 0.2f, 0.2f, 32.0f }; //параметры
```

```
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);

    glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 15); // конус для
                                              //направленного источника
    glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, light_spot_direction);
    glLightf(GL_LIGHT0, GL_SPOT_EXPONENT, 8.0); // экспонента
                                              //убывания интенсивности
  //задействование настроек для источника LIGHT0
    glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);

    glEnable(GL_LIGHT0); // источник света LIGHT0
  }
```

Для включения работы материала

Для корректной работы освещения требуется настроить свет и материал.

Настройка материала:

```
  void Init_Material()
  {
  glEnable(GL_COLOR_MATERIAL); //разрешения использования
                                              //материала
    glShadeModel(GL_SMOOTH); // сглаживает границы
    GLfloat material_ambient[] = { 0.2f, 0.2f, 0.2f, 1.0f };
    GLfloat material_diffuse[] = { 1.0f, 1.0f, 1.0f, 1.0f };
    GLfloat material_specular[] = { 1.0f, 1.0f, 1.0f, 32.0f };
    GLfloat material_shininess[] = { 50.0f }; //блеск материала

    glMaterialfv(GL_FRONT, GL_AMBIENT, material_ambient);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, material_diffuse);
    glMaterialfv(GL_FRONT, GL_SPECULAR, material_specular);
    glMaterialfv(GL_FRONT, GL_SHININESS, material_shininess);
  }
```

Для работы с нормалями

К уже существующей функции для отображения шахматной поверхности нужно добавить:

```
  …
  glEnableClientState(GL_NORMAL_ARRAY);
```

```
   glNormalPointer(GL_FLOAT,0,&normal_vert);
…
```
Заранее создав массив: float normal_vert[]={0,0,1, 0,0,1, 0,0,1, 0,0,1};
И в конце функции не забывайте:
```
…
glDisable(GL_NORMAL_ARRAY);
…
```

Задания

1. Включить и настроить освещение

При включённом освещении и повороте камеры площадка должна изменятся.

2. Создать куб с нормалями и посмотреть, как он будет работать при включенном освещении. Сделайте несколько кубов в разных местах.

Функция для создания куба с нормалями

```
…
void Draw_Cube(){

GLfloat vertices[] = {
   -0.5f, -0.5f, -0.5f,
    0.5f, -0.5f, -0.5f,
    0.5f,  0.5f, -0.5f,
   -0.5f,  0.5f, -0.5f,
   -0.5f, -0.5f,  0.5f,
    0.5f, -0.5f,  0.5f,
    0.5f,  0.5f,  0.5f,
   -0.5f,  0.5f,  0.5f
};

GLuint indices[] = {
   0, 1, 2,
   2, 3, 0,
   1, 5, 6,
   6, 2, 1,
   7, 6, 5,
   5, 4, 7,
   4, 0, 3,
   3, 7, 4,
   4, 5, 1,
   1, 0, 4,
   3, 2, 6,
```

```
    6, 7, 3
};
GLfloat normals[] = {
    0.0f, 0.0f, -1.0f,
    0.0f, 0.0f, -1.0f,
    0.0f, 0.0f, -1.0f,
    0.0f, 0.0f, -1.0f,
    0.0f, 0.0f, 1.0f,
    0.0f, 0.0f, 1.0f,
    0.0f, 0.0f, 1.0f,
    0.0f, 0.0f, 1.0f,
    -1.0f, 0.0f, 0.0f,
    -1.0f, 0.0f, 0.0f,
    -1.0f, 0.0f, 0.0f,
    -1.0f, 0.0f, 0.0f,
    1.0f, 0.0f, 0.0f,
    1.0f, 0.0f, 0.0f,
    1.0f, 0.0f, 0.0f,
    1.0f, 0.0f, 0.0f,
    0.0f, -1.0f, 0.0f,
    0.0f, -1.0f, 0.0f,
    0.0f, -1.0f, 0.0f,
    0.0f, -1.0f, 0.0f,
    0.0f, 1.0f, 0.0f,
    0.0f, 1.0f, 0.0f,
    0.0f, 1.0f, 0.0f,
    0.0f, 1.0f, 0.0f
};
glEnableClientState(GL_VERTEX_ARRAY);
glVertexPointer(3, GL_FLOAT, 0, vertices);

glEnableClientState(GL_NORMAL_ARRAY);
glNormalPointer(GL_FLOAT, 0, normals);

glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT, indices);

glDisableClientState(GL_VERTEX_ARRAY);
glDisableClientState(GL_NORMAL_ARRAY);
}
…
```

3. Добавьте перемещение источнику света вокруг куба.