Лабораторна робота № 4

Освітлення: загальні принципи, джерела, налаштування параметрів.

Мета: *Набути практичних навичок побудови реалістичного тривимірного зображення шляхом освітлення.*
Завдання до роботи: *Розробити програму в якій будуть використані засоби OpenGL для роботи з освітлення тривімірного об'єкту.*

Увага:
Для отримання оцінки “задовільно”, необхідно виконати завдання рівню А
Для отримання оцінки “добре” ,необхідно виконати завдання рівнів А, Б
Для отримання оцінки “віднінно” ,необхідно виконати завдання рівнів А, Б, В

Рівень “А”

Завдання до роботи:

1.1 Визначити вектор нормалі для кожної вершини кожного об'єкта. Ці нормалі задають орієнтацію об'єкта по відношенню до джерел світла.
1.2 Створити, вибрати і позиціонувати одне джерело світла (згідно з примірником).
1.3 Створити і вибрати модель освітлення, яка визначає рівень глобального фонового світла та ефективне положення точки спостереження (для обчислень, пов'язаних з освітленням).
1.4 Поставити властивості матеріалу для об'єктів сцени.

Таблиця : Об’єкти та матеріали

	Латунь	**Хром**	**Золото**	**Срібло**	**Чорний пластик**	**Бронза**	**Смарагд**	**Обсидіан**	**Нефрит**	**Рубін**
Коло	1	2	3	4	5	6	7	8	9	10
Трикутник	11	12	13	14	15	16	17	18	19	20

Чайник	21	22	23	24	25	26	27	28	29	30

	GL_AMBIENT	**GL_DIFFUSE**	**GL_SPECULAR**	**GL_SHININESS**
Латунь	(0.309412, 0.223529, 0.027431, 1.0)	(0.870390, 0.568627, 0.113725, 1.0)	(0.99457, 0.941176, 0.807842, 1.0)	(27.8974)
Хром	(0.25, 0.25, 0.25, 1.0)	(0.4, 0.4, 0.4, 1.0)	(0.774597, 0.774597, 0.774597,1.0)	(76.8)
Золото	(0.24725, 0.1995, 0.0745, 1.0)	(0.75164, 0.60648, 0.29648, 1.0)	(0.628281, 0.555802, 0.366065, 1.0)	(51.2)
Срібло	(0.19225, 0.19225, 0.19225,1.0)	(0.50754, 0.50754, 0.50754, 1.0)	(0.508273, 0.508273, 0.508273, 1.0)	(51.2)
Чорний пластик	(0.02, 0.02, 0.02, 0.02)	(0.01, 0.01, 0.01, 0.01)	(0.4, 0.4, 0.4, 0.4)	(10)
Бронза	(0.2125, 0.1275, 0.054, 1.0)	(0.714, 0.4284, 0.18144, 1.0)	(0.393548, 0.271906, 0.1666721,1.0)	(25.6)
Смарагд	(0.0215, 0.1745, 0.0215, 0.55)	(0.0768, 0.61424, 0.07568, 0.55)	(0.633, 0.7227811, 0633, 055)	(76.8)
Обсидіан	(0.06375, 0.05, 0.06625, 0.82)	(0.18275, 0.17, 0.2525, 0.82)	(0.332741, 0.328634, 0.346435, 0.82)	(38,4)
Нефрит	(0.135, 0.2225, 0.1575, 0.95)	(0.54, 0.89, 0.63, 0.95)	(0.316228, 0.95, 0.95, 0.95)	(12.8)
Рубін	(0.1745, 0.01175, 0.01175, 0.55)	(0.61424, 0.04136, 0.04136, 0.45)	(0.727811, 0.626959, 0.626959, 0.55)	(76.8)

За замовчуванням освітлення при створенні зображення в системі OpenGL відключене. Включається освітлення командою за допомогою команди

```
glEnable(GL_LIGHTING)
```

Без освітлення наприклад сфера завжди буде відображатися як коло, конус – як коло або трикутник в залежності від положення конуса, а якщо монотонний об’єкт освітлено рівномірно, не має можливості побачити його рельєф.

Фонове випромінювання - це світло, що настільки розподілений середовищем (предметами, стінами і так далі), що його напрямок визначити неможливо - здається, що він виходить звідусіль. Коли фоновий світло падає на поверхню, він однаково розподіляється у всіх напрямках.

Дифузний компонент - це світло, що йде з одного напрямку, таким чином, він виглядає яскравіше, якщо падає на поверхню під прямим кутом, і виглядає тьмяним, якщо стосується її лише побіжно. Однак, коли він падає на поверхню, він розподіляється однаково у всіх напрямах, тобто його яскравість однакова незалежно від того, з якого боку ви дивитеся на поверхню. Ймовірно, будь-яке світло, що виходить із певного напряму або позиції, має дифузний компонент.

Дзеркальне освітлення виходить з певного напряму і відбивається від поверхні в певному напрямку. Блискучий метал або пластик має високий дзеркальний компонент, а шматок килима або плюшева іграшка - ні. Ви можете думати про дзеркальності як про те, наскільки блискучим виглядає матеріал.

Крім фонового, дифузного і дзеркального кольорів, матеріали можуть також мати вихідний колір, що імітує світло, що виходить від самого об'єкта. У моделі освітлення OpenGLicходящій світло поверхні додає об'єкту інтенсивності, але на нього не впливають жодні джерела світла, і він не виробляє додаткового світла для сцени в цілому

Визначення вектора нормалі для кожної вершини кожного об'єкта

Нормалі об'єкта задають його орієнтацію щодо джерел світла. OpenGLиспользует нормаль вершини для визначення того, як багато світла ця вершина отримує від кожного джерела. У наведеному прикладі процес визначення нормалей відбувається всередині функції glutSolidSphere ().

Для правильного освітлення нормалі поверхні повинні мати одиничну довжину. Ви повинні бути обережні і пам'ятати, що матриця модельного перетворення не масштабує вектора нормалей автоматично, і результуючі нормалі можуть вже не мати одиничну довжину. Щоб переконатися, що нормалі мають одиничну довжину, можливо, вам доведеться викликати команду glEnable () з аргументами GL_NORMALIZE або GL_RESCALE_NORMAL.

GL_RESCALE_NORMAL веде до того, що кожний компонент вектора нормалі до поверхні буде помножений на одне і те ж число, вилучене з матриці модельних перетворень. Таким чином, операція працює коректно тільки у випадку, якщо нормалі масштабувати рівномірно і спочатку мали одиничну довжину.

GL_NORMALIZE - більш складна операція, ніж GL_RESCALE_NORMAL. Коли активовано механізм нормалізації (GL_NORMALIZE), спочатку обчислюється довжина вектора нормалі, а потім кожен з компонентів вектора ділиться на це число. Ця операція гарантує, що результуючі нормалі будуть мати одиничну довжину, але вона може бути більш витратною в сенсі швидкості, ніж просте масштабування нормалей.

Зауваження:

Деякі реалізації OpenGL можуть виконувати операцію GL_RESCALE_NORMAL не за рахунок простого масштабування, а шляхом все тієї ж повної нормалізації (GL_NORMALIZE). Однак не існує способу з'ясувати чи так це у вашій реалізації, і, в будь-якому випадку, ви не повинні покладатися на це.

Створення, позиціонування та включення одного або більше джерел світла

У прикладі 5-1 використовується всього один джерело білого світла. Його місце розташування задається викликом команди glLightfv (). У цьому прикладі для нульового джерела світла (GL_LIGHT0) задається білий колір для дифузного і дзеркального відображення. Якщо вам потрібно світло іншого кольору, змініть параметри glLight * ().

Крім того, ви можете додати до вашої сцені як мінімум 8 джерел світла різних кольорів. (Конкретна реалізація OpenGL може дозволяти й більше 8-ми джерел.) За замовчуванням колір усіх джерел світла крім GL_LIGHT0 - чорний. Ви можете розташовувати джерела світла скрізь, де тільки захочете, наприклад, близько до сцени (як настільну лампу) або нескінченно далеко за нею (симулюють сонячне світло). Крім того, ви можете керувати тим, випускає чи джерело сфокусований, вузький промінь або ширший. Пам'ятайте, що кожне джерело світла вимагає додаткових (і немалих)

розрахунків для відображення сцени, так що швидкодію вашого додаток залежить від кількості джерел.

Після налаштування параметрів джерела світла ви повинні активізувати його командою glEnable (). Крім того, ви повинні викликати команду glEnable () з аргументом GL_LIGHTING, щоб підготувати OpenGLк виконання розрахунків, пов'язаних з освітленням.

Вибір моделі освітлення

У OpenGL поняття моделі освітлення поділяється на 4 компоненти:

- Інтенсивність глобального фонового світла.
- Чи вважається положення точки спостереження локальним до сцени або нескінченно віддаленим.
- Чи повинен розрахунок освітленості проводитися по-різному для лицьових і зворотних граней об'єктів.
- Чи повинен дзеркальний колір відділятися від фонового і дифузного і накладатися на об'єкт після операцій текстурування.

Цей розділ пояснює, як задавати модель освітлення. Тут також обговорюється, як включати освітлення - тобто, як сказати OpenGL, що вона повинна робити розрахунок освітленості.

glLightModel * () - це команда, яка використовується для завдання всіх параметрів моделі освітлення. glLightModel * () приймає два аргументи: ім'я параметра моделі освітлення у вигляді константи і значення для цього параметра.

```
void glLightModel(if)(GLenum pname, TYPE param);
void glLightModel(if)v(GLenum pname, TYPE * param);
```

Встановлює властивості моделі освітлення. Встановлювана характеристика моделі освітлення визначається аргументом pname (таблиця 5-2). param задає величину, в яку встановлюється pname; якщо використовується векторна версія команди, то це вказівник на групу величин, якщо застосовується невекторная версія - у param міститься сама величина. Невекторная версія команди може використовуватися тільки для встановлення параметрів, що визначаються одній величиною (і не може застосовуватися для GL_LIGHT_MODEL_AMBIENT).

Таблиця 5-2. Значення за замовчуванням для параметра pname моделі освітлення
Імена параметрів Значення за замовчуванням Сенс

Назви параметрів	Значення за замовчуванням	Зміст
GL_LIGHT_MODEL_AMBIENT	(0.2,0.2,0.2,1.0)	RGBA інтенсивність всієї сцени
GL_LIGHT_MODEL_LOCAL_VIEWER	0.0 або GL_FALSE	спосіб обчислення кутів дзеркального відображення
GL_LIGHT_MODEL_TWO_SIDE	0.0 або GL_FALSE	вибір між одностороннім і двостороннім освітленням
GL_LIGHT_MODEL_COLOR_CONTROL	GL_SINGLE_COLOR	обчислюється чи дзеркальний колір окремо від фонового і дифузного

Глобальне фонове світло

Як обговорювалося раніше, кожне джерело світла може додавати до сцени фоновий світло. Крім того, може бути присутнім інший фоновий світло, що не належить ніякому конкретного джерела. Щоб задати RGBA інтенсивність такого глобального фонового світла, використовуйте параметр GL_LIGHT_MODEL_AMBIENT наступним чином:

```
GLfloat lmodel_ambient[] = (0.2,0.2,0.2,1.0);
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, lmodel_ambient);
```

У цьому прикладі значення для lmodel_ambient збігаються зі значеннями за замовчуванням для GL_LIGHT_MODEL_AMBIENT. Оскільки ці числа задають невелику кількість білого фонового світла, ви можете бачити об'єкти сцени навіть в тому випадку, якщо відсутні будь-які додаткові джерела світла.

Двостороннє освітлення

Розрахунок освітленості проводиться для всіх полігонів, чи є вони лицьовими або зворотними. Оскільки ви зазвичай налаштовуєте джерела світла, розмірковуючи про лицьових полігонах, зворотні можуть бути

невірно освітлені. У прикладі 5-1, де як об'єкт використовується сфера, завжди видно тільки лицьові грані, оскільки саме вони знаходяться зовні сфери. Таким чином, в даному випадку неважливо, як виглядають зворотні. Однак, якщо частина сфери буде відсічена і її внутрішня поверхня стане видимою, вам можливо захочеться, щоб ця поверхня була висвітлена у відповідності з заданими вами умовами освітлення; можливо вам також буде потрібно задати для зворотнього поверхні інші властивості матеріалу, ніж для лицьової. Коли ви включаєте двосторонню освітлення викликом:

```
glLightModeli(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE);
```

OpenGL вважає нормаль до зворотних поверхонь нормалі до особових, спрямовані в протилежному напрямку. Зазвичай це означає, що вектори нормалей для видимих зворотних полігонів (як і для лицьових) спрямовані на, а не від спостерігача. У результаті всі полігони висвітлюються правильно. Однак додаткові операції із забезпечення двостороннього освітлення роблять його більш повільним, ніж прийняте за замовчуванням одностороннє освітлення.

Щоб вимкнути двосторонню освітлення, змініть GL_TRUE в попередньому рядку коду на GL_FALSE. Ви також можете керувати тим, які межі OpenGL вважає лицьовими за допомогою команди glFrontFace ().

Відділення дзеркального кольору

При звичайному розрахунку освітленості, фоновий, дифузний, дзеркальний вклади джерела світла в загальну освітленість і інтенсивність вихідного світла обчислюються і просто складаються разом. За замовчуванням накладення текстури проводиться після розрахунку освітленості, в результаті дзеркальний полиск може виглядати каламутним, а текстура може бути перекручена. Якщо ж ви справите наступний виклик:

```
glLightModeli(GL_LIGHT_MODEL_COLOR_CONTROL, GL_SEPARATE_SPECULAR_COLOR);
```

OpenGL відділить розрахунок дзеркального освітлення від його застосування. Після цього виклику при розрахунку освітленості проводиться

два кольори на кожну вершину: первинний, що складається з суми всіх незеркальних внесків усіх джерел світла, і вторинний, що складається з суми всіх дзеркальних внесків усіх джерел світла. Під час текстурування з квітами текстури комбінується тільки первинний колір. Вторинний світло додається до комбінації первинного і текстурних кольорів після операції текстурування. Об'єкти, освітлені та текстуровані із застосуванням окремого дзеркального кольору, як правило, мають більш чіткий виділяється дзеркальний полиск.

Щоб відновити метод розрахунку освітленості за замовчуванням, викличте:

```
glLightModeli (GL_LIGHT_MODEL_COLOR_CONTROL, GL_SINGLE_COLOR);
```

Після цього виклику первинний колір буде складатися з усіх вкладів джерела світла в освітленість вершини: фонового, дифузного і дзеркального, а також з вихідного кольору матеріалу самого об'єкта. Ніякі додаткові внески в освітленість не додаються після текстурування.

Якщо ви не робите накладення текстури на об'єкт, відділення дзеркального кольору від інших не має ніякого сенсу.

Включення розрахунку освітленості

При використанні OpenGL вам необхідно включати (або вимикати) механізм розрахунку освітленості. Якщо цей механізм не включений, поточний колір просто асоціюється з поточною вершиною, не натискати жодної обчислень, що стосуються нормалей, джерел світла, моделі освітлення та властивостей матеріалу. Розрахунок освітленості включається командою:

```
glEnable (GL_LIGHTING);
```

і вимикається командою:

```
glDisable (GL_LIGHTING);
```

Крім того, вам потрібно включати кожне джерело світла, після того, як ви визначили його параметри. У прикладі 5-1 використовується тільки одне джерело світла - GL_LIGHT0:

```
glEnable (GL_LIGHT0);
```

Вказівка властивостей матеріалу

Ви бачили, як створити джерело світла з певними характеристиками і як задати потрібну модель освітлення. У цьому розділі описано, як задати властивості матеріалу для об'єктів на сцені: фоновий, дифузний і дзеркальний кольору, ступінь сяйва (наскільки блискучим виглядає об'єкт) і колір вихідного від об'єкта світла. Більшість властивостей матеріалу концептуально схожі на ті, які використовувалися при створенні джерел світла. Механізм із зазначення також аналогічний встановлення параметрів джерела світла за винятком того, що тут використовується команда glMaterial * ().

```
void glMaterial(if)(GLenum face, GLenum pname, TYPE param);
void glMaterial(if)v(GLenum face, GLenum pname, TYPE * param);
```

Задає властивість матеріалу для використання при розрахунку освітленості. Аргумент face може приймати значення GL_FRONT, GL_BACK або GL_FRONT_AND_BACK, вказуючи для яких граней об'єкта задається властивість матеріалу. Встановлюване властивість матеріалу визначається значенням аргументу pname, а його значення міститься в param (у вигляді покажчика на вектор величин у разі векторної версії команди або у вигляді самої величини при використанні невекторного варіанту). Невекторная версія команди працює тільки для параметра GL_SHININESS.Можливі значення для аргументу pname перераховані в таблиці

Назви параметрів	Значення за замовчуванням	Зміст
GL_AMBIENT	(0.2,0.2,0.2,1.0)	Фоновий колір матеріалу
GL_DIFFUSE	(0.8,0.8,0.8,1.0)	Диффузійний колір матеріалу
GL_AMBIENT_AND_DIFFUSE		Фоновий та диффузійний колір матеріалу
GL_SPECULAR	(0.0,0.0,0.0,1.0)	Дзеркальний колір
GL_SHININESS	0.0	Показний дзеркального відображення
GL_EMISSION	(0.0,0.0,0.0,1.0)	Вихідний колір матеріалу
GL_COLOR_INDEXES	(0,1,1)	Індекси фонового, диффузійного, дзеркального кольорів

Зауважте, що константа GL_AMBIENT_AND_DIFFUSE дозволяє вам одночасно встановити фоновий і дифузний кольору матеріалу в один і той же RGBAзначеніе. Двостороннє освітлення

Розрахунок освітленості проводиться для всіх полігонів, чи є вони лицьовими або зворотними. Оскільки ви зазвичай налаштовуєте джерела світла, розмірковуючи про лицьових полігонах, зворотні можуть бути невірно освітлені. У прикладі 5-1, де як об'єкт використовується сфера, завжди видно тільки лицьові грані, оскільки саме вони знаходяться зовні сфери. Таким чином, в даному випадку неважливо, як виглядають зворотні. Однак, якщо частина сфери буде відсічена і її внутрішня поверхня стане видимою, вам можливо захочеться, щоб ця поверхня була висвітлена у відповідності з заданими вами умовами освітлення; можливо вам також буде потрібно задати для зворотнього поверхні інші властивості матеріалу, ніж для лицьової. Коли ви включаєте двосторонню освітлення викликом:

```
glLightModeli (GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE);
```

OpenGL вважає нормаль до зворотних поверхонь нормалі до особових, спрямовані в протилежному напрямку. Зазвичай це означає, що вектори нормалей для видимих зворотних полігонів (як і для лицьових) спрямовані на, а не від спостерігача. У результаті всі полігони висвітлюються правильно. Однак додаткові операції із забезпечення двостороннього освітлення роблять його більш повільним, ніж прийняте за замовчуванням одностороннє освітлення.

Щоб вимкнути двосторонню освітлення, змініть GL_TRUE в попередньому рядку коду на GL_FALSE. Ви також можете керувати тим, які межі OpenGL вважає лицьовими за допомогою команди glFrontFace ().

Визначення властивостей матеріалу для об'єктів сцени

Властивості матеріалу об'єктів визначають, як він відображає світло і, таким чином, з якого реального матеріалу він зроблений (в зоровому сприйнятті). Оскільки взаємодія між поверхнею матеріалу і входять світлом досить складне, досить важко задати такі параметри матеріалу, щоб об'єкт мав певний, бажаний вид. Ви можете задавати фоновий, дифузний і дзеркальний кольору матеріалу і те, наскільки блискучим він буде виглядати.

У наведеному прикладі з допомогою команди glMaterialfv () були явно задані тільки дві властивості матеріалу: дзеркальний колір матеріалу і вихідний колір.

Приклад створення сфери (сірого кольору), та освітлення її:

```
#include <stdafx.h>
#include <windows.h>
#include <GL/gl.h>
#include <GL/glut.h>
#include <math.h>


void init(void)
{
   GLfloat mat_specular[]={1.0,1.0,1.0,1.0};//колір бліку
   GLfloat mat_shininess[]={10.0};//відбіваюча властивість
матеріалу
   GLfloat light_position[]={0.0,1.0,1.0,0.0};//позиція
освітлення
   GLfloat white_light[]={1.0,1.0,1.0,1.0};//колір освітленної
частини


   glClearColor(0.0,0.0,0.0,0.0);
   glShadeModel(GL_SMOOTH);
   glMaterialfv(GL_FRONT,GL_SPECULAR,mat_specular);
   glMaterialfv(GL_FRONT,GL_SHININESS,mat_shininess);
   glLightfv(GL_LIGHT0,GL_POSITION,light_position);
   glLightfv(GL_LIGHT0,GL_DIFFUSE,white_light);
   glLightfv(GL_LIGHT0,GL_SPECULAR,white_light);

   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
   glEnable(GL_DEPTH_TEST);
}

//Відображення
void display(void)
{
   glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
   glutSolidSphere(0.8,50,14);//параметри фігури
   glFlush();
}

//Зміна розмірів вікна
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,-
0.5*(GLfloat)h/(GLfloat)w,0.5*(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();

}

int main(int argc, char **argv)
{
   glutInit(&argc,argv);
   glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB|GLUT_DEPTH);
   glutInitWindowSize(500,500);
   glutInitWindowPosition(100,100);
   glutCreateWindow("Rendering a Lit Sphere");
   init();
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   glutMainLoop();
   return 0;
}
```

РІВЕНЬ “Б”

2.1. Встановити декілька додаткових джерел світла різної інтенсивності та в різних точках простору. Для створених джерел освітлення встановити індивідуальні значення напрямку, кута та інтенсивності розповсюдження світла.

2.2 Додати один прожектор з заданими параметрами у заданому положенні (Табл 2.2)

Табл. 2.2

	Кутова ширина світлового променя	**Концентрація світлового променя**	**Напрямок світла прожектора (x, y, z)**
1	90	0.9	(0.1,0.1,1.0)
2	85	0.8	(0.2,0.2,1.0)
3	56	1.0	(0.2,1.0,1.0)

4	54	0.7	(0.3,0.9,0.9)
5	21	0.6	(0.4,0.3,1.0)
6	15	0.5	(0.5,0.4,0.5)
7	77	0.4	(0.6,0.5,0.4)
8	68	0.3	(0.7,1.0,1.0)
9	63	0.9	(0.8,0.9,1.0)
10	22	1.0	(0.9,0.8,0.3)
11	44	0.6	(1.0,0.7,0.4)
12	76	1.0	(0.9,0.6,0.3)
13	49	0.7	(0.8,0.5,0.5)
14	62	0.4	(0.7,0.4,0.7)
15	75	0.3	(0.6,0.3,1.0)
16	23	0.7	(0.5,0.2,-1.0)
17	74	0.9	(0.4,1.0,0.4)
18	88	1.0	(0.3,0.1,1.0)
19	66	0.3	(0.2,0.3,0.2)
20	14	0.6	(0.1,0.6,0.3)
21	73	0.2	(0.7,0.7,0.4)
22	25	0.8	(0.6,0.2,1.0)
23	39	0.2	(0.8,0.3,0.5)
24	57	0.8	(0.9,0.4,1.0)
25	72	0.3	(0.4,1.0,1.0)

Створення джерел світла

Джерела світла мають декілька параметрів, таких як колір, позиція і напрямок. Наступні розділи пояснюють, як контролювати ці властивості, на що буде схожий результуючий джерело світла. Команда, яка використовується для зазначення всіх параметрів світла - це glLight * (). Вона приймає три аргументи: ідентифікатор джерела світла, ім'я властивості і бажане для нього значення.

```
void glLight(if)(GLenum light, GLenum pname, TYPE param);
void glLight(if)v(GLenum light, GLenum pname, TYPE * param);
```

Створює джерело, що задається параметром light (який може приймати значення GL_LIGHT0, GL_LIGHT1, ..., GL_LIGHT7). Задаються характеристика світла визначається аргументом pname у вигляді константи (таблиця 5-1). У параметрі param задається значення або значення, в які слід встановити характеристику pname. Якщо використовується векторна версія команди, param являє собою вектор величин, а якщо невекторная, то param - одне єдине значення. Невекторная версія команди може використовуватися тільки для вказівки параметрів, чиє значення виражається одним числом.

Таблиця 5-1. Значення за замовчуванням для властивостей джерела світла

Імена параметрів	**Значення за замовчуванням**	**Сенс**
GL_AMBIENT	(0.0,0.0,0.0,1.0)	Інтенсивність фонового світла
GL_DIFFUSE	(1.0,1.0,1.0,1.0) або (0.0,0.0,0.0,1.0)	Інтенсивність дифузного світла (значення за замовчуванням для 0-го джерела - білий світ, для інших - чорний)
GL_SPECULAR	(1.0,1.0,1.0,1.0) або (0.0,0.0,0.0,1.0)	Інтенсивність дзеркального світла (значення за замовчуванням для 0-го джерела - білий світ, для інших - чорний)
GL_POSITION	(0.0,0.0,1.0,0.0)	Положення джерела світла (x, y, z, w)
GL_SPOT_DIRECTION	(0.0,0.0,-1.0)	Напрямок світла прожектора (x, y, z)
GL_SPOT_EXPONENT	0.0	Концентрація світлового променя
GL_SPOT_CUTOFF	180.0	Кутова ширина світлового променя
GL_CONSTANT_ATTENUATION	1.0	Постійний чинник ослаблення
GL_LINEAR_ATTENUATION	0.0	Лінійний фактор ослаблення
GL_QUADRATIC_ATTENUATION	0.0	Квадратичный фактор ослабления

Значення за замовчуванням для GL_DIFFUSE і GL_SPECULAR в таблиці 5-1 розрізняються для GL_LIGHT0 та інших джерел світла (GL_LIGHT1, GL_LIGHT2, ...). Для параметрів GL_DIFFUSE і GL_SPECULAR джерела світла GL_LIGHT0 значення за замовчуванням - (1.0, 1.0, 1.0, 1.0). Для інших джерел світла значення тих же параметрів за замовчуванням - (0.0, 0.0, 0.0, 1.0).

У прикладі використання glLight * ():

Вказівка кольорів і позиції для джерел світла

```
GLfloat light_ambient[]={0.0,0.0,0.0,1.0};
GLfloat light_diffuse[]={1.0,1.0,1.0,1.0};
GLfloat light_specular[]={1.0,1.0,1.0,1.0};
GLfloat light_position[]={1.0,1.0,1.0,0.0};
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 );
```

Як ви можете бачити, для величин параметрів визначені масиви, і для установки цих параметрів кілька разів викликається команда glLightfv (). У цьому прикладі перші три виклики glLightfv () є надлишковими, оскільки вони використовуються для вказівки таких же значень параметрів GL_AMBIENT, GL_DIFFUSE і GL_SPECULAR, які встановлені за замовчуванням.

Зауваження:

Пам'ятайте, що кожне джерело світла потрібно включити командою glEnable ().

Всі параметри команди glLight * () та їх можливі значення описуються в наступних розділах. Ці параметри взаємодіють з параметрами моделі освітлення конкретної сцени і параметрами матеріалу об'єктів.

Колір

OpenGL дозволяє вам асоціювати з кожним джерелом світла три різних параметра, пов'язаних з кольором: GL_AMBIENT, GL_DIFFUSE і GL_SPECULAR. Параметр GL_AMBIENT задає RGBA інтенсивність фонового світла, який кожен окремий джерело світла додає до сцени. Як ви

можете бачити у таблиці 5-1, за замовчуванням джерело сету не додає до сцени фонового світла, так як значення за замовчуванням для GL_AMBIENT дорівнює (0.0, 0.0, 0.0, 1.0). Саме ця величина була використана в прикладі 5-1. Якщо б для того ж джерела світла був заданий синій фоновий світло,

```
GLfloat light_ambient[] = (0.0,0.0,1.0,1.0);
glLightfv (GL_LIGHT0, GL_AMBIENT, light_ambient);
```

то результат був би таким, який показаний на малюнку 5-2.

Малюнок 5-2. Сфера, освітлена одним джерелом, який додає синій фоновий світло

Параметр GL_DIFFUSE напевно найбільш точно збігається з тим, що ви звикли називати «кольором світла». Він визначає RGBA колір дифузного світла, який окреме джерело світла додає до сцени. За замовчуванням для GL_LIGHT0 параметр GL_DIFFUSE дорівнює (1.0, 1.0, 1.0, 1.0), що відповідає яскравому білому світу. Значення за замовчуванням для всіх інших джерел світла (GL_LIGHT1, GL_LIGHT2, ..., GL_LIGHT7) дорівнює (0.0, 0.0, 0.0, 0.0).

Параметр GL_AMBIENT впливає на колір дзеркального відблиску на об'єкті. У реальному світі на об'єктах на зразок скляної пляшки є дзеркальний полиск відповідного висвітлення кольору (часто білого). Для створення ефекту реалістичності встановіть GL_SPECULAR в те ж значення, що й GL_DIFFUSE. За замовчуванням GL_SPECULAR дорівнює (1.0, 1.0, 1.0, 1.0) для GL_LIGHT0 і (0.0, 0.0, 0.0, 0.0) для інших джерел.

Зауваження: Альфа компонент всіх цих кольорів використовується, тільки якщо включено колірне накладення.

Позиція і ослаблення

Як було зазначено раніше, ви можете вибрати, чи слід вважати джерело світла розташованим нескінченно далеко від сцени або близько до неї. На джерела світла першого типу посилаються як на направлення (directional): ефект від нескінченно далекого розташування джерела полягає в тому, що всі промені його світла можуть вважатися паралельними до моменту досягнення об'єкта. Прикладом реального спрямованого джерела світла може служити сонце. Джерела другого типу називаються позиційними

(positional), оскільки їх точне положення на сцені визначає їх ефект і, зокрема, направлення з якого йдуть промені. Прикладом позиційного джерела світла є настільна лампа. Світло, що використовується у прикладі 5-1, є спрямованим:

```
GLfloat light_position [] = (1.0,1.0,1.0,0.0);
glLightfv (GL_LIGHT0, GL_POSITION, light_position);
```

Як видно, для установки GL_POSITION ви задаєте вектор з чотирьох величин (x, y, z, w). Якщо остання величина w дорівнює 0, відповідний джерело світла вважається направленим, і величини (x, y, z) визначають його напрям. Цей напрям перетворюється видовий матрицею. За замовчуванням параметру GL_POSITION відповідають значення (0, 0, 1, 0), що задає світло, спрямований уздовж негативного напрямку осі z. (Зауважте, що ніхто не забороняє вам створити світ, спрямований в (0, 0, 0), проте таке світло не дасть належного ефекту).

Якщо значення w не дорівнює 0, світло є позиційним, і величини (x, y, z) задають його місце розташування джерела світла в однорідних об'єктних координатах. Це положення перетворюється видовий матрицею і зберігається в видових координатах. Крім того, за замовчуванням позиційний світло випромінюється у всіх напрямках, але ви можете обмежити розповсюдження світла, створивши конус випромінювання, що визначає прожектор.

```
glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, 2.0);
glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, 1.0);
glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, 0.5);
```

Зауважте, що послаблюються і фонова, і дифузна, і дзеркальна інтенсивності. Лише вихідні від об'єктів інтенсивності і глобальна фонова інтенсивність не підлягають ослаблення. Також відзначте, що оскільки розрахунок ослаблення вимагає додаткової операції ділення (і можливо інших операцій) для кожного обчислюваного кольору, використання послаблюємо джерел світла може сповільнити роботу додатка.

Прожектори

Як було зазначено раніше, ви можете змусити позиційний джерело світла діяти в якості прожектора, обмеживши поширення світла конусом випромінювання. Щоб створити прожектор, вам потрібно визначити бажану ширину світлового конуса. (Пам'ятаєте, що оскільки прожектор є позиційним джерелом світла, вам необхідно розташувати його в бажаному місці. Також зауважте, що ніхто не забороняє вам створювати прожектори з направлених джерел світла, але в цьому випадку ви напевно не отримаєте бажаний результат.) Щоб задати кут між віссю конуса і променем, що йдуть уздовж, ребра конуса, використовуйте параметр GL_SPOT_CUTOFF. Тоді кут між ребрами конуса, освіченими вершиною і кінцями діаметру підстави, буде дорівнює заданому вами кутку, помноженому на 2.

Зауважте, що світло не випромінюється за ребрами конуса. За замовчуванням, робота з прожекторами заблокована, оскільки параметр GL_SPOT_CUTOFF дорівнює 180.0. Ця величина означає, що світло випромінюється у всіх напрямках (кут при вершині дорівнює 360 градусам, що взагалі не визначає конус). Значення для GL_SPOT_CUTOFF повинні потрапляти в діапазон [0.0, 90.0] (значення також може бути одно спеціальної величиною 180.0). Наступний рядок встановлює параметр GL_SPOT_CUTOFF рівним 45 градусам:

```
glLightf (GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);
```

Вам також потрібно задати напрямок прожектора, яке визначає центральну вісь світлового конуса:

```
GLfloat spot_direction[] = (-1.0, -1.0, 0.0);
glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, spot_direction);
```

Напрямок вказується в об'єктних координатах. За замовчуванням напрямок – (0.0, 0.0, -1.0), так що якщо ви явно не змінюєте параметр GL_SPOT_DIRECTION, джерело світла направлений вздовж негативного напрямку осі z. Також майте на увазі, що напрям прожектора перетворюється видовий матрицею також, як якщо б це був вектор нормалі, і результат зберігається в видових координатах.

На додаток до кута світлового конуса та напрямку прожектора, існує два способи управління розподілом інтенсивності світла всередині конуса.

По-перше, ви можете задати фактор ослаблення світла, який перемножується з інтенсивністю. Ви також можете встановити параметр GL_SPOT_EXPONENT (який за замовчуванням дорівнює 0), щоб керувати концентрацією світла. Світло має найвищу інтенсивність в центрі конуса. При русі від центру до ребер він послаблюється з коефіцієнтом рівним косинус кута між напрямком світла і напрямом від джерела світла до освітлюваної вершині, зведеного в ступінь GL_SPOT_EXPONENT. Таким чином, більший експонентний коефіцієнт розкиду світла (GL_SPOT_EXPONENT) веде до більш фокусувати світло.

Кілька джерел світла

Як було відзначено, у вас на сцені може бути як мінімум вісім джерел світла (можливо більше, але це залежить від реалізації OpenGL). Оскільки OpenGL проводить обчислення для того, щоб визначити, скільки світла отримує кожна вершина від кожного джерела, збільшення джерел зльоту вагомо впливає на швидкодію. Константи, що використовуються для посилання на 8 джерел світла - це GL_LIGHT0, GL_LIGHT1, GL_LIGHT2, GL_LIGHT3 і так далі. У попередніх обговореннях параметри встановлювалися для джерела GL_LIGHT0. Якщо вам потрібні додаткові джерела світла, ви повинні задати і їх параметри. Також пам'ятайте, що значення параметрів за замовчуванням не збігаються для GL_LIGHT0 та інших джерел. У прикладі 5-3 визначається білий слабший прожектор:

Приклад 5-3. Друге джерело світла

```
GLfloatlight_ambient [] = (0.2,0.2,0.2,1.0);
GLfloat light_diffuse [] = (1.0,1.0,1.0,1.0);
GLfloat light_specular [] = (1.0,1.0,1.0,1.0);
GLfloat light_position []={- 2.0,2.0,1.0,1.0);
GLfloat spot_direction []={- 1.0, -1.0,0.0);
glLightfv (GL_LIGHT1, GL_AMBIENT, light_ambient);
glLightfv (GL_LIGHT1, GL_DIFFUSE, light_diffuse);
glLightfv (GL_LIGHT1, GL_SPECULAR, light_specular);
glLightfv (GL_LIGHT1, GL_POSITION, light_position);
glLightf (GL_LIGHT1, GL_CONSTANT_ATTENUATION, 1.5);
glLightf (GL_LIGHT1, GL_LINEAR_ATTENUATION, 0.5);
glLightf (GL_LIGHT1, GL_QUADRATIC_ATTENUATION,
0.2);
```

```
glLightf (GL_LIGHT1, GL_SPOT_CUTOFF, 45.0);
glLightfv (GL_LIGHT1, GL_SPOT_DIRECTION,
spot_direction);
glLightf (GL_LIGHT1, GL_SPOT_EXPONENT, 2.0);
glEnable (GL_LIGHT1);
```

Якщо ці рядки додати приклад 5-1, сфера буде освітлена двома джерелами: одним спрямованим і одним прожектором.

Рівень “B”

Завдання до роботи:

3.1. Створити довільний обєкт (3-15 точок). Перемістити джерело світла навколо цього об’єкта (таблиця 3). Врахувати положеня об’єкту (чорний колір), початкову позицію джерела світла (зелений) та його траєкторію (червоний)

Таблиця 3 Завдання до роботи.

№	Рисунок	№	Рисунок	№	Рисунок
1		**2**		**3**	
4		**5**		**6**	
7		**8**		**9**	
10		**11**		**12**	
13		**14**		**15**	
16		**17**		**18**	
19		**20**		**21**	
22		**23**		**24**	

Освітлення має бути виконано в такий кольорах:

ВАРІАНТИ:

Червоний	4, 9, 14, 19, 24, 29, 34
Синій	3, 8, 13, 18, 23, 28, 33
Зелений	2, 7, 12, 17, 22, 27, 32
Жовтий	1, 6, 11, 16, 21, 26, 31

Чорний (чорно-білий) 5, 10, 15, 20, 25, 30

3.2. Створити джерело світла, яке буде переміщуватися разом з точкою спостереження

Управління позицією і напрямком джерел світла

OpenGL звертається з позицією і напрямком джерела світла так само, як з позицією геометричного примітиву. Іншими словами, джерело світла є суб'єктом тих же матричних перетворень, що й примітив. Говорячи більш визначено, коли команда glLight * () викликається для вказівки позиції або напряму джерела світла, ця позиція або напрямок перетворюються поточної видовий матрицею і зберігається в видових координатах. Це означає, що ви можете маніпулювати позицією і напрямком джерел світла, змінюючи вміст видовий матриці. (Проекційна матриця не надає впливу на позицію або напрямок джерела світла.) У цьому розділі пояснюється, як добитися трьох перерахованих далі ефектів, змінюючи місце в програмі (щодо видових і модельних перетворень), де задаються джерела світла:

- Позиція джерела світла залишається фіксованою
- Джерело світла рухається навколо нерухомого об'єкту
- Джерело світла рухається разом з точкою спостереження

Стаціонарне джерело світла

У найпростішому випадку, положення джерела світла не змінюється. Щоб домогтися цього ефекту, вам слід встановлювати позицію джерела світла після всіх використовуваних видових і / або модельних перетворень. Приклад 5-4 демонструє, як може виглядати відповідний об'єднаний код з init () і reshape ().

Приклад 5-4. Стаціонарний джерело світла

```
glViewport(0,0, (GLsizei) w, (GLsizei) h);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
if (w <= h)
   glOrtho(-1.5,1.5, -0.5 * (GLfloat) h / (GLfloat) w,
```

```
   0.5 * (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();

// Далі функції init ()
GLfloat light_position[] = (1.0,1.0,1.0,0.0);
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
```

Як ви бачите, спочатку встановлюється порт перегляду і проекційна матриця. Потім у видову матрицю завантажується одинична, і тільки після цього задається положення джерела світла. Оскільки використовується одинична матриця, явно встановлене положення джерела світла - (1.0, 1.0, 1.0) не змінюється множенням на нього видовий матриці. Після цього, оскільки на той момент ні позиція джерела, ні видова матриця не змінені, напрям світла залишається тим самим - (1.0, 1.0, 1.0).

Незалежно-рухоме джерело світла

Тепер припустимо, що вам потрібно обертати джерело світла навколо стаціонарного об'єкта або переміщати джерело світла навколо нього. Один зі способів зробити це полягає у визначенні джерела світла після специфічного модельного перетворення, яке змінює позицію джерела. Ви можете почати з викликів тих же команд в init () як і в попередньому розділі. Потім вам слід виконати потрібне модельне перетворення (на стеці видових матриць) і скинути позицію джерела (ймовірно, у функції display ()). Приклад 5-5 демонструє можливий код функції display ().

Приклад 5-5. Незалежне рух джерела світла

```
GLdouble spin;
void display ()
{
   GLfloat light_position [] = {0.0,0.0,1.5,1.0};
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glPushMatrix();
   gluLookAt(0.0,0.0,5.0,0.0,0.0,0.0,0.0,1.0,0.0);
   glPushMatrix();
```

```
    glRotated(spin, 1.0,0.0,0.0);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glPopMatrix();
    glutSolidTorus (0.275,0.85,8,15);
    glPopMatrix();
    glFlush();
}
```

spin - це глобальна змінна, значення якої, найімовірніше, визначається пристроєм введення. Функція display () викликає перемальовування сцени з джерелом світла зверненим навколо нерухомого Торус на spin градусів. Зауважте, що дві пари команд glPushMatrix () і glPopMatrix () ізолюють видові і модельні перетворення. Оскільки в прикладі 5-5 точка спостереження залишається незмінною, поточна матриця проштовхується в стек і далі до потрібного видове перетворення задається командою gluLookAt (). Далі вийшла, матриця знову проштовхується в стек перед зазначенням перетворення повороту командою glRotate (). Потім задається положення джерела світла в новій повернутою системі координат, таким чином, джерело світла виявляється повернутим щодо свого первинного положення. (Пам'ятаєте, що позиція джерела світла зберігається в видових координатах, що виходять після перетворення видовий матрицею.) Торус малюється після того, як матриця виштовхується з стека. Приклад 5-6 представляє собою лістинг програми, повертає джерело світла навколо нерухомого об'єкта. При натисканні лівої кнопки миші кут повороту джерела світла збільшується на 30 градусів. Позиція джерела світла представлена маленьким неосвітленим дротяним кубом.

Переміщення джерела світла за допомогою модельних перетворень:

```
.
#include <windows.h>
#include <GL/gl.h>
#include <GL/glut.h>
#include <math.h>

//Ініціалізація
void init(void)
{
   glClearColor(0.0,0.0,0.0,0.0);
   glShadeModel(GL_SMOOTH);
   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
   glEnable(GL_DEPTH_TEST);
}

void display(void)
{
   GLfloat position[]={0.0,0.0,1.5,1.0};//позиція
освітлення при зміні положення КУБУ

   glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
   glPushMatrix();
   glTranslatef(0.0,0.0,-5.0);//положення фігури в
просторі
   glPushMatrix();
   glRotated((GLdouble)spin,5.0,20.0,0.0);//траекторія
КУБУ
   glLightfv(GL_LIGHT0,GL_POSITION,position);
   glTranslated(0.0,0.0,1.5);//стартова позиція КУБУ
   glDisable(GL_LIGHTING);
   glColor3f(0.0,0.4,1.0);//колір КУБУ
   glutWireCube(0.1);//розмір КУБУ
   glEnable(GL_LIGHTING);
   glPopMatrix();
   glutSolidTorus(0.275,0.85,70,3);//Параметри фігури
   glPopMatrix();
   glutSwapBuffers();
}

//Зміна розміру вікна
void reshape(int w, int h)
{
```

```
   glViewport(5,0,(GLsizei) w, (GLsizei) h);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluPerspective(40,(GLfloat)w/(GLfloat)h,1.0,200.0);
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
}

void light_moving(void)
{
   spin=(spin+3)%360;//швидкість та повне обертання
КУБУ навколо центру
   glutPostRedisplay();
}

void mouse(int button,int state,int x, int y)
{
   switch(button)
   {
      case GLUT_LEFT_BUTTON:
         if(state==GLUT_DOWN)
            glutIdleFunc(light_moving);
         else
            glutIdleFunc(NULL);
         break;
   }
}

int main(int argc, char **argv)
{
   glutInit(&argc,argv);

glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
   glutInitWindowSize(500,500);
   glutInitWindowPosition(100,100);
   glutCreateWindow("Moving a Light with Modeling
Transformations");
   init();
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   glutMouseFunc(mouse);
   glutMainLoop();
   return 0;
}
```

Переміщення джерела світла разом з точкою спостереження

Щоб створити джерело світла, який пересувається разом з точкою спостереження, вам слід задати його позицію до видового перетворення. Тоді видове перетворення буде однаковим чином впливати і на джерело світла і на точку спостереження. Пам'ятайте, що положення джерела світла зберігається в видових координатах - це один з невеликого числа випадків, в яких видові координати вельми важливі. У прикладі 5-7 позиція джерела світла задається у функції init () і зберігається у видових координатах (0, 0, 0). Іншими словами, світло випромінюється з лінзи камери.

Приклад 5-7. Джерело світла, що переміщується разом з точкою спостереження

```
GLfloat light_position[]={0.0, 0.0, 0.0, 1.0};

glViewport(0,0,(GLsizei) w, (GLsizei) h);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(40.0,(GLfloat)w/(GLfloat)h,1.0,100.0
);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glLightfv(GL_LIGHT0,GL_POSITION,light_position);
```

Тепер, якщо точка спостереження переміститься, джерело світла переміститься разом з нею, зберігаючи дистанцію (0, 0, 0) щодо очі. У продовженні прикладу 5-7, наступним далі, глобальні змінні (ex, ey, ez) керують становищем точки спостереження, а (upx, upy, upz) визначають вектор верхнього напрямку. Функція display (), що викликається з циклу обробки повідомлень для перемальовування сцени, може виглядати наступним чином:

```
GLdlouble ex, ey, ez, upx, upy, upz;

void display()
{

glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
   glPushMatrix();
```

```
        gluLookAt(ex, ey, ez, 0.0, 0.0, 0.0, upx,
upy, upz);
        glutSolidTorus(0.275, 0.85, 8, 15);
    glPopMatrix();
    glFlush();
}
```

Коли перемальовується освітлений Торус, і джерело світла, і крапка спостереження переміщуються в один і той же місце. Зі зміною величин, що передаються до gluLookAt () (і переміщенням спостерігача), об'єкт ніколи не буде темним, оскільки він завжди освітлюється з позиції спостерігача. Навіть якщо ви не зміните координати позиції джерела світла явно, він все одно буде переміщатися, оскільки змінюється видова координатна система.

Цей метод може бути дуже корисний при емуляції ліхтаря на касці шахтаря. Іншим прикладом може бути свічка або лампа, що несуть у руці. Позиція джерела світла, що задаються викликом glLightfv (GL_LIGHTi, GL_POSITION, position), за змістом буде відповідати вказівкою відстаней джерела від спостерігача по x, y і z. Потім при зміні положення спостерігача, джерело світла буде залишатися на тому ж відносному відстані.