МИНОБРНАУКИ РОССИИ

Санкт-Петербургский государственный электротехнический университет « ЛЭТИ» им. В. И.Ульянова(Ленина)

Задание для лабораторной работы № 7

" Реализация трехмерного объекта

с использованием библиотеки OpenGL "

Преподаватель: Герасимова Т.В.

Санкт-Петербург

2021 г.

Разработать программу, реализующую представление разработанной вами трехмерной сцены с добавлением возможности формирования различного типа проекций теней, используя предложенные функции OpenGL и GLSL.

ПРЕОБРАЗОВАНИЕ ОБЪЕКТОВ

В OpenGL используются как основные три системы координат: левосторонняя, правосторонняя и оконная.

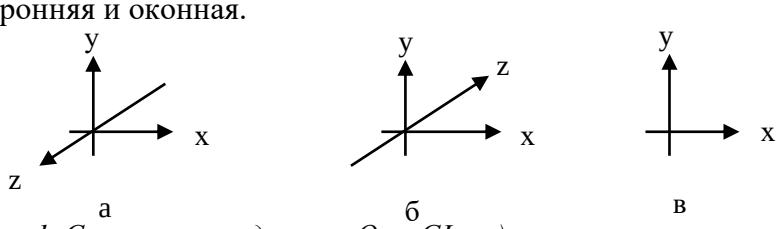

Рис..1. Системы координат в OpenGL: а) правосторонняя система; б) левосторонняя система; в) оконная система

Первые две системы являются трехмерными и отличаются друг от друга направлением оси z: в правосторонней она направлена на наблюдателя, в левосторонней – в глубину экрана. Ось x направлена вправо относительно наблюдателя, ось y – вверх.

Левосторонняя система используется для задания значений параметрам команды gluPerspective(), glOrtho(), которые будут рассмотрены в пункте 0. Правосторонняя система координат используется во всех остальных случаях. Отображение трехмерной информации происходит в двумерную оконную систему координат.

Строго говоря, OpenGL позволяет путем манипуляций с матрицами моделировать как правую, так и левую систему координат. Но на данном этапе лучше пойти простым путем и запомнить: основной системой координат OpenGL является правосторонняя система.

1. РАБОТА С МАТРИЦАМИ

Для задания различных преобразований объектов сцены в OpenGL используются операции над матрицами, при этом различают три типа матриц: модельно-видовая, матрица проекций и матрица текстуры. Все они имеют размер 4x4. Видовая матрица определяет преобразования объекта в мировых координатах, такие как параллельный перенос, изменение масштаба и поворот. Матрица проекций определяет, как будут проецироваться трехмерные объекты на плоскость экрана (в оконные координаты), а матрица текстуры определяет наложение текстуры на объект.

Умножение координат на матрицы происходит в момент вызова соответствующей команды OpenGL, определяющей координату (как правило, это команда glVertex*)

Для того чтобы выбрать, какую матрицу надо изменить, используется команда: void glMatrixMode (GLenum mode), вызов которой со значением параметра mode равным GL_MODELVIEW, GL_PROJECTION или GL_TEXTURE включает режим работы с модельно-видовой матрицей, матрицей проекций, или матрицей текстуры соответственно. Для вызова команд, задающих матрицы того или иного типа, необходимо сначала установить соответствующий режим.

Для определения элементов матрицы текущего типа вызывается команда void glLoadMatrix[f d] (GLtype *m), где m указывает на массив из 16 элементов типа float или double

в соответствии с названием команды, при этом сначала в нем должен быть записан первый столбец матрицы, затем второй, третий и четвертый. Еще раз обратим внимание: в массиве m матрица записана по столбцам.

Команда void glLoadIdentity (void) заменяет текущую матрицу на единичную.

Часто бывает необходимо сохранить содержимое текущей матрицы для дальнейшего использования, для чего применяются команды void glPushMatrix (void) и void glPopMatrix (void). Они записывают и восстанавливают текущую матрицу из стека, причем для каждого типа матриц стек свой. Для модельно-видовых матриц его глубина равна как минимум 32, для остальных – как минимум 2.

Для умножения текущей матрицы на другую матрицу используется команда void glMultMatrix[f d] (GLtype *m), где параметр m должен задавать матрицу размером 4x4. Если обозначить текущую матрицу за M, передаваемую матрицу за T, то в результате выполнения команды glMultMatrix текущей становится матрица M * T. Однако обычно для изменения матрицы того или иного типа удобно использовать специальные команды, которые по значениям своих параметров создают нужную матрицу и умножают ее на текущую.

В целом, для отображения трехмерных объектов сцены в окно приложения используется последовательность, показанная на рисунке 2.

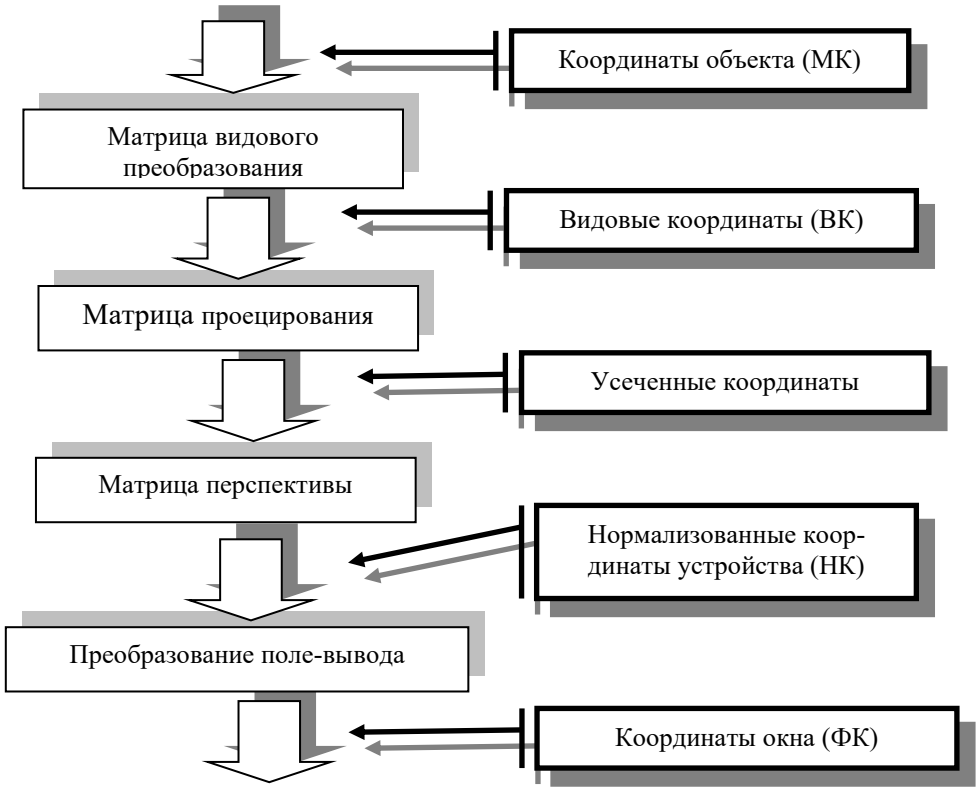

Рис.2. Преобразования координат в OpenGL

2. Модельно-видовые преобразования

К модельно-видовым преобразованиям будем относить перенос, поворот и изменение масштаба вдоль координатных осей. Для проведения этих операций достаточно умножить на соответствующую матрицу каждую вершину объекта и получить измененные координаты этой вершины:

$(x', y', z', 1)^T = M * (x, y, z, 1)^T$

где M – матрица модельно-видового преобразования. Перспективное преобразование и проектирование производится аналогично. Сама матрица может быть создана с помощью следующих команд:

```
void glTranslate[f d] (GLtype x, GLtype y, GLtype z);
void glRotate[f d] (GLtype angle, GLtype x, GLtype y,  GLtype z);
void glScale[f d] (GLtype x, GLtype y, GLtype z).
```

glTranlsate*() производит перенос объекта, прибавляя к координатам его вершин значения своих параметров.

glRotate*() производит поворот объекта против часовой стрелки на угол angle (измеряется в градусах) вокруг вектора (x,y,z).

glScale*() производит масштабирование объекта (сжатие или растяжение) вдоль вектора (x,y,z), умножая соответствующие координаты его вершин на значения своих параметров.

Все эти преобразования изменяют текущую матрицу, а поэтому применяются к примитивам, которые определяются позже. В случае, если надо, например, повернуть один объект сцены, а другой оставить неподвижным, удобно сначала сохранить текущую видовую матрицу в стеке командой glPushMatrix(), затем вызвать glRotate() с нужными параметрами, описать примитивы, из которых состоит этот объект, а затем восстановить текущую матрицу командой glPopMatrix().

Кроме изменения положения самого объекта, часто бывает необходимо изменить положение наблюдателя, что также приводит к изменению модельно-видовой матрицы. Это можно сделать с помощью команды void gluLookAt (GLdouble eyex, GLdouble eyey, GLdouble eyez, GLdouble centerx, GLdouble centery, GLdouble centerz, GLdouble upx, GLdouble upy, GLdouble upz), где точка (eyex,eyey,eyez) определяет точку наблюдения, (centerx, centery, centerz) задает центр сцены, который будет проектироваться в центр области вывода, а вектор (upx,upy,upz) задает положительное направление оси y, определяя поворот камеры. Если, например, камеру не надо поворачивать, то задается значение (0,1,0), а со значением (0,-1,0) сцена будет перевернута.

Строго говоря, эта команда совершает перенос и поворот объектов сцены, но в таком виде задавать параметры бывает удобнее. Следует отметить, что вызывать команду gluLookAt() имеет смысл перед определением преобразований объектов, когда модельно-видовая матрица равна единичной.

В общем случае матричные преобразования в OpenGL нужно записывать в обратном порядке. Например, если вы хотите сначала повернуть объект, а затем передвинуть его, сначала вызовите команду glTranslate(), а только потом – glRotate(). Ну а после этого определяйте сам объект.

3. Проекции

В OpenGL существуют стандартные команды для задания ортографической (параллельной) и перспективной проекций. Первый тип проекции может быть задан командами

void glOrtho (GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far) и void gluOrtho2D (GLdouble left, GLdouble right, GLdouble bottom, GLdouble top);

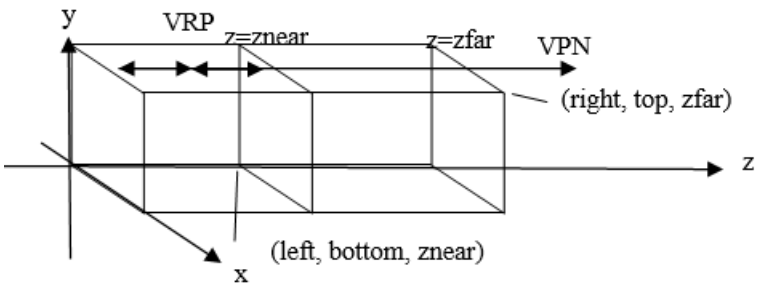

Рис. 3. Ортографическая проекция

Первая команда создает матрицу проекции в усеченный объем видимости (параллелепипед видимости) в левосторонней системе координат. Параметры команды задают точки (left, bottom, znear) и (right, top, zfar), которые отвечают левому нижнему и правому верхнему углам окна вывода. Параметры near и far задают расстояние до ближней и дальней плоскостей отсечения по удалению от точки (0,0,0) и могут быть отрицательными.

Во второй команде, в отличие от первой, значения near и far устанавливаются равными –1 и 1 соответственно. Это удобно, если OpenGL используется для рисования двумерных объектов. В этом случае положение вершин можно задавать, используя команды glVertex2*().

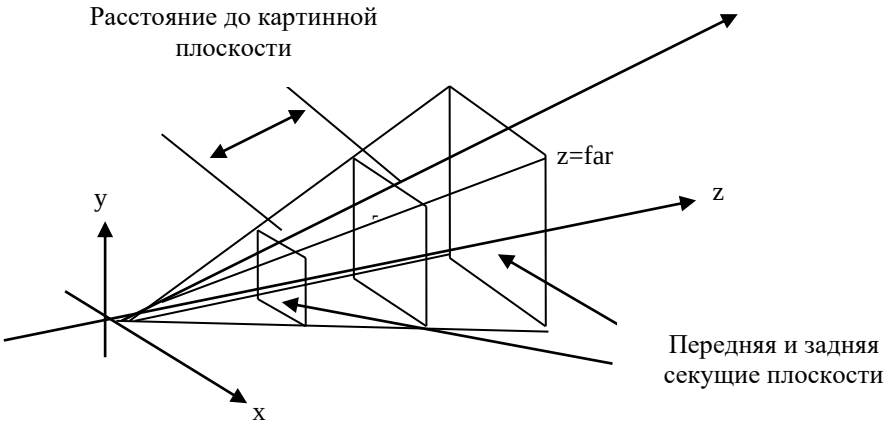

Рис. 4. Перспективная проекция

Перспективная проекция определяется командой void gluPerspective (GLdouble angley, GLdouble aspect, GLdouble znear, GLdouble zfar) , которая задает усеченный конус видимости в левосторонней системе координат. Параметр angley определяет угол видимости в градусах по оси y и должен находиться в диапазоне от 0 до 180. Угол видимости вдоль оси x задается параметром aspect, который обычно задается как отношение сторон области вывода (как правило, размеров окна) Параметры zfar и znear задают расстояние от наблюдателя до плоскостей отсечения по глубине и должны быть положительными. Чем больше отношение zfar/znear, тем хуже в буфере глубины будут различаться расположенные рядом поверхности, так как по умолчанию в него будет записываться ‘сжатая’ глубина в диапазоне от 0 до 1 (см. п. 0.).

Прежде чем задавать матрицы проекций, не забудьте включить режим работы с нужной матрицей командой glMatrixMode(GL_PROJECTION) и сбросить текущую, вызвав glLoadIdentity(). Например:

/* ортографическая проекция */

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(0, w, 0, h, -1.0, 1.0);
```

Область вывода

После применения матрицы проекций на вход следующего преобразования подаются так называемые усеченные (clipped) координаты. Затем находятся нормализованные координаты вершин по формуле:

$(x_n, y_n, z_n)^T = (x_c/w_c , y_c/w_c, z_c/w_c)^T$

Область вывода представляет собой прямоугольник в оконной системе координат, размеры которого задаются командой:

void glViewPort (GLint x, GLint y, GLint width, GLint height)

Значения всех параметров задаются в пикселах и определяют ширину и высоту области вывода с координатами левого нижнего угла (x,y) в оконной системе координат. Размеры оконной системы координат определяются текущими размерами окна приложения, точка (0,0) находится в левом нижнем углу окна.

Используя параметры команды glViewPort(), OpenGL вычисляет оконные координаты центра области вывода (o_x,o_y) по формулам $o_x=x+width/2$, $o_y=y+height/2$.

Пусть $p_x=width$, $p_y=height$, тогда можно найти оконные координаты каждой вершины:

$(xw, yw, zw)T = ((px/2) xn+ ox , (py/2) yn+ oy , [(f-n)/2] zn+(n+f)/2)T$

При этом целые положительные величины n и f задают минимальную и максимальную глубину точки в окне и по умолчанию равны 0 и 1 соответственно. Глубина каждой точки записывается в специальный буфер глубины (z-буфер), который используется для удаления невидимых линий и поверхностей. Установить значения n и f можно вызовом функции

void glDepthRange (GLclampd n, GLclampd f);

Команда glViewPort() обычно используется в функции, зарегистрированной с помощью команды glutReshapeFunc(), которая вызывается, если пользователь изменяет размеры окна приложения.

4. МОДЕЛИ ОСВЕЩЕНИЯ

Механизм отражения света от текущей поверхности очень сложен и зависит от многих факторов. Некоторые из них являются геометрическими - например, относительные направления источника света, глаза наблюдателя и нормали к поверхности. Другие факторы относятся к характеристикам поверхности, таким как шероховатость, и к цвету этой поверхности.

Модель закрашивания определяет, как свет рассеивается по поверхности или отражается от нее. Здесь рассматриваются простые модели закрашивания, в особенности ахроматическое освещение, у которого есть только яркость, но не цвет. Ахроматическое освещение дает только оттенки серого цвета, и поэтому оно описывается единственной величиной: его интенсивностью. В модели закрашивания, часто используемой в компьютерной графике, предполагается, что объекты сцены освещаются двумя типами источников: точечными источниками света и фоновым светом. Эти источники света "сверкают" на различных

поверхностях объектов, и падающий свет взаимодействует с поверхностью одним из трех возможных способов:

- некоторая часть поглощается поверхностью и превращается в тепло;
- некоторая часть отражается от поверхности;
- некоторая часть проходит внутрь объекта, как в случае куска стекла.

Если весь падающий свет поглощается, то данный объект воспринимается как черный, поэтому его называют абсолютно черным телом. Если весь свет проходит сквозь объект, то он виден только в результате эффекта рефракции.

Здесь рассматривается та составляющая света, которая отражается от поверхности или рассеивается ею. Некоторая часть этого отраженного света движется как раз в таком направлении, чтобы достигнуть глаза, вследствие чего объект становится виден. Количество света, падающего в глаз, прежде всего, определяется геометрией окружения. Различают два типа отражения падающего света:

- диффузное рассеяние происходит, когда часть падающего света слегка проникает внутрь поверхности и излучается обратно равномерно по всем направлениям. Диффузный свет сильно взаимодействует с поверхностью, поэтому его цвет обычно зависит от природы, из которого сделана эта поверхность;
- зеркальные отражения больше похожи на зеркало и имеют ярко выраженную направленность: падающий свет не поглощается объектом, а отражается прямо от его наружной поверхности. Это порождает блики, и поверхность выглядит блестящей. В простейшей модели зеркального света отраженный свет имеет такой же цвет, что и падающий свет, что делает материал похожим на пластмассу. В более сложной модели цвет отраженного света пробегает интервал бликов, что дает лучшее приближение металлических поверхностей.

4.1. Геометрические составляющие отраженного света

Для того чтобы вычислить диффузный и зеркальный компоненты света, необходимо найти три вектора. На рис. 4.1. показаны три главных вектора, необходимых для нахождения количества света, попадающего в глаз из точки P:

- нормаль m к поверхности в точке P;
- вектор v, соединяющий точку P с глазом наблюдателя;
- вектор s, соединяющий точку P с источником света.

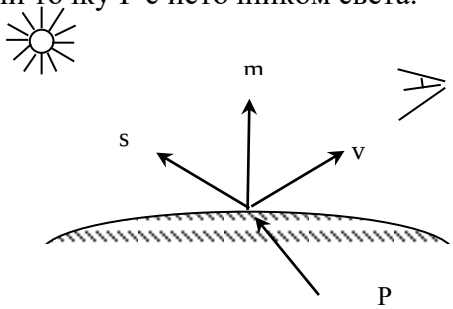

Рис. 4.1. Основные направления векторов, используемых при вычислении отраженного света

Углы между этими тремя векторами составляют основу для вычисления интенсивностей освещения. Обычно эти углы вычисляются в мировых координатах, поскольку при некоторых преобразованиях (таких, как перспективное преобразование) углы не сохраняются.

4.1.1. Диффузная компонента

Диффузный компонент представляет собой свет, исходящий из одного направления, поэтому он ярче, когда падает прямо на поверхность, чем, когда он просто скользит по поверхности. Как только он сталкивается с поверхностью, он равномерно рассеивается во всех направлениях, так что он кажется равномерно ярким светом, независимо от положения наблюдателя. Любой свет, исходящий из определенного места или направления, должен содержать диффузный компонент.

Предположим, что свет падает от точечного источника света на одну из сторон микрограни поверхности. Какая-то доля света с этой стороны микрограни диффузно переизлучается во всех направлениях. А какая-то часть этой переизлучаемой доли достигает глаза с интенсивностью, обозначаемой Id.

Поскольку рассеяние одинаково во всех направлениях, ориентация микрограни относительно глаза не имеет значения. Поэтому Id не зависит от угла между векторами m и v (кроме случая v•m<0 - скалярное произведение, это говорит о том, что микрогрань не видна глазу, Id=0). С другой стороны, количество света, освещающего данную грань, зависит от ориентации грани по отношению к источнику света: оно пропорционально "наблюдаемой" источником площади грани, то есть видимой площади грани.

Соотношение между яркостью и ориентацией поверхности часто называют законом Ламберта. Для малых углов Q яркость слабо зависит от угла, поскольку косинус в окрестности нуля изменяется медленно. Однако при приближении угла Q к 90 яркость быстро снижается до нуля.

Поскольку cos(Q) является скалярным произведением ортов s и m. Тогда можно получить следующее выражение [1] для интенсивности диффузного компонента:

$$I_d = I_s p_d \frac{s \bullet m}{|s||m|},$$

где I_s - интенсивность источника света, а p_d - коэффициент диффузного отражения.

Если грань направлена в сторону от глаз, то это скалярное произведение отрицательно, и в этом случае мы принимаем $I_d=0$. Следовательно, более точная формула [2] для диффузного компонента имеет вид:

$$I_d = I_s p_d \max\left(\frac{s \bullet m}{|s||m|}, 0\right).$$

На самом деле механизм диффузного отражения намного сложнее, чем эта упрощенная модель. Коэффициент отражения p_d зависит от длины волны (от цвета) падающего света, от угла Q, и от различных физических свойств поверхности. Однако для простоты и для уменьшения времени вычислений при визуализации изображений эти влияния обычно игнорируются. Для каждой поверхности "разумное" значение коэффициента p_d выбирается из реалистичности результирующего изображения, иногда методом проб и ошибок.

4.1.2. Зеркальная компонента

Зеркальный (отраженный) свет исходит из определенного направления и при столкновении с поверхностью отражается в определенном направлении.

Реальные объекты не рассеивают свет равномерно во всех направлениях, поэтому к модели закраски добавляются зеркальный компонент. Зеркальное отражение порождает блики,

которые могут существенно увеличить реалистичность изображения, заставив объекты блестеть.

Существует модель поведения отраженного света по Фонгу. Эта модель легко реализуется, и OpenGL поддерживает хорошее приближение к ней. Блики, создаваемые отраженным светом по Фонгу, придают объекту вид сделанного из пластмассы, поэтому модель Фонга хороша, когда предназначается для объектов, выполненных из блестящего пластика или стекла. С объектами, имеющими блестящую металлическую поверхность, данная модель работает хуже, однако, путем подбора параметров цвета с помощью OpenGL аппроксимация по-прежнему возможна.

На самом деле физический механизм зеркального отражения намного сложнее, чем предлагается в модели Фонга. В более реалистичной модели коэффициент зеркального отражения зависит от длины волны (от цвета), и от угла падения света. В этом случае коэффициент отражения - коэффициент Френеля, который описывает физические характеристики отражения света от поверхностей из материалов определенного класса.

4.1.3. Роль фонового света

Диффузный и зеркальный компоненты отраженного света находятся путем упрощения "правил", по которым физический свет отражается от физических поверхностей. Включение зависимости этих компонентов от относительных положений глаза, объекта и источников света значительно улучшает реалистичность изображения при визуализациях, которые просто заполняют каркасную модель тенями.

Фоновое (рассеянное) освещение представляет собой свет, настолько рассеянный в окружающей среде, что его направление невозможно определить – создается впечатление, что он исходит из всех направлений. Фоновое освещение в комнате имеет большую составляющую рассеянного света, так как большая часть света, достигающего глаз, до этого была отражена множеством поверхностей. Когда рассеянный свет сталкивается с поверхностью, он равномерно рассеивается во всех направлениях.

Если использовать только диффузное и зеркальное отражение, то любые части поверхности, заслоненные от точечного источника, вообще не получают света и поэтому рисуются черным цветом.

Для того чтобы преодолеть трудности, связанные со сплошными черными тенями, вводят равномерное свечение фона - фоновый свет. Источник фонового света не располагается в каком-либо определенном месте, и этот свет распространяется во всех направлениях одинаково. Этот источник света характеризуется интенсивностью I_a. Каждой грани в данной модели соответствует определенное значение коэффициента фонового отражения p_a, а член $I_a * p_a$ просто добавляется к этому диффузному и зеркальному свету, который попадает в глаз из каждой точки P на этой грани. Значения I_a и p_a обычно подбираются экспериментально путем варьирования различных величин и выбора наиболее подходящих. Недостаток фонового света делает тени слишком глубокими и резкими; избыток же его делает изображение размытым и мягким.

4.1.4. Комбинирование компонентов освещения

Теперь можно сложить три компоненты освещения - диффузный, зеркальный и фоновый, чтобы получить суммарное количество света I [1]., которое попадает в глаз из точки P:

$$I = I_a p_a + I_d p_d \times lambert + I_{sp} p_s \times phong^f ,$$

где $lambert = \max\left(0, \frac{s \bullet m}{|s||m|}\right)$ и $phong = \max\left(0, \frac{h \bullet m}{|h||m|}\right)$.

Величина I зависит от различных интенсивностей источников и коэффициентов отражения объекта, а также от соотношения положений точки P, глаза и точечного источника света. В OpenGL предоставляется возможность отдельного задания интенсивностей диффузной и зеркальной. На практике обе эти интенсивности обычно имеют одну и ту же величину.

4.2. Добавление цвета

Расширим рассматриваемую модель закрашивания на случай отражения цветного освещения от цветных поверхностей. Свет любого цвета может быть синтезирован путем сложения определенных количеств красного, зеленого и синего. При работе с цветными источниками и цветными поверхностями мы вычисляем каждый компонент цвета отдельно и затем просто складываем их, чтобы получить окончательный цвет отраженного света. Тогда для вычисления красного, зеленого и синего компонентов отраженного света необходимо применить базовое уравнение [2]:

Ir=Iar* par+Idr* pвr*lambert+Ispr* psr*phongf,
Ig=Iag* pag+Idg* pвg*lambert+Ispg* psg*phongf,
Ib=Iab* pab+Idb* pвb*lambert+Ispb* psb*phongf.

При этом делается предположение о наличии у источников света трех типов цвета:

Фоновый = (Iar,Iag,Iab),
Диффузный = (Idr,Idg,Idb),
Зеркальный = (Ispr,Ispg,Ispb).

Обычно цвета диффузного и зеркального освещения одинаковы. Кроме того, переменные lambert и phong не зависят от того, компонент какого цвета вычисляется, так что их нужно вычислить только один раз. Развивая этот подход, следует определить девять коэффициентов отражения: Коэффициенты фонового отражения - par, pag, pab; Коэффициенты диффузного отражения - pdr, pdg, pdb; Коэффициенты зеркального отражения - pspr, pspg, pspb.

Коэффициенты фонового и диффузного отражения определяются цветом самой поверхности. Под "цветом" поверхности понимается тот цвет, который отражается от нее при освещении ее белым светом. Поэтому поверхность является красной, если она выглядит красной при освещении белым цветом. Если же поверхность освещается каким-либо другим цветов, то она продемонстрирует совершенно иной цвет.

В силу своего названия зеркальный компонент часто имеет такой же цвет, что и источник. Например, из опыта известно, что блик, видимый на гладком красном яблоке, освещенном желтым цветом, имеет скорее желтый, чем красный цвет. Такой же эффект можно наблюдать с блестящими предметами, изготовленными из материала типа пластика. Для создания на пластмассовой поверхности отражающих бликов нужно придать коэффициентам зеркального отражения, используемым в уравнении, одно и то же значение, так чтобы коэффициенты отражения стали по своей природе "серыми" и не изменяли цвет падающего света.

4.3. Общее уравнение отраженного света

Расширим теперь уравнение интенсивности отраженного света так, чтобы оно включало в себя вклады дополнительных величин, вычисляемых OpenGL. Полный комплект красного цвета задается выражением [1,2]:

Ir=er+Imr*par+Σ atteni*spoti*(Iiar* par+Iidr* pвr*lamberti+Iispr* psr*phongfi).

Выражения для зеленого и синего компонентов задаются аналогично. Слагаемое er означает эмиссионный свет, а Imr - глобальный фоновый свет, введенный в модель освещения. Знак суммы означает, что для всех источников суммируется вклады фонового, диффузного и зеркального света. Для i-го источника atteni является коэффициентом ослабления света; spoti - это коэффициент прожектора. Все эти члены должны вычисляться заново для каждого источника света. Если в результате вычислений интенсивность получается больше, чем 1.0, то OpenGL урезает его до этого значения, так как максимальная яркость любого компонента света составляет - 1.0.

4.4. Закраска и графический конвейер

На рис. 4.2, показано на каком этапе работы так называемого графического конвейера происходит закраска. Основная идея состоит в том, что вершины сетки посылаются в конвейер вместе со связанными с ними нормалями, и все вычисления по закраске осуществляются именно с вершинами.

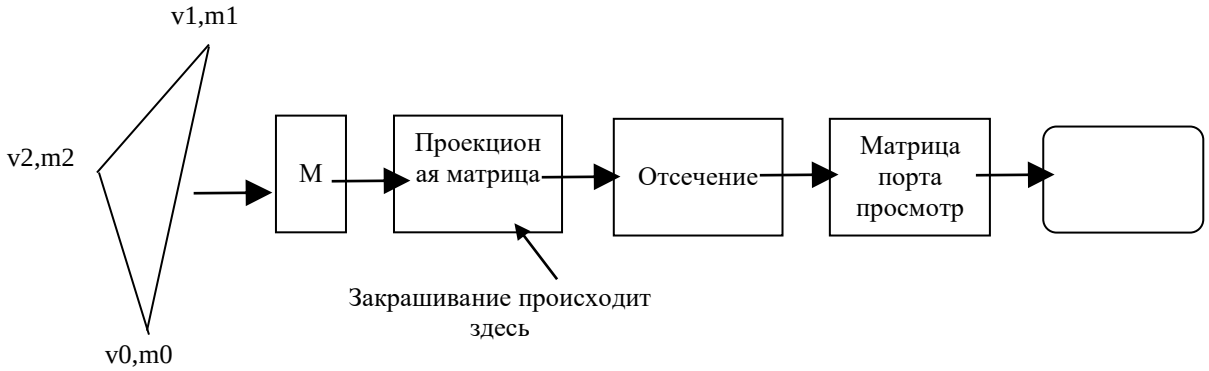

Рис. 4.2. Графический конвейер

В процессе визуализации треугольника с вершинами v0, v1, v2. С каждой вершиной vi связана нормаль mi. Эти величины пересылаются в конвейер с помощью следующих вызовов

```
GlBegin(GL_POLYGON)
for (int i=0;i<3;i++)
{ glNormal3f(norm[i].x,norm[i].y,norm[i].z);
glVertex3f(pt[i].x,pt[i].y,pt[i].z);
}
glEnd();
```

Вызов программы glNormal3f() устанавливает "текущую нормаль", которая применяется ко всем вершинам, последовательно пересылаемым в конвейер с помощью glVertex3f(). Эта нормаль остается текущей вплоть до ее изменения при следующем вызове glNormal3f(). В вышеприведенном коде с каждой вершиной связывается новая нормаль.

Вершины преобразуются с помощью матрицы моделирования M, которая эффективно переводит их в координаты камеры (глаза). Нормали тоже подвергаются преобразованиям, однако, векторы преобразуются не так, как точки. При преобразовании точек поверхности посредством матрицы M нормаль m в каждой точке превращается в нормаль M-Tm на

преобразованной поверхности, где M-T - транспонированная обратная к матрице M матрица. OpenGL выполняет эти вычисления нормалей автоматически. OpenGL предоставляет возможность задавать различные источники света и их расположение. Источники света тоже являются объектами, и координаты источников света также преобразуются посредством матрицы моделирования-вида.

Таким образом, после преобразования моделирования-вида все величины будут выражены в координатах камеры. С этой точки зрения применима модель уравнения интенсивностей с учетом цвета по каждой компоненте, причем к каждой вершине "привязан" цвет. Вычисление этого цвета требует знания векторов m, s, v, однако все они в этот момент доступны в конвейере.

По мере дальнейшего продвижения по конвейеру создается элемент, содержащий псевдоглубину, после чего вершины подвергаются перспективному преобразованию. К каждой вершине прикреплена информация о цвете. Вершины проходят через преобразование порта просмотра, где они преобразуются в экранные координаты

4.5. Использование источников света в OpenGL

Для создания реалистических изображений необходимо определить как свойства самого объекта, так и свойства среды, в которой он находится. Первая группа свойств включает в себя параметры материла, из которого сделан объект, способы нанесения текстуры на его поверхность, степень прозрачности объекта. Ко второй группе можно отнести количество и свойства источников света, уровень прозрачности среды. Все эти свойства можно задавать, используя соответствующие команды OpenGL.

В OpenGL содержится целый ряд функций для установки и использования источников света, а также для придания поверхности свойств определенного материала. Довольно трудно охватить все их возможные варианты и детали, поэтому рассмотрим лишь основные - как устанавливать на сцене различные виды источников света.

4.5.1. Создание источника света

Добавить в сцену источник света можно с помощью команд

void glLight[i f](GLenum light, GLenum pname, GLfloat param);

void glLight[i f](GLenum light, GLenum pname, GLfloat *params).

Параметр light однозначно определяет источник, и выбирается из набора специальных символических имен вида GL_LIGHTi, где i должно лежать в диапазоне от 0 до GL_MAX_LIGHT, которое не превосходит восьми.

Рассмотрим назначение остальных двух параметров (вначале описываются параметры для первой команды, затем для второй) pname:

GL_SPOT_EXPONENT параметр param должен содержать целое или вещественное число от 0 до 128, задающее распределение интенсивности света. Этот параметр описывает уровень сфокусированности источника света или задает экспоненциальное распределение интенсивности светового пучка прожектора. Значение по умолчанию: 0 (рассеянный свет);

GL_SPOT_CUTOFF параметр param должен содержать целое или вещественное число между 0 и 90 или равное 180, которое определяет максимальный угол разброса света. Значение этого параметра есть половина угла в вершине конусовидного светового потока, создаваемого

источником. Подробнее будет рассмотрено ниже. Значение по умолчанию: 180 (рассеянный свет);

GL_SPOT_DIRECTION параметр params должен содержать четыре целых или вещественных числа, которые определяют направление света. Значение по умолчанию: (0.0,0.0,-1.0,1.0);

GL_AMBIENT параметр params должен содержать четыре целых или вещественных значения цветов RGBA, которые определяют цвет фонового освещения. Значение по умолчанию: (0.0,0.0,0.0,1.0);

GL_DIFFUSE параметр params должен содержать четыре целых или вещественных значения цветов RGBA, которые определяют цвет диффузного освещения. Значение по умолчанию: (1.0,1.0,1.0,1.0) для LIGHT0 и (0.0,0.0,0.0,1.0) для остальных;

GL_SPECULAR параметр params должен содержать четыре целых или вещественных значения цветов RGBA, которые определяют цвет зеркального отражения. Значение по умолчанию: (1.0,1.0,1.0,1.0) для LIGHT0 и (0.0,0.0,0.0,1.0) для остальных;

GL_POSITION параметр params должен содержать четыре целых или вещественных, которые определяют положение источника света. Если значение компоненты w равно 0.0, то источник считается бесконечно удаленным и при расчете освещенности учитывается только направление на точку (x, y, z), в противном случае считается, что источник расположен в точке (x, y, z, w). Значение по умолчанию: (0.0,0.0,1.0,0.0);

GL_CONSTANT_ATTENUATION_EXPONENT параметр param должен содержать целое или вещественное число от 0 до 1.0, задающее постоянный коэффициент затухания - kc. Значение по умолчанию: 1.0;

GL_LINEAR_ATTENUATION_EXPONENT параметр param должен содержать целое или вещественное число от 0 до 1.0, задающее линейный коэффициент затухания – kl. Значение по умолчанию: 0.0;

GL_QUADRATIC_ATTENUATION_EXPONENT параметр param должен содержать целое или вещественное число от 0 до 1.0, задающее квадратичный коэффициент затухания – kq. Значение по умолчанию: 0.0.

При изменении положения источника света следует учитывать следующие факты: если положение задается командой glLight..() перед определением ориентации взгляда (командой glLookAt()), то будет считаться, что источник находится в точке наблюдения. Если положение устанавливается между заданием ориентации и преобразованиями видовой матрицы, то оно фиксируется и не зависит от видовых преобразований. В последнем случае, когда положение задано после ориентации и видовой матрицы, его положение можно менять, устанавливая как новую ориентацию наблюдателя, так и меняя видовую матрицу.

Для использования освещения сначала надо установить соответствующий режим вызовом команды glEnable(GL_LIGHTNING), а затем включить нужный источник командой glEnable(GL_LIGHTn).

Например, для создания источника, расположенного в точке (3,6,5) в мировых координатах, следует выполнить следующий код:

```
Glfloat myLightPosition[]={3.0,6.0,5.0,1.0};
GlLightfv(GL_LIGHT0,GL_POSITION,myLightPosition);
GlEnable(GL_LIGHTING);//ВКЛЮЧЕНИЕ СВЕТА
```

GlEnable(GL_LIGHT0);//ВКЛЮЧЕНИЕ КОНКРЕТНОГО ИСТОЧНИКА СВЕТА

Массив myLightPosition[] определяет положение источника света и передается в функцию glLightfv() вместе с именем GL_LIGHT0, для того чтобы связать его с конкретным источником, обозначенным именем GL_LIGHT0.

Некоторые источники света, такие как настольная лампа, находятся "внутри" сцены, в то время как другие, например солнце, бесконечно удалены от сцены. OpenGL позволяет создавать источники света обоих типов посредством задания положения источника в однородных координатах.

Тогда получается (x, y, z, 1) - локальный источник света в положении (x, y, z), и (x, y, z, 0) - вектор к бесконечно удаленному источнику света в направлении (x, y, z).

Можно также разложить источник света на различные цвета. OpenGL позволяет присвоить различный цвет каждому из трех типов света, испускаемого источником: фоновому, диффузному и зеркальному. Преимущество привязки фонового света к источнику заключается в том, что его можно включать и выключать во время работы приложения. С помощью приведенного ниже кода определяются массивы для хранения цветов, испускаемых источниками света. Эти массивы затем передаются в функцию glLightfv():

```
//определяем некоторые цвета
GLfloat amb0[]={0.2,0.4,0.6,1.0};
GLfloat diff0[]={0.8,0.9,0.5,1.0};
GLfloat spec0[]={1.0,0.8,1.0,1.0};
//привязываем их к LIGHT0
glLightfv(GL_LIGHT0,GL_AMBIENT,amb0);
glLightfv(GL_LIGHT0,GL_DIFFUSE,diff0);
glLightfv(GL_LIGHT0,GL_SPECULAR,spec0);
```

Цвета задаются в так называемом формате RGBA, что означает: Red - красный, Green - зеленый, Blue - синий и Alpha - альфа. Величина альфа иногда используется для смешения двух цветов на экране. Источники света имеют различные значения по умолчанию.

Для всех источников: фоновая компонента по умолчанию - (0,0,0,1), наименьшая яркость - черный. Для источника GL_LIGHT0: диффузная компонента по умолчанию - (1,1,1,1), наибольшая яркость – белый; зеркальная компонента по умолчанию - (1,1,1,1), наибольшая яркость - белый. Для других источников света значения диффузного и зеркального компонентов по умолчанию устанавливаются в черный цвет.

4.5.2. Прожекторы

Как упоминалось ранее, можно заставить позиционный источник света работать в качестве прожектора. Источники света по умолчанию являются точечными источниками. Это означает, что они излучают свет равномерно по всем направлениям. Однако OpenGL позволяет превратить их в прожекторы, так чтобы они излучали свет в ограниченном числе направлений. На рис. 4.3 показан прожектор, нацеленный в направлении d с углом пропускания ("cutoff angle") - a.

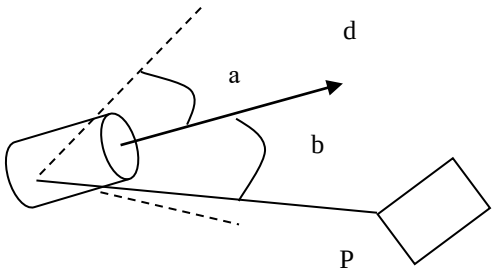

Рис. 4.3. Свойства прожектора

Для создания прожектора необходимо определить границы конуса света. В точках, лежащих вне конуса пропускания, свет не виден вообще. Для таких вершин, как P, которые лежат внутри этого конуса, количество света, достигающего точки P, пропорционально множителю cosE (b), где b - угол между вектором d и прямой, соединяющий источник с точкой P. Также необходимо задать направление света прожектора, которое определяет ось конуса света. По умолчанию это направление равно (0,0,-1), то есть свет испускается вдоль отрицательной оси z. В дополнение к направлению и углу прямого выхода излучения для прожектора существуют два способа управления интенсивностью распространения света внутри конуса. Во-первых, можно задать коэффициент затухания, который умножается на интенсивность света. Так же можно установить параметр GL_SPOT_EXPONENT для управления концентрацией света. Интенсивность имеет наивысшее значение в центре конуса. И она затухает по направлению к краям конуса – по косинуса угла, возведенному в степень точечной экспоненты. Показатель степени E выбирается пользователем так, чтобы обеспечить нужное уменьшение интенсивности света в зависимости от угла.

Параметры прожектора устанавливаются следующим образом: с помощью функции glLightf() задается его единственный параметр, а с помощью функции glLightfv() - вектор:

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,45.0);// угол пропускания - 45
glLightf(GL_LIGHT0,GL_SPOT_EXPONENT,4.0);// значение E
GLfloat dir[]={2.0,1.0,-4.0};// направление прожектора
glLightfv(Gl_LIGHT0,GL_SPOT_DIRECTION,dir);
```

По умолчанию установлены следующие значения этих параметров: d=(0,0,-1), a=180, E=0, что соответствует действующему во всех направлениях точечному источнику света.

4.5.3. Ослабление света с расстоянием

OpenGL также дает возможность задавать скорость ослабевания света при удалении от источника. Хотя было раньше допущено упрощение, что не учитывался этот фактор. В OpenGL предусмотрено ослабление силы позиционного источника света с помощью, так называемого коэффициента ослабления:

$$atten = \frac{1}{k_c + k_l D + k_q D^2},$$

где k_c, k_l, k_q - коэффициенты, а D - расстояние между точкой расположения источника света и рассматриваемой вершиной. Коэффициент ослабления отключен в случае источников направленного света, поскольку они находятся бесконечно далеко. Это выражение является достаточно гибким, чтобы позволить смоделировать любую комбинацию постоянной, линейной

и квадратичной зависимости от расстояния до источника. Параметры управляются с помощью вызовов функции

glLightf(GL_LIGHT0,GL_CONSTNT_ATTENUATION,2.0); и подобным же образом для GL_LINEAR_ATTENUATION и GL_QUADRATIC_ATTENUATION. Значения по умолчанию составляют kc=1, kl=0, kq=0 , что исключает какое-либо ослабление света.

4.5.4. Модель освещения в OpenGL

В OpenGL используется модель освещения Фонга, в соответствии, с которой цвет точки определяется несколькими факторами: свойствами материала и текстуры, величиной нормали в этой точке, а также положением источника света и наблюдателя. Для корректного расчета освещенности в точке надо использовать единичные нормали, однако, команды типа glScale..(), могут изменять длину нормалей. Чтобы это учитывать, используется уже упоминавшийся режим нормализации нормалей, который включается вызовом команды glEnable(GL_NORMALIZE).

OpenGL предусматривает задание трех параметров, определяющих общие законы применения модели освещения. Эти параметры передаются в функцию glLightModel() и некоторые ее модификации. Для задания глобальных параметров освещения используются команды:

void glLightModel[i f](GLenum pname, GLenum param)

void glLightModel[i f]v(GLenum pname, const GLtype *params)

Аргумент pname определяет, какой параметр модели освещения будет настраиваться и может принимать следующие значения:

GL_LIGHT_MODEL_LOCAL_VIEWER - является ли точка наблюдения локальной или удаленной. OpenGL вычисляет зеркальные отражения с помощью "промежуточного вектора" h=s+v, описанного ранее. Истинные направления s и v, различаются для каждой вершины сетки. Если источник света является направленным, то вектор s-величина постоянная, а v все же изменяется от вершины к вершине. Скорость визуализации возрастает, если сделать и вектор v постоянным для всех вершин. По умолчанию OpenGL использует значение v=(0,0,1), при этом вектор указывает в сторону положительной оси z в координатах камеры. В то же время можно принудительно заставить графический конвейер вычислять истинное значение вектора v для каждой вершины с помощью выполнения оператора:

GlLightModel(GL_LIGHT_MODEL_LOCAL_VIEWER, GL_TRUE);

Параметр param должен быть булевским и задает положение наблюдателя. Если он равен FALSE, то направление обзора считается параллельным оси z, вне зависимости от положения в видовых координатах. Если же он равен TRUE, то наблюдатель находится в начале видовой системы координат. Это может улучшить качество освещения, но усложняет его расчет. Значение по умолчанию: FALSE.;

GL_LIGHT_MODEL_TWO_SIDE - правильно ли происходит закрашивание обеих сто-рон полигона. Параметр param должен быть булевским и управляет режимом расчета освещенности, как для лицевых, так и для обратных граней. Если он равен FALSE, то освещенность рассчитывается только для лицевых граней. Если же он равен TRUE, расчет проводится и для обратных граней. Значение по умолчанию: FALSE. Каждая полигональная грань модели имеет две стороны. При моделировании можно рассматривать внутреннюю сторону и внешнюю. Принято заносить эти вершины в список против часовой стрелки, если смотреть с

внешней стороны объекта. Большинство каркасных объектов представляют сплошные тела, ограничивающие некоторое пространство, так что четко определены понятия внешней и внутренней стороны. Для таких объектов камера может наблюдать только внешнюю поверхность каждой грани (если конечно камера не находится внутри объекта). При правильном удалении невидимых поверхностей внутренняя поверхность каждой грани скрыта от глаза какой-нибудь более близкой гранью.

В OpenGL нет понятия "внутри" и "снаружи", он может различать только "лицевые грани" и "нелицевые грани". Грань является лицевой (front face), если ее вершины расположены в списке против часовой стрелки, в том порядке, каком их видит глаз. Можно заменить этот порядок на обратный с помощью функции glFrontFace(GL_CW),в которой обусловлено, что грань является лицевой только в том случае, если ее вершины занесены в список в порядке по часовой стрелке. Для объекта, ограничивающего некоторое пространство, все грани, которые видит глаз, являются лицевыми, и OpenGL правильно рисует и закрашивает их. Нелицевые грани также рисуются в OpenGL, но, в конце концов, они скрываются за более близкими лицевыми гранями. Можно ускорить работу процессора, если запретить OpenGL визуализацию нелицевых граней. При этом используется следующий код:

glCullFace(GL_BACK);

glEnable(GL_CULL_FACE);.

Для правильного закрашивания нелицевых граней нужно проинструктировать OpenGL с помощью оператора glLightModel (GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE). При выполнении этой команды OpenGL изменяет направления нормалей всех нелицевых граней таким образом, чтобы они указывали на наблюдателя, после чего закрашивание осуществляется корректно. Замена величины GL_TRUE на GL_FALSE отключает эту опцию. Грани, нарисованные с помощью OpenGL, не отбрасывает теней, поэтому все нелицевые грани получают от источника такой же свет, даже если между ними и источником света находится какая-нибудь другая грань;

GL_LIGHT_MODEL_AMBIENT - цвет глобального фонового света. Параметр params должен содержать четыре целых или вещественных числа, которые определяют цвет фоно-вого освещения даже в случае отсутствия определенных источников света. Для любой задан-ной сцены можно установить глобальный фоновый свет, не зависимый ни от какого опреде-ленного источника. Для создания такого освещения следует задать его цвет с помощью следующих команд:

GLfloat amb[] = {0.2, 0.3, 0.1, 1.0};

glLightModelfv(GL_LIGHT_MODEL_AMBIENT, amb);

Этот код придает источнику фонового света цвет (0.2, 0.3, 0.1). Значение по умолчанию составляет (0.2, 0.2, 0.2, 0.1), так что фоновый свет присутствует всегда, если только умышленно не изменить его. Задание фоновому источнику ненулевого значения обеспечивает видимость объектов сцены, даже если вы не активизировали ни одной функции освещения;

GL_LIGHT_MODEL_COLOR_CONTROL отделение зеркальной составляющей цвета. Для обычных расчетов освещенности фоновая, диффузная, зеркальная и эмиссионная составляющие вычисляются и просто складываются друг с другом. По умолчанию отображение текстур применяется после освещения, так что зеркальные блики могут появиться приглушенными, или текстурирование будет выглядеть по-другому. При следующем вызове –

glLightModel(GL_LIGHT_MODEL_COLOR_CONTROL,GL_ SEPARATE_SPECULAR_COLOR) OpenGL отделяет вычисление зеркального цвета из приложения. После этого освещение генерирует два цвета для каждой вершины: первоначальный цвет, состоящий из неотраженных составляющих освещенности, и второй цвет, являющийся суммой зеркальных составляющих освещенности. При отображении текстур только первый цвет комбинируется с цветами текстуры. После выполнения операции текстурирования второй цвет добавляется к итоговой комбинации первого и текстурного компонентов цвета. Объекты, для которых выполнено освещение и текстурирование с отделением зеркального цвета, обычно более видимы и имеют более заметные зеркальные блики. Для возвращения к установкам по умолчанию необходимо сделать вызов glLightModeli(GL_LIGHT_MODEL_COLOR_CONTROL,GL_SINGLE_ COLOR). После этого опять первоначальный цвет будет состоять из всех составляющих цвета: рассеянной, диффузной, эмиссионной и зеркальной. Составляющие освещения не прибавляются после текстурирования.

4.5.5. Перемещение источников света

OpenGL обрабатывает положение и направление источника света так же, как он обрабатывает положение геометрического примитива. Другими словами, источник света подвергается тем же матричным преобразованиям, что и примитив. Более точно, при вызове команды glLight..() для задания позиции или направления источника света, эти позиции или направление преобразуются с помощью текущей матрицы видового преобразования и хранятся в видовой системе координат. Это означает, что можно управлять положением источника света или его направлением, изменяя содержимое матрицы видового преобразования.

Следовательно, источники света могут быть перемещены с помощью соответствующих вызовов функций преобразований. Функции преобразования - glRotated(), glTranslated().

Массив position, заданный подпрограммой glLightfv(GL_LIGHT0, GL_POSITION, position), изменяется с помощью матрицы моделирования-вида, что происходит в момент вызова функции glLightfv().

Поэтому для изменения положения источника света с помощью преобразований и для независимого перемещения камеры, команду позиционирования источника нужно вставить внутрь стековых скобок "push - pop", как это сделано в следующем коде:

```
//функция вывода изображения
void display()
{
GLfloat position[]={2,1,3,1};//начальное положение источника света
glMatrixMode(GL_MODELVIEW);//очистка буферов глубины и цвета
glLoadIdentity();
glPushMatrix();//сохранение текущей матрицы вида
glRotated(…);//поворот
glTranslated(…);//смещение
glLightfv(GL_LIGTH0,Gl_POSITION,position);
glPopMatrix();//восстановление матрицы вида
gluLookAt(…);//положение камеры
glutSwapBuffers();//рисование объекта
}
```

С другой стороны, для того чтобы источник света перемещался вместе с камерой, необходимо использовать следующий код:

```
GLfloat pos[]={0,0,0,1};
GlMatrixMode(GL_MODELVIEW);
GlLoadIdentity();
GlLightfv(GL_LIGHT0,GL_POSITION,position);// положение источника света
GluLookAt(…);//перемещение камеры и источника света
```

Этот код устанавливает источник света прямо в глаз, и тогда свет перемещается вместе с камерой.

4.6. Работа со свойствами материалов в OpenGL

Влияние источника света можно увидеть только при отражении света от поверхности объекта. В OpenGL предусмотрены возможности задания различных коэффициентов отражения, фигурирующих в уравнении интенсивности отраженного света. Эти коэффициенты устанавливаются с помощью различных версий функции glMaterial(), причем коэффициенты можно устанавливать отдельно для лицевых и нелицевых граней.

Для задания параметров текущего материала используются команды:

```
void glMaterial[i f](GLenum face, GLenum pname, GLtype param);
void glMaterial[i f]v(GLenum face, GLenum pname, GLtype *params);
```

С их помощью можно определить рассеянный, диффузный и зеркальный цвета материала, а также цвет, степень зеркального отражения и интенсивность излучения света, если объект должен светиться. Какой именно параметр будет определяться значением param, зависит от значения pname:

GL_AMBIENT параметр params должен содержать четыре целых или вещественных значения цветов RGBA, которые определяют рассеянный цвет материала (цвет материала в тени). Значение по умолчанию: (0.2,0.2,0.2,1.0);

GL_DIFFUSE параметр params должен содержать четыре целых или вещественных значения цветов RGBA, которые определяют цвет диффузного отражения материала. Значение по умолчанию: (0.8,0.8,0.8,1.0);

GL_SPECULAR параметр params должен содержать четыре целых или вещественных значения цветов RGBA, которые определяют цвет зеркального отражения материала. Значение по умолчанию: (0.0,0.0,0.0,1.0);

GL_SHININESS параметр params должен содержать одно целое или вещественное значение в диапазоне от 0 до 128, которое определяет степень зеркального отражения материала. Значение по умолчанию: 0;

GL_EMISSION параметр params должен содержать четыре целых или вещественных значения цветов RGBA, которые определяют интенсивность излучаемого света материала. Здесь устанавливается эмиссионный цвет для грани. В дополнение к рассеянному, диффузному и зеркальному компоненту цвета материалы могут иметь эмиссионный цвет, который имитирует свет, исходящий от объекта. В модели освещения OpenGL, эмиссионный цвет поверхности добавляет объекту яркость, но на него не влияют никакие источники света. Значение по умолчанию: (0.0,0.0,0.0,1.0);

GL_AMBIENT_AND_DIFFUSE эквивалентно двум вызовам команды glMaterial..() со значением pname GL_AMBIENT и GL_DIFFUSE и одинаковыми значениями params.

Задаваемые коэффициенты фонового и диффузного отражения устанавливаются равным одному и тому же значению. Такая установка сделана для удобства, поскольку коэффициенты фонового и диффузного отражения часто выбираются одинаковыми.

Из этого следует, что вызов команды glMaterial[i f]() возможен только для установки степени зеркального отражения материала. В большинстве моделей учитывается диффузный и зеркальный отраженный свет; первый определяет естественный цвет объекта, а второй – размер и форму бликов на его поверхности. Параметр face определяет тип граней, для которых задается этот материал и может принимать следующие значения:

GL_FRONT - задается коэффициент отражения для лицевых граней;

GL_BACK - задается коэффициент отражения для нелицевых граней;

GL_FRONT_AND_BACK - задается коэффициент отражения и для лицевых граней, и для нелицевых граней.

Если в сцене материалы объектов различаются лишь одним параметром, рекомендуется сначала установить нужный режим, вызвав glEnable() с параметром GL_COLOR_MATERIAL, а затем использовать команду:

void glColorMaterial(GLenum face, GLenum pname);

где параметр face имеет аналогичный смысл, а параметр pname может принимать все перечисленные значения. После этого, значения, выбранного с помощью pname, свойства материала для конкретного объекта (или вершины) устанавливается вызовом команды glColor(), что позволяет избежать вызовов более ресурсоемкой команды glMaterial() и повышает эффективность программы. Например, следующий код:

GLfloat myDiffuse[]= {0.8, 0.2, 0.0, 1.0};

glMaterialfv(GL_FRONT, GL_DIFFUSE, myDiffuse);

присваивает коэффициенту диффузного отражения (pdr,pdg,pdb) значение (0.8,0.2,0,0) для всех последовательно заданных лицевых граней. Коэффициенты отражения задаются в формате RGBA в виде четверки, аналогично заданию цвета.

4.7. Цель, требования и рекомендации к выполнению задания

Разработать программу, реализующую представление разработанной вами трехмерной сцены с добавлением возможности формирования различного типа проекций, используя предложенные функции OpenGL и возможности использования различных видов источников света, используя предложенные функции OpenGL (матрицы видового преобразования, проецирование, модель освещения, типы источников света, свойства материалов).

Разработанная программа должна быть пополнена возможностями остановки интерактивно различных атрибутов через вызов соответствующих элементов интерфейса пользователя (замена типа источника света, управление положением камеры, изменение свойств материала модели, как с помощью мыши, так и с помощью диалоговых элементов)

Задания

Разработать программу, реализующую представление трехмерной сцены с добавлением возможности формирования различного типа проекций, отражений, используя предложенные функции OpenGL(модель освещения, типы источников света, свойства материалов).

Разработанная программа должна быть пополнена возможностями остановки интерактивно различных атрибутов через вызов соответствующих элементов интерфейса пользователя (замена типа источника света, управление положением камеры, изменение свойств материала модели, как с помощью мыши, так и с помощью диалоговых элементов)