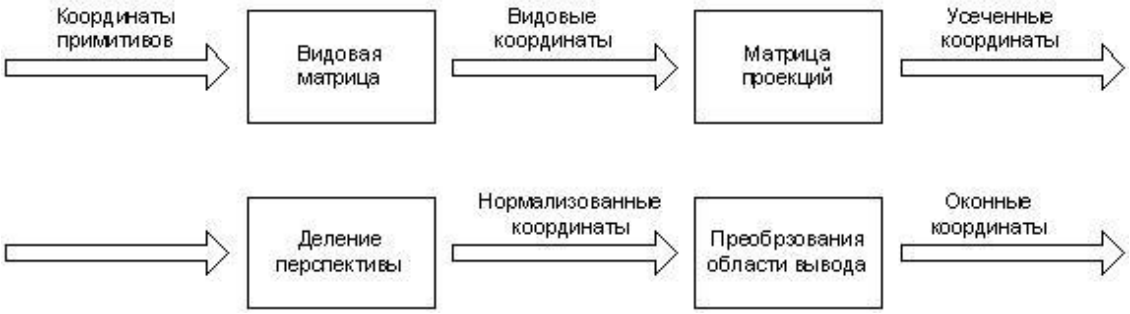
Лабораторная работа №6 «Проекции и источники света».

Для построения проекции трехмерного изображения библиотека выполняет ряд преобразований, последовательность которых показана на рисунке:



Для выполнения координатных преобразований в библиотеке используется три матрицы: видовая *матрица*, *матрица проекций* и *матрица* текстур.

Значения матриц хранятся внутри библиотеки. OpenGL предоставляет набор команд для операций над ними.

Обычно видовая матрица используется для таких преобразований как перенос, масштабирование и поворот. Для выполнения этих операций библиотека реализует команды Translate, Scale, Rotate(их мы рассматривали в Лабораторной работе №4).

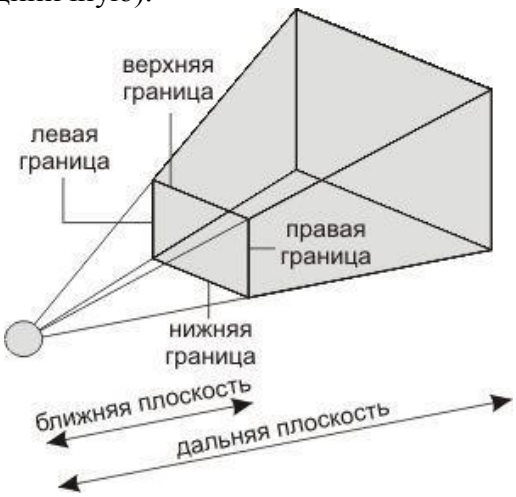
На этапе перспективных преобразований видовые координаты вершин (полученные на предыдущем этапе вычислений) умножаются на матрицу проекций P.

$$\begin{pmatrix} x_c \\ y_c \\ z_c \\ w_c \end{pmatrix} = P \begin{pmatrix} x_e \\ y_e \\ z_e \\ w_e \end{pmatrix}$$

В библиотеке реализованы две команды для формирования наиболее часто используемых проекций: команда Ortho - для формирования матрицы параллельной проекции и команда Frushtum – для получения матрицы центральной перспективной проекции.

При выполнении этих команд определяется область видимости трехмерной сцены. Матрица проекции, кроме выполнения проекционных преобразований, также выполняет масштабирование координат видимой области к интервалу [-1,1] для каждой из осей координат. Преобразованные таким образом координаты вершин называются усечёнными координатами.

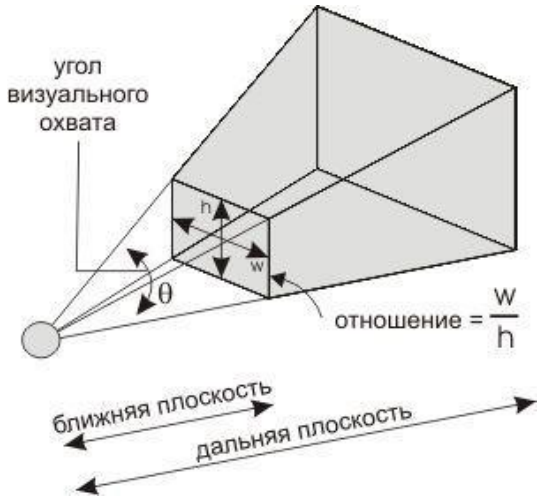
Команда определения объема видимости в форме усеченной пирамиды **glFrustum()** вычисляет матрицу, выполняющую перспективное проецирование, и умножает на нее текущую матрицу проекции (обычно единичную).



```
void glFrustum (GLdouble left, GLdouble right, GLdouble bottom, GLdouble top,
GLdouble near, GLdouble far);
```

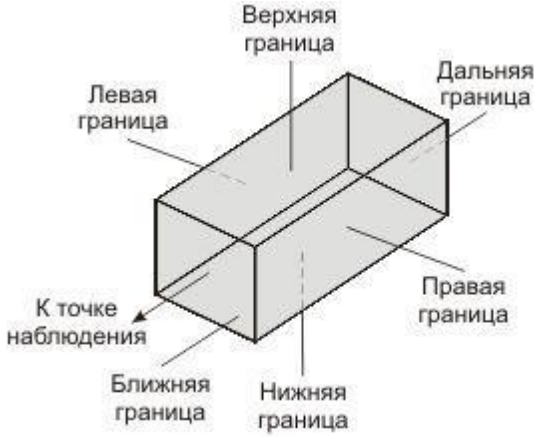
Создает матрицу перспективного проецирования и умножает на нее текущую матрицу. Объем видимости задается параметрами (*left*, *bottom*,-*near*) и (*right*, *top*,-*near*) определяющими координаты (*x*, *y*, *z*) левого нижнего и правого верхнего углов ближней отсекающей плоскости; *near*и *far* задают дистанцию от точки наблюдения до ближней и дальней отсекающих плоскостей (они всегда должны быть положительными).

Вместо нее вы можете попробовать использовать функцию **gluPerspective()** из библиотеки утилит. Эта функция создает объем видимости той же формы, что и **glFrustum()**, но вы задаете его параметры иным путем. Вместо указания углов ближней отсекающей плоскости, вы задаете угол визуального охвата (θ или тета) в вертикальном направлении *y* и отношение ширины к высоте (*x/y*). (Для квадратной части экрана отношение ширины к высоте равно *1.0*.) Этих двух параметров достаточно для определения неусеченной пирамиды вдоль направления обзора. Вы также задаете дистанцию между точкой наблюдения и ближней и дальней отсекающими плоскостями, таким образом, отсекая пирамиду. Заметьте, что **gluPerspective()** ограничена созданием только пирамид симметричных вдоль линии обзора по *x*- и *y*-осям, но обычно это именно то, что и требуется.




```
void gluPerspective (GLdouble fovy, GLdouble aspect, GLdouble near, GLdouble far);
```

При использовании ортографической проекции объем видимости представляет собой прямоугольный параллелепипед или коробку. В отличие от перспективной проекции размер объема видимости не изменяется от одного конца к другому, таким образом, дальность от камеры не влияет на размер объектов в результирующем изображении. Этот тип проекции используется для таких приложений, как системы автоматизированного проектирования, где важны реальные размеры объектов относительно друг друга и точность отображения углов между ними.



Команда **glOrtho()** создает параллельный объем видимости ортографической проекции. Как и в случае **glFrustum()** вы задаете углы ближней отсекающей плоскости и расстояния до ближней и дальней плоскостей.

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

Создает матрицу для параллельного объема видимости ортографической проекции и умножает на нее текущую матрицу. (*left, bottom, -near*) и (*right, top, -near*) задают точки на ближней плоскости отсечения, которые будут спроецированы соответственно на нижний левый и верхний правый углы порта просмотра. (*left, bottom, -far*) и (*right, top, -far*) это точки на дальней плоскости отсечения, которые будут спроецированы на те же углы порта просмотра. Значения для *far* и *near* могут быть положительными, отрицательными или даже нулевыми, однако они не должны быть равны между собой. Если никакие другие преобразования не используются, направление проецирования совпадает с осью *z*, а направление обзора – с ее отрицательным направлением.

На третьем этапе выполняется преобразование нормализованных координат вершин к декартовым координатам, путем деления каждой из первых трех координат на четвёртую:

$$\begin{pmatrix} x_d \\ y_d \\ z_d \end{pmatrix} = \begin{pmatrix} x_c/w_c \\ y_c/w_c \\ z_c/w_c \end{pmatrix}$$

В терминах OpenGL это преобразование носит название "деление перспективы". Результатом выполнения этого преобразования является получение декартовых координат вершин.

OpenGL реализует несколько матриц, содержание которых хранится в самой библиотеке. Получить доступ в каждый конкретный момент времени можно только к одной из этих матриц. Матрица, с которой будут выполняться те или иные преобразования, должна быть предварительно выбрана в качестве текущей с помощью команды MatrixMode. Выбранная матрица остается текущей до следующего вызова команды MatrixMode.

Параметр mode определяет выбираемую для дальнейших преобразований матрицу и может принимать значения, перечисленные в таблице:

Назначение параметра	Значение параметра
Видовая матрица	GL_MODELVIEW
Матрица проекции	GL_PROJECTION
Матрица текстуры	GL_TEXTURE

После инициализации библиотеки текущей является видовая матрица.

Для создания реалистичных изображений необходимо определить как свойства самого объекта, так и свойства среды, в которой он находится. Первая группа свойств включает в себя параметры материала, 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, а также модель освещения. Все эти свойства можно задавать, вызывая соответствующие команды OpenGL.

Модель освещения

В OpenGL используется модель освещения, 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, используйте режим нормализации векторов нормалей, который включается вызовом команды glEnable(GL_NORMALIZE).

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

GL_LIGHT_MODEL_TWO_SIDE параметр *param* должен быть булевым и управляет режимом расчета освещенности, как для лицевых, так и для обратных граней. Если он равен **GL_FALSE**, то освещенность рассчитывается только для лицевых граней. Если же он равен **GL_TRUE**, расчет проводится и для обратных граней. Значение по умолчанию: **GL_FALSE.**

GL_LIGHT_MODEL_AMBIENT параметр *params* должен содержать четыре целых или вещественных числа, которые определяют цвет фонового освещения даже в случае отсутствия определенных источников света. Значение по умолчанию: (0.2, 0.2, 0.2,1.0).

Спецификация материалов

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 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 диффузный цвет 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 зеркальный цвет 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. 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.*

Из этого следует, что вызов команды glMaterial[i f]() возможен только для установки степени зеркального отражения материала (shininess). Команда `glMaterial[i f]v()` используется для задания остальных параметров.

Параметр *face* определяет тип граней, для которых задается этот материал и может принимать значения **GL_FRONT**, **GL_BACK** или **GL_FRONT_AND_BACK.**

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) устанавливаются вызовом команды `glColor*()`, что позволяет избежать вызовов более ресурсоемкой команды glMaterial*() и повышает эффективность программы.

Пример определения свойств материала:

```
float mat_dif[]={0.8,0.8,0.8};
float mat_amb[] = {0.2, 0.2, 0.2};
float mat_spec[] = {0.6, 0.6, 0.6};
float shininess = 0.7 * 128;
…
glMaterialfv (GL_FRONT_AND_BACK,GL_AMBIENT, mat_amb);
glMaterialfv (GL_FRONT_AND_BACK,GL_DIFFUSE, mat_dif);
glMaterialfv (GL_FRONT_AND_BACK,GL_SPECULAR, mat_spec);
glMaterialf  (GL_FRONT,GL_SHININESS, shininess);
```

Описание источников света

Определение свойств материала объекта имеет смысл, только если в сцене есть источники света. Иначе все объекты будут черными (или, строго говоря, иметь цвет, равный рассеянному цвету материала, умноженному на интенсивность глобального фонового освещения, см. команду glLightModel). 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 света. Он выбирается из набора специальных символических имен вида **GL_LIGHTi** , где i должно лежать в диапазоне от 0 до константы **GL_MAX_LIGHT**, которая обычно не превосходит восьми.

Параметры *pname* и *params* имеют смысл, аналогичный команде `glMaterial*()`. Рассмотрим значения параметра *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. 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. Значение по умолчанию: 180 (рассеянный свет).

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 **GL_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 **GL_LIGHT0** и (0.0, 0.0, 0.0, 1.0) для остальных.

GL_POSITION параметр *params* должен содержать четыре целых или вещественных числа, 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). В первом случае ослабления света при удалении от источника не происходит, т.е. источник считается бесконечно удаленным. Значение по умолчанию: (0.0, 0.0, 1.0, 0.0).

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). Эта характеристика источника имеет смысл, если значение **GL_SPOT_CUTOFF** отлично от 180 (которое, кстати, задано по умолчанию).

GL_CONSTANT_ATTENUATION,
GL_LINEAR_ATTENUATION,
GL_QUADRATIC_ATTENUATION параметр *params* задает значение одного из трех коэффициентов, определяющих ослабление интенсивности света при удалении от источника. Допускаются только неотрицательные значения. Если источник не является направленным (см. **GL_POSITION**), то ослабление обратно пропорционально сумме:

$\mathbf{att}_{constant} + att_{linear} * d + att_{quadratic} * d^2$,

где d – расстояние между источником света и освещаемой им вершиной, $\mathbf{att}_{constant}$, att_{linear} и $at\text{-}t_{quadratic}$ равны параметрам, заданным с помощью констант **GL_CONSTANT_ATTENUATION**, **GL_LINEAR_ATTENUATION** и **GL_QUADRATIC_ATTENUATION** соответственно. По умолчанию эти параметры задаются тройкой (1, 0, 0), и фактически ослабления не происходит.

При изменении положения источника света следует учитывать следующий факт: в OpenGL источники света являются объектами, во многом такими же, как многоугольники и точки. На них распространяется основное правило обработки координат в OpenGL – параметры, описывающее положение в пространстве, преобразуются текущей модельно-видовой матрицей в момент формирования объекта, т.е. в момент вызова соответствующих команд OpenGL. Таким образом, формируя источник света одновременно с объектом сцены или камерой, его можно привязать к этому объекту. Или, наоборот, сформировать стационарный источник света, который будет оставаться на месте, пока другие объекты перемещаются.

Общее правило такое:

Если положение источника света задается командой `glLight*()` перед определением положения виртуальной камеры (например, командой `glLookAt()`), то будет считаться, что

координаты (0,0,0) источника находится в точке наблюдения и, следовательно, положение источника света определяется относительно положения наблюдателя.

Если положение устанавливается между определением положения камеры и преобразованиями модельно-видовой матрицы объекта, то оно фиксируется, т.е. в этом случае положение источника света задается в мировых координатах.

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

Еще раз обратим внимание на то, что при выключенном освещении цвет вершины равен текущему цвету, который задается командами `glColor*()`. При включенном освещении цвет вершины вычисляется исходя из информации о материале, нормалях и источниках света.

При выключении освещения визуализация происходит быстрее, однако в таком случае приложение должно само рассчитывать цвета вершин.

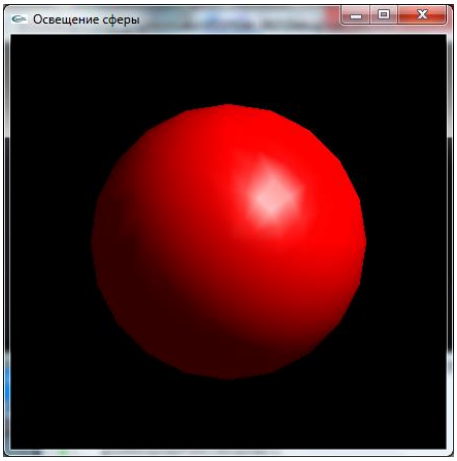
Задание 1:

1. Отладить программу, приведенную в листинге:

```
void init(void)
{
   GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
   GLfloat mat_shininess[] = { 50.0 };
   GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };

   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);

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

void display(void)
{
   glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glColor3f(1.0,0.0,0.0,1.0);
   glutSolidSphere (1.0, 20, 16);
   glFlush ();
}

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

void keyboard(unsigned char key, int x, int y)
{
```

```
    switch (key) {
       case 27:
          exit(0);
          break;
    }
}

void main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize (500, 500);
    glutInitWindowPosition (100, 100);
    glutCreateWindow ("Освещение сферы");
    init ();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMainLoop();
 }
```

Результат выполнения программы



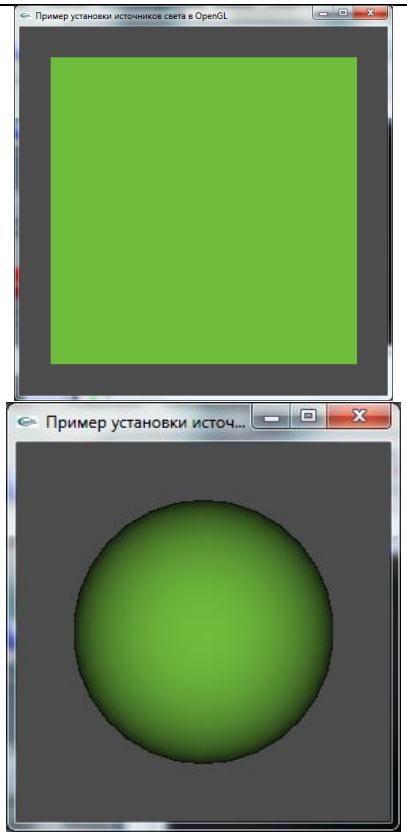
2. Отключить все источники света, кроме глобального фонового освещения.
3. Добавить точечный источник света.
4. Превратить точечный источник света в прожектор.
5. Изменить свойства материала в соответствии с вариантом.

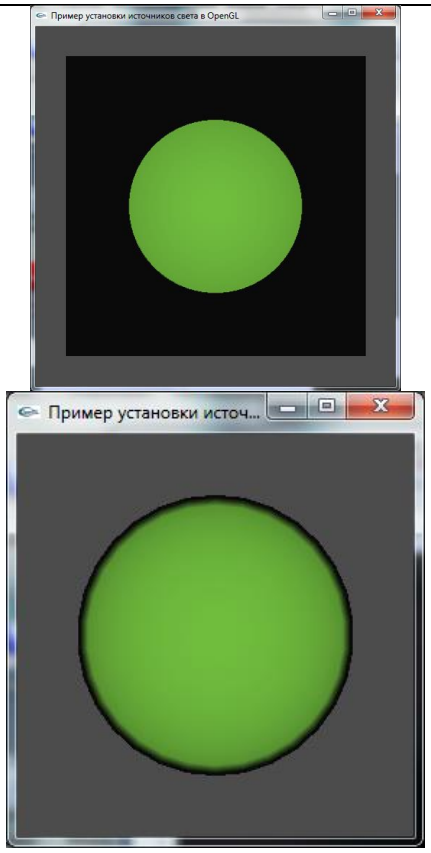
Вари-ант	Диффузионная составляющая (GL_DIFFUSE)	Зеркальная составляющая (GL_SPECULAR)	Фоновая составляющая (GL_AMBIENT)	Коэффициент резкости бликов (GL_SHININESS)	Эмиссионные свойства (GL_EMISSION)
1	0.7, 0.7, 0.7, 1.0	0.0, 0.0, 0.0, 1.0	0.0, 0.0, 0.0, 1.0	0.0	0.0, 0.0, 0.0, 1.0
2	0.1, 0.5, 0.8, 1.0	1.0, 1.0, 1.0, 1.0	0.0, 0.0, 0.0, 1.0	100.0	0.0, 0.0, 0.0, 1.0
3	0.1, 0.5, 0.8, 1.	1.0, 1.0, 1.0, 1.0	0.0, 0.0, 0.0, 1.0	5.0	0.0, 0.0, 0.0, 1.0
4	0.1, 0.5, 0.8, 1.0	0.0, 0.0, 0.0, 1.0	0.0, 0.0, 0.0, 1.0	0.0	0.3, 0.2, 0.2, 0.0
5	0.1, 0.5, 0.8, 1.0	0.0, 0.0, 0.0, 1.0	0.7, 0.7, 0.7, 1.0	0.0	0.0, 0.0, 0.0, 1.0
6	0.1, 0.5, 0.8, 1.0	1.0, 1.0, 1.0, 1.0	0.7, 0.7, 0.7, 1.0	5.0	0.0, 0.0, 0.0, 1.0
7	0.1, 0.5, 0.8, 1.0	1.0, 1.0, 1.0, 1.0	0.7, 0.7, 0.7, 1.0	100.0	0.0, 0.0, 0.0, 1.0
8	0.1, 0.5, 0.8, 1.0	0.0, 0.0, 0.0, 1.0	0.7, 0.7, 0.7, 1.0	0.0	0.3, 0.2, 0.2, 0.0
9	0.1, 0.5, 0.8, 1.0	0.0, 0.0, 0.0, 1.0	0.8, 0.8, 0.2, 1.0	0.0	0.3, 0.2, 0.2, 0.0
10	0.1, 0.5, 0.8, 1.0	1.0, 1.0, 1.0, 1.0	0.8, 0.8, 0.2, 1.0	5.0	0.0, 0.0, 0.0, 1.0

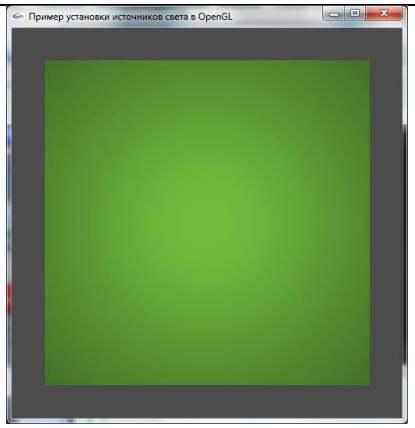
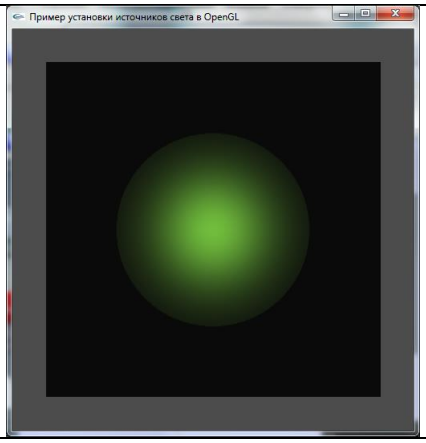
Задание 2:

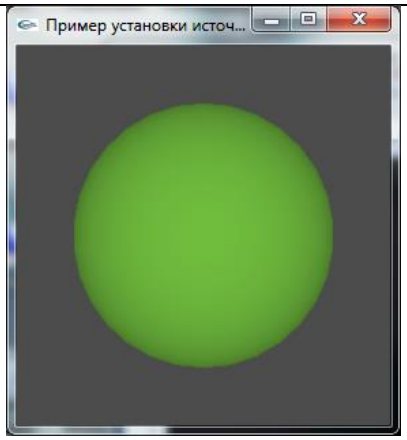
Написать программу, которая изменяет тип освещения при нажатии на клавиши ‘1’ – ‘6’. В качестве объектов сцены использовать плоскость и сферу.

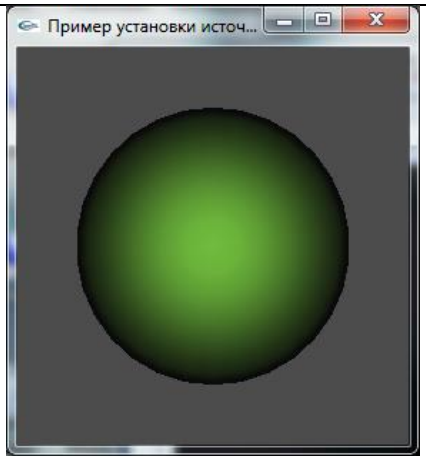
Примерный результат выполнения программы

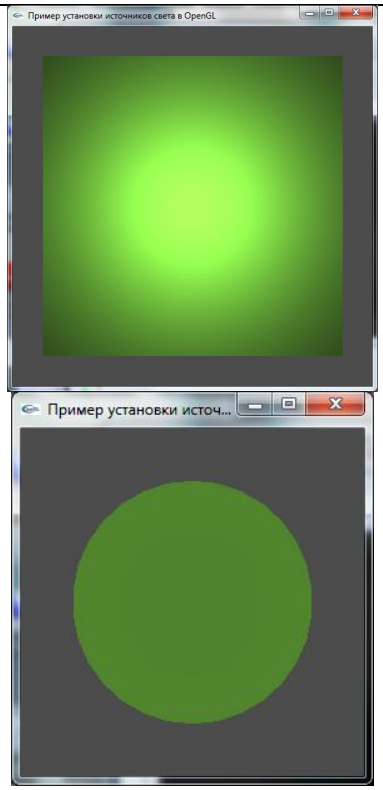
<table>
<tr>
<td> 

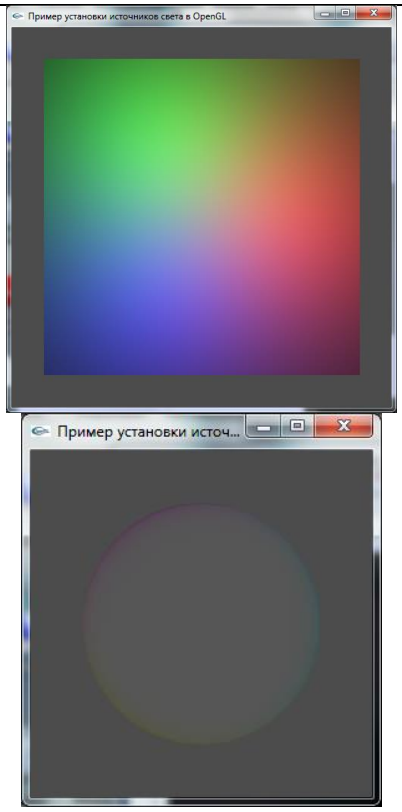
‘1’ - Направленный источник света</td>
<td> 

‘4’ – Прожектор убывание интенсивности с расстоянием отключено половина угла при вершине 30 градусов направление на центр плоскости</td>
</tr>
<tr>
<td> 
</td>
<td> 
</td>
</tr>
</table>

<table>
<tr>
<td> 

‘2’ - Точечный источник света, убывание интенсивности с расстоянием выключено</td>
<td> 

‘5’ - Прожектор, половина угла при вершине 30 градусов направление на центр плоскости включен расчет убывания интенсивности</td>
</tr>
<tr>
<td> 

‘3’ - Точечный источник света, убывание интенсивности с расстоянием задано функцией f(d) = 1.0 / (0.4 * d * d + 0.2 * d)</td>
<td> 

‘6’ - Несколько источников света</td>
</tr>
</table>

Задание 3:

Используя функции библиотек OpenGL, доработать программу, реализующую задание по лабораторной работе №5 (часть 2) для отображения трёхмерной сцены с учётом нескольких источников освещения (точечных, бесконечно удалённых, светящихся поверхностей).