Методические указания к лабораторной работе №3 «Реализация проекций в OpenGL» по курсу «Компьютерная графика» для специальностей:

Цель лабораторной работы: Понять принципы формирования плоских геометрических проекций в OpenGL

Лабораторная работа основывается на лекциях: «Плоские геометрические проекции» и «Введение в OpenGL».

Контрольные вопросы по лекционному курсу:

Назовите ключевые отличия проекций, представленных на рис.1.

Что такое «видимый объем»? Какой командой он задается в OpenGL.

Какой командой в OpenGL задется параллельная проекция? Поясните смысл передаваетмых в команду параметров. Укажите положения центра проекции и проекционной плоскости.

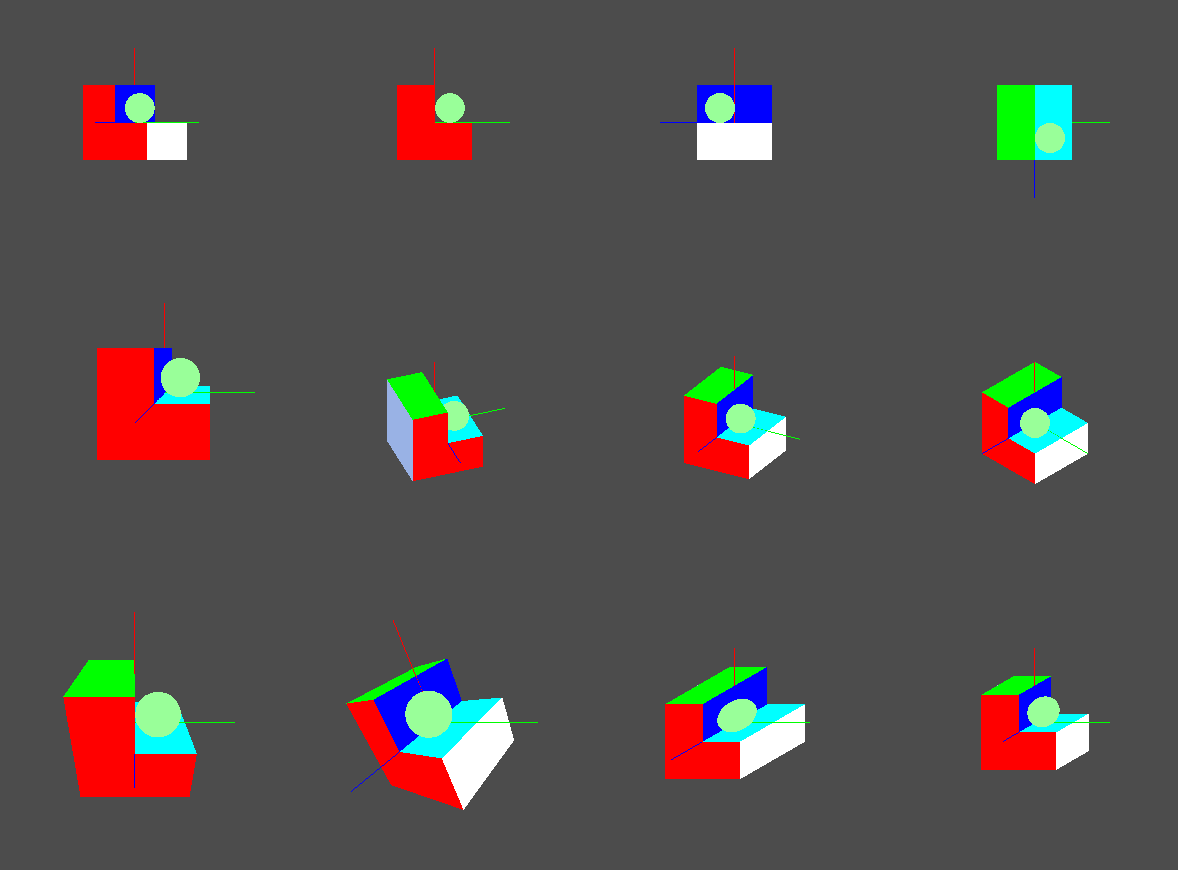
Какой командой в OpenGL задется центральная проекция? Поясните смысл передаваетмых в команду параметров. Укажите положения центра проекции и проекционной плоскости.

Что такое укорачивание? Чем оно характеризуется? Назовите отличия перспективного укорачивания от простого.



Задание на лабораторную работу:

Создать, используя OpenGL, программу, воспроизводящую все одиннадцать видов проекций, представленных на рисунке 1. Двенадцатый вид, расположенный в левом верхнем углу дает возможность рассмотреть сцену со всех сторон за счет поворота вокруг осей X и Y мировой системы координат.



Замечания по реализации программы.

Процедура main аналогична, использованной в предыдущей работе. Только создаваемое окно должно иметь соотношение высоты и ширины – 3:4. Такое окно удобно разделять на 12 квадратных областей.

Из процедуры инициализации (Initialize) исключите определение проекции. Проекции теперь будут определяться для каждого вида отдельно.

Процедура формирования сцены (scene) должна включать в себя создание куба с вырезанной четвертью и создание сферы.

Формирование всех проекций можно выполнить в процедуре Display.

Последовательность действий формирующая различные проекции будет следующая:

- задать поле вывода;

- сделать текущим стек матриц проецирования;

- сделать текущую матрицу (находящуюся на вершине стека) единичной;

- задать требуемую проекцию;

- сделать текущим стек модельно-видовых матриц;

- сделать текущую матрицу (находящуюся на вершине стека) единичной;

- задать дополнительные преобразования необходимые для создания требуемого вида проекции

- передать на конвейер описание сцены;

Например, для формирования ортографической проекции «вид сверху» требуется к простому ортографическому проецированию на плоскость Z=0, задаваемому командой Ortho, добавить преобразование поворота на угол 90 градусов относительно вектора совпадающего с осью X мировой системы координат. Соответствующий код может выглядеть следующим образом:

glViewport(VP \* 3, VP \* 2, VP, VP);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(-2.0, 2.0, -2.0, 2.0, -2.0, 2.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glRotatef(90, 1, 0, 0);

scene();

Наиболее сложным может оказаться формирование косоугольных проекций, так как для их формирования требуется преобразование скоса, не реализованное в OpenGL. Создать матрицу скоса можно и явным образом, объявив массив 4\*4 и заполнив его константами. После этого необходимо загрузить массив в модельно-видовую матрицу.

Существует более простой способ формирования матриц:

- объявить массив;

- сделать текущим стек модельно-видовых матриц;

- сделать текущую матрицу (находящуюся на вершине стека) единичной;

- загрузить текущую модельно-видовую матрицу в массив;

- внести в массив тербуемые изменения;

- загрузить массив в модельно-видовую матрицу.

Фрагметр кода формирования матрицы:

GLfloat a[16];

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

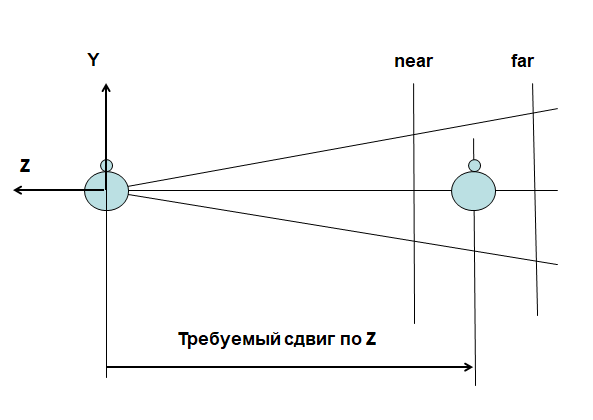
glGetFloatv(GL\_MODELVIEW\_MATRIX, a);

a[8] = ..

a[9] = ..

glLoadMatrixf(a);

Формирование центральных проекций так же имеет свои особенности. Прежде всего, они связаны с положение центра проекции. Как Вы должны понить из лекционного курса при реализации одноточечной центральной проекции средствами OpenGL (glFristum) центр проекции будет находится в начале мировой системы координат. Ближняя и дальняя секущие плоскости видимого объема должны находится дальше от наблюдателя, чем центр проекции. Напомним, что значения near и far не могут быть отрицательными (они задаются в видовых координатах). Создаваемые нами модели часто формируются вокруг начала мировой системы координат, следовательно, при центральном проецировании их надо сдвигать в сторону отрицательной полуоси Z мировой системы координат, так что бы они оказались внутри видимого объема

.

Другая особенность создания одноточечной центральной проекции связана с тем, что при задании видимого объема симметричного относительно оси Z, точка схода линий параллельных этой оси будет не видна! Внешене, такая проекция будет неотличима от фронтальной проекции. Что бы избежать данной «неприятности» надо выполнить сдвиг ь видимого объема и модели по осям X и Y.

Приложение 1 процедура создания куба в вырезанным фрагментом.

void draw\_model() {

glBegin(GL\_LINES);

glColor3f(0, 1, 0);

glVertex3f(0, 0, 0);

glVertex3f(1, 0, 0);

glColor3f(1, 0, 0);

glVertex3f(0, 0, 0);

glVertex3f(0, 1, 0);

glColor3f(0, 0, 1);

glVertex3f(0, 0, 0);

glVertex3f(0, 0, 1);

glEnd();

glColor3f(1, 1, 1);

glBegin(GL\_POLYGON);

glColor3f(1, 0, 0);

glVertex3f(0, 0, 0.5);

glVertex3f(0.5, 0, 0.5);

glVertex3f(0.5, -0.5, 0.5);

glVertex3f(0, -0.5, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(-0.5, 0.5, 0.5);

glVertex3f(0, 0.5, 0.5);

glVertex3f(0, -0.5, 0.5);

glVertex3f(-0.5, -0.5, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(0.6, 0.7, 0.9);

glVertex3f(-0.5, -0.5, 0.5);

glVertex3f(-0.5, -0.5, -0.5);

glVertex3f(-0.5, 0.5, -0.5);

glVertex3f(-0.5, 0.5, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(0, 1, 0);

glVertex3f(-0.5, 0.5, 0.5);

glVertex3f(-0.5, 0.5, -0.5);

glVertex3f(0, 0.5, -0.5);

glVertex3f(0, 0.5, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(0, 0, 1);

glVertex3f(0, 0, 0.5);

glVertex3f(0, 0.5, 0.5);

glVertex3f(0, 0.5, -0.5);

glVertex3f(0, 0, -0.5);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(1, 1, 0);

glVertex3f(0, 0, -0.5);

glVertex3f(0, -0.5, -0.5);

glVertex3f(0.5, -0.5, -0.5);

glVertex3f(0.5, 0, -0.5);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(0, 0.5, -0.5);

glVertex3f(-0.5, 0.5, -0.5);

glVertex3f(-0.5, -0.5, -0.5);

glVertex3f(0, -0.5, -0.5);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(1, 0, 1);

glVertex3f(0.5, -0.5, 0.5);

glVertex3f(0.5, -0.5, -0.5);

glVertex3f(-0.5, -0.5, -0.5);

glVertex3f(-0.5, -0.5, 0.5);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(0, 1, 1);

glVertex3f(0.5, 0, 0.5);

glVertex3f(0, 0, 0.5);

glVertex3f(0, 0, -0.5);

glVertex3f(0.5, 0, -0.5);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(1, 1, 1);

glVertex3f(0.5, -0.5, 0.5);

glVertex3f(0.5, 0, 0.5);

glVertex3f(0.5, 0, -0.5);

glVertex3f(0.5, -0.5, -0.5);

glEnd();

}