

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
«Ижевский государственный технический университет
имени М.Т. Калашникова»

Кафедра «Программное обеспечение»

Л.Н. Левицкая

Компьютерная графика

методическое пособие по выполнению курсовой работы
«Формирование и визуализация 3D-изображения
с использованием однородных координат»
направление 09.04.04 «Программная инженерия»,
программа «Разработка программно-информационных систем»

Ижевск, 2025 г.

Л.Н. Левицкая, доцент кафедры «Программное обеспечение»

Рецензент:

Телегина Марианна Викторовна

доцент кафедры «Автоматизированные системы
обработки информации и управления»

Рекомендовано к изданию на заседании ученого совета института «Информатика и вычислительная техника» ФГБОУ ВО ИжГТУ имени М.Т. Калашникова (протокол № 2 от 21.02.2022 г.)

Методическое пособие для выполнения курсовой работы «Формирование и трансформация 3D-изображения с использованием однородных координат» при изучении дисциплины «Компьютерная графика» для студентов, обучающихся по направлению «Программная инженерия»/ сост. Л.Н. Левицкая, Ижевск, Изд-во ФГБОУ ВО ИжГТУ имени М.Т. Калашникова, 2022.-15с.

В учебно-методическом пособии рассмотрены цели, задачи, состав и содержание курсовой работы, а также методика основных этапов программирования. Представлен теоретический материал, необходимый для выполнения курсовой работы. Раскрывается последовательность программирования поставленной задачи. Методические рекомендации направлены на ознакомление студентов с основами компьютерной 3D-графики, освоением и практическим применением представленного теоретического материала.

Для закрепления материала автором предусмотрены вопросы для самоконтроля после выполнения курсовой работы.

Издание предназначено для использования преподавателями и студентами, обучающимися по направлению подготовки 09.04.04 «Программная инженерия», при изучении дисциплины «Компьютерная графика».

УДК 004.42(07)

ОГЛАВЛЕНИЕ

Цели и задачи выполнения курсовой работы	5
Содержание отчета	5
Понятие проекции в машинной графике	6
Понятие однородных координат	7
Линейное преобразование векторов и суперпозиция линейных преобразований	7
Структура матрицы однородных координат (3D-пространство)	9
Аппарат однородных координат для трехмерного пространства	9
Системы координат	10
Матрица ортогонального проецирования	11
Камера (наблюдатель)	13
Пространственное описание сцены (файл <i>obj</i>)	15
Освещение и нормаль к плоскости	16
Порядок выполнения курсовой работы	17
Контрольные вопросы	18
Литература	18

Цели и задачи выполнения курсовой работы

Целью курсовой работы является:

- закрепление теоретических знаний по дисциплине «Компьютерная графика»;
- приобретение практических навыков для разработки программ визуализации 3D-изображений и их трансформации с использованием математического аппарата однородных координат.

Курсовая работа проводится с бакалаврами, обучающимися по направлению 09.04.04 «Программная инженерия».

В ходе выполнения курсовой работы студенты должны ознакомиться с теоретическим материалом и написать программу визуализации 3D-изображения и его трансформации.

Этапы выполнения курсовой работы:

- загрузить изображение (текстовый файл с расширением *obj*);
- определить проекцию;
- реализовать z-буфер;
- подключить освещение;
- определить местоположение наблюдателя;
- визуализировать 3D-изображение и реализовать движение наблюдателя с учетом его местоположения в пространстве сцены.

Разработка программ осуществляется в среде Visual Studio с использованием языков программирования C, C++, C#, Java.

Содержание отчета

Предварительно представить отчет по курсовой работе можно в электронном виде по почте *lsda8959@gmail.com* (в формате *.docx или *.pdf; дополнительно к сообщению необходимо прикрепить тексты исходных и exe-файлов). Окончательный вариант отчета представляется студентом преподавателю в печатном виде.

Отчет должен соответствовать требованиям оформления документов такого типа и содержать следующие пункты:

1. Титульный лист с указанием темы курсовой работы
2. Постановку задачи
3. Описание программы:
 - структура входных данных;
 - структура выходных данных;
 - алгоритм программы.
4. Листинг текста программы
5. Выводы

При сдаче курсовой работы, студент должен ответить на контрольные вопросы, список которых представлен ниже.

Понятие проекции в машинной графике

ПРОЕКЦИЯ – результат проецирования, определяемого центром проекции – некоторой точкой S пространства, и плоскостью проекции (в машинной графике – плоскостью изображения) – некоторой плоскостью P , не проходящей через точку S . Центральная проекция (в машинной графике – перспективная проекция) точки A пространства есть точка A^* – точка пересечения прямой SA с плоскостью P (рис. 1).

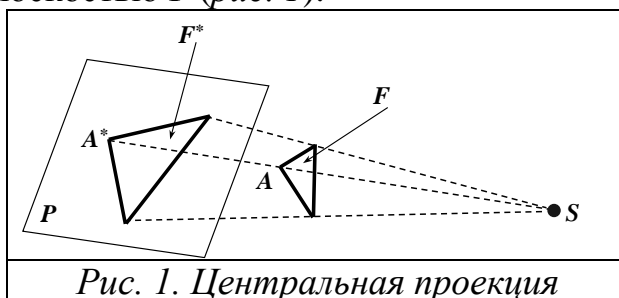


Рис. 1. Центральная проекция

Итак, для определения центральной проекции необходимо задать только два элемента – точку (центр проекции) и плоскость проекции. При этом плоскость проекции не должна проходить через центр проекции. Все линии проецирования проходят через одну точку – центр проекции.

Кроме того, для центральной проекции произведем следующие уточнения (рис. 2). Линия, проходящая через центр проекции S и перпендикулярная плоскости проекции P , называется оптической осью, а расстояние f от точки S до точки пересечения оптической оси с плоскостью P – фокусным расстоянием.

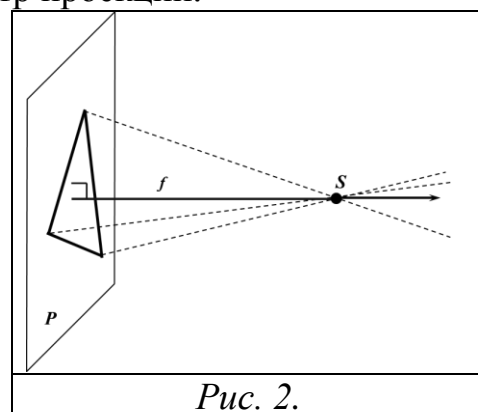


Рис. 2.

Другой также важный случай проекции – параллельная проекция.

Проекция с бесконечно удаленным центром проецирования называется параллельной. В этом случае все линии проецирования определяются направлением проецирования, они параллельны друг другу и, в отличие от центральной проекции, не проходят через одну точку – центр проекции (рис. 3).

В случае, если плоскость проекции расположена перпендикулярно к направлению проецирования, то проекция называется ортогональной.

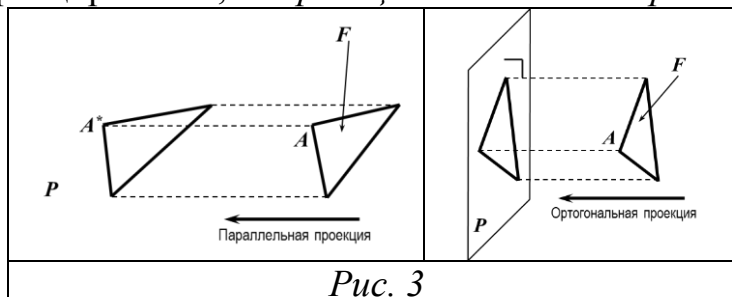


Рис. 3

Понятие однородных координат

Координаты точек проекции могут быть получены методом геометрических построений. Однако в машинной графике применяется другой метод, основанный на математическом аппарате однородных координат, который является гораздо более эффективным. Эта эффективность проявляется как при использовании аппарата однородных координат человеком, так и при его аппаратно-программной реализации в компьютере.

Переход от обычных пространственных координат к однородным сопровождается не совсем обычным математическим трюком – увеличением количества координат с двух до трех – для двумерного пространства, с трех до четырех – для трехмерного пространства. Поэтому, можно рассматривать вектор как точку в этом пространстве с соответствующими значениями координат, либо как вектор, проведенный из начала координат в эту точку.

Например, двумерный вектор (x, y) в однородных координатах записывается в виде (wx, wy, w) , где $w \neq 0$. Число w называется масштабным множителем. Для того, чтобы из вектора, записанного в однородных координатах, получить вектор в обычных декартовых координатах необходимо разделить первые две координаты на третью:

$$(wx, wy, w) \rightarrow (wx/w, wy/w) \rightarrow (x, y).$$

Линейное преобразование векторов и суперпозиция линейных преобразований

Операция линейного преобразования векторов реализуется путем умножения матрицы на вектор. Вектор записывается как последовательность числовых значений его компонент. Количество компонент соответствует размерности того пространства, которое мы рассматриваем. В двумерном пространстве – 2, в трехмерном – 3, в четырехмерном – 4 и т.д. Значения этих компонент есть не что иное, как координаты точки в этом пространстве.

Существует два способа хранения матриц: *column-major* и *row-major*. На лекциях по линейной алгебре как раз используется схема *row-major*. По большому счёту представление матриц в памяти компьютера не имеет значения, потому что матрицу всегда можно перевести из одного вида представления в другое простым транспонированием. А раз разницы нет, то для всех последующих расчётов можно использовать классические *row-major* матрицы. В программу матрицу нужно передавать как есть, а в программе производить умножение не вектора на матрицу, а матрицы на вектор (рис. 4).

$\begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 3 & 1 \end{pmatrix}$	\times	$\begin{pmatrix} 4 \\ 1 \\ 2 \\ 1 \end{pmatrix}$	$=$	$\begin{pmatrix} 6 \\ 5 \\ 7 \\ 7 \end{pmatrix}$
Рис. 4				

Полученный результат представляет собой компоненты результирующего

вектора в пространстве однородных координат.

В отличие от вектора – объекта-точки в пространстве, матрица описывает линейный оператор преобразования этого пространства, т.е. функцию, преобразующую любую исходную точку пространства в некоторую другую точку этого же пространства. Поэтому результатом перемножения матриц является новое результирующее линейное преобразование. Это новое линейное преобразование соответствует последовательному применению преобразований, что собственно и называется суперпозицией преобразований.

Допустим, мы имеем некоторую точку пространства \mathbf{a} . Мы хотим сначала эту точку переместить (матрица \mathbf{T}), а затем повернуть (матрица \mathbf{G}).

Мы можем сначала $\mathbf{a}_T = \mathbf{T} \times \mathbf{a}$ – сместить точку.

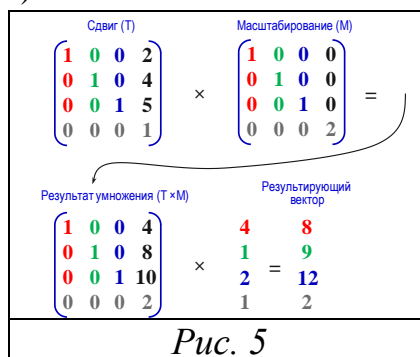
Затем $\mathbf{a}_{TG} = \mathbf{G} * \mathbf{a}_T$ – повернуть.

А можно поступить иначе:

$\mathbf{T} * \mathbf{G} = \mathbf{TG}$, а затем $\mathbf{TG} \times \mathbf{a} = \mathbf{a}_{TG}$.

Таким образом, вместо последовательного применения двух или нескольких преобразований мы можем с помощью перемножения матриц получить одно результирующее преобразование, и затем применять его к точкам пространства. В этом смысле операция перемножения матриц соответствует суперпозиции преобразований

Правило перемножения матриц достаточно простое (основной принцип – строка \times столбец) (рис. 5).



На рис.5 показано перемножение матрицы сдвига и матрицы масштабирования (суперпозиция преобразований). После этого результирующая матрица умножается на вектор-столбец, содержащий компоненты исходного вектора. Результат – преобразованный вектор в пространстве однородных координат.

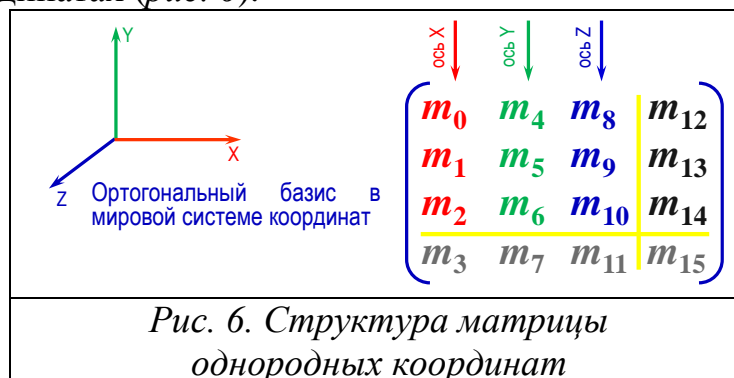
Чтобы перейти в пространство декартовых координат, необходимо первые три компоненты результирующего вектора (x, y, z) разделить на последнюю компоненту (w) (масштабный множитель):

$$(8, 9, 12, 2) \rightarrow 8/2, 9/2, 12/2 \rightarrow (4, 4.5, 6).$$

Структура матрицы однородных координат (3D-пространство)

Рассмотрим аппарат однородных координат для трехмерного пространства. Матрицы преобразований в однородных координатах увеличивают свои размеры с 3×3 до 4×4 . Т.е. трехмерному пространству декартовых координат соответствует четырехмерное пространство однородных координат.

Произведем некоторый анализ структуры матриц преобразований в однородных координатах (рис. 6).



В общем случае матрицу можно разбить на 4 подматрицы размерами 3×3 , 3×1 , 1×3 и 1×1 (здесь речь идет о трехмерном пространстве декартовых координат и соответствующем ему четырехмерном пространстве однородных координат).

Первая матрица размером 3×3 отвечает за преобразования поворота и тройного масштабирования.

Вторая матрица размера 3×1 (вектор-столбец) отвечает за преобразование сдвига.

Третья матрица размера 1×3 (вектор-строка) отвечает за преобразования центрального и ортогонального проецирования.

Четвертая матрица размера 1×1 (одноэлементная матрица) отвечает за преобразование общего масштабирования.

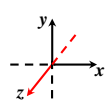
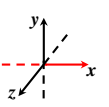
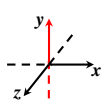
Перемножая матрицы отдельных частных видов преобразований, в результате суперпозиции получим матрицу сложного комбинированного преобразования, в которой будут содержаться параметры всех этих частных преобразований.

Однако, сама по себе ситуация в трехмерном пространстве усложняется. Во-первых, в отличие от двумерного пространства, здесь становится возможным поворот вокруг любой оси, произвольно ориентированной в пространстве, или, иначе, становятся возможными три степени свободы поворотов вокруг каждой из трех координатных осей.

Аппарат однородных координат для трехмерного пространства

Все пространственные преобразования объектов реализуются в пространстве однородных координат. Для дальнейшего рассмотрения примем масштабный множитель равным единице ($w = 1$). Также, при этом, оптическая ось совпадает с осью z (см. рис. 6).

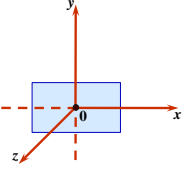
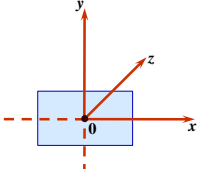

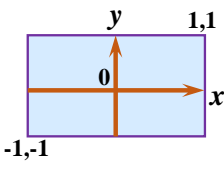
На рис. 7 представлены матрицы преобразований однородных координат.

$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{f} & 1 \end{bmatrix}$		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$			
Матрица преобразования центрального проецирования (f – фокусное расстояние)		Матрица преобразования ортогонального проецирования			
$\begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & k \end{bmatrix}$	$\begin{bmatrix} k_x & 0 & 0 & 0 \\ 0 & k_y & 0 & 0 \\ 0 & 0 & k_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$			
Матрица сдвига		Матрица масштабирования		Матрица тройного масштабирования	
$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$		$\begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	
	Матрица поворота вокруг оси z		Матрица поворота вокруг оси x		Матрица поворота вокруг оси y
Рис. 7. Матрицы преобразований однородных координат					

Системы координат

В курсовой работе предполагается, что мы будем ориентироваться на графическую систему OpenGL.

В OpenGL используются *три основные системы координат*: правосторонняя, левосторонняя и оконная (экранная) (рис. 8).

			
правосторонняя	левосторонняя	экранная	нормализованные координаты
Рис. 8			

Первые две системы являются трехмерными и отличаются друг от друга направлением оси z . В правосторонней ось z направлена на наблюдателя, а в левосторонней – в глубь экрана. Расположение осей x и y одинаково в обеих системах.

Левосторонняя система используется для установки параметров, связанных с формированием проекций. Т.е., эти параметры используются в командах формирования матриц проекций (перспективной, ортогональной). При выполнении курсовой работы будем формировать *ортогональную* проекцию.

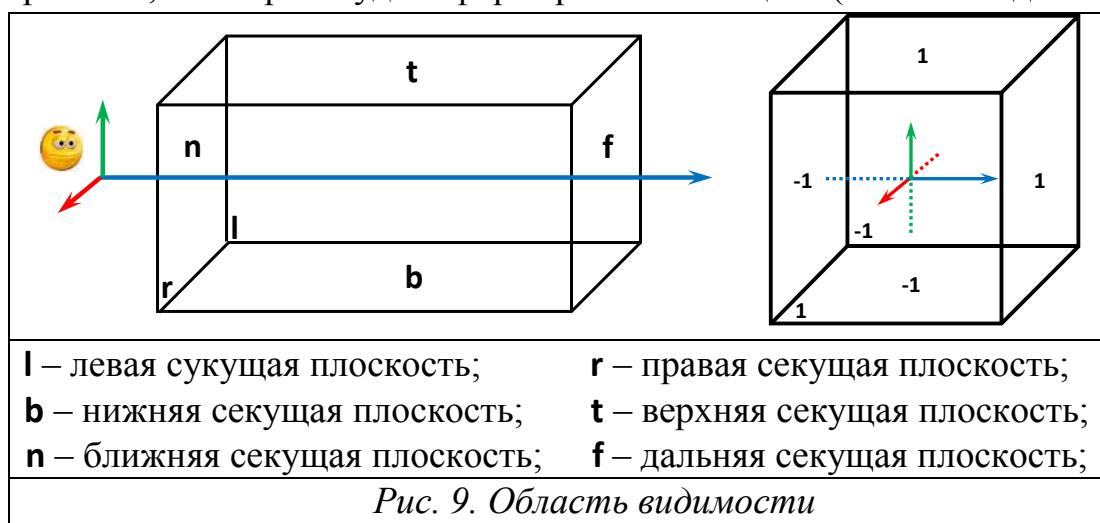
Отображение трехмерной информации происходит в двумерную оконную (экранную) систему координат.

Нормализованной координатой называют координату, заданную в промежуточной, независимой от устройств, системе координат и нормированную относительно некоторого диапазона, обычно от 0 до 1. На *рис. 8* нормализованные координаты заданы в диапазоне от -1 до +1. В курсовой работе координатное описание модели (*.obj) задано в нормализованных координатах в диапазоне от -1 до +1. Считается, что наблюдатель находится в центре системы координат и его взгляд направлен в сторону отрицательной полуоси z .

Отображение трехмерной информации происходит в двумерную экранную систему координат. Для визуализации объекта, в конечном итоге, нам необходимо получить экранные координаты, т.е. целочисленные.

Матрица ортогонального проецирования

Для формирования 3D-сцены, необходимо определить и ограничить пространство, в котором будет сформирована 3D-сцена (область видимости).



Например, в OpenGL реализована команда, которая задает область видимости сцены и формирует матрицу проекции. На рисунке показано ограниченное пространство сцены (объем видимости). Всё, что попадает в эту область, подлежит растеризации, всё, что находится вне области, игнориру-

ется. Всё, что частично выходит за границы объема, подлежит алгоритмам отсечения.

Матрица ортогональной проекции выглядит следующим образом (рис. 10). При умножении координат на матрицу проекции исходные координаты преобразуются в нормализованные. Т.е., изображение, сформированное в области видимости (а область может быть любого размера), заданной в команде формирования проекции, масштабируется в область куба размером $2 \times 2 \times 2$ (рис. 9).

Прототип функции, формирующей ортогональную проекцию, имеет следующий вид:

```
Matrix ProjOrto(double l, double r,  
double b, double t, double n, double f);
```

И, наконец, нормализованные координаты преобразуются в экранные координаты. Обычно такая функция называется *Viewport*. Эта функция формирует матрицу (рис. 11). Здесь:

w – ширина окна;

h – высота окна;

x, y – координаты левого нижнего угла прямоугольника вывода.

Умножая нормализованные координаты на матрицу *Viewport*, получим целочисленные координаты в пределах клиентской области окна.

$2.0/(r-l)$	0	0	$-(r+l)/(r-l)$
0	$2.0/(t-b)$	0	$-(t+b)/(t-b)$
0	0	$-2.0/(f-n)$	$-(f+n)/(f-n)$
0	0	0	1

Рис. 10. Матрица ортогональной проекции

$w / 2.0$	0	0	$x + w / 2.0$
0	$h / -2.0$	0	$y + h / 2.0$
0	0	$depth / 2.0$	$depth / 2.0$
0	0	0	1

Рис. 11. Матрица Viewport

Z-буфер

Алгоритм, использующий *z-буфер*, это один из простейших алгоритмов удаления невидимых поверхностей. Впервые он был предложен Кэтмулом. Работает этот алгоритм в пространстве изображения.

Идея *z-буфера* является простым обобщением идеи о буфере кадра. Буфер кадра используется для запоминания атрибутов (интенсивности/цвета) каждого пиксела в пространстве изображения.

Z-буфер – это отдельный буфер глубины, используемый для запоминания координаты z или глубины каждого видимого пиксела в пространстве изображения.

Главное преимущество алгоритма – его простота.

Основной недостаток алгоритма – большой объем требуемой памяти. Например, буфер кадра размером $512 \times 512 \times 24$ бит в комбинации с *z-буфером* размером $512 \times 512 \times 20$ бит требует почти 1,5 мегабайт памяти.

Другой недостаток алгоритма *z-буфера* состоит в трудоемкости и высокой стоимости устранения лестничного эффекта, а также реализации эффектов прозрачности и просвечивания.

Формальное описание алгоритма *z-буфера*:

- заполнить *z-буфер* минимальным значением z ;
- преобразовать каждый треугольник в растровую форму в произвольном порядке;
- для каждого пиксела (x, y) в треугольнике вычислить его глубину $z(x, y)$
- сравнить глубину $z(x, y)$ со значением *z-буфера* $z(x, y)$, хранящимся в этой же позиции.

Если $z(x, y) > z\text{-буфер}(x, y)$, то записать атрибут этого многоугольника (интенсивность, цвет и т. п.) в буфер кадра и заменить *z-буфер* $z(x, y)$ на $z(x, y)$. В противном случае никаких действий не производить.

Камера (наблюдатель)

Как было сказано выше, при выполнении курсовой работы мы ориентируемся на графическую библиотеку OpenGL. В этой библиотеке отсутствует концепция камеры. Однако можно попытаться её симитировать, перемещая все объекты сцены в направлении противоположном движению наблюдателя, и тем самым создать иллюзию, что движемся мы сами.

Когда мы говорим о пространстве камеры (наблюдателя), мы подразумеваем вид всех вершин с точки зрения наблюдателя, положение которого в этом пространстве является базовой точкой начала координат. Т.е. у наблюдателя своя система координат.

Матрица вида преобразует мировые координаты (правосторонняя система координат, рис. 8) в координаты вида, которые измеряются относительно расположения и направления камеры. Т.е. пересчитывает значения координат из мировой системы координат – в систему координат наблюдателя.

Для однозначного математического описания камеры, нам необходимо ее положение в мировом пространстве; направление, в котором она смотрит; вектор, указывающий правое направление, и вектор, указывающий направление вверх. На самом деле мы собираемся создать систему координат с 3 перпендикулярными осями и позицией камеры в качестве точки отсчета.

Желательно привести полученные векторы к единичному размеру. Т.е. осуществить нормализацию вектора. Нормализация вектора – это преобразование заданного вектора в вектор в том же направлении (то есть в коллинеарный, параллельный вектор), но с единичной длиной.

Для нормализации вектора нужно каждую компоненту поделить на длину вектора.

Получить позицию камеры легко.

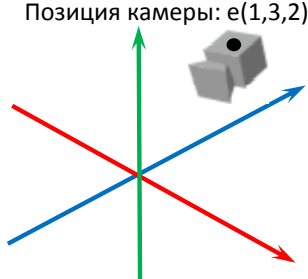
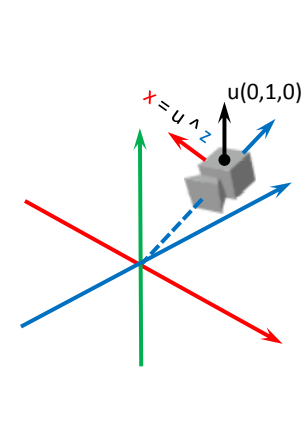
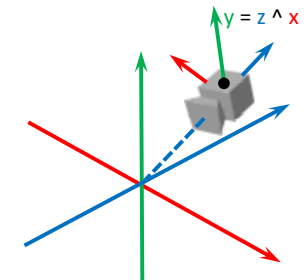
 <p>Позиция камеры: $c(1,3,2)$</p>	<p>Положение камеры – это вектор, содержащий координаты камеры в мировом пространстве. В нашем случае – $(1,3,2)$.</p>
 <p>$z(0,0,0)$</p>	<p>Прежде всего, необходимо получить вектор направления камеры. На рисунке камера нацелена в базовую точку сцены – $(0,0,0)$. Если вычесть два вектора друг из друга, то получим вектор, являющийся разностью исходных векторов. Вычитание вектора положения камеры из точки начала координат даст нам вектор направления камеры – ось Z.</p>
 <p>$x = u \wedge z$ $u(0,1,0)$</p>	<p>Далее необходимо получить вектор, указывающий в правую сторону и представляющий положительное направление оси X камеры. Для его вычисления, воспользуемся небольшим трюком: сначала зададим вектор, указывающий направление вверх – $u(0,1,0)$ (в мировом пространстве). Затем реализуем векторное произведение: $X = u \wedge Z$.</p> <p>Так как результатом векторного произведения является вектор, перпендикулярный исходным векторам, то получим вектор, указывающий в положительном направлении оси X.</p>
 <p>$y = z \wedge x$</p>	<p>Теперь, перемножив векторы двух осей X и Z, получим вектор, указывающий в положительном направлении оси Y.</p>

Рис. 12. Система координат наблюдателя

С помощью векторного произведения и небольших хитростей мы смогли рассчитать все векторы, которые задают пространство наблюдателя.

Теперь можно сформировать матрицу *LookAt*, необходимую для создания камеры.

Одно из замечательных свойств матриц заключается в том, что задав координатное пространство с помощью 3 перпендикулярных (или линейно независимых) осей, и вектора смещений, можно сформировать матрицу, умножение на которую преобразует любые векторы (точки пространства) в это

определенное координатное пространство. Это именно то, что делает матрица *LookAt*.

Использование матрицы *LookAt* позволяет эффективно преобразовать мировые координаты в заданное пространство наблюдателя (рис. 13).

x_x	x_y	x_z	$-c_x$	(x_x, x_y, x_z) – правый вектор (y_x, y_y, y_z) – вектор, указывающий вверх (z_x, z_y, z_z) – вектор направления камеры (c_x, c_y, c_z) – позиция камеры
y_x	y_y	y_z	$-c_y$	
z_x	z_y	z_z	$-c_z$	
0	0	0	1	

Рис. 13. Матрица *LookAt*

Пространственное описание сцены (файл *obj*)

Генерация объемных изображений представляет сложную вычислительную задачу, в связи этим на практике выполняют ее декомпозицию. Сложные изображения формируют из фрагментов объектов, для чего их разбивают на составные части. Процесс разбиения поверхности объектов на полигоны получил название тесселяции. Процесс разбиения полигональной области со сложной конфигурацией на набор треугольников называется *триангуляцией*.

В компьютерной графике аппроксимация сложных поверхностей наиболее часто осуществляется сеткой треугольников с последующим оперированием простейшими полигональными областями, т.е. с каждым из треугольников.

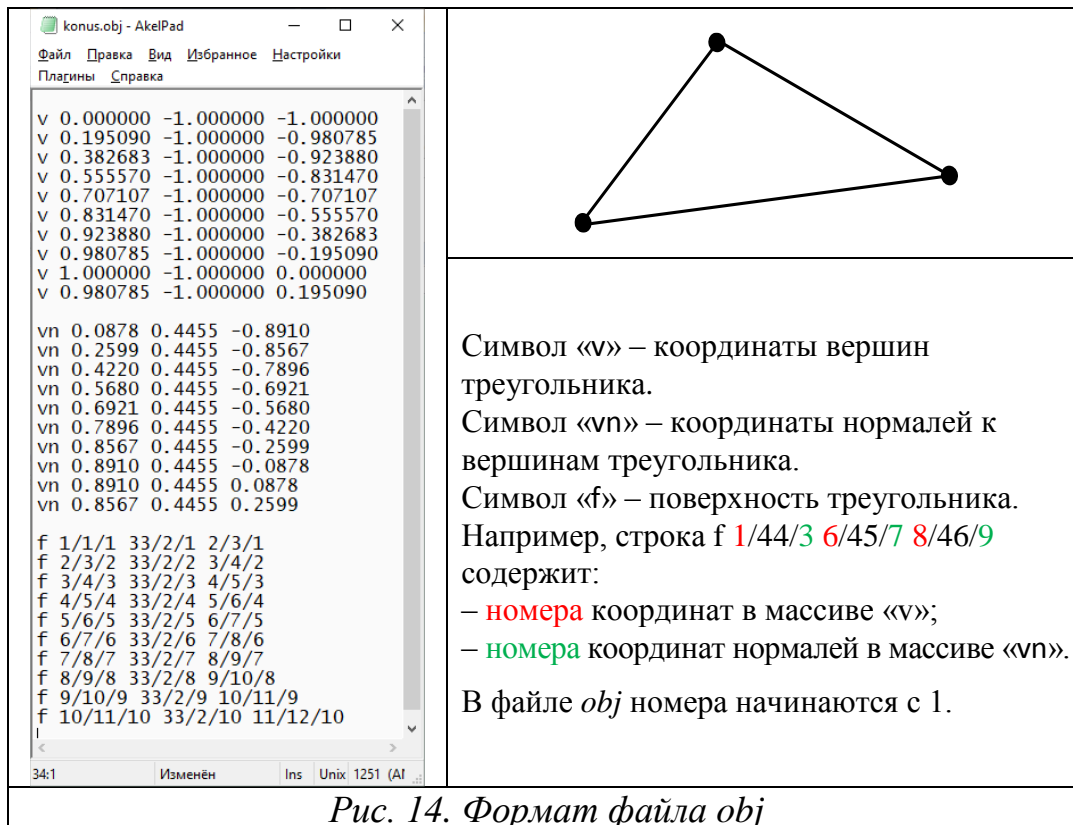
Трехмерные объекты, пространственное описание которых представлено в файле *obj*, аппроксимированы треугольниками. Поэтому, по сути, формат файла *obj* – это координатное описание отдельных треугольников.

Файл *obj* содержит геометрию 3D-модели:

- список вершин;
- координаты и нормали вершин;
- карту текстур;
- полигональную сетку и т.д.

Формат *obj* является общепринятым стандартом для хранения, импорта, экспорта 3D-объектов и поддерживается различным программным обеспечением трехмерного моделирования.

Пример файла *obj* и его краткое описание представлен на рис. 14.

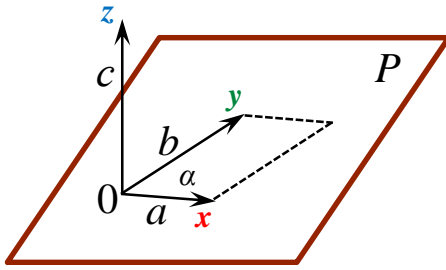
Рис. 14. Формат файла *obj*

Освещение и нормаль к плоскости

Нормаль к плоскости есть вектор, перпендикулярный данной плоскости. Именно нормаль определяет ориентацию плоскости в трехмерном пространстве. Используя вектор нормали, графическая система осуществляет раскрашивание плоской поверхности, учитывая положение источника света в пространстве сцены и его оптические характеристики, а также положение наблюдателя в пространстве сцены и направление его взгляда.

Так как аппроксимация сложных поверхностей объектов осуществляется треугольниками (триангуляция) и, по сути, необходимо для каждого треугольника определить свою нормаль, то процесс этот достаточно сложный.

Поэтому рассмотрим основные принципы расчета вектора нормали к плоской поверхности. Плоскость однозначно определяется тремя точками, не лежащими на одной прямой. Если на плоскости P мы возьмем три точки и проведем из точки O два вектора (a и b), то векторы a и b параллельны плоскости P и для определения нормали используется их векторное произведение $a \times b = c$ (рис. 15).

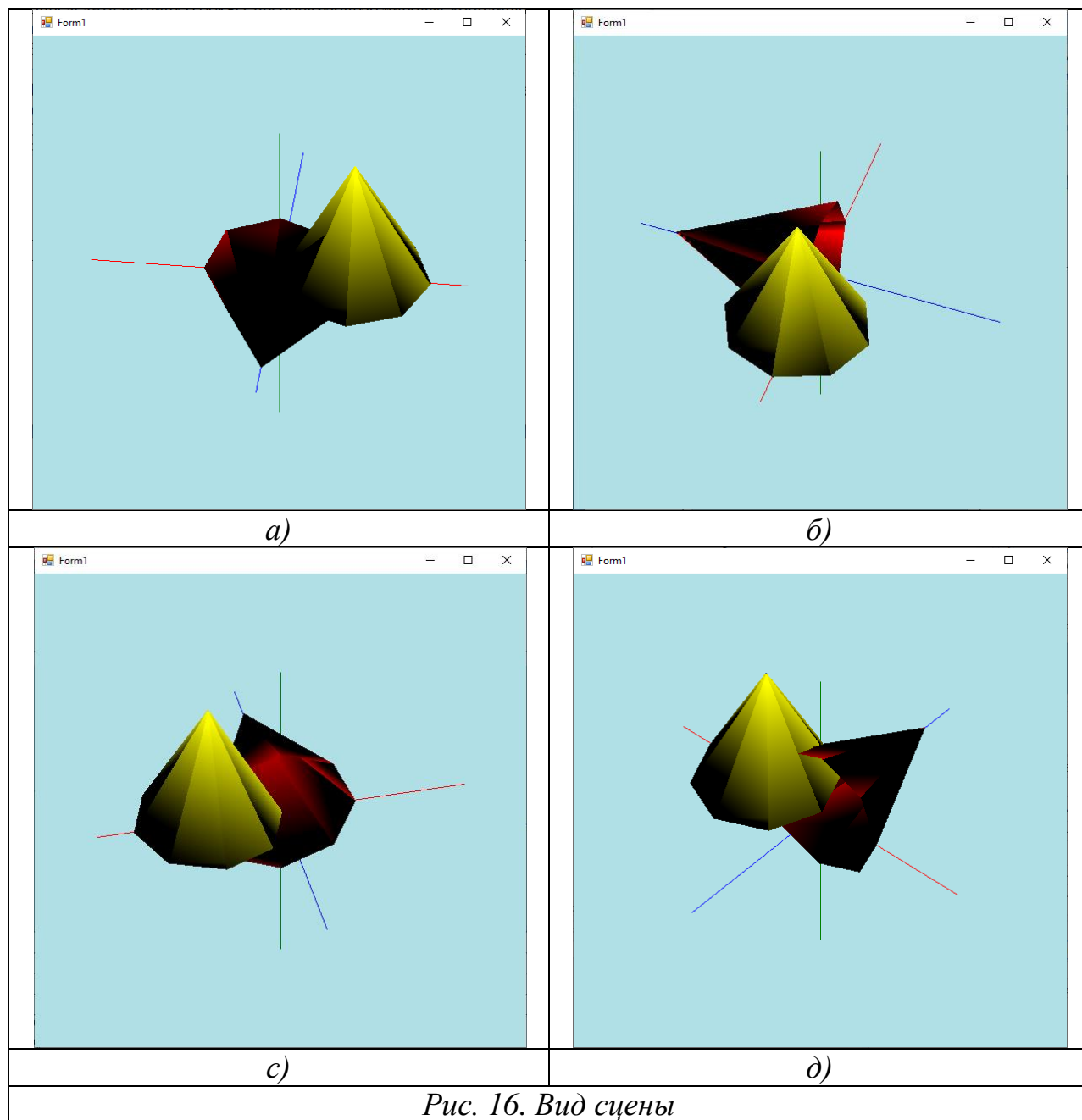
	<p>Векторным произведением вектора a (множимое) на не коллинеарный ему вектор b (множитель) называется третий вектор c (произведение), который строится следующим образом:</p> <ul style="list-style-type: none"> • его модуль равен площади параллелограмма (лежащего в плоскости P), построенного на векторах a и b, т.е. он равен: $a \times b \times \sin \alpha$; • его направление перпендикулярно плоскости P; • при этом направление вектора c (из двух возможных) выбирается так, чтобы векторы a, b, c составляли правую систему, или, иначе, выбирается по правилу буравчика.
Рис. 15. Векторное произведение	

В файле *obj* в строке «vn» представлено координатное описание нормали к соответствующей вершине треугольника. Считаем интенсивность освещения для каждой вершины треугольника и интерполируем интенсивность внутри треугольника. Получить интенсивность светового потока для каждой вершины треугольника просто. Для этого необходимо реализовать скалярное произведение двух векторов: вектор источника света (позиция в пространстве сцены) и вектор (координаты) вершины треугольника в трехмерном пространстве.

Порядок выполнения курсовой работы

1. Загрузить в разработанные структуры данных файл с расширением *obj*.
2. Используя матрицу ортогональной проекции определить область видимости сцены в нормализованных координатах.
3. Определить матрицы видового преобразования: поворота, сдвига, масштабирования.
4. Определить положение наблюдателя в пространстве сцены.
5. Определить матрицу (Lookat), преобразующую мировые координаты в пространство наблюдателя.
6. Задать матрицу (Viewport), преобразующую нормализованные координаты объекта в экранные.
7. Используя суперпозицию линейных преобразований (перемножение матриц) определить текущую матрицу преобразований. Например:
 $MT = VP \times MP \times Lookat$;
8. Реализовать движение наблюдателя вокруг сцены.

На рис. 16 показан вид сцены с определенного положения наблюдателя. Наблюдатель перемещается вокруг объектов по кругу против часовой стрелки, начиная с положения $a) \rightarrow b) \rightarrow c) \rightarrow d)$.



Контрольные вопросы

1. Основные элементы, необходимые для задания центральной проекции.
2. Понятие параллельной проекции. Ее основное отличие от центральной проекции.
3. Однородные координаты и их количество для трехмерного пространства.
4. Описание точки в пространстве однородных координат и ее преобразование в пространство декартовых координат.
5. Продолжить определение: матрица в пространстве однородных координат – это
6. Продолжить определение: суперпозиция линейных преобразований – это
7. Структура матрицы однородных координат для трехмерного пространства.
8. Отличие правосторонней и левосторонней систем координат.
9. Что такое Z-буфер?
10. Нормаль к плоскости и ее роль в раскрашивании плоской поверхности.