лекция №6

Тема: Трехмерные координаты. Трехмерное векторное изображение

цель: Реализовать каркасное изображение трехмерных объектов.

ПЛАН

1. Матричный запись преобразования трехмерных координат

2. Трехмерная сцена

3. Камера

4. Тест глубины, отсечение невидимых точек

1. Матричный запись преобразования трехмерных координат

В пятой лекции показано вывода матрицы преобразования для двумерного пространства:

$$\begin{pmatrix} y_1 \\ x_1 \\ 1 \end{pmatrix} = \begin{pmatrix} -m_x \sin(\alpha) & m_y \cos(\alpha) & dy \\ m_x \cos(\alpha) & m_y \sin(\alpha) & dx \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} y_0 \\ x_0 \\ 1 \end{pmatrix}.$$

Воспроизведение трехмерных объектов требует работы с тремя пространственными координатами, и для учета третьей координаты, которой преобразования не касается матрица превращается так:

$$\begin{pmatrix} y_1 \\ z_1 \\ x_1 \\ 1 \end{pmatrix} = \begin{pmatrix} -m_x \sin(\alpha) & m_y \cos(\alpha) & 0 & dy \\ 0 & 0 & 1 & 0 \\ m_x \cos(\alpha) & m_y \sin(\alpha) & 0 & dx \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} y_0 \\ z_0 \\ x_0 \\ 1 \end{pmatrix}.$$

В приведенной матрицы использовано преобразования масштабирования размеров по осям OX, OY а также поворота на угол относительно оси OZ. Однако более выгодно комбинировать преобразования отдельно:

$$\begin{pmatrix} y_1 \\ z_1 \\ x_1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 & m_y & 0 & dy \\ 0 & 0 & m_z & dz \\ m_x & 0 & 0 & dx \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 & dx \\ 0 & \cos(\gamma) & \sin(\gamma) \\ 0 & -\sin(\gamma) & \cos(\gamma) \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \\ \cos(\beta) & 0 & \sin(\beta) \end{pmatrix} \cdot \begin{pmatrix} \sin(\alpha) & 0 & 0 \\ \cos(\alpha) & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} y_0 \\ z_0 \\ x_0 \\ 1 \end{pmatrix}.$$

По этому преобразованием существенное значение имеет порядок выполненных преобразований, поскольку в

случае переноса точки с последующим вращением, вращаться уже перенесена точка. Напротив, комбинация вращения и переноса даст другое положение фигуры в пространстве. Поэтому в приведенном примере полного преобразования использовано последовательность вращения вокруг осей, а затем операцию переноса, что позволит сохранить ожидаемое место переноса с указанным увеличением.

Указанная схема преобразований, позволяет строить преобразования вращения вокруг заданной точки в

пространстве. Пусть точкой вращения является точка (*a, b, c)*. Тогда вращения можно обозначить так: (x_1

$$\begin{matrix} & & \sin(\alpha)\ 0\ 0 & & \\ y_1 & 0\ 1\ 0\ b & -\sin(\alpha)\ \cos(\alpha)\ 0\ 0 & 0\ 1\ 0\ -b & y_0 \\ z_1 & 0\ 0\ 1\ c & 0\quad 0\quad 1\ 0 & 0\ 0\ 1\ -c & z_0 \\ 1) = (1\ 0\ 0\ 0\ a\ 1) \cdot (\cos(\alpha) & 0\quad 0 & 0\ 1) \cdot (1\ 0\ 0\ 0\ -a) \cdot (x_0 & 1) . \end{matrix}$$

Смысл в таком преобразовании есть в повестке преобразований: 1) передвигаем объект в центр вращения; 2) обращаем на заданный угол; 3) поворачиваем фигуру в прежнее положение. Этот пример наглядно показывает преимущество матричного записи преобразования, где вращения вокруг заданной точки с помощью выражения записать довольно трудно.

2. Трехмерная сцена

Трехмерная сцена является совокупность трехмерных объектов, заданные списком координат точек и перечнем плоских полигонов по порядковым номерам указанных точек. Также на объект, как общая величина определяется цвет и правило обработки освещенности, правила окрашивания. Для определения положения всех объектов задаются матрицы преобразования координат. При этом, для одного и того же объекта может быть несколько матриц преобразования. Например, такое можно использовать при наличии нескольких копий объекта: за столом расположить несколько стульев. Существенно, что загружать много объектов-стульев не обязательно, достаточно каждому из стульев указать нужный набор вершин и полигонов и матрицу преобразования для нужного расположения стула в пространстве.

С вышеупомянутого можно сделать вывод, что трехмерная сцена является совокупность объектов, которые являются совокупностями вершин и полигонов, а также совокупность ссылок на объекты с матрицами преобразования для правильного их расположения в пространстве. Один объект может быть на сцене расположен несколько раз.

3. Камера

Вид трехмерной сцены зависит от места наблюдения и направления взгляда. Совокупность информации, задающей место и направление наблюдения называют "камера", что является ассоциацией с кинематографической камерой, которая снимает сцену из фильма. Также на вид сцены влияет и "фокусное расстояние" "объектива" камеры, или перспективное масштабирования удаленных объектов. Если уменьшение объектов с изменением расстояния не происходит, то такая проекция называется "параллельной". Все проекции являются своеобразными преобразованиями координат, после которых учитываются только две координаты (x, y). Рассмотрим на начало параллельную проекцию.

Для создания параллельной проекции нужно всю сцену вернуть таким образом, чтобы центр экрана совпадал с началом координат, а направление зрения был по направлению оси глубины OZ. Зададим положение глаза с помощью трех координат (*a, b, c,* 1). Создадим матрицу, которая передвинет всю сцену так, чтобы точка наблюдения была в точке (0,0,0,1):

$$\begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & -a \\ 0 & 1 & 0 & -b \\ 0 & 0 & 1 & -c \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{pmatrix}.$$

Теперь, после переноса, сцену нужно повернуть на угол $-\gamma$, который определяет угол на который "поднят голову". Знак минус означает, что сцену нужно повернуть так, чтобы компенсировать наклон головы:

$$\begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix} = \begin{pmatrix} \cos(\gamma) & -\sin(\gamma) & 0 & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & -a \\ 0 & 1 & 0 & -b \\ 0 & 0 & 1 & -c \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{pmatrix}.$$

Поворот камеры "слева-направо" задает угол поворота вокруг вертикальной оси OY

$-\beta$: (x_1

$$\begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix} = \begin{pmatrix} \cos(\beta) & 0 & -\sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\beta) & 0 & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos(\gamma) & -\sin(\gamma) & 0 & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & -a \\ 0 & 1 & 0 & -b \\ 0 & 0 & 1 & -c \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{pmatrix}$$

Иногда используют дополнительно преобразования изменения масштаба и смещения координат, чтобы превратить собственные координаты к координатам совместимых с координатами устройства вывода графических примитивов. Как указано выше, для создания изображения на экране, после преобразования камеры используют только две координаты:

$$(x_1, y_1).$$

Для учета перспективных искажений используется соотношение:

$$x_e = F x / (F + z), \; y_e = F y / (F + z).$$

Здесь в качестве условной фокусного расстояния объектива использованы величину F, чем меньше фокусное расстояние, тем более значительными перспективные искажения. Также это превращение можно записать с помощью матричного произведения:

$$\begin{pmatrix} F & 0 & 0 & 0 \\ 0 & F & 0 & 0 \\ 0 & 0 & F & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} F x \\ F y \\ F z \\ z + F \end{pmatrix}.$$

В этом выражении перспективной проекции использовано однородность трехмерных координат, когда последняя координата является делителем трех предыдущих. То есть, для перехода от однородных координат к декартовым нужно воспользоваться преобразованием:

$$\begin{pmatrix} F x \\ F y \\ F z \\ z + F \end{pmatrix} \rightarrow (\text{Переход к декартовых координат}) \rightarrow \begin{matrix} x_e = F x / (z + F) \\ y_e = F y / (z + F) \\ z_e = F z / (z + F) \end{matrix}$$

что и соответствует поставленной задаче получения перспективной проекции.

4. Тест глубины, отсечение невидимых точек

Это один из самых простых алгоритмов удаления невидимых поверхностей. Впервые он был предложен Кэтмул. Применим этот алгоритм для рисования трехмерных сцен, где объекты состоят из плоских полигонов. Для лучшей наглядности рассматривать в качестве полигонов только треугольники.

Формальное описание алгоритма z-буфера:

1) заполнить буфер кадра фоновым значением интенсивности или цвета

2) заполнить z-буфер минимальным значением z;

3) превратить каждый многоугольник в растровую форму в произвольном порядке;

4) для каждого пикселя (x, y) в фоне многоугольника вычислить его глубину координату-глубину z, сравнить глубину z (x, y) со значением которое уже находится в буфере Z (x, y)

5) если z (x, y)> Z (x, y), то записать атрибут этого многоугольника (интенсивность, цвет и т. п.) в буфер кадра и заменить Z (x, y) = z (x, y)

6) в противном случае оставить цвет пикселя как есть (там уже нарисован более близок пиксель).

Для оптимизации вывода в z-буфер и для снижения количества вычислений используется тот факт, что чем раньше видимая граница будет выведена в z-буфер, тем больше невидимых граней, которые закрывают ней, будет отвергнуто. Для этого строится список тех граней, которые были выведены в предыдущем кадре. В дальнейшем кадре эти границы выводятся в zбуфер первыми, так как почти наверняка они будут оставаться видимыми и в этом кадре.

Этот алгоритм решает задачу об удалении невидимых поверхностей и делает тривиальной задачу визуализации перекрытия сложных поверхностей. Сцены могут быть любой сложности. Поскольку габариты пространства изображения фиксированы, оценка вычислительной трудоемкости алгоритма является линейной. Поскольку элементы сцены или картинки можно заносить в буфер кадра или в z-буфер в произвольном порядке, их не нужно предварительно сортировать по приоритету глубины. Это экономит вычислительный время, затрачиваемое на сортировку по глубине.

Основной недостаток алгоритма - большой объем требуемой памяти. Если сцена подвергается видовому преобразованию и отсекается до фиксированного диапазона координат z значений, то можно использовать z-буфер с фиксированной точностью. По глубине нужно обрабатывать с большей точностью, чем координатную информацию на плоскости (x, y) обычно бывает достаточно 20 бит. Буфер кадра размером 512 * 512 * 24 бит в комбинации с z-буфером размером 512 * 512 * 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. Поскольку алгоритм заносит пиксели в буфер кадра в произвольном порядке, то нелегко получить информацию, необходимую для методов устранения лестничного эффекта, основанные на предварительной фильтрации. При реализации эффектов прозрачности и просвечивания пиксели могут заноситься в буфер кадра в некорректном порядке, ведет к локальным ошибок.