Модуль 2. Основы 2D и 3D компьютерной графики

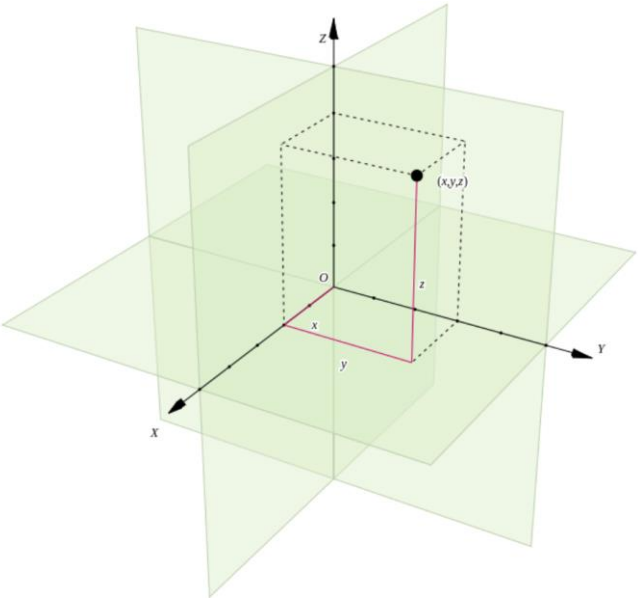
Лекция 2.1. Введение в 3D графику

Вектора и системы координат

Возьмем две взаимно перпендикулярные прямые с заданным масштабом. Их пересечение даст нам декартову прямоугольную систему координат на плоскости. Точку пересечения O в таком случае именуют началом координат, одну из осей именуют OX, или осью абсцисс, а другую - OY, или осью ординат.

Предположим, что у нас имеется некоторая точка M в пространстве. При этом точки Mx и My будут являться ее проекциями (здесь и далее мы будем подразумевать прямоугольное проецирование) на оси абсцисс и ординат соответственно, причем длину OMx мы примем за x, а длину OMy за y . В таком случае, пара значений (x, y) будет называться декартовыми координатами точки M на плоскости.

Аналогично, три взаимно перпендикулярных прямых с масштабом при пересечении дадут декартову прямоугольную систему координат в пространстве. Третья ось при этом будет именоваться OZ или осью аппликат, а тройка чисел (x, y, z), соответствующая трем проекциям на оси будет являться декартовыми координатами точки M в пространстве.



Пусть определена некоторая декартова прямоугольная система координат на плоскости. При этом имеются две точки с координатами (x_1,y_1) и (x_2,y_2) соответственно. Тогда, воспользовавшись теоремой Пифагора, можно определить расстояние между двумя этими точками. Для этого достаточно использовать формулу:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Аналогично, имея две точки в пространстве с координатами (x1,y1,z1) и (x2,y2,z2) , можно вычислить расстояние между ними по формуле:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

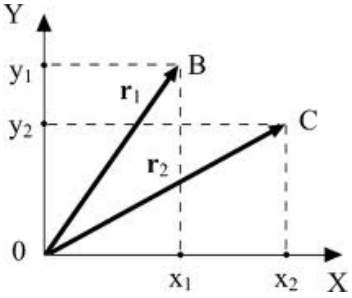
Отрезок в пространстве задается при помощи двух точек - концов. Геометрическим вектором будем называть упорядоченную пару точек, одна из которой будет являться его началом, а другая - концом. Начало вектора обычно именуют точкой приложения вектора. При этом длина

вектора будет равна математическому расстоянию между его началом и концом.

Если длина вектора равна 1 то такой вектор называют единичным.

Если точки начала и конца вектора совпадают, то такой вектор именуется нулевым вектором.

Если точка приложения вектора располагается точно в начале координат, то такой вектор именуют *радиус- вектором.* Радиус вектор характеризуется координатами своего конца и, в зависимости от контекста, может использоваться как для описания некоторой точки в пространстве, так и для задания направления (в компьютерной графике направление принято задавать при помощи единичных векторов).



Два вектора называются коллинеарными если они параллельны.

Три вектора называются компланарными, если все они лежат в одной плоскости.

Два вектора всегда лежат в одной плоскости, равно как и три точки.

Два вектора считаются равными тогда и только тогда, когда они коллинеарны, сонаправлены и имеют одну длину. Таким образом, на равенстве векторов не как не отражается точка приложения вектора, поэтому в машинной графике любой вектор принято задавать через равный ему радиус-вектор.

Линейными операциями над векторами называются операции сложения (вычитания) векторов, а так же умножения (деления) вектора на число (что эквивалентно его масштабированию, либо в случае деления - получению доли вектора).

Каждый вектор имеет обратный себе вектор. При сложении вектора и обратного ему результатом будет нулевой вектор. В таком случае операцию вычитания векторов можно заменить на операцию сложения

с обратным вектором. Для получения обратного вектора достаточно взять все координаты исходного с обратным знаком.

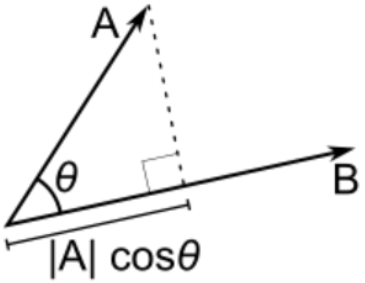
Для получения единичного вектора достаточно поделить вектор на свою длину (однако так как операция деления является более сложной по отношению к операции умножения, то ее заменяют на умножение на число обратное данному, т.е. единице, поделенной на данное число).

Скалярным произведением векторов является число, равное произведению длин этих векторов на косинус угла между ними:

$$\langle \mathbf{a}, \mathbf{b} \rangle = |\mathbf{a}| \cdot |\mathbf{b}| \cdot \cos \angle(\mathbf{a}, \mathbf{b})$$

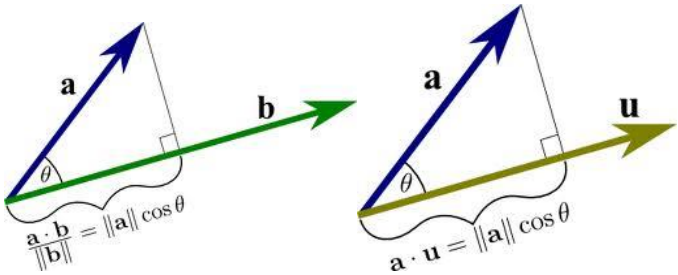
С другой стороны, скалярное произведение можно выразить через координаты векторов следующим образом:

$$\langle \mathbf{x}, \mathbf{y} \rangle = x_1 y_1 + x_2 y_2 + x_3 y_3$$



Скалярное произведение достаточно часто используется в компьютерной графике. Одним из примеров его использования является определение величины угла между векторами. Скалярное произведение обращается в 0 если вектора взаимно перпендикулярны, становится больше нуля, если угол между векторами острый, и меньше нуля - при тупом угле.

Так же одним из примеров использования скалярного произведения может послужить проецирование некого вектора на орт.



Проецирование

Растровое изображение, формируемое на экране, является ничем иным, как совокупностью растровых точек, расположенных в одной плоскости - плоскости монитора. Следовательно, любое изображение, формируемое с использованием компьютера так же должно быть плоским, а значит, нет никакой возможности вывести на экран компьютера действительное изображение трехмерных объектов.

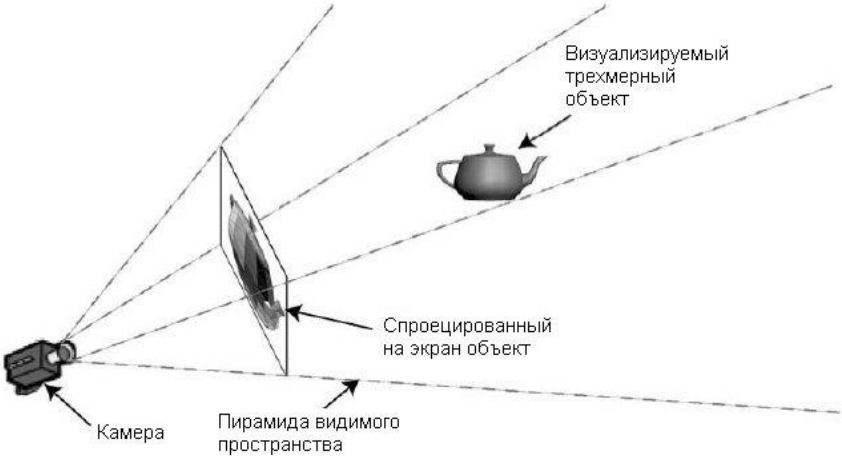
Зрение человека, в то же время, бинокулярно. Используя оба глаза, наша зрительная система способна сформировать и передать в соответствующий отдел головного мозга объемное изображение. Однако если использовать лишь один глаз, изображение так же станет плоским (в каком-то смысле).

Данная возможность появилась у человека благодаря специфике строения органа зрения. Фоторецепторы, располагающиеся в сетчатке человека, распределены в одном слое, который может быть принят за плоскость (каждый рецептор в такой плоскости может быть определен в двумерных координатах). Таким образом, можно провести параллель между плоским растровым изображением и изображением, формируемым в органе зрения человека.

Подобным образом формируется изображение и в фотоаппаратах, когда лучи света, отраженные от реальных трехмерных объектов, преломляясь в линзе, фокусируются на плоскости, создавая плоское изображение.

Подобный принцип был назван проецированием.

Проецирование - отображение точек из системы координат с некоторой размерностью в систему координат с меньшей размерностью.



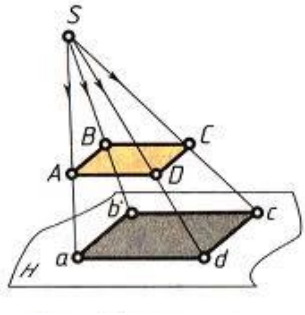
Проецирование некоторой точки осуществляется при помощи некоторой плоскости проекции, на которой определена система координат, в которой окажется спроецированная точка, а так же проектор - луч, испущенный из проецируемой точки, который при пересечении с плоскостью проекции (или плоскостью проецирования) даст конечный результат.

Все проекции по виду делятся на центральные и параллельные.

В центральных проекциях все проекторы сходятся в единой точке, называемой точкой схода. Разумеется, что угол, под которым они пересекают плоскость проекции, различен.

В параллельных проекциях все проекты направлены параллельно. Они входят в плоскость проекции под одним и тем же углом, а точка схода в таком случае может рассматриваться как максимально удаленная.

В случае параллельной проекции, когда угол между проекторами и плоскостью проекции составляет 90 градусов, такой вид проекции называется прямоугольной проекцией. Иначе - косоугольной.



Пытаясь растеризовать некоторых трехмерный объект на плоском экране можно пойти двумя путями:

Разработать алгоритмы растеризации объекта в трехмерном пространстве (в данном случае речь идет о поиске набора трехмерных точек), затем для каждой полученной трехмерной точки произвести проецирование. Данный способ нетривиален и разрешение задачи в таком случае становится крайне не очевидно (хотя, похожие методы существуют, популярны, но слишком медлительны, чтобы использоваться в реальном времени).

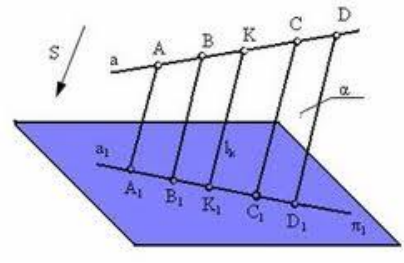
Описать объект посредством некоторых опорных вершин, затем, применяя только линейные преобразования спроецировать эти вершины на плоскость с последующей растеризацией объекта, построенного на базе этих вершин. Данный способ реализуется намного проще, работает эффективней, что делает его наилучшим для приложений реального времени. Однако он обладает и рядом недостатков. Ограничение на линейность преобразований не позволяет производить ряд «изощренных» проекций, таких как, к примеру, рыбий

глаз. Так же, зачастую описание объекта через вершины приводит к его «огрубению».

Геометрические примитивы. Точка

Точка является простейшим геометрическим примитивом. Проецирование точки описывалось выше.

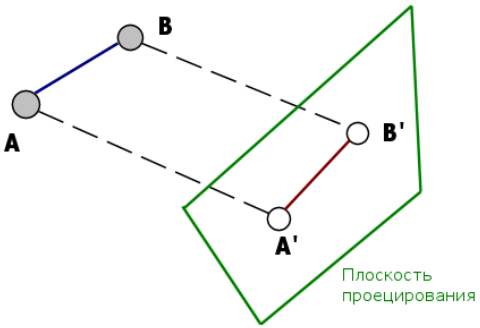
Единственным объектом, который можно описать при помощи точек является *список точек (Point List).*



Отрезок

Данный вид примитива часто используется в компьютерной графике при описании границ, направлений, путей, сеточных моделей и т.д. Часто в англоязычной терминологии используется термин линия (line), хотя, по сути, и подразумевается отрезок.

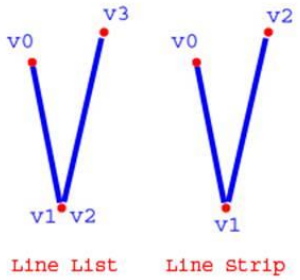
Для проецирования отрезка необходимо найти проекции его концов, которые будут использоваться при его растеризации:



На основе отрезков можно построить два наиболее часто встречающихся примитива: список отрезков (line list) и ломаную (line strip). В приложениях реального времени и тот и другой объект описываются одинаковым образом - как список вершин (концов отрезков). Для того, чтобы определить, какой из видов объекта используется вводят понятие типа топологии (в данном случае топологию следует понимать как способ связывания друг с другом вершин, или правило, в соответствии с которым две вершины считаются концами одного отрезка).

Для топологии типа Line List действует правило: каждый новый отрезок образуется каждой новой парой вершин.

Для топологии типа Line Strip действует правило: первый отрезок определяется первой парой вершин, каждый новый - последней вершиной предыдущего отрезка и новой вершиной.



Многоугольник

Многоугольник является плоской фигурой, т.е. все его вершины всегда находятся в одной плоскости. Часто в пространстве он описывается упорядоченным списком вершин, в котором каждые две вершины определяют ребро (в том числе и последняя с первой).

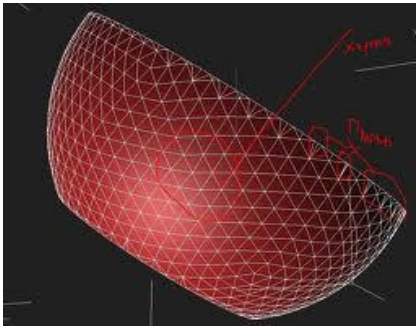
Для проецирования многоугольника на экран достаточно спроецировать каждую из его вершин.

Для отображения трехмерного прямоугольника на экране необходимо спроецировать каждую вершину, а затем, используя

изначальную взаимосвязь вершин, растеризовать плоской многоугольник.

По причине погрешности машинного представления вещественных чисел, не всегда удается расположить четыре и более вершин в одной плоскости. В тоже время малейшая погрешность может выродится в визуальный артефакт, равно как и в ошибку вычислений.

Поэтому на практике вместо многоугольников (в полном смысле этого слова) зачастую используют их частный случай – треугольники, так как три точки всегда будут находится в одной плоскости. Тогда каждый многоугольник (имеющий более, чем три точки) представляется в виде совокупности нескольких треугольников, разделяющих один и тот же набор вершин.



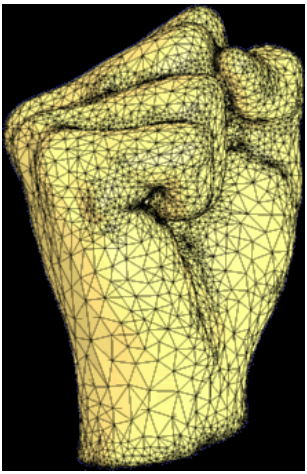
Полигональные сетки

Полигональные сетки являются объектами, в классическом понимании составленными из полигонов (или многоугольников). Однако из-за погрешности машинного представления вещественных чисел, было принято снизить количество вершин полигонов, составляющих сетку до трех (так как три точки всегда лежат в одной плоскости). В этом случае каждый многоугольник можно представить в качестве набора треугольников, разделяющих одни и те же вершины. Составляющий полигональную сетку треугольник принято называть гранью (face).

В англоязычной терминологии полигональные сетки известны как меши (mesh). Они повсеместно используются в приложениях графики

реального времени для описания практически всех поверхностей. Не смотря на то, что аппроксимация некоторых объектов при помощи набора граней приводит к их «огрубению», производительность систем, основанных на растеризации треугольников, более подходит для нужд реального времени, чем криволинейные поверхности, использующиеся в приложениях, ориентированных на качество.

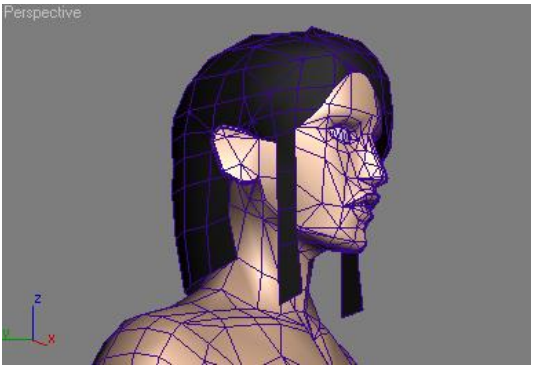
Ниже представлен один из примеров применения полигональной сетки для моделирования головы персонажа. Как не трудно заметить, подобный способ представления приводит к появлению некоторой «грубости» формы, однако качество можно повысить, увеличив общую детализацию.



Поверхности

Зачастую бывает сложно описать некую поверхность в пространстве аналитически. К примеру, практически невозможно описать формулой строение человеческого лица.

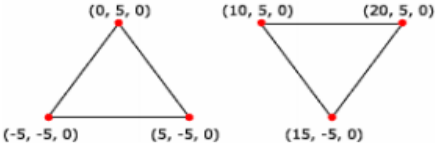
В то же время, любая объемная поверхность может быть представлена упрощенно в виде совокупность множества плоских граней (face).



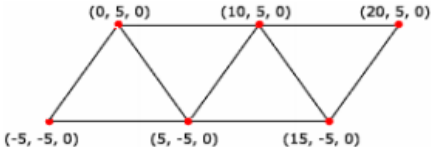
Подобный способ представления принято называть полигональной сеткой, либо мешью (mesh)

Как и в случае с отрезками, на базе граней можно задать несколько видов топологии. Наиболее часто встречающиеся из них перечислены далее:

Triangle List: каждые новые три вершины задают новый треугольник:

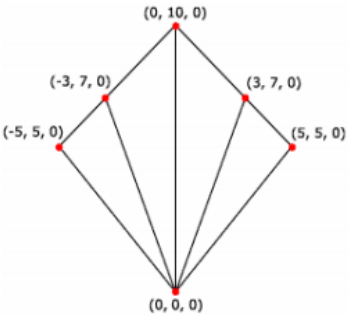


Triangle Strip: первые три вершины задают треугольник, каждый следующий треугольник задается при помощи новой вершины и двух последних вершин предыдущего треугольника. На практике получается подобие фермы:



Данная топология обладает меньшей избыточностью, однако далеко не все поверхности можно описать таким образом.

Triangle Fan: первые три вершины образуют первый треугольник, каждый новый треугольник образуется первой вершиной списка, последней вершиной предыдущего треугольника и новой вершиной:

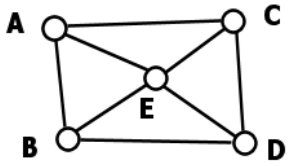


Индексируемые и неиндексируемые меши

Описанные выше меши, представляемые в виде списка вершин и характерным типом топологии, получили название простых или неиндексируемых мешей. Данный способ описания поверхностей достаточно прост, но в то же время обладает порядочной избыточностью. В современных приложениях реального времени напрямую используются ресурсы аппаратного обеспечения, такие как шины для передачи данных и видео-память, что налагает ограничения на объемы используемых данных. В сложившейся ситуации избыточность может оказаться неприемлемой.

Индексируемые меши

Предположим, следующую топологию, относящуюся к типу List.



Предположим, что она является частью более сложной топологии и не может быть представлена ни одним другим типом, кроме как List. Для представления 4х треугольников нам понадобится 12-ть вершин. Переводя данную информацию в байты мы будем иметь дело с 12*12 = 144 байт информации. В то же время реально нам достаточно лишь 5ти вершин, то есть – 60 байт информации. Целых 84 байта являются избыточными (более 140% избытка).

Дабы устранить подобную избыточность данных вводятся индексируемые меши.

Геометрические преобразования

Так как каждый объект, двумерный, либо трехмерный можно описать как совокупность опорных вершин, то само преобразование как раз и будет сведено к преобразованиям, производимым над этими вершинами, вернее над их координатами.

В дальнейшем мы не будем рассматривать случаи преобразования топологий, считая их неизменными. Кроме того, на данном этапе нами будут опущены преобразования, распространяющиеся на объект неравномерно.

Преобразования, производимые над объектом, будем считать линейными, либо аффинными. Для данного семейства преобразований будут действительны следующие свойства:

- прямые остаются прямыми;
- плоскости остаются плоскостями;
- параллельные прямые и параллельные плоскости остаются параллельными;
- отношения отрезков и площадей сохраняются;

Больше всего из подобного вида преобразований нас будут интересовать три: сдвиг, масштаб и поворот.

Сдвиг

Сдвиг (перенос) представляет собой приращение по каждой из координат на соответствующую величину.

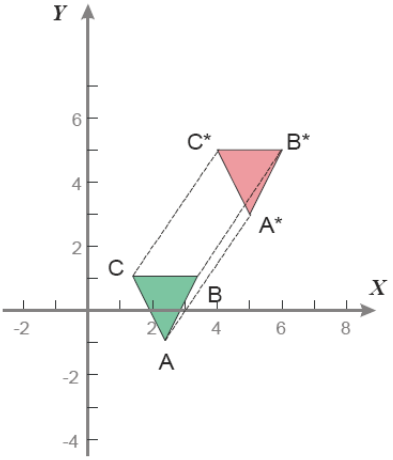
В скалярной форме сдвиг может быть описан как:

$$x_n = x + t_x$$

$$y_n = y + t_y$$

В векторной:

$$\vec{P}_n = \vec{P} + \vec{T}$$



$$\begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 150 \\ 0 & 1 & 150 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 220 \\ 220 \\ 1 \end{bmatrix} = \begin{bmatrix} 220 + 0 + 150 \\ 0 + 220 + 150 \\ 0 + 0 + 1 \end{bmatrix} = \begin{bmatrix} 370 \\ 370 \\ 1 \end{bmatrix}$$

Масштаб

Масштаб есть домножение соответствующей координаты на некоторое число. Операция масштабирования всегда производится относительно начала координат.

В скалярной форме она может быть записана как:

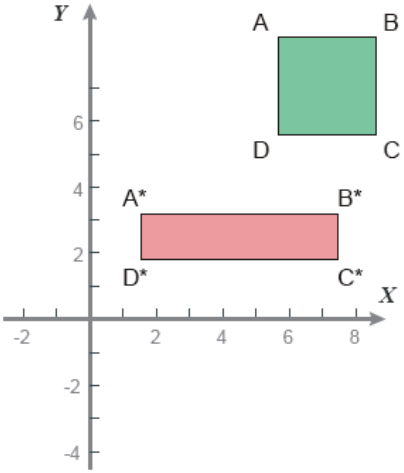
$$x_n = x \cdot S_x \qquad \begin{matrix} x = x_0 \cdot S_x \\ y = y_0 \cdot S_y \\ z = z_0 \cdot S_z \end{matrix}$$

$$y_n = y \cdot S_y$$

В матричной:

$$\vec{P}_n = \vec{P} \cdot \vec{S}$$

$$[x_n, y_n] = [x, y] \cdot \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix},$$



$$\underset{(m \times m)}{D} = \begin{pmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & d_m \end{pmatrix}, \qquad \underset{(m \times 1)}{X} = \begin{pmatrix} x_1 \\ x_2 \\ \\ x_m \end{pmatrix}$$

$$Y = D * X = \begin{pmatrix} d_1 x_1 \\ d_2 x_2 \\ \\ d_m x_m \end{pmatrix}$$ - наиболее общий случай масштабирования

Поворот

Поворот, как и масштаб всегда производится относительно начала координат.

В скалярной форме описывается:

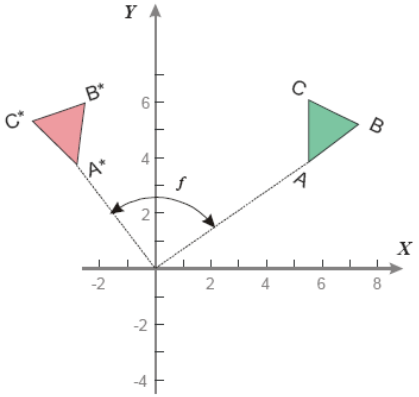
$$x_n = x \cdot \cos(f) - y \cdot \sin(f) \quad x = x_0 \cos\alpha - y_0 \sin\alpha$$

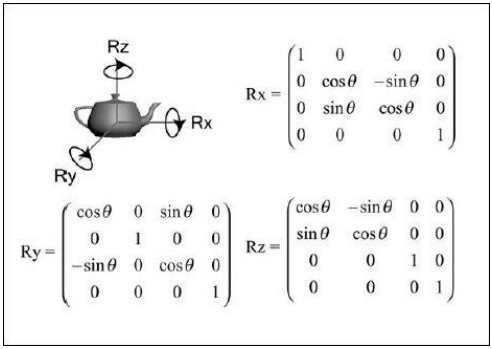
$$y_n = x \cdot \sin(f) + y \cdot \cos(f) \quad y = x_0 \sin\alpha + y_0 \cos\alpha$$

В матричной форме:

$$[x, y] = [x_0, y_0] \cdot \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix}$$

где *a* - угол поворота, отсчитываемый против часовой стрелки.





$$\vec{P}_n = \vec{P} \cdot \vec{R}$$

$$R = \begin{bmatrix} \cos f & \sin f \\ -\sin f & \cos f \end{bmatrix} = \begin{bmatrix} R_{11} & R_{11} \\ R_{21} & R_{22} \end{bmatrix}$$

Поворот в пространстве

В отличие от поворота в плоскости, поворот в пространстве представляет собой уже не движение по дуге, а движение по сфере. В этом случае сам поворот становится не очевиден. Для разрешения неопределенности принято производить поворот в пространстве

относительно некоторой оси, в качестве которой может быть выбрана *OX*, *OY* или *OZ*.

Преобразования вращения вокруг осей могут быть выведены из случая для плоскости с фиксацией одной из осей. Далее перечислены матрицы для вращения:

OX: $\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix}$ OY: $\begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{bmatrix}$ OZ: $\begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Для совершения поворота относительно произвольно направленной оси необходимо сначала произвести совмещение оси с одной из осей координат, затем повернуть на необходимый угол, после чего произвести преобразования обратные смещению.

Однородные координаты

Из-за различности форм описания преобразований, формирование суммарного преобразования затруднено. Для возможности приведения способов проведения преобразований к единой форме были введены однородные координаты: для плоскости [x, y, w] и для пространства [x, y, z, w].

Новая величина w расширяет координатную систему, увеличивая размерность. Для перехода от однородных координат к декартовым применяется следующая формула:

$$x_д = \frac{x}{w},\ y_д = \frac{y}{w},$$

и для пространства

$$z_д = \frac{z}{w}.$$

Для простоты, однородные координаты точки можно представить как декартовы координаты, отмасштабированные на коэффициент *w*. Обычно в компьютерной графике в качестве величины дополнительно коэффициента выбирают 1, что позволяет упростить процесс перехода между системами координат.

Таким образом, все вышеописанные преобразования приобретают единую форму. Для пространства они будут выглядеть следующим образом:

$$\begin{bmatrix} A & B & C & 0 \\ D & E & F & 0 \\ G & H & I & 0 \\ T_x & T_y & T_z & W \end{bmatrix}$$

Здесь вектор $[T_x, T_y, T_z]$ описывает сдвиг, подматрица

$$\begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix}$$

является либо матрицей масштабирования, либо матрицей поворота, а параметр W отвечает за общий масштаб и обычно выбирается равным 1.

Аналогично единая матрица преобразования определяется и для плоскости:

$$\begin{bmatrix} A & B & 0 \\ C & D & 0 \\ T_x & T_y & W \end{bmatrix}$$

Примечание: не смотря на то, данная матрица является единой для всех видов преобразований, сочетать различные виды преобразования в ней не рекомендуется. Каждое новое преобразование должно описываться новой матрицей.

Композиция преобразований

Результатом каждого произведенного преобразования является произведение вектора и матрицы, то есть так же вектор. Данный вектор может быть снова использован для проведения преобразований, и так далее. Таким образом, будет справедливо следующее:

$$P_1 = P_0 \cdot M_1$$
$$P_2 = P_1 \cdot M_2$$
$$P_3 = P_2 \cdot M_3$$

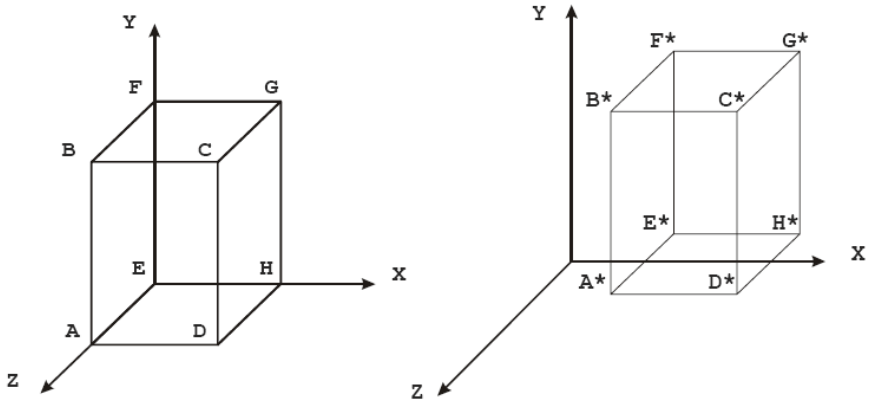
В то же время в силу дистрибутивности операции умножения матриц имеем:

$$P_3 = ((P_0 \cdot M_1) \cdot M_2) \cdot M_3 = P_0 \cdot (M_1 \cdot M_2 \cdot M_3)$$

Данное выражение говорит о том, что все преобразования, которые мы собираемся произвести над объектом, могут быть сведены к умножению всего одного вектора на одну матрицу, если предварительно все матрицы преобразования были последовательно перемножены. С одной стороны, мы явно не получаем прироста производительности, более того - количество операции увеличилось.

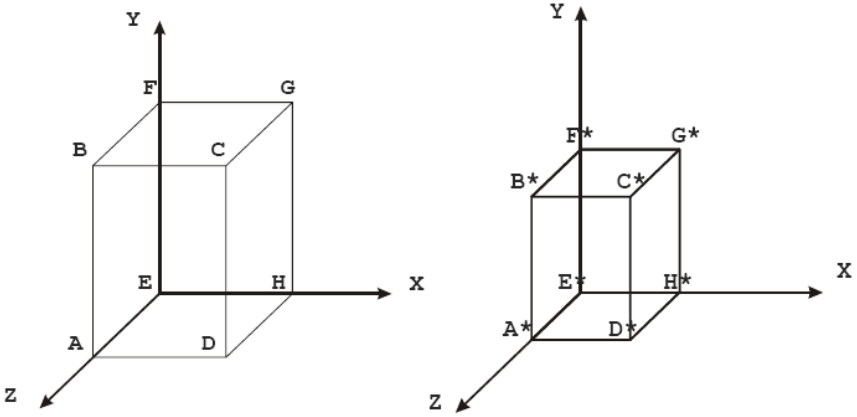
Однако в приложениях реального времени при проведении одной и той же последовательности преобразований над тысячами вершин, прирост экономии операций может стать существенной. Более того, композиция преобразований позволяет не сохранять все произведенные над объектом преобразования, а накапливать их в одной единственной матрице.

Сдвиг



$$T(Tx, Ty, Tz) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ Tx & Ty & Tz & 1 \end{bmatrix}.$$

Масштаб



$$S(Sx, Sy, Sz) = \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Повороты

$$Rz(f_z) = \begin{bmatrix} \cos f_z & \sin f_z & 0 & 0 \\ -\sin f_z & \cos f_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Rx(f_x) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos f_x & \sin f_x & 0 \\ 0 & -\sin f_x & \cos f_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

$$Ry(f_y) = \begin{bmatrix} \cos f_y & 0 & -\sin f_y & 0 \\ 0 & 1 & 0 & 0 \\ \sin f_y & 0 & \cos f_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

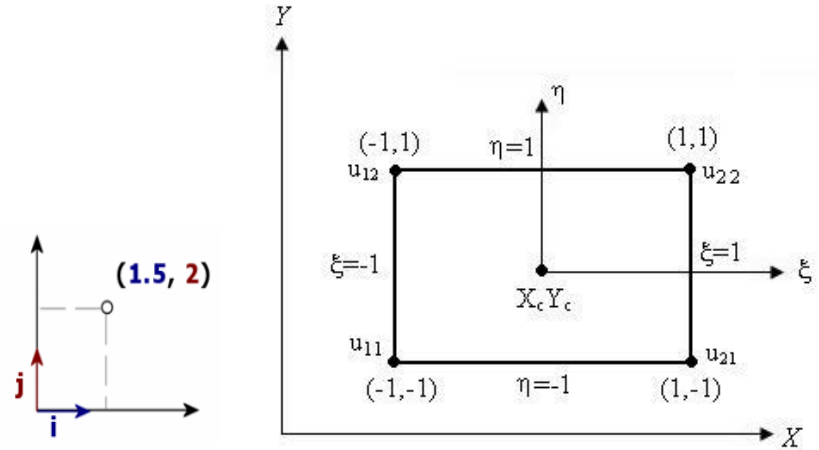
Глобальные и локальные системы координат

Любая ортогональная система координат описывается неким ортогональным базисом - системой линейно-независимых векторов. В двумерных координатах данная система будет состоять из двух векторов i и j, принятых за единичные. Тогда координаты любой точки в данной системе координат будут представлять из себя величины проекций на соответствующие вектора.

В то же время данная система координат может являться локальной по отношению к некой другой системе координат, называемой в таком случае глобальной.

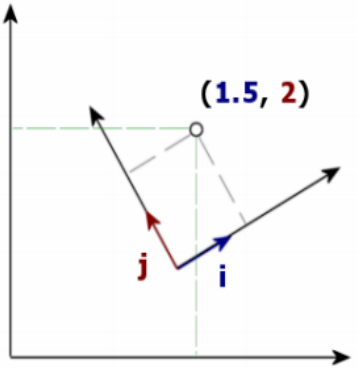
Локальная система координат может находится в любой позиции и ориентации относительно глобальной. В то же время, при изменении позиции либо ориентации локальной системы, локальные координаты объекта будут оставаться неизменными.

Для того чтобы преобразовать координаты некоторого объекта достаточно преобразовать его локальную систему координат, а затем вынести его координаты из локальной в глобальную систему.



В этом случае матрица преобразования как раз и описывает локальную систему координат объекта, а операция вынесения

координат из локальной системы в глобальную и есть умножение вектора на матрицу преобразования.



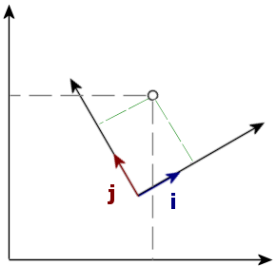
Обратная задача

Предположим, нам известно местоположение некой точки в пространстве относительно глобальной системы координат.

Предположим, нам дана некая локальная система координат, заданная матрицей преобразования. Как в таком случае определить координаты точки относительно этой локальной системы?

Для определения координат точки относительно локальной системы координат достаточно получить проекции этой точки на ее оси i и j (внести точку). Так как система координат задана матрицей, эта задача становится не настолько простой.

Однако в силу вступает замечательное свойство матриц. Для каждой (большинства применимых) матрицы существует матрица, обратная ей, такая, что если исходную матрицу умножить на обратную, результатом будет единичная матрица. Операция умножения вектора (координаты) на матрицу, обратную данной, эквивалентна внесению ее координат в ее (матрицы) локальную систему.



Конвейер преобразований

Зачастую случаются ситуации, когда одна и та же полигональная сетка может описывать до сотни одинаковых (клонируемых) объектов в пространстве, различающихся лишь своим положением и ориентацией (к примеру – стулья в аудитории).

Для того, чтобы правильно ориентировать и позиционировать клонированный объект в пространстве необходимо взять координаты всех вершин исходного (клонируемого) объекта (в нетронутом состоянии), а затем совершить над ними такую последовательность преобразований, что конечный клон примет необходимое положение и ориентацию.

Однако, хранить для каждого объекта полный набор трансформированных (преобразованных) вершин может быть накладно.

Вместо этого зачастую применяют так называемый конвейер преобразований (Transform Pipeline)

Модельная (мировая) матрица

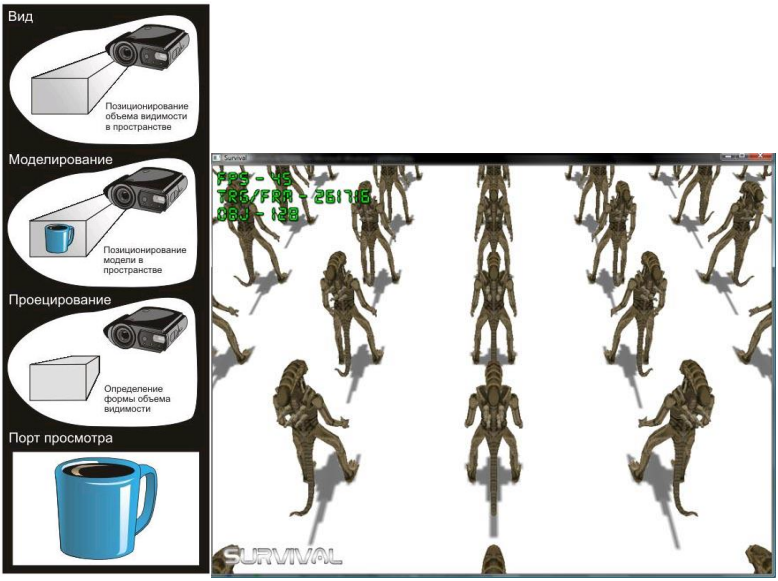
Основная задача модельной матрицы - сэкономить память для описания одинаковых объектов, присутствующих в различных частях сцены. Действительно, многие приложения реального времени, такие, как компьютерные игры, зачастую вынуждены использовать множество одинаковых по своей структуре полигональных объектов. При этом, с одной стороны, дублирование информации могло бы оказаться весьма избыточным. С другой - каждый объект характеризуется уникальной позицией и ориентацией в пространстве.

Для разрешения именно этой ситуации вводится модельная матрица (модельное (мировое) преобразование).

Изначально объект описывается в неком исходном положении - в центре своей собственной системы координат. Модельная матрица же представляет собой композицию преобразований, необходимых для того, чтобы позиционировать клонированный объект в конечную точку с заданной ориентацией и масштабом.

Таким образом, конечное положение i-ой вершины может быть вычислено как произведение исходной i-ой вершины и модельной матрицы. Необходимости хранить все преобразованные вершины нет. Они могут вычисляться непосредственно перед визуализацией.

Существует так же и иной подход к модельной матрице. Фактически, модельная матрица описывает не только композицию преобразований, но и положение, а также ориентацию, системы координат исходного объекта для данного клона. Она используется для того, чтобы вынести координаты конкретной вершины из исходного состояния - локальной системы координат в глобальную.



Видовая матрица

Матрица вида на логическом уровне ассоциируется с местоположением в пространстве и ориентацией некоторого наблюдателя. Как известно, наблюдатель может перемещаться по сцене абсолютно независимо от перемещения располагающихся на сцене объектов. Если преобразования, связанные с определением ориентации каждого объекта относительно наблюдателя сохранять в модельной матрице каждого объекта, то при перемещении самого наблюдателя матрицы пришлось бы пересчитывать, даже несмотря на тот факт, что сами объекты не претерпели никаких изменений.

Для обеспечения независимых изменений в различные системы координат вводится вторая матрица - видовая.

Для получения координаты точки на экране относительно наблюдателя необходимо внести ее координаты в локальную систему его координат. Поэтому, хотя матрица вида и описывает положение и ориентацию наблюдателя, на практике используется обратная ей матрица - осуществляющая преобразование внесения точки в систему координат.

Матрица проецирования

Матрица проецирования является последней, замыкающей конвейер преобразований. Фактически она описывает само преобразование проекции, т.е. получения координат пересечения проектора точки и плоскости проецирования. Данное преобразование получить не сложно, так как нам уже известны координаты точки в локальной системе наблюдателя, а также расстояние до плоскости проецирования, равно как и ряд других параметров.

Вычисления матрицы проецирования связаны с видом проекции, выбранным для отображения объектов. Существует не малое количество алгоритмов, позволяющих создавать как ортогональные, так и перспективные, и многие другие виды проекций.

Три вышеперечисленных матрицы преобразований применяются в качестве последовательной композиции преобразований для каждой вершины обрабатываемого в данный момент объекта, что позволяет сократить время вычислений до необходимого минимума.