Министерство науки и высшего образования Российской Федерации

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

А.Г. ЗАДОРОЖНЫЙ, М.Г. ПЕРСОВА,
Ю.И. КОШКИНА

ВВЕДЕНИЕ В ТРЕХМЕРНУЮ КОМПЬЮТЕРНУЮ ГРАФИКУ С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕКИ OPENGL

Утверждено
Редакционно-издательским советом университета
в качестве учебного пособия

НОВОСИБИРСК
2018

УДК 004.92(075.8)
З-156

Рецензенты:
канд. техн. наук, доцент *В.С. Карманов*
д-р техн. наук, профессор *М.Э. Рояк*

Работа подготовлена на кафедре прикладной математики НГТУ

Задорожный А.Г.

З-156 Введение в трехмерную компьютерную графику с использованием библиотеки OpenGL: учебное пособие / А.Г. Задорожный, М.Г. Персова, Ю.И. Кошкина. – Новосибирск: Изд-во НГТУ, 2018. – 100 с.

ISBN 978-5-7782-3744-5

В данном учебном пособии рассмотрены элементы теории из раздела вычислительной геометрии для работы с трехмерной компьютерной графикой и соответствующие функции графической библиотеки OpenGL. Пособие может быть рекомендовано как для самостоятельного изучения курсов «Компьютерная графика» и «Вычислительная геометрия», так и для подготовки к лабораторным, практическим и расчетно-графическим заданиям.

УДК 004.92(075.8)

ISBN 978-5-7782-3744-5

ВВЕДЕНИЕ

Суть *координатного метода* заключается в том, что для описания объектов используются пространственные координаты в различных системах координат с соответствующими преобразованиями из одной системы в другую. А поскольку объекты задаются множеством своих вершин, то эти преобразования применяются к вершинам объектов. На основании такого подхода было разработано множество алгоритмов вычислительной геометрии, которые и легли в основу компьютерной графики.

В данном учебном пособии кратко рассматриваются следующие элементы вычислительной геометрии:

- Системы координат и преобразования между ними;
- Аффинные геометрические преобразования (перемещение, поворот и растяжение);
- Геометрические проекции и проективные преобразования;
- Модельно-видовые преобразования;
- Кватернионы;
- Некоторые методы удаления невидимых линий и поверхностей.

Помимо этого рассматриваются и команды библиотеки OpenGL, которые позволяют создавать трехмерную графику без наложения текстур, но с использованием освещения по модели Фонга.

СИСТЕМЫ КООРДИНАТ

Система координат (СК) – это способ определения местоположения точки с помощью чисел, называемых *координатами*, что представляет собою реализацию метода координат. Обобщением системы координат являются системы отсчета и системы референции.

Для двух- и трехмерных систем приняты следующие названия осей: первая ось называется *осью абсцисс*, вторая – *осью ординат*, а третья – *осью аппликат*.

Вообще говоря, координаты можно вводить различными способами, но при решении конкретных математических или физических задач методом координат имеет смысл выбрать такую систему координат, которая упростила бы процесс решения.

Системы координат можно классифицировать по множеству критериев, в частности, по форме и направлению осей, по назначению, по точке отсчета, по размерности пространства и т.д.

По форме осей системы координат можно разделить на прямолинейные и криволинейные. В частности, декартова СК является прямолинейной системой, а вот биполярная СК – уже криволинейной, поскольку в качестве осей использует окружности.

По мерности пространства системы координат можно разделить на следующие классы:

- Двумерные (биангулярные, эллиптические, ...);
- Трехмерные (тороидальные, конические, ...);
- Многомерные (проективные, барицентрические, ...).

Также можно отдельно выделить и физические координаты, например:

- Координаты Риндлера;
- Координаты Борна;
- Небесные координаты;
- Географические и геодезические координаты;
- Ортодромические координаты.

БАЗОВЫЕ СИСТЕМЫ КООРДИНАТ

Самая простая и часто используемая система координат – это *прямоугольная система координат*, т.е. прямолинейная СК с взаимно перпендикулярными осями. Данная СК легко вводится для пространств любой размерности.

Прямоугольная система может быть описана набором *ортов* (единичных ортогональных векторов, сонаправленных с осями координат). В трехмерном случае орты обычно обозначаются векторами $\{i, j, k\}$ или $\{e_x, e_y, e_z\}$.

Прямоугольная система представляет собой частный случай косоугольной системы, которая, в свою очередь, является прямолинейной системой в аффинном пространстве. На рис.1 приведены двумерные примеры прямоугольной и косоугольной систем координат.

В основном из прямоугольных координат используются двумерные и трехмерные декартовы координаты. Из криволинейных же систем координат чаще всего используются полярные, цилиндрические и сферические координаты.

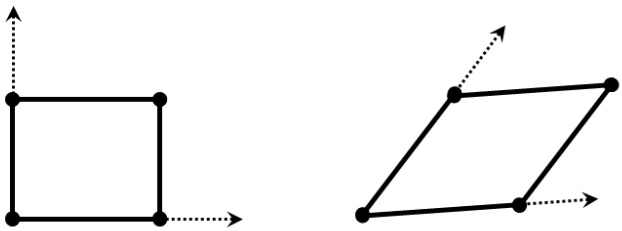

Рис.1. Прямоугольная и косоугольная двумерные системы координат.

ДЕКАРТОВЫ КООРДИНАТЫ

Если масштабы по осям одинаковы, тогда прямоугольная система называется *декартовой системой координат* (ДСК).

ДСК является *правосторонней* (правой, положительной, стандартной), если при повороте вокруг положительного направления оси OZ против часовой стрелки на 90° положительное направление оси OX совпадет с положительным направлением оси OY (правило буравчика, правой руки). В двумерном же случае у правосторонней системы ось OX направлена вправо, а OY – вверх. На рис.2 представлены различные варианты ДСК.

В частности, экранная система координат является правосторонней, поэтому ось OZ направлена от экрана, а не вглубь.

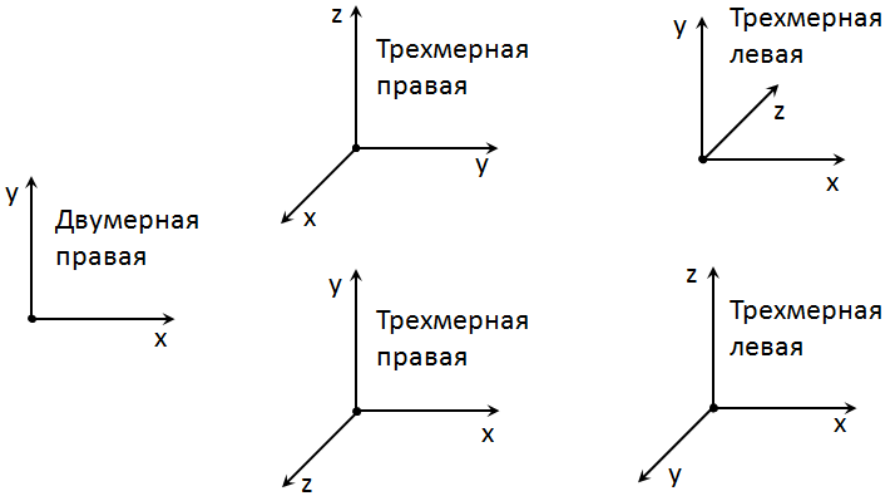

Рис.2. Варианты декартовой системы координат.

ПОЛЯРНАЫЕ КООРДИНАТЫ

В полярной системе координат (ПСК) положение любой точки определяется ее расстоянием до начала координат r и углом φ ее радиус-вектора к оси OX (рис.3). Радиус r называется *полярным радиусом* (радиальной координатой, угловым расстоянием). Угол φ называется *полярным углом* (угловой координатой, азимутом, позиционным углом).

Полярные координаты оказываются удобнее декартовых при задании кривых на плоскости и применяются, например, в астрономии и медицине.

Переход из ДСК в ПСК и обратно осуществляется по следующим формулам:

$$\left.\begin{aligned} r &= \sqrt{x^2 + y^2} \\ \varphi &= \operatorname{arctg}\left(\frac{y}{x}\right) \end{aligned}\right\} \Leftrightarrow \begin{cases} x = r \cdot \cos(\varphi) \\ y = r \cdot \sin(\varphi) \end{cases}.$$

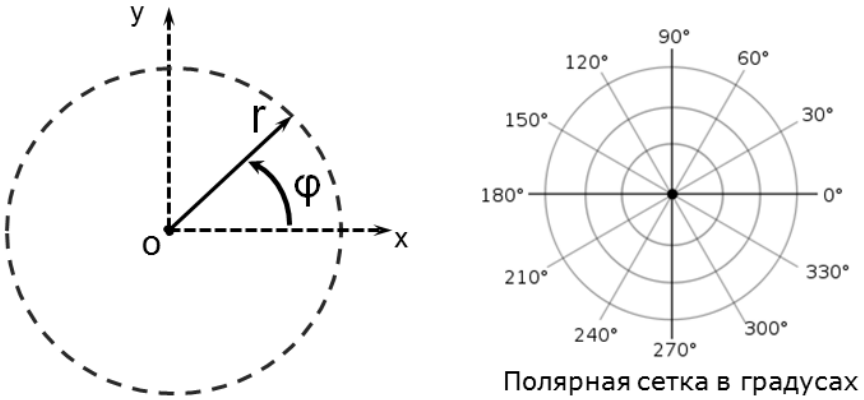

Рис.3. Полярная система координат.

ЦИЛИНДРИЧЕСКЫЕ КООРДИНАТЫ

Цилиндрической системой координат (ЦСК) называют трехмерную систему координат, являющуюся расширением ПСК путем добавления третьей координаты – z, которая задает высоту точки над плоскостью (рис.4). Цилиндрические координаты удобны при анализе поверхностей и объектов, симметричных относительно какой-либо оси (например, конусов и цилиндров).

Переход из ДСК в ЦСК и обратно осуществляется по следующим формулам:

$$\left.\begin{array}{l} r = \sqrt{x^2 + y^2} \\ \varphi = \operatorname{arctg}\left(\dfrac{y}{x}\right) \\ z = z \end{array}\right\} \Leftrightarrow \begin{cases} x = r \cdot \cos(\varphi) \\ y = r \cdot \sin(\varphi) \\ z = z \end{cases}.$$

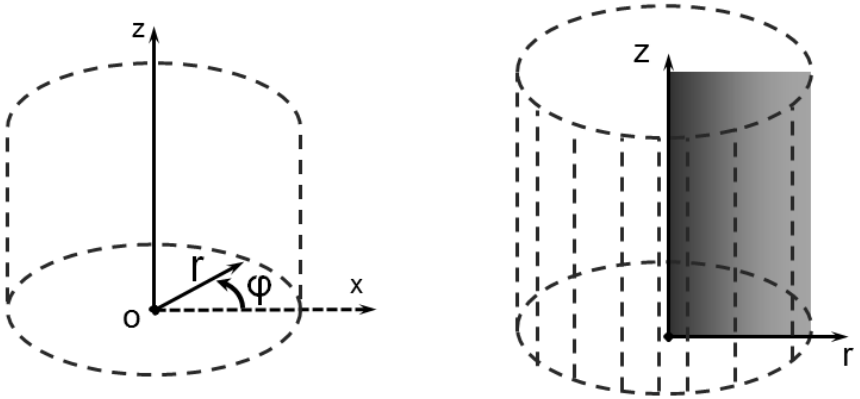

Рис.4. Цилиндрическая система координат.

СФЕРИЧЕСКЫЕ КООРДИНАТЫ

Сферической системой координат (ССК) называют трехмерную систему координат, являющуюся расширением ПСК путем добавления третьей координаты – угла θ (рис.5), который называется *зенитным углом* (полярным, нормальным углом).

Фундаментальная (основная) плоскость – это плоскость XY, в которой находится начало координат и отсчитывается полярный угол. Соответственно, зенитный угол отсчитывается от оси OZ, перпендикулярной к фундаментальной плоскости.

Переход из ДСК в ССК и обратно осуществляется по следующим формулам:

$$\left.\begin{aligned} \varphi &= \operatorname{arctg}\left(\frac{y}{x}\right) \\ r &= \sqrt{x^2+y^2+z^2} \\ \theta &= \operatorname{arctg}\left(\frac{\sqrt{x^2+y^2}}{z}\right) \end{aligned}\right\} \Leftrightarrow \begin{cases} x = r\cdot\cos(\varphi)\cdot\sin(\theta) \\ y = r\cdot\sin(\varphi)\cdot\sin(\theta) \\ z = r\cdot\cos(\theta) \end{cases}$$

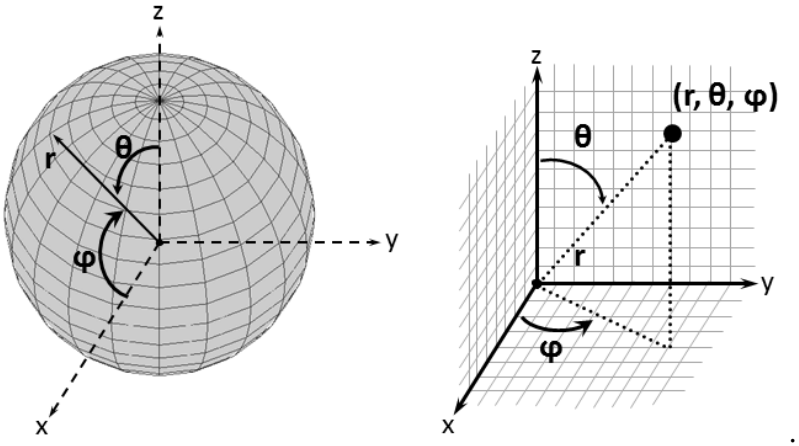

Рис.5. Сферическая система координат.

Особое распространение сферические координаты получили в астрономии и географии. В этом случае форма объектов рассматривается как *сфероид* (эллипсоид вращения) – поверхность вращения в трехмерном пространстве, образованная при вращении эллипса вокруг одной из трех его главных осей. Эллипсоид вращения является частным случаем эллипсоида, две из трех полуосей которого имеют одинаковую длину:

$$\frac{x^2}{a^2}+\frac{y^2}{a^2}+\frac{z^2}{b^2}=1.$$

В частном случае, когда все три полуоси равны, эллипсоид вырождается в сферу:

$$x^2+y^2+z^2=r^2.$$

Для удобства работы с поверхностью сфероидов ($r=1$) вводятся дополнительные определения.

Полюса – это точки, в которой ось вращения сфероида пересекается с его поверхностью. В зависимости от направления оси OZ выделяют северный ($\theta=0°$) и южный ($\theta=180°$) полюса.

Экватор – это окружность, являющаяся линией сечения поверхности сфероида плоскостью, которая проходит через центр сфероида перпендикулярно оси его вращения ($z=0$). *Параллель* – это окружность, являющаяся линией сечения поверхности сфероида плоскостью, параллельной плоскости экватора ($\theta=const$). Длины параллелей различны: они увеличиваются при приближении к экватору (самой длинной параллели) и уменьшаются до нуля к полюсам.

Меридиан – это окружность, являющаяся линией сечения поверхности сфероида плоскостью, проходящей через ось вращения сферы ($\varphi=const$) перпендикулярно плоскости экватора.

Часто используется понятие точки наблюдения на поверхности сфероида, соответственно, для этой точки наблюдателя вводятся понятия «надир» и «зенит». *Надир* – это направление вовнутрь сфероида, которое в рассматриваемой точке ортогонально касательной плоскости, проведенной через эту точку. *Зенит* – это направление, обратное к надиру. Фактически, для наблюдателя надир – это направление вниз (внутренняя нормаль), а зенит – направление вверх (внешняя нормаль).

ПЛАНЕТАРНЫЕ СИСТЕМЫ КООРДИНАТ

Строго говоря, планета Земля не имеет правильной геометрической формы, в связи с чем и было разработано множество систем координат и проекций для работы с картами Земли.

Фигура Земли получила название «геоид». *Геоид* – это выпуклая замкнутая поверхность, примерно совпадающая с поверхностью воды в морях и океанах в спокойном состоянии и перпендикулярная к направлению силы тяжести в любой ее точке. Геоид более точно, чем эллипсоид вращения, описывает уникальную форму планеты Земля, отражая свойства потенциала силы тяжести вблизи земной поверхности. На рис.6 приведен пример отличия геоида от эллипсоида вращения, а на рис.7 – форма геоида (https://ru.wikipedia.org/wiki/Геоид). Поскольку фигура геоида не имеет точного математического выражения и является практически неопределимой, поэтому нередко вместо геоида используется его приближение – *квазигеоид*. Квазигеоид, в отличие от геоида, однозначно определяется по результатам измерений и совпадает с геоидом на территории Мирового океана, незначительно отклоняясь в высоких горах.

В основном же форма Земля рассматривается как сфероид, размеры которого по Красовскому приведены в табл.1.

Таблица 1

Размеры эллипсоида Красовского

Параметр	Значение, км
Малая полуось (полярный радиус)	6356.863
Большая полуось (экваториальный радиус)	6378.245
Средний радиус сферы	6371.110
Длина меридиана	40008. 550
Длина экватора	40075.696

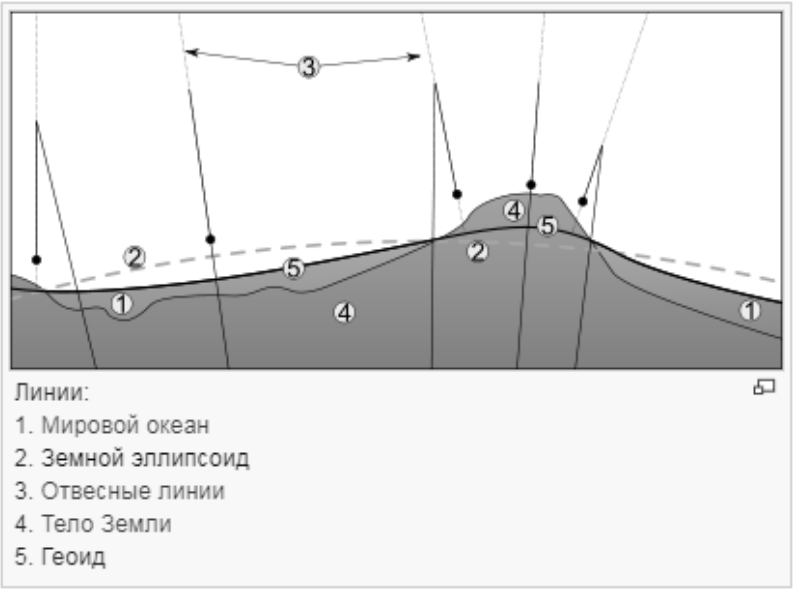

Рис.6. Пример отличия геоида от эллипсоида вращения.

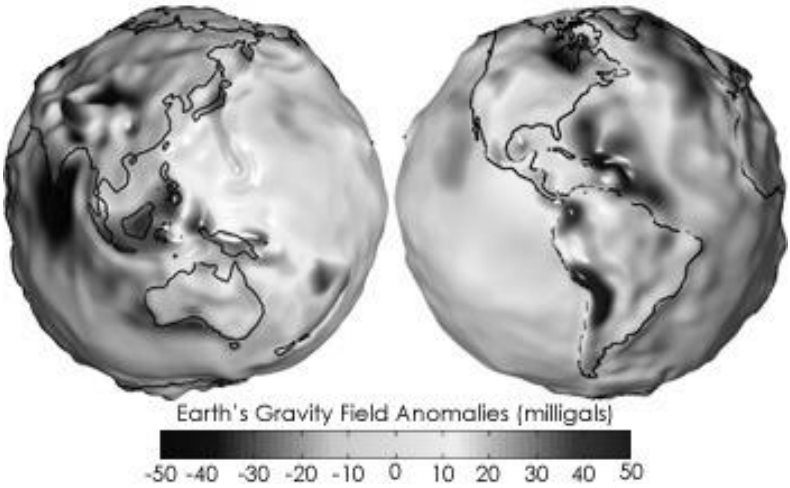

Рис.7.Форма геоида.

ГЕОГРАФИЧЕСКИЕ КООРДИНАТЫ

Географической системой координат называют систему координат, при которой положение точки на поверхности сферической Земли, определяется угловыми координатами – широтой и долготой, которые несколько отличаются от зенитного и полярного углов. Фундаментальной плоскостью является плоскость экватора.

Географическая широта – это угол $[-90°, 90°]$ между зенитом и плоскостью экватора. *Северной широтой* является широта точек $[0°, 90°]$, лежащих к северу от экватора, а *южной широтой* – широта точек $[-90°, 0°]$, лежащих к югу. Широтам $90°$ и $-90°$ соответствует Северный и Южный географические полюса.

Географическая долгота – это угол $[-180°, 180°]$ между плоскостью меридиана, проходящего через данную точку, и плоскостью нулевого меридиана, от которого ведется отсчет долготы. *Восточной долготой является* долгота точек $[0°, 180°]$ к востоку от нулевого меридиана, а *западной долготой* – долгота точек $[-180°, 0°]$ к западу. Обычно за нулевой меридиан принимают *Гринвичский меридиан*, проходящий через обсерваторию в Гринвиче, но, например, Международная служба вращения Земли (International Earth Rotation and Reference Systems Service) использует *международный опорный меридиан*, который сдвинут на 5.3" (102 метра) к востоку от Гринвического.

Долгота и широта позволяют однозначно определить положение точки на поверхности эллипсоида, но в трехмерном пространстве необходима третья координата – высота (радиус). Однако вместо расстояния до центра планеты (эллипса) обычно применяется *высота над уровнем моря*, отсчитываемая от поверхности эллипсоида (уровня моря в его спокойном состоянии). В России такой отметкой является Кронштадтский футшток, расположенный на уровне Балтийского моря.

Исторически, координаты (широта и долгота) записываются в градусах (°), минутах (′) и секундах (″) с десятичной дробью, а современный вариант записи – только в градусах в виде десятичной дроби:

- N 54°59′16″ (с.ш.), E 82°54′22″ (в.д.)
- $(54.987714,\ 82.906201)°$

ГЕОДЕЗИЧЕСКИЕ КООРДИНАТЫ

В зависимости от текущей задачи в геодезии помимо географических координат используются и другие системы координат.

Собственно, *геодезические координаты* аналогичны географическим и определяются тремя значениями: геодезическими широтой и долготой (которые совпадают с географическими широтой и долготой) и геодезической высотой (расстоянием до поверхности эллипсоида).

Также в геодезии применяются и прямоугольные *координаты зон Гаусса-Крюгера*, которые являются сохраняющими углы проекцией земного шара на плоскость. В этом случае, земной шар делится по долготе на 60 шестиградусных зон, каждая из которых имеет собственную систему прямоугольных координат XY (рис.8), с осевым меридианом посередине и экватором, проходящим горизонтально по центру. Для однозначного определения местоположения на территории всего земного шара перед значениями (x, y) ставится номер зоны.

На небольших участках местности могут использоваться полярная и прямоугольная системы координат. В полярной системе используются угол от северного направления меридиана и расстояние до точки. В прямоугольной системе считаются положительными северное направление оси OX и восточное направление оси OY.

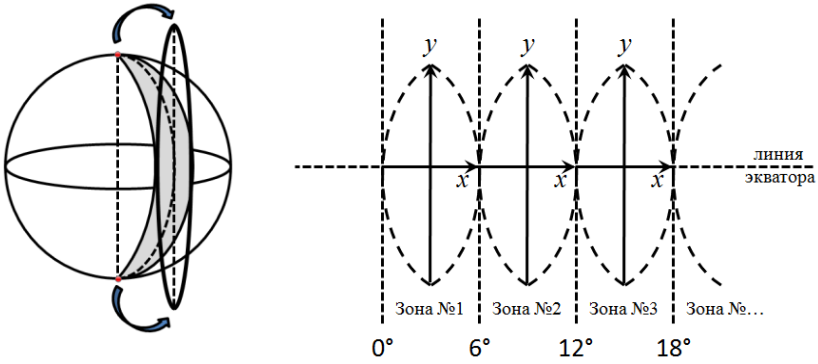

Рис.8. Координаты зон Гаусса-Крюгера.

ОРТОДРОМИЧЕСКИЕ КООРДИНТЫ

В картографии и навигации *ортодромия* – это дуга, являющаяся кратчайшим расстоянием между двумя точками на поверхности сферической Земли. Экватор и меридианы являются частными случаями ортодромии, при этом в общем случае ортодромии могут пересекать меридианы под разными углами.

В ортодромических системах осями являются две ортодромии, перпендикулярные друг другу в начале системы координат. В зависимости от того, как направлены эти оси и где расположено начало системы координат, различают две их группы: главноортодромические и частноортодромические.

В *главноортодромической* СК начало обычно размещается в исходном пункте маршрута, а одну из осей, называемую *главной ортодромией*, направляют вдоль маршрута так, чтобы она проходила через конечный пункт либо вблизи всех промежуточных пунктов (рис.9). В этом случае главная ортодромия является условным экватором новой ССК со своей широтой и долготой (хотя координаты могут выражатсья не только в градусах, но и в километрах).

В *частноортодромической* СК система координат задается для каждого участка маршрута.

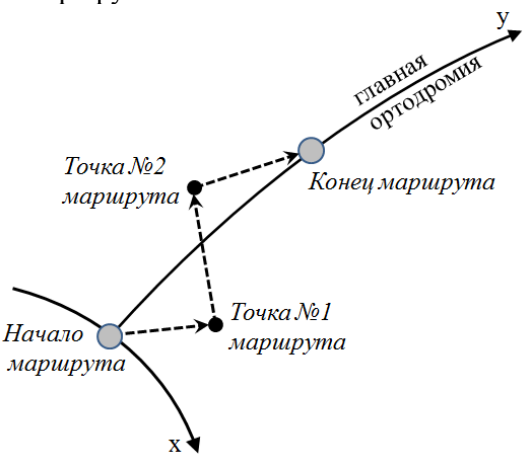

Рис.9. Главноортодромическая система.

НЕБЕСНЫЕ СИСТЕМЫ КООРДИНАТ

Система небесных координат используется в астрономии для описания положения точек на небесной сфере, соответственно координаты задаются только парой угловых величин. Системы небесных координат отличаются друг от друга выбором основной плоскости (экватора) и точкой начала отсчета. Расчеты положений светил на небесной сфере производятся с помощью небесной механики и сферической тригонометрии, составляя предмет сферической астрономии.

Небесная сфера – это сфера произвольного радиуса, на которую проецируются небесные тела, имеющие форму сфер. За центр принимается точка наблюдателя, а позиция объектов на сфере определяется соответствующими точками пересечения сферы и прямой, соединяющая центр сферы с центром тела. По аналогии с Землей, у небесной сферы есть небесный экватор, небесный меридиан, северный и южный полюса мира, ось мира (ось вращения сферы).

Для наблюдателя, находящегося на поверхности Земли вращение небесной сферы воспроизводит суточное движение светил на небе. Суточное вращение небесной сферы отсчитывается в пределах $[0°, 360°]$ или от $[0°, 180°]$ к западу и $[-180°, 0°]$ к востоку. Часто также используется и понятие «*склонение светила*» – это угол, который (в отличие от зенитного) отсчитывается от фундаментальной плоскости, соответственно, склонение отсчитываются в пределах $[0°, +90°]$ к северному полюсу и от $[0°, -90°]$ к южному полюсу.

В *горизонтальной системе координат* центром является позиция наблюдателя на поверхности Земли, а основной плоскостью является плоскость, касательная к сфере в этой точке, в результате чего зенит указывает на северный полюс мира. Координатами данной системы являются азимут и либо высота светила (склонение светила), либо зенитное расстояние (сферический зенитный угол). *Азимут* – это сферический полярный угол, азимуты отсчитываются в сторону суточного вращения небесной сферы.

В *первой экваториальной системе координат* основной плоскостью является плоскость экватора Земли. Координатами данной систе-

мы являются часовой угол и либо склонение светила, либо полярное расстояние (сферический зенитный угол). *Часовой угол* – это двугранный угол между плоскостями небесного меридиана и круга склонения светила, часовые углы отсчитываются в сторону суточного вращения небесной сферы (нередко этот угол отсчитывается не в градусах, а в часах: $[0^h, 24^h]$, либо $[0^h, 12^h]$ к западу и $[-12^h, 0^h]$ к востоку).

Вторая экваториальная система координат эквивалентна первой экваториальной за исключением часового угла, вместо которого используется *прямое восхождение светил* – угол между направлением на точку весеннего равноденствия и плоскостью круга склонения светила. Прямые восхождения отсчитываются в сторону, противоположную суточному вращению небесной сферы, в пределах $[0°, 360°]$ или $[0^h, 24^h]$ и является астрономическим эквивалентом земной долготы (нулевой отметкой является место на небе, где Солнце пересекает небесный экватор в весеннее равноденствие).

В *эклиптической системе координат* основной плоскостью является плоскость эклиптики (плоскость земной орбиты). Координатами данной системы являются эклиптическая широта и эклиптическая долгота. *Эклиптическая широта светила* – это, фактически, склонение светила. *Эклиптическая долгота светила* – это угол между направлением на точку весеннего равноденствия и плоскостью круга широты светила, долгота отсчитываются в сторону видимого годового движения Солнца по эклиптике, то есть с запада к востоку от точки весеннего равноденствия в пределах $[0°, 360°]$.

В *галактической системе координат* основной плоскостью является плоскость нашей Галактики. Координатами данной системы являются галактическая широта и галактическая долгота. *Галактическая широта* – это аналог склонения. *Галактическая долгота* – это угол между направлением на точку начала отсчета и плоскостью круга галактической широты светила. Точка начала отсчета находится вблизи направления на галактический центр, но не совпадает с ним в следствии небольшой приподнятости Солнечной системы над плоскостью галактического диска, ее выбирают таким образом, чтобы точка пересечения галактического и небесного экваторов с прямым восхождением 280° имела галактическую долготу 32.93192°.

ОДНОРОДНЫЕ КООРДИНАТЫ

Однородные координаты точки на плоскости – это тройки пропорциональных чисел (x', y', z'), не равные нулю одновременно, которые связаны с декартовыми координатами точки (x, y) простым соотношением:

$$(x, y) = \left(\frac{x'}{w}, \frac{y'}{w}\right),$$

причем число $w \neq 0$ часто называют *масштабным множителем*, а сама операция возвращения из однородных координат называется *нормализацией*. Соответственно, при $w = 1$ точке с координатами (x, y) сопоставляется точка с координатами $(x, y, 1)$, что позволяет легко получать однородные координаты.

Аналогично определяются однородные координаты точки в трехмерном пространстве: точке с координатами (x, y, z) ставится в соответствие точка с однородными координатами $(w \cdot x, w \cdot y, w \cdot z, w)$.

Фактически, однородные координаты – это специальная система координат, используемая в проективной геометрии, которая позволяет (в отличие от декартовой системы) естественным образом:

- описать бесконечно удаленную точку;
- провести различие между точками и векторами;
- использовать единое матричное представление для всех используемых геометрических преобразований.

Помимо этого, уравнения кривых линий и поверхностей, записанные в однородных координатах, часто имеют простой и симметричный относительно текущих координат вид.

НЕКОТОРЫЕ СВОЙСТВА

1. Из определения следует, что преобразование из однородных координат в декартовы является однозначным, а вот обратное преобразование таковым уже не является.

2. Если точка имеет однородные координаты (x, y, z), то эта же точка имеет и однородные координаты вида $(a \cdot x, a \cdot y, a \cdot z)$, где a – это любое действительное число, не равное нулю. Это свойство при правильно подобранном коэффициенте a позволяет упростить расчеты, например, перейти к целочисленной арифметике или избежать ситуации переполнения при работе с очень большими числами.

3. Переход к однородным координатам позволяет дополнить евклидову плоскость *несобственными* (бесконечно удаленными точками) вида $(x, y, 0)$, что приводит к понятию проективной плоскости в проективной геометрии. Например, бесконечно удаленной по оси Y точке (x, ∞) соответствует точка вида $(x, 1, 0)$, к которой уже можно математически применять различные геометрические преобразования.

4. В трехмерном пространстве с ортами $\vec{i}$, $\vec{j}$ и $\vec{k}$ любой вектор $\vec{v}$ можно представить в виде линейной комбинации базисных векторов, а любую точку P можно рассматривать как смещение на соответствующий вектор относительно начала координат:

$$\vec{v} = v_1 \cdot \vec{i} + v_2 \cdot \vec{j} + v_3 \cdot \vec{k}$$
$$P = O + p_1 \cdot \vec{i} + p_2 \cdot \vec{j} + p_3 \cdot \vec{k}.$$

При использовании же однородных координат вектор и точка примут простое однозначное представление:

$$\vec{v} = (v_1, v_2, v_3, 0)$$
$$P = (p_1, p_2, p_3, 1).$$

ГЕОМЕТРИЧЕСКИЙ СМЫСЛ

Рассмотрим прямую, проходящую через начало координат $(0,0,0)$ и точку $(x, y, 1)$. Эта прямая пересекает плоскость $z = 1$ в точке $(x, y, 1)$, которая однозначно и определяет точку (x, y) координатной плоскости XY.

Любая точка на этой прямой может быть задана тройкой чисел вида $(w \cdot x, w \cdot y, w)$. Данная тройка может быть интерпретирована как направляющий вектор из точки $(0,0,0)$ в точку $(w \cdot x, w \cdot y,\ w)$, которую и можно считать новыми координатами точки (x, y).

Таким образом, однородные координаты представляют собой вложение промасштабированной с коэффициентом w двумерной плоскости в плоскость $z = w$ трехмерного пространства (рис.10).

Другими словами, преобразование из однородных координат в эвклидовы эквивалентно проекции точки на плоскость $w = 1$ вдоль линии, соединяющей эту точку с началом координат.

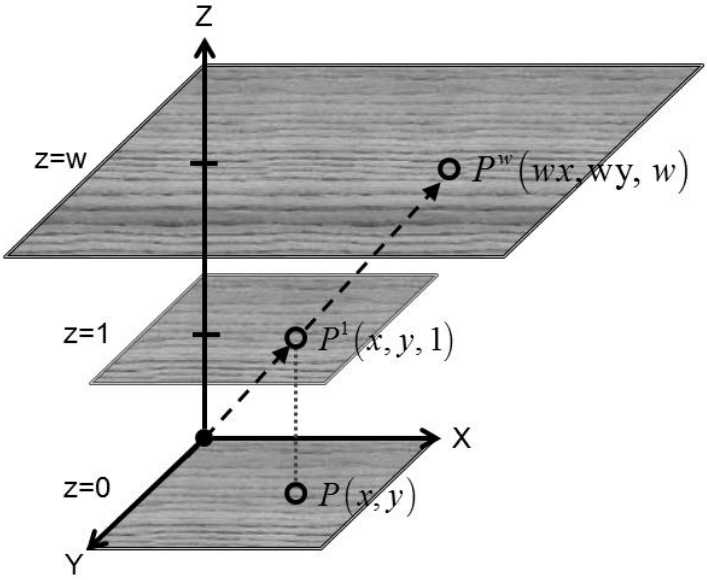

Рис.10. Геометрическая интерпретация однородных координат.

АФФИННЫЕ ПРЕОБРАЗОВАНИЯ

Аффинные преобразования – это такое отображение плоскости или пространства в себя, при котором параллельные, пересекающиеся и скрещивающиеся прямые таковыми и остаются. В частности, сохраняются отношения площадей и объемов, а точки, лежащие на одной прямой, после преобразования все также лежат на одной прямой.

К аффинным преобразованиям относятся операции сдвига, масштаба и поворота. Каждое из данных преобразований само по себе является коммутативным, однако их композиция уже не обладает свойством коммутативности (поскольку рассматриваемые преобразования меняют не сам объект, а систему координат, в которой он задается). Например, две операции сдвига можно выполнять в любом порядке (рис.11), в отличие от комбинации сдвига и поворота (рис.12).

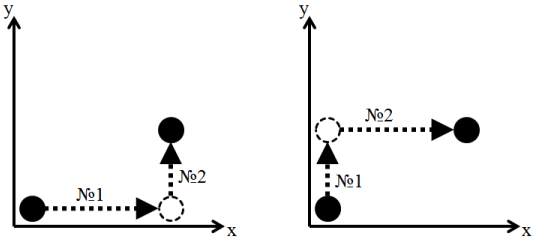

Рис.11. Пример коммутативности аффинных операций.

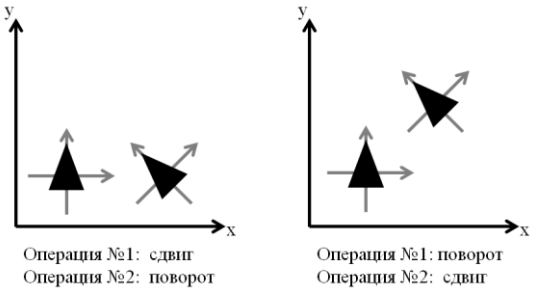

Рис.12. Пример некоммутативности аффинных операций.

ЭЛЕМЕНТАРНЫ ПРЕОБРАЗОВАНИЯ

В общем случае аффинное преобразование $\varphi(x) = P \cdot x + Q$ можно записать как матрицу перехода в однородных координатах:

$$\begin{bmatrix} \varphi(x) \\ 1 \end{bmatrix} = \begin{bmatrix} P & Q \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ 1 \end{bmatrix},$$

что позволяет использовать единую (общую) матрицу M, в которой заключены все комбинируемые преобразования.

Таким образом, процесс выполнения данных геометрических преобразований трехмерного пространства, переводящих точку x в точку x', состоит из следующих этапов:

1) Переход в однородные координаты путем добавления четвертой компоненты w:
$$(v_1, v_2, v_3, v_4)^T := (x_1, x_2, x_3, w)^T;$$

2) Выполнение преобразований путем умножения на соответствующую матрицу размерности 4x4:
$$(v_1', v_2', v_3', v_4')^T := M_{4\times4} \cdot (v_1, v_2, v_3, v_4)^T;$$

3) Возвращение в исходную систему координат и получение $\vec{x}'$:
$$(x_1', x_2', x_3')^T := \left(\frac{v_1'}{v_4'}, \frac{v_2'}{v_4'}, \frac{v_3'}{v_4'}\right)^T.$$

При необходимости выполнить последовательно два преобразования M^1 и M^2, результирующая матрица M получается путем умножения матрицы M^2 **слева** на матрицу M^1, поскольку второе преобразование применяется к результату первого, т.е. не в исходной СК:

$$\vec{y} = M^2 \cdot \left(M^1 \cdot \vec{x}\right) = \left(M^2 \cdot M^1\right) \cdot \vec{x} = M \cdot \vec{x}.$$

ПРЕОБРАЗОВАНИЕ СДВИГА

Двумерный сдвиг (перенос) **осей координат** на вектор $(dx, dy)^T$ осуществляется следующим образом:

$$\begin{cases} x' = x + dx \\ y' = y + dy \end{cases}.$$

В матричной форме это преобразование имеет вид:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}.$$

Для получения обратного преобразования (сдвига в обратном направлении) достаточно поменять знак у вектора сдвига, т.е. осуществить сдвиг в обратную сторону:

$$T^{-1}(dx, dy) = T(-dx, -dy).$$

Последовательный сдвиг по двум векторам $(dx_1, dy_1)^T$ и $(dx_2, dy_2)^T$ эквивалентен сдвигу по вектору, являющемуся суммой исходных векторов:

$$T(dx_1, dy_1) \cup T(dx_2, dy_2) = T(dx_1 + dx_2,\ dy_1 + dy_2).$$

Однородная матрица элементарного сдвига в трехмерном случае имеет следующий вид:

$$T(dx, dy, dz) = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

ПРЕОБРАЗОВАНИЕ МАСШТАБА

Двумерное масштабирование (растяжение и сжатие) **осей координат** на вектор $(kx, ky)^T$ осуществляется следующим образом:

$$\begin{cases} x' = x \cdot kx \\ y' = y \cdot ky \end{cases}.$$

В матричной форме это преобразование имеет вид:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} kx & 0 & 0 \\ 0 & ky & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}.$$

Если **коэффициенты масштабирования не равны нулю**, то преобразование остается аффинным и для получения обратного преобразования достаточно отмасштабировать с обратными коэффициентами:

$$S^{-1}(kx, ky) = S\left(\frac{1}{kx}, \frac{1}{ky}\right).$$

Последовательное масштабирование по двум векторам $(kx_1, ky_1)^T$ и $(kx_2, ky_2)^T$ эквивалентно масштабированию по вектору, компоненты которого являются произведением компонент исходных векторов:

$$S(kx_1, ky_1) \cup S(kx_2, ky_2) = S(kx_1 \cdot kx_2,\ ky_1 \cdot ky_2)$$

Однородная матрица элементарного масштаба в трехмерном случае имеет следующий вид:

$$S(kx, ky, kz) = \begin{bmatrix} kx & 0 & 0 & 0 \\ 0 & ky & 0 & 0 \\ 0 & 0 & kz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

ПРЕОБРАЗОВАНИЕ ПОВОРОТА

В двумерном случае поворот **осей координат против часовой стрелки** на угол θ **вокруг центра правосторонней** ДСК описывается следующими преобразованиями:

$$\begin{cases} x' = x \cdot \cos\theta - y \cdot \sin\theta \\ y' = x \cdot \sin\theta + y \cdot \cos\theta \end{cases}.$$

В матричной форме это преобразование имеет вид:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix},$$

где $c = \cos\theta$, $s = \sin\theta$.

Фактически, данное преобразование описывает двумерный поворот вокруг оси OZ трехмерного пространства.

Для получения обратного преобразования достаточно поменять знак угла поворота (что эквивалентно повороту в обратную сторону) или же просто транспонировать матрицу:

$$R^{-1}(\theta) = R(-\theta) = R^T(\theta).$$

Последовательный поворот по двум углам θ_1 и θ_2 эквивалентен повороту на угол, являющемуся суммой исходных углов:

$$R(\theta_1) \cup R(\theta_2) = R(\theta_1 + \theta_2).$$

Поворот в трехмерном пространстве удобно представить в виде комбинации *элементарных поворотов* (поворотов по каждой из осей).

Однородные матрицы элементарных поворотов в этом случае имеют следующий вид:

$$R_x(\theta)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & c & -s & 0\\ 0 & s & c & 0\\ 0 & 0 & 0 & 1\end{bmatrix},$$

$$R_y(\theta)=\begin{bmatrix}c & 0 & -s & 0\\ 0 & 1 & 0 & 0\\ s & 0 & c & 0\\ 0 & 0 & 0 & 1\end{bmatrix},$$

$$R_z(\theta)=\begin{bmatrix}c & -s & 0 & 0\\ s & c & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}.$$

Общую матрицу поворота на угол θ вокруг **единичного** вектора $\vec{v}$ можно получить путем последовательного поворота по каждой из осей. Соответствующая матрица имеет вид:

$$R\left(\theta,\vec{v}\right)=\begin{bmatrix}c+b\cdot v_1^2 & b\cdot v_1\cdot v_2-s\cdot v_3 & b\cdot v_1\cdot v_3+s\cdot v_2 & 0\\ b\cdot v_1\cdot v_2+s\cdot v_3 & c+b\cdot v_2^2 & b\cdot v_2\cdot v_3-s\cdot v_1 & 0\\ b\cdot v_1\cdot v_3-s\cdot v_2 & b\cdot v_2\cdot v_3+s\cdot v_1 & c+b\cdot v_3^2 & 0\\ 0 & 0 & 0 & 1\end{bmatrix},$$

где $b=1-c$.

При большом количестве последовательных поворотов происходит существенное накопление ошибки. Например, результат последовательности из 1440 поворотов на один и тот же угол 0.25° (общий поворот составит 360°) исходного вектора с координатами $(9,5)^T$:

$$\begin{bmatrix}9\\5\end{bmatrix}\xrightarrow{\text{поворот на }360^\circ}\begin{bmatrix}8.977884\\5.039098\end{bmatrix}.$$

КОМБИНИРОВАННЫЕ ПРЕОБРАЗОВАНИЯ

Рассмотрим несколько примеров более сложных аффинных преобразований с использованием однородных координат, в частности, композицию «поворот + сдвиг», зеркальное отражение, поворот вокруг точки.

ЗЕРКАЛЬНОЕ ОТРАЖЕНИЕ.

Различные варианты отражений выполняются через операцию элементарного поворота. Отражения обычно выполняются вокруг какой-то оси или центра СК.

Отражение точки (a, b) относительно начала координат может быть выполнено следующим преобразованием:

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -a \\ -b \\ 0 \\ 1 \end{bmatrix}.$$

Отражение точки (a, b) относительно плоскости $x = 0$ (для двумерного случая это эквивалентно отражению относительно оси OY) имеет вид:

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -a \\ b \\ 0 \\ 1 \end{bmatrix}.$$

ПОВОРОТ И СДВИГ ТРЕУГОЛЬНИКА.

Рассмотрим ситуацию, когда требуется сдвинуть треугольник ABC на 2 по оси OY и повернуть его на угол 90° вокруг оси OX, координаты треугольника следующие:

$$A=(0,0)$$
$$B=(0,3).$$
$$C=(5,3)$$

Треугольник имеет смысл перевести в однородные координаты и представить в виде прямоугольной матрицы так, чтобы столбцы этой матрицы соответствовали вершинам треугольника:

$$\begin{bmatrix} 0 & 0 & 5 \\ 0 & 3 & 3 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}.$$

В этом случае получится умножение матрицы на матрицу: матрица преобразований будет умножаться на матрицу координат треугольника, в результате чего получится матрица, содержащая новые координаты треугольника. Такой подход очень эффективен с точки зрения конвейерной обработки, если перейти к работе с транспонированными матрицами (в связи с этим в литературе часто приводятся уже транспонированные матрицы преобразований).

При комбинации «сдвиг + поворот» получается следующее преобразование:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 5 \\ 0 & 3 & 3 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 5 \\ 2 & 2 & 2 \\ 0 & 3 & 3 \\ 1 & 1 & 1 \end{bmatrix}.$$

При комбинации «поворот + сдвиг» получается преобразование, не равное предыдущему:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 5 \\ 0 & 3 & 3 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 5 \\ 0 & 0 & 0 \\ 2 & 5 & 5 \\ 1 & 1 & 1 \end{bmatrix}.$$

ПОВОРОТ ВОКРУГ ТОЧКИ

Поворот вокруг точки (a, b, c) на угол ϑ состоит из следующей последовательности элементарных операций:

1) Переход в новую СК, где центром станет точка (a, b, c) путем переноса на вектор $(-a, -b, -c)$;
2) Поворот на угол ϑ вокруг центра новой СК;
3) Возвращение в исходную СК путем обратного переноса на вектор (a, b, c).

Например, поворот на $90°$ точки $(2,1)$ вокруг точки $(3,4)$ осуществляется следующими матричными преобразованиями через однородные координаты:

$$\begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -3 \\ 0 & 1 & 0 & -4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 3 \\ 0 \\ 1 \end{bmatrix}.$$

ПОИСК ПЕРЕСЕЧЕНИЯ ДВУХ ПРЯМЫХ

Также через однородные координаты можно реализовать и операцию нахождения точки пересечения двух прямых. В этом случае процесс состоит из следующих моментов:

1) Уравнения прямых приводятся к виду:

$$\begin{cases} a_1 x + b_1 y + c_1 = 0 \\ a_2 x + b_2 y + c_2 = 0 \end{cases};$$

2) Из коэффициентов прямых составляется матрица вида:

$$\begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ 0 & 0 & 1 \end{bmatrix};$$

3) Условие пересечения двух прямых имеет вид:

$$\begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix};$$

4) Поиск точки пересечения через обратную матрицу:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

Например, точка пересечения двух прямых $y = 3 - 2x$ и $4x = 8y$ ищется следующим образом:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} -2 & -1 & 3 \\ 4 & -8 & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 6/5 \\ 3/5 \\ 1 \end{bmatrix}.$$

УГЛЫ ЭЙЛЕРА

Углы Эйлера – это углы, описывающие поворот в трехмерном евклидовом пространстве. В этом случае матрица произвольного поворота представляется в виде произведения трех матриц (R_x, R_y, R_z), каждая из которых зависит только от одного параметра – угла поворота по соответствующей оси.

При работе с углами Эйлера рассматриваются две прямоугольные системы координат, имеющие общий центр: начальная (исходная, подвижная) система XYZ и конечная (результирующая, неподвижная) система $\acute{X}\acute{Y}\acute{Z}$. В классическом варианте (вращение гироскопа) поворот состоит из следующих трех последовательных этапов (рис.13):

1. Переход в систему $\bar{X}\bar{Y}Z$: поворот на угол α (φ, *угол прецессии*) вокруг оси Z, после чего ось X окажется в плоскости $\acute{X}\acute{Y}$. Такое совмещение произойдет по линии N (*линии узлов*) пересечения плоскостей XY и $\acute{X}\acute{Y}$.

2. Переход в систему $\bar{X}\bar{\bar{Y}}\acute{Z}$: поворот на угол β (θ, *угол нутации*) вокруг оси $\acute{X}$, после чего оси Z и $\acute{Z}$ совпадут. При этом ось $\acute{Y}$ окажется в плоскости $\acute{X}\acute{Y}$.

3. Переход в систему $\acute{X}\acute{Y}\acute{Z}$: поворот на угол γ (ψ, *угол собственного вращения*) вокруг оси $\acute{Z}$, после чего оси $\bar{X}$ и $\bar{\bar{Y}}$ совпадут с соответствующими осями $\acute{X}$ и $\acute{Y}$.

Результирующая матрица поворота собирается из однородных матриц элементарных поворотов следующим образом:

$$R = R_z(\gamma) \cdot R_x(\beta) \cdot R_z(\alpha).$$

Как можно видеть, такие повороты некоммутативны (конечное положение системы координат зависит от порядка, в котором совершаются повороты).

Рассмотренная последовательность поворотов называется ZXZ (по названиям осей) или 3-1-3 (по номерам осей). Но также возможно еще 5 вариантов: XYX, XZX, YXY, YZY и ZYZ.

Если все три оси вращении разные, тогда соответствующие углы поворота еще называют *углами Тэйт-Брайана*, соответственно, будет также 6 вариантов поворотов: ZYX, YZX, XZY, ZXY, YXZ и XYZ.

В зависимости от того, вращаются ли оси вместе с поворачиваемым объектом, углы поворота разделяют на *внутренние* (angles of intrinsic rotation) и *внешние* (angles of extrinsic rotation), при этом внешние углы комплементарны внутренним, но примененным в обратном порядке: например, внешним углам $(10°, 20°, 30°)$ соответствуют внутренние углы $(30°, 20°, 10°)$.

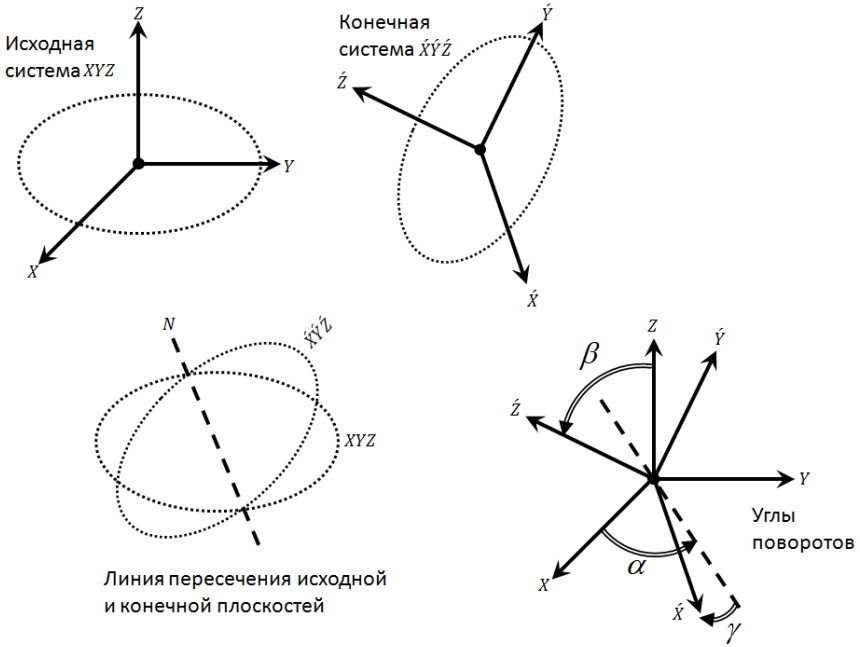

Рис.13. Поворот через углы Эйлера.

Углы Эйлера являются простым, естественным способом вращения объекта. Однако их использование при моделировании компьютерной анимации сталкивается с рядом трудностей:

- выбор определенной последовательности поворотов;
- сложность обеспечения плавности поворота;
- возможность попадания в ситуацию «Gimbal lock».

Ситуация «Gimbal lock» («Шарнирный замок», «Блокировка осей») возникает, когда в результате промежуточного поворота одна из осей совпадет со старой позицией другой оси. Например, для последовательности XYZ и угла $\beta = -90°$ получается следующая общая матрица поворота:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \cdot \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \cdot \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} 0 & 0 & 1 \\ \sin(\alpha+\gamma) & \cos(\alpha+\gamma) & 0 \\ -\cos(\alpha+\gamma) & \sin(\alpha+\gamma) & 0 \end{bmatrix}.$$

В результате после вращения вокруг оси OY на 90 градусов, ось OX совмещается с исходной осью OZ, и все последующие вращения происходят вокруг именно этой оси.

В этом случае необходимо поменять последовательность поворотов либо осуществлять повороты через кватернионы.

КВАТЕРНИОНЫ

Кватернионы – это система гиперкомплексных чисел, образующая векторное пространство размерностью четыре над полем вещественных чисел.

Кватернионы удобны для описания аффинных преобразований евклидовых пространств, поэтому и получили широкое распространение в механике и трехмерной графике (для реализации поворотов). Кватернионы реализованы в таких популярных графических пакетах, как DirectX, Uniti3D и 3D MAX. В сравнении с матрицами кватернионы обладают большей вычислительной устойчивостью и могут оказаться более эффективными.

Кватернион q представляет собою число вида

$$q = a + b \cdot i + c \cdot j + d \cdot k ,$$

где:

a, b, c, d – вещественные числа,

i, j, k – мнимые единицы.

В комплексном пространстве кватернион можно представить как пару комплексных чисел z_1 и z_2:

$$q = z_1 + z_2 \cdot j ,$$

где:

$z_1 = a + b \cdot i,$

$z_2 = c + d \cdot i.$

Кватернионы можно представить и в виде обычной вещественной матрицы следующего вида:

$$Q = \begin{bmatrix} a & -b & -c & -d \\ b & a & -d & c \\ c & d & a & -b \\ d & -c & b & a \end{bmatrix},$$

при этом сопряженному кватерниону соответствует транспонированная матрица Q^T, а определитель матрицы Q равен четвертой степени модуля кватерниона:

$$\det Q = |q|^4 .$$

В зависимости от ситуации, кватернионы можно записать различными способами, например, в виде вектор-строки:

$$q = (a, b, c, d) .$$

Также кватернионы удобно записывать в виде упорядоченной пары скаляра s и вектора $\vec{v} = (x, y, z)^T$:

$$q = [s, \vec{v}].$$

Кватернион $[s, \vec{0}]^T$ называется *чисто скалярным*, а $[0, \vec{v}]^T$ – *чисто векторным*. Как можно видеть, любое вещественное число можно представить в виде кватерниона с нулевой мнимой частью.

Кватернион называется *единичным* (нормированным), если его длина равна единице:

$$s^2 + (\vec{v}, \vec{v}) = 1.$$

Аргумент кватерниона – это угол в четырехмерном пространстве между кватернионом и вещественной единицей:

$$\arg(q) = arccos\left(\frac{s}{|q|}\right)$$

ОСНОВНЫЕ ОПЕРАЦИИ

Всего существует четыре базисных кватерниона:

$$\mathbf{1} = [1,0,0,0]$$
$$\mathbf{i} = [0,1,0,0]$$
$$\mathbf{j} = [0,0,1,0]$$
$$\mathbf{k} = [0,0,0,1]$$

Правила умножения базисных кватернионов приведены в табл.2. Как можно видеть, результат перемножения кватернионов не является коммутативным.

В табл.3 приведены основные операции над кватернионами. Из введенных таким образом операций следует ряд равенств:

$$\overline{q_1 q_2} = \overline{q_2}\,\overline{q_2},$$
$$|q_1 q_2| = |q_1|\,|q_2|.$$

Таблица 2

Таблица умножения базисных кватернионов $\mathbf{1}, \boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$

	1	i	j	k
1	1	i	j	k
i	i	-1	k	$-j$
j	j	$-k$	-1	i
k	k	j	$-i$	-1

Таблица 3

Операции над кватернионами

Операция		Результат
Сложение	$q_1 + q_2$	$[s_1 + s_2,\ \vec{v}_1 + \vec{v}_2]$
Умножение	$q_1 q_2$	$[s_1 s_2 - \vec{v}_1 \cdot \vec{v}_2,\ s_1\vec{v}_2 + s_2\vec{v}_1 + \vec{v}_1 \times \vec{v}_2]$
	αq	$[\alpha s, \alpha\vec{v}]$
	$q\overline{q}$	$\left[s^2 + \vec{v} \cdot \vec{v}, \vec{0}\right]$
Скалярное произведение	$q_1 \cdot q_2$	$\frac{\overline{q}_1 q_2 + \overline{q}_2 q_1}{2} = s_1 s_2 + \vec{v}_1 \cdot \vec{v}_2$
Векторное произведение	$q_1 \times q_2$	$\frac{q_1 q_2 - q_2 q_1}{2}$
Сопряжение	$\overline{q}$	$[s, -\vec{v}]$
Модуль	$\lvert q \rvert$	$\sqrt{q\overline{q}} = \sqrt{q \cdot q} = \sqrt[4]{detQ} = \sqrt{s^2 + \vec{v} \cdot \vec{v}}$
Норма	$\lVert q \rVert$	$\lvert q \rvert^2 = s^2 + \vec{v} \cdot \vec{v}$
Знак	$sgn\ q$	$\frac{q}{\lvert q \rvert}$
Обращение	q^{-1}	$\frac{\overline{q}}{\lvert q \rvert^2}$

ПРЕДСТАВЛЕНИЕ НА СФЕРЕ

Единичный кватернион можно записать в следующем виде:

$$q = [cos\theta,\ \ sin\theta \cdot \vec{e}\,],$$

где: $\vec{e} = \frac{1}{r}\vec{v}$ – единичный вектор,

$\theta = arg(s + i \cdot r)$,

$r = \sqrt{x^2 + y^2 + z^2}$.

Такой кватернион может быть представлен в виде ориентированной дуги экватора единичной сферы с центром в начале координат, причем, экватор строится перпендикулярно вектору $\vec{e}$ (рис.14). Дуга ориентирована относительно вектора $\vec{e}$ по правилу буравчика и описывает угол θ (соответственно, и ее длина равна θ). Для наблюдателя, находящегося в точке $p = (\vec{e}_x, \vec{e}_y, \vec{e}_z)$, представляющей кватернион, дуга направлена против часовой стрелки.

Соответственно, кватернион q^{-1} изображается как та же дуга, но только направленная в обратную сторону, а кватернион $\sqrt{q}$ – как половина исходной дуги.

Кватернион, образованный из вектора $\vec{e}$, представляется на единичной сфере как точка p и дуга с углом $\theta = \pi/2$.

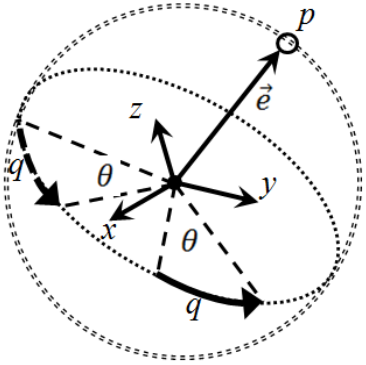

Рис.14. Преставление кватерниона на сфере.

ОПЕРАЦИЯ ПОВОРОТА

Поворот вектора $\vec{v}'$ на угол θ вокруг единичного вектора $\vec{v}$, в результате чего получается вектор $\vec{v}''$, рассчитывается по следующей формуле:

$$q'' = \mathrm{q}\mathrm{q}'\mathrm{q}^{-1},$$

где:

$q = \left[cos\left(\frac{\theta}{2}\right), sin\left(\frac{\theta}{2}\right)\vec{v}\right]$ – единичный кватернион поворота,

$q' = [0, \vec{v}']$ – кватернион исходного вектора $\vec{v}'$,

$q'' = [0, \vec{v}'']$ – кватернион конечного вектора $\vec{v}''$.

Пример кватерниона для поворота на угол -45° вокруг оси OZ:

$$q = \left[-\frac{\sqrt{2}}{2}, 0, 0, \frac{\sqrt{2}}{2}\right],$$

как можно видеть, скалярная компонента кватерниона отвечает за угол поворота.

Обычно все вращения выполняются для единичного кватерниона $[s, x, y, z]$, матрица поворота которого имеет вид:

$$R = \begin{bmatrix} 1-2\left(y^2+z^2\right) & 2\left(xy-sz\right) & 2\left(xz+sy\right) \\ 2\left(xy+sz\right) & 1-2\left(x^2+z^2\right) & 2\left(yz-sx\right) \\ 2\left(xz-sy\right) & 2\left(yz+sx\right) & 1-2\left(x^2+y^2\right) \end{bmatrix}.$$

Кватернион вращения фактически поворачивает оси, соответственно, умножение кватерниона q^2 на кватернион q^1 означает, что второе преобразование будет выполняться с уже измененными осями.

Выполнение нескольких поворотов эквивалентно выполнению одного поворота, кватернион которого равен произведению кватернионов соответствующих поворотов.

СВЯЗЬ С УГЛАМИ ЭЙЛЕРА

Кватернионы, соответствующие углам Эйлера, рассчитываются следующим образом:

$$q_a = \left[cos\left(\frac{a}{2}\right), \quad 0, \quad 0, \quad sin\left(\frac{a}{2}\right) \right],$$

$$q_\beta = \left[cos\left(\frac{\beta}{2}\right), \quad sin\left(\frac{\beta}{2}\right), \quad 0, \quad 0 \right],$$

$$q_\gamma = \left[cos\left(\frac{\gamma}{2}\right), \quad 0, \quad sin\left(\frac{\gamma}{2}\right), \quad 0 \right].$$

Кватернион, соответствующий суммарному повороту, считается через умножение кватернионов углов Эйлера:

$$q = q_\alpha q_\beta q_\gamma = \begin{bmatrix} cos\left(\frac{\beta}{2}\right) \cdot cos\left(\frac{\alpha+\gamma}{2}\right) \\ sin\left(\frac{\beta}{2}\right) \cdot cos\left(\frac{\alpha-\gamma}{2}\right) \\ sin\left(\frac{\beta}{2}\right) \cdot cos\left(\frac{\alpha-\gamma}{2}\right) \\ cos\left(\frac{\beta}{2}\right) \cdot cos\left(\frac{\alpha+\gamma}{2}\right) \end{bmatrix}.$$

В результате, углы Эйлера можно найти следующим образом:

$$\alpha = atan2\left(q_1 q_2 - q_3 q_4,\ q_1 q_3 + q_4 q_2\right),$$

$$\beta = arccos\left(q_1^2 - q_2^2 - q_3^2 + q_4^2\right),$$

$$\gamma = atan2\left(q_1 q_2 + q_3 q_4,\ q_1 q_3 - q_4 q_2\right).$$

ПЛАВНЫЙ ПОВОРОТ

Гиперсферическое пространство трехмерных вращений может быть охарактеризовано тремя углами Эйлера, однако любое такое представление начинает вырождаться на некоторых точках гиперсферы, что и приводит к ситуации «Gimbal lock». В сравнении с углами Эйлера, кватернионы позволяют проще комбинировать вращения и избегать эффекта блокировка вращения по одной из осей.

Кратчайший путь интерполяции между начальным и конечным положениями поворачиваемого объекта представляет собою дугу (вращение вокруг некой фиксированной оси). Фактически, поворот означает изменение по сфере положения объекта.

При интерполяции вращения через углы Эйлера интерполируются углы поворотов, однако при этом промежуточные положения поворачиваемого объекта представляют собой путь, который не является кратчайшим. Путь интерполяции в этом случае будет лежать на дуге только в частных ситуациях, например, когда все вращения происходят по оси, совпадающей с базовой.

У кватернионов же есть замечательное свойство: линейная комбинация двух кватернионов даст кватернион, который как раз и принадлежит плоскости вращения по кратчайшей дуге между этими кватернионами. На рис.15 приведен пример изменения осей при поворотах с помощью кватернионов и углов Эйлера.

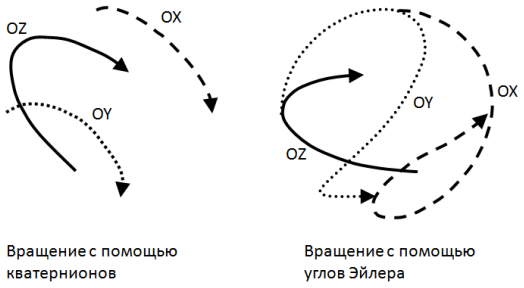

Рис.15. Пример изменения осей при повороте.

Реализация плавного поворота может быть совершена с помощью стандартного уравнения линейной интерполяции:

$$q^t = \alpha q^0 + \beta q^1,$$

где:

q^0 – начальный кватернион;

q^1 – конечный кватернион;

q^t – промежуточный кватернион;

$t \epsilon [0,1]$ – параметр интерполяции с некоторым шагом изменения;

$\alpha = \alpha(t)$ и $\beta = \beta(t)$ – коэффициенты интерполяции.

При простейшей интерполяции коэффициенты α и β выбираются следующим образом:

$$\alpha = 1 - t,$$

$$\beta = t.$$

Однако в этом случае длина исходных и получаемых кватернионов будет отличаться, а сферический угол поворота при этом интерполируется уже не линейно. Соответственно, при плавном изменении параметра интерполяции вращение будет идти неравномерно, хотя на практике для взгляда это малозаметно.

При сферической линейной интерполяции SLERP (spherical linear interpolation), которая приводит к получению единичных кватернионов, коэффициенты α и β выбираются следующим образом:

$$\alpha = \frac{\sin(w - wt)}{sin(w)},$$

$$\beta = \frac{sin(wt)}{sin(w)},$$

где $w = \arccos(q^0 \cdot q^1) = \arccos(s^0 s^1 + v^0 \cdot v^1)$.

Недостатком такого подхода является сложность, связанная с выполнением большого количества тригонометрических операций.

ПЛОСКИЕ ГЕОМЕТРИЧЕСКИЕ ПРОЕКЦИИ

Для перевода трехмерного изображения в двумерное, его нужно спроецировать на *картинную плоскость* (плоскость проекции). В классе *плоских геометрических проекций* (рис.16) проекция трехмерного объекта (представленного набором вершин) строится при помощи прямых проекционных лучей (*проекторов*), которые проходят через каждую вершину объекта и при пересечении картинной плоскости образуют проекцию. К данному этому классу не относятся многие картографические проекции и проекция типа «Рыбий глаз» (перспективная проекция на сферу).

Линии пересечения плоскостей проекций называются *осями проекций* (осями координат) и обычно обозначаются как OX, OY, OZ. Точка пересечения трех осей координат (точка O) является началом координат. Угол, образованный тремя плоскостями проекций, называется *координатным углом*.

В зависимости от того, проекторы параллельны или же имеют точку схода, разделяют *параллельные* и *центральные* (перспективные) проекции (рис.17).

В свою очередь, в зависимости от того, перпендикулярно ли направление проецирования картинной плоскости, параллельные проекции разделяют на *ортогональные* (прямоугольные, ортографические) и *косоугольные* (рис.18).

И, наконец, ортогональные проекции разделяются на *видовые* и *аксонометрические* – в зависимости от того, совпадает ли картинная плоскость с координатной.

Интересный факт. В связи с психологическими и физиологическими особенностями человеческого зрения, ближний план воспринимается в обратной перспективе, средний – в аксонометрической, а дальний – в прямой. Такая обобщая перспектива получила название *перцептивной*.

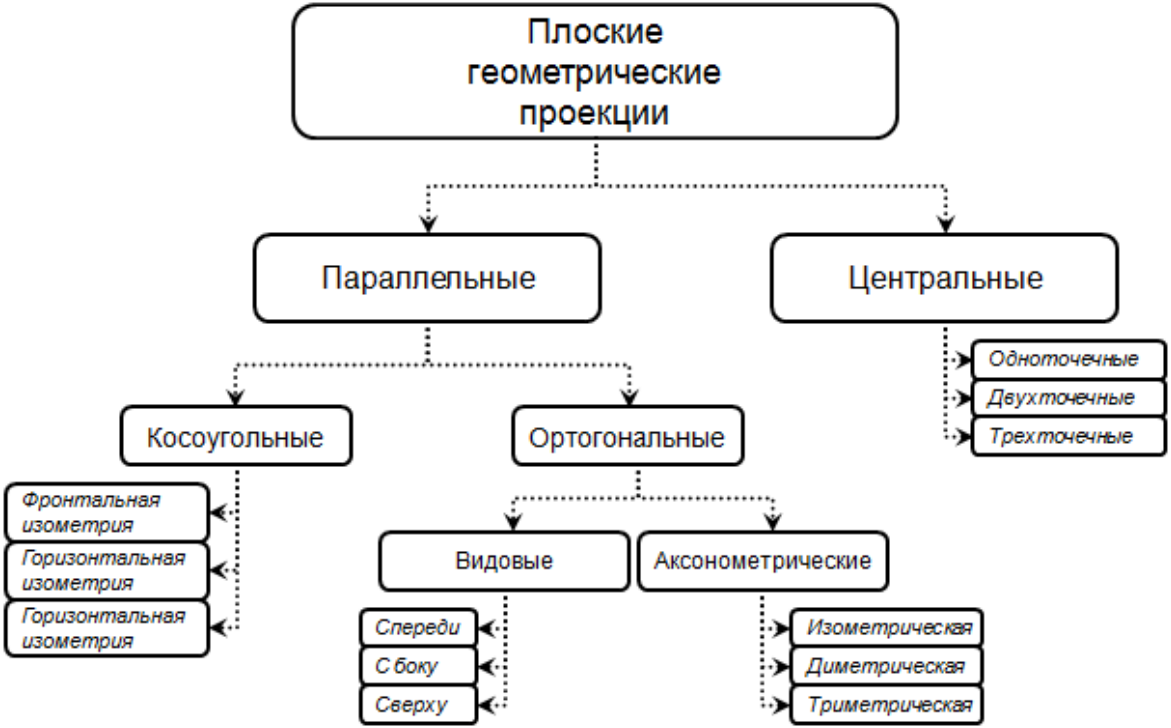

Рис.16. Иерархия плоских геометрических проекций.

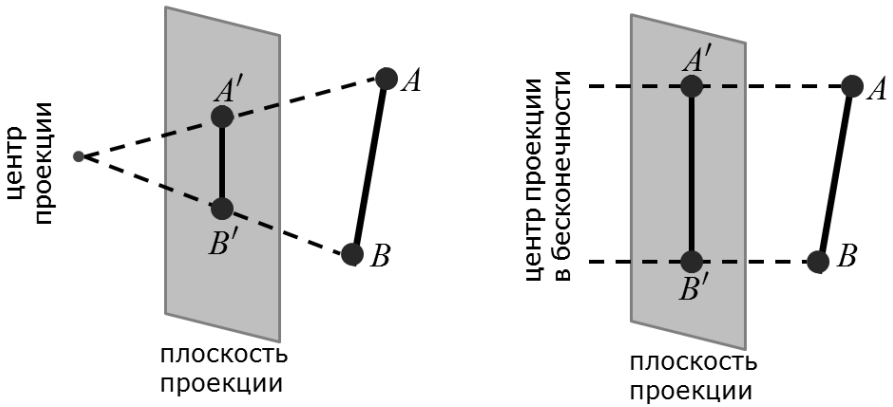

Рис.17. Проекции: центральная и параллельная.

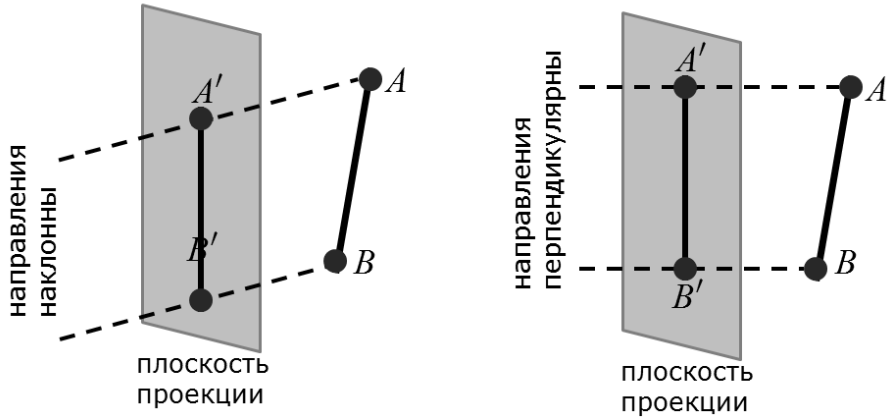

Рис.18. Параллельные проекции: косоугольная и ортогональная.

ПЕРСПЕКТИВНЫЕ (ЦЕНТРАЛЬНЫЕ) ПРОЕКЦИИ

В проекциях этого вида проекторы имеют точку схода, возможно, и не одну (рис.19). В большинстве случаев используется *одноточечная* проекция (с одной точкой схода), но в некоторых ситуациях используют проекции с двумя и тремя точками схода.

Вследствие такого способа проецирования происходит искажение пропорций и формы объектов при их визуальном восприятии, что соответствует с особенностями человеческого зрения (например, параллельные рельсы кажутся сходящимися в точку на горизонте).

Также в ряде случаев (например, в живописи) используется *обратная* перспектива, когда точка схода установлена в позицию наблюдателя. В этом случае размеры ближних объектов уменьшаются, а отдаленных – увеличиваются.

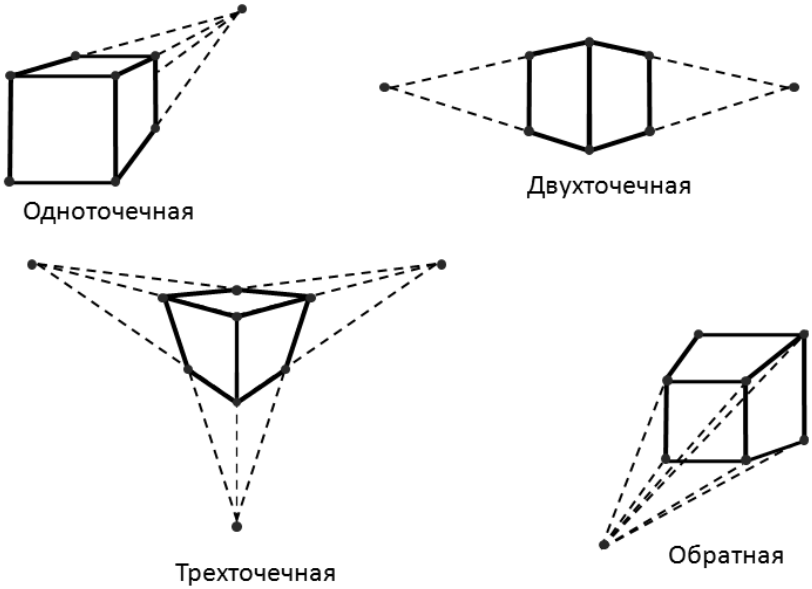

Рис.19. Перспективные проекции для куба.

ВИДОВЫЕ ПРОЕКЦИИ

Обычно *видовые проекции* (иногда их некорректно называют ортографическими) – это три стандартных вида (рис.20):

- вид спереди (проекция на фронтальную плоскость V),
- вид сверху (проекция на горизонтальную плоскость H),
- вид сбоку (проекция на профильную плоскость W).

Если спроецировать объект на 6 плоскостей (спереди, сзади, сверху, снизу, справа и слева) и развернуть их, тогда поученное изображение называется *эпюром*.

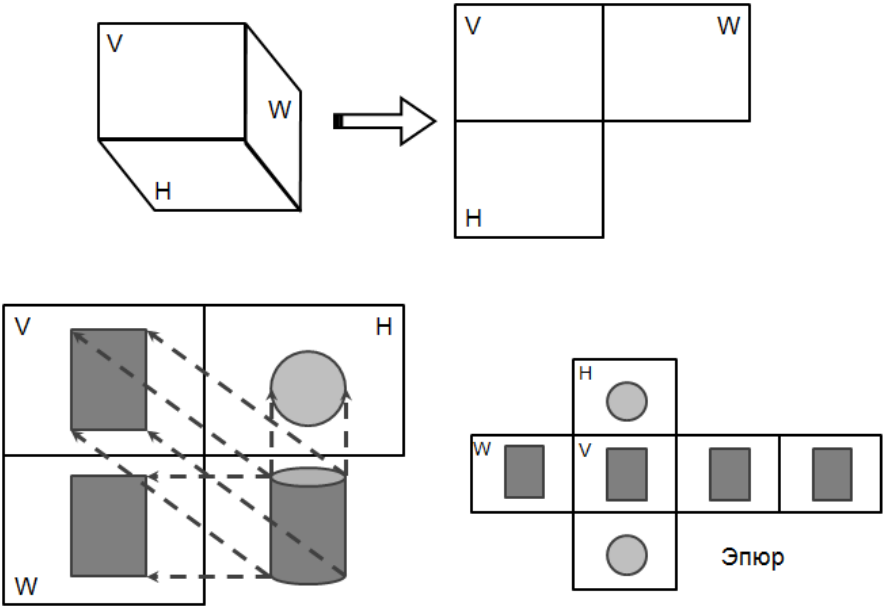

Рис.20. Видовые проекции для цилиндра.

АКСОНОМЕТРИЧЕСКИЕ ПРОЕКЦИИ

При аксонометрическом проецировании сохраняется параллельность прямых, но углы изменяются. *Коэффициенты искажения* (масштаб, отношение длины проекции отрезка к реальной длине) по каждой из проецируемых координатных осей в общем случае не равны друг другу. Оси OX, OY и OZ направлены влево-вниз, вправо-вниз и вверх соответственно. При отображении окружности, лежащей в плоскости, параллельной одной из плоскостей проекций, она проецируются в эллипс, у которого большая полуось перпендикулярна оси OX (для плоскости YZ), OY (для плоскости XZ) или OZ (для плоскости XY). В зависимости от коэффициентов искажения и углов поворота выделяют изометрическую, диметрическую и триметрическую проекции.

Изометрическая проекция (рис.21) обладает следующим свойством: масштаб по всем трем осям одинаков, так же, как и углы между проекциями координатных осей (табл.4). Параметры изменения окружности приведены в табл.5. У данной проекции есть не только классический, но и упрощенный вариант.

Диметрическая проекция (рис.22) обладает следующим свойством: коэффициенты искажения одинаковы только по двум осям (табл.6), параметры искажения окружности приведены в табл.7. У данной проекции также есть не только классический, но и упрощенный вариант.

Триметрическая проекция (как общий случай для изометрической и диметрической проекций) строится последовательными поворотами вокруг координатных осей, совершаемыми в произвольном порядке, с последующим проецированием на плоскость $z = 0$. Коэффициенты искажения не равны друг другу.

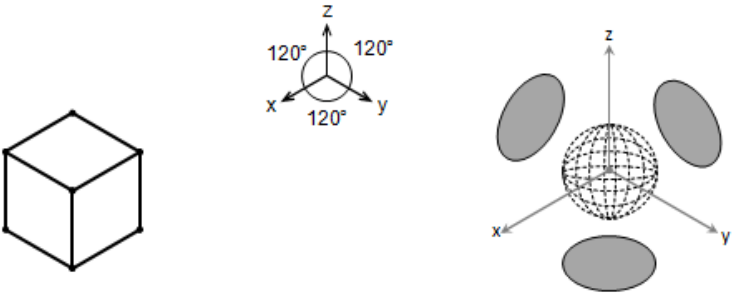

Рис.21. Изометрическая проекция: куб и сфера.

Таблица 4

Изометрическая проекция: искажение осей

Ось	Масштаб		Угол поворота относительно OZ
	Классический	Упрощенный	
OX	82%	100%	120°
OY	82%	100%	−120°
OZ	82%	100%	

Таблица 5

Изометрическая проекция: искажение окружности

Полуось	Масштаб		Угол к противооси
	Классический	Упрощенный	
Малая	58%	71%	0°
Большая	100%	122%	90°

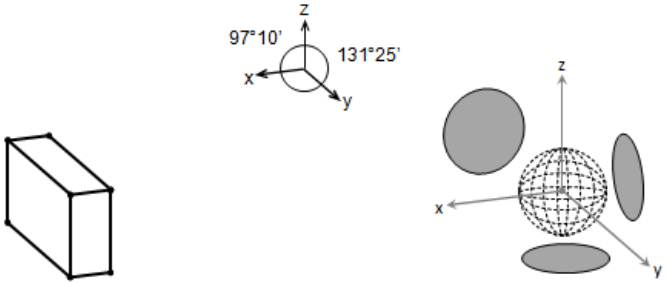

Рис.22. Диметрическая проекция: куб и сфера.

Таблица 6

Диметрическая проекция: искажение осей

Ось	Масштаб по осям		Угол поворота относительно OZ
	Классический	Упрощенный	
OX	94%	100%	$97°10'$
OY	47%	50%	$-131°25'$
OZ	94%	100%	

Таблица 7

Диметрическая проекция: искажение окружности

Плос-кость	Полуось	Масштаб		Угол к противооси
		Классический	Упрощенный	
XY (H)	Малая	33%	35%	0°
	Большая	100%	106%	90°
XZ (V)	Малая	90%	95%	0°
	Большая	100%	106%	90°
YZ (W)	Малая	33%	35%	0°
	Большая	100%	106%	90°

КОСОУГОЛЬНЫЕ ПРОЕКЦИИ

Косоугольные (наклонные) проекции сочетают в себе свойства видовых проекций со свойствами аксонометрии. В этом случае проекционная плоскость перпендикулярна главной координатной оси, поэтому сторона объекта, параллельная этой плоскости, проецируется так, что можно измерить углы и расстояния. Проецирование других сторон объекта также допускает проведение линейных (но не угловых) измерений вдоль главных осей. Среди косоугольных проекций выделяют фронтальную и горизонтальную изометрии и фронтальную диметрию.

Во *фронтальной косоугольной изометрии* (рис.23) длины сторон не меняются, но при этом оси OX и OZ взаимно перпендикулярны (табл.8). Параметры искажения окружностей приведены в табл.9. Полное название такой проекции: аксонометрическая косоугольная фронтальная изометрическая проекция.

В *горизонтальной косоугольной изометрии* (рис.24) длины сторон не меняются, но при этом оси OX и OY взаимно перпендикулярны (табл.10). Параметры искажения окружностей приведены в табл.11. Другие названия этой проекции: Кавалье (cavalier), военная проекция, свободная проекция. Полное название такой проекции: аксонометрическая косоугольная горизонтальная изометрическая проекция.

Во *фронтальной косоугольной диметрии* (рис.25) длины сторон только по осям OX и OZ не меняются, при этом оси OX и OZ взаимно перпендикулярны (табл.12). Параметры искажения окружностей приведены в табл.13. Другие названия этой проекции: Кабине (cabinet), кабинетная проекция. Полное название такой проекции: аксонометрическая косоугольная фронтальная диметрическая проекция.

.

.

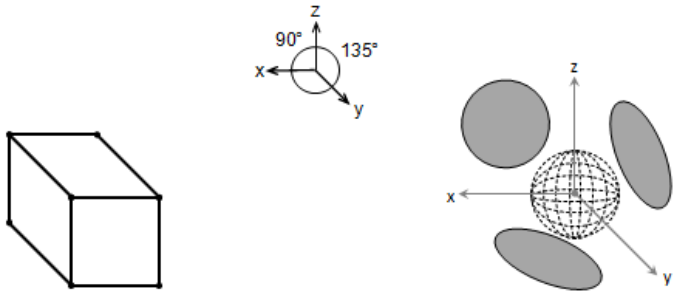

Рис.23. Фронтальная изометрическая проекция: куб и сфера.

Таблица 8

Фронтальная изометрическая проекция: искажение осей

Ось	Масштаб	Угол поворота относительно OZ		
OX	100%	90°		
OY	100%	−135°	−120°	−150°
OZ	100%	-		

Таблица 9

Фронтальная изометрическая проекция: искажение окружности

Плоскость	Полуось	Масштаб	Угол к противооси
XY (H)	Малая	54%	90° − 22°30′
	Большая	130%	22°30′
XZ (V)	Малая	100%	90°
	Большая	100%	0°
YZ (W)	Малая	54%	90° − 22°30′
	Большая	130%	22°30′

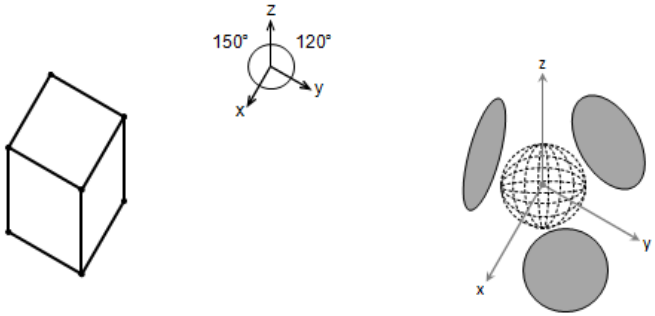

Рис.24. Горизонтальная изометрическая проекция: куб и сфера.

Таблица 10

Горизонтальная изометрическая проекция: искажения осей

Ось	Масштаб	Угол поворота относительно OZ		
OX	100%	150°	135°	120°
OY	100%	−120°	−135°	−150°
OZ	100%	-		

Таблица 11

Горизонтальная изометрическая проекция: искажение окружности

Плоскость	Полуось	Масштаб	Угол к противооси
XY (H)	Малая	100%	90°
	Большая	100%	0°
XZ (V)	Малая	37%	90° − 15°
	Большая	137%	15°
YZ (W)	Малая	71%	90° − 30°
	Большая	122%	30°

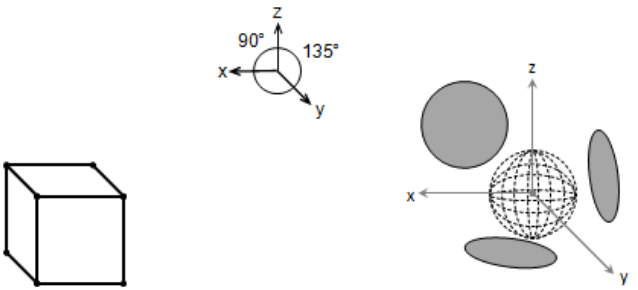

Рис.25. Фронтальная диметрическая проекция: куб и сфера.

Таблица 12

Фронтальная диметрическая проекция: искажение осей

Ось	Масштаб	Угол поворота относительно OZ		
OX	100%	90°		
OY	50%	135°	120°	150°
OZ	100%	-		

Таблица 13

Фронтальная диметрическая проекция: искажение окружности

Плоскость	Полуось	Масштаб	Угол к противооси
XY (H)	Малая	33%	$90° - 7°14'$
	Большая	107%	$7°14'$
XZ (V)	Малая	100%	90°
	Большая	100%	0°
YZ (W)	Малая	33%	$90° - 7°14'$
	Большая	107%	$7°14'$

ПРЕОБРАЗОВАНИЯ КАМЕРЫ

Фактически, камера эквивалентна человеку-наблюдателю. И все перемещения по сцене (миру) реализуются через модельно-видовые преобразования, к которым относятся рассмотренные ранее преобразования сдвига, масштаба и поворота в однородных координатах.

Реальной камере свойственно несколько характеристик, которые легко преобразуются в математические описания, необходимые для трехмерной графики (рис.26):

1) Определенная позиция в пространстве (трехмерная точка);
2) Направление наблюдения (трехмерный вектор, получаемый путем вычитания позиции камеры из точки наблюдения);
3) Поперечный наклон (трехмерный вектор направления вверх, перпендикулярный вектору наблюдения).

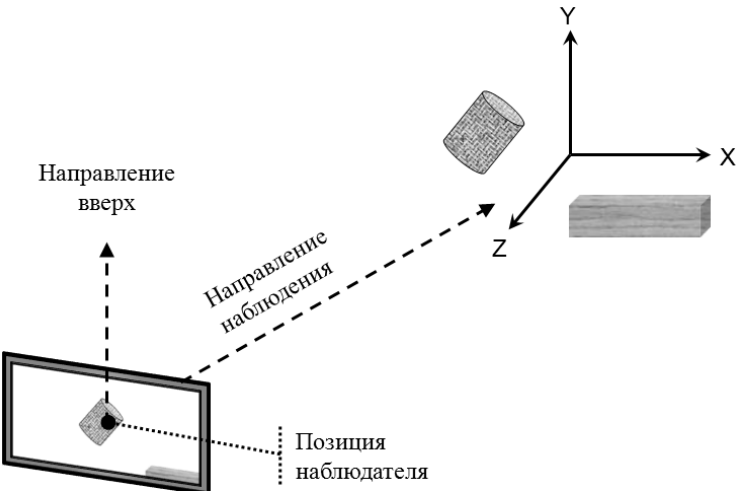

Рис.26. Задание камеры.

Для вывода трехмерного изображения на экран, его нужно не просто спроецировать на экранную плоскость, но еще и осечь все, что не попадает в так называемый *объем видимости* (куб или пирамиду видимости), который соответствует полю зрения человека (рис.27).

Поле зрения – это угловое пространство, видимое глазом при фиксированном взгляде и неподвижной голове. По вертикали среднестатистический человек имеет *угол зрения* примерно 110°, а по горизонтали – до 180° (но распознавать объекты как трехмерные только в пределах 110°, а как полноцветные – в ещё меньшем диапазоне). У некоторых же птиц поле зрения достигает почти 360°.

В компьютерной графике у камеры появляется дополнительная характеристика, невозможная в реальной жизни – вид проекции (перспективная или ортографическая). Перспективная проекция более соответствует реальности (чем дальше объект удален от камеры, тем меньшими кажутся его размеры), а ортографическая проекция используется для чертежной графики.

По сути, камера представляет собой преобразования вида (view) и проекции (projection): видовая матрица размещает и ориентирует камеру в трехмерном пространстве, а проекционная матрица проекции задает перспективу. Эти преобразования камеры применяются после всех остальных матричных преобразований, используемых для позиционирования объектов, причем преобразование вида применяется до проецирования.

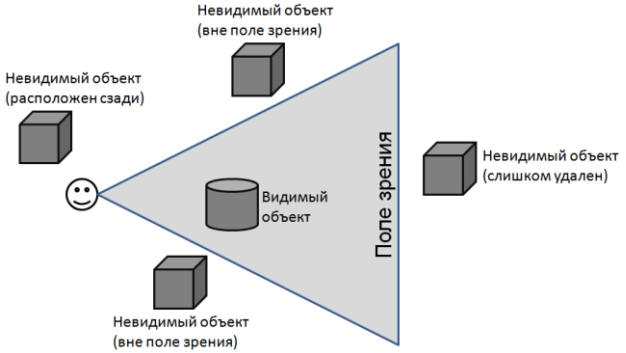

Рис.27. Поле зрения: видимые и невидимые объекты.

МОДЕЛЬНО-ВИДОВЫЕ ПРЕОБРАЗОВАНИЯ

Как можно понять из названия, модельно-видовые преобразования состоят из двух преобразований: модельного и видового. Модельное преобразование соответствует заданию объекта, а видовое – камеры.

Выполнение подобных преобразований не изменяет четвертую компоненту однородных координат (она остается равной 1).

ПРЕОБРАЗОВАНИЕ ВИДА

Преобразование вида, также представляет собою общее модельное преобразование (единую матрицу преобразований), применяемое не к одному объекту, а сразу ко всей сцене.

Например, сдвиг объекта «вправо-вверх» эквивалентен сдвигу камеры «влево-вниз» (рис.28).

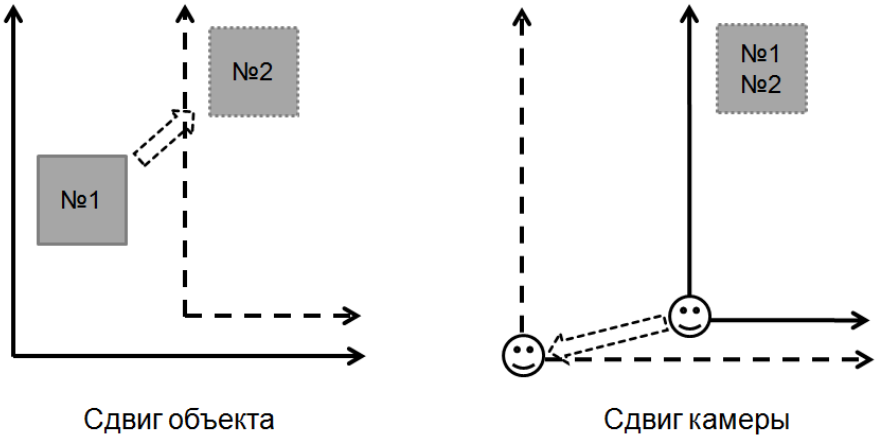

Рис.28. Перемещение камеры.

ПРЕОБРАЗОВАНИЕ МОДЕЛИ

Допустим, необходимо задать два квадрата (определяемых четверками вершин), один из которых расположен у начала координат, а другой – сдвинут и повернут (рис.29). И если для первого квадрата почти нет проблем получить координаты его вершин, то для второго квадрата уже начинаются затруднения.

Эти проблемы можно разрешить, если квадраты задавать в своей собственной (*локальной*) системе координат, например, где левая нижняя вершина будет находиться в точке (0,0), а стороны квадрата параллельны осям. Тогда для перехода в исходную (*глобальную*) систему координат необходимо будет применить ко всем его вершинам операции сдвига и поворота. Вообще говоря, такой способ облегчит задание даже первого квадрата.

Таким образом, имеет смысл каждый объект задавать в локальной (*модельной*) системе координат, а потом преобразовывать в глобальную путем выполнения единого (*модельного*) преобразования (составленного сразу для всех его вершин из рассмотренных ранее операций сдвига, масштаба и поворота).

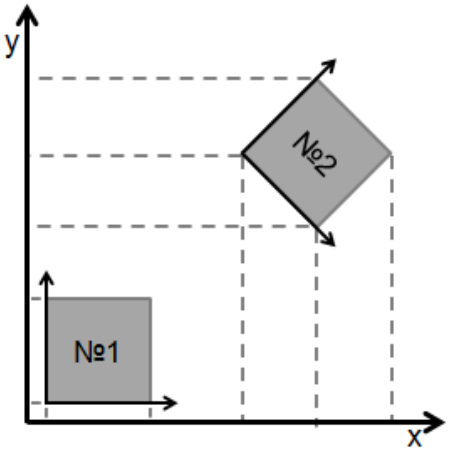

Рис.29. Локальные и глобальная системы координат.

ПРОЕКТИВНЫЕ ПРЕОБРАЗОВАНИЯ

В большинстве случаев в компьютерной графике используются ортографическое и перспективное проецирование. В результате этих преобразований система координат преобразуется в **левостороннюю**, а четвертая компонента однородных координат перестает быть равной 1.

В связи с тем, что изображение выводится на прямоугольный экран, область видимости представляется в виде усеченной пирамиды с четырехугольным основанием (рис.30), задаваемой шестью гранями:

- левой (left);
- правой (right);
- нижней (botom);
- верхней (top);
- передней (near);
- задней (far).

Таким образом, левому нижнему углу окна вывода соответствуют точка (left, bottom, near), а правому верхнему – (right, top, near).

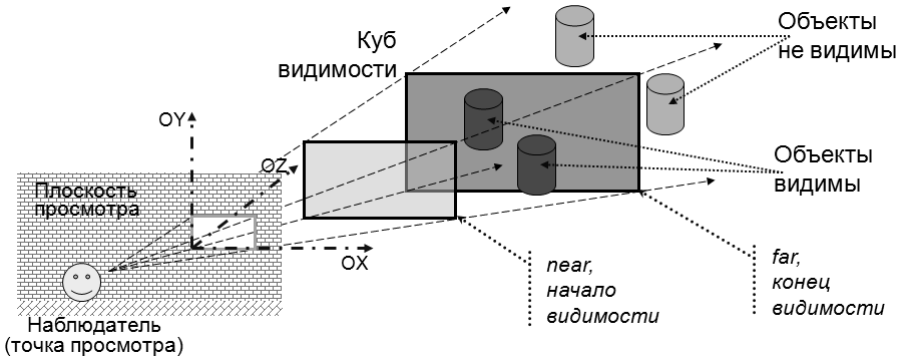

Рис.30. Пирамида видимости и камера.

ОРТОГРАФИЧЕСКЕЙ ПРОЕЦИРОВАНИЕ

В этом случае проецирование на область вывода осуществляется как ортогональное (выполняются только преобразования сдвига и масштаба). Таким образом, областью видимости является прямоугольный параллелепипед, описываемый матрицей вида:

$$P = \begin{pmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bottom} & 0 & -\frac{top+bottom}{top-bottom} \\ 0 & 0 & \frac{-2}{far-near} & -\frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

ПЕРСПЕКТИВНОЕ ПРОЕЦИРОВАНИЕ

В этом случае проецирование на область вывода осуществляется как центральное одноточечное, т.е. выполняются преобразования сдвига, масштаба и изменения по оси OZ. Таким образом, областью видимости является усеченная пирамида, описываемая матрицей вида:

$$P = \begin{pmatrix} \frac{2\times near}{right-left} & 0 & \frac{right+left}{right-left} & 0 \\ 0 & \frac{2\times near}{top-bottom} & \frac{top+bottom}{top-bottom} & 0 \\ 0 & 0 & -\frac{far+near}{far-near} & -\frac{2\times far\times near}{far-near} \\ 0 & 0 & -1 & 0 \end{pmatrix}.$$

УДАЛЕНИЕ «НЕВИДИМЫХ» ЭЛЕМЕНТОВ

Алгоритмы удаления невидимых элементов заключаются в определении примитивов или их частей, которые видимы или невидимы для наблюдателя, находящегося в заданной точке пространства. Сложность данной задачи привела к появлению большого числа различных алгоритмов ее решения для различных ситуаций. Например, алгоритм Робертса (один из старейших алгоритмов) был разработан для штриховой отрисовки выпуклых многоугольников. Более поздний алгоритм Варнока основывается на гипотезе о способе обработки информации, содержащейся в сцене, глазом и мозгом человека.

Самым универсальным (и самым затратным) является алгоритм трассировки лучей, когда из позиции наблюдателя в нужных направлениях испускаются лучи и ищутся их ближайшие пересечения со всеми объектами. Если объект прозрачен, тогда луч продолжает трассироваться и дальше.

Необходимость удаления невидимых элементов (линий, поверхностей и т.п.) проиллюстрирована на рис.31. Рисунок наглядно демонстрирует, что изображение без удаления невидимых линий воспринимается неоднозначно.

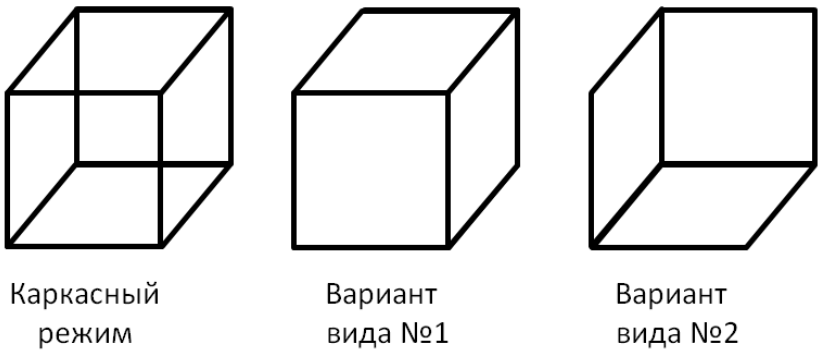

Рис.31. Неоднозначность восприятия изображения куба.

МЕТОД ХУДОЖНИКА

Метод художника основан на том, как создает изображение художник: сначала накладывается фон, потом рисуются дальние объекты и только в конце – ближние. Соответственно, суть метода художника состоит в сортировке граней по удаленности от наблюдателя с последующей их отрисовкой, начиная с более дальних.

Проблема данного метода состоит в неоднозначности и затратности самой сортировки. На рис.32 приведен пример ситуации, когда заданы две пересекающиеся грани AB и CD, пунктирными линиями показаны части граней, которые должны стать невидимыми. Если организовать сортировку граней по расстоянию до их середин, тогда пропадет часть грани CD (рис.33), а если по расстоянию до ближайшей точки – пропадет часть грани AB (рис.34). Данную проблему можно решить путем разрезания граней.

Данный метод используется в методе *двоичного разбиения пространства,* строящего BSP-дерево (Binary Space Partitioning). Здесь пространство рекурсивно разбивается плоскостями на полупространства до тех пор, пока в каждой секции не окажется по одной грани, при этом, каждый раз определяется, какая часть должна рисоваться ранее.

Попытка разрешения конфликтов при сортировке привела к появлению метода Ньюэла-Нюэла-Санча с его пятью тестами-вопросами на пересечение граней.

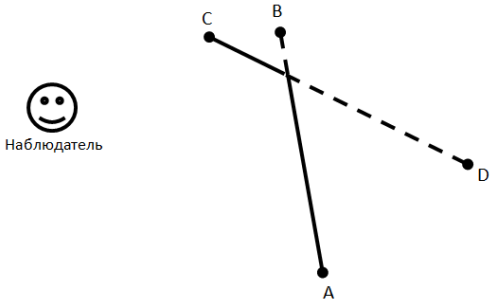

Рис.32. Метод художника: пересекающиеся грани AB и CD.

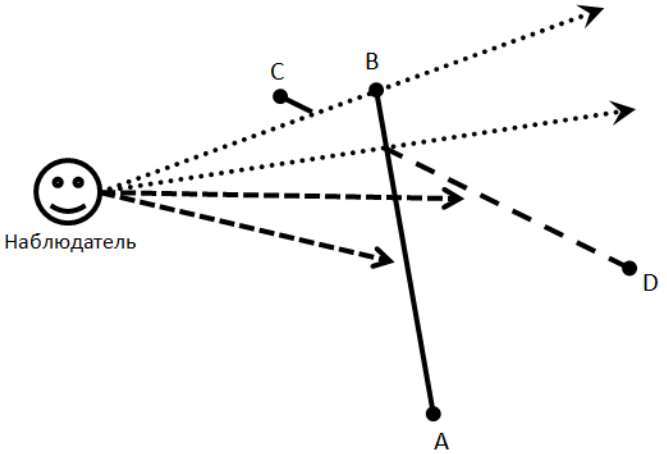

Рис.33. Метод художника: сортировка по серединам граней.

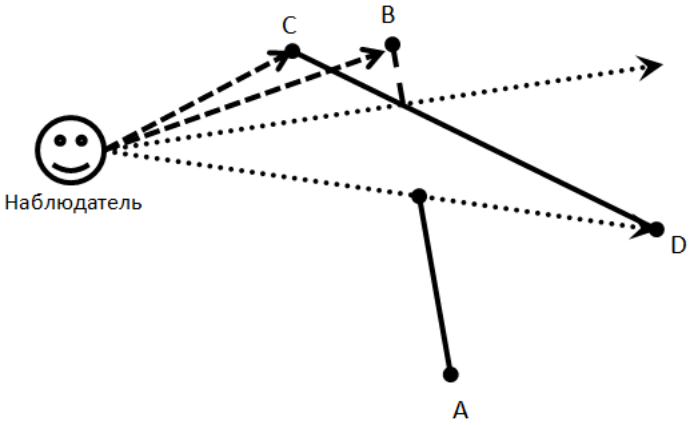

Рис.34. Метод художника: сортировка по ближайшей вершине.

МЕТОД ЛИЦЕВЫХ ГРАНЕЙ

Трехмерный объект задается набором плоских граней, например, куб задается шестью гранями. Каждая грань представляет собою часть плоскости, описываемой каноническим уравнением вида

$$ax + by + cz + d = 0\,.$$

В однородных координатах вектор "расширенной" нормали и произвольная точка P примут вид:

$$\vec{n} = \overrightarrow{(a,b,c,d)}\,,$$
$$P = (x, y, z, 1)\,.$$

Пример для куба, повернутого к наблюдателю одной гранью, приведен на рис.35.

Скалярное произведение $(\vec{n}, P)$ обладает следующими свойствами:

1) $(\vec{n}, P) > 0$, если P находится "снаружи" плоскости;
2) $(\vec{n}, P) = 0$, если P находится на плоскости;
3) $(\vec{n}, P) < 0$, если P находится "внутри" плоскости.

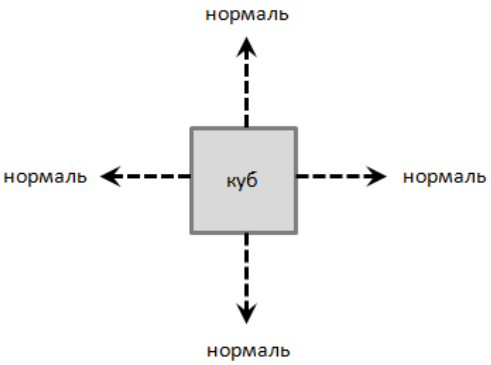

Рис.35. Нормали для граней куба.

Замечание. Очевидно, что для определения принадлежности точки кубу скалярные произведения всех шести граней с этой точкой не должны быть больше нуля. На этом принципе основан один из алгоритмов поиска принадлежности точки многоугольнику.

В зависимости от того, куда направлен вектор нормали, грани разделяют на *лицевые* (нормаль направлена из объекта) и *нелицевые* (нормаль направлена внутрь объекта). Лицевые грани могут быть видимы с различных позиций наблюдения, а нелицевые грани не должны быть видимы.

Данный способ весьма эффективно отбраковывает все грани, которые даже частично не будут видимы. Использование данного подхода в качестве полноценного метода удаления невидимых граней наталкивается на серьезные ограничения:

1) все объекты должны быть выпуклыми;
2) объекты не должны перекрывать друг друга.

Поскольку второе ограничение очевидно, то подробнее будет рассмотрено только первое. На рис.36 показана ситуация, когда у невыпуклой фигуры ABCDEF помимо граней AB и AF становится видимой еще и грань CD (хотя ее она должна быть закрыта гранями AB и AF).

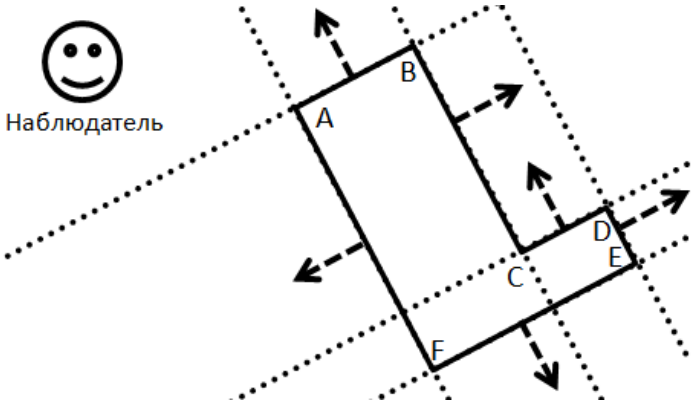

Рис.36. Пример видимости граней.

МЕТОРД БУФЕРА ГЛУБИНЫ

Метод буфера глубины (Z-буфер) является одним и простейших, но эффективных методов удаления невидимых элементов для сложных систем объектов (в условиях сложной геометрии).

Идея Z-буфера является простым обобщением идеи о буфере цвета. Буфер цвета используется для запоминания цвета каждого пикселя в картинной плоскости, а Z-буфер предназначен для запоминания глубины (расстояния от картинной плоскости) каждого видимого пикселя в пространстве изображения. Поскольку в качестве картинной плоскости выступает плоскость XY, то z-компонента и будет этой самой глубиною.

При растеризации примитивов, глубина каждого его пикселя сравнивается с глубиной пикселя, который уже занесен в z-буфер. Если это сравнение показывает, что новый пиксель расположен ближе запомненного пикселя, тогда новый пиксель и его глубина записываются в буфер цвета и z-буфер соответственно.

Главным недостатком данного метода является возможность потери эффектов, связанных с прозрачностью, поскольку при запоминании пикселя теряется информация о его прозрачности. В этом случае необходимо перед растеризацией отсортировать объекты по удаленности и прозрачности.

По сути, алгоритм буфера глубины является частным случаем *трассировки лучей*, когда через каждый пиксель экрана под прямым углом испускаются лучи, для каждого из которых ищется ближайшее пересечение данного луча с объектами, искаженными проекцией камеры.

КОМАНДЫ БИБЛИОТЕКА OPENGL

Общее описание библиотеки OpenGL, разбор ее команд и типов данных для создания двумерной графики и создание консольного приложения с использованием библиотеки GLUT приведены в [1].

В данной же работе упор сделан на описание команд для создания трехмерной графики:

- задание вершин в 3D и 4D;
- задание вектора нормали;
- включение режима буфера глубины и лицевых граней;
- включение освещения;
- работа с модельно-видовыми преобразованиями;
- задание режима проекции.

В качестве примера рассматривается консольное приложение, визуализирующее разноцветный куб в различных режимах без использования шейдеров. Текст базового кода данного приложения приведен в Приложении.

ЗАДАНИЕ АТРИБУТОВ ВЕРШИН

Все объекты задаются набором примитивов, а примитивы, в свою очередь, набором вершин и атрибутов. Вершины задаются либо в операторных скобках `glBegin/glEnd`, либо списком вершин.

У каждой вершины есть стандартный набор атрибутов, к которым относятся:

- геометрические координаты;
- текстурные координаты (здесь не рассматриваются);
- цвет в формате RGB/RGBA;
- вектор нормали.

ЗАДАНИЕ ЦВЕТА

Команды задания цвета в режиме RGB/RGBA подробно рассмотрены в [1]. В большинстве случаев достаточно использовать только следующие четыре варианта команды:

```
void glColor3ub  (GLubyte R, GLubyte G, GLubyte B);
void glColor3ubv (GLubyte *RGB);
void glColor3f  (GLfloat R, GLfloat G, GLfloat B);
void glColor3fv (GLfloat *RGB).
```

ЗАДАНИЕ НОРМАЛИ

Вектор нормали используется при расчете освещения и в методе лицевых граней. Вектор нормали может быть выбран произвольно, однако по смыслу он должен:

1) быть перпендикулярным к поверхности в данной вершине;
2) иметь единичную длину;
3) задаваться только для лицевой стороны поверхности.

Для задания нормали используется команда `glNormal`. Всего существует 10 вариантов (в зависимости от типа данных) этой команды, они приведены в табл.14. В основном же используется следующая пара команд:

```
glNormal3f(GLfloat x, GLfloat y, GLfloat z);
glNormal3fv(const GLfloat *xyz).
```

Нормаль задается для каждой вершины внутри операторных скобок `glBegin/glEnd` перед заданием координат данной вершины. Если у всех вершин примитива нормаль одна и та же, ее можно задать только один раз (для первой вершины) – как и при задании цвета.

При использовании модельно-видовых преобразований вектор нормали меняется вместе с координатами вершин, в результате чего длина нормали может измениться. Соответственно, автоматическое приведение всех нормалей к единичной длине можно обеспечить включением режима автонормализации командой

```
glEnable(GL_NORMALIZE).
```

Таблица 14

Варианты команды задания нормали

Тип данных				
GLbyte	*GLshort*	*GLint*	*GLfloat*	*GLdouble*
glNormal3b glNormal3bv	glNormal3s glNormal3sv	glNormal3i glNormal3iv	glNormalfs glNormalfsv	glNormal3d glNormal3dv

ЗАДАНИЕ КООРДИНАТ

Для задания координат вершин используется команда `glVertex`. Всего существует 24 варианта этой команды (табл.15) – в зависимости от размерности пространства и типа данных. При задании трехмерных объектов чаще всего используются следующие варианты команды:

```
void glVertex3f (GLfloat x, GLfloat y),
void glVertex3fv(GLfloat *xy).
```

Поскольку OpenGL работает через однородные координаты, все двух- и трехмерные координаты вершин автоматически преобразуются в четырехмерные (четвертая координата принимает значение 1). При задании двумерной вершины координата z задается нулевой (считается, что работа происходит в плоскости z=0). Например, точка с координатами (3,2) будет автоматически преобразована в точку (3,2,0,1).

Таблица 15

Варианты команды задания точки

Размерность	Тип данных			
	GLshort	*GLint*	*GLfloat*	*GLdouble*
2D	glVertex2s glVertex2sv	glVertex2i glVertex2iv	glVertex2f glVertex2fv	glVertex2d glVertex2dv
3D	glVertex3s glVertex3sv	glVertex3i glVertex3iv	glVertex3f glVertex3fv	glVertex3d glVertex3dv
4D	glVertex4s glVertex4sv	glVertex4i glVertex4iv	glVertex4f glVertex4fv	glVertex4d glVertex4dv

ЗАДАНИЕ СПИСКА ВЕРШИН

При работе с многополигональными объектами используется большое количество вершин. В этом случае вместо операторных скобок имеет смысл воспользоваться списками вершин, которые позволяют разом задать атрибуты вершин с помощью массивов:

1. Команда `glVertexPointer` задает координаты вершин.
2. Команда `glColorPointer` задает цвета вершин.
3. Команда `glNormalPointer` задает нормали к вершинам.

В качестве параметров этих команд нужно задать число элементов, тип данных, сдвиг и сам массив элементов. Если вершина, ее цвет и нормаль хранятся подряд (в соседних позициях общего массива), тогда в качестве сдвига нужно задавать не 0, а суммарную размерность в байтах элементов цвета и нормали.

Включение данного режима рисования выполняется командой `glEnableClientState` с параметром `GL_VERTEX_ARRAY` для вершин, `GL_COLOR_ARRAY` для цвета или `GL_NORMAL_ARRAY` для нормалей. Выключение же режима рисования выполняется командой `glDisableClientState` с теми же самыми параметрами.

Команда `glDrawArrays` выполняет отрисовку указанного массива вершин с учетом массивов цветов и нормалей. В качестве параметров требуется указать тип примитива, начальную позицию в массивах и количество отрисовываемых вершин.

Пример кода для задания списка вершин, нормалей и цветов:

```
glEnableClientState ( GL_VERTEX_ARRAY );
glEnableClientState ( GL_COLOR_ARRAY );
glEnableClientState ( GL_NORMAL_ARRAY );
 glVertexPointer ( 3, GL_FLOAT, 0, xyz );
 glColorPointer  ( 3, GL_FLOAT, 0, rgb );
 glNormalPointer ( GL_FLOAT, 0, normals );
  glDrawArrays   ( GL_QUADS, 0, count );
glDisableClientState ( GL_NORMAL_ARRAY );
glDisableClientState ( GL_COLOR_ARRAY );
glDisableClientState ( GL_VERTEX_ARRAY );
```

УДАЛЕНИЕ НЕВИДИМЫХ ЭЛЕМЕНТОВ

В библиотеке OpenGL из множества методов удаления невидимых элементов реализованы только методы лицевых граней и буфера глубины.

Библиотека предоставляет возможность дополнительно отсечь часть получаемого изображения специальными плоскостями.

Также библиотека позволяет зафиксировать "ножницами" часть окна, запретив изменение изображения за пределами этой области.

ОТСЕЧЕНИЕ ЧАСТИ ИЗОБРАЖЕНИЯ ПЛОСКОСТЯМИ

Дополнительное отсечение элементов (например, с целью заглянуть внутрь фигуры) происходит путем использования специальных плоскостей отсечения, всего таких плоскостей может быть 6, они задаются своими константами-идентификаторами:

`GL_CLIP_PLANE0` ÷ `GL_CLIP_PLANE5`.

Включение/выключение режима выполняется стандартно командами `glEnable/glDisable` с указанием одного из вышеприведенных идентификаторов.

Задание уравнения плоскости отсечения выполняется командой `glClipPlane`, первым параметром которой выступает идентификатор плоскости, а вторым – массив из четырех элементов типа `GLdouble`, где хранятся коэффициенты уравнения плоскости. Для уравнения $ax + by + cz + d = 0$ содержимое данного массива будет следующим:

$$\{a, b, c, d\}.$$

Следующий код соответствует рассечению куба плоскостью $50x - 50y = 10$, результат представлен на рис.37:

```
GLdouble equation[4] = { 50, -50, 0, 10};
glEnable ( GL_CLIP_PLANE0 );
glClipPlane ( GL_CLIP_PLANE0, equation );

   DrawCube();

glDisable ( GL_CLIP_PLANE0 );
```

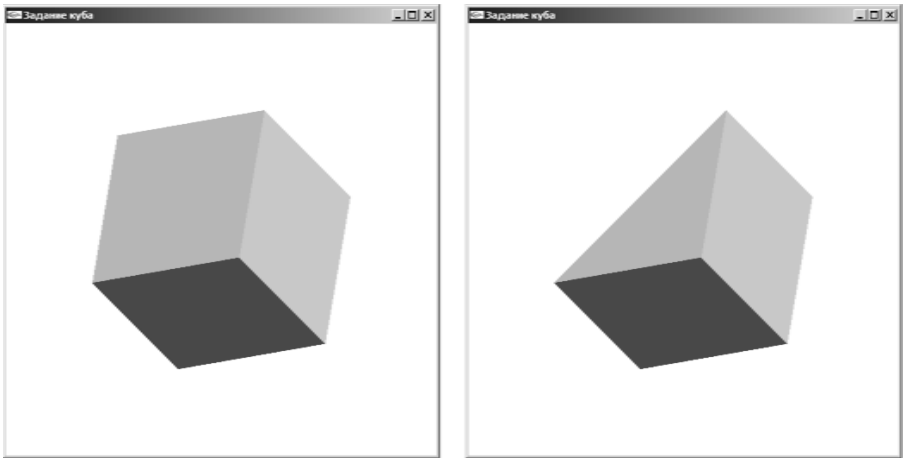

Рис.37. Рассечение куба плоскостью.

ОТСЕЧЕНИЕ ЧАСТИ ИЗОБРАЖЕНИЯ ПРЯМОУГОЛЬНИКОМ

Также можно воспользоваться и «ножницами» в виде прямоугольника (части окна вывода), указывающего, какие пиксели получаемого изображения разрешено модифицировать (все, что не попало в пределы этого прямоугольника, остается неизменным).

Включение/выключение режима выполняется командами `glEnable/glDisable` с указанием идентификатора `GL_SCISSOR_TEST`.

Сама же область вырезания задается командой `glScissor`, в качестве параметров которой выступают левый нижний угол плюс ширина и высота этого прямоугольника в пикселях.

Например, выполнение следующего кода приведет к тому, что будет отрисована только верхняя половина окна с соответствующей частью куба (рис.38):

```
glEnable ( GL_SCISSOR_TEST );
glScissor ( 0, WinHeight/2, WinWidth, WinHeight/2 );
   DrawCube();
glDisable ( GL_SCISSOR_TEST );
```

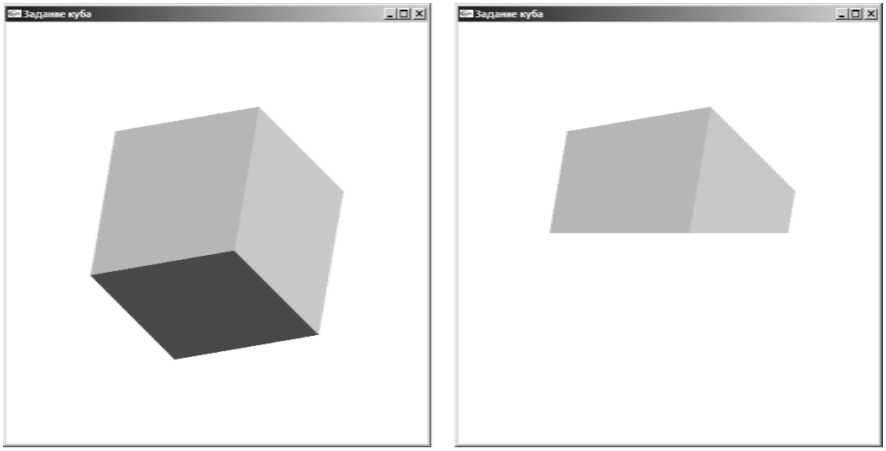

Рис.38. Отсечение куба ножницами.

ВКЛЮЧЕНИЕ МЕТОДА БУФЕРА ГЛУБИНЫ

Создание буфера глубины выполняется следующей командой библиотеки GLUT:

```
glutInitDisplayMode ( GLUT_DEPTH ).
```

Включение/выключение режима выполняется командами `glEnable/glDisable` с указанием идентификатора `GL_DEPTH_TEST`. На рис.39 представлен результат работы буфера глубины.

При каждой отрисовке окна необходимо очищать буфер глубины от предыдущей информации, это выполняется командой `glClear` с параметром `GL_DEPTH_BUFFER_BIT`.

Дополнительные настройки буфера глубины можно выполнить с помощью команд `glDepthMask` и `glDepthFunc`.

Команда `glDepthMask` при запуске с параметром `GL_FALSE` временно отключает запись (но не саму проверку) в буфер, соответственно, выполнение команды `glDepthMask(GL_TUE)` обратно включает возможность записи.

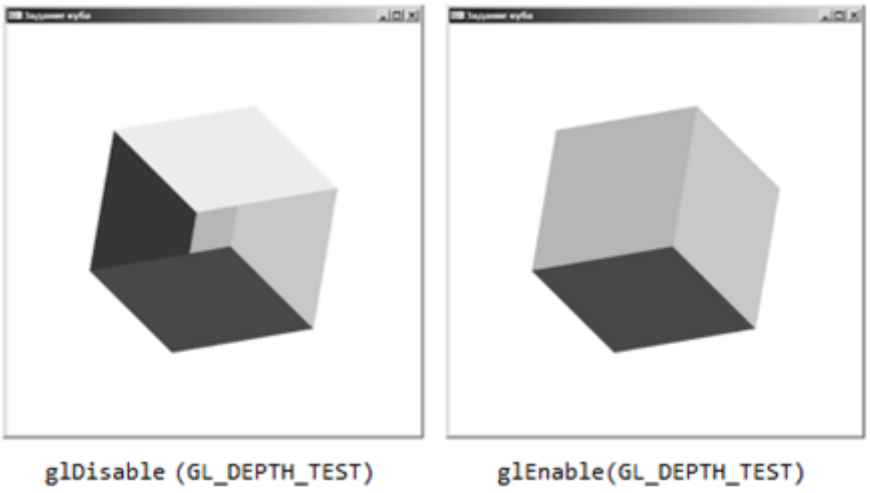

Рис.39. Пример работы буфера глубины.

Настроить параметры записи новых значений в буфер можно с помощью команды `glDepthFunc`, у которой есть только один параметр, (его возможные значения приведены в табл.16). В частности, выполнение команды `glDepthFunc(ALWAYS)` приведет к тому, что будут видны только те полигоны, которые были занесены последними.

Таблица 16

Значения параметра функции `glDepthFunc`

Значение	Описание условия для прохождения (выполнения) теста глубины (условие записи в буфер)
GL_ALWAYS	ВСЕГДА (буфер не работает, всегда перезаписывается)
GL_NEVER	НИКОГДА (буфер не работает, никогда не перезаписывается)
GL_EQUAL	ЕСЛИ новое значение РАВНО сохраненному значению
GL_NOTEQUAL	ЕСЛИ новое значение НЕ РАВНО сохраненному значению
GL_LESS	ЕСЛИ новое значение МЕНЬШЕ, чем сохраненное значение (значение по умолчанию)
GL_LEQUAL	ЕСЛИ новое значение МЕНЬШЕ ИЛИ РАВНО сохраненному значению
GL_GREATER	ЕСЛИ новое значение БОЛЬШЕ, чем сохраненное значение
GL_GEQUAL	ЕСЛИ новое значение БОЛЬШЕ ИЛИ РАВНО сохраненному значению

ВКЛЮЧЕНИЕ МЕТОДА ЛИЦЕВЫХ ГРАНЕЙ

Включение/выключение режима выполняется командами `glEnable/glDisable` с указанием идентификатора `GL_CULL_FACE`.

Сама же отбраковка нелицевых граней выполняется командой

```
glCullFace ( GL_FRONT ).
```

При необходимости отбраковки лицевых граней можно воспользоваться командой

```
glCullFace ( GL_BACK ).
```

Направление нормали зависит от того, в каком направлении были перечислены вершины грани. Если они перечисляются против часовой стрелки, то это соответствует команде

```
glFrontFace ( GL_CCW ),
```

а для перечисления по часовой стрелке – команда

```
glFrontFace ( GL_CW ).
```

На рис.40 представлен результат отбраковки лицевых и неолицевых граней. Как можно видеть, при задании куба не были согласованы направления нормалей и обхода вершин граней, поэтому результат получился ошибочным.

Рис.40. Пример "ошибочной" работы лицевых граней.

ЗАДАНИЕ ОСВЕЩЕНИЯ

В OpenGL используется модель освещения Фонга, в соответствии, с которой цвет точки определяется несколькими факторами:

- свойствами материала;
- свойствами текстуры (здесь не рассматривается);
- вектором нормали;
- положением и направлением источника света;
- положением наблюдателя.

Для корректного расчета освещенности в точке необходимо использовать единичные нормали.

Поскольку в качестве модели освещения используется модель Фонга, для учета освещения используются следующие компоненты освещенности:

- фоновая (ambient);
- диффузная (diffuse);
- зеркальная (specular).

Переменные, которые задают связанные с цветом значения, представляют собой вектор из четырех вещественных или целочисленных компонент (модель RGBA). Соответственно, переменные, которые задают связанные с положением или направлением значения, представляют собой аналогичный вектор (однородные координаты).

В данной работе приводится только краткое описание освещения. Пример задания простого освещения приведен в Приложении.

ТИПЫ ИСТОЧНИКОВ СВЕТА

В табл.17 приведены допустимые в OpenGL источники освещения (лампы).

Таблица 17

Типы источников света

Тип источника	Описание действия	Способы задания
Глобальный фоновый	Равномерный свет во всем пространстве (без теней)	Интенсивность
Эмиссионный	Подсветка объекта самого себя	Интенсивность в точке
Точечный	Излучение равномерное по всем направлениям из точки	Интенсивность Положение
Прожекторный	Излучение конуса света по направлению из точки	Интенсивность Положение Направление Угол Распределение
Направленный	Бесконечно удаленный (все лучи параллельны)	Интенсивность Направление

ЗАДАНИЕ МОДЕЛИ ОСВЕЩЕНИЯ

Базовая модель освещения задается командой `glLightModel`:

```
void glLightModeli(GLenum pname, GLenum param),
void glLightModeliv(GLenum pname, const GLtype *params),
void glLightModelf(GLenum pname, GLenum param),
void glLightModelfv(GLenum pname, const GLtype *params).
```

С помощью этой команды можно настроить параметры глобального освещения, но обычно устанавливается режим двустороннего освещения и значение глобальной фоновой составляющей.

Включение режима двустороннего освещения (одновременный расчет освещенности как для лицевых, так и для и нелицевых граней), выполняется следующим образом:

```
glLightModelfv(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE);
```

Изменение значения глобальной фоновой составляющей (независящей от включаемых источников света), по умолчанию равной {0.2, 0.2, 0.2, 1}, выполняется следующим образом:

```
float ambient[4] = { 0.5, 0.5, 0.5, 1};
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambient);
```

ВКЛЮЧЕНИЕ ОСВЕЩЕНИЯ

Включение/выключение режима выполняется командами `glEnable/glDisable` с указанием идентификатора `GL_LIGHTING`.

В библиотеке доступно к использованию до 8 источников света (количество можно узнать через значение константы `GL_MAX_LIGHT`), которые включаются/выключаются командами `glEnable/glDisable` с указанием соответствующих констант-идентификаторов:

`GL_LIGHT0 ÷ GL_LIGHT7`.

ЗАДАНИЕ ПАРАМЕТРОВ ИСТОЧНИКА СВЕТА

Параметры источника света задаются командой `glLight`:

```
void glLighti (GLenum light, GLenum pname, GLfloat param);
void glLightiv(GLenum light, GLenum pname, GLfloat param);
void glLightf (GLenum light, GLenum pname, GLfloat param);
void glLightfv(GLenum light, GLenum pname, GLfloat param).
```

Здесь первым параметром (`light`) указывается идентификатор источника, вторым и третьим – идентификатор параметра освещения (`pname`) и его новое значение (`param`). Краткое описание параметров команды приведено в табл.18.

Каждый источник задается своей позицией и тремя параметрами излучения света: фоновым, диффузным и зеркальным. Позиция источника может быть изменена модельно-видовыми преобразованиями.

По умолчанию источник считается направленным (param[3] = 0), соответственно, первые 3 компоненты этого вектора задают направление светового излучения. Установка последней компоненты вектора в 1 означает, что источник точечный, соответственно, первые 3 компоненты этого вектора задают позицию источника. При желании, источник можно задать как прожекторный, излучение от которого распространяется конусом, причем распределение света в пятне конуса по умолчанию считается константным (но можно задать и экспоненциальным).

По умолчанию считается, что свет распространяется без затухания. Затухание с расстоянием можно включить, при этом коэффициент затухания будет рассчитываться по следующей обратно-квадратчиной формуле:

$$f\left(dist\right)=\frac{1}{k_{const}+k_{linear}\cdot dist+k_{quadratic}\cdot dist^{2}}\ ,$$

где $dist$ – расстояние от позиции источника до освещаемой точки.

Таблица 18

Параметры функции glLight

Параметр `pname`	Описание	Значение `param` по умолчанию
GL_AMBIENT	Фоновая составляющая	$\{ 0, 0, 0, 1 \}$
GL_DIFFUSE	Диффузная составляющая	$\{ 1, 1, 1, 1 \}$ для `LIGHT0` $\{ 0, 0, 0, 1 \}$ для остальных
GL_SPECULAR	Зеркальная составляющая	
GL_POSITION	Координаты источника	$\{ 0, 0, 1, 0 \}$ направленный
GL_SPOT_EXPONENT	Прожектор: фокус	0
GL_SPOT_CUTOFF	Прожектор: угол излучения	180
GL_SPOT_DIRECTION	Прожектор: направление	$\{ 0, 0, -1, 1 \}$
GL_CONSTANT_ATTENUATION	Коэффициенты затухания	$k_{const} = 1$
GL_LINEAR_ATTENUATION		$k_{linear} = 0$
GL_QUADRATIC_ATTENUATION		$k_{quadratic} = 0$

ЗАДАНИЕ МАТЕРИАЛА

Материал может рассеивать, отражать и излучать свет. Свойства материала устанавливаются при помощи функции `glMaterial`:

```
void glLightModeli(GLenum face, GLenum pname, GLfloat param),
void glLightModeliv(GLenum face, GLenum pname, GLfloat param),
void glLightModelf(GLenum face, GLenum pname, GLfloat param),
void glLightModelfv(GLenum face, GLenum pname, GLfloat param).
```

Параметр `face` определяет грань, для которой устанавливаются свойства, может принимать одно из следующих значений:

- `GL_BACK` (нелицевая грань);
- `GL_FRONT` (лицевая грань);
- `GL_FRONT_AND_BACK` (обе грани).

Параметр `pname` определяет свойство материала, которое будет установлено, может принимать следующие значения:

- `GL_AMBIENT` (фоновая компонента);
- `GL_DIFFUSE` (диффузная компонента);
- `GL_AMBIENT_AND_DIFFUSE` (обе компоненты);
- `GL_SPECULAR` (зеркальная компонента);
- `GL_EMISSION` (эмиссионная компонента);
- `GL_SHININESS` (шероховатость материала).

Параметром `param` передается значение выбранной компоненты в виде вещественного массива из четырех компонент (для компоненты `GL_SHININESS` – число из диапазона $[0,128]$, а не массив).

ЗАДАНИЕ КАМЕРЫ

Как таковой, камеры в OpenGL нет, ибо она считается зафиксированной в точке $(0,0,0)$ с направлением взгляда $z = -1$.

В матричном виде преобразование камеры можно записать следующим образом через позицию камеры P и вектора направлений вправо $\vec{r}$, вверх $\vec{u}$ и взгляда $\vec{d}$:

$$LookAt = \begin{bmatrix} r_x & r_y & r_z & 0 \\ u_x & u_y & u_z & 0 \\ d_x & d_y & d_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -p_x \\ 0 & 1 & 0 & -p_y \\ 0 & 0 & 1 & -p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

ИМИТАЦИЯ КАМЕРЫ

В библиотеке GLU введена функция `gluLookAt`, реализующая камеру через модельно-видовые преобразования:

```
void gluLookAt(
GLdouble eyex, GLdouble eyey, GLdouble eyez,
GLdouble centerx, GLdouble centery, GLdouble centerz,
GLdouble upx, GLdouble upy, GLdouble upz),
```

где:

точка (eyex, eyey, eyez) – позиция наблюдателя,

точка (centerx, centery, centerz) – позиция наблюдения,

вектор (upx, upy, upz) – направление вверх.

Координаты данной функции задаются **в текущей** системе координат.

МОДЕЛЬНО-ВИДОВЫЕ ПРЕОБРАЗОВАНИЯ

Поскольку результат модельно-видовых преобразований в общем случае является вещественным, поэтому у команд OpenGL, реализующих данные преобразования, есть только по два варианта – с одинарной и двойной точностью.

Операция сдвига на вектор $(x, y, z)^T$ выполняется командой `glTranslate`:

```
void glTranslatef ( GLfloat x, GLfloat y, GLfloat z ),
void glTranslated ( GLdouble x, GLdouble y, GLdouble z ).
```

Операция масштабирования на вектор $(x, y, z)^T$ выполняется командой `glScale`:

```
void glScalef ( GLfloat x, GLfloat y, GLfloat z ),
void glScaled ( GLdouble x, GLdouble y, GLdouble z ).
```

Операция поворота на угол `angle` вокруг вектора $(x, y, z)^T$ выполняется командой `glRotate`:

```
void glRotatef (GLfloat angle,
                GLfloat x, GLfloat y, GLfloat z ),
void glRotated (GLdouble angle,
                GLdouble x, GLdouble y, GLdouble z ).
```

Все эти операции необходимо задавать **до** операторных скобок, поскольку задаваемые в них вершины умножаются на матрицу преобразований, установленных вышеописанными командами.

ЗАДАНИЕ МАТРИЦ ПРОЕКЦИЙ

Матрица орографической проекции задается командой

```
void glOrtho (GLdouble left, GLdouble right,
              GLdouble bottom, GLdouble top,
              GLdouble zNear, GLdouble zFar).
```

Матрица перспективной проекции задается командой

```
glFrustum  (GLdouble left, GLdouble right,
            GLdouble bottom, GLdouble top,
            GLdouble zNear, GLdouble zFar).
```

В библиотеке GLU предусмотрен альтернативный способ задания перспективной проекции:

```
void gluPerspective ( GLdouble fovy, GLdouble aspect,
                      GLdouble zNear, GLdouble zFar),
```

где:

`fovy=2*arctg(top/near)` – угол $[0°, 180°]$ видимости по вертикали,

`aspect=right/top` – пропорции экрана.

На рис.41 представлен куб в режиме ортографической и перспективной проекций, куб пришлось сдвинуть назад по оси OZ.

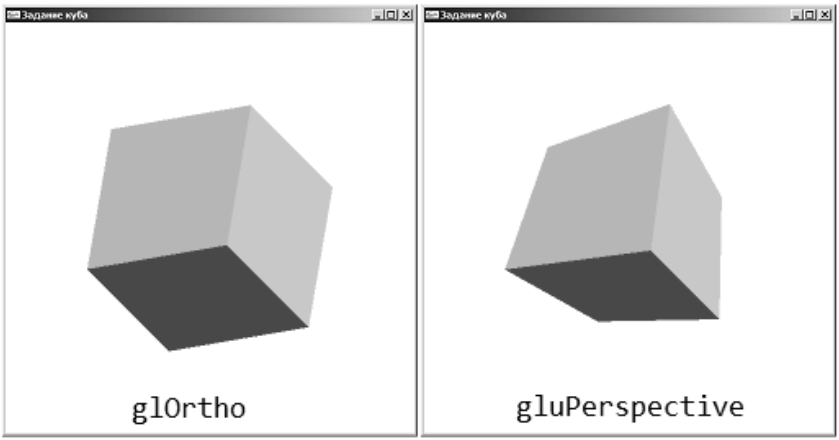

Рис.41. Куб в ортографической и центральной проекциях.

ОПЕРАЦИИ С МАТРИЦАМИ

Для ускорения работы в OpenGL матрицы хранятся не по строкам, а по столбцам, в виде одномерного вещественного массива размерностью 16 элементов.

Работа с матрицами начинается с выбора того типа матрицы, к которой будут применяться все последующие операции, командой `glMatrixMode:`

```
void glMatrixMode (GLenum mode),
```

параметр `mode` может принимать следующие значения:

- `GL_MODELVIEW` (для модельно-видовой матрицы);
- `GL_PROJECTION` (для проекционной матрицы)
- `GL_TEXTURE` (для текстурной матрицы).

Загрузка текущей матрицы выполняется командой `glLoadMatrix:`

```
void glLoadMatrixd (const GLdouble *m);
void glLoadMatrixв (const GLfloat *m);
```

Сохранение матрицы выполняется командой `glGet`, первый параметр которой для модельно-видовой матрицы должен быть равен `GL_MODELVIEW_MATRIX`, а для перспективной – `GL_PROJECTION:`

```
void glGetDoublev(GLenum pname, GLdouble *m);
void glGetFloatv (GLenum pname, GLfloat *m).
```

Сброс матрицы в начальное (единичное) состояние осуществляется командой

```
void glLoadIdentity (void).
```

Умножение матрицы слева осуществляется командой `glMultMatrix:`

```
void glMultMatrixf (const GLfloat *m);
void glMultMatrixd (const GLdouble *m);
```

Сохранение матрицы в стеке и восстановление обратно выполняется следующими командами:

```
void glPushMatrix (void);
void glPopMatrix (void).
```

КОНВЕЙЕР СИСТЕМ КООРДИНАТ

Визуализация объектов разбивается на несколько этапов, в процессе которых последовательно преобразуется система координат:

1. Перевод в однородные координаты всех вершин в момент их задания:

$$(x, y, z) \to (x, y, z, 1).$$

2. Умножение на модельную матрицу.

$$(x, y, z, 1) \to (x^m, y^m, z^m, 1).$$

3. Умножение на видовую матрицу (получение видовых координат):

$$(x^m, y^m, z^m, 1) \to (x^v, y^v, z^v, 1).$$

4. Умножение на проекционную матрицу (получение усеченных координат):

$$(x^v, y^v, z^v, 1) \to (x^p, y^p, z^p, w).$$

5. Возвращение в декартовы координаты (нормализация).

$$(x^p, y^p, z^p, w) \to \left(\frac{x'}{w}, \frac{y'}{w}, \frac{z'}{w}\right).$$

6. Переход в оконные координаты через функции порта просмотра и буфера глубины.

Примечание. После проецирования из рассмотрения отбрасываются все примитивы, чьи координаты не попадают в область куба видимости. После нормализации все z-координаты находятся в диапазоне $[-1,1]$, что соответствует ближней и дальней плоскостям отсечения.

ФУНКЦИИ ПОСТРОЕНИЯ 3D ОБЪЕКТОВ

В чистой OpenGL не предусмотрены такие сложные примитивы, как сфера, цилиндр, тор. Данные объекты реализованы в дополнительных библиотеках – таких, как GLU и GLUT.

При создании подобного рода объектов для однозначности восприятия полезно включить освещение и буфер глубины. Также в ряде случаев полезно дополнительно нарисовать этот же самый объект в каркасном режиме и чуть большего размера.

ПРИМИТИВЫ БИБЛИОТЕКИ GLU

В данной библиотеке реализованы следующие квадратичные объекты:

- сфера;
- цилиндр;
- диск (круговое кольцо);
- частичный диск;
- сплайны (здесь не рассматриваются).

Для отрисовки такого примитива необходимо создать (а потом не забыть удалить) специальный объект типа `GLUquadricObj`. Создание объекта выполняется командой `gluNewQuadric`, а удаление – командой `gluDeleteQuadric`. Созданный таким образом примитив будет располагаться в центре текущей системы координат.

Для настройки объектов предоставляются возможности изменения стиля рисования с помощью команды `gluQuadricDrawStyle`, первый параметр которой задает объект, для которого выполняется эта команда, а второй – стиль рисования: сплошной (`GLU_FILL`), каркасный (`GLU_LINE`) или точечный (`GLU_POINT`). Для изменения учета нормалей применяются команды `gluQuadricOrientation` и `gluQuadricNormals`.

На рис.42 приведен пример сферы, отрисованной в сплошном, каркасном и точечном режимах, для удобства восприятия повернутой на угол –45 по вектору $(1,1,0)^T$.

Поскольку сфера собирается из стандартных плоских примитивов, при ее задании необходимо указать уровень дискретизации по и вокруг оси OZ (на рис.43 приведен пример генерации сферы с малым уровнем разбиения).

Пример кода для создания каркасной сферы:

```
GLUquadricObj *obj = gluNewQuadric();
 gluQuadricDrawStyle ( obj, GLU_LINE );
 gluSphere ( obj, 200, 20, 20 );
gluDeleteQuadric ( obj );
```

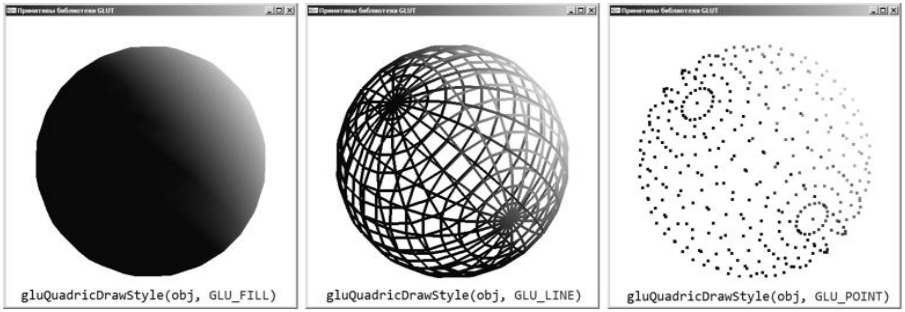

Рис.42. Библиотека GLU: сфера в различных режимах отрисовки.

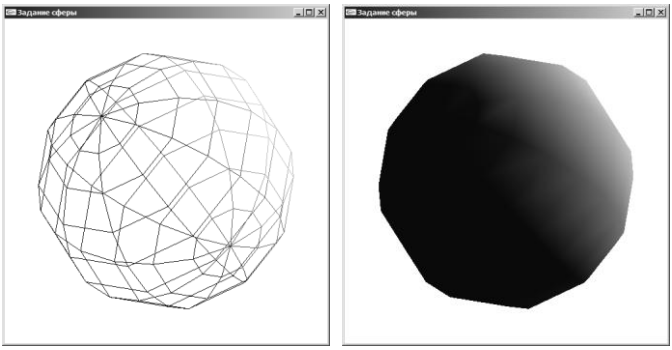

Рис.43. Библиотека GLU: малополигональная сфера.

ПРИМИТИВЫ БИБЛИОТЕКИ GLUT

Список объектов этой библиотеке приведен в табл.19, а на рис.44 приведен пример отрисовки специального примитива «чайник». Примеры отображения некоторых объектов в режиме каркаса приведены на рис.45.

Таблица 19

Объекты библиотеки GLUT

Сплошной режим	Каркасный режим	Описание
`glutSolidSphere`	`glutWireSphere`	Сфера
`glutSolidCube`	`glutWireCube`	Куб
`glutSolidCone`	`glutWireCone`	Конус
`glutSolidTorus`	`glutWireTorus`	Тор
`glutSolidTetrahedron`	`glutWireTetrahedron`	Тетраэдр единичного размера
`glutSolidOctahedron`	`glutWireOctahedron`	Октаэдр единичного размера
`glutSolidDodecahedron`	`glutWireDodecahedron`	Додекаэдр единичного размера
`glutSolidlcosahedron`	`glutWirelcosahedron`	Икосаэдр единичного размера
`glutSolidTeapot`	`glutWireTeapot`	Чайник

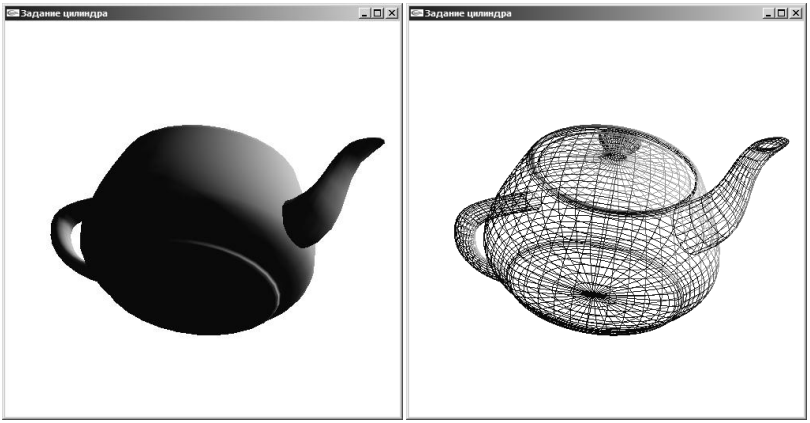

Рис.44. Библиотека GLUT: сплошной и каркасный чайник.

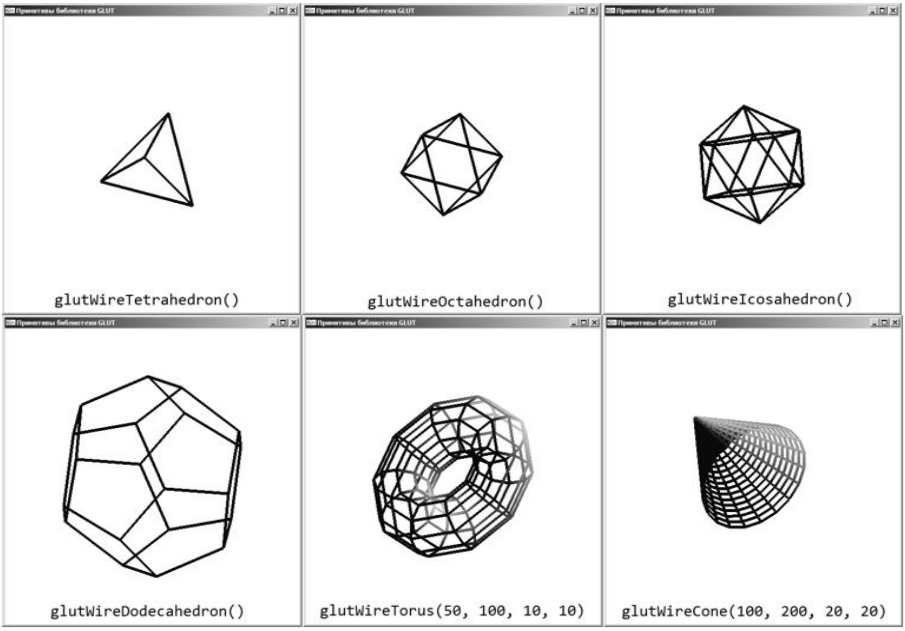

Рис.45. Библиотека GLUT: дополнительные примитивы в режиме каркаса.

ПРИЛОЖЕНИЕ.
ЛИСТИНГ ПРОГРАММЫ «ОТРИСОВКА КУБА»

```
#include "glut.h"


/* Разрмер окна */
int Width = 1000;
int Height = 1000;

/* Задание координат вершин всех 6 граней куба */
float x0 = 0, x1 = 100;
float y0 = 0, y1 = 100;
float z0 = 0, z1 = 100;
float xyz[6*4*3] = {
       x0,y0,z0, x0,y1,z0, x1,y1,z0, x1,y0,z0, // грань z=0
       x0,y0,z1, x0,y1,z1, x1,y1,z1, x1,y0,z1, // грань z=1
       x0,y0,z1, x0,y1,z1, x0,y1,z0, x0,y0,z0, // грань x=0
       x1,y0,z1, x1,y1,z1, x1,y1,z0, x1,y0,z0, // грань x=1
       x0,y0,z1, x0,y0,z0, x1,y0,z0, x1,y0,z1, // грань y=0
       x0,y1,z1, x0,y1,z0, x1,y1,z0, x1,y1,z1  // грань y=1
};


/* Задание цветов вершин */
float r0 = 0, g0 = 0, b0 = 1;
float r1 = 0, g1 = 1, b1 = 0;
float r2 = 1, g2 = 0, b2 = 0;
float r3 = 0, g3 = 1, b3 = 1;
float r4 = 1, g4 = 0, b4 = 1;
float r5 = 1, g5 = 1, b5 = 0;
float rgb[6*4*3] = {
       r0,g0,b0, r0,g0,b0, r0,g0,b0, r0,g0,b0, // грань z=0
       r1,g1,b1, r1,g1,b1, r1,g1,b1, r1,g1,b1, // грань z=1
       r2,g2,b2, r2,g2,b2, r2,g2,b2, r2,g2,b2, // грань x=0
       r3,g3,b3, r3,g3,b3, r3,g3,b3, r3,g3,b3, // грань x=0
       r4,g4,b4, r4,g4,b4, r4,g4,b4, r4,g4,b4, // грань y=0
       r5,g5,b5, r5,g5,b5, r5,g5,b5, r5,g5,b5  // грань y=1
};
```

```
/* Задание координат нормалей вершин */
float nor[6*4*3] = {
       0, 0,-1,  0, 0,-1,  0, 0,-1,  0, 0,-1, // грань z=0
       0, 0, 1,  0, 0, 1,  0, 0, 1,  0, 0, 1, // грань z=1
      -1, 0, 0, -1, 0, 0, -1, 0, 0, -1, 0, 0, // грань x=0
       1, 0, 0,  1, 0, 0,  1, 0, 0,  1, 0, 0, // грань x=1
       0,-1, 0,  0,-1, 0,  0,-1, 0,  0,-1, 0, // грань y=0
       0, 1, 0,  0, 1, 0,  0, 1, 0,  0,-1, 0  // грань y=1
};

/* Задание координат источника света №0 */
float light0_positionT[] = { 0, 0, 0, 1 };


/* Отрисовка куба двумя вариантами */
bool ModePointer = false; // режим Begin/End или VertexPointer
void DrawCube()
{
       if( ModePointer )
       {
              glEnableClientState(GL_VERTEX_ARRAY);
              glEnableClientState(GL_COLOR_ARRAY);
              glEnableClientState(GL_NORMAL_ARRAY);
              glVertexPointer ( 3, GL_FLOAT, 0, xyz);
              glColorPointer  ( 3, GL_FLOAT, 0, rgb);
              glNormalPointer (    GL_FLOAT, 0, nor);
              glDrawArrays(GL_QUADS, 0,6*4);
              glDisableClientState(GL_NORMAL_ARRAY);
              glDisableClientState(GL_COLOR_ARRAY);
              glDisableClientState(GL_VERTEX_ARRAY);
       }
       else
       {
              glBegin(GL_QUADS);
              for( int i = 0; i < 6 * 4; i++ )
              {
                     glColor3fv(&rgb[3 * i]);
                     glVertex3fv(&xyz[3 * i]);
              }
              glEnd();
       }
}
```

```
/* Функция изменения содержимого окна */
void Display(void)
{
	glClearColor(1, 1, 1, 1);
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

	glRotatef( -45, 1, 1, 0); // поворот куба как при изометрии

	DrawCube();

	glFinish();
}

/* Функция изменения размеров окна */
void Reshape(GLint w, GLint h)
{
	Width = w;    Height = h;
	glViewport(0, 0, w, h);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	glOrtho(-w/2, w/2, -h/2, h/2, -1000, 1000);
	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
}

/* Головная программа */
void main(int argc, char *argv[])
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_RGB | GLUT_DEPTH);
	glutInitWindowSize(Width, Height);
	glutCreateWindow("Проекции");
	glutDisplayFunc(Display);
	glutReshapeFunc(Reshape);

	glEnable(GL_DEPTH_TEST);
	glEnable(GL_LIGHTING);
	glEnable(GL_LIGHT0);
	glLightModelf(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE);
	glLightfv(GL_LIGHT0, GL_POSITION, light0_positionT);

	glutMainLoop();
}
```

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Задорожный А.Г.* Введение в двумерную компьютерную графику с использованием библиотеки OpenGL: учебное пособие / А. Г. Задорожный, Д. В. Вагин, Ю. И. Кошкина.– Новосибирск : Изд-во НГТУ, 2018. – 102 с. – Режим доступа: https://elibrary.nstu.ru/source?id=73094.

2. *Залогова Л.А.* Компьютерная графика / Л. Залогова. – М., 2005. – 319 с.

3. *Дегтярев В.М.* Инженерная и компьютерная графика : учебник / В.М. Дегтярев, В.П. Затыльникова. – М., 2010. – 238 с.

4. *Порев В.Н.* Компьютерная графика / В. Порев. – СПб., 2005. – 428 с.

5. *Евченко А.И.* OpenGL и DirectX: программирование графики / А.И. Евченко. – СПб., 2006. – 349 с.

6. *Пономаренко С.И.* Пиксел и вектор. Принципы цифровой графики. – СПб., 2002. – 477 с.

7. *Петров М.Н.* Компьютерная графика: учеб. пособие / М.Н. Петров, В.П. Молочков. - СПб., 2002. – 735 с.

8. *Глушаков С.В.* Компьютерная графика: учебный курс / С.В. Глушаков, Г.А. Кнабе. – Харьков, 2001. – 500 с.

9. *Коцюбинский А.О.* Компьютерная графика: практич. пособие. – М., 2001. – 750 с.

СОДЕРЖАНИЕ

Задорожный Александр Геннадьевич
Персова Марина Геннадьевна
Кошкина Юлия Игоревна

ВВЕДЕНИЕ В ТРЕХМЕРНУЮ КОМПЬЮТЕРНУЮ ГРАФИКУ С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕКИ OPENGL

Учебное пособие

В авторской редакции

Выпускающий редактор *И.П. Брованова*
Дизайн обложки *А.В. Ладыжская*

Налоговая льгота – Общероссийский классификатор продукции
Издание соответствует коду 95 3000 ОК 005-93 (ОКП)

Подписано в печать 05.12.2018. Формат 60 × 84 1/16. Бумага офсетная.
Тираж 100 экз. Уч.-изд. л. 5,81. Печ. л. 6,25. Изд. №315. Заказ № 11
Цена договорная

Отпечатано в типографии
Новосибирского государственного технического университета
630073, г. Новосибирск, пр. К. Маркса, 20