Unity. Precision. Perfection.

КОНСПЕКТ ЛЕКЦИЙ
по дисциплине «Компьютерная графика»

Лектор: Герасимова Тамара Владимировна

Страниц: 55

Последнее обновление: 6 сентября 2019 г.

Автор: Корытов Павел, 6304

Санкт-Петербург

2019

Содержание

1. 2D преобразования

1.1. Геометрические преобразования в компьютерной графике

Преобразования в КГ:

- Проекционные
- Геометрические
  - Афинные

В машинной графике представлена большая группа геометрических преобразований (аффинная группа): перенос, поворот, масштабирование. Геометрические преобразования позволяют изменять и визуализировать объект.

Компьютерная графика имеет дело со множеством геометрических объектов, таких как точки, многоугольники и многогранники. Все разнообразие геометрических объектов можно свести к ограниченному множеству простейших сущностей. Нам понадобятся для этого три базовых типа — скаляры, точки и векторы.

Существуют, по крайней мере, три варианта определения этих сущностей, в зависимости от того, с какой точки зрения их рассматривать:

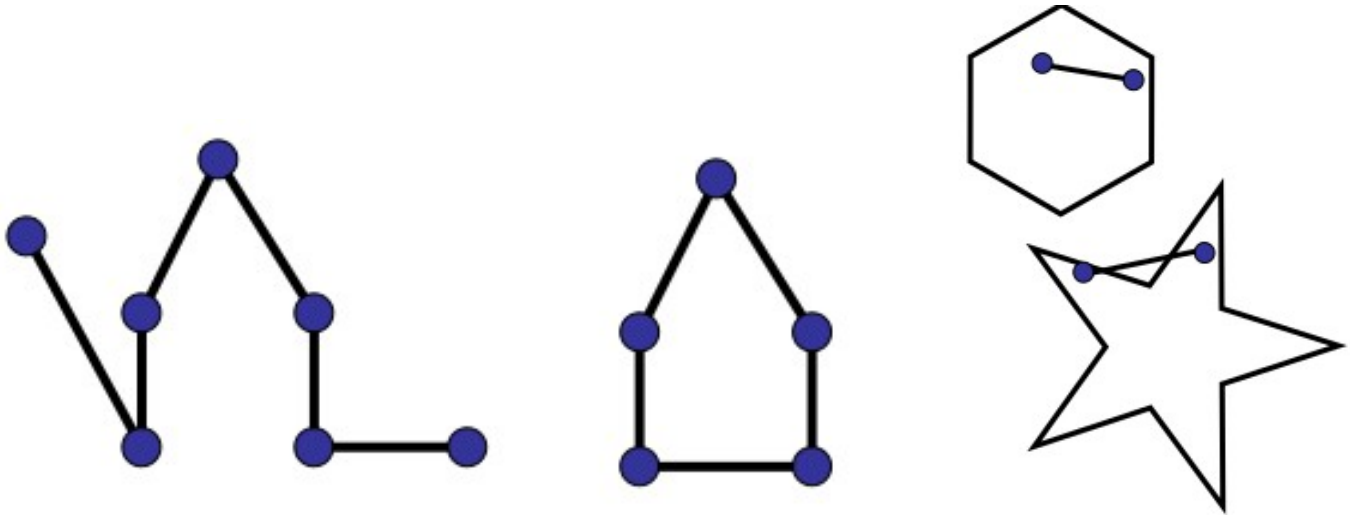
- чисто математической (формальной)
- геометрической
- с точки зрения программной реализации

В конечном счете нам придется иметь дело со всеми тремя вариантами.

1.2. Описание 2D объектов

Lines & Polylines. Линия, нарисованная по заданным точкам. Соединенные конечные точки дают **замкнутую полилинию** или **полигон**

Выпуклый и вогнутый многоугольники Выпуклый: Для каждой пары точек в многоугольнике линия между ними полностью содержится в многоугольнике.
Вогнутый: Не выпуклый: произвольные две точки соединенные в многоугольнике линией не полностью содержавшаяся в многоугольнике



(a) Линии (b) Многоугольники

Рисунок 1 – Графические примитивы

Окружность. Состоит из всех точек равноудаленных от единой точки – центра окружности. $r^2 = x^2 + y^2$. Окружность может быть аппроксимирована как полигон с множеством сторон.

Рисунок 2 – Аппроксимация окружности

Эллипс. Окружность, промасштабированная вдоль осей x или y.

1.3. Введение в вычислительную геометрию

1.3.1. Аффинные и проективные преобразования на плоскости и в 3D-пространстве

Affinis (лат.) – смежный, соседний, родственный Признаки:

- прямые линии остаются прямыми
- параллельные - параллельными

Примеры:

- подобие (сохраняются углы)
- сжатие/растяжение (без подобия)

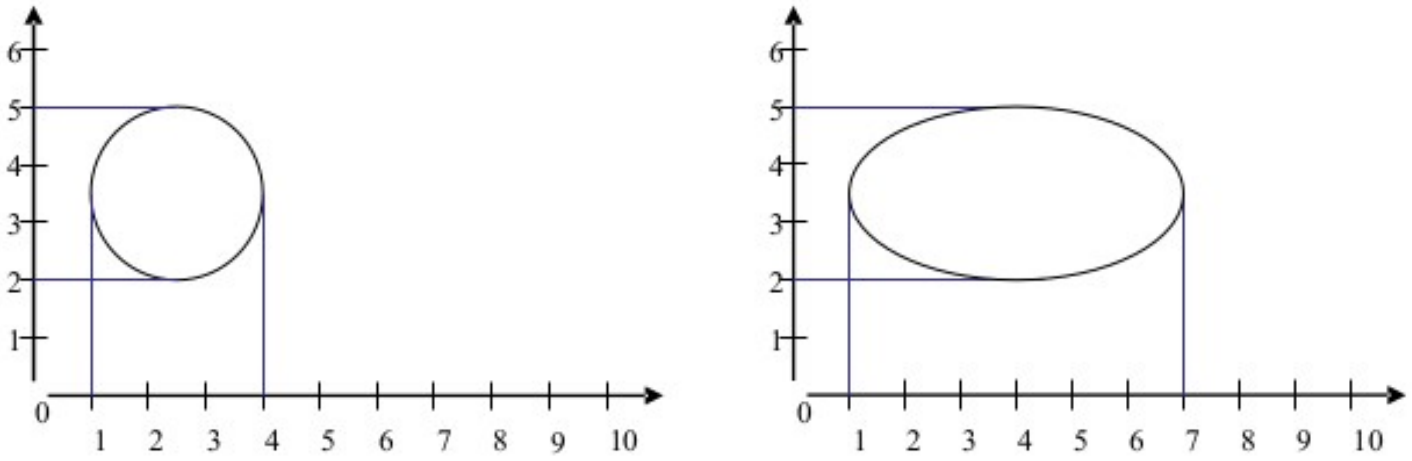


Рисунок 3 – Эллипс

- движение (перенос и вращение – ортогональные преобразования 1-го рода)

А.П. можно определить с помощью невырожденных линейных преобразований точек пространства. **А.П.** - это подмножество проективных преобразований (П.П.). При **П.П.** параллельность может нарушаться. Сохранение параллельности является «границей», при нарушении которой **А.П.** превращаются в П.П.

1.3.2. Элементарные преобразования на плоскости

Универсальная, но чаще всего неудобная форма записи:

$$\begin{cases} x' = ax_1 + by_1 + c_1 \\ y' = ax_2 + by_2 + c_2 \end{cases} ; \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix} \neq 0$$

Более пригодная форма для использования в алгоритмах визуализации «собирается» из элементарных преобразований:

- поворот вокруг начала координат на угол ϕ
- растяжение (сжатие) вдоль координатных осей
- отражение относительно координатных осей
- перенос

Теоремы аффинной геометрии идентичны теоремам евклидовой геометрии, в которой важными понятиями являются параллельность и соотношение между параллельными линиями.

Аффинные преобразования формируют удобную подсистему билинейных преобразований, так как произведение двух аффинных преобразований также является аффинным. Преобразования связаны с некоторыми изменениями объекта.

Базовыми являются следующие преобразования:

- Перенос (Move/Translation);
- Поворот (Rotate);
- Масштабирование (Scale);
- Отражение (Reflect);
- Сдвиг по одной из координат (Shear).

Преобразование объекта = преобразование всех его точек; преобразование полигона = преобразование всех его вершин

1.3.3. Умножение матрицы на вектор

$$M_v = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix} \cdot \begin{bmatrix} v_x \\ v_y \\ x_z \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

Линейное преобразование вектора преобразует один вектор в другой и сохраняет линейные комбинации Любое линейное преобразование м.б. представлено с помощью матриц:

- Масштабирование (scaling)
- Поворот (rotation)
- Перенос (translation)

1.3.4. О порядке перемножения матриц и векторов

1. $b = a \cdot M$

   Неудобно: Не совпадает расположение элементов в матрице и коэффициентов в записи системы уравнений.

   Удобно: Совпадают порядок перемножения матриц и порядок выполнения суперпозиции преобразований:

   $P_4(P_3(P_2(P_1))) \to M_1 \cdot M_2 \cdot M_3 \cdot M_4$
2. $b^T = M^T \cdot a^T$

   Порядок перемножения матриц обратен порядку преобразований в записи их суперпозиции, матрицы транспонированы по отношению к их записи в (1):

$P_4(P_3(P_2(P_1))) \to M_4^T \cdot M_3^T \cdot M_2^T \cdot M_1^T$

Удобно: Совпадает расположение элементов в матрице и коэффициентов в записи систем

Примечания:

1. Строго говоря, следовало бы считать ^{T} ввиду запись матрицы по строкам, однако во избежание нагромождения символов, считаем здесь и далее запись матрицы по столбцам не транспонированной
2. В математической литературе чаще используется порядок перемножения (1)
3. В литературе по API OpenGL частенько используется порядок перемножения (2)
4. «Удобства» и «неудобства» относительны (многое зависит от контекста алгоритма)

1.4. 2D перенос (Translation)

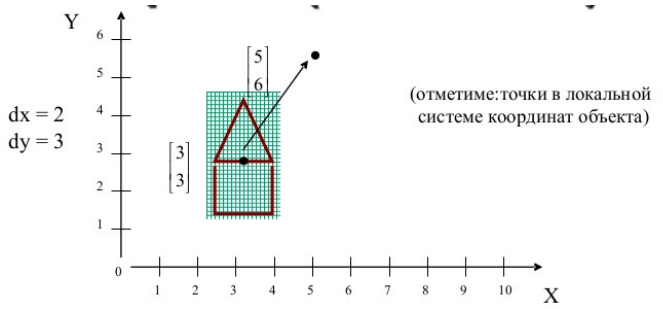


Рисунок 4 – Перенос

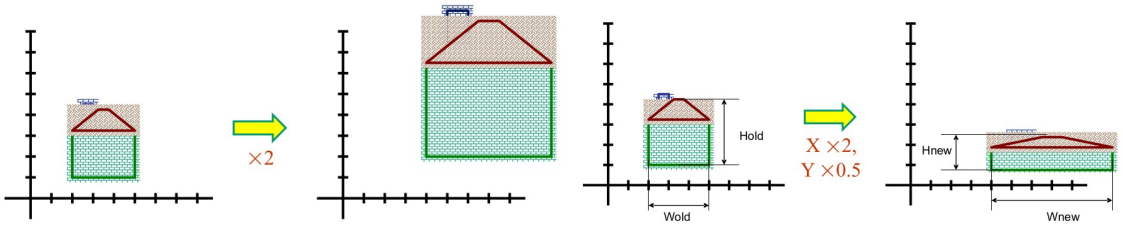
$$v' = v + t, \text{где } v = \begin{bmatrix} x \\ y \end{bmatrix} v' = \begin{bmatrix} x' \\ y' \end{bmatrix} t = \begin{bmatrix} \delta x \\ \delta y \end{bmatrix}$$

$$\begin{cases} x' = x + \delta x \\ y' = y + \delta y \end{cases}$$

Длины и углы сохраняются.

1.5. 2D масштабирование

Масштабирование координаты означает умножать каждый из ее компонентов на скаляр. **однородное** масштабирование означает, что этот скаляр - одинаковый для всех компонентов. Соответственно, при **неоднородном** скаляры различны для каждого компонен-



(a) Однородное (b) Неоднородное

Рисунок 5 – Масштабирование

та.

$$v' = S \cdot v; \text{где } v = \begin{bmatrix} x \\ y \end{bmatrix}, v' = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

$$\begin{cases} x' = s_x x \\ y' = s_y y \end{cases} ; S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

В таком случае длины и углы не сохраняются. Кроме того, имеется побочный эффект - на картинке дом перемещается относительно центра

1.6. 2D поворот

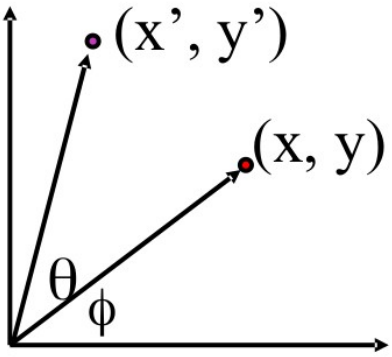


Рисунок 6 – Поворот

Выразим x, y, x', y' в полярных координатах:

$$\begin{cases} x = r\cos\phi \\ y = r\sin\phi \end{cases} ; \begin{cases} x' = r\cos\phi + \theta \\ y' = r\sin\phi + \theta \end{cases}$$

Раскроем последнюю систему:

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

После замены получается следующая система:

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

Это легко выразить в матричной форме:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Если что объект не расположен в центре СК а его нужно масштабировать и вращать, переместитесь в НК, масштабируйте и/или вращайте его в местной системе координат, затем переместите объект обратно. Эта последовательность предлагает потребность составить последовательные (составные) преобразования.

1.7. Однородные координаты

Однородные координаты - это математический механизм, связанный с определением положения точек в пространстве. Привычный аппарат декартовых координат, не подходит для решения некоторых важных задач в силу следующих соображений:

- В декартовых координатах невозможно описать бесконечно удаленную точку. А многие математические и геометрические концепции значительно упрощаются, если в них используется понятие бесконечности. Например, "бесконечно удаленный источник света".
- С точки зрения алгебраических операций, декартовы координаты не позволяют провести различия межу точками и векторами в пространстве. Действительно, $(1, 2, 5)$ - это направление или точка?
- Невозможно использовать унифицированный механизм работы с матрицами для выражения преобразований точек. С помощью матриц 3×3 можно описать вращение и масштабирование, однако описать смещение $(x = x + a)$ нельзя.

- Аналогично, декартовы координаты не позволяют использовать матричную запись для задания перспективного преобразования (проекции) точек.

Для решения этих проблем используются однородные координаты.

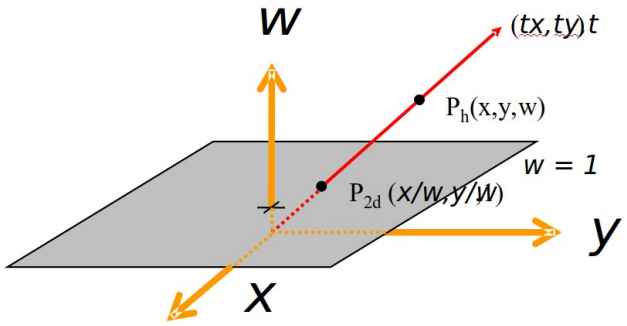


Рисунок 7 – Однородные координаты

P_{2d} - пересечение линии, определённой P_h с плоскостью $w = 1$. Композицию преобразований трудно выразить, пока перенос не выражен как матричное умножение. Но, если мы выразим точки в однородных координатах, то все три преобразования можно реализовать с помощью умножений.

Однородные координаты были введены в геометрии и впоследствии использованы в графике. С однородными координатами и преобразованиями над ними работают многие пакеты графических подпрограмм и некоторые дисплейные процессоры.

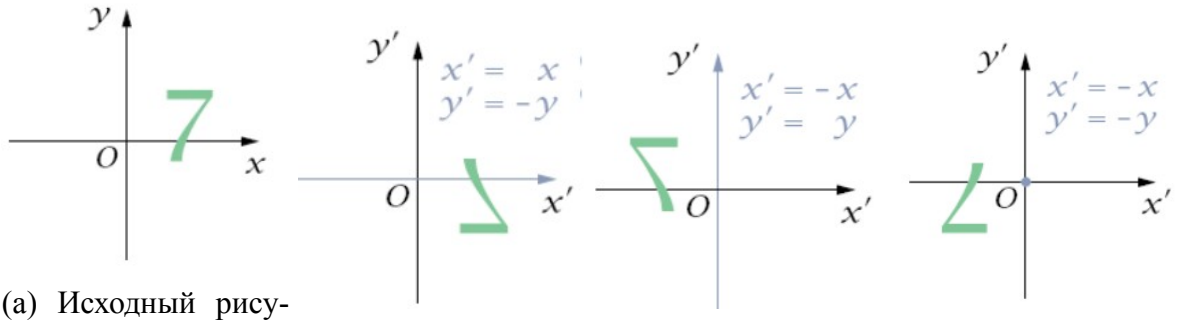
В одних случаях эти координаты используются прикладной программой непосредственно при задании параметров для графического пакета, в других — применяются лишь внутри самого пакета и недоступны для программиста. Теперь матрицы преобразований выглядят так:

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}; \; S(a,b) = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{bmatrix}; \; T(x,y) = \begin{bmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix}$$

Перенос работает следующим образом:

$$\begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x+a \\ y+b \\ 1 \end{bmatrix}$$

1.8. Отражение



(a) Исходный рисунок (b) Вдоль x (c) Вдоль y (d) Вдоль обоих

Рисунок 8 – Отражение

$$\begin{bmatrix} x'_h \\ y'_h \\ h \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_h \\ y_h \\ z_h \end{bmatrix}$$

Соответственно, матрица отражения вдоль x:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

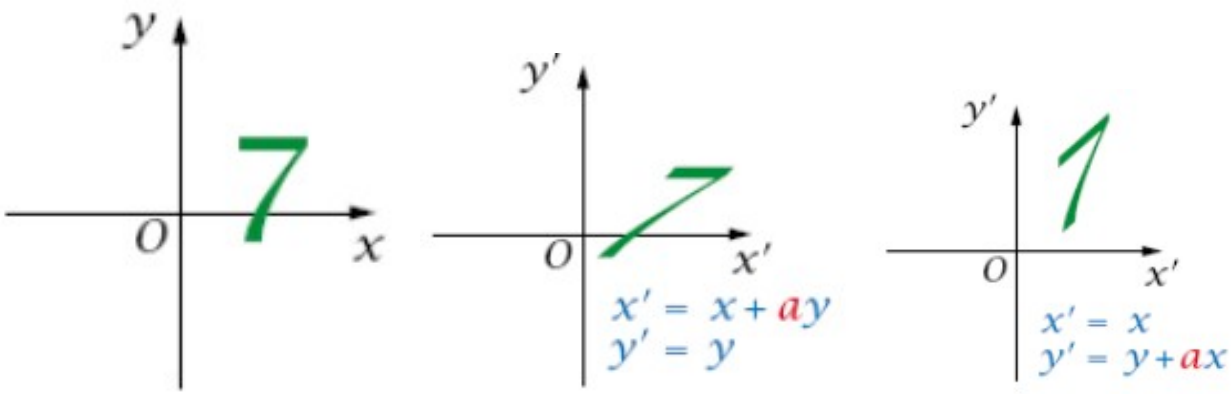
Для отражения вдоль оси y или обеих осей используются следующие матрицы:

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}; \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

1.9. Смещение (Shear)

Матрицы смещения вдоль x и y:

$$\begin{bmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}; \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



(a) Исходный рисунок (b) Вдоль x (c) Вдоль y

Рисунок 9 – Смещение

1.10. Порядок выполнения преобразований

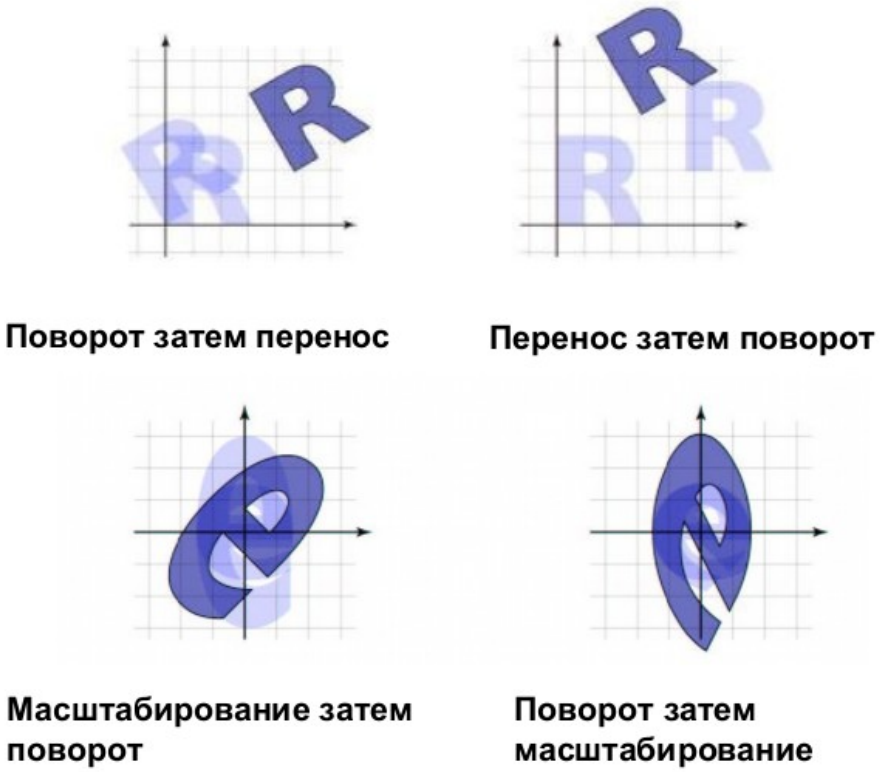


Рисунок 10 – Некоммутативность

Важно, что преобразования не коммутативны.

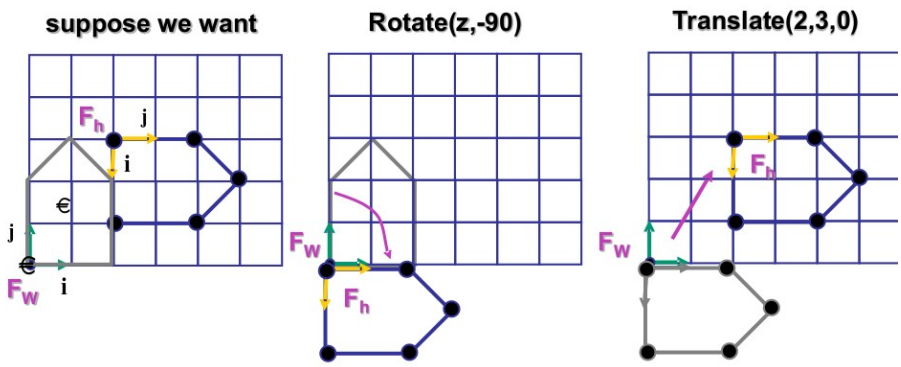


Рисунок 11 – Композиция преобразований

1.11. Создание преобразований

Для получения нужного результата в данном случае нужно применить два преобразования подряд:

$$p' = R(z, -90^\circ)p;\ p'' = T(2,3,0)p' \ \rightarrow\ p'' = T(2,3,0)R(z,90^\circ)p = TRp$$

1.11.1. Поворот вокруг произвольной точки O

$$C(O,\theta) = T(-O_x, -O_y)R(\theta)T(O_x, O_y)$$

1.11.2. Общий подход к композиции преобразований

- Преобразование геометрии объектов в такую систему координат, где операция становится более простой;
- Выполните операцию;
- Преобразуйте геометрию объектов сцены назад к первоначальной системе координат

2. 3D преобразования

2.1. Системы координат, используемые в графике

Координаты экрана - физическое местоположение на определённом устройстве вывода (отображения). Определяют местоположение пикселей.

- Изменения трудны для прикладной программы
- Приводит к зависимому от устройства кода

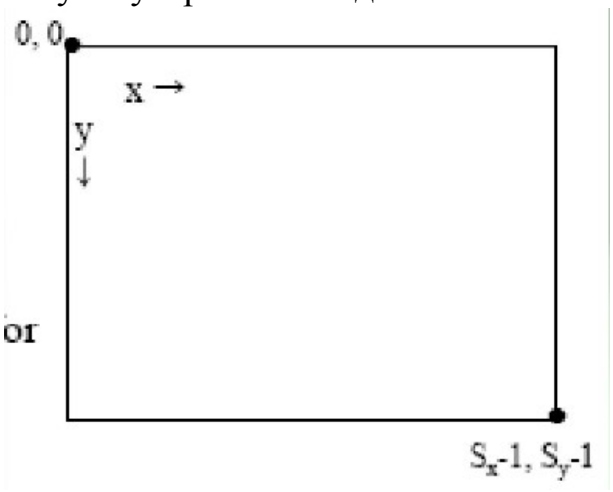


Рисунок 12 – Коррдинаты экрана

Координаты окна - экранные координаты относительно центра окна. Система GUI дает приложению позицию на экране; оконная система преобразует координаты окна в экранные координаты

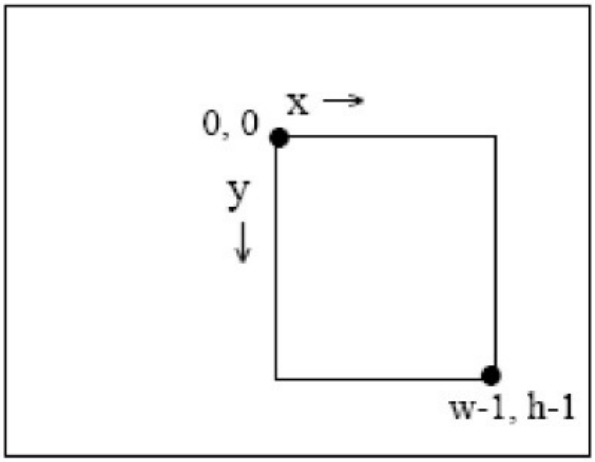


Рисунок 13 – Координаты окна

Нормализованные координаты устройства (NDC) - независимая от устройства абстракция окна экрана.

- Абстракция координат
- Значение координат - вещественные числа с плавающей запятой от 0 до 1

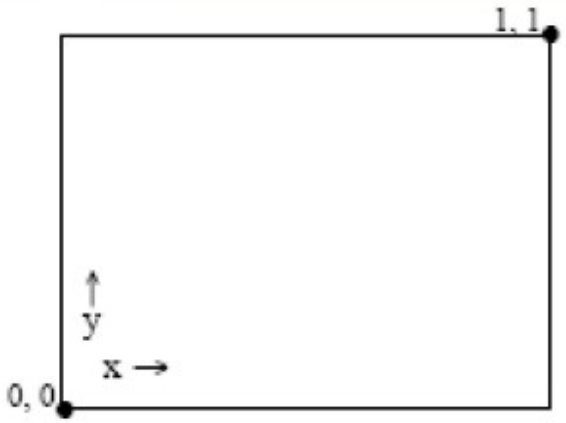


Рисунок 14 – Нормализованные координаты

Мировые координаты - родовая система координат, которую может использовать любое приложение, чтобы определить объекты для показа

- Приложение должно быть в состоянии определить объекты в форму, естественную для области. Например, от 0 до 10^4 км для карты России
- Мировые координаты - 2D или 3D значение с плавающей точкой, которые могут представить местоположения и размеры объекта, который будет показан.

2.2. Преобразования в мировом пространстве

Шаги преобразования:

- Объект в МК
- Окно переносится в начало координат
- Масштабирование окна под размер поля вывода (нормализация в uv)
- Перенос в окончательную позицию

2.3. Использование нормализованных координат

Трехмерная точка в трехмерном пространстве $P(x, y, z)$ представится четырёхмерным вектором $\begin{bmatrix} x & y & z & 1 \end{bmatrix}$ или $\begin{bmatrix} X & Y & Z & W \end{bmatrix}$

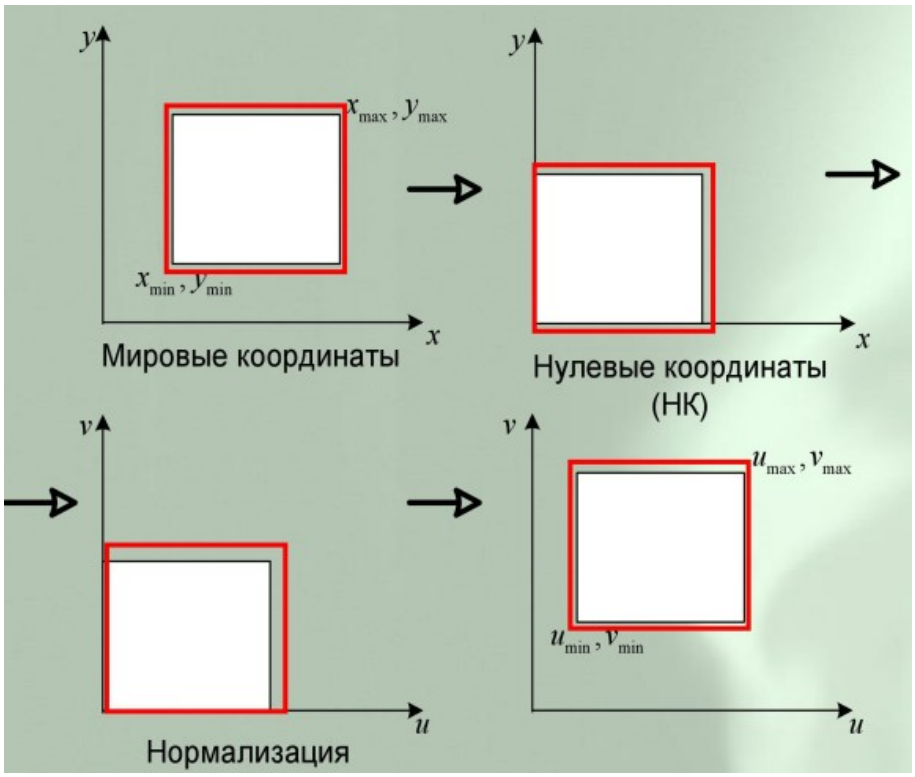


Рисунок 15 – Преобразования координат

Преобразования из однородных координат описываются соотношениями:

$$\begin{bmatrix} X & Y & Z & W \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \cdot T$$

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} \frac{X}{W} & \frac{Y}{W} & \frac{Z}{W} & 1 \end{bmatrix}$$

Где T - некоторая матрица преобразования. Обобщенная матрица преобразования 4×4 для трехмерных координат имеет вид:

$$T = \begin{bmatrix} a & b & c & p \\ d & e & f & q \\ h & i & j & r \\ I & m & n & s \end{bmatrix}$$

Эта матрица может быть представлена в виде четырёх отдельных частей:

$$\begin{array}{c|c} [3 \times 3] & [3 \times 1] \\ \hline [1 \times 3] & [1 \times 1] \end{array}$$

Матрица 3×3 осуществляет линейное преобразование в виде изменения масштаба, сдви-

га и вращения. Матрица-строка 1×3 производит перенос, а матрица-столбец 3×1 - преобразование в перспективе. Последний скалярный элемент выполняет общее изменение масштаба.

Важно, что 3D-преобразования **некоммутативны**, т.е. изменения порядка применения преобразований может привести к изменению результата

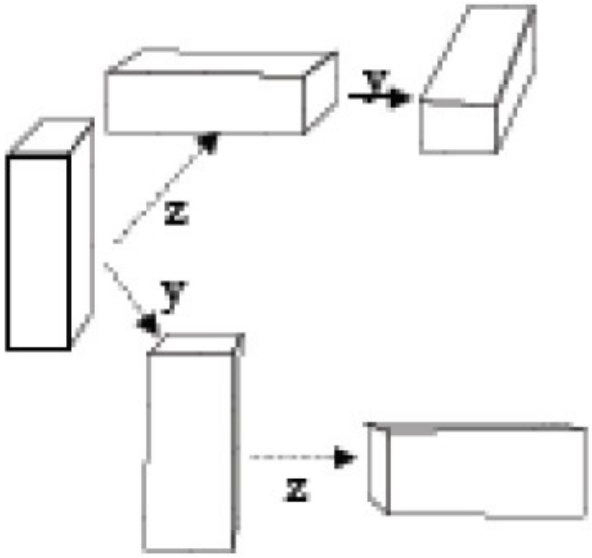


Рисунок 16 – Коммутативность 3D-преобразований

2.3.1. Перенос

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

2.3.2. Масштабирование

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

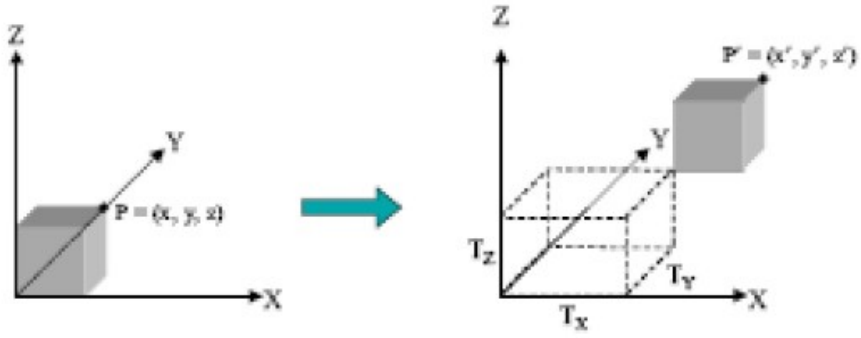


Рисунок 17 – Перенос

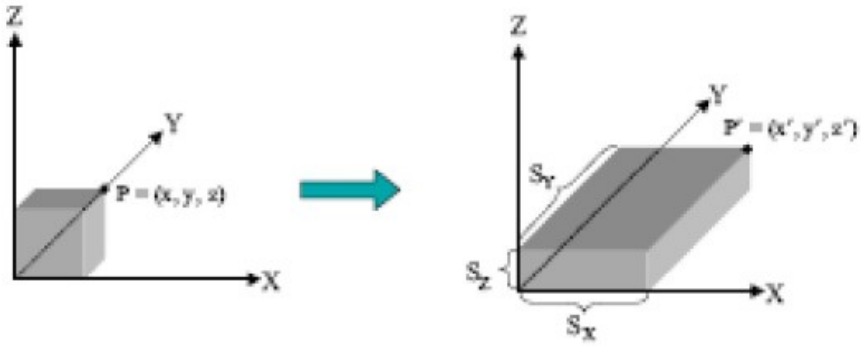


Рисунок 18 – Масштабирование

2.3.3. Повороты

Поворот по оси z

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

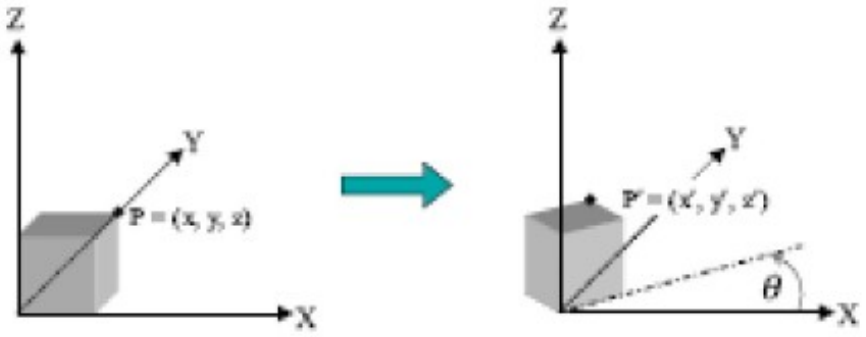


Рисунок 19 – Поворот по оси z

Поворот по оси x

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Поворот по оси y

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

2.3.4. 3D положительное вращение

Ось вращения	Положительным будет направление поворота
x	от y к z
y	от z к x
z	от x к y

2.4. Формирование матрицы преобразования

2.4.1. Примеры преобразований

`glTranslate` и `glRotate` всегда относятся к текущему состоянию матрицы. Например, если вы вызываете `glTranslate`, вы переводите из текущей "позиции" матрицы, а не из начала координат. Но если вы хотите начать с начала координат, когда вы вызываете `glLoadIdentity()`, а затем вы можете взять `glTranslate` из матрицы, которая теперь находится в начале координат, или `glRotate` из матрицы, которая теперь ориентирована по умолчанию.

```
mat4 m = Identity();
```

$$m = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
mat4 m = Identity();
mat4 t = Translate(3,6,4);
m = m * t;
```

Identity Matrix · Translation Matrix = CTM Matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 6 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 6 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
mat4 m = Identity();
mat4 t = Scale(1,2,3);
m = m * s;
```

Identity Matrix · Scaling Matrix = CTM Matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
mat4 m = Identity();
mat4 s = Scale(1,2,3);
mat4 t = Translate(3,6,4);
m = m * s * s;
```

Identity Matrix · Scaling Matrix · Translate Matrix = Final CTM

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 6 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 3 \\ 0 & 2 & 0 & 12 \\ 0 & 0 & 3 & 12 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

2.4.2. Применение преобразований в OpenGL

В приложении. Массив точек загружается в VBO; вызывается `glDrawArrays()`. CTM, сформированная в приложении, подается в вершинный шейдер, например, следующим образом:

```
matrix_loc = glGetUniformLocation(program, "model_view");
glUniformMatrix4fv(matrix_loc, 1, GL_TRUE, m);
```

В вершинном шейдере. Каждая вершина объекта умножается на CTM, чтобы получить позицию вершины

```
in vec4 vPosition;
uniform mat4 model_view;

void main(){
    gl_Position = model_view * vPosition;
}
```

2.5. Пример 3D преобразований

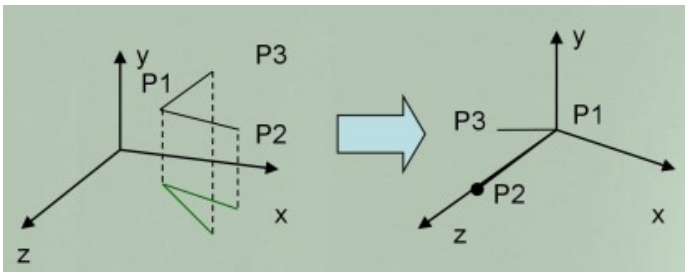


Рисунок 20 – Целевое преобразование

Шаг №1. Перенос P_1 в начало координат

$$T(-x_1,-y_1,-z_1)=\begin{vmatrix}1&0&0&-x_1\\0&1&0&-y_1\\0&0&1&-z_1\\0&0&0&1\end{vmatrix}$$

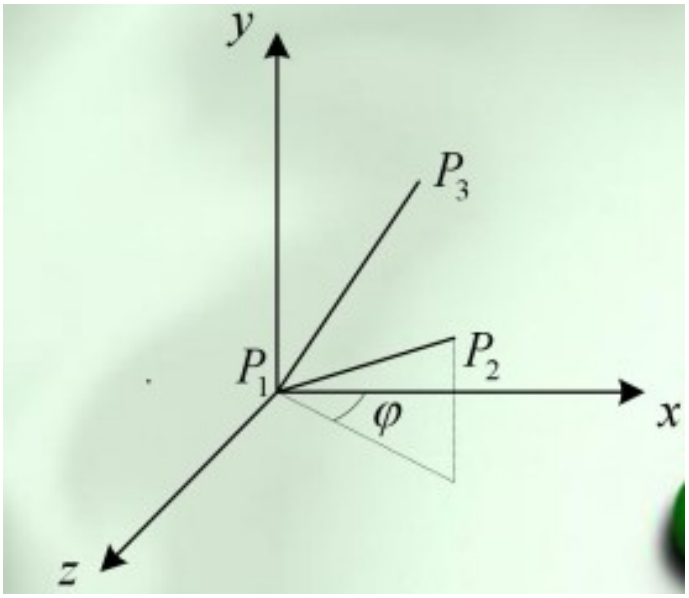


Рисунок 21 – После первого шага

Шаг №2. Поворот вокруг оси Y_1, т.е., чтобы P_1P_2 переместилась в плоскость $0YZ$

$$\theta' = -(90^\circ - \theta)$$

$$\cos\theta = \frac{z_2'}{D_1}; sin\theta = -\frac{x_2'}{D_1}$$

$$D_1 = \sqrt{(z_2')^2 + (x_2')^2}; P_2'' = R_y(\theta - 90^\circ)P_2'$$

Шаг №3. Поворот вокруг оси X, чтобы P_1P_2 совместилась с осью Z

$$\cos\psi = \frac{z_2''}{D_2}; \sin\psi = \frac{y_2''}{D_2}$$

$$D_2 = |P_1''P_2''| = |P_1P_2| = \sqrt{(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2}$$

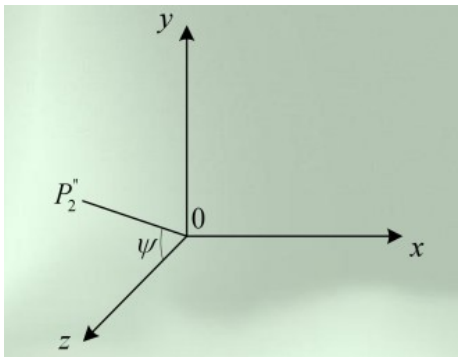


Рисунок 22 – После второго шага

$$P_2'' = R_x(\psi)R_2'' = R_x(\psi)R_y(\theta - 90^\circ)P_2'$$

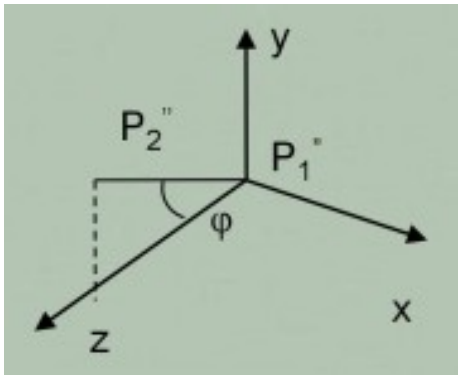


Рисунок 23 – После третьего шага

Шаг №4. Поворот вокруг оси Z. Поместить P_3 в плоскость YOZ

$$P_3''' = |X_3''' \quad Y_3''' \quad Z_3'''|T = R_X(\phi)R_Y(\theta - 90^\circ)T(-x_1, -y_1, -z_1)P_3$$

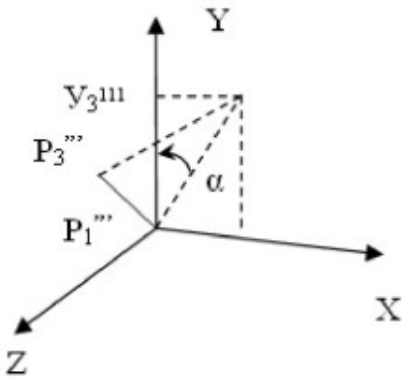


Рисунок 24 – После четвертого шага

Полная матрица будет выглядеть так:

$$R_Z(\alpha)R_X(\phi)R_Y(\theta - 90)T(-x_1, -y_1, -z_1)P_3$$

2.6. Трехмерное вращение вокруг произвольной оси

Для этого можно использовать **теорему Эйлера**: любая последовательность вращений = одно вращение вокруг некоторой оси.
Наш подход:

1. Хотим вращать β вокруг оси u через НК и произвольной точки
2. Используем два вращения, чтобы выравнять u и ось X
3. Делаем x-поворот через угол β
4. Два обратные предыдущих вращения, чтобы девыровнять u и ось X

$$R_u(\beta) = R_y(-\theta)R_z(\phi)R_x(\beta)R_z(-\phi)R_y(\theta)$$

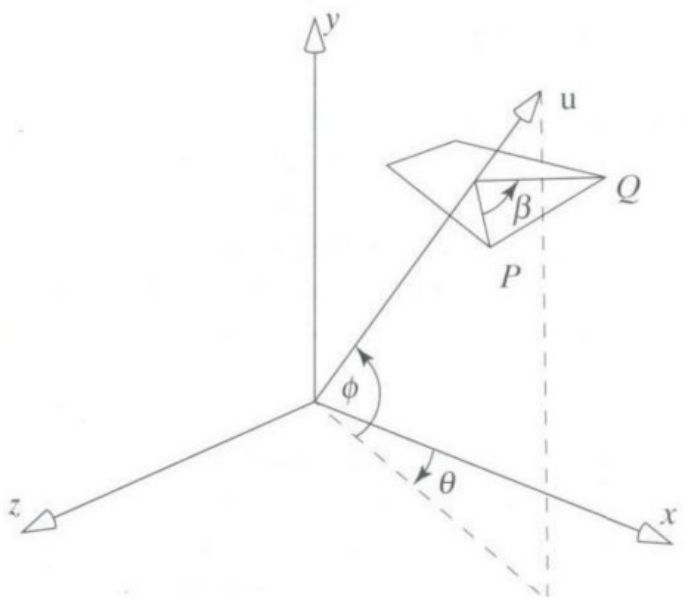


Рисунок 25 – Вращение вокруг произвольной оси

2.7. Примеры решения наиболее важных задач

2.7.1. Поворот вокруг точки на плоскости

Построить матрицу поворота на угол ϕ вокруг точки $A(a, b)$ на плоскости

Шаг №1. Перенос на вектор $-A = (-a, -b)$ для совмещения оси поворота с началом координат

$$T_{-A} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -a & -b & 1 \end{pmatrix}$$

Шаг №2. Поворот на угол ϕ

$$R_\phi = \begin{pmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Шаг №3. Перенос на вектор $A = (a, b)$ для возвращения оси поворота в прежнее положение

$$T_A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ a & b & 1 \end{pmatrix}$$

Шаг №4. Формирование матрицы M итогового преобразования

$$M = T_{-A} R_\phi T_A$$

2.7.2. Отражение относительно прямой

Построить матрицу отражения плоского полигона $\{A_i\}$ относительно прямой I, проходящей на плоскости через начало координат под углом ϕ к оси абсцисс.

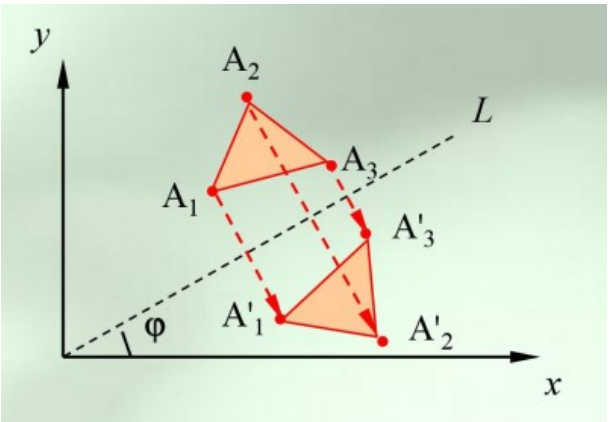


Рисунок 26 – Отражение относительно прямой

Шаг №1. Поворот оси X до совпадения с осью отражения I

$$R_\phi = \begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Шаг №2. Отражение относительно оси X

$$S_X = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Шаг №3. Возврат системы координат в исходное состояния

$$R_\phi^{-1} = \begin{pmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Шаг №4. Формирование матрицы итогового преобразования

$$M = R_\phi S_x R_\phi^{-1}$$

Шаг №5. Преобразование координат каждой вершины полигона

$$A'_i = A_i M, i = 1, 2, ...$$

2.7.3. Поворот вокруг прямой в 3D-пространстве

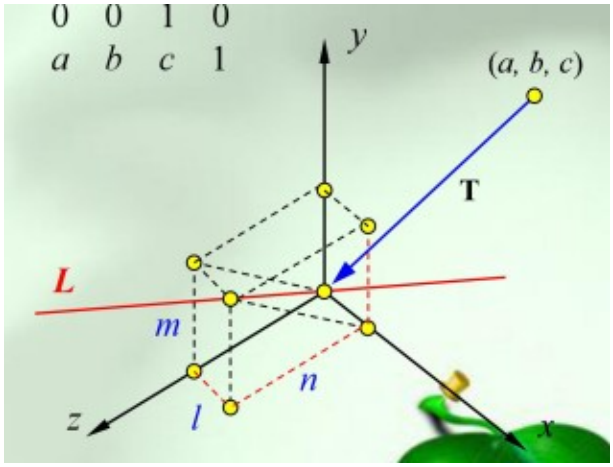


Рисунок 27 – Вращение вокруг прямой в 3D-пространстве

Построить матрицу поворота на угол ϕ вокруг прямой L в 3D-пространстве, проходящей через точку $A = (a, b, c)$ и имеющей направляющий вектор (l, m, m) с модулем, равным единице.

Решение похоже на рассмотренное в 2.7.1, но здесь ось вращения задается в 3D-пространстве и ориентирована произвольно.

Матрицы переноса и возврата:

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -a & -b & -c & 1 \end{bmatrix}; \quad T^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ a & b & c & 1 \end{bmatrix}$$

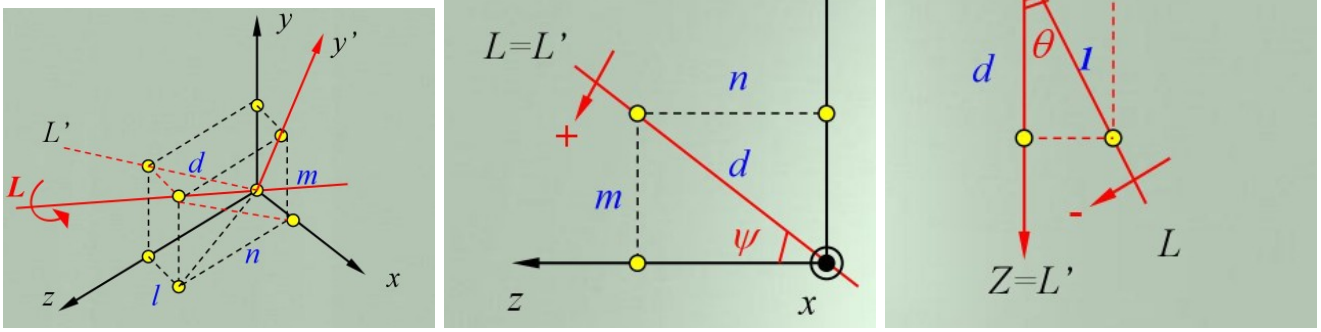
Пусть ось вращения, заданная в условии, будет осью Z системы координат в новом положении, в которое мы её переведём двумя вращениями сначала вокруг оси X, потом вокруг новой оси Y (после вращения вокруг этой оси нужно будет "вернутся обратно")

Зададим матрицы вращения:

$$R_z = \begin{bmatrix} \cos\phi & \sin\phi & 0 & 0 \\ -\sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix};\ R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{n}{d} & \frac{m}{d} & 0 \\ 0 & -\frac{m}{d} & \frac{n}{d} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix};\ R_y = \begin{bmatrix} d & 0 & l & 0 \\ - & 1 & 0 & 0 \\ -l & 0 & d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Где

$$d = \sqrt{m^2 + n^2};\ \sin\psi = \frac{m}{d};\ \cos\psi = \frac{n}{d}$$



(a) Матрица R_z (b) Матрица R_x (c) Матрица R_z

Рисунок 28 – Матрицы поворота

Матрица итогового преобразования:

$$M = T \cdot R_x \cdot R_y \cdot R_z \cdot R_y^{-1} \cdot R_x^{-1} \cdot T^{-1} = Q \cdot R_z \cdot Q^-1$$

где

$$Q = T \cdot R_x \cdot R_y$$

2.8. Граф сцены

3D-сцены часто хранятся в направленном ациклическом графе (DAG), называемом **граф сцены**. Обычно такой граф состоит из:

- Объекты (кубы, сферы, конусы, многогранники)
- Атрибуты (цвет, текстурные карты)
- Преобразования

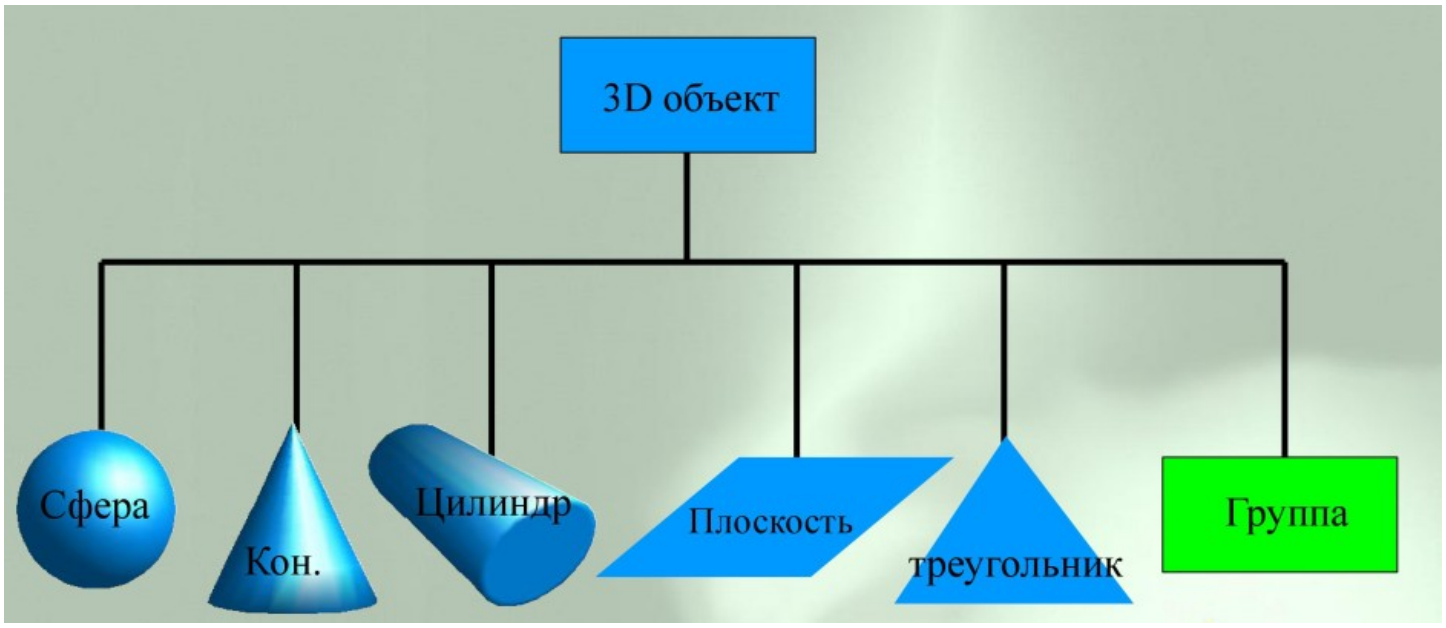
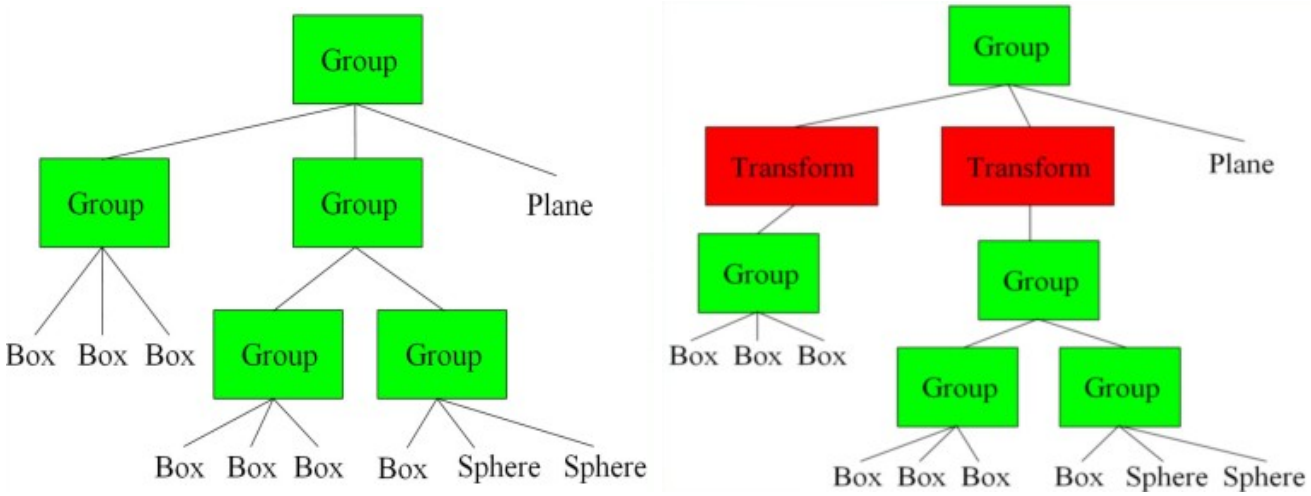


Рисунок 29 – Граф сцены

Преобразования выступают такими же узлами в графе, как и объекты.



(a) Без преобразований (b) С преобразованиями

Рисунок 30 – Преобразования

3. Проецирование

3.1. История проецирования

3.2. Ранний вид проецирования

(a) Месопотамия, XX в. до н.э. (b) Греция, VI до н.э.

Рисунок 31 – Раннее процирование

31a - вид сверху (параллельный, особенно орфографический, проекционный) из Месопотамии (2150 г. до н.э.): самый ранний известный технический рисунок
31b - греческая ваза конца VI века до нашей эры: показывает признаки попыток ракурса перспективы! Обратите внимание на относительные размеры бедер и нижних конечностей минотавра Древнеегипетское искусство:

- Несколько точек обзора
- Параллельная проецирование (нет попытки изобразить перспективный ракурс)

32 - Могила Нефертари, Фивы (19-й династии, - 1270 г. до н.э.). Королева, ведомая Исидой. Фреска.
Обратите внимание, как изображение тела подразумевает вид спереди, но ноги и голова подразумевают вид сбоку (ранний кубизм!)

Рисунок 32 – Египет, 1270 г. до н.э.

3.2.1. Эпоха Возрождения

Начиная с XIII века : новый акцент на важности индивидуальной точки зрения, мировой интерпретации, способности наблюдения (в частности природы: астрономии, анатомии и т.д.

- Мазаччо, Донателло, Вселенная да Винчи в качестве часового механизма

Перестраивая вселенную более системно и механически:

- Тихо Браге и Рудольф Ив Праге (деталь часового механизма), ок. 1585
- Коперник, Кеплер, Галилей, Ньютон ...: от земно-ориентированной до гелиоцентрической модели (механистической) вселенной, законы которой можно обнаружить и понять

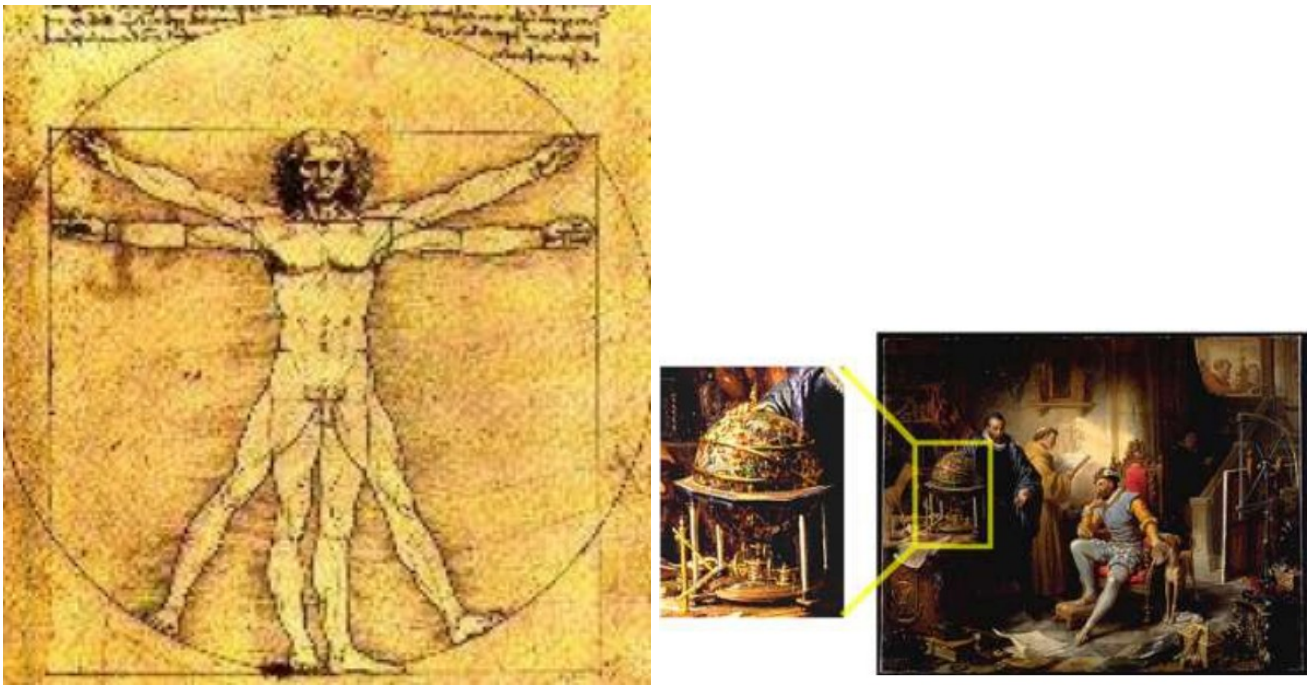


Рисунок 33 – Проецирование в эпоху Возрождения

3.2.2. Ранние попытки в перспективе

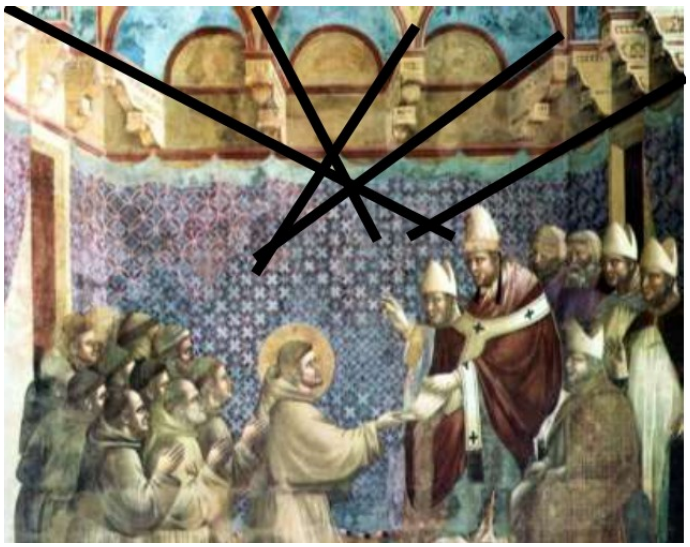


Рисунок 34 – Джотто. Утверждение о францисканском правительстве

В искусстве более реалистично пытаться представить 3D-пространство. Ранее работы вызывали ощущение пространства 3D, но не систематически. 34 - параллельные линии сходятся, но существует несколько точек схода

3.2.3. Брунелесски и Вермеер

Брунеллески изобрел систематический метод определения перспективной проекции (начало 1400-х годов). Он создал демонстрационные панели с особыми ограничениями про-

Рисунок 35 – Перспективный рисунок для церкви Санто Спирито во Флоренции

смотра для полной точности воспроизведения.
Вермеер и другие создали перспективные коробки, где изображение, просматриваемое через отверстие для обзора, имело правильную перспективу

Брунеллески определял точность своих картин, сделав отверстие в точке схода, и исследуя отражение в зеркале и сравнивая сходимость линии к реальной модели.

Реализм его картин свидетельствует о том, что у Брунеллески был некоторый систематический метод определения перспективных проекций, хотя процедура, которую он использовал, никогда не была задокументирована. Его иллюзия вдохновила других художников на изучение линейной перспективы.

3.2.4. Camera Obscura

Камера-обскура (лат. camera obscura — «тёмная комната»). Простейший вид устройства представляет собой светонепроницаемый ящик с отверстием в одной из стенок и экраном (матовым стеклом или тонкой белой бумагой) на противоположной стене. Лучи света, проходя сквозь малое отверстие (диаметр которого зависит от «фокусного расстояния» камеры, приблизительно 0,1—5 мм) создают перевёрнутое изображение на экране.

- На основе камеры-обскуры были сделаны фотокамеры - фотографические аппара-

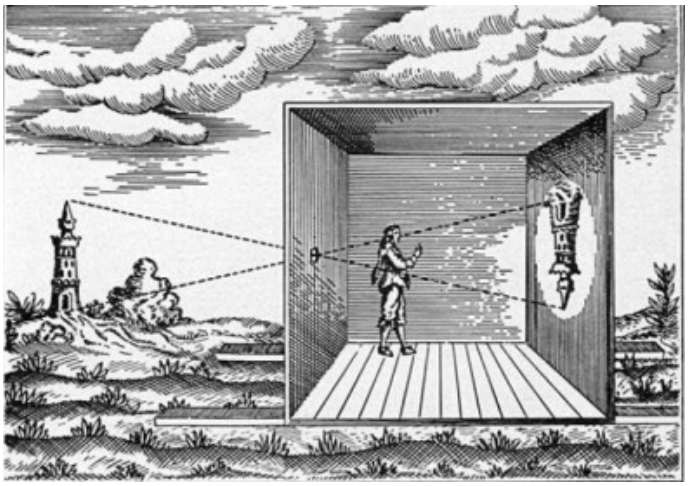
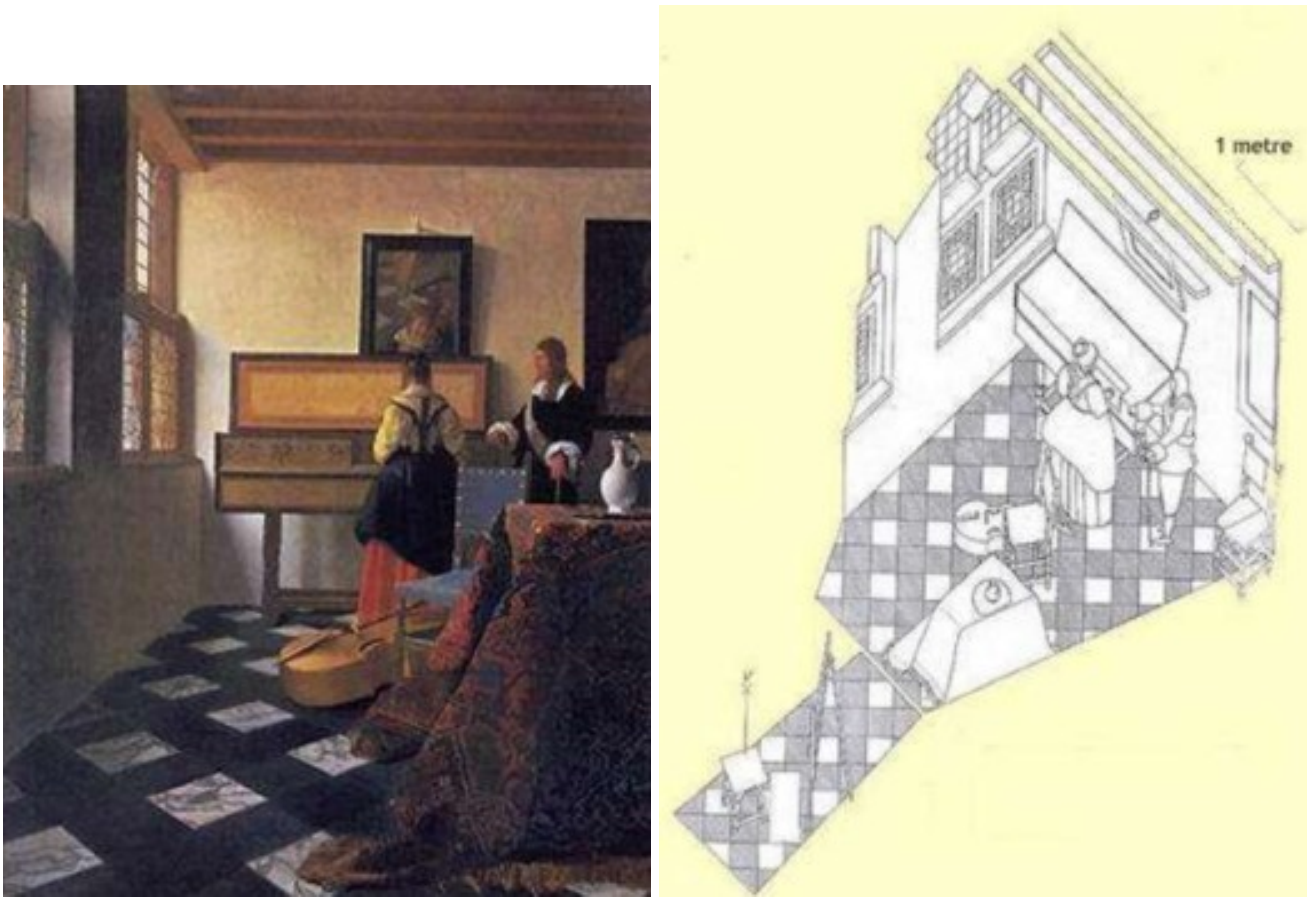


Рисунок 36 – Камера обскура

ты без объектива

- Камера-обскура является частным случаем устройства с кодирующей апертурой.
- Ввиду отсутствия в камере-обскуре оптических элементов прямо воздействующих на свет (за исключением границы отверстия) она подходит для создания изображений в высокоэнергетических областях спектра

Вермеер, вероятно, использовал камеру-обскура, чтобы помочь формированию его шедевров; Также были созданы перспективные коробки, где у картины, когда она рассматривается через отверстие, была бы правильная перспектива.



(a) Картина (b) Реконструкция

Рисунок 37 – Вермеер, урок музыки. 1662-1665

3.2.5. Диего Веласкес, Пьерро дела Франческа

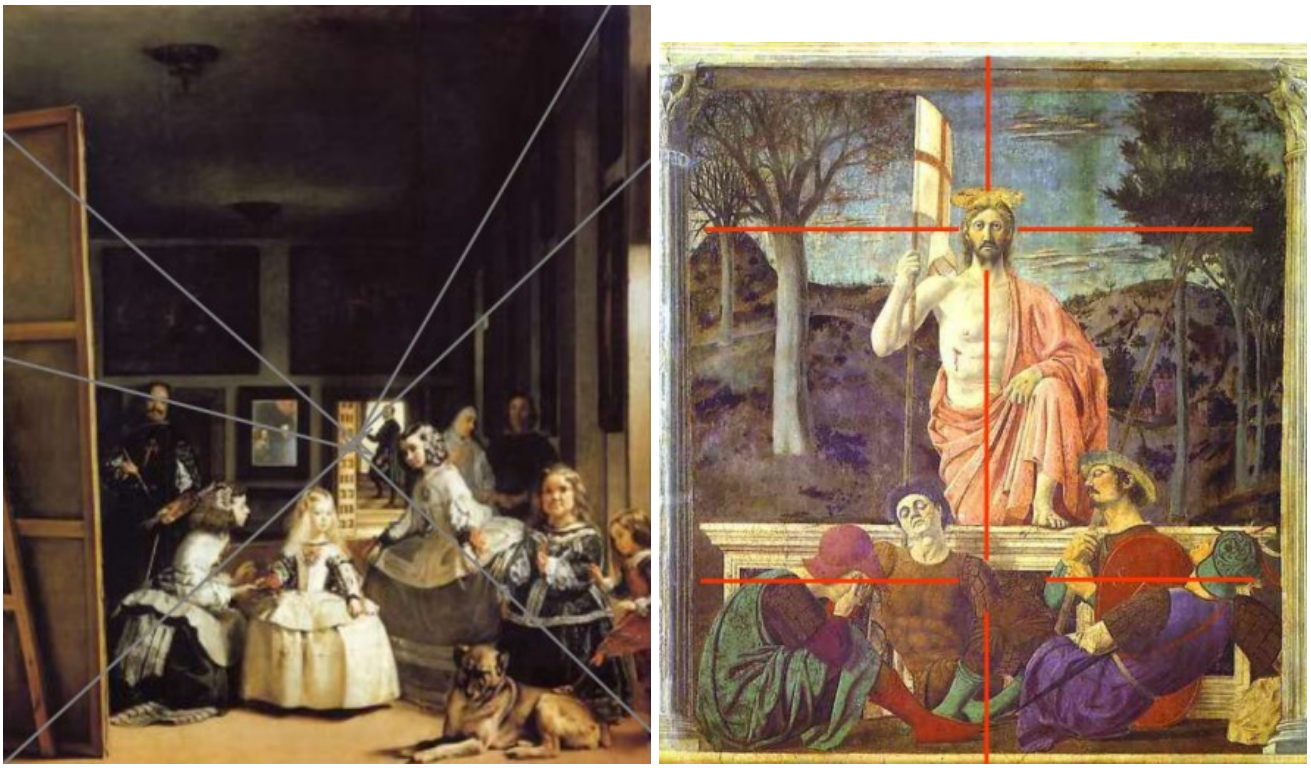
Точка зрения влияет на содержание и значение того, что отображается. 38a - Королевская пара находится в зеркале, но действительно ли это их изображение?
Анализ через компьютерную реконструкцию нарисованного места дает приговор: королевская пара в зеркале - отражение от холста на переднем плане, а не фактических людей.

Перспектива может использоваться, как способ управлять восприятием. 38b - Использование двух точек зрения концентрирует внимание зрителя поочередно на Христа и саркофаг.

3.2.6. Правила линейной перспективы

Идеи линейной перспективы:

- Параллельные линии сходятся (по 1, 2 или 3 осям) к точке схода.
- Объекты, расположенные дальше, более искажены (т. е. меньше), чем ближе — тем больше



(a) Диего Веласкес, 1656. Королевская семья (b) Пьеро дела Франческа (1460)

Рисунок 38 – Влияние перспективы на на содержание

- Пример: перспективный куб

3.3. Геометрическое построение проекций

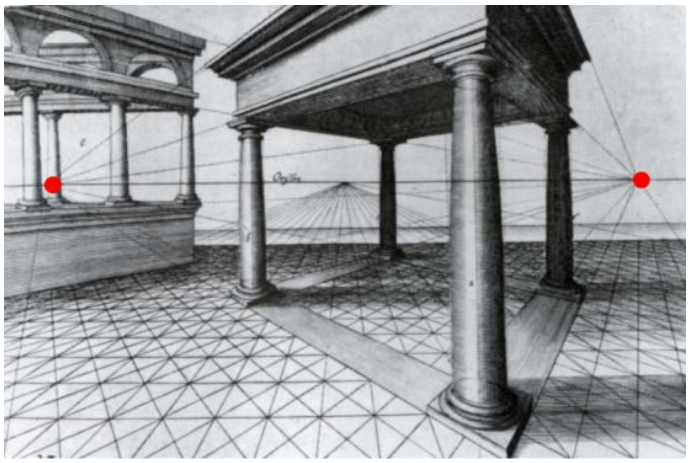


Рисунок 39 – Двухточечная проекция

3.3.1. Концептуальная модель

При визуализации двумерных изображений достаточно задать окно видимости в системе координат пользователя и порт отображения на экране дисплея, в котором будет выдаваться изображение из окна.

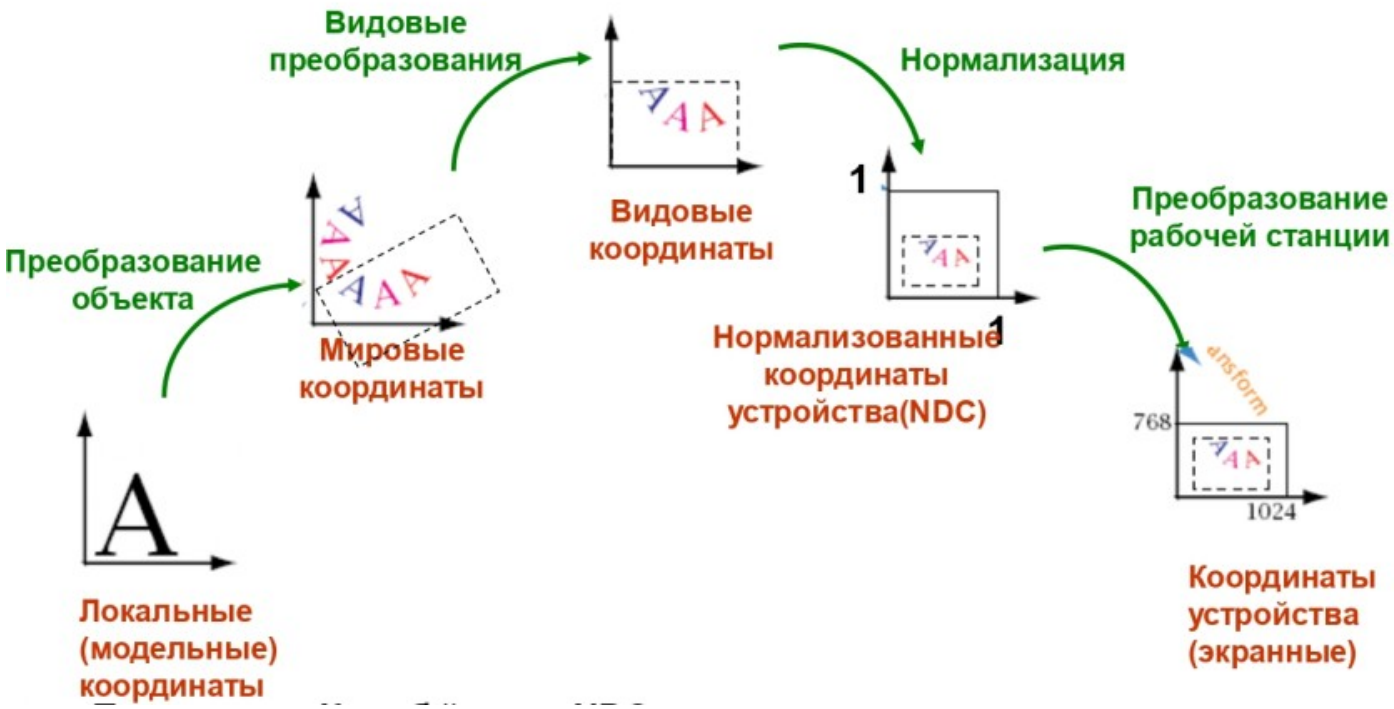


Рисунок 40 – Концептуальная модель процесса 2D вывода

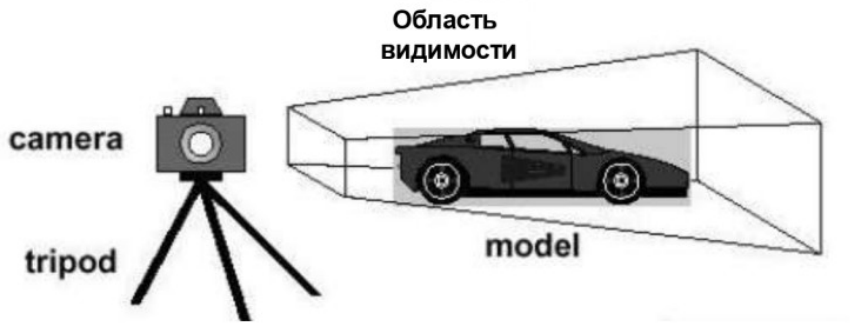


Рисунок 41 – Модель и область видимости

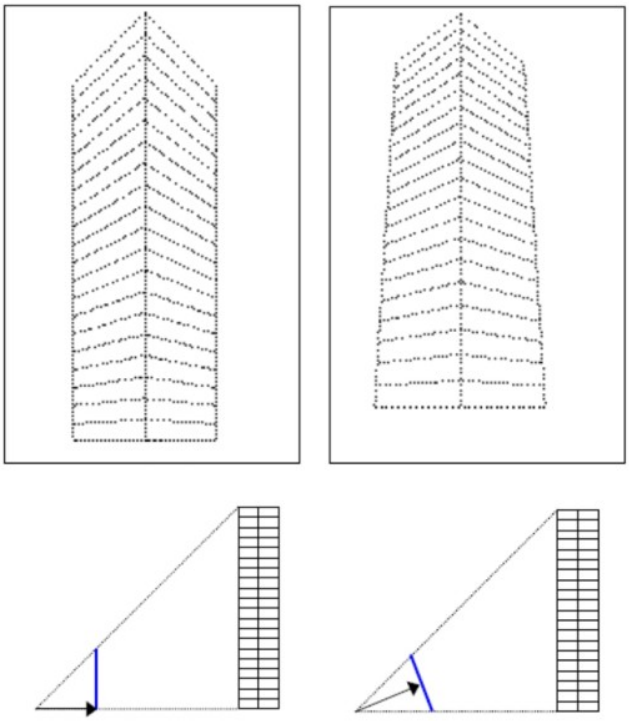
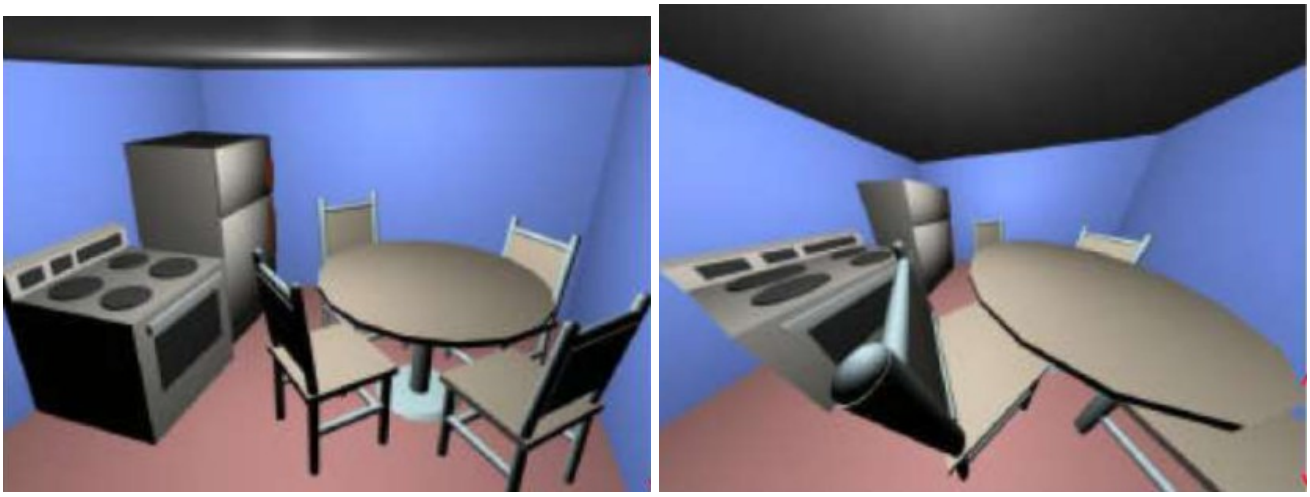


Рисунок 42 – Угол обзора

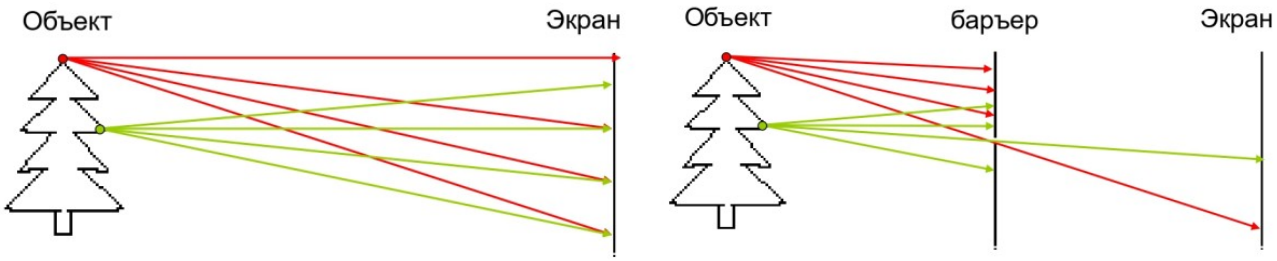


(a) Правильно (b) Неправильно

Рисунок 43 – Настройка проекции

3.4. Ортографическое и перспективное проектирование

3.4.1. Камера



(a) Бесполезная камера (b) Камера-обскура

Рисунок 44 – Камера

Простейшая камера. Положим кусок пленки перед объектом. На каждую точку пленки падают лучи от всех точек объекта; получается бессмысленное изображение.

Камера-обскура. Добавим преграду с отверстием, чтобы отсечь лишние лучи.

- На каждую точку пленки падает один луч!
- Картинка должна быть четкой
- Отверстие в преграде называется **апертурой** или **диафрагмой**

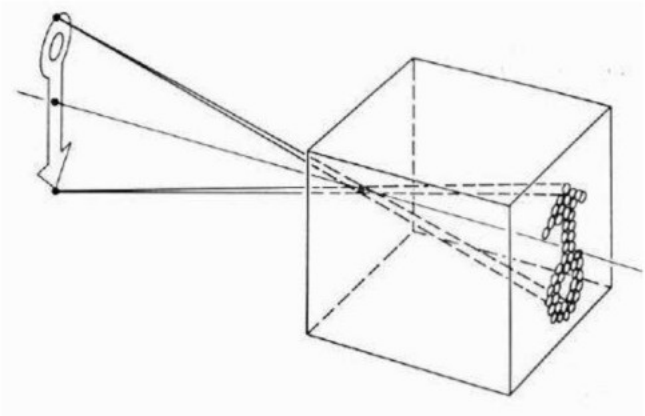
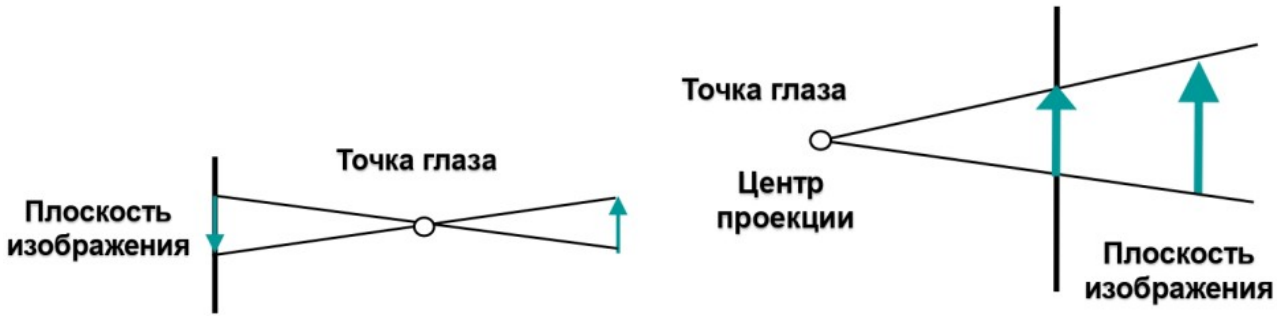


Рисунок 45 – Камера-обскура

Модель камеры-обскуры

- В преграде отверстие размером в одну точку
- Все лучи проходят через одну точку
- Эта точка называется **центром проекции** (ЦП)
- Изображение формируется на **картинной плоскости**
- **Фокусным расстоянием** f называется расстояние от ЦП до картинной плоскости



(a) Реальная камера (b) Подходящий эквивалент в КГ

Рисунок 46 – Камера в графике

Камера в графике. В реальном мире изображение формируется на задней поверхности камеры, в компьютерной графике - изображение формируется перед объективом камеры.

3.4.2. Преобразования камеры

Упростим, перемещая точку зрения камеры в центр, таким образом, чтобы смотреть в направлении отрицательной оси Z.

3.5. Перспективное проектирование

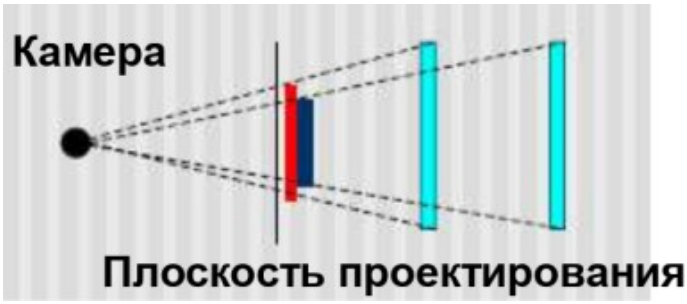


Рисунок 47 – Перспективная проекция

- Подобность реальному миру
- Характеризуется видимым в перспективе объектом
- Объекты кажутся большими, если они ближе к камере
- Что необходимо:
  - Центр проектирования
  - Плоскость проектирования
- Проектирование: соединение объекта к центру проектирования

Картинная плоскость определяется некоторой точкой на плоскости, которую будем называть **опорной точкой** (ОТ) и нормалью к картинной плоскости (НКП).

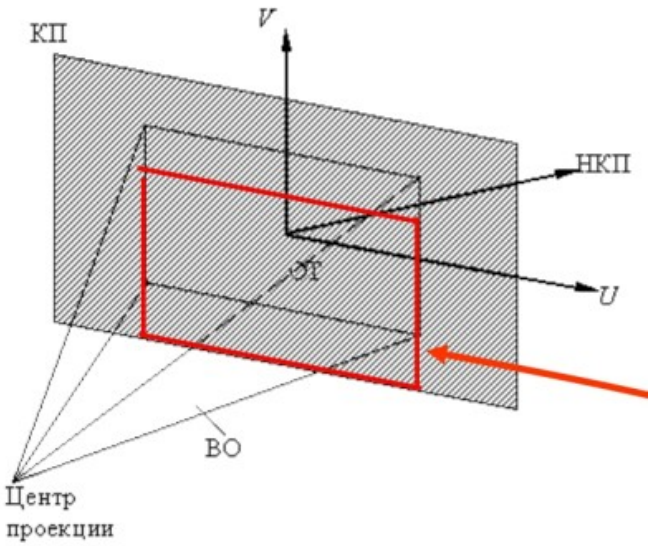


Рисунок 48 – Проекция

КП может произвольным образом располагаться относительно проецируемых объектов, заданных в мировых координатах. Она может пересекать их, проходить впереди или позади объектов.

Для того, чтобы задать окно, нам необходима система координат, которую назовём системой координат UV. Началом ее служит ОТ.

Направление оси V на КП определяет **вектор вертикали** (ВВ) - проекция ВВ на КП совпадает с осью V.

ОТ и два направления вектора НКП и ВВ определяются d правосторонней мировой системе координат. Имея на КП систему UV, можем задать минимальное и максимальное значения U и V, определяющие окно — поле вывода.

В случае центральной проекции ВО определяется также центром проекции. Этот параметр задается в мировых координатах относительно ОТ.

3.5.1. Область представления

Определяет, сколько из мира взято в картину. Чем больше область представления, тем меньше размер объекта проектирования

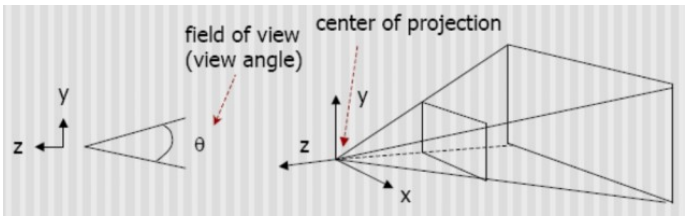


Рисунок 49 – Область представления

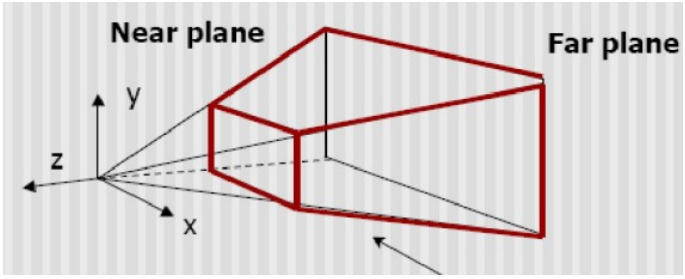


Рисунок 50 – Плоскости отсечения

3.5.2. Передняя и задняя плоскости отсечения

- Выводятся только объекты между передней (near) и задней (far) плоскостями отсечения.
- Передняя плоскости отсечения + задняя плоскости отсечения + объем видимости = пространство видимости

3.6. Графический конвейер

В большинстве подсистем трехмерной графики применяется графический конвейер. **Конвейер** - это логическая группа вычислений, выполняемых последовательно, которые дают на выходе синтезируемую сцену. Наличием переходов между этапами конвейера обеспечивается возможность выбора между программной и аппаратной реализацией очередного этапа

Такой подход к настройке конвейера позволяет приложениям трехмерной графики получать преимущества аппаратной реализации, когда таковая доступна. Таким образом, реализация конвейера может чисто программной, полностью аппаратной или смешанной (программно-аппаратной)

3.6.1. Состав графического конвейера

На входе:

- **Геометрическая модель**: Описание всех объектов, поверхностей и геометрии источников света и преобразований
- **Световая модель**: Описания вычислительные свойств объектов и источников света
- **Точка наблюдения (или камера)**: Позиция наблюдателя и угол обзора

Собственно, конвейер:

1. Модельные преобразование
2. Освещение (Закраска, Shading)
3. Видовое преобразование (Перспектива)
4. Отсечение (Clipping)
5. Проецирование в экранное пространство
6. Растеризация (Rasterization)
7. Видимость (Вывод)

Растровый вывод - пиксельная решетка, куда выводится изображение. Может быть в формате цвет/интенсивность.

3.6.2. Модельные преобразования

- Трехмерные модели определены в их собственной системе координат (объектное пространство - object space)
- Моделирование преобразовывает модели в пределах общей координационной структуры (мировое пространство - world space)

Рисунок 51 – Пространство объекта и мировое пространство

3.6.3. Освещение (закраска)

Вершины освещены (закрашены, затенены) согласно свойствам материала, свойствам поверхностей (нормали) и источникам света

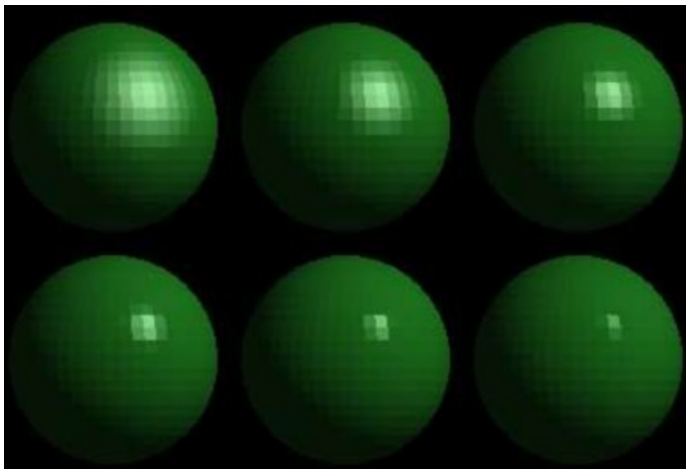


Рисунок 52 – Различные модели освещения

3.6.4. Видовое преобразования

Для чего нужно еще одно преобразование? Проективные преобразования описывают «стандартные» проекции, т.е. проецируют фиксированную часть пространства

Что если мы хотим переместить наблюдателя? Варианты:

- Изменить матрицу проекции, чтобы включить в нее информации о камере
- Применить дополнительное преобразование, «подгоняющее» объекты под стандартную камеру

Стандартная камера в OpenGL:

- Наблюдатель в $(0, 0, 0)$
- Смотрит по направлению $(0, 0, -1)$
- Верх $(0, 1, 0)$

Мировое пространство преобразовывается в видовое. Положение наблюдателя, преобразованное к центру и направлению, сориентировано вдоль некоторой оси (обычно 2)

3.6.5. Отсечение (Clipping)

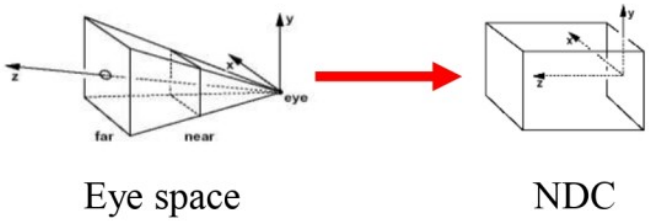


Рисунок 53 – Отсечение

Преобразует в нормализованные координаты устройства. Части объекта вне объема представления удалены.

3.6.6. Проецирование

Объекты проецируются в пространство изображения (экранное пространство

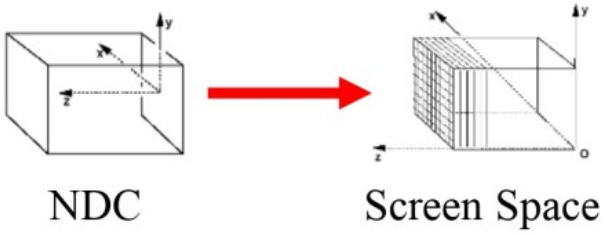


Рисунок 54 – Проецирование

3.6.7. Растеризация

Объекты растеризуются в пиксели.

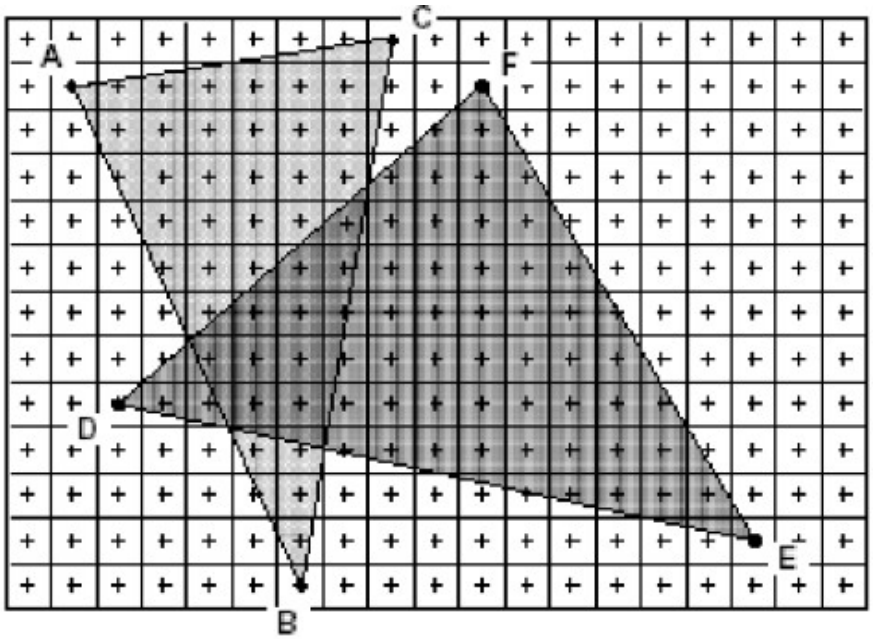


Рисунок 55 – Растеризация

3.6.8. Видимость / Вывод

Каждый пиксел помнит самый близкий объект (буфер глубины).

Почти каждый шаг в графическом конвейере вызывает изменение системы координат. Преобразования являются центральными к пониманию трехмерной компьютерной графики.

3.6.9. Координатные системы в конвейере

Этап конвейера	**Координатные системы**
Модельные преобразование	Object Space
Освещение (Закраска, Shading)	World Space
Видовое преобразование (Перспектива)	Eye Space
Отсечение (Clipping)	Clip Space (NDC)
Проецирование в экранное пространство	Screen Space
Растеризация (Rasterization)	
Видимость (Вывод)	

3.6.10. Координатные системы

- Объектное пространство — объектная система координат.
  Локальное для каждого объекта
- Мировое пространство - мировая система координат
  Общее для всех объектов
- Видовое пространство / Камера — видовая система координат
  Выводится из области видимости
- Пространство отсечения / нормализованное пространство устройства(NDC) - НСК
  $[-1,-1,-1] > [1,1,1]$
- Экранное пространство — экранные координаты
  индексируется в соответствии с атрибутами hardware

3.6.11. Концептуальная модель процесса 3D-вывода

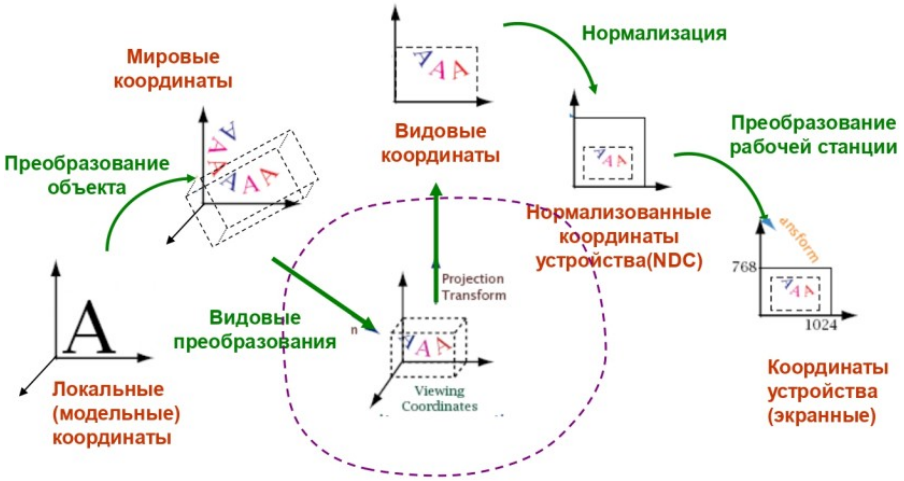


Рисунок 56 – 3D-вывод

3.6.12. Картинная плоскость

Рисунок 57 – Картинная плоскость

- **Картинная плоскость**

  Характеризуется:
  - Опорной точкой
  - Нормалью к картинной плоскости
  - Расстоянием до картинной плоскости (КП)
- **Опорная точка**

  Определяет картинную плоскость. Берется в МК, обычно принадлежит объекту или находится внутри него (ТПВ — точка привязки вида (VRP view reference point).
- **Расстояние до КП** - расстояние от опорной точки в направлении вектора нормали картинной плоскости

- **Нормаль к картинной плоскости** - задает ориентацию картинной плоскости. Для однозначного задания положения камеры недостаточно знать ориентацию картинной плоскости, так как остается еще одна степень свободы камеры — поворот вокруг нормали к картинной плоскости. Задание вектора вертикали вид ВВКП зафиксирует однозначно ориентацию камеры.

3.7. Простейшая перспективная проекция

Спроектируем все точки вдоль оси z к плоскости $z = d$, точка наблюдения в НК:

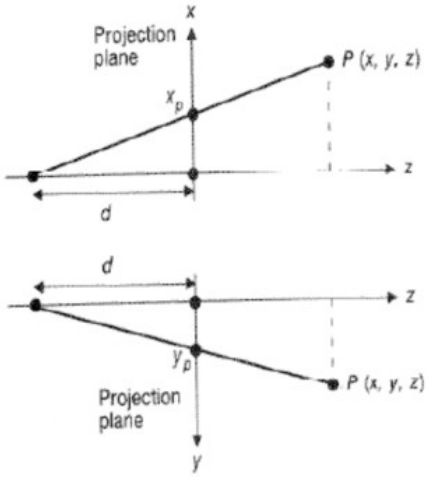


Рисунок 58 – Простейшая проекция

$$\frac{x_p}{d} = \frac{x}{z+d}; \; \frac{y_p}{d} = \frac{y}{z+d}$$

$$x_p = \frac{d \cdot x}{z+d} = \frac{x}{\frac{z}{d}+1}$$

$$y_p = \frac{d \cdot y}{z+d} = \frac{y}{\frac{z}{d}+1}$$

Представление обобщённой спроектированной точки: $P_p = \begin{bmatrix} X & Y & Z & W \end{bmatrix}^T$. В таком случае то же самое в матричном виде:

$$P_p = M_{per} \cdot P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} X & Y & Z & W \end{bmatrix}^T = \begin{bmatrix} x & y & z & z/d \end{bmatrix}^T$$

3.8. Виды плоских геометрических проекций

По расположению центра проекции относительно плоскости проекции различаются **центральная** и **параллельная** проекции.

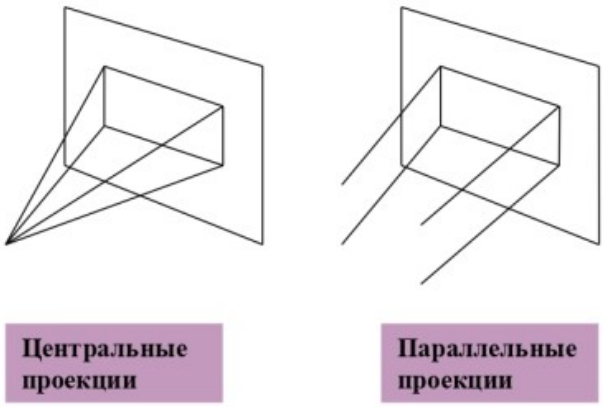


Рисунок 59 – Центральная и параллельная проекции

Центральная проекция имеет центр проекции, а параллельная проекция не имеет центра проекции, т.е. считается, что этот центр располагается в бесконечности.

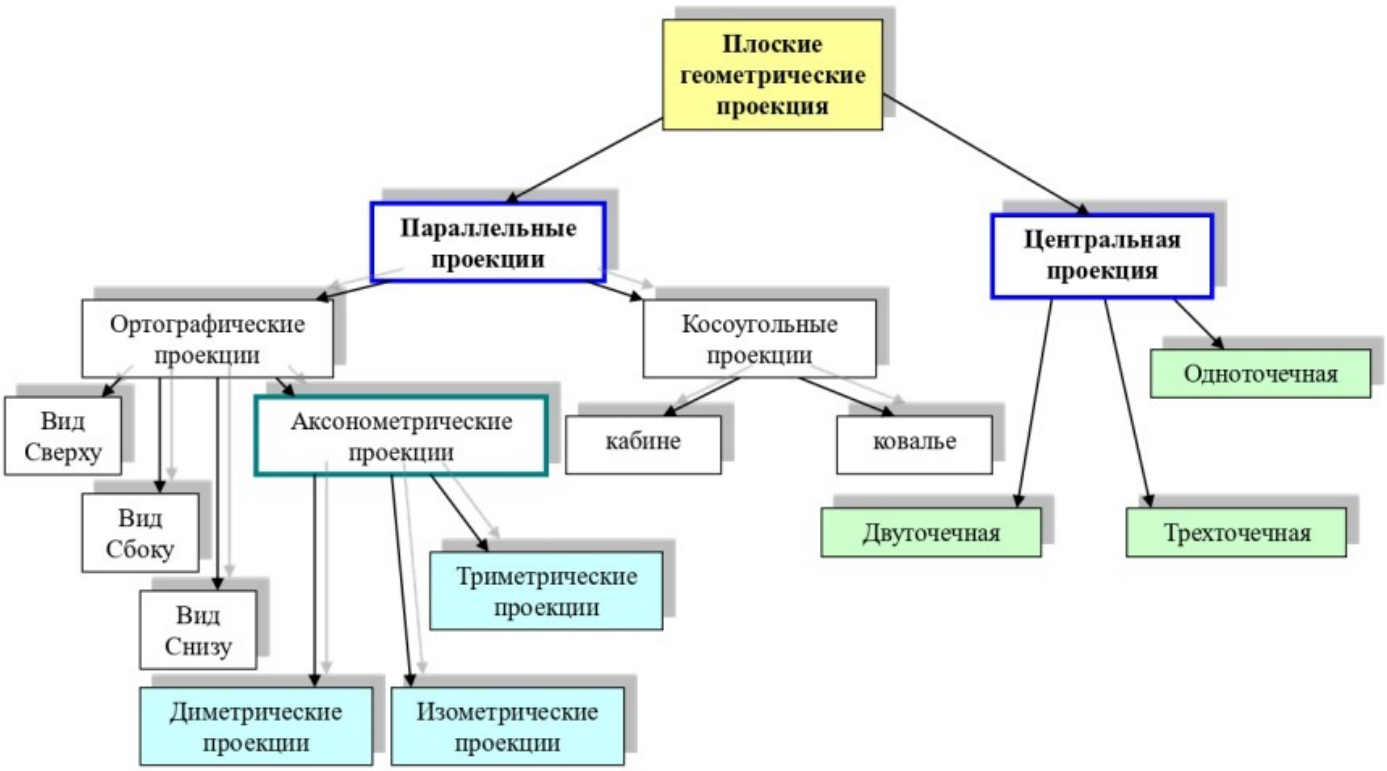


Рисунок 60 – Виды проекций

Параллельные проекции делятся на два типа в зависимости от соотношения между направлением проецирования и нормалью к проекционной плоскости:

- **ортографические** - направления совпадают, т. е. направление проецирования является нормалью к проекционной плоскости;
- **косоугольные** - направление проецирования и нормаль к проекционной плоскости не совпадают.

3.8.1. Мультивидовая ортографическая

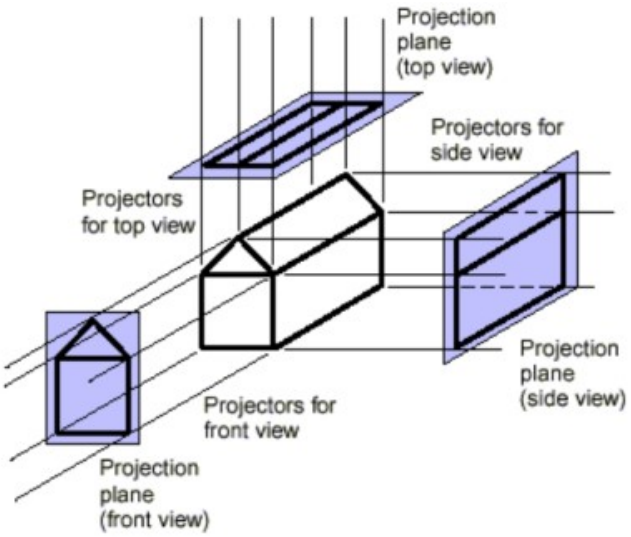


Рисунок 61 – Мультивидовая ортографическая проекция

Используются для:

- Технических рисунков машин, машинных частей
- рабочие архитектурные рисунки

Доводы "за":

- точное возможное измерение
- все представления в том же самом масштабе

Доводы "против":

- не обеспечивает "реалистическое" представление или смысл трехмерной формы

$$M_{orth} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.8.2. Аксонометрические проекции

Тот же самый метод как мультивидовое орфографическое проектирование, кроме того, что плоскости проектирования не параллельны любой из координатных плоскостей; параллельные линии одинаково видятся в перспективе

- **Изометрическая**: углы между всеми тремя основными осями равны (120°). Одинаковые отношения масштаба применяются вдоль каждой оси
- **Диметрическая**: углы между двумя из основных осей равны; отношения масштаба применяются вдоль двух осей
- **Триметрическая**: углы, различные между тремя основными осями; три отношения масштаба

Триметрическая: углы, различные между тремя основными осями; три отношения масштаба Наиболее общей формой является изометрическая проекция. Расстояние по всем

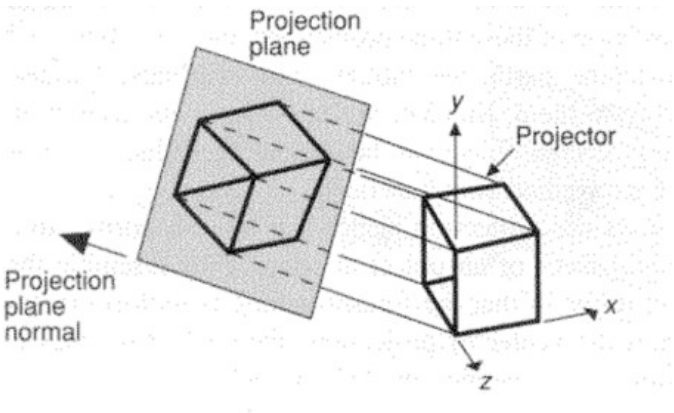


Рисунок 62 – Аксонометрическая проекция

трем главным осям укорачивается с одним и тем же коэффициентом, поэтому возможно с одним и тем же масштаб коэф. по всем трем осям. Для ее образования нужно одинаковое число раз сократить все три оси.

$$\begin{bmatrix} \cos\phi & \sin\phi \cdot \sin\theta & -\sin\phi \cdot \cos\theta & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ \sin\phi & -\cos\phi \cdot \sin\theta & \cos\phi \cdot \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

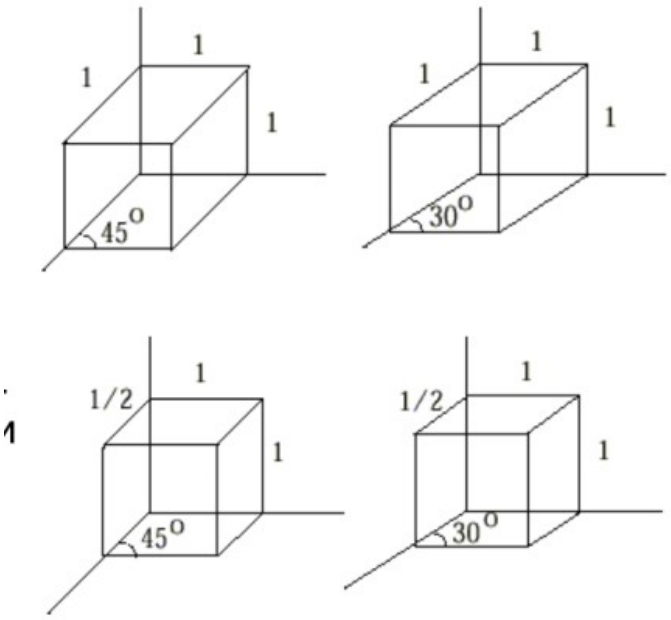


Рисунок 63 – Косоугольные проекции

3.8.3. Косоугольные проекции

Косоугольные проекции Кавалье и Кабине не сохраняют ортогональности системы координат. Данные проекции нечасто используются в инженерной практике.

Кавалье (Военная проекция): две оси кажутся перпендикулярными и не сокращены; третья ось наклонена по отношению к горизонтали (на угол 45°) и не сокращена

Кабине: угол между проекторами и плоскостью проекции $arctan(2) = 63.4°$. Взаимно перпендикулярные плоскости проецируются с коэффициентом 50

3.8.4. Совмещенное представление видов параллельных проекций

- Многовидовая ортография
  - VPN паралельна всем осям
  - DOP паралельна VPN
  - Видна вся грань
- Аксонометрическая проекция
  - VPN не паралельна осям кооординат
  - DOP параллельна VPN
  - Изменяется размер. Размер — функция угла между нормалью грани и DOP
- Косоугольная проекция

– VPN паралельна основной координатной оси
– DOP не паралельна VPN
– Изменение граней за исключением параллельной КП

3.8.5. Центральная проекция

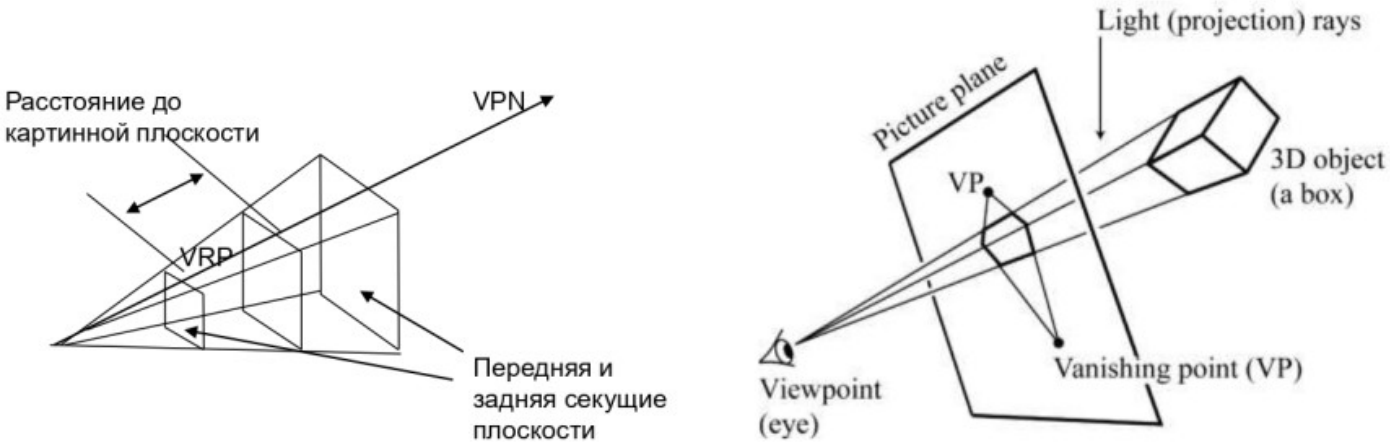
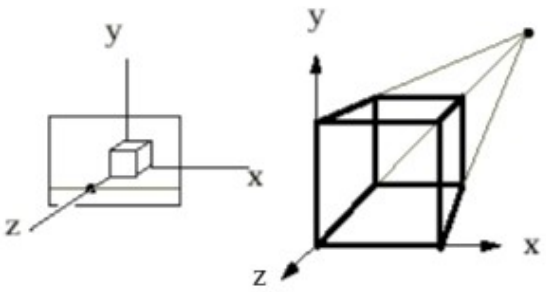
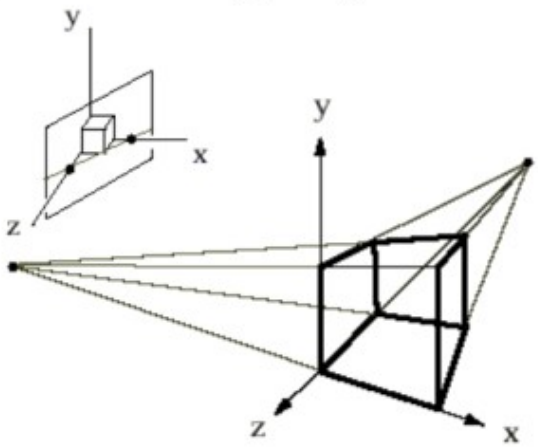


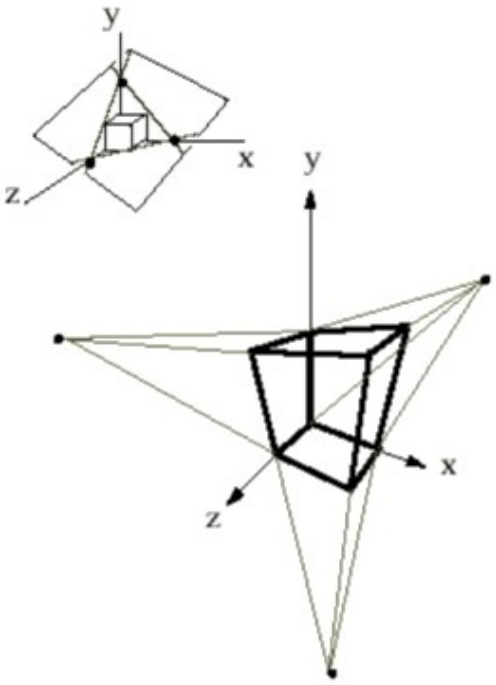
Рисунок 64 – Пирамида видимости



One Point Perspective
(*z*-axis vanishing point)



Two Point Perspective
(*z*, and *x*-axis vanishing points)



Three Point Perspective
(*z*, *x*, and *y*-axis vanishing points)

Рисунок 65 – Виды перспектив в центральной проекции