Компьютерная Графика

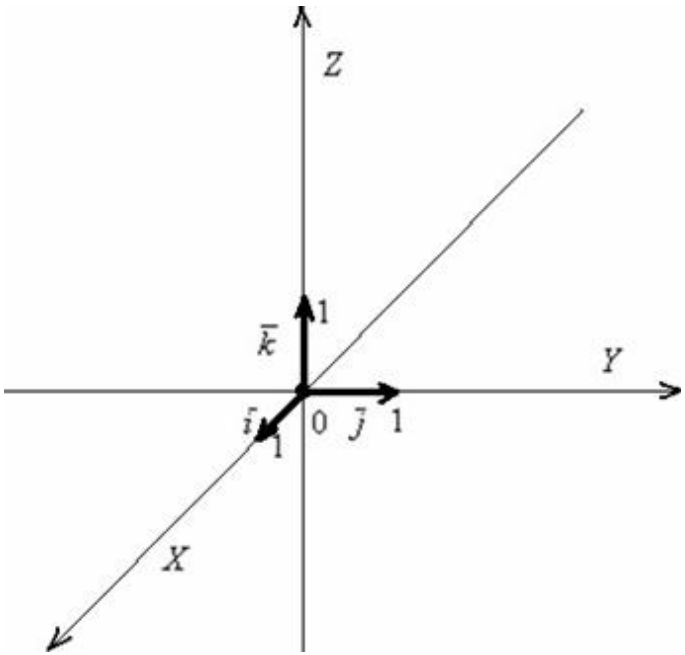
Р.Р.Нигматуллин

КФУ ИВМиИТ(ВМК)

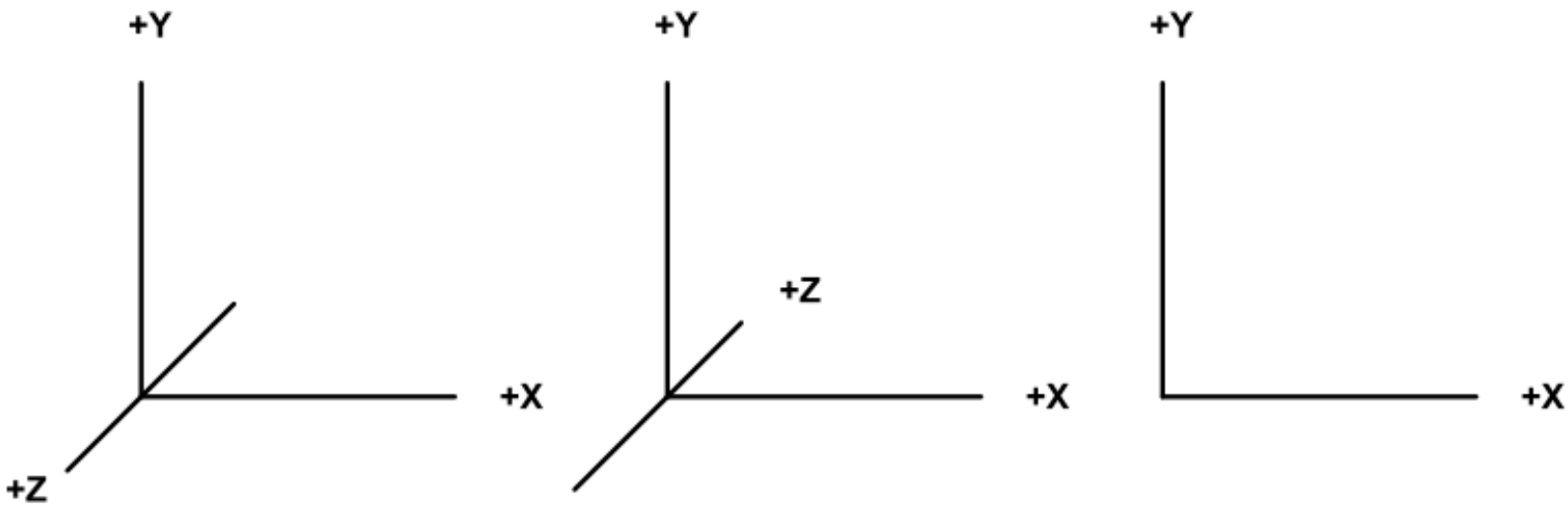
Казань, 2019

Система координат в трехмерной графике

Система координат – представляет собой точку отсчета и базис пространства.



Система координат в трехмерной графике



Системы координат : правосторонняя, левосторонняя, оконная.

Система координат в трехмерной графике

Отображение сцены на экране и её анимация основываются на понятиях о нескольких системах координат:

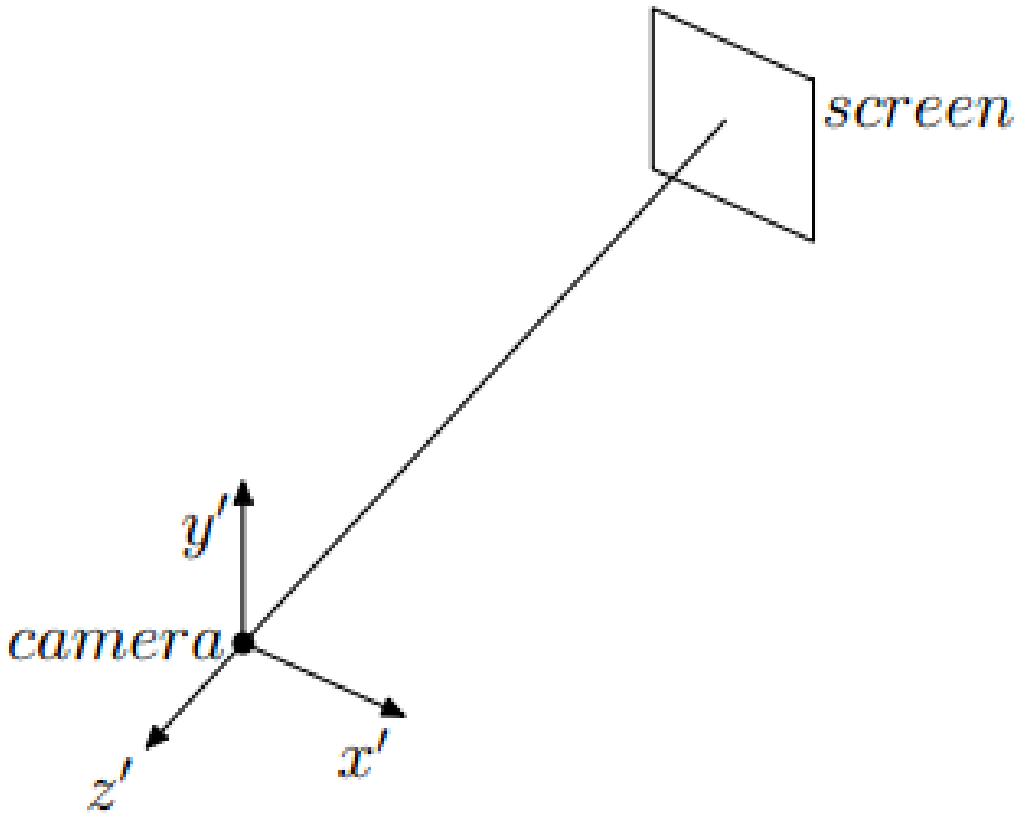
- **Локальная система координат**
- **Мировая система координат**
- **Видовая система координат**
- **Нормализованная система координат**
- **Оконная система координат**

Система координат в трехмерной графике

Расположение камеры:

- Координаты камеры совпадают с центром системы координат.
- Экран перпендикулярен направлению наблюдения, которое совпадает с направлением Oz.
- Положительное направление оси Oz, направлено к камере, оси Ox и Oy параллельны сторонам экрана. Ось Ox смотрит вправо, ось Oy вверх (правая система координат).

Система координат в трехмерной графике



Объекты и их атрибуты

- вершины
- ребра
- грани
- нормали
- текстуры
- материалы

Переход от одной системы координат к другой

Переход от одной системы координат к другой осуществляется с помощью матрицы перехода M.
Обратный переход осуществляется с помощью матрицы M^{-1}

$$x^{'} = Mx$$

Нотация

Вершины могут быть представлены как векторы-столбцы или векторы-строки. В зависимости от представления, переходы и преобразования записываются тем или иным способом. Так же представление вершин влияет на порядок применения последовательных переходов или преобразований.
Вектор стобец:

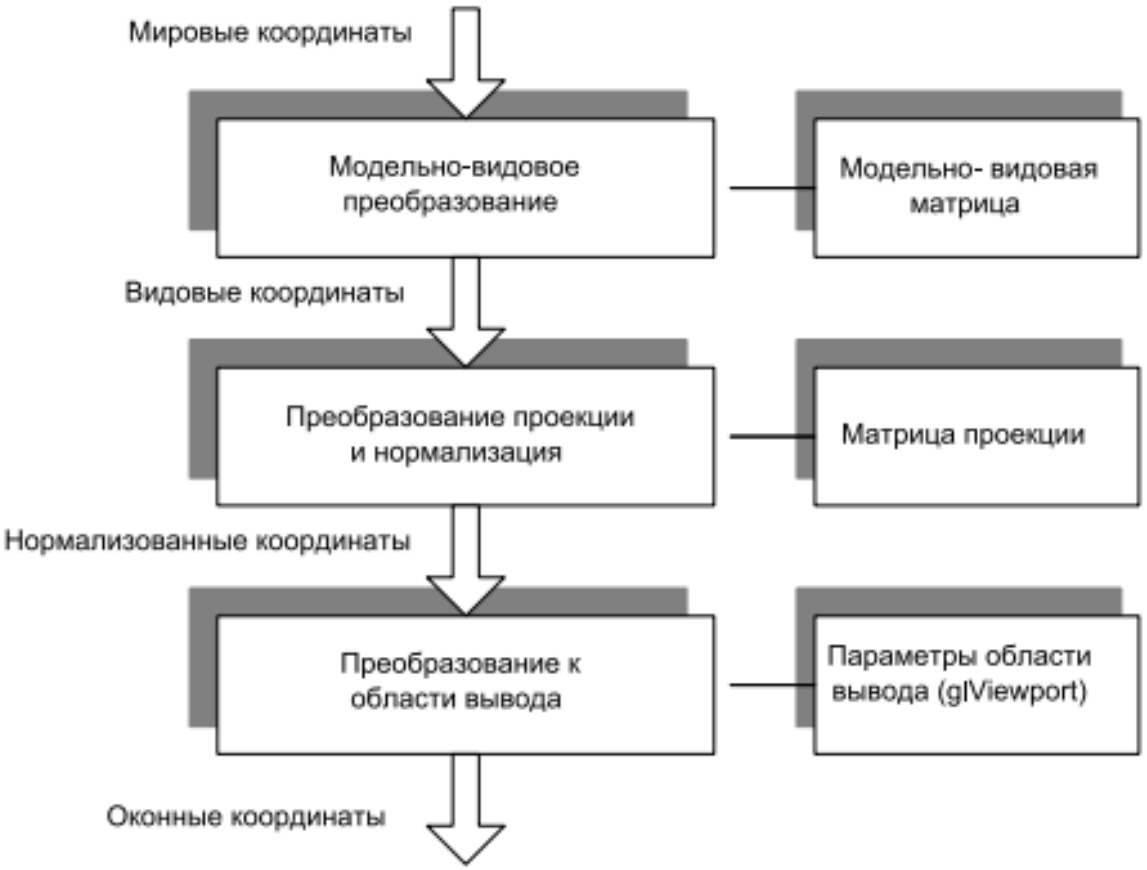
$$x^{'} = Mx$$

Вектор строка:

$$x^{'} = xM$$

Первое представление используется в OpenGL, второе в DirectX.

Отображение объектов



Преобразование объектов

Вид преобразования:

- Сдвиг (параллельный перенос)
- Поворот
- Масштабирование

Являются аффинными преобразованиями в трехмерном пространстве.

Преобразование объектов

Объекты на сцене представляются в проективных координатах (четырехмерные). Для преобразования объектов на сцене используются операции над матрицами. Преобразование сводится к умножению на соответствующую матрицу. Размер матриц 4x4.

$$
\begin{aligned}
x_1^{'} &= a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + c_1x_4 \\
x_2^{'} &= a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + c_2x_4 \\
x_3^{'} &= a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + c_3x_4 \\
x_4^{'} &= x_4
\end{aligned}
$$

В матричном виде:

$$X^{'} = AX$$

где,

$$X^{'} = \begin{pmatrix} x_1^{'} \\ x_2^{'} \\ x_3^{'} \\ x_4^{'} \end{pmatrix}, \quad X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}, \quad A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & c_1 \\ a_{21} & a_{22} & a_{23} & c_2 \\ a_{31} & a_{32} & a_{33} & c_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Параллельный перенос

$$A = \begin{pmatrix} 1 & 0 & 0 & c_1 \\ 0 & 1 & 0 & c_2 \\ 0 & 0 & 1 & c_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

где, вектор $c = (c_1, c_2, c_3)$ - вектор, на который осуществляется перенос

Поворот

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

где, A - ортогональная матрица поворота

Матрица поворота вокруг оси Ox на угол α.

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\alpha) & -sin(\alpha) & 0 \\ 0 & sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Матрица поворота вокруг оси Oy на угол α.

$$A = \begin{pmatrix} cos(\alpha) & 0 & sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(\alpha) & 0 & cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Матрица поворота вокруг оси Oz на угол α.

$$A = \begin{pmatrix} cos(\alpha) & -sin(\alpha) & 0 & 0 \\ sin(\alpha) & cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Масштабирование

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

Производится масштабирование в $a, b, c,$ раз по соответствующей оси.

Преоразование нормалей происходит не по той же матрице, что и вершины.

$\vec{n}$ - вектор нормали к грани

$\vec{v}$ - некоторый вектор на плоскости

Условие ортогональности:

$$\vec{n}^T \vec{v} = 0$$
$$\vec{n}^T I \vec{v} = 0$$
$$\vec{n}^T M^{-1} M \vec{v} = 0$$
$$\vec{n}^T M^{-1} \vec{v}' = 0$$
$$\vec{n}'^T \vec{v}' = 0$$
$$\vec{n}'^T = \vec{n}^T M^{-1}$$
$$\vec{n}' = (\vec{n}^T M^{-1})^T$$
$$\vec{n}' = (M^T)^{-1} \vec{n}$$

Размещение камеры

Задача: имеется описание сцены. Требуется в координатах сцены разместить камеру в точке $A = (x_0, y_0, z_0)$ так, чтобы она была направлена в точку $B(x_1, y_1, z_1)$.

Так же здесь добавляется условие на направление оси y' (системы координат камеры), т.к. без него данных для ориентации камеры недостаточно. Ось y' должна быть по возможности параллельна оси y (мировой системы координат).

Требуется найти аффинное преобразваоение, которое переведет вектор BA, в вектор параллельный оси z', а точку B в точку с координатами $(0, 0, -|BA|)$.

Размещение камеры

Решение получается в виде последовательности переноса и поворота.

1. Перенос сцены таким образом, чтобы точка A перешла в точку начала координат.
2. Поворот с помощью ортогональной матрицы.

Построение матрицы поворота:

1. Вектор $(0, 0, 1)$ переходит в вектор BA (после нормировки). Т.е. последний столбец матрицы будет иметь вид:

$$\gamma = c(x_0 - x_1, y_0 - y_1, z_0 - z_1)$$

2. Второй столбец должен быть ортогонален третьему столбцу.

$$\beta = (0, 1, 0) - \gamma[1]\gamma$$

3. Первый стоблец должен быть ортогонален двум другим:

$$\alpha = [\beta, \gamma]$$

В итоге получается следующая матрица:

$$\begin{pmatrix} \alpha^T & \beta^T & \gamma^T & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Полученная матрица умножается на текущую матрицу.

Поворот камеры

В общем случае поворот камеры осуществляется некоторым аффиным преобразованием с ортогональной матрицей.

Так же поворот можно осуществить комбинацией поворотов относительно осей Ox, Oy, Oz. Это соответвует перемножению матриц поворотов вокруг осей Ox, Oy, Oz.

Поворот камеры

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\alpha) & -sin(\alpha) & 0 \\ 0 & sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$ Поворот вокруг оси Ox.

$$\begin{pmatrix} cos(\alpha) & 0 & sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(\alpha) & 0 & cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$ Поворот вокруг оси Oy.

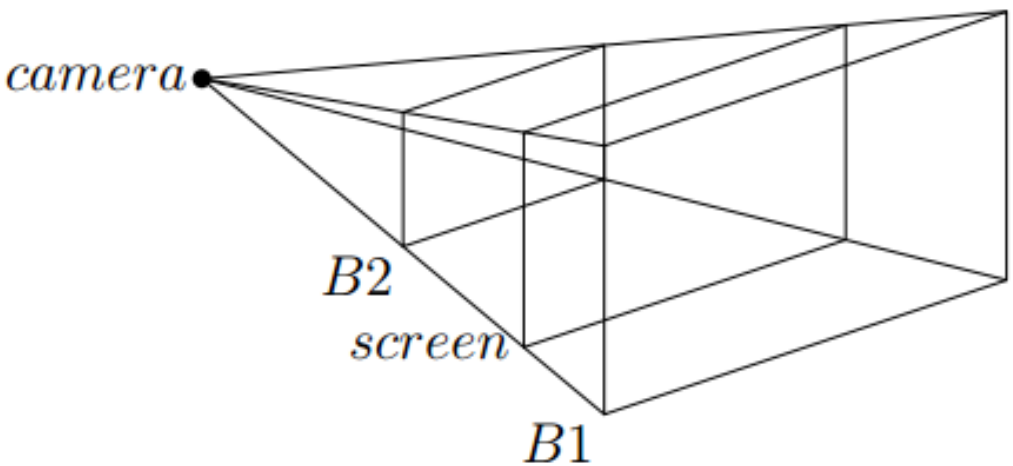
$$\begin{pmatrix} cos(\alpha) & -sin(\alpha) & 0 & 0 \\ sin(\alpha) & cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$ Поворот вокруг оси Oz.

Проекция

Для рендеринга необходимо осуществить преобразование координат объекта на экран для последующего отображения. Это осуществляется путем проекции.

Пирамида отсечения

Для переноса изображения на экран используют понятие пирамида отсечения. Пирамида отсечения представляет собой видимую часть сцены. Пространство пирамиды ограничено отсекающими плоскостями.



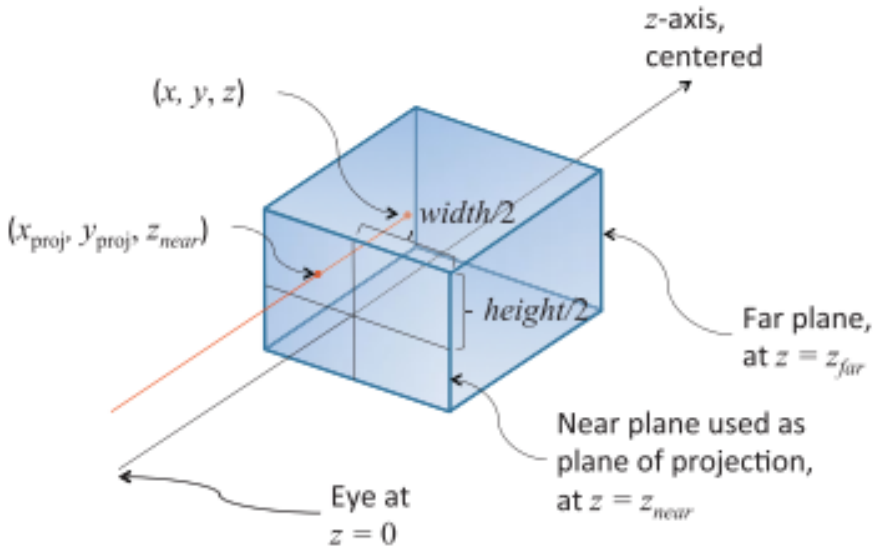
Проекция

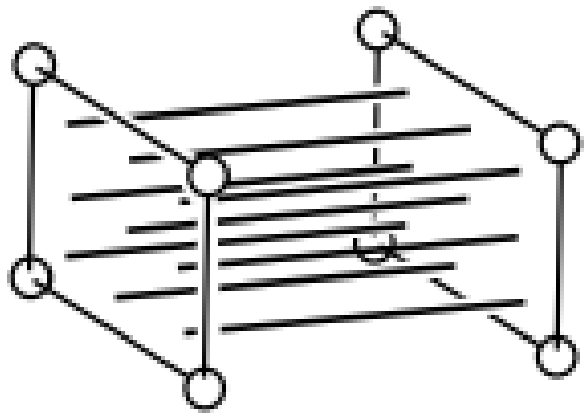
Рассмотрим два вида проекций, которые могут быть использованы для переноса координат со сцены на экран:

- Ортографическая проекция
- Перспективная проекция

При этом координаты объектов представляются в системе координат, связанной с камерой.

Ортографическая (параллельная проекция)





При оортографической проекции сохраняются координаты x, y, а координата z меняется на расстоание до экрана.

Проекция I

Вывод матрицы ортографической проекции.
После проецирования существует необходимость и в z координате точек, поэтому проецирование рассматривается как перевод пространства пирамиды отсечения в некоторый куб. При это накладывается ограничение, что куб должен иметь координты вершин ± 1.
$[l, r] -> [-1, 1]$
$[b, t] -> [-1, 1]$
$[n, f] -> [-1, 1]$
Рассморим ортографическую проекцию некоторой точки $P(x, y, z)$ из пространтсва пирамиды отсечения.
$P'(x', y', z')$ - получаемая при проецировании точка.
Координата x в пирамиде отсечения имеет следующие ограничения:

Проекция II

$$l < x < r$$

$$l < x < r \quad | - l$$

$$0 < x - l < r - l$$

$$0 < x - l < r - l \qquad | / (r - l)$$

$$0 < \frac{x - l}{r - l} < 1$$

$$0 < \frac{x - l}{r - l} < 1 \quad | \cdot 2$$

$$0 < \frac{2(x - l)}{r - l} < 2$$

$$0 < \frac{2(x-l)}{r-l} < 2 \quad |-1$$

$$-1 < \frac{2(x-l)}{r-l} - 1 < 1$$

$$-1 < \frac{2(x-l)}{r-l} - \frac{r-l}{r-l} < 1$$

$$-1 < \frac{2x-2l-r+l)}{r-l} < 1$$

$$-1 < \frac{2x}{r-l} - \frac{r+l}{r-l} < 1$$

Откуда:

$$x' = \frac{2x}{r-l} - \frac{r+l}{r-l}$$

Аналогичным образом получаем y':

$$y' = \frac{2y}{t-b} - \frac{t+b}{t-b}$$

Для координаты z ограничения записываются с учетом того факта, что камера направлена в сторону отрицательной оси Oz, а значения n и f представляют собой расстояния до передней и заданей отсекающих плолскостей:

$$n < -z < f$$

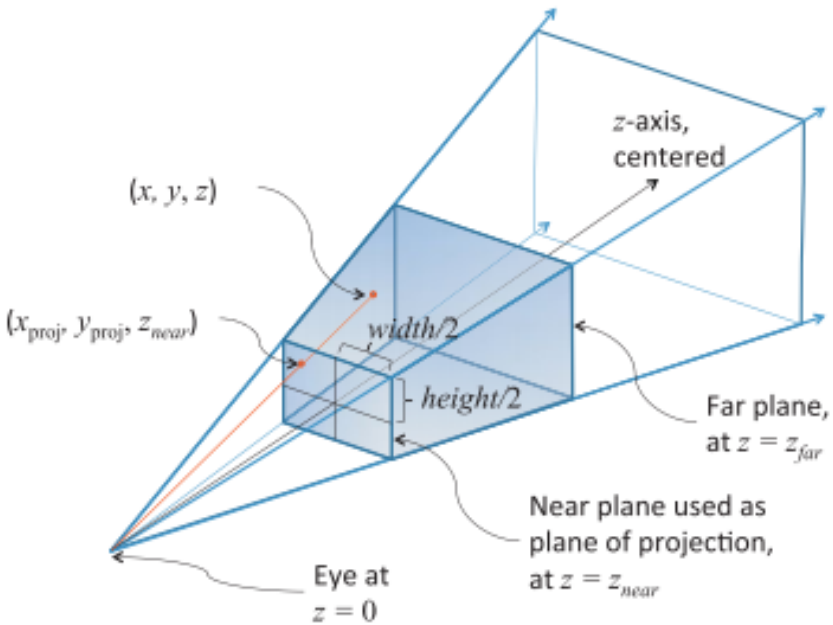
Проделав аналогичный вывод как и для x, y получаем следующий результат для z':

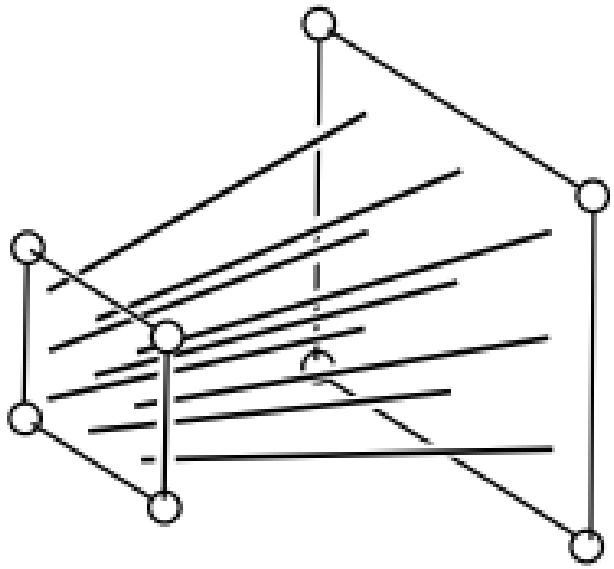
$$z' = \frac{-2z}{f-n} - \frac{f+n}{f-n}$$

Соединив выражения для x', y', z' получаем матрицу ортографической проекции:

$$\begin{pmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

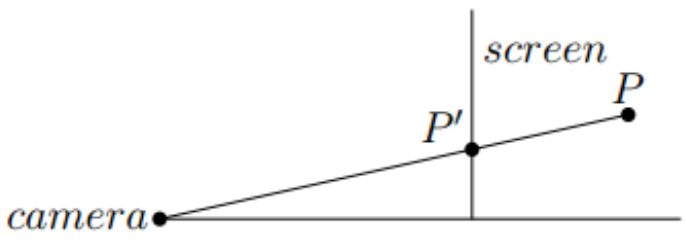
Перспективная проекция





Перспективная проекция

$$x_s = xD/z$$
$$y_s = yD/z$$
$$z_s = D$$



Проективное преобразование

В проективный координатах преобразование рендеринга сводится к следующей формуле:

$$\begin{pmatrix} x_1' \\ x_2' \\ x_3' \\ x_4' \end{pmatrix} = \begin{pmatrix} D & 0 & 0 & 0 \\ 0 & D & 0 & 0 \\ 0 & 0 & D & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

Вывыд матрицы перспективной проекции.
Воспользуемся формулами для координат проекции и ограничениями на размер экрана.
$P'(x', y', z')$ - получаемая при проецировании точка.

$$l < x' < r$$

$$l < x' < r \quad | - l$$

$$0 < x' - l < r - l$$

$$0 < x' - l < r - l \quad | / (r - l)$$

$$0 < \frac{x' - l}{r - l} < 1$$

$$0 < \frac{x' - l}{r - l} < 1 \quad | \cdot 2$$

Проекция II

$$0 < \frac{2(x' - l)}{r - l} < 2$$

$$0 < \frac{2(x' - l)}{r - l} < 2 \quad | -1$$

$$-1 < \frac{2(x' - l)}{r - l} - 1 < 1$$

$$-1 < \frac{2(x' - l)}{r - l} - \frac{r - l}{r - l} < 1$$

$$-1 < \frac{2x' - 2l - r + l)}{r - l} < 1$$

$$-1 < \frac{2x'}{r - l} - \frac{r + l}{r - l} < 1$$

Ранее было показано, что :

$$x' = -xn/z$$

$$-1 < \frac{-2xn}{z(r-l)} - \frac{r+l}{r-l} < 1$$

Аналогичным образом получаем y':

$$-1 < \frac{-2yn}{z(t-b)} - \frac{t+b}{t-b} < 1$$

Для вычисления преобразования z координаты воспользуемся тем фактом, что x и y не влияют на вычисления. Получаем следующее выражение для преобразования координаты z:

$$Az + B$$

A и B неизвестные коэффициенты, которые необходимо определить. Воспользуемся граничными условиями и запишем равенства в декартовых координатах:

$$\begin{cases} \frac{-An+B}{n} = -1, & z = -n \\ \frac{-Af+B}{f} = 1, & z = -f \end{cases}$$

$$\begin{cases} -An + B = -n \\ -Af + B = f \end{cases}$$

Выразим B через A в первом уравнении и подставим во второе уравнение:

$B = -n + An$

$-fA - n + An = f$, откуда $A = -\frac{f+n}{f-n}$

Подставляя полученный результат в выражение для B, получаем следующее:

$B = -\frac{2fn}{f-n}$

Собирая все результаты вместе получаем матрицу перспективной проекции:

Проекция II

$$\begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Нормализованные координаты

После применения матрицы проекций на вход следующего преобразования подаются так называемые нормализованные координаты, для которых значения всех компонент (x_n, y_n, z_n) находятся в диапазоне $[-1, 1]$.

Область вывода

Область вывода, представляет собой прямоугольник в оконной системе координат, имеющий определенное положение и размеры. Пусть левый нижний угол прямоугольника имеет координаты (x, y), а размеры прямоугольника равны *width* и *height*. Тогда оконные координаты центра области вывода вычисляются по формулам:

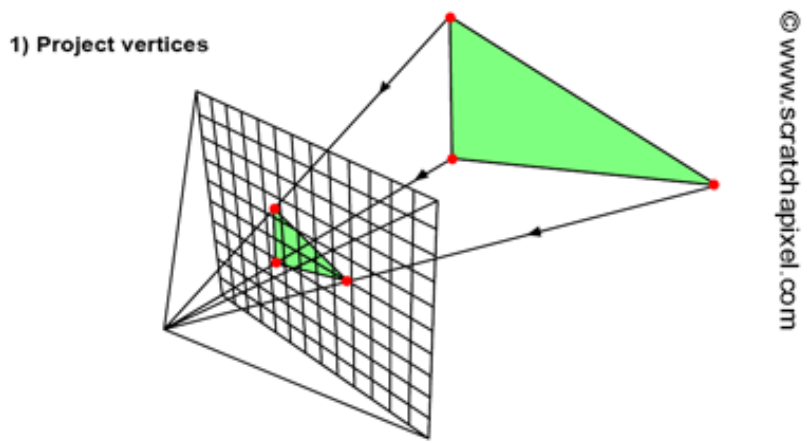
$$o_x = x + width/2$$
$$o_y = y + height/2$$

Оконные координаты вершины при этом будут вычисляться следующим образом:

$$x_w = (width/2) * x_n + o_x$$
$$y_w = (height/2) * y_n + o_y$$

Растеризация

После получения оконных координат вершин грани (в области вывода), производится отображение проекции грани на самом изображении. Этот процесс называется растеризацией.
Для растеризации необходимо знать цвет грани, которым будет отображена проекция грани, либо иметь соответвующую текстуру.
При растеризации необходимо определить множество пикселей на выходном изображении, которые соответствуют проекции грани.

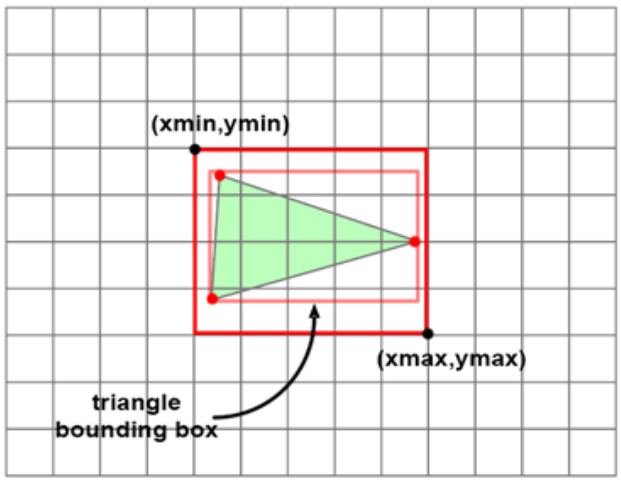


Для определения множества пикселей, относящихся к проекции грани, можно воспользоваться следующим подходом:

1. Находим минимальные и максимальные координаты по каждой из осей среди вршин грани и строим ограничивающий прямоугольник:

$$x_{min} = min\{v_{0x}, v_{1x}, v_{2x}\} \quad x_{max} = max\{v_{0x}, v_{1x}, v_{2x}\}$$

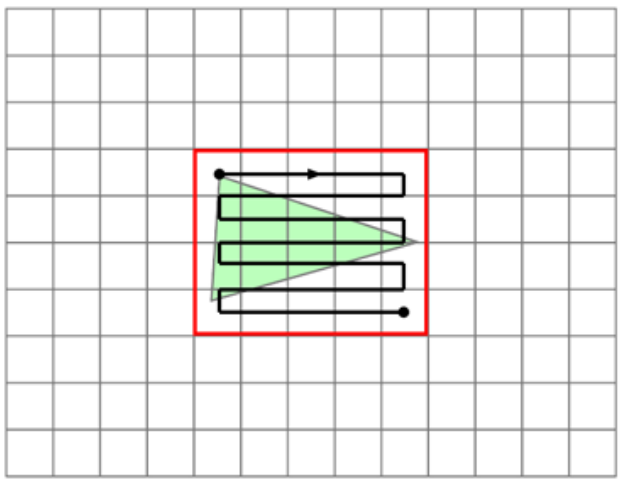
$$y_{min} = min\{v_{0y}, v_{1y}, v_{2y}\} \quad y_{max} = max\{v_{0y}, v_{1y}, v_{2y}\}$$



2. Проходим по пикселям из ограничивающего прямоугольника и проверяем принадлежность каждого пикселя проекции грани, образованной вершинами. Проверку принадлежности можно осуществить с помощью барицентрических координат пикселя относительно вершин:

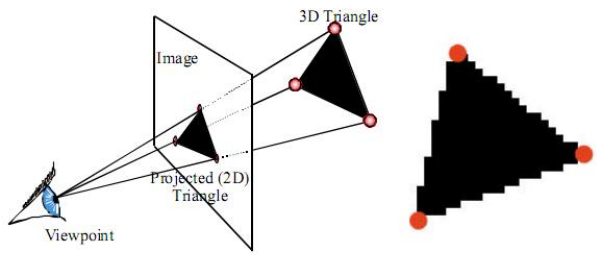
$$p(x, y) = a \cdot v_0(x_0, y_0) + b \cdot v_1(x_1, y_1) + c \cdot v_2(x_2, y_2)$$

$$a \geq 0, \quad b \geq 0, \quad c \geq 0$$



© www.scratchapixel.com

3. Для каждого пикселя из проекции грани происходит заполнение цветом:
- цветом грани
- цветом из текстуры, полученном на основе интерполяции по барицентричесим координатам
- овсещение, вычисленное по определенной модели с возможным учетом материала. Нормали для модели вычисляются на основе интерполяции по барицентричесим координатам

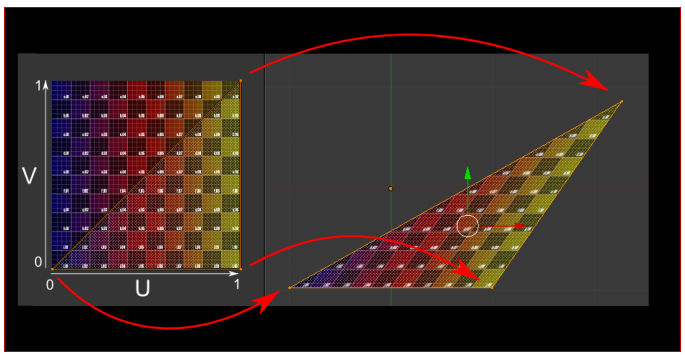


Текстура

Текстура представляет собой некоторую функцию $F(u, v)$ определенную на единичном кадрате.
Примеры текстур:
- цвет $(R(u, v), G(u, v), B(u, v))$
- интенсивность $I(u, v)$
- нормали $(N_x(u, v), N_y(u, v), N_z(u, v))$
- прозрачность $A(u, v)$
- высота поверхности $H(u, v)$

Текстура

Элементы текстуры имеют координаты (u, v), определенные на единочном квадрате. Значения (0,0) соответствуют левому нижнему углу текстуры, значения (1,1) - правому верхнему. С помощью текстурных координат происходит сопоставление точек поверхности модели и элементов текстуры. К описанию вершины добаляются текстурные координаты, по котором в текстуре находится элемент соответствующий вершине.



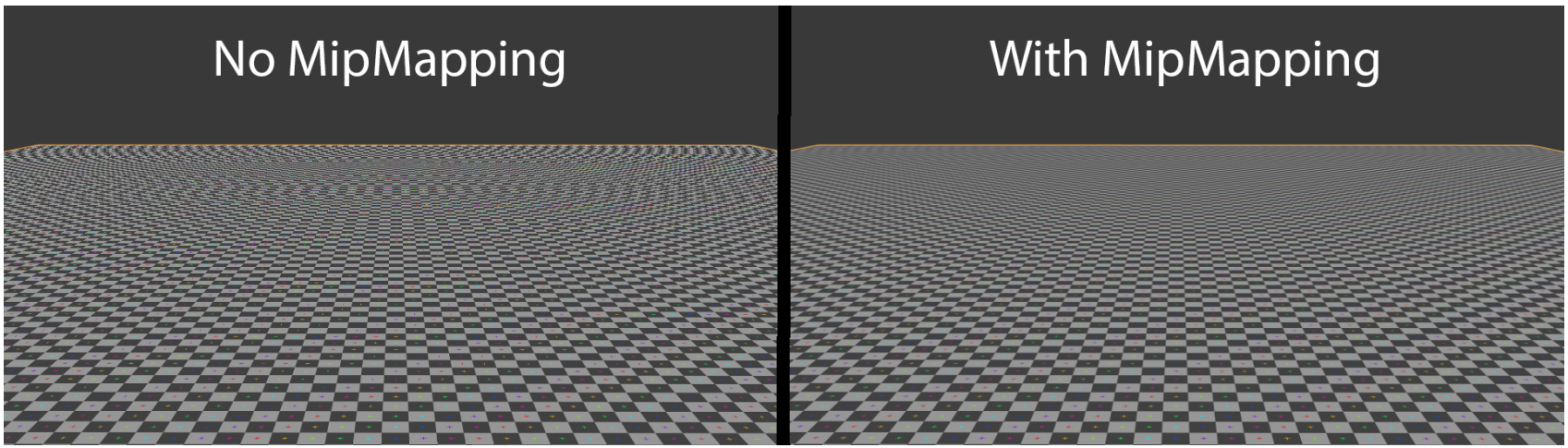
Для получения элемента текстуры некоторого растеризованного пикселя $p(x, y)$ необходимо сначала вычислить его текстурные координаты (u, v) путем интерполяции через барицентрические координаты (a, b, c) относительно вершин проекции грани v_0, v_1, v_2.

$$p(x, y) = a \cdot v_0(x_0, y_0) + b \cdot v_1(x_1, y_1) + c \cdot v_2(x_2, y_2)$$

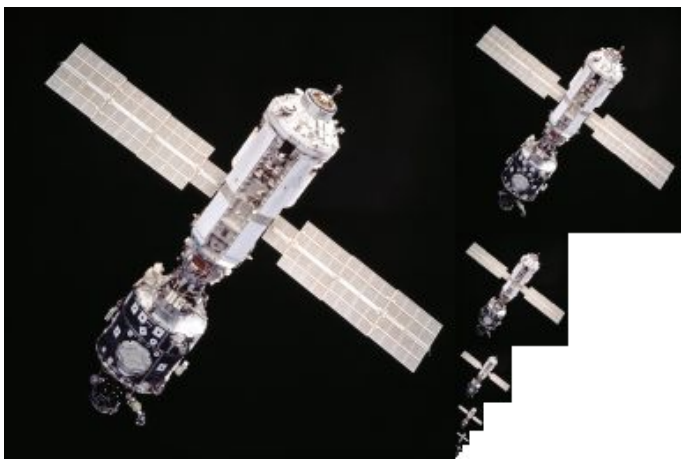
$$(u, v) = a \cdot (u_0, v_0) + b \cdot (u_1, v_1) + c \cdot (u_2, v_2)$$

После получения текстурных координат, нужный элемент из массива текстуры можно получить по индексам $u \cdot w$, $v \cdot h$, где w и h ширина и высота массива текстуры.

При отображении текстуры может возникнуть поблема связанная с различным разрешеним текстуры и экрана. Вследствие чего возникают артефакты в виде размытия, потери мелких деталей и мерцания. Излишне высокое разрешение текстуры сказывается на производительность при передачи данных.



В качестве обхода указанных проблем предлагается использовать последовательность текстур с различным разрешением - MIP maps (от лат. multum in parvo — «много в малом».). Строится последовательность текстур от максимального разрешения до разрешения 1x1 с шагом 2.



Нужная структура выбирается с учетом расстояния от объекта до камеры.

$$mip_{level} = log_2(\frac{D}{texel_{size} \cdot resolution}) + mip_{bias},$$

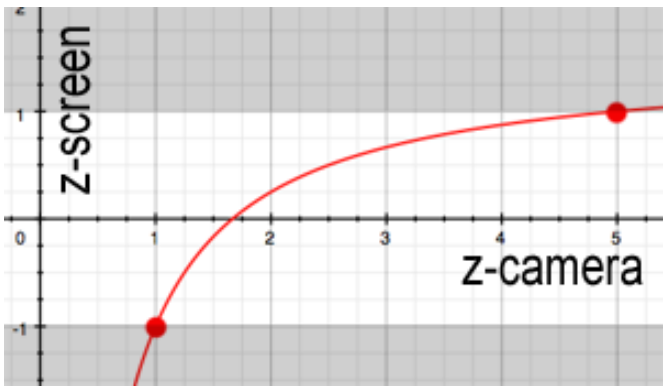
где: mip_{level} - номер текстуры из последовательности, D - расстояние до объекта, $texel_{size}$ - размер текселя (единица текстуры), *resolution* - разрешение камеры (количество пикселей, которое будет в объекте размером в 1 ед., расположенном в 1 ед. от камеры), mip_{bias} - сдвиг по нумерации.
Полученное число округляется до целого, и текстура с соответствующим номером уровня (нулевая — самая детальная, первая — вдвое меньшая и т. д.) накладывается на объект.

Расход памяти при использовании MIP maps увеличивается на 1/3.

$$\sum_{i=0}^{\infty}(\frac{1}{4})^i = 1\frac{1}{3}$$

z - fighting problem.
При вычислении z координаты в случае перспективной проекции возникает проблема с точностью представления результата. Точки, находящиеся ближе к передней отсекающей плоскости имеет более точное численное представление, чем точки расположенные ближе к задней отсекающей плоскости. Такая особенность связанна с формулой по которой идет расчет:

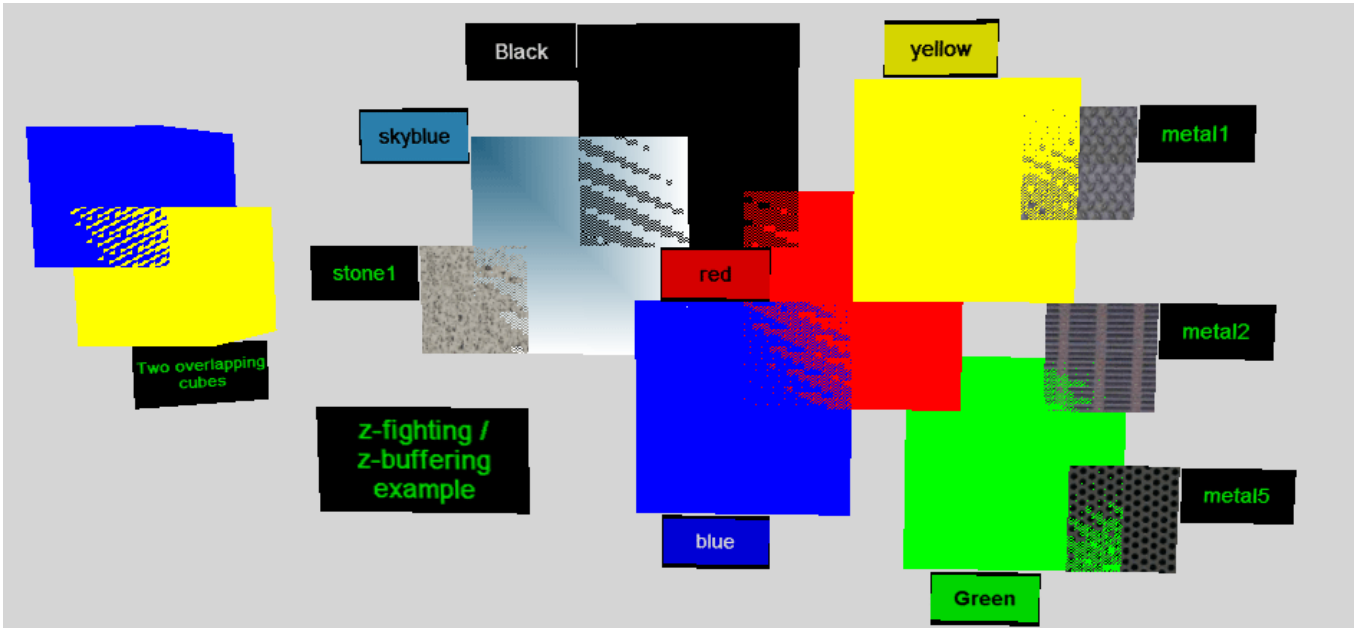
$$z_n = \frac{z(f+n)+2fn}{z(f-n)}$$



Проекция II

Это приводит к тому, что объекты, близкие к задней отсекающей плоскости и имеющие разные z-координаты в системе координат камеры, могут иметь одинаковую z-координату при перспективной проекции. Такая ситуация приводит к появлянию дефектов на выходном изображении при рендеринге.

z-fighting



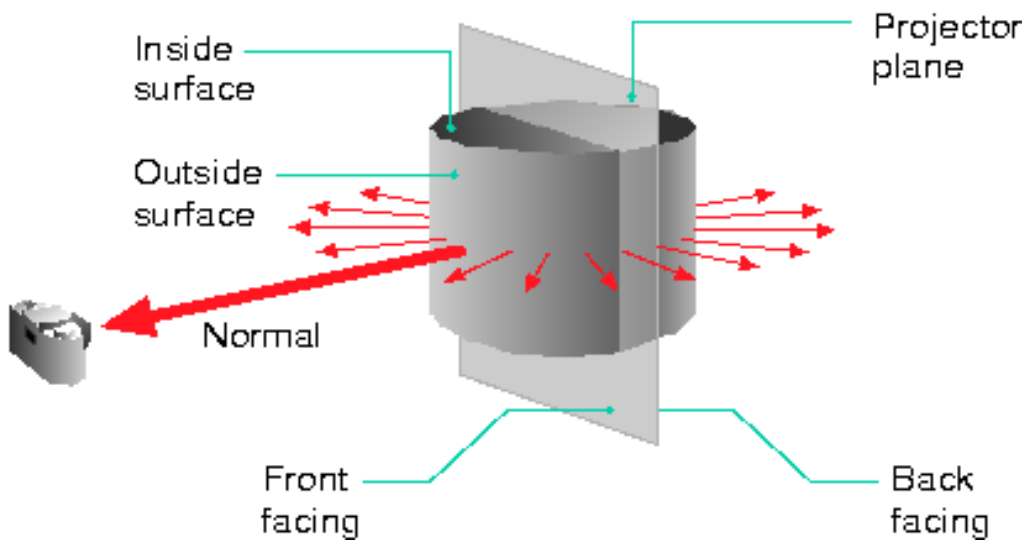
Возможные решения проблемы:

- Сдвиг объектов на небольшую величину для покрытия погрешности
- Уменьшение расстояния между передней и задней отсекающими плоскостями

Проблема перекрывающихся граней.
При проецировании моделей состоящих из перекрывающихся граней может произойти отображение "невидимой"(перекрытой) грани поверх видимой (с позиции камеры) грани. В результате получается некорректный рендеринг.
Даже, если последовательность граней при рендеринге соблюдена верно, то все равно будет тратиться время на отображение "невидимой"грани.

Один из возможных способов избежать такой ситуации - использование алгоритма back face culling. Алгоритм работает с нормалями граней и проверяет является ли грань видимой на основе направления нормали по отношению к вектору исходящему из камеры к грани.

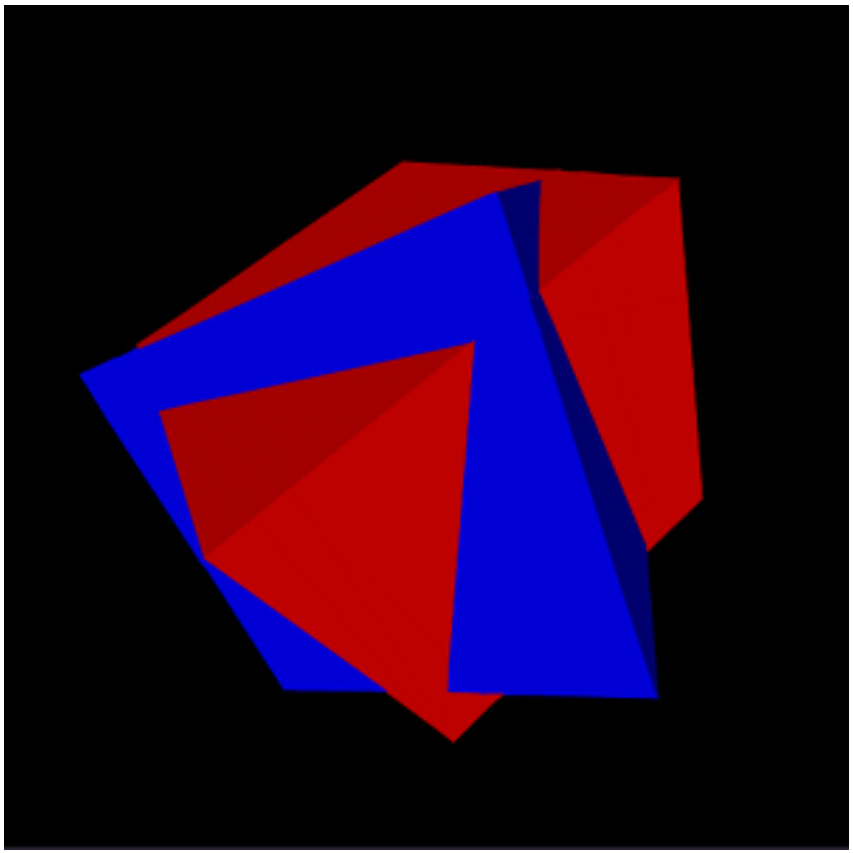
figure 206
Inside and
outside surface



Проверка осуществляется по знаку скалярного произведения вектора нормали и вектора направленного из позиции камеры к одной из вершин грани. $(\vec{V_i} - \vec{P}) \cdot \vec{N}$, где: $\vec{V_i}$ - одна из вершин, рассматриваемой грани; $\vec{P}$ - точка расположения камеры; $\vec{N}$ - вектор номали к грани. Вектор нормали грани может быть вычислен как векторное произведение от векторов на сторонах грани. При этом важно учитывать обход (по часовой стрелке или против):

$$\vec{N} = (\vec{V_1} - \vec{V_0}) \times (\vec{V_2} - \vec{V_1})$$

Проблемные ситуации для back-face culling:
- пересекающиеся грании
- модели с перекрывающимися видимыми гранями



Указанные проблемы можно решить с помощью алгоритма z-буфер (z-buffer, depth buffer).

Инициализация:

```
z_buf = array(image.shape)
z_buf[i,j] = 1
...
```

Цикл по всем граням:

Цикл по все растеризованным пикселям грани:

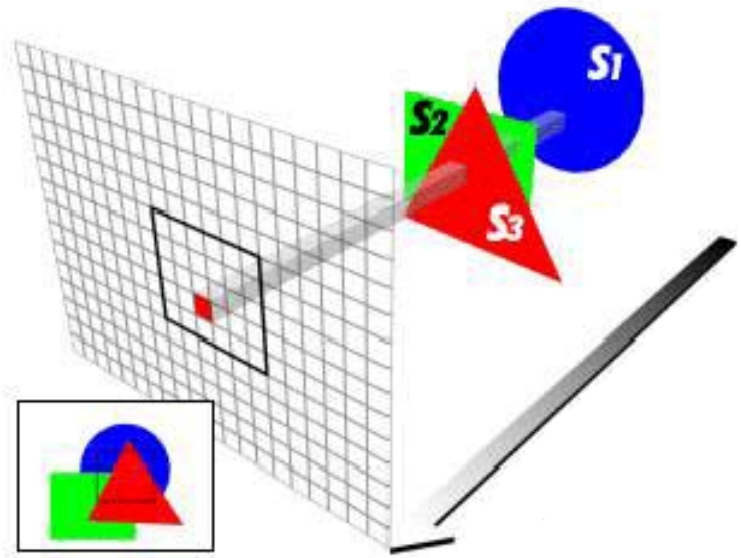
```
...
```

$a, b, c = get\_barycentric\_coords(p(x, y), v_0, v_1, v_2)$

$z = a \cdot v_{0z} + b \cdot v_{1z} + c \cdot v_{2z}$

```
If z < z_buf[x,y],
    image[x,y] = color
    z_buf[x,y] = z
Else,
    go to next pixel.
```

z-буфер.



z-буфер.