Компьютерная графика
Лекция 3

Трёхмерные преобразования координат

Преобразования координат

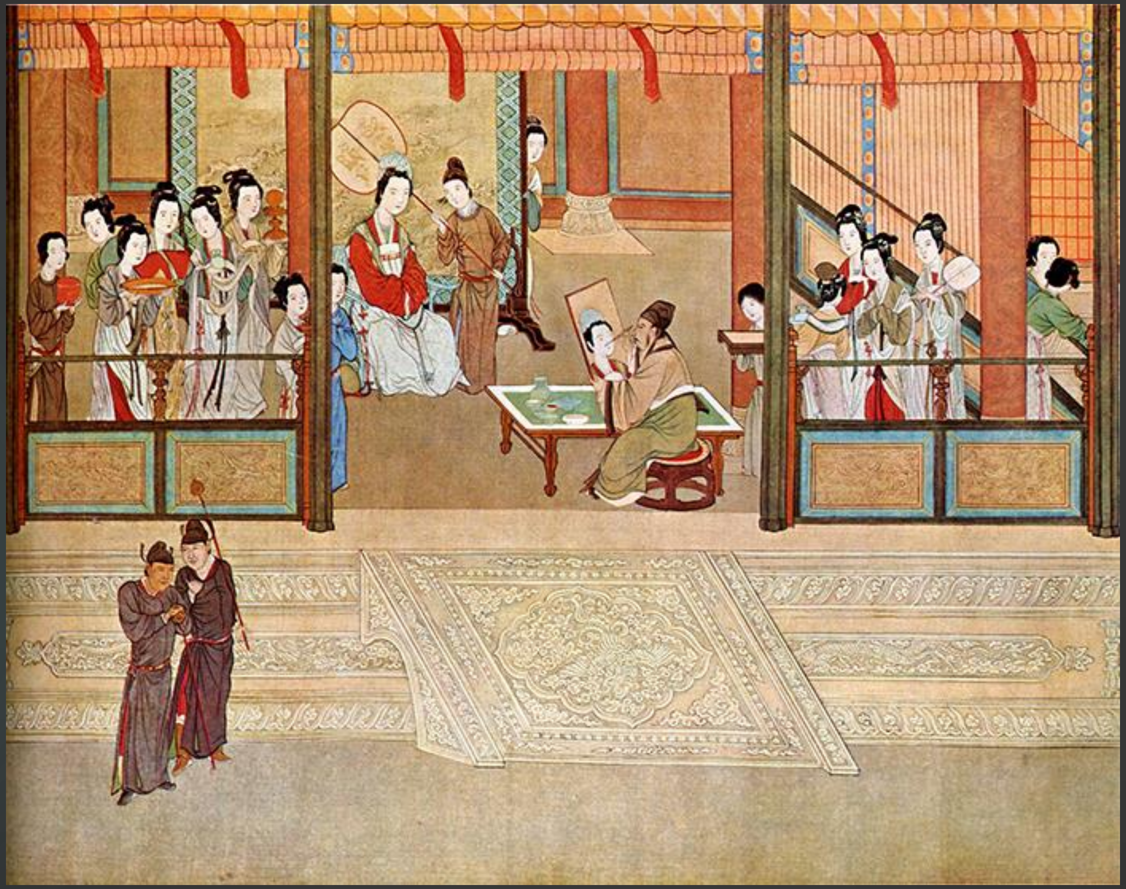
Трёхмерное аффинное преобразование

- Поворот и сдвиг объекта
- Поворот и сдвиг камеры

Проективное преобразование

- Проекция на плоскость
- Нормализованные координаты устройства (преобразование с учётом ближней и дальней плоскостей)

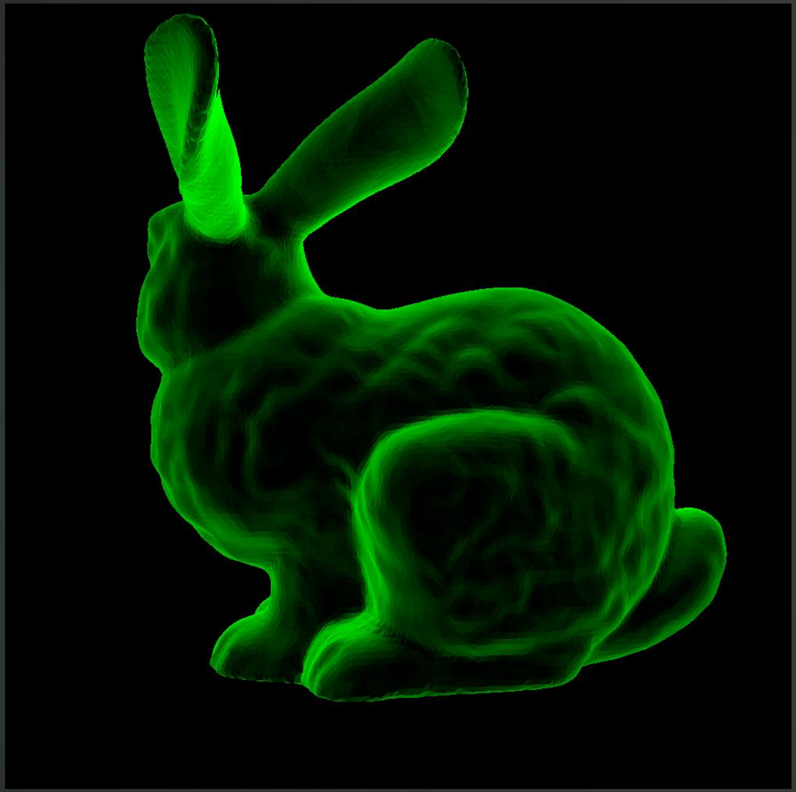
Перспектива на изображениях



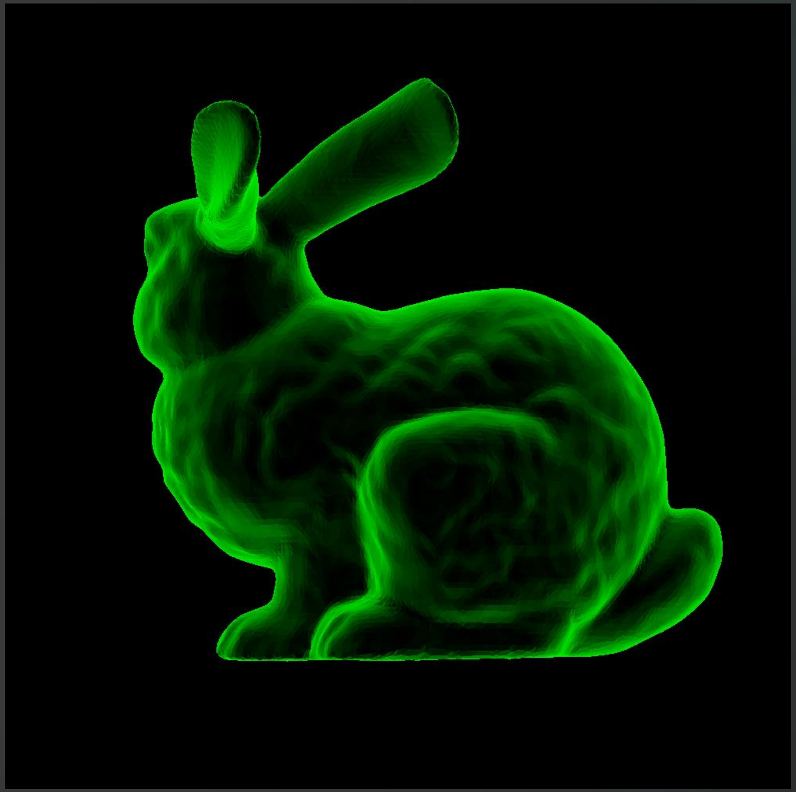
Влияние угла обзора камеры на вид изображения

Влияние угла обзора камеры на вид изображения

«Обычная» камера

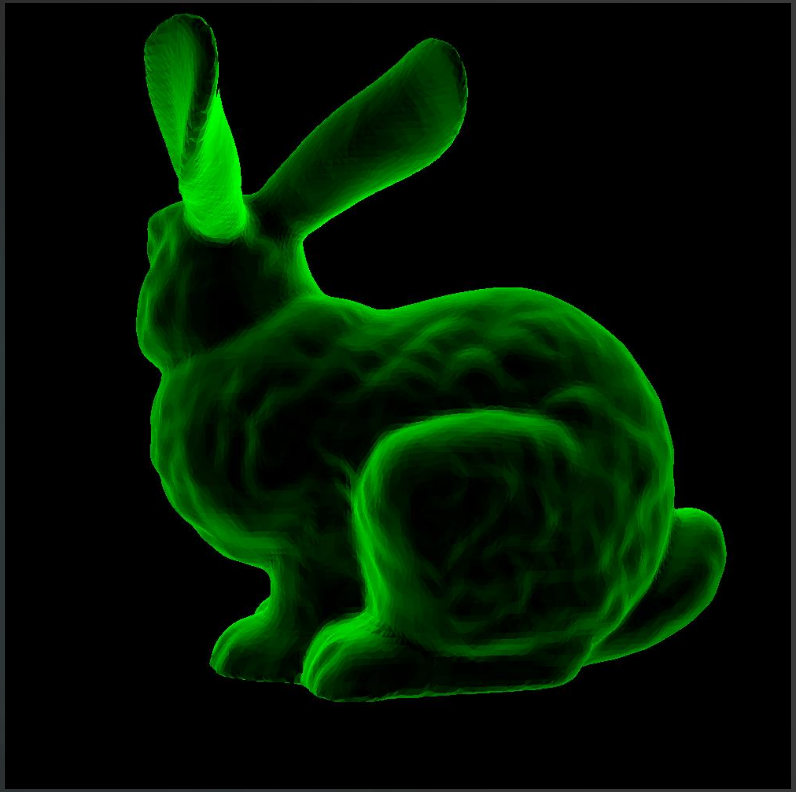


Узкоугольная камера

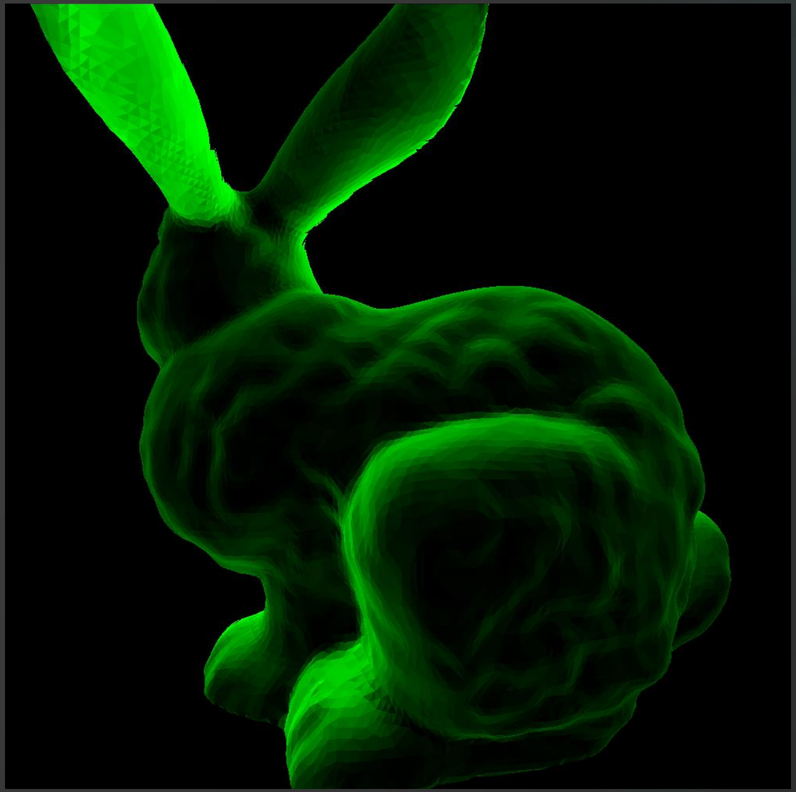


Влияние угла обзора камеры на вид изображения

«Обычная» камера



Широкоугольная камера



Проекция на плоскость Z=1

- Как добиться такого результата?
- Самый простой и понятный способ: поделить на координату Z.

$$x = \frac{X}{Z}, \qquad y = \frac{Y}{Z}$$

- Но это ещё не пиксельные координаты. Чтобы получить координаты в пикселях нужно умножить на коэффициент масштабирования и добавить координату центра камеры:

$$u = a_x \frac{X}{Z} + u_0, \qquad v = a_y \frac{Y}{Z} + v_0$$

Векторная форма проекции

- Как это выразить в векторной форме?

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} a_x & 0 & u_0 \\ 0 & a_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

- Здесь '~' – это знак пропорциональности, означающий, что результат матричного умножения надо разделить на последнюю координату.
- Вот эта матрица называется матрицей внутренних параметров камеры (intrinsic).

$$\begin{bmatrix} a_x & 0 & u_0 \\ 0 & a_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Комментарий по лабораторным

- Вы можете в своей программе использовать такой способ вычисления проекций. Он вполне функционален (по крайней мере в рамках наших задач).
- Обратите внимание, что выражение

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} a_x & 0 & u_0 \\ 0 & a_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

заменяет преобразования трёхмерных координат к пиксельным координатам. При этом исходные трёхмерные координаты по-прежнему должны храниться в памяти программы.

Расширенная версия

- Вместе с тем, OpenGL (как и в других графических движках) это реализовано несколько иначе.
- Обычно трёхмерные координаты преобразуются в проективные трёхмерные координаты следующим образом:

$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} \sim \begin{bmatrix} e_x & 0 & 0 & 0 \\ 0 & e_y & 0 & 0 \\ 0 & 0 & \dfrac{f+n}{f-n} & \dfrac{2fn}{f-n} \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- где e_x, e_y определяют угол обзора ($e = \frac{1}{\tan(FOV/2)}$) , а n и f – ближняя и дальняя плоскости, соответственно.

Проективное преобразование

- Давайте посмотрим, что получается в результате деления на последнюю координату

$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} \sim \begin{bmatrix} e_x & 0 & 0 & 0 \\ 0 & e_y & 0 & 0 \\ 0 & 0 & \dfrac{f+n}{f-n} & -\dfrac{2fn}{f-n} \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} e_x X \\ e_y Y \\ \dfrac{f+n}{f-n} Z - \dfrac{2fn}{f-n} \\ Z \end{bmatrix}$$

$$\begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} = \begin{bmatrix} \dfrac{e_x X}{Z} \\ \dfrac{e_y Y}{Z} \\ \dfrac{f+n}{f-n} - \dfrac{2fn}{f-n} \cdot \dfrac{1}{Z} \end{bmatrix}$$

Нормализованные координаты устройства

$$\begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} = \begin{bmatrix} \dfrac{e_x X}{Z} \\ \dfrac{e_y Y}{Z} \\ \dfrac{f+n}{f-n} - \dfrac{2fn}{f-n} \cdot \dfrac{1}{Z} \end{bmatrix}$$

- Эти координаты называются нормализованными координатами устройства. На изображение попадут только точки, находящиеся в пределах от -1.0 до 1.0 по каждой координате.
- Проверим точки ближней и дальней плоскости.

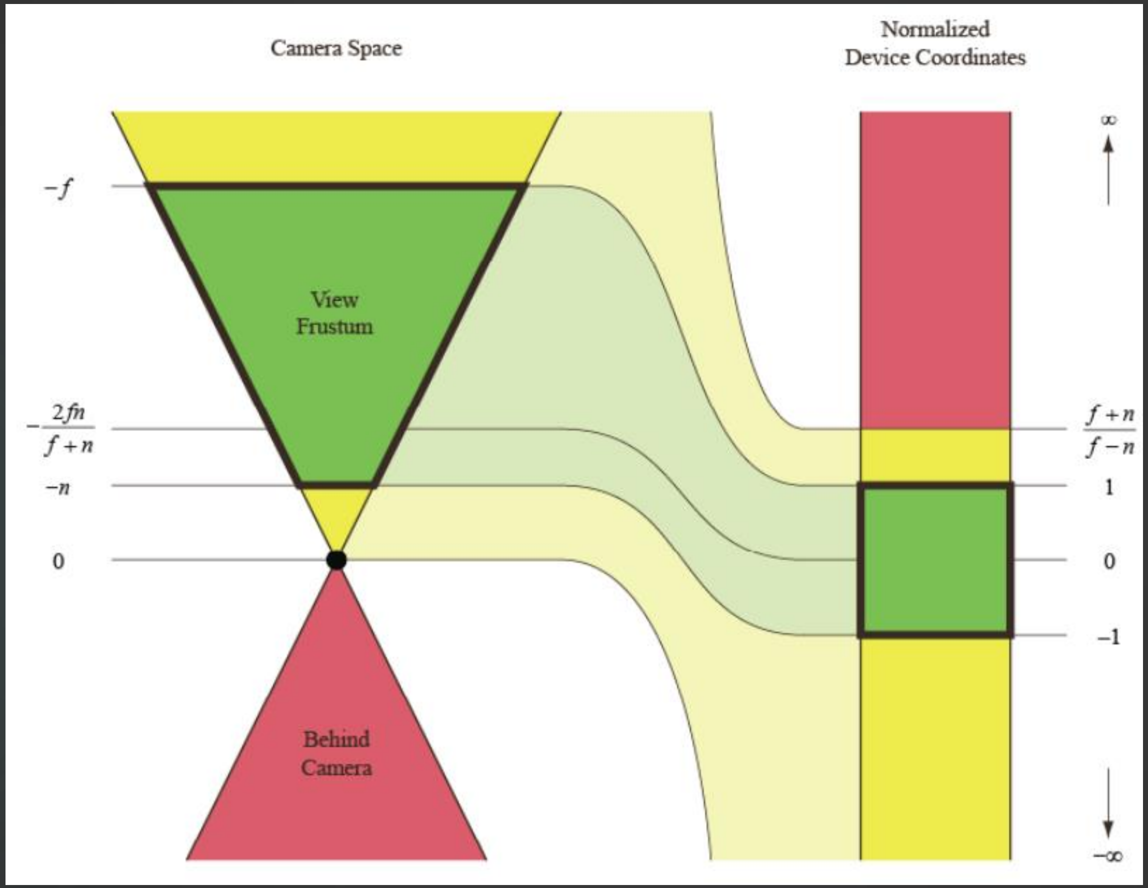
Ближняя и дальняя плоскости

$$Z = n: \qquad \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} = \begin{bmatrix} \dfrac{e_x X}{Z} \\ \dfrac{e_y Y}{Z} \\ \dfrac{f+n}{f-n} - \dfrac{2fn}{f-n} \cdot \dfrac{1}{n} \end{bmatrix} = \begin{bmatrix} \dfrac{e_x X}{Z} \\ \dfrac{e_y Y}{Z} \\ \dfrac{f+n}{f-n} - \dfrac{2f}{f-n} \end{bmatrix} = \begin{bmatrix} \dfrac{e_x X}{Z} \\ \dfrac{e_y Y}{Z} \\ \dfrac{n-f}{f-n} \end{bmatrix} = \begin{bmatrix} \dfrac{e_x X}{Z} \\ \dfrac{e_y Y}{Z} \\ -1 \end{bmatrix}$$

$$Z = f: \qquad \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} = \begin{bmatrix} \dfrac{e_x X}{Z} \\ \dfrac{e_y Y}{Z} \\ \dfrac{f+n}{f-n} - \dfrac{2fn}{f-n} \cdot \dfrac{1}{f} \end{bmatrix} = \begin{bmatrix} \dfrac{e_x X}{Z} \\ \dfrac{e_y Y}{Z} \\ \dfrac{f+n}{f-n} - \dfrac{2n}{f-n} \end{bmatrix} = \begin{bmatrix} \dfrac{e_x X}{Z} \\ \dfrac{e_y Y}{Z} \\ \dfrac{f-n}{f-n} \end{bmatrix} = \begin{bmatrix} \dfrac{e_x X}{Z} \\ \dfrac{e_y Y}{Z} \\ 1 \end{bmatrix}$$

Аналогично точки на левой/правой/верхней/нижней границах будут иметь координаты x_p и y_p равные -1 и 1.

Иллюстрация NDC



Из NDC в пиксельные координаты

- Финальное преобразование в пиксельные координаты тогда становится предельно простым:

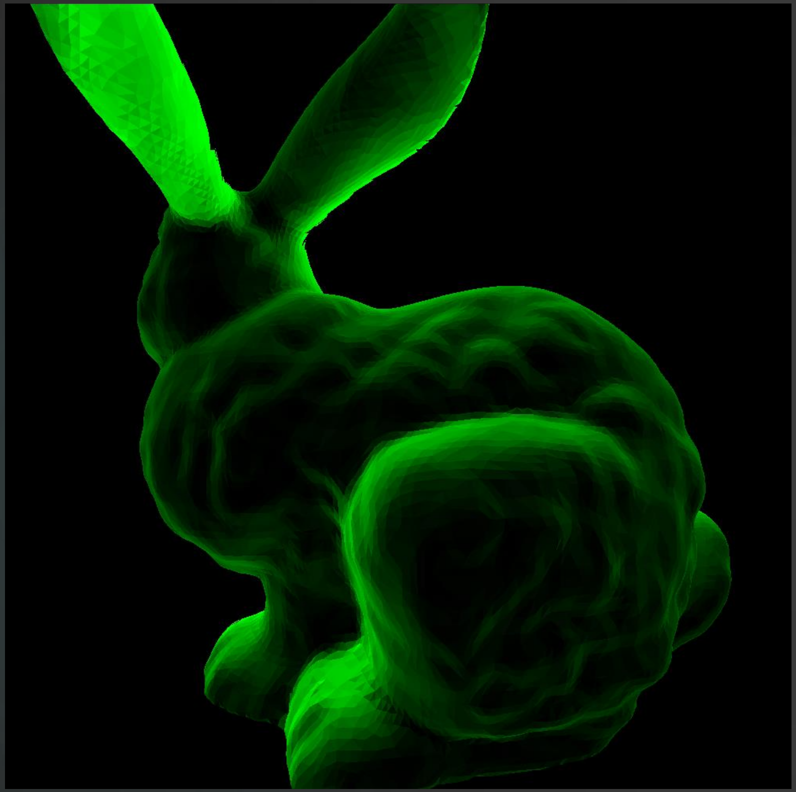
$$u = (1.0 + x_p) \cdot \frac{ImageSizeX}{2}, \qquad v = (1.0 + y_p) \cdot \frac{ImageSizeY}{2}$$

Комментарий к лабораторным

- Если вы попробуете воспользоваться этой формулой, то изображение не отрендерится.
- Проблема в том, что координаты Z точек исходной модели расположены вокруг нуля. Когда вы начинаете проецировать её координат на Z=1, происходят странные вещи. Этого можно избежать при помощи clipping plane (ограничив плоскости n и f), но нам всё ещё нужно нарисовать модель.
- На данном этапе используем простое решение. После загрузки модели из файла координаты (X,Y,Z) точек меняем на (X,Y,Z+Z0).
- Z0 в данном случае – точка, в которую мы поместим модель: для узкоугольной камеры больше, для широкоугольной меньше.

Ещё раз:

Широкоугольная камера



Узкоугольная камера