

# **Алгоритмические основы мультимедийных технологий**

Рябинин Константин Валентинович

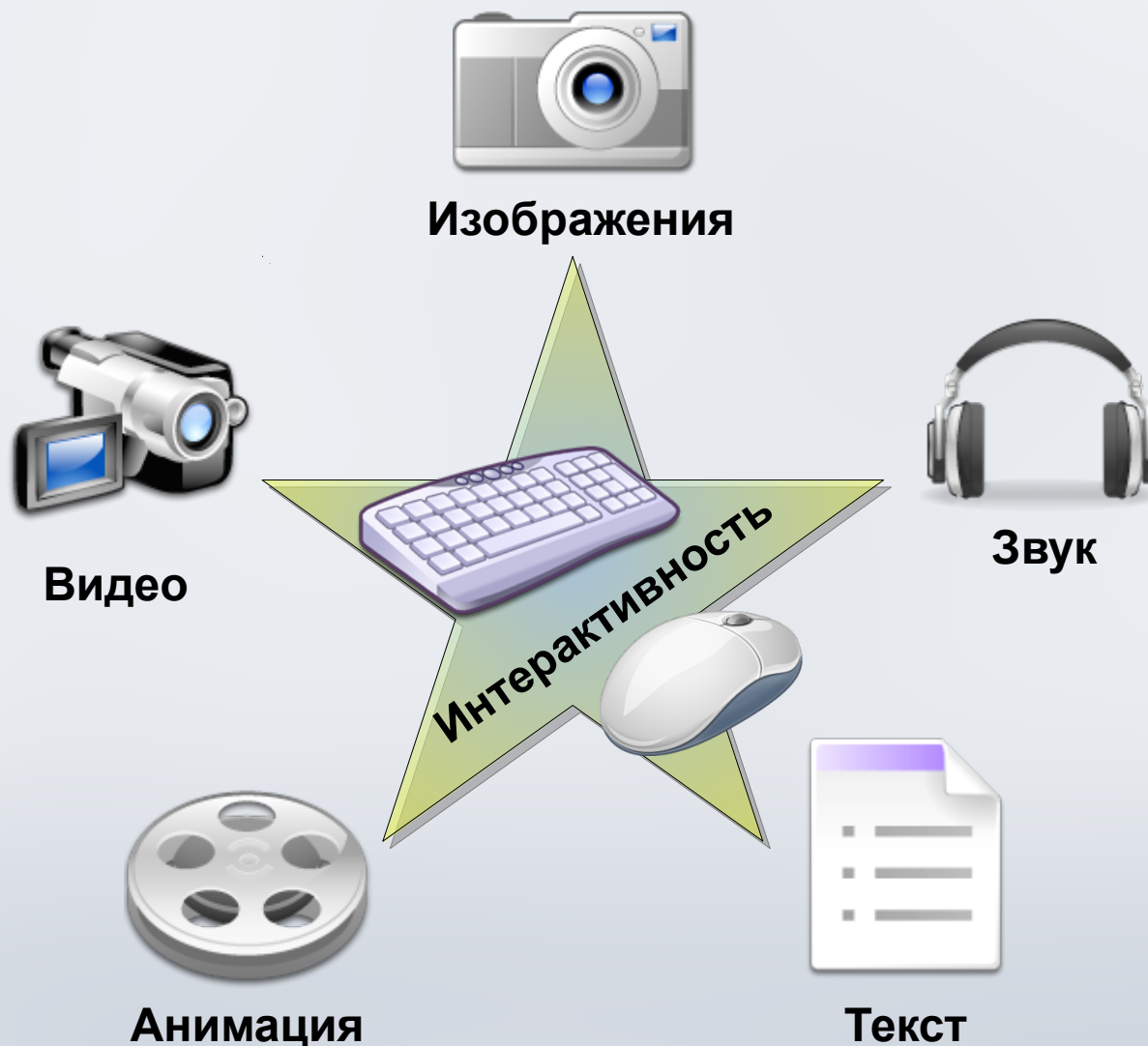
e-mail: [icosaeder@ya.ru](mailto:icosaeder@ya.ru)

jabber: [icosaeder@jabber.ru](mailto:icosaeder@jabber.ru)

Пермь, 2012

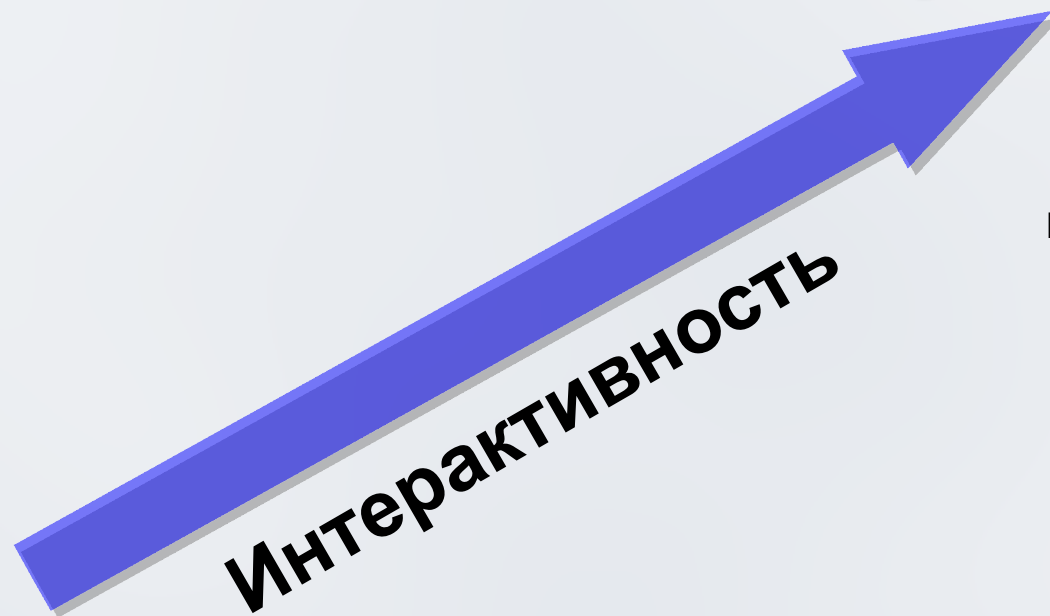


**Мультимедиа** – комплекс аппаратных и программных средств, позволяющих пользователю работать в диалоговом режиме с разнородными данными (графика, текст, звук, видео), организованными в виде единой информационной среды.





**Линейная  
мультимедиа**



**Нелинейная  
мультимедиа**

## Вопросы к рассмотрению:

- Программный синтез двумерных изображений
- Программный синтез трёхмерных изображений
- Обработка изображений
- Воспроизведение звука

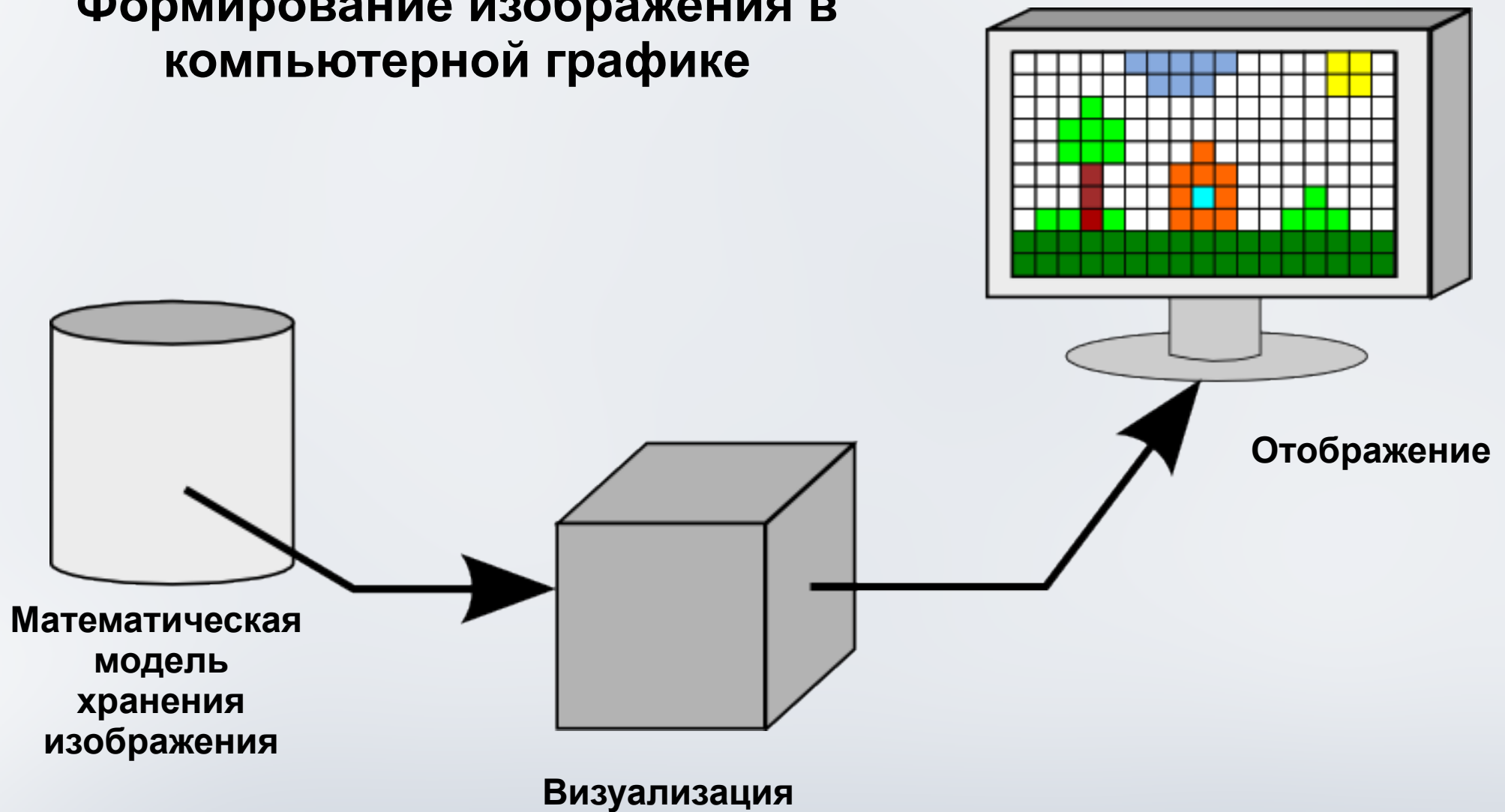
- **Форма отчётности – экзамен**
- **Для получения положительной оценки необходимо сдать индивидуальное задание: программную систему, удовлетворяющую заданным требованиям, а также ответить на теоретические вопросы**
- **Срок сдачи индивидуального задания – две последних пары курса**
- **Возможны автоматы (при сдаче в срок и ответе на теоретические вопросы сразу после сдачи)**
- **Контрольных не будет**

## Окружение демонстрации

- ОС Mac OS X & GNU / Linux
- Компилятор GCC, язык реализации C
- Библиотека визуализации стандарта  
OpenGL (+ GLUT)
- Библиотека воспроизведения звука стандарта  
OpenAL (+ ALUT)

- Ричард С. Райт-мл., Бенджамин Липчак. OpenGL суперкнига. М.: Вильямс, 2006
- Френсис Хилл. OpenGL программирование компьютерной графики. СПб: Питер, 2002
- Дональд Херн, Паулин Бейкер. Компьютерная графика и стандарт OpenGL. М.: Вильямс, 2005
- Эдвард Эйнджелл. Интерактивная компьютерная графика. Вводный курс на базе OpenGL. М.: Вильямс, 2001
- NeonHelium. URL: <http://nehe.gamedev.net/>
- Теоретические основы OpenGL. URL: <http://www.songho.ca/opengl/index.html>

## Формирование изображения в компьютерной графике





## Классификация графики по способу хранения (от способа хранения зависит визуализация)

	2D	3D
Растровая	Матрица точек	Воксели
Векторная	Описание контуров и заливок	Описание многоугольников

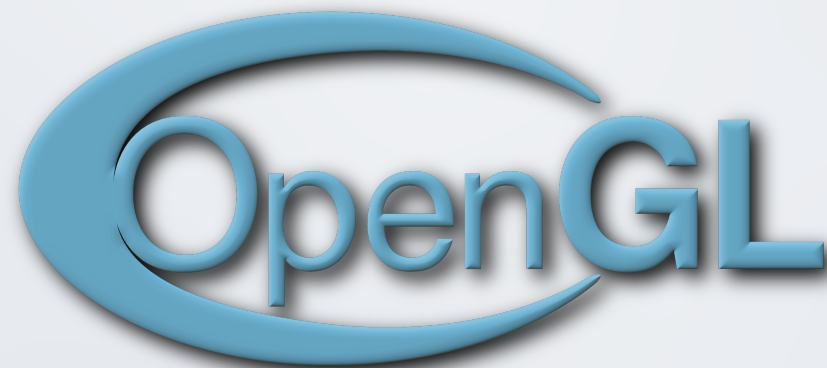
## Классификация графики по способу хранения (от способа хранения зависит визуализация)

	2D	3D
Растровая	Матрица точек	Воксели
Векторная	Описание контуров и заливок	Описание многоугольников

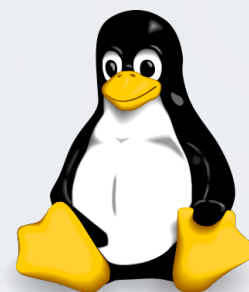
**Описание многоугольников – описание вершин и связей между ними**



Microsoft Windows



Microsoft Windows



GNU / Linux



FreeBSD



Mac OS

...

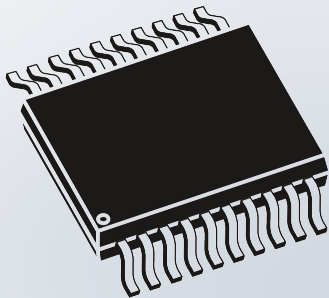
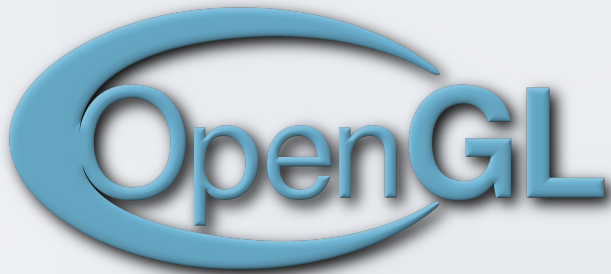
## Устройство графического приложения:



**Логика**



**Движок**



**Графическое  
оборудование**



**OpenGL** – спецификация, определяющая независимый от языка программирования кросс-платформенный программный интерфейс для написания приложений, использующих двумерную и трёхмерную компьютерную графику.

***OpenGL* – открытый интерфейс к графическому оборудованию для создания трёхмерных изображений в реальном времени**

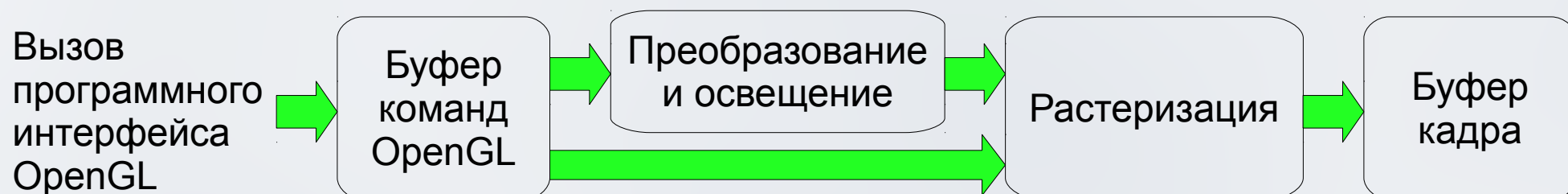
**Основные особенности:**

- Полная универсальность
- Низкий уровень функций

**Существует большое количество как полностью программных, так и программно-аппаратных реализаций стандарта OpenGL**

- Стандарт предполагает полностью процедурный характер работы
- Стандарт включает только функции работы с графикой, не затрагивая управление окнами, работу с файлами, взаимодействие с пользователем и т. д.
  - Для этих целей существует библиотека GLUT
- Предполагается, что библиотека стандарта OpenGL напрямую взаимодействует с графическим драйвером

## Конвейер



## Машина состояний

- Хранилище различных настроек, представляющих собой множество пар свойство-значение
- Бинарные атрибуты изменяются функциями  
`void glEnable(GLenum cap)`  
`void glDisable(GLenum cap)`
- Небинарные атрибуты изменяются специализированными функциями
- Для состояний есть свой стек, управляемый функциями  
`void glPushAttrib(GLbitfield mask)`  
`void glPopAttrib(void)`



- Атомарная управляемая единица геометрии – **вершина**
- Вершина имеет набор атрибутов (типа float), интерпретация которых, вообще говоря, лежит на программисте:
  - Координаты в пространстве
  - Координаты нормали
  - Координаты текстуры
  - Цвет
  - ...
- Вершины объединяются в примитивы:
  - Линии
  - Многоугольники (треугольники)

- Для произведения растеризации, системе для каждой вершины необходимо получить **от программиста** как минимум следующую информацию:
  - Тип примитива, в который входит вершина (соответственно, вместе с данной вершиной должны быть указаны все остальные, входящие в примитив)
  - Проекцию вершины на плоскость экрана, выраженную **в однородных координатах**
- Перед тем, как быть спроектированной на плоскость экрана, вершина может претерпевать различные пространственные преобразования. Этими преобразованиями достигается «размещение» объектов на сцене

- Все типовые задачи «размещения» объектов решаются при помощи аффинных преобразований вершин этих объектов:
  - Параллельного переноса
  - Масштабирования
  - Поворота
- Для этого удобно использовать аппарат матриц, так как он предоставляет единый механизм осуществления как аффинных преобразований, так и преобразований проекции
- Если «классический» («олдскульный») OpenGL *заставлял* использовать матрицы (имел функции работы с ними на уровне API), новые версии дают полную свободу выбора механизма (и не имеют никаких вспомогательных функций для этого)

## Стеки матриц

- Все преобразования делаются при помощи матриц размерности 4x4
- Матрицы по назначению делятся на четыре типа:
  - матрица проекции
  - матрицы преобразования
  - матрица цвета
  - матрица текстуры
- Для каждого типа матриц существует свой стек
- Активной в каждый конкретный момент является одна матрица каждого типа из вершины соответствующего стека
- Работа со всеми типами матриц осуществляется единообразно

- При использовании матриц все преобразования сводятся к умножению вектора однородных координат вершины на матрицу преобразования, в результате чего получается вектор «новых» координат данной вершины:

$$\begin{pmatrix} m_0 & m_4 & m_8 & m_{12} \\ m_1 & m_5 & m_9 & m_{13} \\ m_2 & m_6 & m_{10} & m_{14} \\ m_3 & m_7 & m_{11} & m_{15} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}$$

ось X новой системы координат

ось Y новой системы координат

ось Z новой системы координат

начало новой системы координат

Матрица переноса

Матрица переноса

Матрица поворота вокруг оси

$$\begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

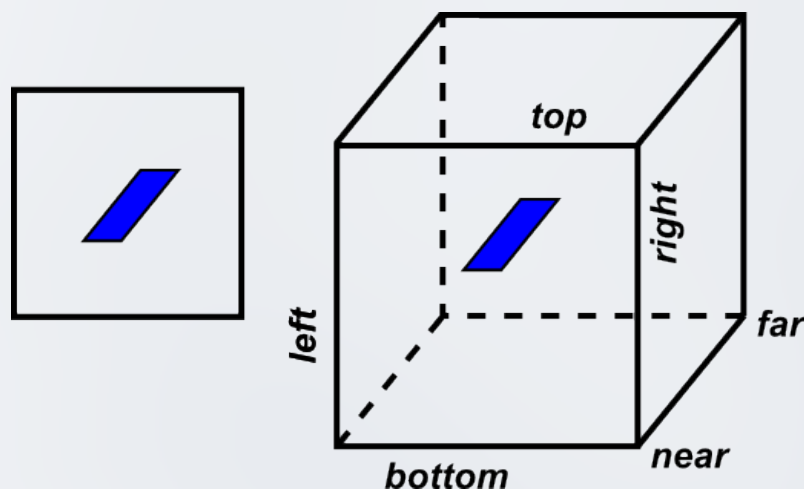
$$\begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} x^2(1-c)+c & xy(1-c)-zs & xz(1-c)+ys & 0 \\ yx(1-c)+zs & y^2(1-c)+c & yz(1-c)-xs & 0 \\ xz(1-c)-ys & yz(1-c)+xs & z^2(1-c)+c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

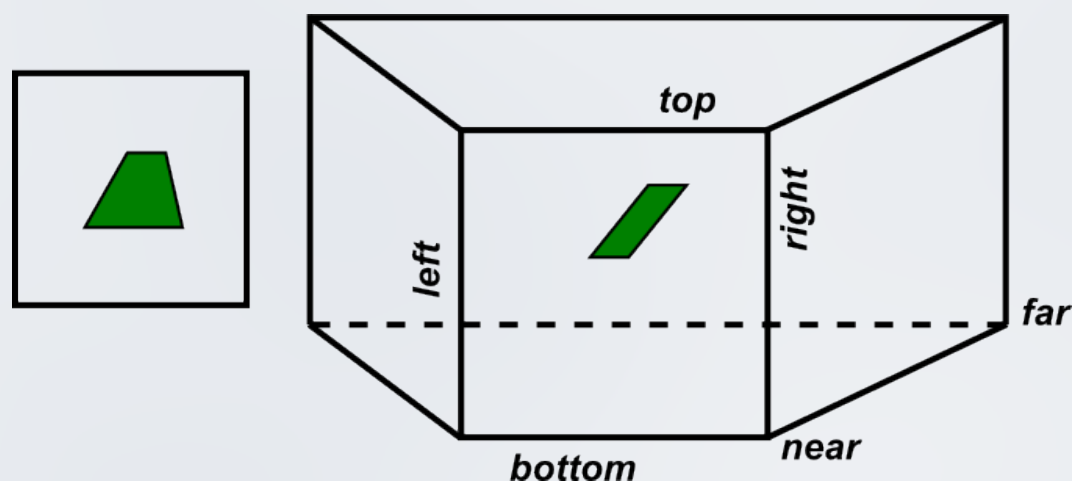
$$c = \cos \theta, \quad s = \sin \theta, \quad |(x, y, z)| = 1$$

→ Заданием параметров проекции определяется видимая область пространства

Параллельная проекция



Перспективная проекция

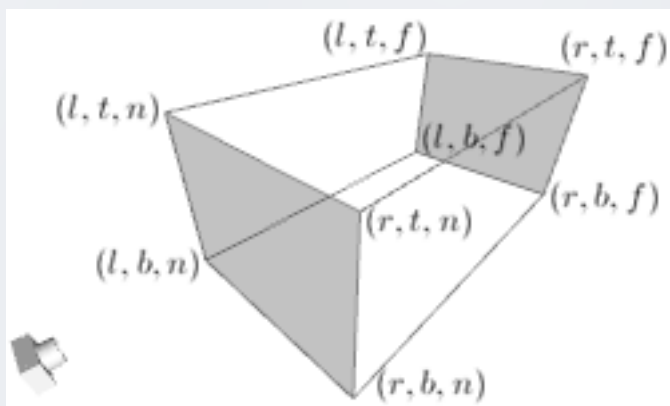


$$\begin{pmatrix} \frac{2}{\text{right} - \text{left}} & 0 & 0 & -\frac{\text{right} + \text{left}}{\text{right} - \text{left}} \\ 0 & \frac{2}{\text{top} - \text{bottom}} & 0 & -\frac{\text{top} + \text{bottom}}{\text{top} - \text{bottom}} \\ 0 & 0 & \frac{-2}{\text{far} - \text{near}} & -\frac{\text{far}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

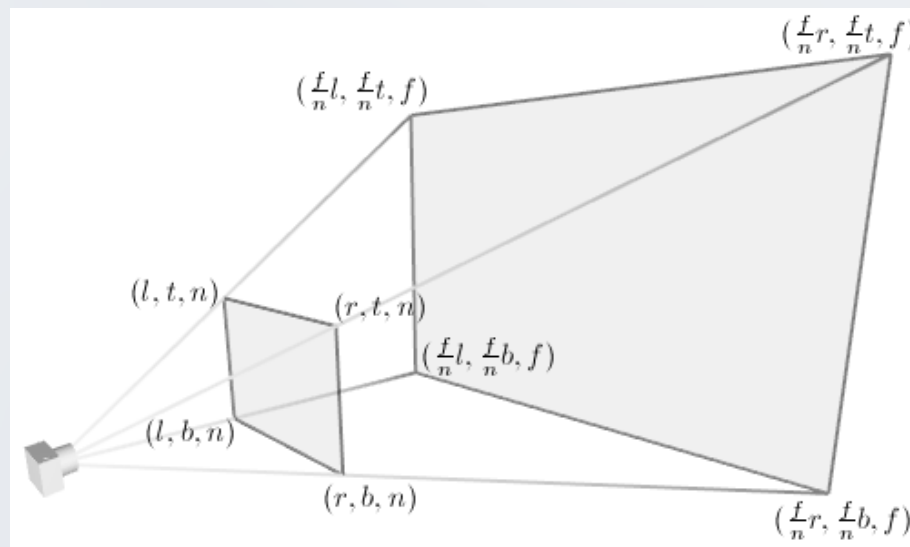
$$\begin{pmatrix} \frac{2 \text{ near}}{\text{right} - \text{left}} & 0 & \frac{\text{right} + \text{left}}{\text{right} - \text{left}} & 0 \\ 0 & \frac{2 \text{ near}}{\text{top} - \text{bottom}} & \frac{\text{top} + \text{bottom}}{\text{top} - \text{bottom}} & 0 \\ 0 & 0 & -\frac{\text{far} + \text{near}}{\text{far}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

- Заданием параметров проекции определяется видимая область пространства

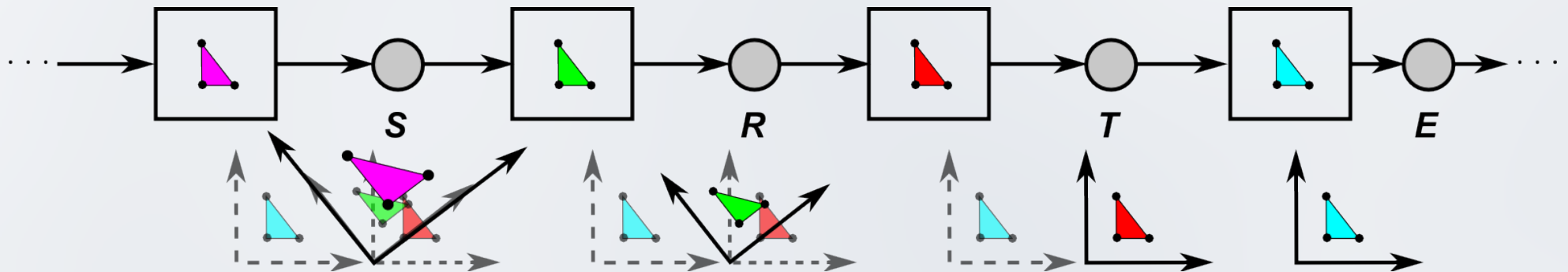
Параллельная проекция



Перспективная проекция



- Важным свойством матричных преобразований является их комбинированность:



- В связи с этим, в компьютерной графике имеется паттерн хранения и применения преобразований **Model-View-Projection**:

$$v' = (P \cdot V \cdot M) \cdot v$$

$v'$  – вектор координат, передаваемый системе для произведения растеризации

$v$  – вектор координат вершины

$P$  – матрица проекции

$V$  – матрица вида (преобразование камеры)

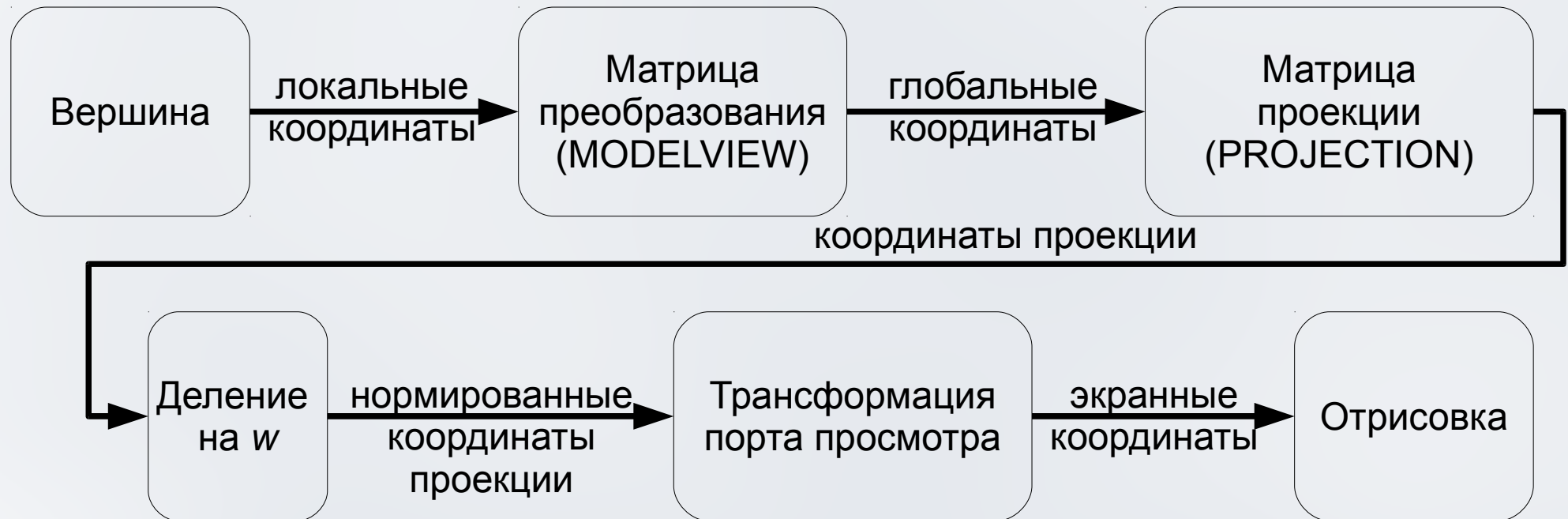
$M$  – матрица модели (преобразование размещения объекта на сцене)

$$M = M_{\text{родителя}} \cdot M_{\text{объекта}}$$



- Камера – это псевдообъект в трёхмерном пространстве, характеризующий положение наблюдателя
- Камера – лишь полезная метафора, на низком уровне она выражена матричным преобразованием, математически ничем не отличающимся от всех остальных
- Часто преобразование камеры является лишь аффинным
- В связи с этим, *иногда* преобразование камеры не хранят отдельно, а «смешивают» его с преобразованием размещения, получая матрицу, которую принято называть ModelView (в «классическом» OpenGL было именно так)

- В итоге, преобразование координат, осуществляемое в графическом приложении, имеет вид:



\* В **новых версиях** OpenGL, преобразования из **первого ряда** должен выполнять **программист**, а преобразования из **второго ряда** **система** выполняет автоматически