

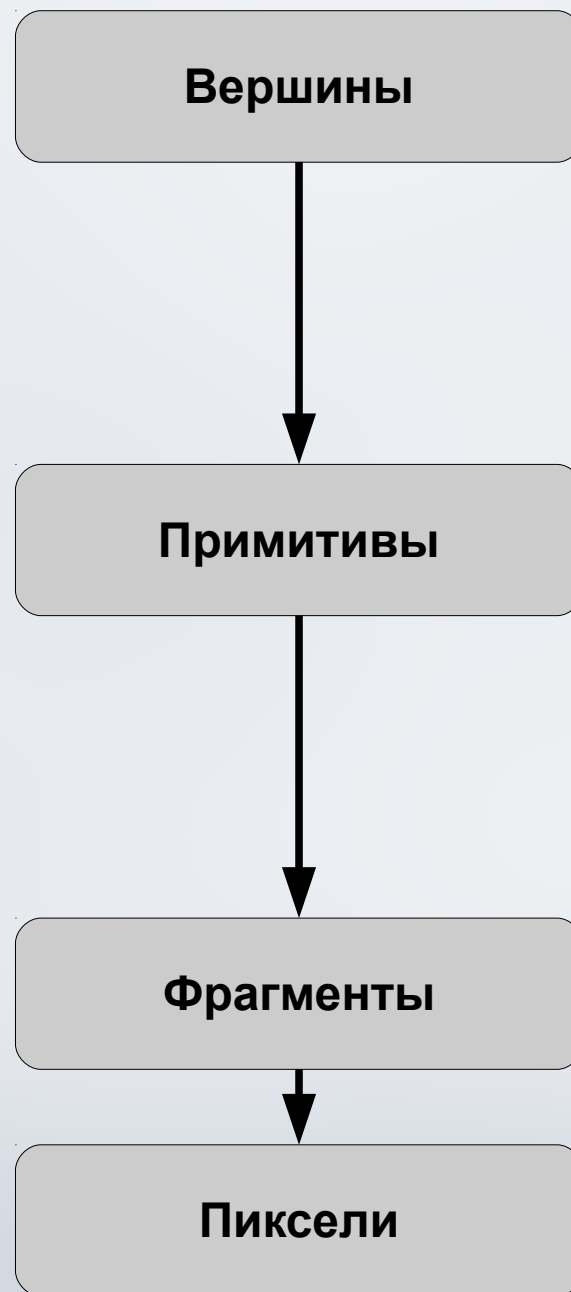
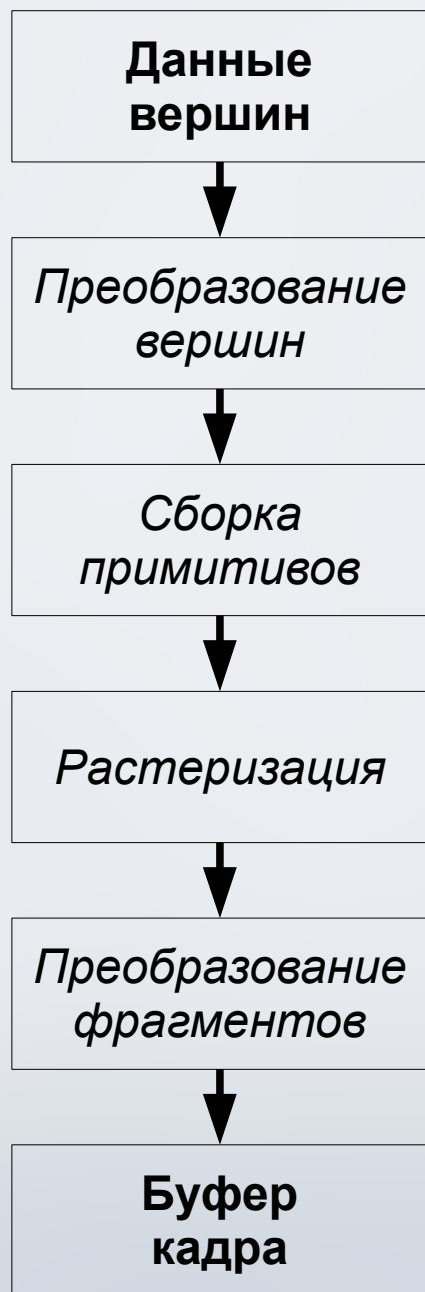
# Вычислительная геометрия и алгоритмы компьютерной графики

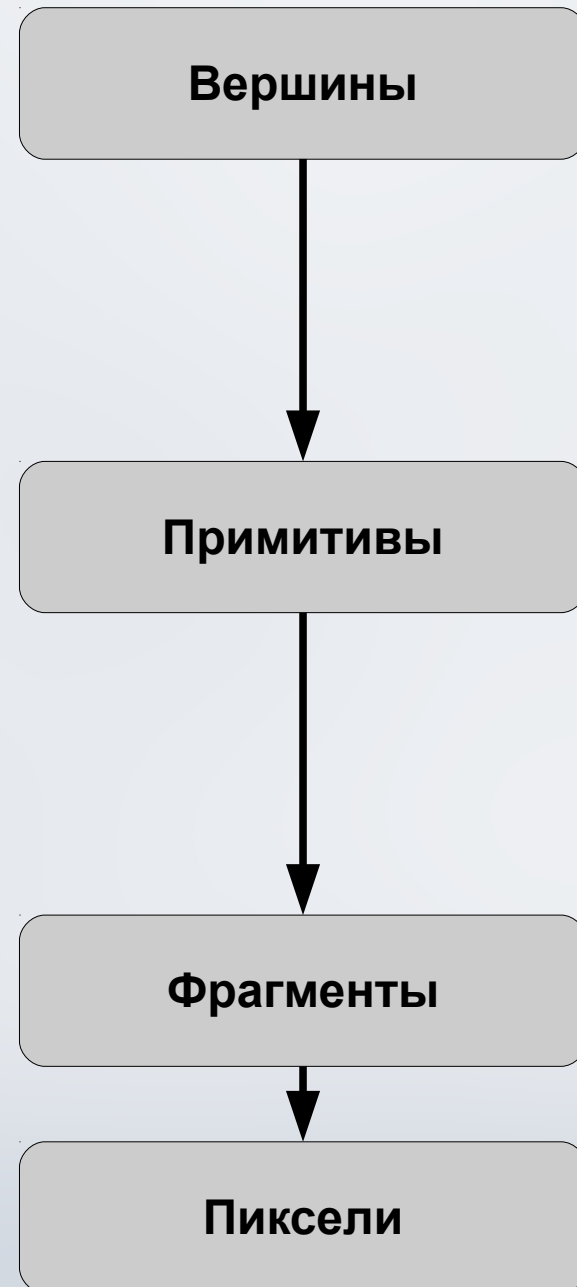
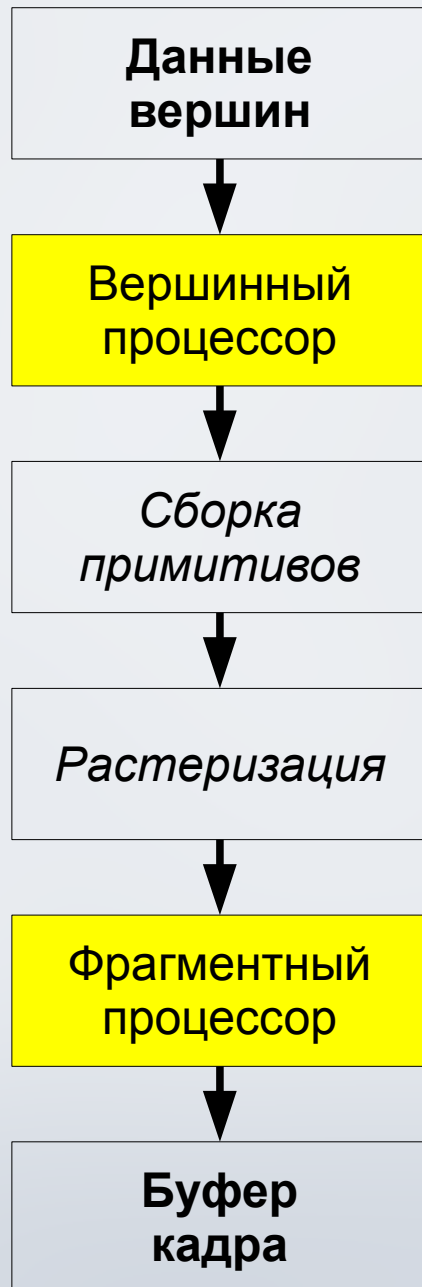
## *Лекция №3: Шейдеры*

к.ф.-м.н.  
Рябинин Константин Валентинович

e-mail: [kostya.ryabinin@gmail.com](mailto:kostya.ryabinin@gmail.com)

Пермь, 2016





**Шейдер** – это микропрограмма для одной из ступеней графического конвейера, используемая для определения окончательных параметров объекта или изображения

Свойства:

- Автономность
- Аппаратная поддержка
- Специализированный язык программирования
- Параллелизм по данным
- Узкая специализация

## Назначение шейдеров:

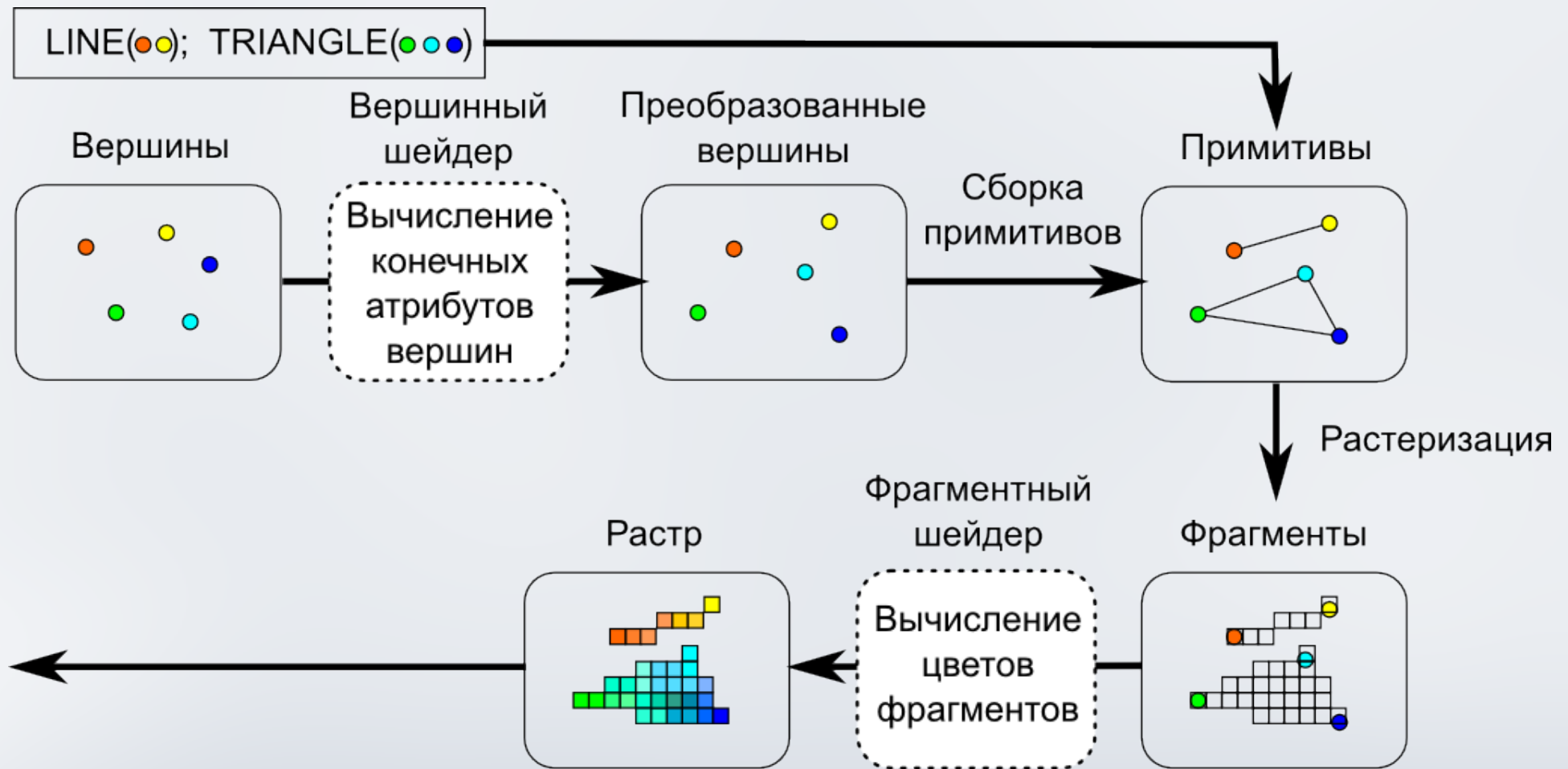
- Создание визуальных эффектов
  - Превращение графического конвейера из неуправляемого в управляемый
    - Значительное увеличение свободы управления результатом визуализации
- Унификация механизма создания визуальных эффектов любой сложности

## Достоинства шейдеров:

- Очень высокая эффективность
  - Свобода создания визуальных эффектов
  - Децентрализация кода
  - Межпроектное переиспользование
- В современной практике шейдеры составляют основу всех визуальных эффектов, без их применения не обходится ни одна мультимедийная система

- **Загрузка из файла (или из строковой константы)**
- **Компиляция («на лету», во время выполнения основной программы)**
- **Встраивание в конвейер («активация»)**
- **Множественное выполнение (для каждой обрабатываемой данной ступенью конвейера сущности за один рендеринг шейдер выполняется ровно один раз)**
- **Отсоединение от конвейера («деактивация»)**
- **Удаление из памяти**

- **Вершинный шейдер** – микропрограмма, обрабатывающая за раз **одну вершину объекта**  
→ Задача: определить положение и прочие характеристики вершины
- **Фрагментный шейдер** – микропрограмма, обрабатывающая за раз **один фрагмент изображения**  
→ Задача: определить цвет и прочие характеристики фрагмента
- . . .





**По сути шейдер – общее название для семейства специализированных микропрограмм**

**Для написания шейдеров используются специализированные языки программирования, характеризующиеся:**

- Процедурной парадигмой**
- Тьюринг-полнотой**
- Наличием специализированных типов данных и встроенных функций для работы с обрабатываемыми сущностями**
- Как правило, за основу шейдерных языков берётся синтаксис C**

## Примеры языков:

- GLSL (Graphics Library Shader Language)
  - Шейдерный язык от ARB
- HLSL (High-Level Shader Language)
  - Шейдерный язык от Microsoft
- DirectX ASM
  - Шейдерный ассемблер от Microsoft
- Cg (C for Graphics)
  - Шейдерный язык от nVidia для Microsoft
- RenderMan
  - Шейдерный язык от Pixar для художников
- Gelato
  - Шейдерный язык от nVidia для художников

**GLSL** – это высокоуровневый процедурный язык программирования для вершинного и фрагментного процессоров OpenGL

## Характеристика

- Программы на GLSL представляют собой абстракцию от аппаратного обеспечения
- Компилятор GLSL является частью библиотеки стандарта OpenGL и генерирует оптимизированный под конкретную видеокарту код
- GLSL основан на синтаксисе C и является чисто процедурным
- Начало выполнения программы – функция `void main()`
- Спецификация языка (от 03.08.2012):  
<http://www.opengl.org/registry/doc/GLSLangSpec.4.30.6.pdf>

- Шейдеры могут получать из основной программы произвольные наборы данных:
  - «глобальных» (одинаковых для всех обрабатываемых сущностей)
  - «локальных» (сцепленных с сущностью, как правило при помощи массивов, индексы в которых соответствуют номерам сущностей)
- Шейдеры более раннего этапа конвейера могут подготавливать и передавать параметры шейдерам более позднего этапа

**Вершинный шейдер** – это программа, обрабатывающая вершины и связанные с ними данные

Входные и выходные данные:

- **Переменные-атрибуты** (*attribute*) – передаются вершинному шейдеру от приложения для описания свойств каждой вершины
- **Однообразные переменные** (*uniform*) – используются для передачи данных как вершинному, так и фрагментному шейдеру. Не могут меняться чаще, чем один раз за полигон
- **Разнообразные переменные** (*varying*) – служат для передачи данных от вершинного к фрагментному шейдеру. Данные переменные могут быть различными для разных вершин, и для каждого фрагмента будет выполняться интерполяция

**Фрагментный шейдер** – это программа, обрабатывающая фрагменты и связанные с ними данные

**Входные и выходные данные:**

- **Разнообразные переменные** (*varying*) – приходят от вершинного шейдера; как встроенные, так и определенные разработчиком
- **Однообразные переменные** (*uniform*) – используются для передачи произвольных относительно редко меняющихся параметров

**Шейдерная программа** – это специализированная структура данных в OpenGL, предназначенная для хранения нескольких шейдеров разных типов для их одновременного использования

- В самом простом случае шейдерная программа используется для связки **ровно двух шейдеров** – одного вершинного и одного фрагментного, так как программируемый конвейер в каждый момент времени нуждается ровно в двух шейдерах
- Теоретически возможна сборка шейдера из нескольких частей, но на практике это почти не используется
- В каждый момент времени может быть активна только одна шейдерная программа
- Шейдерная программа – единственный способ использования шейдеров в OpenGL, поэтому **шейдеры всегда существуют парами** (каждому вершинному должен соответствовать его фрагментный)
- Порядок вызовов:  
`glCreateProgram → glAttachShader → glAttachShader → glLinkProgram → glUseProgram`

- Поверхности в OpenGL хранятся в виде массивов вершин (каждая вершина – набор атрибутов типа float)
- Тип связности вершин (способ объединения в примитивы) указывается уже при отрисовке, в качестве параметра функции рисования
- Массив вершин может храниться как в оперативной памяти, так и в видеопамяти. Второй способ предпочтительнее по скорости
- Для хранения вершин в видеопамяти используется специальная структура, называемая **вершинным буфером** (vertex buffer)
- Работа с вершинными буферами осуществляется при помощи API-функций OpenGL
- Часто кроме массива вершин используется ещё и **массив индексов**, что позволяет устранить дублирование вершин. Индексы так же сохраняются в отдельном буфере