МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

Компьютерная графика:
лабораторный практикум.
Лабораторная работа № 5
«Расширения OpenGL, программируемый графический конвейер. Шейдеры»

Студент гр. 5383 _____ Допира В. Е.

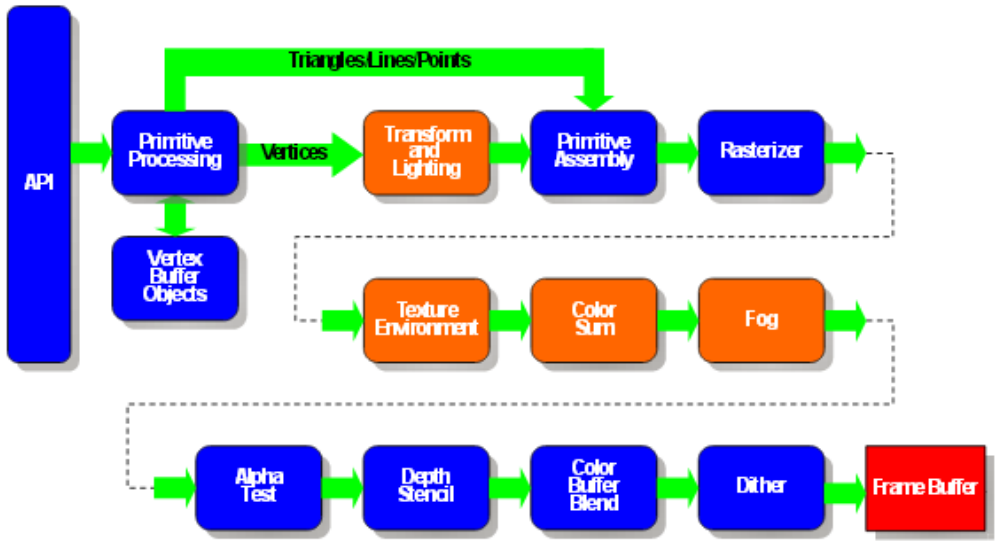
Преподаватель _____ Герасимова Т.В.

Санкт-Петербург
2018

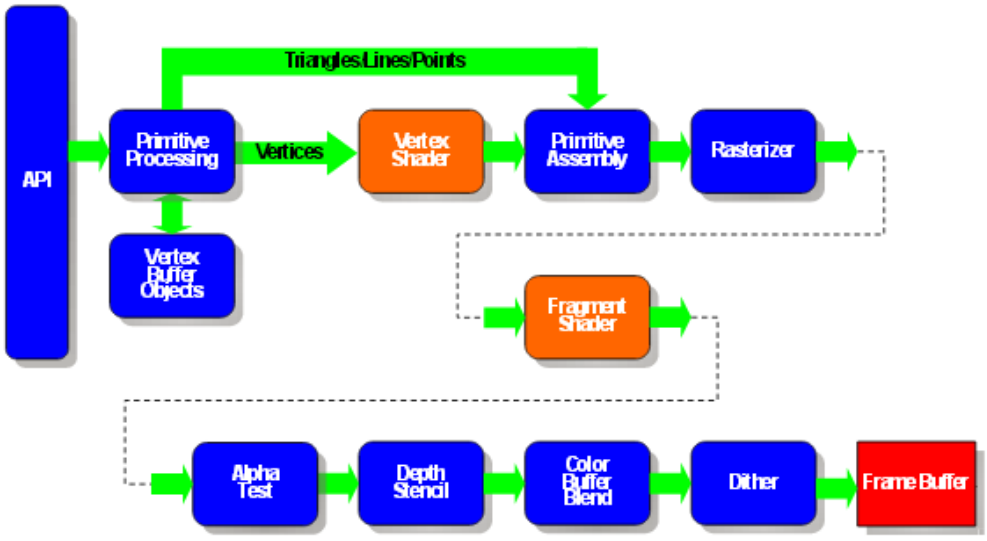
Лабораторная работа № 5

Общие сведения

Конвейер с фиксированной функциональностью



Программированный графический конвейер



Программируемый графический конвейер позволяет обойти фиксированную функциональность стандартного графического конвейера OpenGL.

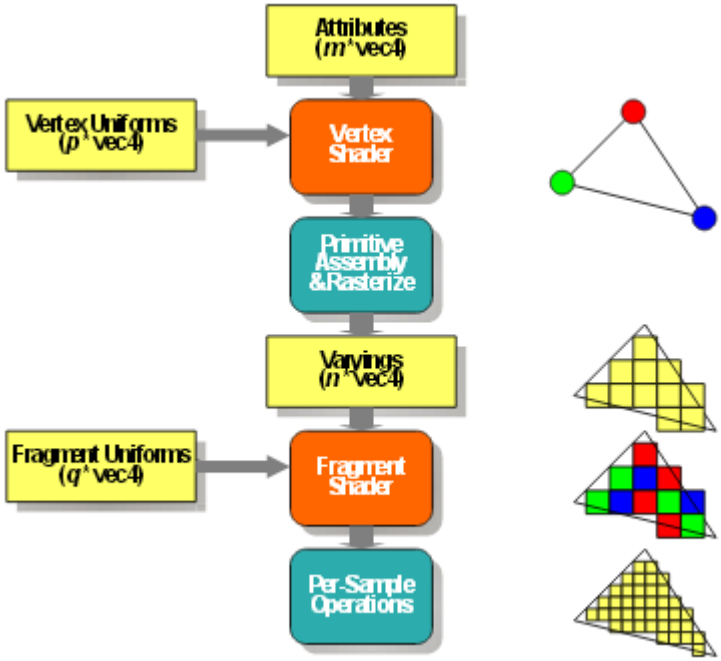
Для вершин – задать необычное преобразование вершин (обычное – это просто умножение координат вершин на модельную и видовую матрицу). Типичное применение - скелетная анимация, анимация волн.

Для геометрических примитивов, таких как треугольник, позволяет сформировать несколько иную геометрию, чем было, например, разбить треугольник на несколько более мелких.

Для фрагментов – позволяет определить цвет фрагмента (пикселя) в обход стандартных моделей освещения. Например, реализовать процедурные текстуры: дерева или мрамора.

Программа, используемая для расширения фиксированной функциональности OpenGL называется шейдер, соответственно различают 3 типа шейдеров вершинный, геометрический и фрагментный

Программируемая модель



Совместно с библиотекой OpenGL, могут быть использованы шейдеры, написанные на языке высокого и низкого уровня. Код шейдеров на языке низкого уровня сходен с кодом ассемблера, однако в действительности вы не кодируете на уровне ассемблера, поскольку каждый производитель аппаратного обеспечения предлагает уникальную структуру графического процессора с собственным представлением инструкций и наборов команд. Все эти процессоры вводят собственные пределы числа регистров констант и команд. Низкоуровневые расширения можно назвать наименьшим общим знаменателем функциональных возможностей, доступных у всех производителей.

Программирование графических процессоров на языке высокого уровня означает меньше кода, более читабельный вид, а, следовательно, более высокую производительность труда.

Язык программирования высокоуровневых расширений называются языком затенения OpenGL (OpenGL Shading Language –GLSL), иногда именуемым языком шедеров OpenGL (OpenGL Shader Language). Этот язык очень похож на язык С но имеет встроенные типы данных и функции полезные в шедерах вершин и фрагментов.

Шейдер является фрагментом шейдерной программы, которая заменяет собой часть графического конвейера видеокарты. Тип шейдера зависит от того, какая часть конвейера будет заменена. Каждый шейдер должен выполнить свою обязательную работу, т. е. записать какие-то данные и передать их дальше по графическому конвейеру.

Шейдерная программа – это небольшая программа, состоящая из шейдеров (вершинного и фрагментного, возможны и др.) и выполняющаяся на GPU (Graphics Processing Unit), т. е. на графическом процессоре видео-карты.

Существует пять мест в графическом конвейере, куда могут быть встроены шейдеры. Соответственно шейдеры делятся на типы:

– вершинный шейдер (vertex shader);

– геометрический шейдер (geometric shader);

– фрагментный шейдер (fragment shader);

– два тесселяционных шейдера (tesselation), отвечающие за два разных этапа тесселяции (они доступны в OpenGL 4.0 и выше).

Дополнительно существуют вычислительные (compute) шейдеры, которые выполняются независимо от графического конвейера.

Разные шаги графического конвейера накладывают разные ограничения на работу шейдеров. Поэтому у каждого типа шейдеров есть своя специфика.

Геометрический и тессеяляционные шейдеры не являются обязательными. Современный OpenGL требует наличия только вершинного и фрагментного

шейдера. Хотя существует сценарий, при котором фрагментный шейдер может отсутствовать

Окружение вершинного шейдера

Вершинные шейдеры – это программы, которые производят математические операции с вершинами, иначе говоря, они предоставляют возможность выполнять программируемые алгоритмы по изменению параметров вершин. Каждая вершина определяется несколькими переменными, например, положение вершины в 3D-пространстве определяется координатами: *x*, *y* и *z*.

Вершины также могут быть описаны характеристиками цвета, текстурными координатами и т. п. Вершинные шейдеры, в зависимости от алгоритмов, изменяют эти данные в процессе своей работы, например, вычисляя и записывая новые координаты и/или цвет. Входные данные вершинного шейдера – данные об одной вершине геометрической модели, которая в данный момент обрабатывается. Обычно это координаты в пространстве, нормаль, компоненты цвета и текстурные координаты. Результирующие данные выполняемой программы служат входными для дальнейшей части конвейера, растеризатор делает линейную интерполяцию входных данных для поверхности треугольника и для каждого пикселя исполняет соответствующий пиксельный шейдер.

Для управления входными и выходными данными вершинного шейдера используются *квалификаторы типов*, определенные как часть языка шейдеров OpenGL:

– 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 процессору. Не могут меняться чаще, чем один раз за полигон – относительно постоянные значения;

– разнообразные переменные (varying) – служат для передачи данных от вершинного к фрагментному процессору. Данные переменные могут быть

различными для разных вершин, и для каждого фрагмента будет выполняться интерполяция.

Окружение фрагментного шейдера

Фрагментный шейдер не может выполнять операции, требующие знаний о нескольких фрагментах, изменить координаты (пара *x* и *y*) фрагмента.

Фрагментный шейдер не заменяет стандартные операции, выполняемые в конце обработки пикселей, но заменяет часть графического конвейера (ГК), обрабатывающего каждый полученный на предыдущих стадиях ГК фрагмент (не пиксель) (рис. 1.4). Обработка может включать такие стадии, как получение данных из текстуры, просчет освещения, просчет смешивания.

Обязательной работой для фрагментного шейдера является запись цвета фрагмента во встроенную переменную gl_FragColor, или его отбрасывание специальной командой discard. В случае отбрасывания фрагмента, никакие расчеты дальше с ним производиться не будут, и фрагмент уже не попадет в буфер кадра.

Если задачей вершинного шейдера являлось вычисление позиции вершины, а также других выходных параметров вершины на основе uniform- и attribute-переменных, то в задачи фрагментного шейдера будет входить вычисление цвета фрагмента и его глубины на основе встроенных и определяемых пользователем varying- и uniform-переменных.

Фрагментный шейдер обрабатывает входной поток данных и производит выходной поток данных – пикселей изображения.

Фрагментный шейдер получает следующие данные:

– разнообразные переменные (varying) от вершинного шейдера – как встроенные, так и определенные разработчиком;

– однообразные переменные (uniform) – для передачи произвольных относительно редко меняющихся параметров.

Фрагментный шейдер не может выполнять операции, требующие знаний о нескольких фрагментах, изменить координаты (пара *x* и *y*) фрагмента.

Фрагментный шейдер не заменяет стандартные операции, выполняемые в конце обработки пикселей, но заменяет часть графического конвейера (ГК), обрабатывающего каждый полученный на предыдущих стадиях ГК фрагмент (не пиксель) (рис. 1.4). Обработка может включать такие стадии, как получение данных из текстуры, просчет освещения, просчет смешивания.

Обязательной работой для фрагментного шейдера является запись цвета фрагмента во встроенную переменную gl_FragColor, или его отбрасывание специальной командой discard. В случае отбрасывания фрагмента, никакие расчеты дальше с ним производиться не будут, и фрагмент уже не попадет в буфер кадра.

Если задачей вершинного шейдера являлось вычисление позиции вершины, а также других выходных параметров вершины на основе uniform- и attribute-переменных, то в задачи фрагментного шейдера будет входить вычисление цвета фрагмента и его глубины на основе встроенных и определяемых пользователем varying- и uniform-переменных.

Фрагментный шейдер обрабатывает входной поток данных и производит выходной поток данных – пикселей изображения.

Фрагментный шейдер получает следующие данные:

– разнообразные переменные (varying) от вершинного шейдера – как встроенные, так и определенные разработчиком;

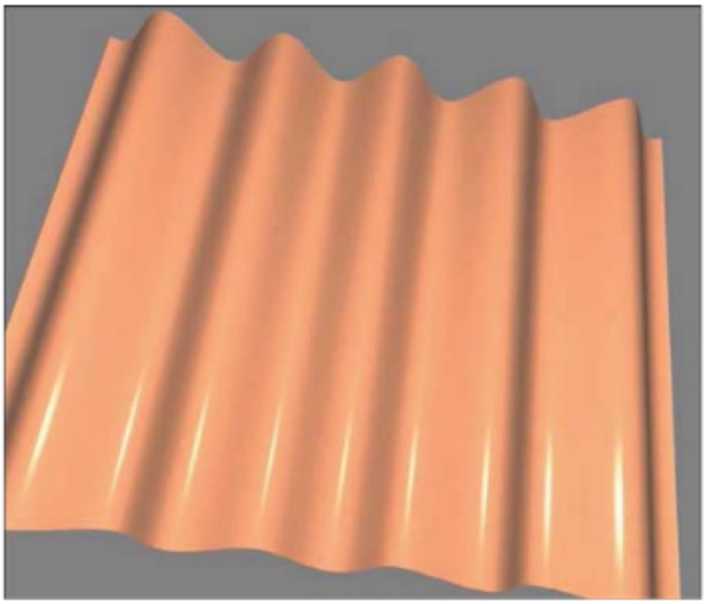
– однообразные переменные (uniform) – для передачи произвольных относительно редко меняющихся параметров.

Анимация поверхности смещением вершин

Самый очевидный способ использования шейдеров в воспроизведении анимационных эффектов – организовать преобразование вершин внутри вершинного шейдера, опираясь на некоторую функцию времени. Основное приложение определяет статическую геометрию, а вершинный шейдер изменяет ее, используя значение екущего времени (передается в uniform-переменной). В результате вычисления, касающиеся координат вершин, перемещаются из

основной программы в шейдер и используются возможности параллелизма, предоставляемые графическим драйвером.

В примере создан эффект волнения поверхности путем преобразования вершин прямоугольника по синусоидальному закону. Программа передает вниз по конвейеру множество треугольников, составляющих прямоугольную поверхность, лежащую в плоскости X–Z, а вершинный шейдер преобразовывает координату Y каждой вершины, опираясь на функцию синуса с временной зависимостью, и вычислять вектор нормали для преобразованной вершины. На рис. показан результат, который предполагается получить. (Волны движутся по поверхности слева направо.)



Эффект волнения поверхности

Задание

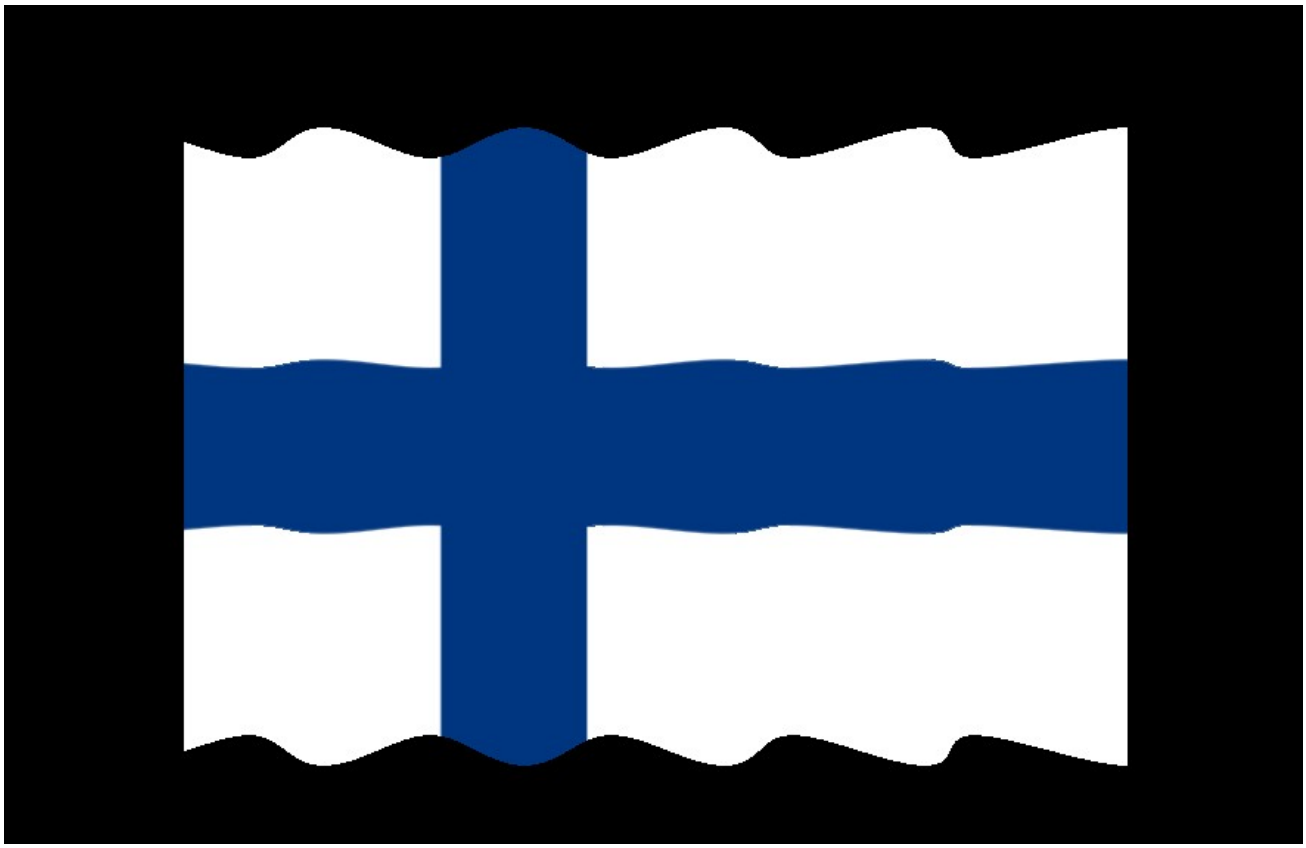
Разработать визуальный эффект по заданию, реализованный средствами языка шейдеров GLSL. Необходимо развить предыдущую лабораторную работу, превратив кривую в поверхность на сцене и добавив в программу дополнительный визуальный эффект, реализованный средствами языка шейдеров.

Вариант Эффекта:

Анимация. Расстояние от вершины до заданной точки меняется по синусоиде.

Тестирование

Результаты тестирования представлены на снимках экрана.



Вывод

В результате выполнения лабораторной работы разработана программа, которая реализует анимацию средствами языка шейдеров GLSL.