

design → ARB-расширение.  
ARB 2003: в OpenGL 2.0.

Shader OpenGL (Lima) GLSL  
 HLLSL high level Shader language (MS)  
 это Cg (Microsoft) C for Graphics

1) обработка вершин → преобразование, масштабирование, сдвиг.  
 2) обработка фрагментов → расгруппировка (вершин. → пиксель)  
 применение текстур  
 добавление  
 цвета пикселя

3) пиксельный (группированный)

Идея шейдеров OpenGL - высокопроизводительный прогрессивный язык

### Функции:

- ✓ 1) переменные матричные - модель,amera и т.д.
- ✓ 2) функции сглаживания → обласи, маскировка
- ✓ 3) программные вычисления → цвета, боя, зум, обмык /Basic Raster/
- ✓ 4) шагающие не геометрические → шаги, цвета, маскировка
- ✓ (5) текстурное письмо → предупреждение текстур  
текстурное письмо → предупреждение нормалей и т.д.
- ✓ (6) программные фрагменты для текстур
- ✓ 7) автоматическое управление текстурами
- ✓ 8) функции управления текстурами
- ✓ 9) многоугольник анимация (анимированное пересечение)

# Learn OpenGL Jerry de Vries

## 4.11. Organization

OpenGL - API, непрограммабельный API для управления графикой.

Immediate mode или fixed function pipeline.

Core-profile mode - включает все новые функции языка программирования V3.2  
↓  
↳ extensions 3.3 (включая 4.5.)

некоторые могут работать только в fixed function.

```
if(GL_ARB_extension-name) { более быстрое } else { медленное выражение }
```

Объекты OpenGL - недопонимание, что представляют на GO состояния OpenGL

координаты - коэффициенты - геометрические объекты - различные языки программирования.

координаты

GLFW, GLFW + CMake

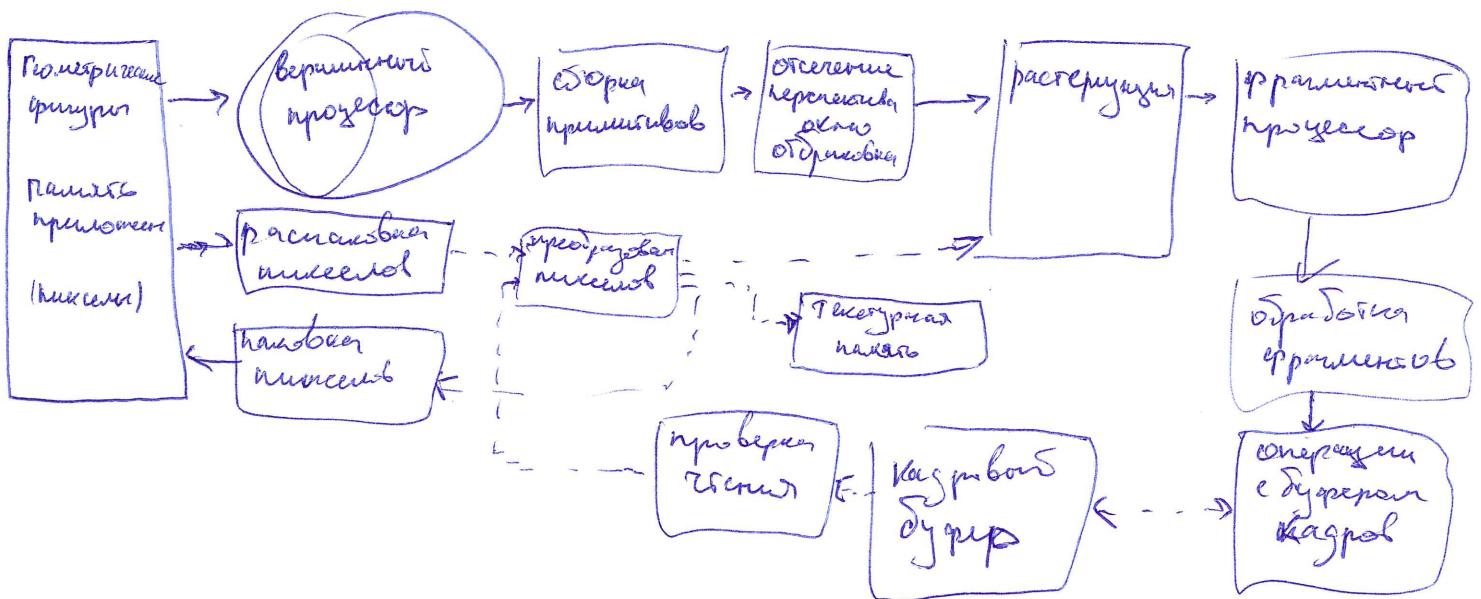
на C

или определение макроса в языке. → Extension Wrangler.

или `#include <GLFW3.h>` #define GLFW\_STATIC

или `-I C:\Program Files\GLFW\include -L C:\Program Files\GLFW\lib`

`-lglfw -lGL -lGLU -lGLESv2 -lX11 -lpthread -lXrandr -lXi`



## Вершиновый процессор →

Операции с геометрической магнитной Вершиной

- 1) преобразование вершин; → транс-прогр.м.
- 2) преобразование нормали, нормализация;
- 3) генерирование текстурных координат;
- 4) преобразование текстурных координат;
- 5) масировка объектов;
- 6) наполнение цвета материала.

## Необходимые

Вычисление перспективы  
создание градиентного освещения  
Сборка примитивов  
отсечение кромок и ненужных  
областией задних поверхностей  
ввод звукодорожного освещения  
плоски многоугольников  
выделение многоугольников  
последний или начало залевания  
изменение цвета материала

## Процессный подход

онер. шаги проектирования:

- 1) определение назначения и профессионального проектирования
  - 2) сборки и скелеты
  - 3) наполнение скелетов
  - 4) создание чертежей деталей
  - 5) наполнение чертежей.
- 

## Основное понятийное

аппаратная поддержка / поддержка от проектировщика  
и ее задачи и цели

(установление производственных (изделия на земле)  
нормативов и т.д.)

Нормативы и требования?

распределение рабочих функций.

## Модель проектирования

# Менюры

## Ускорение

Ког-на лучше использовать  
для закрепленных поверхностей.

методы np - ик

Аппар

Хард  $\leftrightarrow$  Аппарес мон  
Библиотечки API  
Драйверы

Аппарес мон  $\leftrightarrow$  программ

(запросы)  
ускорители

(CPU)

VPU  
 $\downarrow$   
visual

Open GL - <sup>стартует</sup> кратчайшее время - програм интегрирован (API)  
ISS22.

Rendering - визуализация.

Будет крафт  $\rightarrow$  лучше окно

- 1) кон-бо буфера  $\rightarrow$  <sup>без буферов / буферов</sup> (без буферов)  $\rightarrow$  быстрее отображения
- 2) фрейм буферы  $\rightarrow$  удаление плавающих кадров для более плавного отображения
- 3) буферы менеджер  $\rightarrow$  <sup>использование</sup> менеджеров операций менеджеров
- 4) буферы памяти  $\rightarrow$  <sup>использование</sup> памяти сущего увелич. кас. изображения (если память не имеет)
- 5) антиалиасинг буфер.  $\rightarrow$  с более высоким разрешением

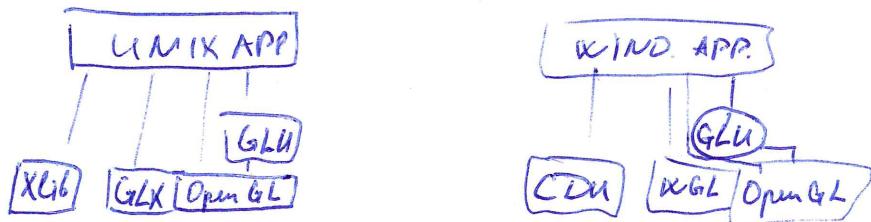
Компилятор - представление единого структура для работы с различными  
платформами позволяющее упрощение обработки данных в различных  
форматах.

с 6 версии

2-ая версия → поддержка shaders (2001)

3-я версия → (2010) → обновленные проблемы, блокир. рекурсии

### API схемы



### SPIR-V (Standard Portable Intermediate Representation)

нагборт коммандные шейдеры или языки пиродотки  
(пиродотки на языке, не языко GLSL (OpenGL Shading Language).

DirectX → HLSL → High Level Shader Language (MS)  
Cg (Nvidia) → C for Graphics

Open GL

GLee

GLFW

lib glew  
GLFW

головные  
заголовки  
дев

- `eglfix`
- `GL`
- `GLFW`

GLSL → GL Shading Language

Video Card

Support OpenGL v. 4.x

Менеджер - небольшая программа, возвращающая на экран. используя  
(команды на C языке) GLSL

(Преобразует строку кода с векторами и матрицами)

Переменные вершинного менеджера - свободны для использования

Кол-во вершин определяет возможное количество одновременно

GL\_MAX\_VERTEX\_ATTRIBS = 32768; /min of 16/

Формат координат:

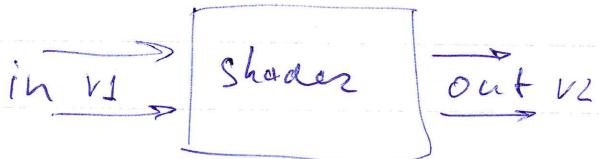
1) Vector → 1-4 float (int, double)

float →	d	vec	1. x
double →	i		2. y
unsigned int			3. z
			4. w

vec 4. different vecs some vec.xyyy;  
 = vec1.xxxx + vec2.yyyy;  
 = vec2(1.0f, 2.0f);

Swizzling → можно (использовать) координаты зеркально

сдвиги



вершины → вершинный  
менеджер

программный → vec4 → есть в формате RGBA

В современных движках  
не используется никаких  
аналогов вершинного менеджера

Логика менеджера

#version version-number

in type variable1;

out type variable2;

uniform type variable3;

void main()

{ ... = variable1;

// Оператор

variable2 = variable1;

}

Порядок  
исчисления  
координат  
текущий  
шаг  
(исходный)  
формат

расширение  
базис  
текущий  
шаг  
текущий  
шаг  
номер  
шага

Вершинный  
менеджер

Геометрический  
менеджер

Органический  
менеджер

Бернирование  
шаги  
координат  
координат

расширение

Удаление  
координат  
координат

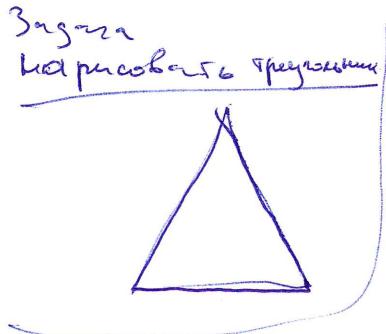
Удаление  
координат  
координат

Удаление  
координат  
координат

Удаление  
координат  
координат

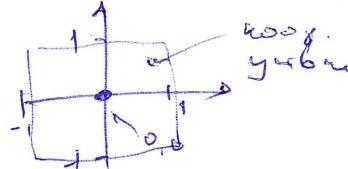
но не расширением  
Удаление координат

Удаление  
координат  
координат



Монтирующие координаты устройства  
Normalized Device Coordinates (NDC)

y coordinate ->  
gl Viewport



1) Определить координаты в ОСК

2) переместить данные на GPU (нужно, но отредактировано)  
gl Buffer Data

VBO → Vertex Buffer Objects

gl Buffer Data → данные

GL\_STATIC\_DRAW → не изменяется или изменяется редко

GL\_DYNAMIC\_DRAW

GL\_STREAM\_DRAW → изменяется при каждом отрисовке.

Вершинный шейдер

version : 330 core → содержит C.S.B. — ...  
layout (location = 0) ← указывает расположение

gl\_Position → позиция, куда выводится

Установка теней : теневая, нормальную карту  
использовать в gl Create Shader (GL\_VERTEX\_SHADER)  
gl Shader Source ← написать  
gl Compile Shader ← скомпилировать

Менеджер программы → содержит шейдеры / компоненты  
шейдер → менеджер → менеджер  
gl Link Shader → бинарные компоненты  
= gl Create Program // gl Attach Shader  
gl Link Program

## Создание вершинных атрибутов

glVertexAttribPointer ( )

0, → layout (location n=0)

3, → vec3 → пять одинаковых байтами

GL\_FLOAT, → float

GL\_FALSE, → форматы ячеек

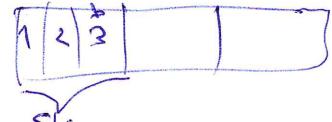
Step → 3 \* sizeof(GLfloat), → шаг между ячейками

Shift → (GLvoid\*) 0 ); → начальные значения

glEnableVertexAttribArray(0);

Shift

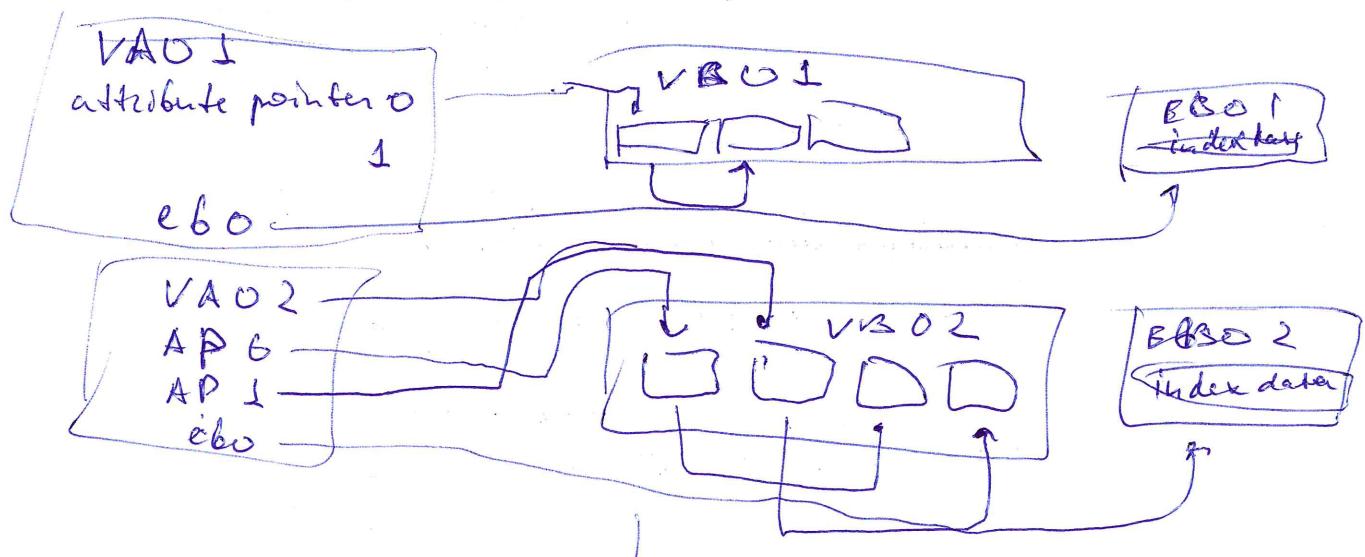
vec3



Step

## VAO Vertex Array Object Оbject вершиновых массивов

Хранение буферов вершинных атрибутов



EBO → Использование с индексами  
или же наборов из 60 вершин.

## Element Buffer Object