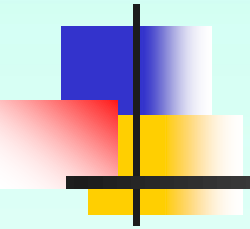
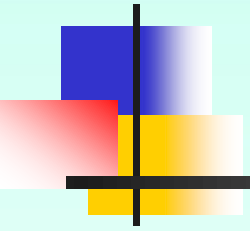


COMPUTER GRAPHICS

ЗАСОБИ ПРОГРАМУВАННЯ КОМП'ЮТЕРНОЇ ГРАФІКИ

Лек. 12 2021 ІПЗ-19



COMPUTER GRAPHICS

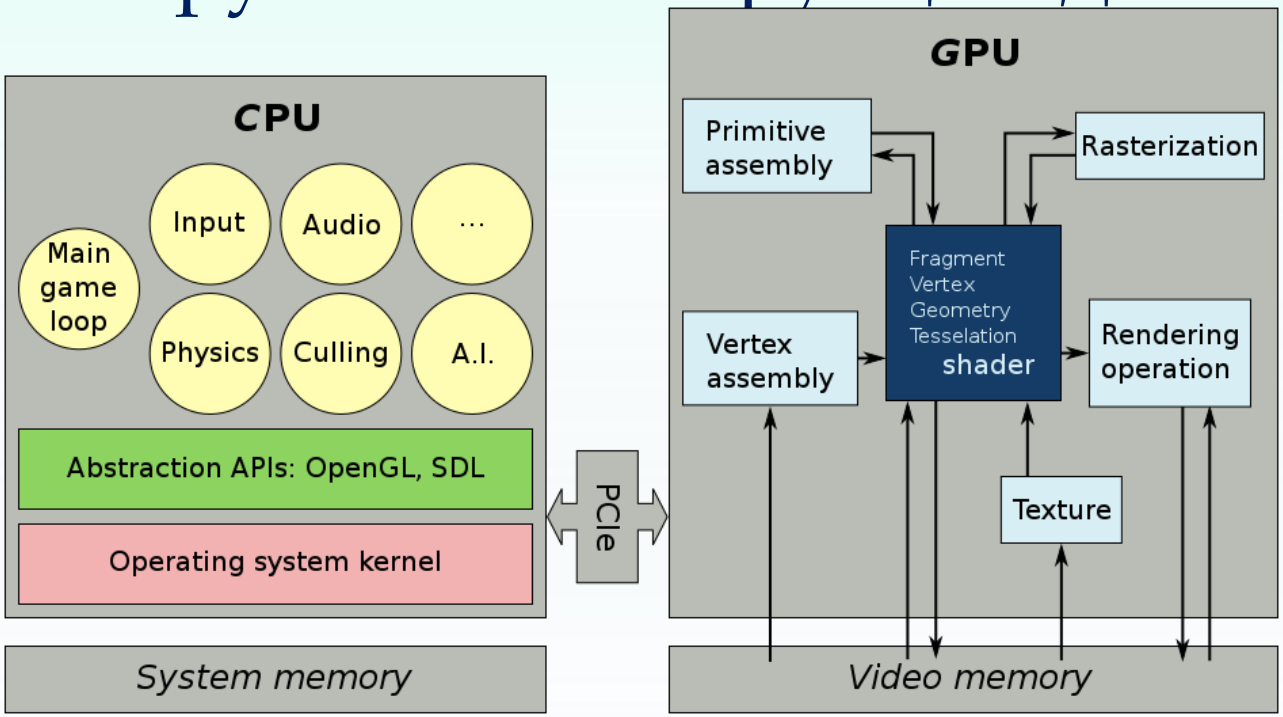
OPEN_GL (part 3)

Open GL. ШЕЙДЕРЫ

- Шейдеры. Типы шейдеров
- GLSL. Основные сведения.
- GLSL. Передача данных в шейдеры
- Примеры

Open GL. ШЕЙДЕРЫ

Ше́йдер (shader — «затеняющая» программа) — программа предназначенная для исполнения GPU. Шейдеры пишутся на одном из специализированных языков и компилируются в инструкции для GPU.

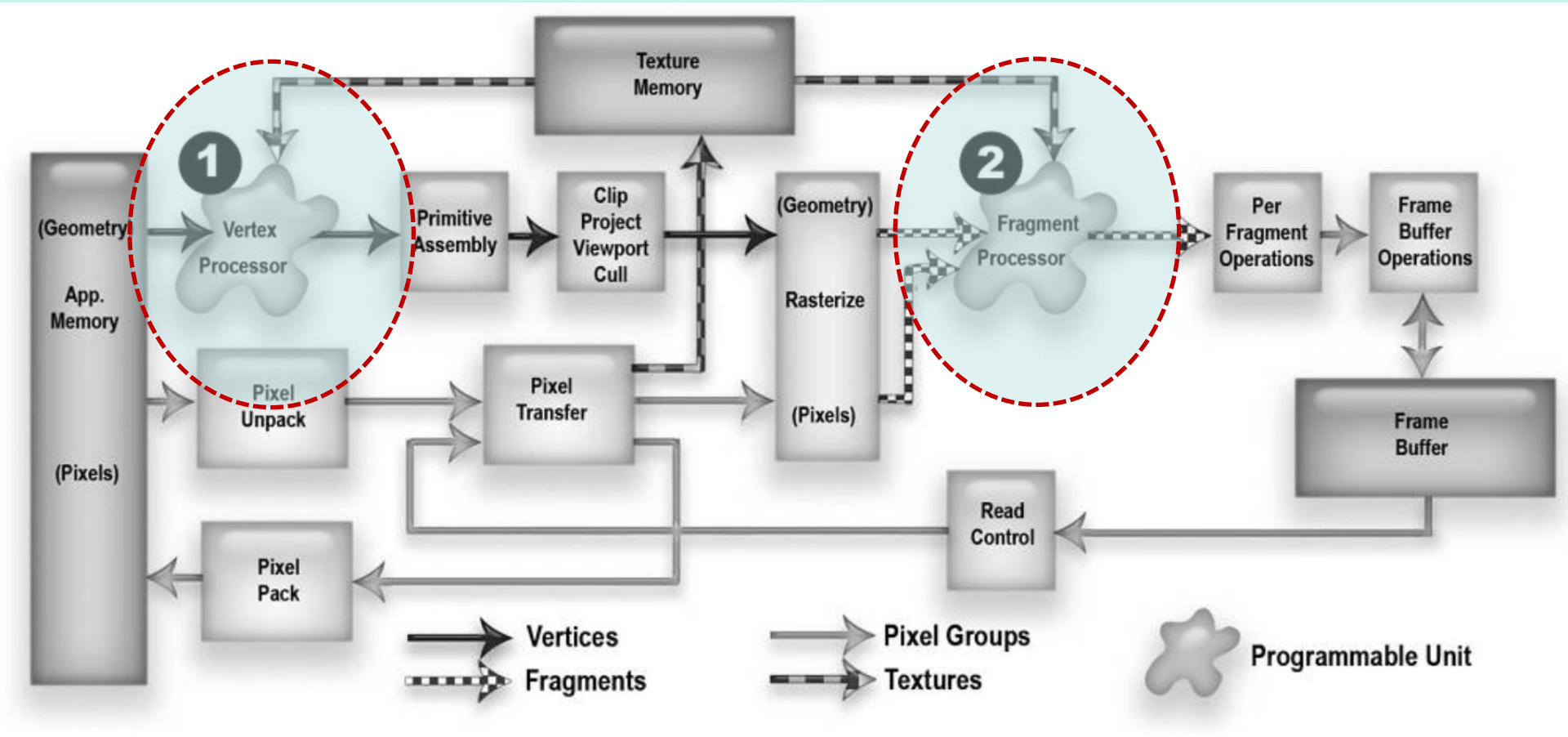


Open GL. ШЕЙДЕРЫ

Языки шейдеров:

- **OpenGL Shading Language (GLSL)**
- **High Level Shading Language (HLSL)-** DirectX 9.0 (Microsoft)
- **C for graphics (Cg) - NVIDIA**

Open GL. ШЕЙДЕРЫ



https://www.khronos.org/opengl/wiki/Shader

Open GL. ШЕЙДЕРЫ

Типы шейдеров:

- **вершинный** (vertex)
  GL_VERTEX_SHADER .
- **фрагментный** (fragment)
  GL_FRAGMENT_SHADER .
- **геометрический (**geometry).
- **2 тесселяционных** шейдера (tessellation control, tessellation evaluate), отвечающие за 2 разных этапа тесселяции.
- **вычислительный.**

https://www.khronos.org/opengl/wiki/Shader

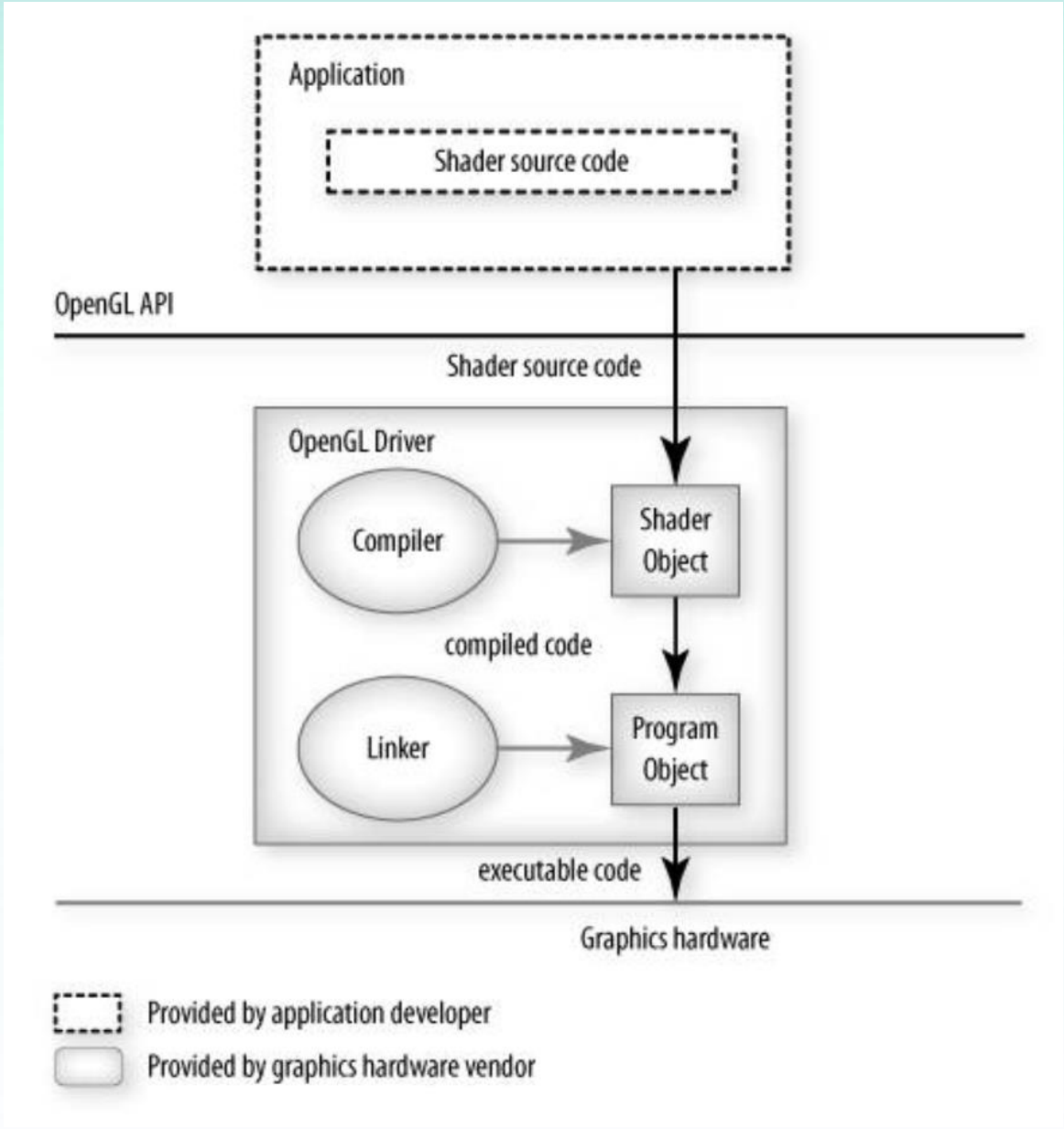
Open GL. ШЕЙДЕРЫ
Типичный шейдер:

```
in type in_variable_name;
in type in_variable_name;
out type out_variable_name;
uniform type uniform_name;
void main()
{
        // Обрабатываем данные ввода и выполняем обработку
        // ...
        // Передаем обработанные данные  в выходную переменную
        out_variable_name =  …..;
}
```

https://www.khronos.org/opengl/wiki/Shader

Open GL. ШЕЙДЕРЫ

Встраивание шейдеров в программу:



Open GL. ШЕЙДЕРЫ

- **Создание шейдерного объекта:**

```
Gluint vs =
glCreateShader(GL_VERTEX_SHADER);

Gluint fs =
glCreateShader(GL_FRAGMENT_SHADER);
```

https://www.khronos.org/opengl/wiki/Shader

Open GL. ШЕЙДЕРЫ

- **Копирование текста GLSL программы в шейдер:**

Const char* *vsText* = " ..... ";
glShaderSource (*vs*, 1, &*vsText*, NULL);

//аналогично для FS

- **Компиляция:**

glCompileShader(*vs*);

//аналогично для FS

- **Проверка на ошибки компиляции**

https://www.khronos.org/opengl/wiki/Shader

Open GL. ШЕЙДЕРЫ

Сборка шейдерной программы:

- **Создание программного объекта:**

Gluint *shadprog* = glCreateProgram();

- **Подсоединение шейдеров к шейдерной программе:**

glAttachShader (*shadprog, vs*);

glAttachShader (*shadprog, fs*);

- **Линкование:**

glLinkProgram( );

- **Проверка на ошибки линкования**

Open GL. ШЕЙДЕРЫ

//Перед биндингом буферов

- **Активация шейдерной программы:**

glUseProgram(*shadprog*);

/****UNIFORM переменные**********/**

glGetUniformLocation() ; glGetUniform()

/****ADDITIONAL **********/**

glDetachShader ()

glDeleteShader ()

glDeleteProgram ()

https://www.khronos.org/opengl/wiki/Shader

GLSL. Типы данных

В GLSL доступны следующие **простые** типы данных:

- **Float – число с пл.з. одинарной точности**
- **Int - целое число с ф.з.**
- **Bool - логическая переменная**

Эти типы данных - точно такие же, как в C.

GLSL. Типы данных

Доступны **векторы** для перечисленных выше типов данных с двумя, тремя или четырьмя компонентами.
Они объявляются как:

- **vec{2,3,4} - вектор из 2/3/4 float**
- **ivec{2,3,4} - вектор из 2/3/4 int**
- **bvec{2,3,4} - вектор из 2/3/4 bool**

GLSL. Типы данных

Также доступны квадратные **матрицы 2x2, 3x3** и **4x4**, т.к. они часто используются в графике.

- **mat2**
- **mat3**
- **mat4**

Массивы в шейдерах объявляются так же, как в C, однако они не могут быть инициализированы при объявлении.

GLSL. Типы данных

Доступен ряд специальных типов данных для работы с текстурами. Они называются "samplers" и нуждаются в доступе к значениям текстур, так же известным как "texels".

Типы данных:

- **sampler1D - для 1D-текстур**
- **sampler2D - для 2D-текстур**
- **sampler3D - для 3D-текстур**
- **samplerCube - для текстур кубических карт**
- **sampler1DShadow - для карт теней**
- **sampler2DShadow - для карт теней**

GLSL. Типы данных

Объявления:

- **float *a,b*; // Две переменные с пл.зпт.**
- **int *c*=2; // Объявление и инициализация**
- **bool *d* = true; // d := true**
- **float *b* = 2; // *Неправильно*, *автоматического указания типа данных НЕТ***
- **float e = (float)2; // *Неправильно*, *для указания типа вызывается конструктор***
- **int *a*=2; float *c* = float(*a*); // Правильно, c == 2.0**
- **vec3 *f*; // Объявление *f* как vec3**
- **vec3 *g* = vec3(1.0, 2.0, 3.0); // Объявление и инициализация вектора *g***

GLSL. Типы данных

Инициализация (вектора):

- **vec2 *a* = vec2(1.0, 2.0);**
- **vec2 *b* = vec2(3.0, 4.0);**
- **vec4 *c* = vec4(*a*,*b*); // *c* = vec4(1.0, 2.0, 3.0, 4.0)**
- **vec2 *g* = vec2(1.0, 2.0);**
- **float *h* = 3.0;**
- **vec3 *j* = vec3(*g*,*h*);**

GLSL. Типы данных

Инициализация (матрицы):

- **vec2 a = vec2(1.0, 2.0);**
- **mat4 m = mat4(1.0) // инициализация диагонали матрицы с 1.0**
- **mat2 n = mat2(a,b);**
- **mat2 k = mat2(1.0,0.0,1.0,0.0); // указаны все элементы**

GLSL. Типы данных

Структуры: Инициализация

```
struct dirlight {       // объявление типа
        vec3 direction;
        vec3 color;
    };
```

- dirlight d1;
- dirlight
  d2 = dirlight(vec3(1.0,1.0,0.0), vec3(0.8,0.8,0.4));

GLSL. Типы данных

Использование букв при присвоении значений векторам!!!

- **vec4 a = vec4(1.0, 2.0, 3.0, 4.0);**
- **float posX = a.x;**
- **float posY = a[1];**
- **vec2 posXY = a.xy;**
- **float depth = a.w;**

Компоненты положения **x, y, z, w**

Компоненты цвета **r, g, b**

Компоненты текстур **s, t, p, q**

GLSL. Типы данных

Массивы. Поддерживаются массивы любого типа.

vec4 *points* a[]; //Без указания размера

vec4 *points* a[10]; //С указанием размера

Тип **void**. Определяет функцию, которая НЕ возвращает значение.

GLSL. Квалификаторы

Квалификаторы задают переменным особый смысл. В GLSL доступны следующие:

- **const** - объявление **НЕИЗМЕНЯЕМОЙ** константы.
- **attribute - глобальные** переменные, которые определяют, каким образом данные из VBO передаются в вершинный шейдер. Атрибуты могут использоваться только в вершинных шейдерах. Для шейдера это - read-only переменная.

attribute vec4 atr_position;

GLSL. Квалификаторы

Квалификаторы задают переменным особый смысл:

- **uniform - глобальная** переменная, которая передаётся OpenGL в шейдеры и устанавливается перед выполнением шейдеров. Может использоваться в обоих типах шейдеров, для шейдеров является read-only (см. далее).

GLSL. Квалификаторы

В целом **Юниформы** — это внешние данные, которые могут быть использованы для расчетов в шейдере, но не могут быть перезаписаны. Униформы могут быть переданы как в вершинный, так и во фрагментный шейдеры. Униформы никак не связаны с конкретной вершиной и являются глобальными константами. Например, в качестве униформ можно передать в шейдер координаты источника света и координаты глаза (камеры).

uniform vec4 cam_position;

https://www.khronos.org/opengl/wiki/Uniform_(GLSL)

GLSL. Квалификаторы

Квалификаторы задают переменным особый смысл:

- **varying – глобальные** переменные, используются для передачи интерполированных данных между вершинным и фрагментным шейдерами. Доступны для записи в вершинном шейдере, и read-only для фрагментного шейдера.

```
varying vec2 v_texCoord;
```

GLSL. Квалификаторы

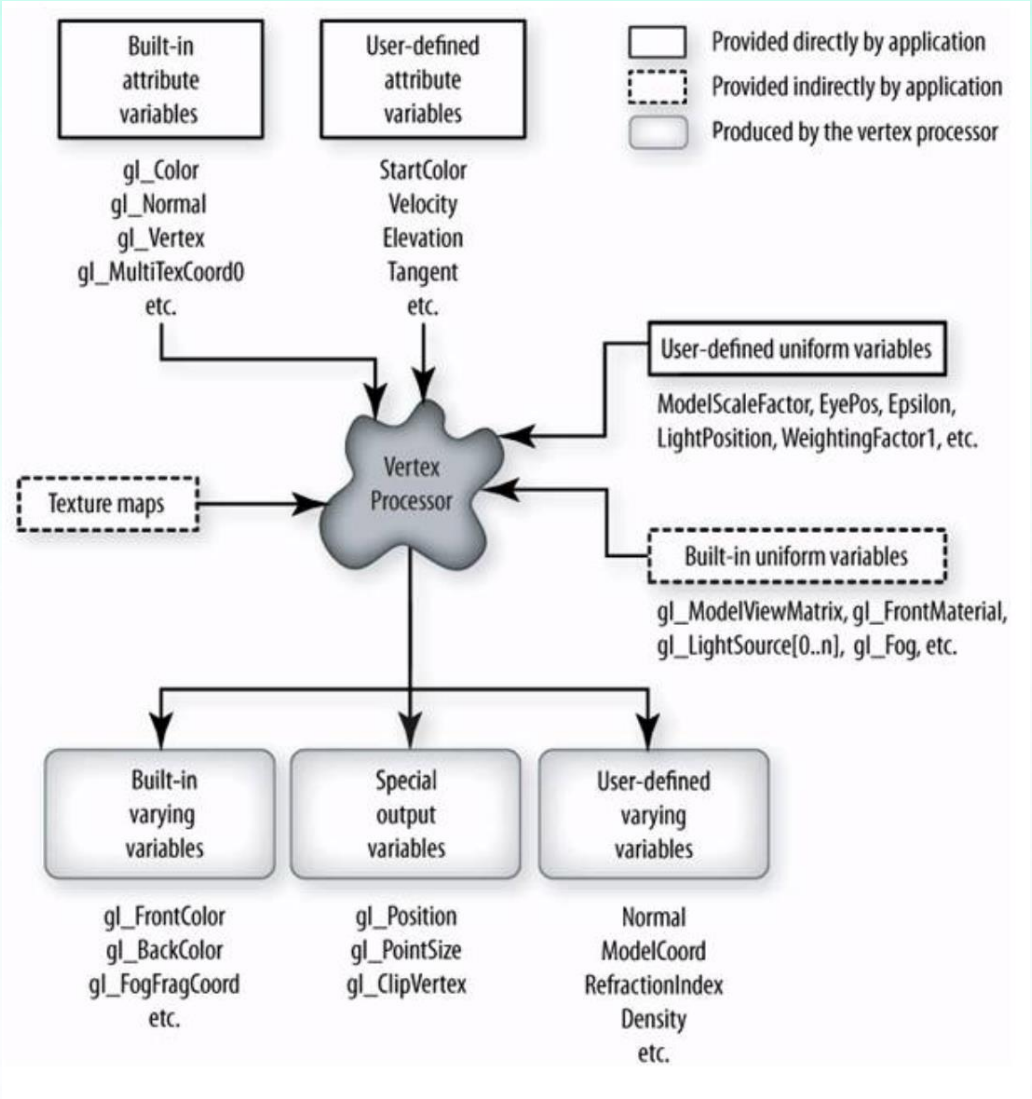
Передача параметров по значению:

- **in** – переменные,
- **out** – переменны,
- **inout** – переменные, используются для передачи интерполированных данных между вершинным и фрагментным шейдерами. Доступны для записи в вершинном шейдере, и read-only для фрагментного шейдера.

```
// Vertex Shader
out VertexData
{
vec3 color;
vec2 texCoord;
} outData;
```

```
// Geometry Shader
in VertexData
{
 vec3 color;
ec2 texCoord;
} inData ;
```

GLSL. Переменные вершинного шейдера



Стандартно вершинная информация поступает через связанный **VBO**

GLSL. Переменные вершинного шейдера

Выходная структура

```
out gl_PerVertex
  { vec4 gl_Position;
     float gl_PointSize;
    float gl_ClipDistance[]; };
```

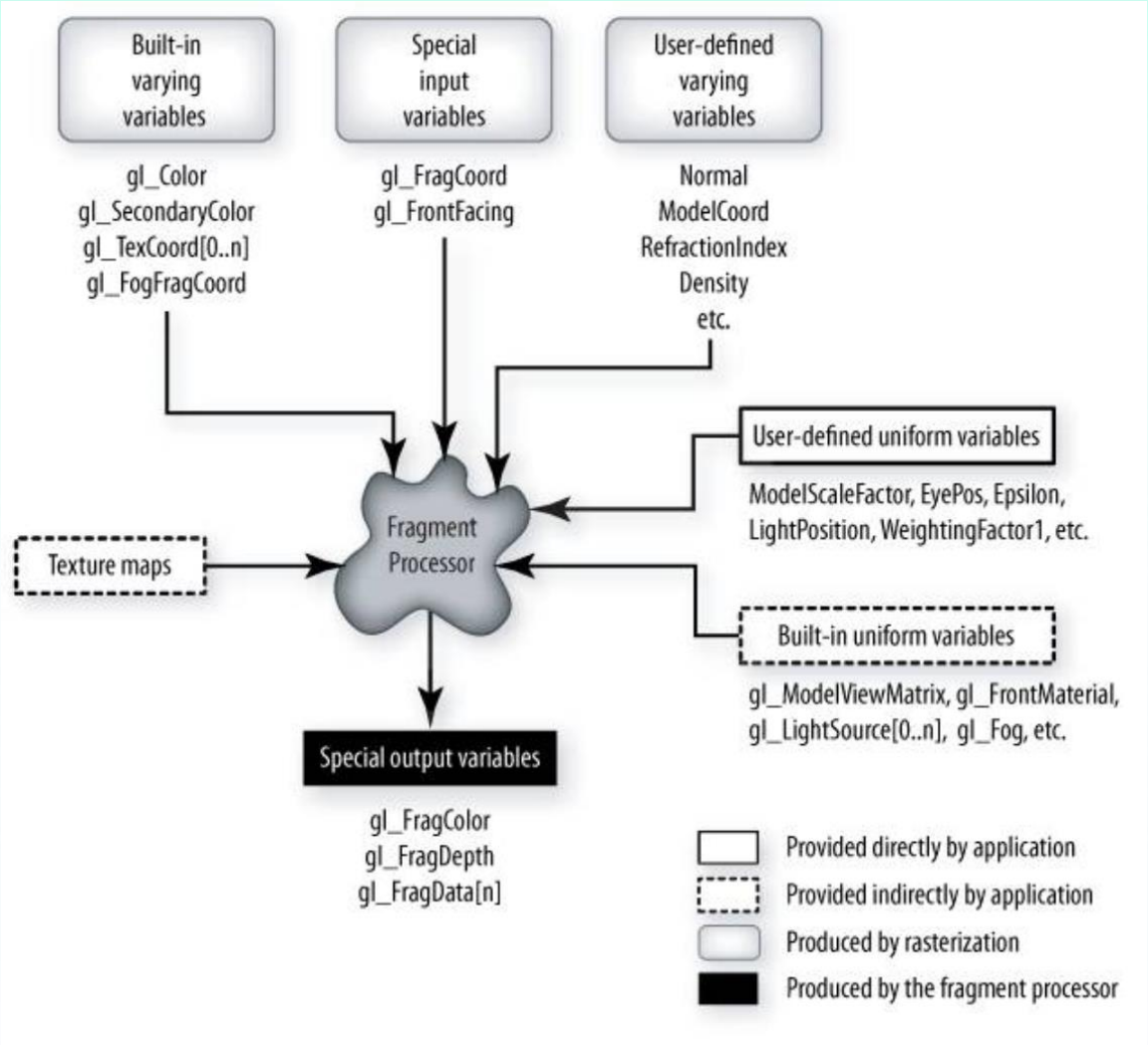
Структура блока вывода

gl_Position – координаты текущей вершины в пространстве отсечения (клиппирования),

gl_PointSize – размер растеризованой вершины (в пикселях),

gl_ClipDistance – расстояние от вершины до поупространства клиппирования.

GLSL. Переменные фрагментного шейдера



GLSL. Переменные фрагментного шейдера

Входные переменные:

`in vec4` gl_FragCoord – позиция фрагмента в оконном пространстве.

`in bool` gl_FrontFacing - **false**, если фрагмент принадлежит обратной грани;

`in vec2` gl_PointCoord - позиция фрагмента в примитиве «точки». Точка в пиксельном представлении имеет размеры. Задает «позицию» фрагмента в точке!

https://www.khronos.org/opengl/wiki/Fragment_Shader

GLSL. Переменные фрагментного шейдера

Стандартно выход фрагментного шейдера передается через буфер цветов glDrawBuffers.

Дополнительно:
out float gl_FragDepth;
out int gl_SampleMaskIn[];

https://www.khronos.org/opengl/wiki/Fragment_Shader

GLSL. Операторы управления

Стандартные Си операторы управления и цикла:

if (условие) {……};

if (условие) {……} **else** {……};

for (; условие;) {……};

while (условие) {……};

do {……} **while** (условие);

условие везде→ scalar **bool**

GLSL. Встроенные функции

Поддерживаются следующие группы функций:

- **Общие функции**
- **Угловые и тригонометрические**
- **Экспоненциальные функции**
- **Геометрические функции**
- **Матричные**
- **Векторные функции**
- **Функции доступа к текстурам**
- **Функции обработки фрагментов**
- **Функции шума (генераторы случайных чисел)**

- https://imperiya.by/show/opengl+tutorial
- OpenGL Tutorial 49: Geometry Shader Introduction https://www.youtube.com/watch?v=C8FK9Xn1gUM
- https://imperiya.by/video/W8YP9Im1pCS/OpenGL-Tutorial-49-Geometry-Shader-Introduction.html

END #12