GLEngine 0.1.0

Техническое описание

Ревизия 1.0

Хрущев Дмитрий aka DimaO

2023

Содержание

1 Требования к движку

2 Данные моделей

2.1 Массивы вершин

Массивами вершин управляют три основных класса: `Buffer`, `Attribute` и `VertexArray`.

Класс `Attribute` яыляется вспомогательным и описывает атрибуты вершины. Давайте вспомним, каким образом осуществляется связывание атрибута с объектом VAO (Vertex Array Object, Объект массива вершин) в библиотеке OpenGL 3.3.

Информация из официальной документации Khronos Group ® по функции `glVertexAttribPointer`.

```
void glVertexAttribPointer (
  GLuint index,
  GLint size,
  GLenum type,
  GLboolean normalized,
  GLsizei stride,
  const GLvoid *pointer);
```

index – номер атрибута. Он связан с параметром `location` (расположение) в лэйауте шейдера. Код шейдера в этом случае выглядит так: `layout(location = 0) in vec3 position;`
То есть, явно указанное расположение в шейдере и есть параметр `index`.

size – размер атрибута. Этот параметр указывает не размер атрибута в байтах, а число элементов атрибута. То есть размер вектора или матрицы в элементах (не в байтах).

type – тип элемента атрибута. Тип данных в элементе. не учитывает число элементов.

normalized – флаг нормализации атрибута.

stride – расстояние между началом соседних атрибутов в массиве. Имеет смысл размера вершины в байтах. Поскольку массив вершин может работать с несколькими буферами, необходимо учитывать, что размер вершины указывается для одного буфера. Атрибуты, хранящиеся в другом буфере, размер текущего не увеличивают.

pointer – смещение атрибута от начала буфера. Несмотря на то, что аргумент описан, как указатель, он таковым не является. Это именно смещение атрибута от начала буфера в байтах; целое число.

Необходимо также понимать, что аргументы `type` и `normalized` могут принимать только определенные, константные значения, отличающиеся по семантике от реального назначения.

Подведем итоги: выберем структуру класса атрибутов, которую можно будет использовать максимально удобно и быстро.

```
class Attribute
{
  private:
  unsigned int m_index;
  AttributeType m_type;
  unsigned int m_gl_type;
  size_t m_element_size;
  size_t m_element_count;
  bool m_normalized;
  ...
};
```

m_index – location (расположение) атрибута в шейдерной программе.

m_type – интерфейсный тип атрибута. Используется перечислитель с ключевым словом `class` для исключения приведения типа из целого.

m_gl_type – промежуточное представление типа для OpenGL. изменяется синхронно с полем `m_type` и предназначено для хранения типа, используемого в функции `glVertexAttribPointer`.

m_element_size – вспомогательное поле, содержащее размер элемента атрибута в байтах.

m_element_count – число элементов в векторе или матрице.

m_normalized – флаг нормализации.

Поля `m_element_size` и `m_element_count` позволяют расчитать размер атрибута в буфере.

Класс `Buffer` является непосредственным хранилищем данных массива вершин. Для полноценного связывания атрибута с шейдерной программой необходимо хранить все атрибуты в объекте буфера вместе с

данными. Узнав размер каждого атрибута, и сложив эти размеры вместе, мы получаем размер вершины. Зная размер вершины и размер буфера, можо определить число вершин в буфере, что необходимо функции `glDrawArrays` для отрисовки.

Таким образом, кроме размеров буфера в байтах нам нужно хранить в нем еще и список всех атрибутов вершин, хранящихся в буфере. Структура класса буфера выглядит так:

```
class Buffer
{
private:
  unsigned int m_handle;
  size_t m_size;
  void *m_data;
  std::vector<Attribute *> m_attributes;
  BufferAccess m_access;
  BufferOptimization m_optimization;
  ...
};
```

m_handle – имя буфера в контексте OpenGL. Получается предварительным вызовом функции `glGenBuffers`.

m_size – размер буфера в байтах.

m_data – непосредственно данные. Размер данных точно равен содержимому поля `m_size`.

m_attributes – список атрибутов.

m_access – поле, определяющее тип доступа к данным (чтение, рисование, копирование).

m_optimization – поле, определяющее оптимизацию хранения (статическое, динамическое, потоковое).

Класс `VertexArray` предназначен для хранения связей данных с атрибутами и