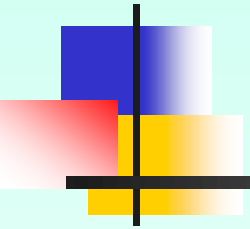
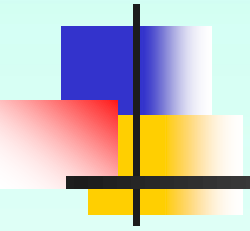


COMPUTER GRAPHICS

ЗАСОБИ ПРОГРАМУВАННЯ КОМП'ЮТЕРНОЇ ГРАФІКИ

Лек. 11 2021 ІПЗ-19



COMPUTER GRAPHICS

OPEN_GL (part 2)

Open GL

- OpenGL. Objects / Объекты.
- OpenGL. Buffers / Буфер.
- Поток вершин. Атрибуты, Индексы.
- VAO, VBO, EBO.
- Отрисовка.

OpenGL. Objects

Объект OpenGL – конструкт, который содержит некоторое состояние.

Объект OpenGL ← → контейнер **состояния**.

Графический контекст – вспомогательный объект для взаимодействия графического приложения, операционной системы и видеокарты.

https://www.khronos.org/opengl/wiki/OpenGL_Object#Object_types

OpenGL. Objects

При создании графического контекста происходит создание окна, в которые осуществляется вывод графики, и объектов (буферов, массивов) для рендеринга.
Когда Объект связан (**bound**) с контекстом OpenGL, состояние, которое он содержит, отображается в состоянии контекста.
!!! Функции, работающие с контекстом, будут использовать это состояние.

*Библиотека **GLFW**– для создания графического контекста.*
*Библиотека **GLEW**– для динамической линковки.*

OpenGL. Objects

Создание (**create**) и деструкция (**delete**).

Создание

void **glGen*** (GLsizei ***n***, GLuint ****objects***);

* тип объекта,

n - количество создаваемых объектов,

****objects*** – целое, идентифицирующее объект (не 0).

Уничтожение

void **glDelete***(GLsizei ***n***, GLuint ****objects***);

https://www.khronos.org/opengl/wiki/OpenGL_Object#Object_types

OpenGL. Objects

Связывание (**binding**) с контекстом.

void **glBind***(Glenum *target*, GLuint *object*);

* тип объекта,

target - область контекста, с которым связывается объект,

object – объект, который связывается.

!!! Если объект связывается с областью контекста, которая уже связана, старая связь разрывается.

Объект **0** не использовать.

https://www.khronos.org/opengl/wiki/OpenGL_Object#Object_types

OpenGL. Objects. Types

Регулярные объекты:

- **Buffer object.**
- Query object.
- Renderbuffer object.
- **Sampler** object.
- Texture object.

Контейнеры:

- **Vertix Array object.**
- Framebuffer object.
- Program Pipeline object.
- Transform Feedback object.

https://www.khronos.org/opengl/wiki/OpenGL_Object#Object_types

OpenGL. Objects. Identifiers

Identifier	Object Type
GL_BUFFER	Buffer Object
GL_SHADER	Shader Object
GL_PROGRAM	Program Object
GL_VERTEX_ARRAY	Vertex Array Object
GL_QUERY	Query Object
GL_PROGRAM_PIPELINE	Program Pipeline Object
GL_TRANSFORM_FEEDBACK	Transform Feedback Object
GL_SAMPLER	Sampler Object
GL_TEXTURE	Texture Object
GL_RENDERBUFFER	Renderbuffer Object
GL_FRAMEBUFFER	Framebuffer Object

https://www.khronos.org/opengl/wiki/OpenGL_Object#Object_types

OpenGL. Buffers

БУФЕР (**Buffer**) - объект контекста OpenGL, который хранит массив неформатированных данных. Они могут использоваться для хранения данных вершин, пиксельных данных выбранных из изображений и др.

VBO – Vertex Buffer Objects

PBO – Pixel Buffer Objects

UBO – Uniform Buffer Objects

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

OpenGL. Buffers

Создание буфера
void **glGenBuffers**(GLsizei ***n***, GLuint ****objects***)

Уничтожение буфера
void **glDeleteBuffers**(GLsizei ***n***, GLuint ****objects***);

Связывание буфера (биндинг)
void **glBindBuffer**(enum ***target***, uint ***bufferName***);

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

OpenGL. Buffers

Инициализация буфера
void **glBufferData**( )

GLenum *target*	Тип буфера (GL_ARRAY_BUFFER, GL_ELEMENT_ARRAY, GL_TEXTURE_BUFFER, ……
GLsizeiptr *size*	Размер создаваемого буфера в байтах
Const Glvoid * *data*	Указатель на данные, которые копируются в буфер
GLenum *usage*	Символическая константа, определяющая использование буфера (GL_STATIC_DRAW, GL_DINAMIC_DRAW, GL_STREAM_DRAW, …

http://docs.gl/gl3/glBufferData

OpenGL. Buffers

Тип буфера:

- **GL_ARRAY_BUFFER -** буфер вершин
- **GL_ELEMENT_ARRAY -** буфер индексов
- **GL_TEXTURE_BUFFER –** буфер текстур
- **GL_UNIFORM_BUFFER –** буфер uniform переменных
- ……..

http://docs.gl/gl3/glBufferData

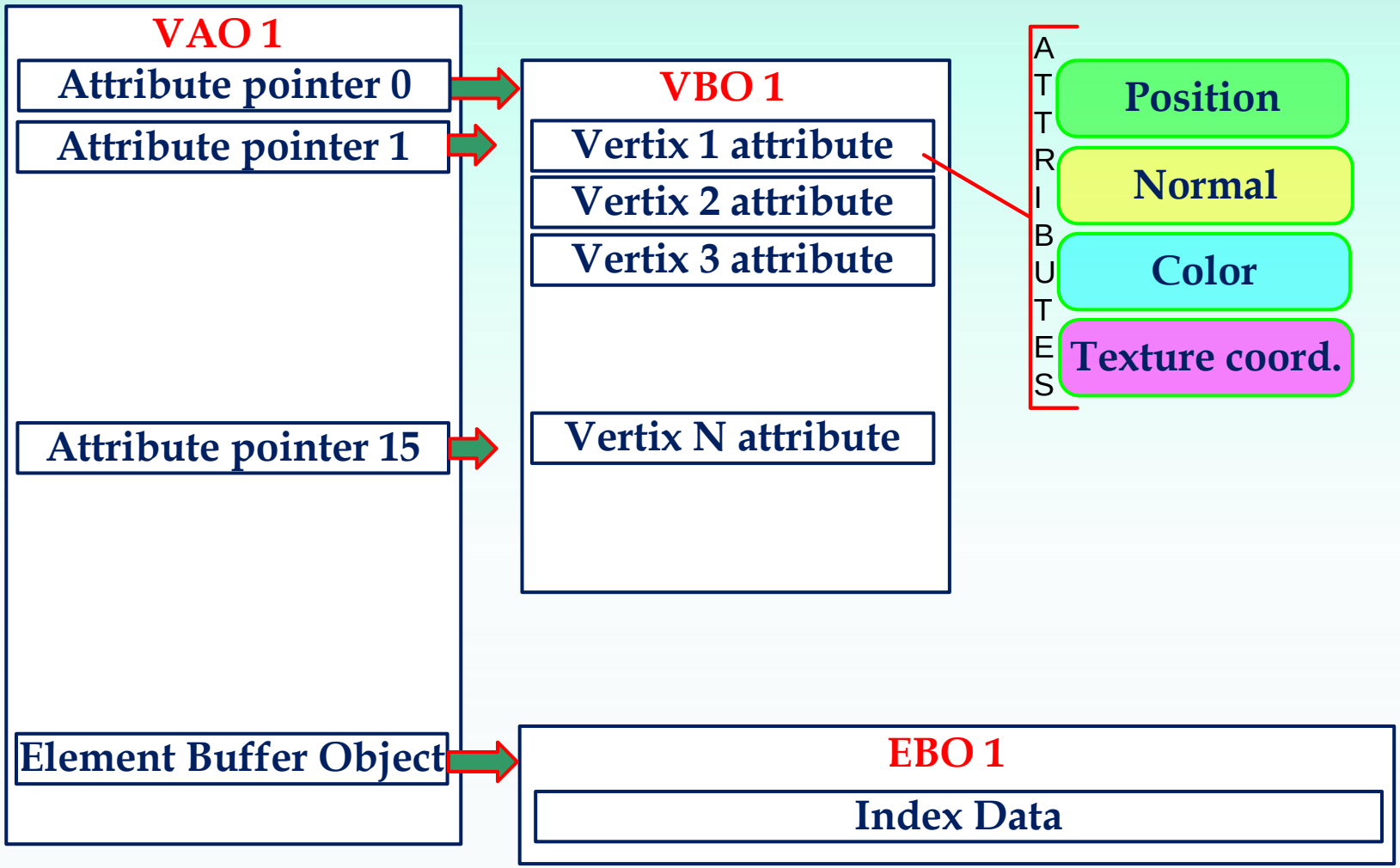
OpenGL. Buffers

Использование буфера данных

STATIC - данные задаются один раз.
DINAMIC - данные задаются (изменяются) редко (частично).
STREAM - данные задаются (изменяются) часто. *После каждого использования!!!*

http://docs.gl/gl3/glBufferData

OpenGL. Buffer & Container



OpenGL. Vertex Data

ATTRIBUTE — это свойство вершины:

- Координаты положения в пространстве

  x, y, z, w

- Компоненты вектора нормали

  Nx, Ny, Nz

- Цвет

  R, G, B, A

- Текстурные координаты

  u, v

Атрибуты доступны **вершинному шейдеру** только для чтения и не могут быть перезаписаны.

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

OpenGL. Vertix Stream

Vertix Pistion

{{1.0,1.0,1.0}, {0.0,0.0,0.0}, {0.0,0.0,1.0}}

Vertix Colors

{{100,50,15}, {200,100,125}, {0,0,0}}

Vertix Normal

{{0.5,0.5,0.5}, {0.1,0.24,0.18}, {0.45,0.25,0.37}}

АТРИБУТЫ ВЕРШИН

Vertix Index {2,1,0, 2, 1, 2}

{[{0.0,0.0,1.0},{0,0,0},{0.45,0.25,0.37}],[{0.0,0.0,0.0},{200,100,125},{0.1,0.24,0.18}],[{1.0,1.0,1.0},{100,50,15},{0.5,0.5,0.5}],[{0.0,0.0,1.0},{0,0,0},{0.45,0.25,0.37}],[{0.0,0.0,0.0},{200,100,125},{0.1,0.24,0.18}],[{0.0,0.0,1.0},{0,0,0},{0.45,0.25,0.37}]

ПОТОК ВЕРШИН (STREAM)

OpenGL. VBO

VBO – Vertex Buffer Objects – буфер вершин (вершинный буфер) — область данных, хранящая информацию обо всех или нескольких атрибутах (элементах)вершин. Буфер, который используется как источник данных о вершинах.

https://www.khronos.org/opengl/wiki/Vertex_Specification#Vertex_Array_Object

OpenGL. VAO

VAO – Vertex Array Objects – дескриптор вершинного массива - это объект OpenGL, который хранит всю информацию, необходимую для формирования потока вершин.

VAO не копирует, не фиксирует и не сохраняет содержимое ссылочных буферов. Если меняются какие-либо данные в буферах, на которые ссылается существующий VAO, эти изменения будут видны пользователям VAO.

OpenGL. VAO

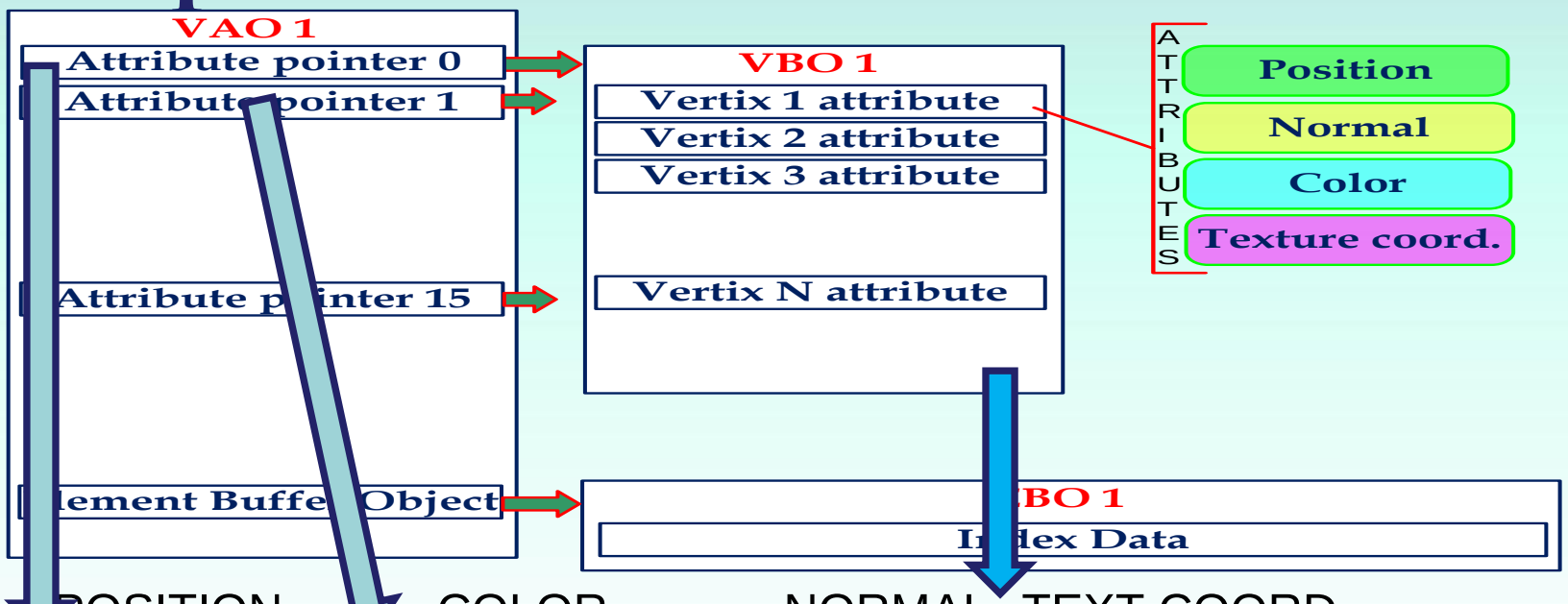
VertexAttribPointer

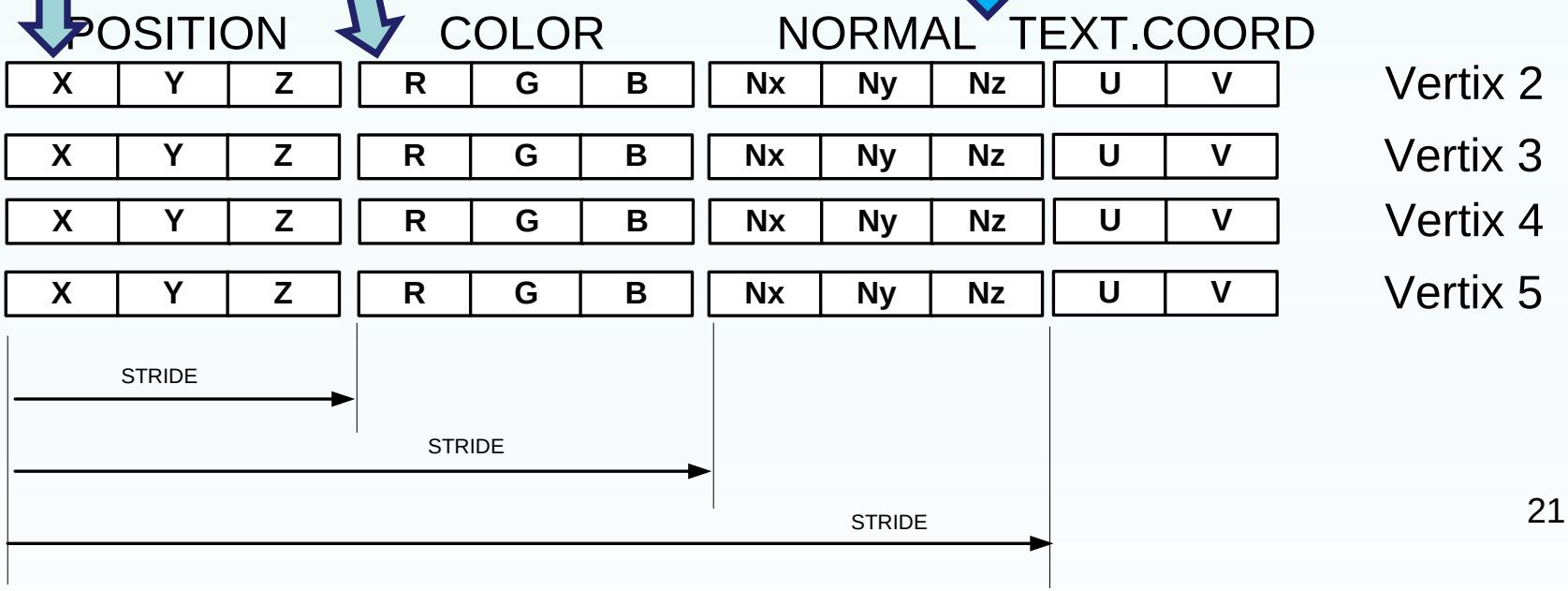
glVertexAttribPointer ( )
glVertexAttribIPointer ( )
glVertexAttribLPointer ( )

GLuint *index*	Индекс атрибута
GLint *size*	Количество компонент атрибута (1,2,3,4)
GLenum *type*	Тип данных каждого компонента атрибута
GLboolean *normalized*	Необходимость нормализовать (fixed point) GL_TRUE
GLsizei *stride*	Смещение между последовательными атрибутами
const GLvoid * *pointer*	Смещение к первому атрибуту в массиве

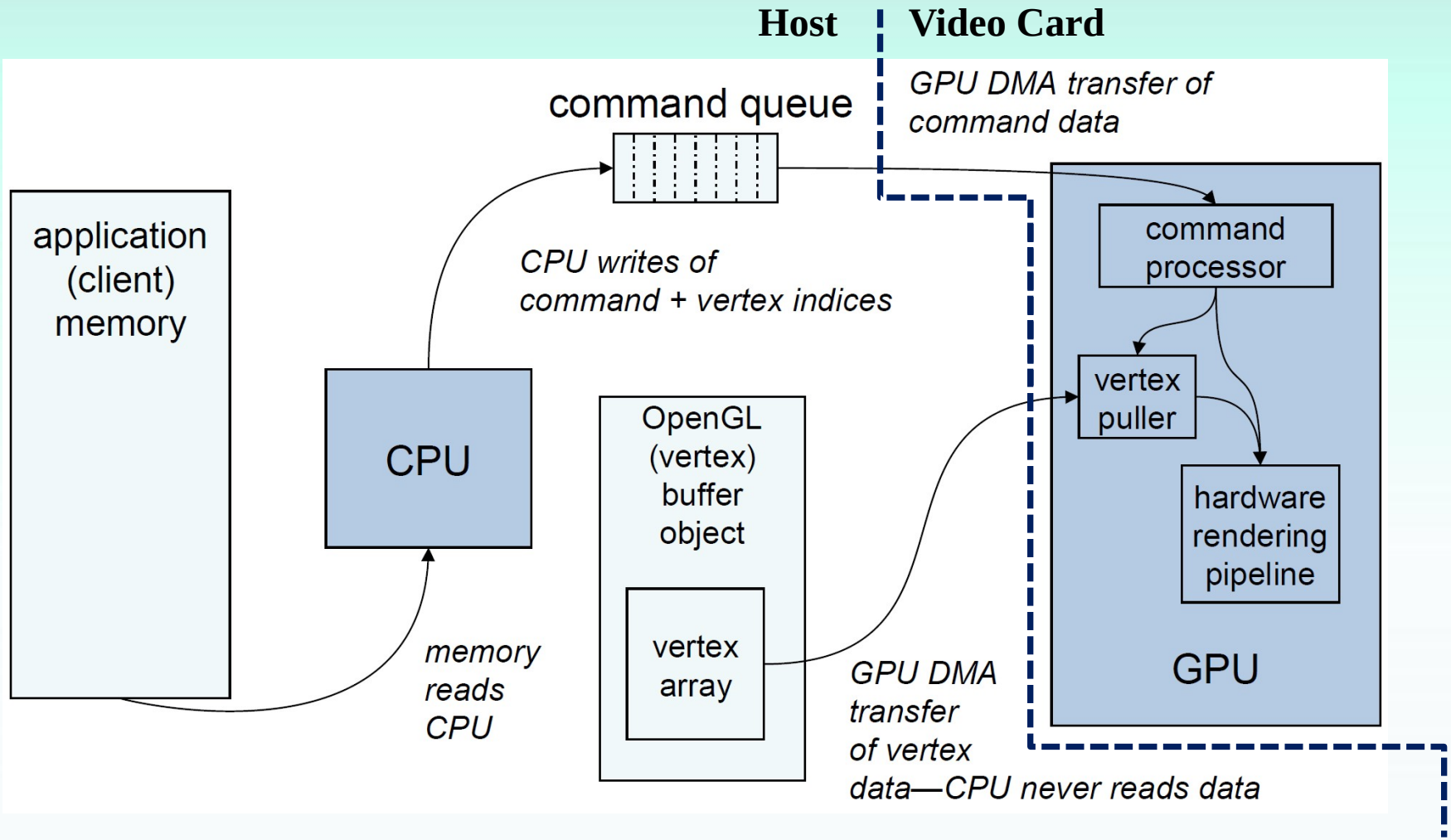
http://docs.gl/gl3/glVertexAttribPointer

OpenGL. Buffer & Container





OpenGL. Draw


Host
Video Card
command queue
GPU DMA transfer of command data
application (client) memory
CPU writes of command + vertex indices
command processor
vertex puller
hardware rendering pipeline
CPU
OpenGL (vertex) buffer object
vertex array
memory reads CPU
GPU DMA transfer of vertex data—CPU never reads data
GPU

OpenGL. Draw

Отрисовка – «естественный» порядок

void **glDrawArrays** ( )

void **glDrawArraysInstanced** ( )

GLenum *mode*	Режим – задает ТИП примитива, который отрабатывается
GLint *first*	индекс первой вершины в массиве
GLsizei *count*	Количество вершин

http://docs.gl/gl3/glDrawArrays

OpenGL. Draw

Отрисовка - порядок задается индексами

void **glDrawElements** ( )

void **glDrawRangeElements** ( )

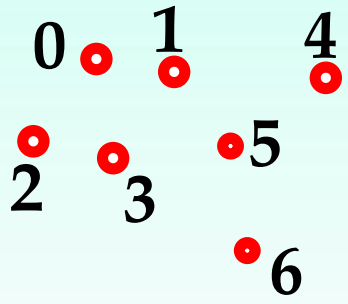
GLenum *mode*	Режим – задает ТИП примитива, который отрабатывается
GLsizei *count*	Количество элементов для отрисовки
GLenum *type*	Тип индексов (GL_UNSIGNED_INT, ….)
Const Glvoid* *indices*	Определяет смещение первого индекса в буфере привязанного к GL_ELEMENT_ARRAY_BUFFER

http://docs.gl/gl3/glDrawElements

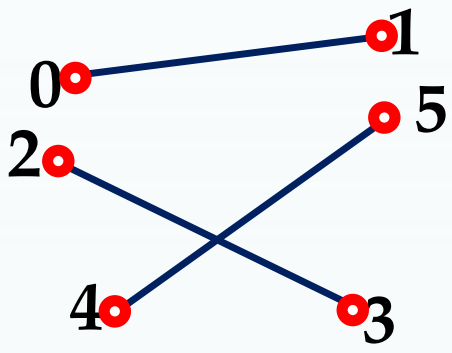
OpenGL. Draw. Primitives

ТИП ПРИМИТИВА - указывает как интерпретировать поток вершин

GL_POINTS *n* - вершин → *n* точек



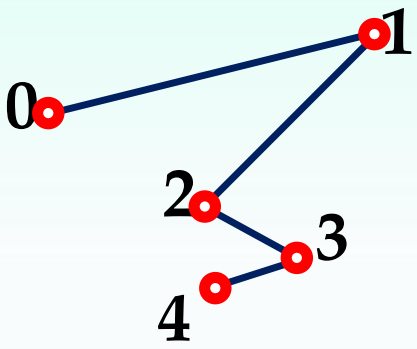
GL_LINES *n* - вершин → *n*/2 отрезков



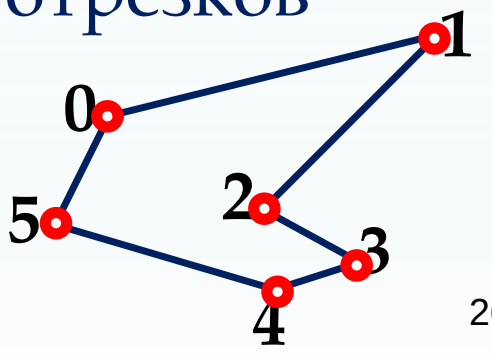
OpenGL. Draw. Primitives

ТИП ПРИМИТИВА - указывает как интерпретировать поток вершин

GL_LINE_STRIP *n* - вершин → *n* -1 отрезков



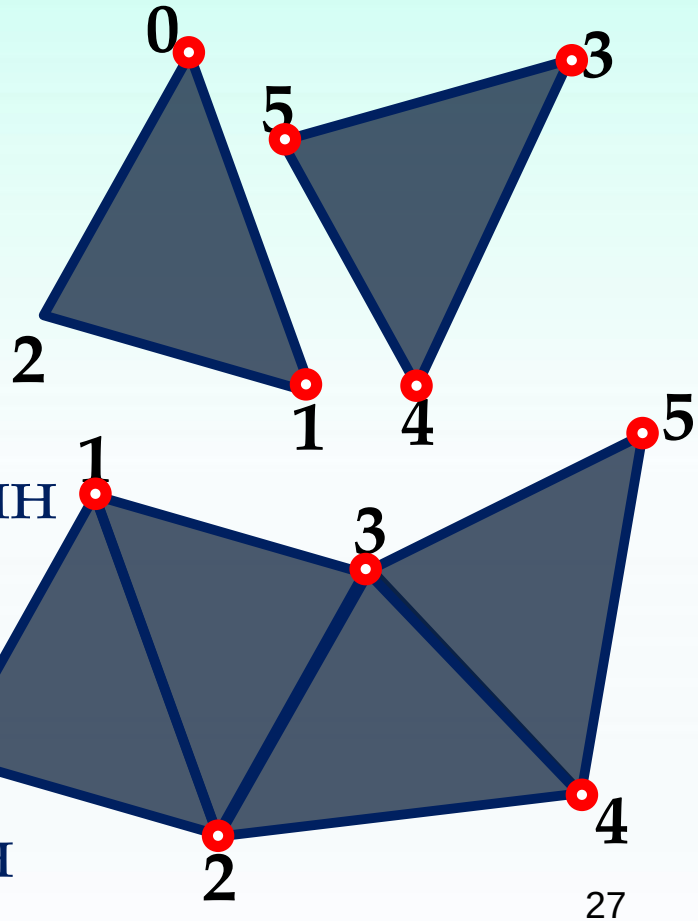
GL_LINE_LOOP *n* - вершин → *n* отрезков



OpenGL. Draw. Primitives

ТИП ПРИМИТИВА - указывает как интерпретировать поток вершин

GL_TRIANGLES *n* – вершин
→ *n/3* - триангл



GL_TRIANGLE_STRIP *n* – вершин
→ *n-2* - триангл

GL_TRIANGLE_FAN *n* – вершин
→ *n-2* - триангл

OpenGL.

Каркас графической программы

```
void main() {
    Инициализация контекста
    Инициализация функций OpenGL
    Создание буферов, текстур, шейдеров …
    while (окно не закрыто) {
        Обработка событий (мышь, клава, …
        Обновление сцены …
        Рендеринг кадра
       Переключение буферов
    }
    Удаление буферов, текстур, шейдеров …
}
```

OpenGL. Draw

Типичная последовательность (1)

```
//initialization
// Создание VAO, VBO, EBO
    GLuint myVBO, myVAO, myEBO;

    glGenVertexArrays(1, &myVAO);
    glGenBuffers(1, &myVBO);
    glGenBuffers(1, &myEBO);
```

OpenGL. Draw

Типичная последовательность (2)

```
//initialization
// Связывание + занесение данных
glBindBuffer(GL_ARRAY_BUFFER, myVBO);
glBufferData(GL_ARRAY_BUFFER, , , , );

glBindBuffer(GL_ELEMENT_ARRAY_BUFFER,
myEBO);
glBufferData(GL_ELEMENT_ARRAY_BUFFER, , , , );
```

OpenGL. Draw

Типичная последовательность (3)

```
//initialization
// Занесение указателей атрибутов
 glBindVertexArray(myVAO);
// Position
 glVertexAttribPointer(0,   , , , , , );
 glEnableVertexAttribArray(0);
// Color attribute
 glVertexAttribPointer(1, , , , ,);
 glEnableVertexAttribArray(1);

 glBindVertexArray(0); / / Unbind VAO
```

OpenGL. Draw

Типичная последовательность (4)

```
//rendering  отрисовка в главном цикле
glBindVertexArray(myVAO);
……
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, myEBO);
glDrawArrays(GL_POINTS,  , );
//or
glDrawElements(GL_TRIANGLES, , , );
……
glBindVertexArray(0);
```

http://docs.gl/gl3/glVertexAttribPointer

END #11