# LEC12: Vertex Buffer Object and Vertex Array Object

Ku-Jin Kim
School of Computer Science & Engineering
Kyungpook National University

## Contents

## Program Examples

- Please prepare the program files in the lecture note board
  - LEC12.0_translate_vs.c
  - LEC12.1_translate_one_vbo.c
  - LEC12.2_translate_two_vbo.c
  - LEC12.3_translate_vao.c

## LEC12.0_translate_vs.c

- Currently, commands in mydisplay() are repeated
- Move those in mydisplay() to myinit()

## Vertex Buffer Object (VBO)

- provides methods for uploading vertex attribute (position, normal vector, color, etc.) to GPU memory for non-immediate-mode rendering.

## LEC12.1_translate_one_vbo.c (1)

```
Gluint vbo[1];
glGenBuffers(1, vbo);
glBindBuffer(GL_ARRAY_BUFFER, vbo[0]);
glBufferData(GL_ARRAY_BUFFER,
                    2 * 3 * 4 * sizeof(GLfloat), NULL, GL_STATIC_DRAW);
glBufferSubData(GL_ARRAY_BUFFER, 0, 3 * 4 * sizeof(GLfloat), vertices);
glBufferSubData(GL_ARRAY_BUFFER, 3 * 4 * sizeof(GLfloat),
                    3 * 4 * sizeof (GLfloat),  colors);
```

## LEC12.1_translate_one_vbo.c (2)

```
GLuint loc;
loc = glGetAttribLocation(prog, "aPosition");
glEnableVertexAttribArray(loc);
glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid*)0);
loc = glGetAttribLocation(prog, "aColor");
glEnableVertexAttribArray(loc);
glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0,
                                    (GLvoid*)(3 * 4 * sizeof(GLfloat)));
```

## glGenBuffers

- returns n buffer object names in buffers

```
Ex) Gluint vbo[1];
    glGenBuffers(1, vbo);
```

- void glGenBuffers( GLsizei n,  GLuint * buffers);
  - n : Specifies the number of buffer object names to be generated.
  - buffers : Specifies an array in which the generated buffer object names are stored.

## glBindBuffer

- binds a buffer object to the specified buffer binding point

  Ex) glBindBuffer(GL_ARRAY_BUFFER, vbo[0]);

- void glBindBuffer( GLenum target,  GLuint buffer);
  - target: Specifies the target to which the buffer object is bound
  - buffer: Specifies the name of a buffer object.

## Buffer Binding Targets

| Buffer Binding Target | Purpose |
|---|---|
| GL_ARRAY_BUFFER | Vertex attributes |
| GL_ATOMIC_COUNTER_BUFFER | Atomic counter storage |
| GL_COPY_READ_BUFFER | Buffer copy source |
| GL_COPY_WRITE_BUFFER | Buffer copy destination |
| GL_DISPATCH_INDIRECT_BUFFER | Indirect compute dispatch commands |
| GL_DRAW_INDIRECT_BUFFER | Indirect command arguments |
| GL_ELEMENT_ARRAY_BUFFER | Vertex array indices |
| GL_PIXEL_PACK_BUFFER | Pixel read target |
| GL_PIXEL_UNPACK_BUFFER | Texture data source |
| GL_QUERY_BUFFER | Query result buffer |
| GL_SHADER_STORAGE_BUFFER | Read-write storage for shaders |
| GL_TEXTURE_BUFFER | Texture data buffer |
| GL_TRANSFORM_FEEDBACK_BUFFER | Transform feedback buffer |
| GL_UNIFORM_BUFFER | Uniform block storage |

## glBufferData

- used for creating a new data store for a buffer object

  Ex) glBufferData(GL_ARRAY_BUFFER, 2 * 3 * 4 * sizeof(GLfloat), NULL, GL_STATIC_DRAW);

- void glBufferData( GLenum target,  GLsizeiptr size,  const void * data, GLenum usage);
  - target: Specifies the target to which the buffer object is bound for glBufferData
  - size: Specifies the size in bytes of the buffer object's new data store.
  - data: Specifies a pointer to data that will be copied into the data store for initialization, or NULL if no data is to be copied.
  - usage: Specifies the expected usage pattern of the data store. It must be GL_STREAM_DRAW, GL_STREAM_READ, GL_STREAM_COPY, GL_STATIC_DRAW, GL_STATIC_READ, GL_STATIC_COPY, GL_DYNAMIC_DRAW, GL_DYNAMIC_READ, or GL_DYNAMIC_COPY.

## The constant for usage

**The frequency of access**

- STREAM
  - The data store contents will be modified once and used at most a few times.
- STATIC
  - The data store contents will be modified once and used many times.
- DYNAMIC
  - The data store contents will be modified repeatedly and used many times.

**The nature of access:**

- DRAW
  - The data store contents are modified by the application, and used as the source for GL drawing and image specification commands.
- READ
  - The data store contents are modified by reading data from the GL, and used to return that data when queried by the application.
- COPY
  - The data store contents are modified by reading data from the GL, and used as the source for GL drawing and image specification commands.

## glBufferSubData

- redefine some or all of the data store for the specified buffer object

```
Ex) glBufferSubData(GL_ARRAY_BUFFER, 0, 3 * 4 *sizeof(GLfloat), vertices);

    glBufferSubData(GL_ARRAY_BUFFER, 3 * 4 * sizeof(GLfloat),
                    3 * 4 * sizeof (GLfloat),  colors);
```

- void glBufferSubData( GLenum target, GLintptr offset, GLsizeiptr size, const void * data);
  - target : Specifies the target to which the buffer object is bound for glBufferSubData
  - offset: Specifies the offset into the buffer object's data store where data replacement will begin, measured in bytes.
  - size: Specifies the size in bytes of the data store region being replaced.
  - data: Specifies a pointer to the new data that will be copied into the data store.

## glVertexAttribPointer

- specify the location and data format of the array of generic vertex attributes at index

```
glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid*)0);
…
glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0,
                                        (GLvoid*)(3 * 4 * sizeof(GLfloat)));
```

- void glVertexAttribPointer( GLuint index, GLint size, GLenum type, GLboolean normalized, GLsizei stride, const void * pointer);
  - pointer: Specifies a offset of the first component of the first generic vertex attribute in the array in the data store of the buffer currently bound to the GL_ARRAY_BUFFER target. The initial value is 0.

## LEC12.2_translate_two_vbo.c

```
GLfloat vertices[] = {
        -0.5, -0.5, 0.0, 1.0,
        +0.5, -0.5, 0.0, 1.0,
        -0.5, +0.5, 0.0, 1.0,
};

GLfloat colors[] = {
        1.0, 0.0, 0.0, 1.0, // red
        0.0, 1.0, 0.0, 1.0, // green
        0.0, 0.0, 1.0, 1.0, // blue
};
```

```
GLfloat vertices2[] = {
        -0.8, -0.8, 0.0, 1.0,
        +0.2, -0.8, 0.0, 1.0,
        -0.8, +0.2, 0.0, 1.0,
};
GLfloat colors2[] = {
        1.0, 0.0, 0.0, 1.0,
        1.0, 0.0, 0.0, 1.0,
        1.0, 0.0, 0.0, 1.0,
};
```

## Drawing two objects by using VBO

- Complicated & Repetition
- For drawing multiple objects, using VAO is a solution !!!

## Vertex Array Object (VAO)

- A Vertex Array Object (VAO) is an OpenGL Object that stores all of the state needed to supply vertex data.
- It stores the format of the vertex data as well as the VBO providing the vertex data arrays.

# LEC12.3_translate_vao.c (1)

```
GLuint vbo[2], vao[2];  // global variables
…
glGenVertexArrays(2, vao);
glBindVertexArray(vao[0]);

glGenBuffers(2, vbo);  ////////////////////////////////////////////////////////////////
glBindBuffer(GL_ARRAY_BUFFER, vbo[0]);
glBufferData(GL_ARRAY_BUFFER, 2 * 3 * 4 * sizeof(GLfloat), NULL, GL_STATIC_DRAW);
glBufferSubData(GL_ARRAY_BUFFER, 0, 3 * 4 * sizeof(GLfloat), vertices);
glBufferSubData(GL_ARRAY_BUFFER, 3 * 4 * sizeof(GLfloat), 3 * 4 * sizeof(GLfloat),  colors);

GLuint loc;
loc = glGetAttribLocation(prog, "aPosition");
glEnableVertexAttribArray(loc);
glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid*)0);
loc = glGetAttribLocation(prog, "aColor");
glEnableVertexAttribArray(loc);
glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid*)(3 * 4 * sizeof(GLfloat)));
```

## LEC12.3_translate_vao.c (2)

```
glBindVertexArray(vao[1]);

glBindBuffer(GL_ARRAY_BUFFER, vbo[1]);
glBufferData(GL_ARRAY_BUFFER, 2 * 3 * 4 * sizeof(GLfloat), NULL, GL_STATIC_DRAW);
glBufferSubData(GL_ARRAY_BUFFER, 0, 3 * 4 * sizeof(GLfloat), vertices2);
glBufferSubData(GL_ARRAY_BUFFER, 3 * 4 * sizeof(GLfloat), 3 * 4 * sizeof(GLfloat), colors2);

loc = glGetAttribLocation(prog, "aPosition");
glEnableVertexAttribArray(loc);
glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid*)0);
loc = glGetAttribLocation(prog, "aColor");
glEnableVertexAttribArray(loc);
glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid*)(3 * 4 * sizeof(GLfloat)));
```

## glGenVertexArrays

- returns n vertex array object names in arrays.

  Ex) glGenVertexArrays(2, vao);

- void glGenVertexArrays( GLsizei n,  GLuint *arrays);
  - n : Specifies the number of vertex array object names to generate.
  - arrays : Specifies an array in which the generated vertex array object names are stored.

## glBindVertexArray

- binds the vertex array object with name array

  Ex) glBindVertexArray(vao[0]);

- void glBindVertexArray( GLuint array);
  - array: Specifies the name of the vertex array to bind.

## mydisplay

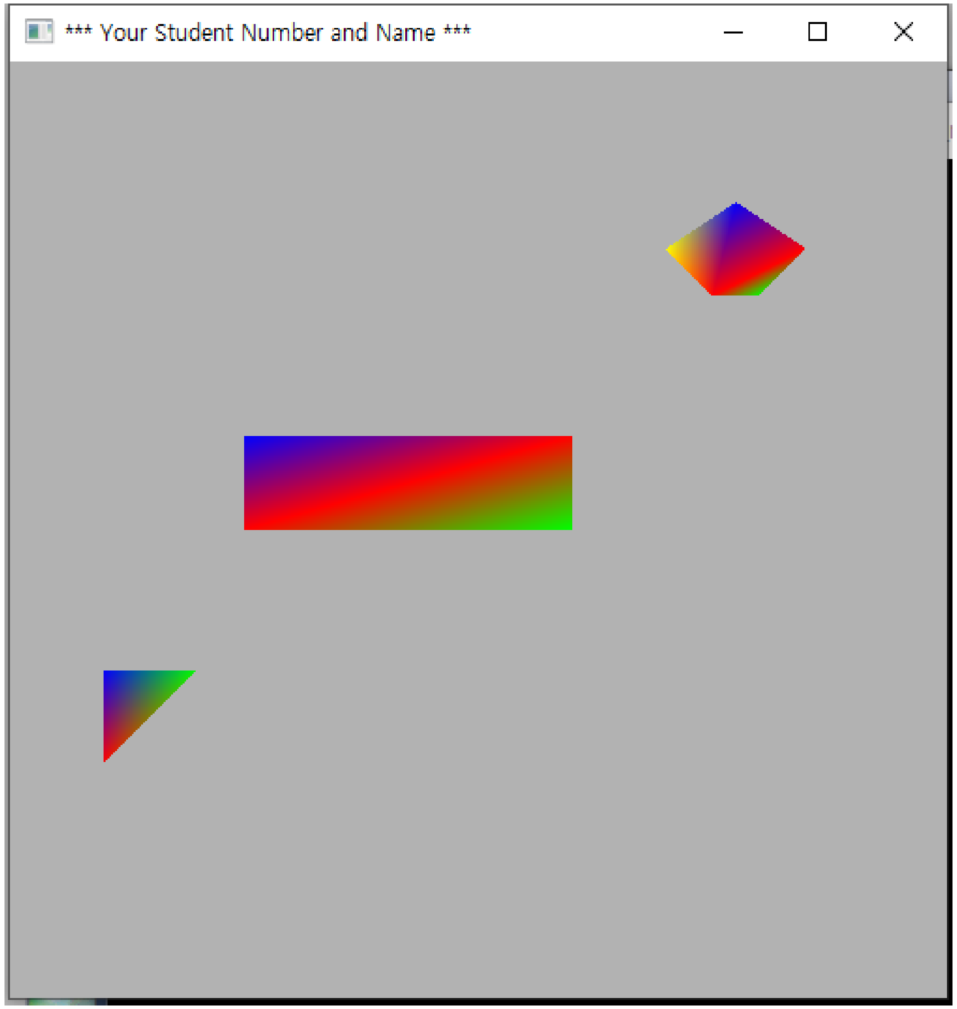
```
glBindVertexArray(vao[0]);
glDrawArrays(GL_TRIANGLES, 0, 3);

glBindVertexArray(vao[1]);
glDrawArrays(GL_TRIANGLES, 0, 3);
```

## HW#12 Using Vertex Array Object (1)

- Due date: This Friday 6:00pm
- Draw a colored pentagon, a colored rectangle, and a colored triangle by using VAO in a static state (no translation).
- Each object must be drawn as a whole shape in the window (no hidden or clipped part).
- Submit the **.c file** through LMS.

## HW#12 Using Vertex Array Object (2)

Execution example


*** Your Student Number and Name ***