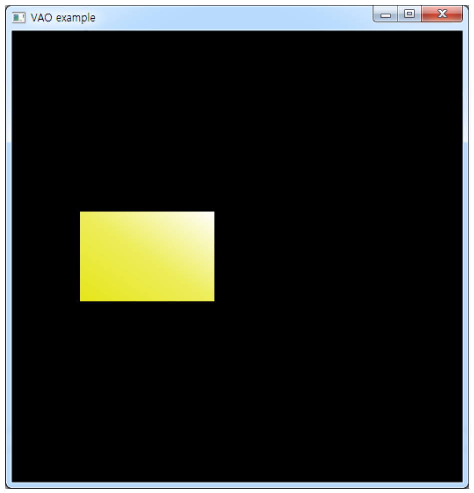
Computer Graphics - [OpenGL] VAO, VBO

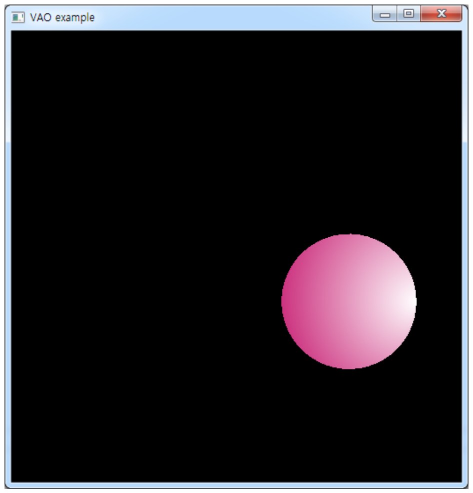
Sung Soo Hwang

Vertex Array Object (VAO)

- What is a VAO?
  - An internal data object of OpenGL that holds references to buffers (i.e., Vertex Buffer Objects (VBO)) associated with vertex attributes (e.g., position, color, normal, etc.)
  - Note that it does not copy the contents of the buffers.

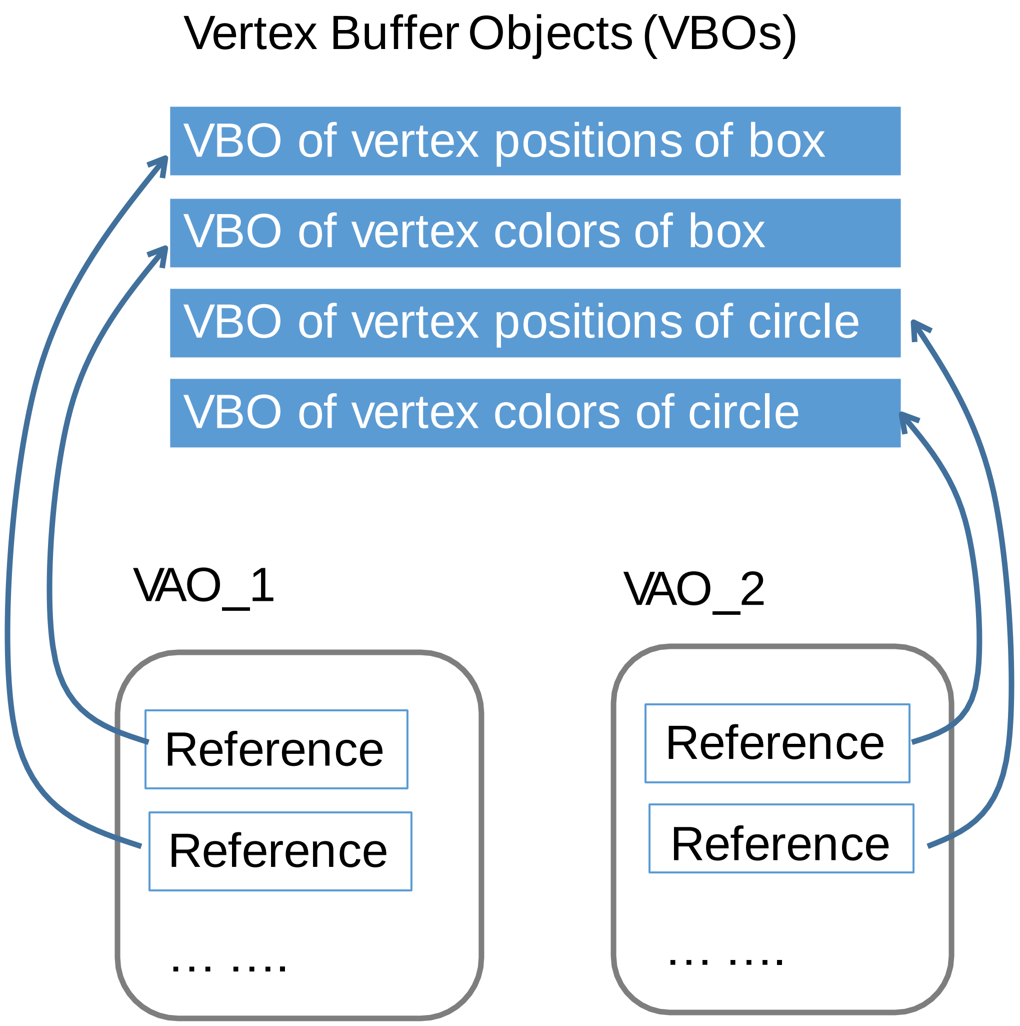


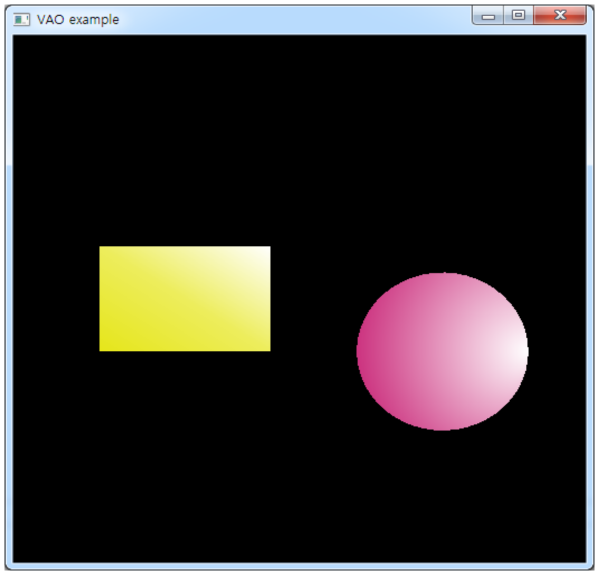
```
glBindVertexArray(VAO_1);
glDrawArrays(......);
```



```
glBindVertexArray(VAO_2);
glDrawArrays(......);
```

Vertex Array Object (VAO)


Vertex Buffer Objects (VBOs)
VBO of vertex positions of box
VBO of vertex colors of box
VBO of vertex positions of circle
VBO of vertex colors of circle
VAO_1
Reference
Reference
… ….
VAO_2
Reference
Reference
… ….



```
glBindVertexArray(VAO_1);
glDrawArrays(……);
glBindVertexArray(VAO_2);
glDrawArrays(……);
```

Vertex Array Object (VAO)

- Typical codes for initializing VAOs

```
GLuint VAOs[2];
glGenVertexArrays(2, VAOs);

glBindVertexArray(VAOs[0]);
// ... initialize vertex buffers to be referenced by VAOs[0] ...

glBindVertexArray(VAOs[1]);
// ... initialize vertex buffers to be referenced by VAOs[1] ...
```

Vertex Array Object (VAO)

- Main functions for VAOs

```
void glGenVertexArrays(    GLsizei          n,
                           GLuint*          arrays);
```

- Returns the ID numbers of ***n*** vertex array objects in ***arrays***.
- There is no guarantee that the ID numbers form a contiguous set of integers.

```
void glDeleteVertexArrays(  GLsizei          n,
                            const GLuint*    arrays);
```

- Deletes ***n*** vertex array objects whose ID numbers are stored in ***arrays***.
- If a vertex array object that is currently bound is deleted, the binding reverts to zero.

```
void glBindVertexArray( GLuint   array);
```

- Binds the vertex array object with ***array***, which is the ID number of a vertex array object previously returned from a call to glGenVertexArrays, or zero to break the existing vertex array object binding.

Vertex Buffer Object (VBO)

- What is a VBO?
  - A data object that represents storage for vertex attributes (e.g., position, normal, etc.)
  - It can be assigned in 3 steps:

```
// Step 1: Generate a new buffer object.
glGenBuffers(1, Buffers);
// Step 2: Bind the buffer object to a specific type.
glBindBuffer(GL_ARRAY_BUFFER, Buffers[0]);
// Step 3: Copy vertex data to the buffer object.
glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices, GL_STATIC_DRAW);
```

Vertex Buffer Object (VBO)

```
void glGenBuffers( GLsizei        n,
                   GLuint*        buffers);
```

- Returns the ID numbers of n buffer objects in ***buffers***.
- There is no guarantee that the ID numbers form a contiguous set of integers.

```
void glDeleteBuffers(    GLsizei          n,
                         const GLuint*    buffers);
```

- Deletes ***n*** buffer objects identified by the elements of the array ***buffers***.
- If a buffer object that is currently bound is deleted, the binding reverts to 0.

```
void glBindBuffer(   GLenum        target,
                     GLuint        buffer);
```

- Binds a buffer object specified by ***buffer*** to a given ***target***.
- ***target*** represents a buffer type to which the buffer object can be bound (see the next slide).

Vertex Buffer Object (VBO)

- Predefined constants that can be used for targets

GL_ARRAY_BUFFER	Vertex attributes (e.g., positions, colors, normals, etc.)
GL_ELEMENT_ARRAY_BUFFER	Vertex array indices
GL_TEXTURE_BUFFER	Texture data
…	…

For the full list of the constants, refer to https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/glBindBuffer.xhtml.

Vertex Buffer Object (VBO)

```
void glBufferData(    GLenum          target,
                      GLsizeiptr      size,
                      const GLvoid*   data,
                      GLenum          usage);
```

- Assigns a new data store for the buffer bound to ***target***; any pre-existing data s tore is deleted.
- ***size*** specifies the size in bytes of the buffer object's new data store.
- **data** specifies a pointer to client data that will be copied into the data store for i nitialization.
- **usage** specifies the expected usage pattern of the data store. It can be one of t he following constants:

GL_STREAM_DRAW	GL_STREAM_READ	GL_STREAM_COPY
GL_STATIC_DRAW	GL_STATIC_READ	GL_STATIC_COPY
GL_DYNAMIC_DRAW	GL_DYNAMIC_READ	GL_DYNAMIC_COPY

Vertex Buffer Object (VBO)

Frequency of access	- **STREAM**:	modified once / used at most a few times
	- **STATIC**:	modified once / used many times
	- **DYNAMIC**:	modified repeatedly / used many times

Nature of access	- **DRAW**:	specified by the client / used as the source for drawing
	- **READ**:	filled with an OpenGL buffer / used by the client
	- **COPY**:	filled with an OpenGL buffer / used as the source for drawing