Computer Graphics Homework 2: OpenGL Vertex array object

Ali Mehmandoost (963624020)

November 17, 2017

Contents

1 What is Vertex array object

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

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 Buffer Objects providing the vertex data arrays. Note that VAOs do not copy, freeze or store the contents of the referenced buffers if you change any of the data in the buffers referenced by an existing VAO, those changes will be seen by users of the VAO.

2 Whats the difference between Vertex array Object and Vertex Buffer object

http://www.songho.ca/opengl/gl_vbo.html

Vertex array functions are in the client state [RAM] and the data in the arrays must be re-sent to the server each time when it is referenced. But Vertex buffer object (VBO) creates "buffer objects" for vertex attributes in high-performance memory on the server side and provides same access functions to reference the arrays, which are used in vertex arrays, such as glVertexPointer(), glNormalPointer(), glTexCoordPointer(), etc.

3 Creating A Vertex array object

To create a Vertex array object first we need to enable the vertex array we want to use with glEnableClientState.

3.1 glEnableClientState

glEnableClientState and glDisableClientState enable or disable individual client-side capabilities. By default, all client-side capabilities are disabled. Both glEnableClientState and glDisableClientState take a single argument, cap, which can assume one of the following values:

3.1.1 GL_COLOR_ARRAY

If enabled, the color array is enabled for writing and used during rendering when glArrayElement, glDrawArrays, glDrawElements, glDrawRangeElements glMultiDrawArrays, or glMultiDrawElements is called.

3.1.2 GL_EDGE_FLAG _ARRAY

If enabled, the edge flag array is enabled for writing and used during rendering when glArrayElement, glDrawArrays, glDrawElements, glDrawRangeElements glMultiDrawArrays, or glMultiDrawElements is called.

3.1.3 GL_FOG_COORD _ARRAY

If enabled, the fog coordinate array is enabled for writing and used during rendering when glArrayElement, glDrawArrays, glDrawElements, glDrawRangeElements glMultiDrawArrays, or glMultiDrawElements is called.

3.1.4 GL_INDEX_ARRAY

If enabled, the index array is enabled for writing and used during rendering when glArrayElement, glDrawArrays, glDrawElements, glDrawRangeElements glMultiDrawArrays, or glMultiDrawElements is called.

3.1.5 GL_NORMAL_ARRAY

If enabled, the normal array is enabled for writing and used during rendering when glArrayElement, glDrawArrays, glDrawElements, glDrawRangeElements glMultiDrawArrays, or glMultiDrawElements is called.

3.1.6 GL _SECONDARY_COLOR_ARRAY

If enabled, the secondary color array is enabled for writing and used during rendering when glArrayElement, glDrawArrays, glDrawElements, glDrawRangeElements glMultiDrawArrays, or glMultiDrawElements is called.

3.1.7 GL_TEXTURE_COORD_ARRAY

If enabled, the texture coordinate array is enabled for writing and used during rendering when glArrayElement, glDrawArrays, glDrawElements, glDrawRangeElements glMultiDrawArrays, or glMultiDrawElements is called.

3.1.8 GL_VERTEX_ARRAY

If enabled, the vertex array is enabled for writing and used during rendering when glArrayElement, glDrawArrays, glDrawElements, glDrawRangeElements glMultiDrawArrays, or glMultiDrawElements is called.

4 Specifies the location and data format

After Creating VAO we need to specifies the location and data format of Array with one of the following functions:

4.1 glVertexPointer

`https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/glVertexPointer.xml`

void glVertexPointer(GLint size, GLenum type, GLsizei stride, const GLvoid * pointer);

4.1.1 Parameters

1. size Specifies the number of coordinates per vertex. Must be 2, 3, or 4. The initial value is 4.

2. type Specifies the data type of each coordinate in the array. Symbolic constants GL_SHORT, GL_INT, GL_FLOAT, or GL_DOUBLE are accepted. The initial value is GL_FLOAT.

3. stride Specifies the byte offset between consecutive vertices. If stride is 0, the vertices are understood to be tightly packed in the array. The initial value is 0.

4. pointer Specifies a pointer to the first coordinate of the first vertex in the array. The initial value is 0.

4.1.2 Description

glVertexPointer specifies the location and data format of an array of vertex coordinates to use when rendering. size specifies the number of coordinates per vertex, and must be 2, 3, or 4. type specifies the data type of each coordinate, and stride specifies the byte stride from one vertex to the next, allowing vertices and attributes to be packed into a single array or stored in separate arrays. If a non-zero named buffer object is bound to the GL_ARRAY_BUFFER target while a vertex array is specified, pointer is treated as a byte offset into the buffer object's data store. Also, the buffer object binding (GL_ARRAY_BUFFER_BINDING) is saved as vertex array client-side state (GL_VERTEX_ARRAY_BUFFER_BINDING). When a vertex array is specified, size, type, stride, and pointer are saved as client-side state, in addition to the current vertex array buffer object binding.

4.2 glColorPointer

https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/glColorPointer.xml

void glColorPointer(GLint size, GLenum type, GLsizei stride, const GLvoid * pointer);

4.2.1 Parameters

1. size Specifies the number of components per color. Must be 3 or 4. The initial value is 4.
2. type Specifies the data type of each color component in the array. Symbolic constants GL_BYTE, GL_UNSIGNED_BYTE, GL_SHORT, GL_UNSIGNED_SHORT, GL_INT, GL_UNSIGNED_INT, GL_FLOAT, and GL_DOUBLE are accepted. The initial value is GL_FLOAT.
3. stride Specifies the byte offset between consecutive colors. If stride is 0, the colors are understood to be tightly packed in the array. The initial value is 0.
4. pointer Specifies a pointer to the first component of the first color element in the array. The initial value is 0.

4.2.2 Description

glColorPointer specifies the location and data format of an array of color components to use when rendering. size specifies the number of components per color, and must be 3 or 4. type specifies the data type of each color component, and stride specifies the byte stride from one color to the next, allowing vertices and attributes to be packed into a single array or stored in separate arrays. If a non-zero named buffer object is bound to the GL_ARRAY_BUFFER target while a color array is specified, pointer is treated as a byte offset into the buffer object's data store. Also, the buffer object binding is saved as color vertex array client-side state. When a color array is specified,size, type, stride, and pointer are saved as client-side state, in addition to the current vertex array buffer object binding.

5 Render primitives from VAO:

After creating vertex attributes in VAO we can render them with one of the following functions:

5.1 glDrawArrays

https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/glDrawArrays.xml void glDrawArrays(GLenum mode, GLint first, GLsizei count);

5.1.1 Parameters

1. mode Specifies what kind of primitives to render. Symbolic constants GL_POINTS, GL_LINE_STRIP, GL_LINE_LOOP, GL_LINES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN, GL_TRIANGLES, GL_QUAD_STRIP, GL_QUADS, and GL_POLYGON are accepted.
2. first Specifies the starting index in the enabled arrays.
3. count Specifies the number of indices to be rendered.

5.1.2 Description

glDrawArrays specifies multiple geometric primitives with very few subroutine calls. Instead of calling a GL procedure to pass each individual vertex, normal, texture coordinate, edge flag, or color, you can prespecify separate arrays of vertices, normals, and colors and use them to construct a sequence of primitives with a single call to glDrawArrays.

When glDrawArrays is called, it uses count sequential elements from each enabled array to construct a sequence of geometric primitives, beginning with element first. mode specifies what kind of primitives are constructed and how the array elements construct those primitives. If GL_VERTEX_ARRAY is not enabled, no geometric primitives are generated.

Vertex attributes that are modified by glDrawArrays have an unspecified value after glDrawArrays returns. For example, if GL_COLOR_ARRAY is enabled, the value of the current color is undefined after glDrawArrays executes. Attributes that aren't modified remain well defined.

5.2 glDrawElements

https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/glDrawElements.xml void glDrawElements(GLenum mode, GLsizei count, GLenum type, const GLvoid * indices);

5.2.1 Parameters

1. mode Specifies what kind of primitives to render. Symbolic constants GL_POINTS, GL_LINE_STRIP, GL_LINE_LOOP, GL_LINES,

GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN, GL_TRIANGLES, GL_QUAD_STRIP, GL_QUADS, and GL_POLYGON are accepted.

2. count Specifies the number of elements to be rendered.

3. type Specifies the type of the values in indices. Must be one of GL_UNSIGNED_BYTE, GL_UNSIGNED_SHORT, or GL_UNSIGNED_INT.

4. indices Specifies a pointer to the location where the indices are stored.

5.2.2 Description

glDrawElements specifies multiple geometric primitives with very few subroutine calls. Instead of calling a GL function to pass each individual vertex, normal, texture coordinate, edge flag, or color, you can prespecify separate arrays of vertices, normals, and so on, and use them to construct a sequence of primitives with a single call to glDrawElements.

When glDrawElements is called, it uses count sequential elements from an enabled array, starting at indices to construct a sequence of geometric primitives. mode specifies what kind of primitives are constructed and how the array elements construct these primitives. If more than one array is enabled, each is used. If GL_VERTEX_ARRAY is not enabled, no geometric primitives are constructed.

Vertex attributes that are modified by glDrawElements have an unspecified value after glDrawElements returns. For example, if GL_COLOR_ARRAY is enabled, the value of the current color is undefined after glDrawElements executes. Attributes that aren't modified maintain their previous values.

6 Example

Lets Draw a color triangle.
`https://github.com/Mehmandoost/ui36cg/tree/master/HW2`

```
#include <GL/glew.h>
#include <GL/glut.h>


void handler_display();

float vertices[] = {                    \
-1.0, -1.0, 0.0, \
```

```
1.0, -1.0, 0.0,  \
0.0, 1.0, 0.0 };

float colors[] = {                  \
1.0, 0.0, 0.0, 1.0, \
0.0, 1.0, 0.0, 1.0, \
0.0, 0.0, 1.0, 1.0 };

int main(int argc, char **argv) {

glutInit(&argc, argv);
glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE);

glutInitWindowSize(500, 500);
glutInitWindowPosition(0, 0);

glutCreateWindow("VAO Test");

glutDisplayFunc(handler_display);
glutMainLoop();

}


void handler_display() {

glClear(GL_COLOR_BUFFER_BIT);

glVertexPointer(3, GL_FLOAT, 0, vertices);
glEnableClientState(GL_VERTEX_ARRAY);

glColorPointer(4, GL_FLOAT, 0, colors);
glEnableClientState(GL_COLOR_ARRAY);

glDrawArrays(GL_TRIANGLES, 0, 3);


glDisableClientState(GL_VERTEX_ARRAY);
glDisableClientState(GL_COLOR_ARRAY);
}
```