Vertex array: This is OpenGL special, direct x doesn’t have such an API. It’s to bind vertex buffer with a layout

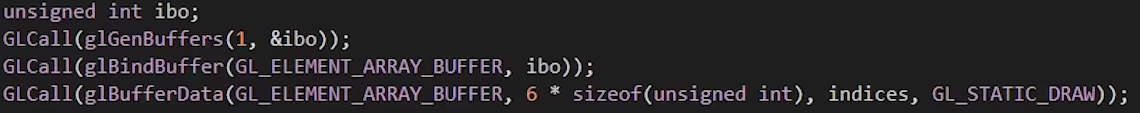
The procedure of drawing is bind vertex buffer, bind index buffer and draw the object. But after we bind the vertex buffer, we also need to specify the layout, and different buffer may have different layout. The layout is not store in the buffer.

Text

Description automatically generated

The layout function is glVertexAttribPointer. Each time when we bind buffer, we need to specify this function.

In bind buffer step, we bind the buffer into GL\_ARRAY\_BUFFER variable



We also need to bind the element array buffer.

After we use the buffer, we need to unbind the buffer

Text

Description automatically generated

When we want to draw the buffer again, we need to do the previous procedure again

A vertex array object will contain a binding between a vertex buffer and that vertex layout

The previous step: bind shader program, bind vertex buffer, set up vertex layout, bind index buffer, draw.

The new step: bind shader program, bind vertex array, bind index buffer, draw

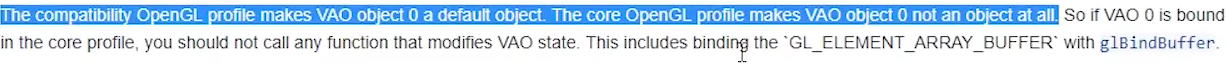
Technically, vertex array is mandatory, even if we don’t specify vertex array, it is used in OpenGL. The compatibility profile creates a vertex array object for us by default, but the core profile does not. So, we need to create it by ourselves, and bind it manually if we using the core profile.

So, we are going to tell OpenGL that we create the window and context with the core profile

A screenshot of a computer

Description automatically generated with medium confidence

In the document:



When we use OpenGL core, there will be no default vertex array context that help us. We need to first create and bind the vertex array context by ourselves, or else we are not able to use glEnableVertexAttribArray(0)

Before we bind the buffer, we define and bind vertex array first, and put the buffer into the vertex array buffer.

Text

Description automatically generated

Let’s check out our result, we still first unbind all the context

Text

Description automatically generated with low confidence

Instead of bind everything again, we just need to bind vao

A picture containing text

Description automatically generated

This function below link vao to the buffer

Text

Description automatically generated

Finally, should we use vertex array buffer by ourselves, or use the default one? The answer is that it depends. You can create one and never unbind the vertex array for the whole duration. And bind vertex and it attribute every time you draw. Another option is you create a vao each time when you create an object, and you bind vao before you draw.

In the past, it is fast to have just one global vao, and bind everything each time. Because more vao will occupy more memory. But now, things change. NVIDIA write 67 pages of report about how to improve the performance of managing many vao. And the OpenGL document encourage you to do that. If you really care about performance, it is recommended that you can make a comparison between these two methods.