**Create object vertex attributes data**

* If data is interleaved
  + Create array of all vertex attributes
* If data is non-interleaved
  + Create array of vertex positions
  + [Optional] Create array of vertex normal
  + [Optional] Create array of vertex texture coordinates
  + [Optional] Create array(s) of additional vertex attributes
* [Optional] Create array of indices to connect vertices in order

**Setup the Vertex Array Object (VAO)**

* Generate VAO descriptor using glGenVertexArrays()
* Bind vaod to be active using glBindVertexArray()

**Setup the Vertex Buffer Object (VBO)**

* Generate VBO descriptor using glGenBuffers()
* Bind vbod to GL\_ARRAY\_BUFFER using glBindBuffer()
* If vertex data is interleaved
* Send data to GPU using glBufferData()
* If vertex data is non-interleaved
* Allocate space to GPU using glBufferData() but do not send any data
* Send each individual array to the GPU using glBufferSubData() – set offset and size appropriately

**[Optional] Setup the Index Buffer Object (IBO)**

* Generate IBO descriptor using glGenBuffers()
* Bind ibod to GL\_ELEMENT\_ARRAY\_BUFFER using glBindBuffer()
* Send index array to GPU using glBufferData()

**When rendering the object**

* Bind vaod using glBindVertexArray()
* If object is dynamic and animated
  + Bind vbod using glBindBuffer()
  + Send updated data to GPU using glBufferSubData() according to interleaved or non-interleaved data
* If using an IBO
* Render object using glDrawElements()
* If not using an IBO
  + Render object using glDrawArrays()

**When cleaning up memory**

* Delete the VAO using glDeleteVertexArrays()
* Delete the VBO using glDeleteBuffers()
* If using an IBO
  + Delete the IBO using glDeleteBuffers()