# Cap1: Vertex Buffers and Drawing a Triangle in OpenGL

* Vertex buffer: memory buffer, a blow of memory where to push bytes on it. A vertex buffer is memory inside the GPU memory and then the drawcall will take it and put it in the window. Furthermore, you need to define how to interpretate that information.
* Shader is a program that runs in the GPU. A bunch of code, very specific but powerful.
* OpenGL operates as a static machine that depends on the buffer, shader and the drawcall. Simple.
* Every object has its own id. Buffer, shader, text etc…
* Doc. for the OpenGL library (glfw): <https://docs.gl/>

# Cap2: Attributes and Layouts in OpenGL

* This is the way you show how is the information in the buffer stored.

# Cap3: Shaders in OpenGL

* Normally, there is an standard to the shaders when you don’t define it.
* Shader is a program that runs in your GPU. Block of code. Run the GPU is because you want to specify the resources of the graphic target to do something. Moreover, the GPU is faster than the CPU.
* Nevertheless, the code will be executed from the CPU and the GPU. Depending on the purpose.
* The shader works to define to the GPU how to work with the objects.
* There are two main types of shaders:
  + Vertex shaders: for every vertex we want to draw. If triangle🡪three call to the vertex shader. It defines where the vertex is.
  + Fragment shaders equivalent to the pixel. This specify the surface of the objtect, filled by the pixels
* Rendering data:
  + Drawcall 🡪 vertex shader🡪 fragment shader

# Cap4: Index buffers

* OpenGL makes all with triangles. So when you need to draw a square, it is made by 2 triangles. Instead of define all the vertex of the triangle, because those are repeating it is possible to just define the new ones.
* In order to reuse the info of vertex it is used the index buffers. Those are other kind of buffer that only is going to index to the vertex buffer to define what are the vertex to draw. It is important to maintain the buffer index to be correct.

# Cap5: Checking errors

* OpenGL works with Flags. For this, it is necessary to call back the flags and see what are the errores they have.
* We have learnt about the Macros and how to wrap the code inside an errorhandling for the debugger.