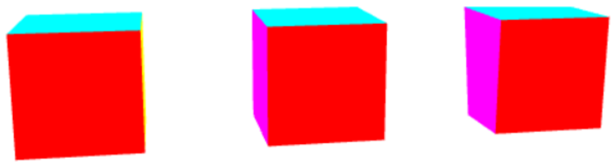
<u>Lab 13</u>

Here, we want to render three versions of the same cube,



and we will do this three different ways:

Next, we will use ***Instancing***.

Instancing provides a mechanism for having the graphics card render multiple copies of the same object using only one call to `gl.drawArrays`***`Instanced`***.

If there are a large number of copies of an object, then this method could provide a large performance boost.

When using instancing, the vertex shader has access to a built-in counter, `gl_InstanceID`.

In this lab, we will use this instance id to calculate how far to translate.

While both the modelview and projection matrices are initialized in the application and sent to the shader as a uniform variable, the modelview matrix is then updated in the vertex shader, using the instance id.

Remember that the matrices defined in the vertex shader for the translations and scaling are column-major, so they look like transposes of the mathematical definitions.

Can you explain how the instance id, `transX`, and the modelview matrix are updated in the vertex shader and work together to render three cubes?