**Graphics 1 – “Week 4, Day 1” lesson plan:**

* Overview of the VAO manager:
  + Note the VAO settings: loading and creating, and drawing
* Setting models in space
* Updating the shader:
  + #version 420
  + attribute 🡪 in
  + gl\_FragColor 🡪 out vec4 outputColour;
  + varying 🡪 out
* Loading models with different vertex layouts
  + Normals: we will need for lighting
  + Adding the vec3 for normals
  + Changing this to a vec4 (position, colour, normal)
  + Point out we will be adding a few more things to the vertex layout\*
* Uniform variables (again), in more depth
  + Split apart the model-view-perspective matrices
  + Point out the performance of doing this on “application” side vs “shader” side:
    - Show the underlying skeleton of the model
    - Show the skinned mesh shader
    - Point out how many 4x4 multiplies are happening…
* Point out the use of the “bool” uniform (really a float, or int) 🡪 *don’t* pass “true” or “false”
* Get rudimentary lighting:
  + Add a “light” at some location (uniform)
  + Add a calculation to give brightness over distance
  + Add another value to change this overall drop off (“attenuation”)
* Switch from wire-frame to solid
* Enable culling:
  + glClear( GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT )
  + glEnable/glDisable( GL\_DEPTH );
  + glEnable/glDisable( GL\_DEPTH\_TEST );
  + glPolygonMode( GL\_FRONT\_AND\_BACK, GL\_LINE/GL\_FILL/GL\_POINT)
  + glEnable/glDisable( GL\_CULL\_FACE );

\*Our final vertex layout (coming into the vertex shader) will eventually be this:

in vec4 vColour; // rgba

in vec4 vPosition; // xyzw

in vec4 vNormal; // normal xyz

in vec4 vUV\_x2; // Texture coordinates (2 of them)

in vec4 vTanXYZ; // Tangent to the surface

in vec4 vBiNormXYZ; // bi-normal (or bi-tangent) to the surface

in vec4 vBoneID; // passing it as 4 values (xyzw)

in vec4 vBoneWeight; // passing it as 4 values (xyzw)