LAB 1 - MY FIRST SHADER

LAB TASKS

1. Join the Slack channel for the module. Use the following link to sign up for the Slack channel for this module. Make sure to use your University email address and real name. The purpose of the Slack channel is to discuss module topics.

    a. https://join.slack.com/t/cmp301/shared_invite/MjMyODY5MzAwNDIzLTE1MDM2NjkzODctYWJkODEwNDI5Zg

2. Download the framework and make sure the example compiles and runs. This may require force building the DirectX ToolKit and the DXFramework, just right-click on the project and click build. The framework can be found on Blackboard. I will be putting a version of the framework up on GitLab please let me know if you want access to it.

3. Modify the pixel shader so the triangle is green instead of red.

4. Modify the vertex shader so the triangle is rendered twice its original size.

5. Within the example project (E01_ColourShader) create a new Mesh class that will render a square. This mesh class should extend the BaseMesh class from within the framework and will be similar to the triangle mesh in the framework but with more geometry data. Remember this works like a vertex array with an index array and you will need to update the vertex and index data. You should never need to modify the framework directly, unless specifically told to do so.

RESEARCH TASKS

Answer the following questions. Provide links or evidence where possible. If you want to check your answers please ask the module tutor.

1. Which shader first received the geometry data to be rendered?
2. When rendering a simple triangle how many times is the vertex shader called?
3. Assuming a window with resolution 800x600, if rendering a quad large enough to fill the entire window what is the minimum number of times the pixel shader will be called? And why?
4. Which of the following stage(s) happen between the Vertex and Pixel shaders when rendering the simple triangle from the lecture? And why? (Hint: look up the rendering pipeline on MSDN or Google or in the lecture slides).
    a. Rasterizer
    b. Hull shader stage
    c. Domain shader stage
    d. Stream output stage
    e. Tessellator stage