LAB 9 TESSELLATION

CODE TASKS

1. Build the triangle tessellation example discussed in the lecture (triangle version). Use the code provided in the lecture and on Blackboard (pre-made tessellation shader set).

2. Modify the tessellation factor:

    a. Increase the tessellation factor and note its effect on the rendered object.

    b. By hard coding values in the hull shader set the tessellation factors unevenly.

        i. For example, set the edges to (2, 4, 4) and set inside to 8.

        ii. Set edges to (1, 2, 3) and set inside to 4.

3. Update the Tessellation Shader Class so you can pass a tessellation factor value from the CPU (app.cpp) to the GPU (Hull Shader). Then, add keyboard controls to increase and decrease tessellation during runtime. You will want to clamp this to sensible values; minimum of 1 and a maximum of 64.

    a. Investigate the other tessellation partitioning types (currently using integer, try fractional_odd and fractional_even) this will require changing the “edge” and “inside” to non-integer values to so see how the effect the partitioning controls tessellation of the shape.

4. The current tessellation mesh contains a single triangle (3-point control patch), create a new tessellation mesh (within your current project, DO NOT modify the framework) that contains more than one triangle. Use this new mesh in your tessellation application. The same tessellation process should happen to all the triangles.

5. Create another tessellation mesh that will render a tessellated quad (4 control point patch). This will require significant changes to the tessellation process in the hull and domain shaders. Aspects of this are discussed in the lecture.

RESEARCH TASK

One of the major advantages of tessellation is it is dynamic. Using the tessellated quad from the previous question, investigate and implement distance controlled tessellation. The quad should have a higher tessellation factor when closer to the camera and a smaller value when further away from the camera.