REPORT

General – Rasterisation

We found the worksheet straightforward enough in the first few sections, running into our first bit of trouble when trying to fill the leftPixels and rightPixels arrays to fill triangles in ComputePolygonRows. We kept getting incorrect values when interpolating over the various vertex coordinates, but managed to get it working after a variety of debugs. We also had a bit of difficulty performing the checks correctly at the end of ComputePolygonRows to fill up the arrays properly. This was solved by a bit of good old pencil and paper working out of what was going on. The rest of the basic worksheet was more or less straightforward from then on, aside from an ongoing struggle with getting rotations working!

The rotation of the camera goes around the y-axis of the cube. The global 'yaw' variable was initially unable to be updated, so we ended up initializing this within the main and passing it through the Draw function within the while loop. The yaw variable was updated within the while loop whenever the user pressed the corresponding keyboard letters. Once the variable was passed over to the Draw function, the vertices points are multiplied by the rotation matrix R (which includes the updated yaw variable). Thus, this initial rotation of the image points gives us the ability to produce a general rotated image once it is drawn out by the DrawPolygonRows function; however once we got the rotation working, it no longer worked once we got to the end of the worksheet (segmentation fault occurs whenever a rotating key is pressed), and we have still been unable to solve the issue.

Extensions – Rasterisation

We implemented a couple of extensions for the rasterizer, both of which we also implemented in the raytracer.

The first extension we added was parallelization of a few for loops using OpenMP; there were many more parallelizable for loops in the rasterizer coursework, so that gave us a nice speed boost. Furthermore we compile with the –O3 flag, which optimizes the compilation for a little extra boost.

The second extension we implemented was a simple specular component to give a bit of a reflective look to the scene, and thus make it a bit more interesting than the basic scene. We implemented this using Phong reflection, which calculates a ratio of the amount of light

reflected; this gave us the image in figure 1. Not much has changed, but you can see a slightly more complex sheen on the tall blue block.

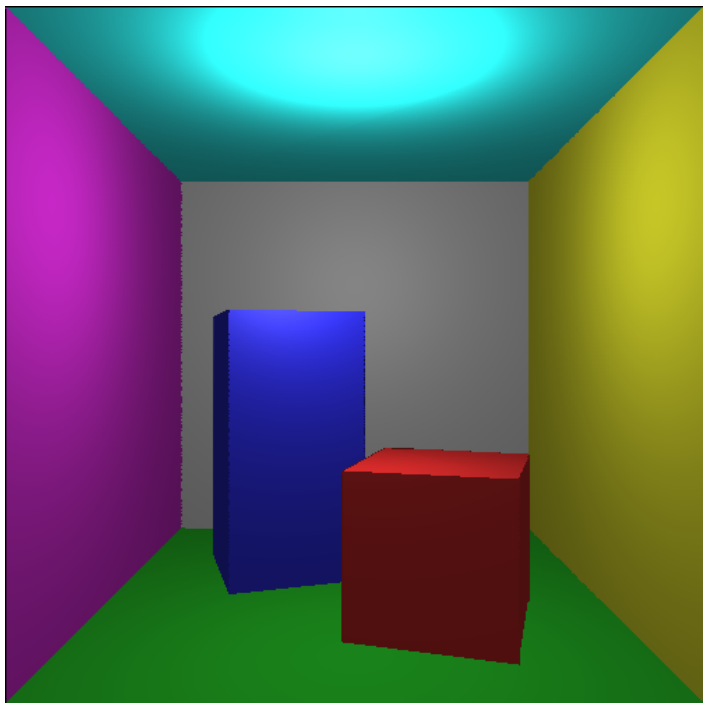

Figure 1 – rasteriser with extensions