CS 425 Lab Exercise – Creating and transforming axes and camera parameters.

Follow instructions below and in "TODO" comments in the code provided (axesExercise.html and axesExercise.js) Submit answers to the given questions via GradeScope, as instructed in class.

1. In axesExercise.js, set values for points[ ], colors[ ], and nPoints. Create 6 points with coordinates (0, 0, 0), (1, 0, 0), (0, 0, 0), (0, 1, 0), (0, 0, 0), and (0, 0, 1), as vec3s. Set the colors of the first two points to red (1.0, 0.0, 0.0), the next two to green (0.0, 1.0, 0.0), and the last two to blue(0.0, 0.0, 1.0), and nPoints to 6.

   Run the program. You should see a yellow rectangle with a horizontal red line and a vertical green line. Why can't you see the blue line?

2. In the render function, create a modelView matrix using lookAt(***eye, at, up***), setting ***eye*** to (0.5, 0.5, 0.5), ***at*** to (0, 0, 0), and ***up*** to (0, 1, 0). Push the modelView matrix to the GPU as a uniform variable. (See comments.) Then in the vertex shader, adjust the code to apply the modelView matrix transformation.

   Run the program again. What do you see now? Why has it changed?

3. Again in the render function, create a camera projection matrix using perspective(***fovy, aspect, near, far***), with a field of view of 60 degrees, a near clipping distance of 0.1, and a far distance of 10.0. The aspect ratio can be calculated from canvas.width / canvas.height in the init( ) function, and passed as a global to render( ). Then in the shader program adjust the code to premultiply the projection matrix times the (modelView times position.)

   You may not detect a great change at this point. You may also want to adjust the camera position, to pull back a little. eye = (1.5, 1.5, 1.5) works well, but you should experiment.

4. Finally create one more transformation matrix, initially as a 4x4 Identity matrix. Pass it to the shader and place it in the multiplication sequence after modelView and before vPosition. This should have no effect at this point. (Verify your program still runs.)

   Then in render after the drawArrays( ) command, add a second drawArrays( ) command, which will draw the axes a second time. In between the two, modify the transformation matrix to be a translation matrix, moving the origin to (0.2, 0.4, 0). You should now see a second set of axes offset from the first.

5. Now apply scaling to the transformation matrix created in step 4, making the translated axes smaller than the original. Note that it still has to be translated, so you will need to create a combination transformation.

6. Finally apply some rotation to the transformation from step 5. Play around with different values and submit an image of your final result, along with copies of your final files.