Durham University

Department of Computing Science

SM Computer Graphics Practical 3

Aim: After the transformation and lighting lectures, it is time to practice with the relevant techniques, helping you get familiar with incorporating lighting effects in a WebGL program and constructing a 3D object that comprises object parts.
Reading: lecture 4 and 5 slides.
Sample program: chair.js. (Carefully study the program code against each feature as follows.)

The sample program shows you how the following features are done:

- Directional lighting [Lec. 5]
- Define data structure of a cube [Lec. 5], having length = 1 on each side.
- Draw x, y, z axes (No lighting is applied) [Lec. 5]
- Draw a cube with interactive rotation allowed (Lighting applied) [Lec. 5].
  Note that the box drawing routine has been moved to the drawbox() function. This allows you to expand the program by reusing drawbox() to draw multiple boxes to construct a complicated object.
- Capture keyboard input [Lec. 5]. The main() program only comprises data structure initialization instructions and a keyboard event capturing routine, which is done through **document.onkeydown**. After a keyboard input is captured, relevant variables are updated and the 3D scene will be redrawn by draw().
- Rendering is mainly done by the vertex shader. The Boolean variable “u_isLighting”, which is set by main() beforedrawing something, provides a control to govern whether lighting is applied or not.
- Two new functions are introduced: pushMatrix() and popMatrix(). This function pair constructs a protected zone, allowing you to locally apply some transformation operations and to draw something within the zone. All these operations will not be propagated to the instructions afterward. (Remember WebGL is a state machine, each operation will inherit all transformation operations that happen before that operation, if no control is applied.) Technically, when you call pushMatrix(), the matrix representing all formerly applied transformation operations will be temporarily push to a stack. Such a saved matrix can be restored when you call popMatrix(). As a test, if you remove the pushMatrix() / popMatrix() pair, from the chair seat, the scaling operation applied on it will propagate to scale the chair seat.

Task 1: Point lighting [Lec. 5]

- Work out the information you need to define point lighting.
- Modify the sample program to support point lighting, using the vertex shader.

Task 2: Fragment shader based point lighting [Lec. 5]

- Now, vertex shader does not process any lighting computation. It only passes vertices in world coordinates, color and updated normal vector of each vertex, to the rasterization processor.
- Implement fragment shader based point lighting.
- Add new hotkeys to your program, manipulating the light position. See whether you see interesting lighting effect.
- If you have time, you may extend the program to add multiple light sources with different colors. The effect of each light source imposing on the 3D model is calculated individually. The final lighting effect on the model will be obtained by adding all individual lighting results.

Task 3: Construct a chair.

- Refresh your memory about the concepts of transformation by studying lecture 4.
- Learn to use the pushMatrix() / popMatrix() functions.
- Modify the sample program, adding four legs to the chair.