**Lecture #5 – OpenGL GLM**

**Homework #1**

Due February 24 @ Midnight

If enough people don't finish, the deadline will be extended

 Problem 2 Explanation

Switch 1

 1

Switch 2

 2

Two lanes. Only one lane should be green and the other lane should be red. Two lanes can NOT be green or red at the same time.

Ie) If lane 1 is green, and cars have been waiting in Lane 2, then when you press the switch 2, Lane 1’s light should turn orange after a delay then turn red. Then, Lane 2 should switch from red to green.

Submit to Sakai one zipped file of…

 Problem 1

OpenGL Program

 Problem 2

Microcontroller Code

Video that explains the finished product & How your circuits work

Will be graded by how much effort you put into the homework

 Extra Credit – By putting extra effort into the homework

 Ie) For problem 1, having a more complex scene like adding a Ferris Wheel

Ie) For problem 2, having a more complex circuitry by adding a switch for emergency service vehicles.

**OpenGL GLM**

A mathematics library that we can utilize to make things move, place object at certain coordinates, and more.

We’ll be utilizing GLM for Transformation & 3D Replication & Camera Movement

void key\_callback()

 Used to notify when a physical key is pressed or released or when it repeats

**5.1 Transformation**

Main Difference is between Lecture 4’s Main.cpp and Lecture 5’s Main.cpp

 Added lines in Lecture 5’s Main.cpp to create Transformation Matrix

glm::mat4 – Creates a 4x4 matrices

* Useful since this will allow us to transform (x,y,z,w) vertices
	+ If w == 1, then the vector (x,y,z,1) is a position in space.
	+ If w == 0, then the vector (x,y,z,0) is a direction.

glm::translate – Offsets the object to a different point

 Expects parameters to be the model matrix, vector where you want object to be translated

glm::rotate – Dynamically rotates an object by using time

 Expects parameters to be the model matrix, rotation angle, rotation axis

 Utilize glm::radians to transform angle float into radians

Then we’ll transfer the matrices to the vertex shader via

 glGetUniformLocation() & glUniformMatrix4fv()

**5.2 Coordinates**

Perspective Projection Matrix

 Utilized to transform vertex coordinates from view space to clip space



We’ve also have glEnable(GL\_DEPTH\_TEST), which allows depth comparison & depth buffer update

We’ve also created ten extra cubes that rotate around. This can be seen in the for() loop in the while() loop.

**5.3 Camera Movements**

fov – Perspective field of view

yaw – One of the three aircraft principal axes



void do\_movement()

 Utilized for camera controls with keyboard

Void scroll\_callback()

 Utilized to notify when user scrolls, whether with a mouse wheel or touchpad gesture