Computer Graphics

Niamh Lawlor

11368226

Lab 1 Report

14 October 2014

Lab Aim : To create a transformation matrix to move the triangle around the screen.

Keyboard Control: This is done by the calling GLFW_PRESS with the key that you want to use for input, checking for the condition to see if it’s true. glfwPollEvents () checks for events that have occured.

```
(GLFW_PRESS == glfwGetKey (window, GLFW_KEY_X))
```

Question 1 : Rotation around the x- y- and z-axis

For rotation around the x-axis my transformation matrix that is multiplied by the vector of points is changed to be :

```
| 1     0        0      0 | | x |          matrix[5] = cos(theta);
| 0     cosQ -sinQ      0 | | y |          matrix[6] = sin(theta);
| 0     sinQ    cosQ    0 | | z |          matrix[9] = -sin(theta);
| 0     0        0      1 | | 1 |          matrix[10] = costheta);
                                          theta+=0.025;
```

Theta is updated on every loop to constantly keep the triangle rotating. Without this the triangle would appear static on the screen rotated by the amount of the constant value theta. This same thing is done for rotation around the Y axis with the corresponding values of the translation matrix written below.

```
| cosQ  0        sinQ   0 | | x |          matrix[0] = cos(theta);
| 0     0        0      0 | | y |          matrix[2] = sin(theta);
| -sinQ 0        cosQ   0 | | z |          matrix[8] = -sin(theta);
| 0     0        0      1 | | 1 |          matrix[10] = cos(theta);
                                           theta+=0.025;
```

For the Z- axis

```
| cosQ  0      sinQ   0 | | x |               matrix[0] = cos(theta);
| 0     0      0      0 | | y |               matrix[1] = -sin(theta);
| -sinQ 0      cosQ   0 | | z |                matrix[4] = sin(theta);
| 0     0      0      1 | | 1 |                matrix[5] = cos(theta);
                                               theta+=0.025;
```

Translation in the x- y- and z- direction

This is done by changing the value sof the of the matrix elements that correspond to the x,y,z values. For the x direction the corresponding matrix is:

```
| 1     0      0      0 | | x |               matrix[12] -= 0.025f; //move left
| 0     1      0      0 | | y |              matrix[12] += 0.025f; //move right
| 0     0      1      0 | | z |
| 0.025 0      0      1 | | 1 |
```

Y- direction

```
| 1     0      0      0 | | x |               matrix[13] -= 0.025f; //move down
| 0     1      0      0 | | y |              matrix[13] += 0.025f; //move up
| 0     0      1      0 | | z |
| 0     0.025  0      1 | | 1 |
```

Z direction

```
| 1     0      0      0 | | x |               matrix[14] -= 0.025f; //move forward
| 0     1      0      0 | | y |              matrix[14] += 0.025f; //move back
| 0     0      1      0 | | z |
| 0     0      0.025  1 | | 1 |
```

Uniform and non-uniform Scaling

```
| 1.S   0      0      0 | | x |              matrix[0] -= 0.005f;
| 0     1.S    0      0 | | y |              matrix[5] -= 0.005f;
| 0     0      1.S    0 | | z |
| 0     0      0.0    1 | | 1 |
```

For uniform scaling assign S to be the same value for x,y,z. For non-uniform scaling assign the S values for each element to be different.

Combined Transformations

```
| 1.S    0      0      0 | | x |            matrix[0] -= 0.005f;
| 0      1.S    0      0 | | y |            matrix[5] -= 0.005f;
| 0      0      1.S    0 | | z |            matrix[12] -= 0.025f; //move left
| 0.25   0.25          1 | | 1 |            matrix[13] -= 0.025f; //move down
```

This combined both scaling and translating. It scaled the triangle by S while moving down and left.

Multiple triangles in the scene

This is done by adding three new points to the points array which are then sent through the same VBO. The transformations are then preformed on both triangles.

Key For User Input :

Right-arrow : translation in x direction. (right)
Left-arrow : translation in x direction. (left)
Up- arrow : translation in y direction. (up)
Down- arrow : translation in y direction. (down)
Key U : Non-uniform scaling (make bigger)
Key J : Non-uniform scaling (make smaller)
Key E : Uniform scaling (make bigger)
Key D : Uniform scaling (make smaller)
Key C : Combined transformations (make bigger)
Key V : Combined transformations (make smaller)
Key X: Rotation around X-axis
Key Y: Rotation around Y-axis
Key Z: Rotation around Z-axis