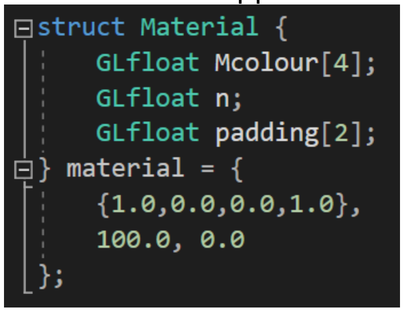
Assignment 2: OpenGL Lighting

Changes in code

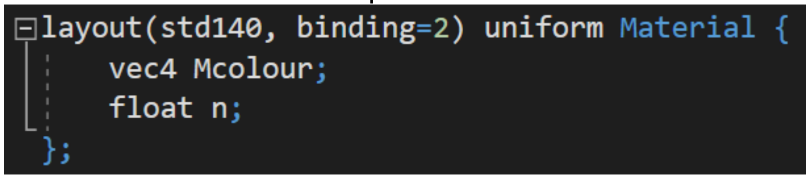
My solution uses `Example6a` as a base with `example6c.vs` and `example6d.fs` as the vertex and fragment shaders, respectively. The main change in the code is the addition of the uniform block `material`, which contains the material properties of the object. This uniform block is added to the fragment shader and `main.cpp` as follows:

`main.cpp`



```
struct Material {
    GLfloat Mcolour[4];
    GLfloat n;
    GLfloat padding[2];
} material = {
    {1.0,0.0,0.0,1.0},
    100.0, 0.0
};
```

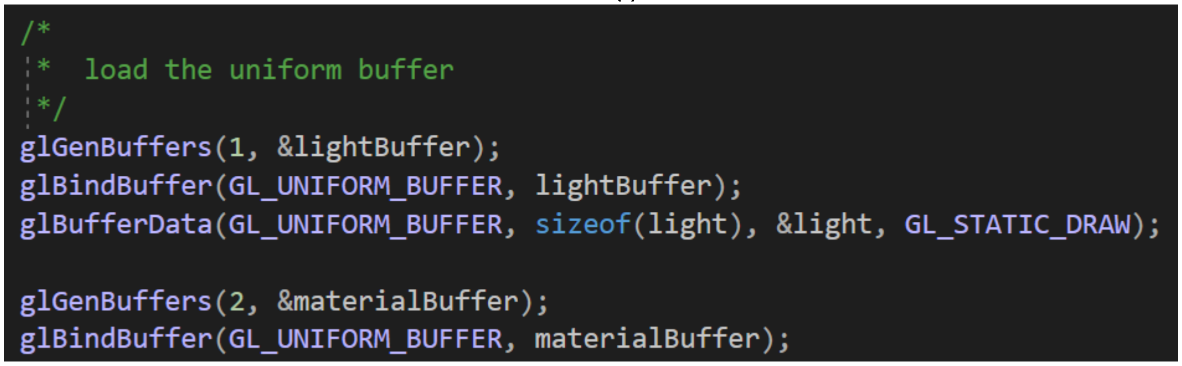
`example6d.fs`



```
layout(std140, binding=2) uniform Material {
    vec4 Mcolour;
    float n;
};
```

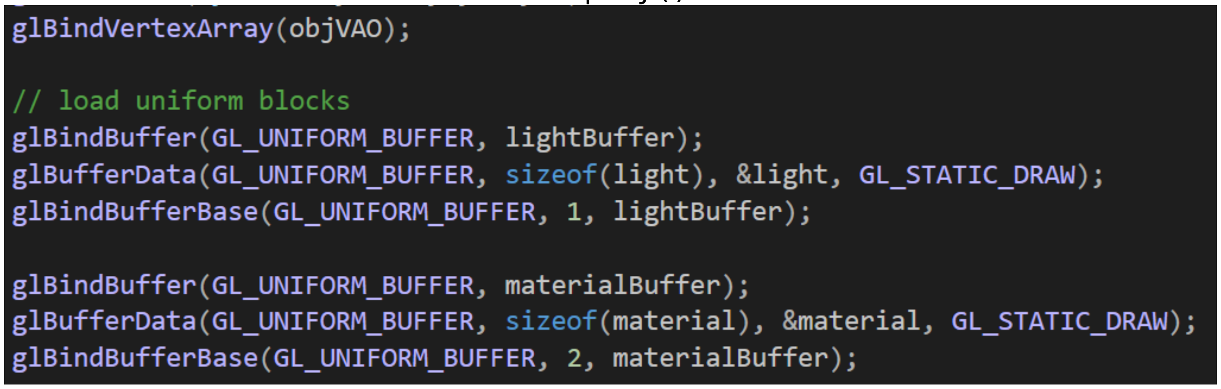
Next, I initialize a uniform buffer called `materialBuffer`. To use this buffer, it needs to be loaded in the same way as the `lightBuffer`. This is done in the `init` and `display` functions:

`init()`



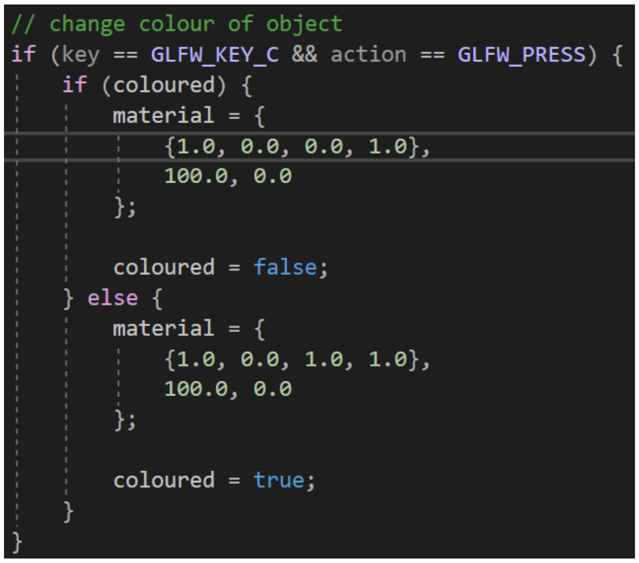
```
/*
 *  load the uniform buffer
 */
glGenBuffers(1, &lightBuffer);
glBindBuffer(GL_UNIFORM_BUFFER, lightBuffer);
glBufferData(GL_UNIFORM_BUFFER, sizeof(light), &light, GL_STATIC_DRAW);

glGenBuffers(2, &materialBuffer);
glBindBuffer(GL_UNIFORM_BUFFER, materialBuffer);
```

`display()`



```
glBindVertexArray(objVAO);

// load uniform blocks
glBindBuffer(GL_UNIFORM_BUFFER, lightBuffer);
glBufferData(GL_UNIFORM_BUFFER, sizeof(light), &light, GL_STATIC_DRAW);
glBindBufferBase(GL_UNIFORM_BUFFER, 1, lightBuffer);

glBindBuffer(GL_UNIFORM_BUFFER, materialBuffer);
glBufferData(GL_UNIFORM_BUFFER, sizeof(material), &material, GL_STATIC_DRAW);
glBindBufferBase(GL_UNIFORM_BUFFER, 2, materialBuffer);
```

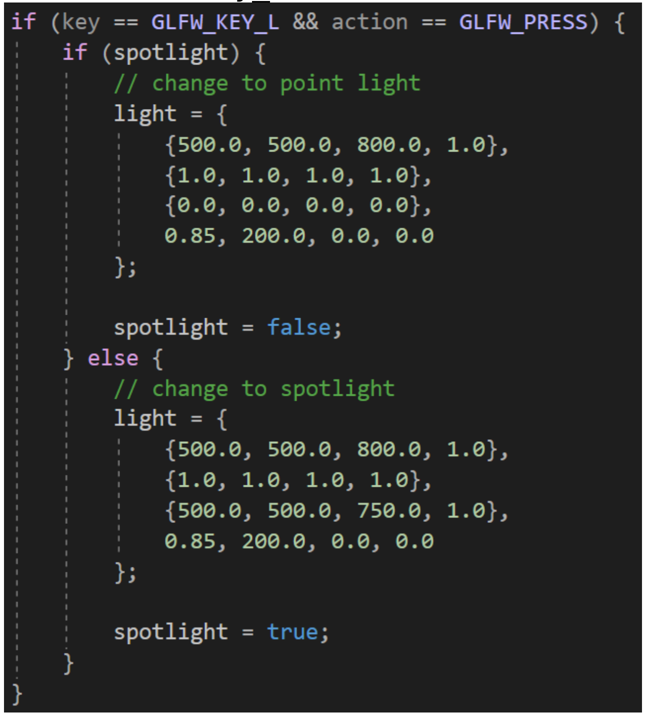
Initially, the colour of the object is red. Toggle between two colours when `C` is pressed. Create a Boolean value, `coloured`, which signifies if the object's color has changed. If `false` then change the value of `Mcolour` in the `material` structure, and revert `Mcolour` back to its original value if `true`. This is all done in the `key_callback` function.



```
// change colour of object
if (key == GLFW_KEY_C && action == GLFW_PRESS) {
    if (coloured) {
        material = {
            {1.0, 0.0, 0.0, 1.0},
            100.0, 0.0
        };

        coloured = false;
    } else {
        material = {
            {1.0, 0.0, 1.0, 1.0},
            100.0, 0.0
        };

        coloured = true;
    }
}
```

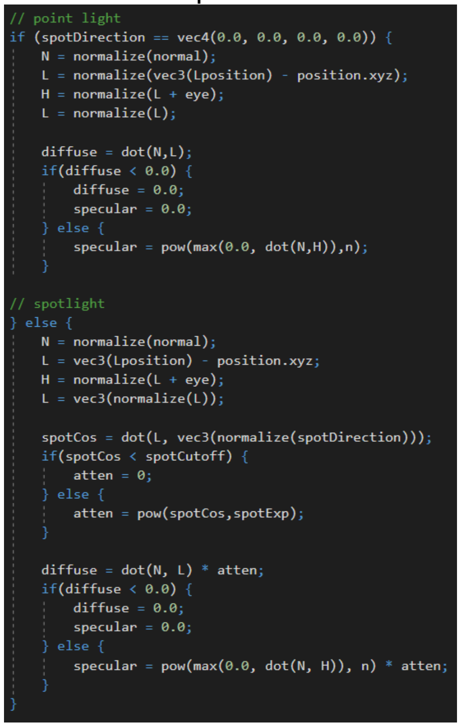
Initially, the object has a spotlight. Toggle between spotlight and point light when `L` is pressed. Create a Boolean value, `spotlight`, which signifies if we are using a spotlight. If `false`, then change the spotlight into a point light, and revert back to the spotlight if `true`. So, the values of the `light` structure are being changed. In order to change the lighting, we make changes in both the `key_callback` function and in the fragment shader.

`key_callback()`



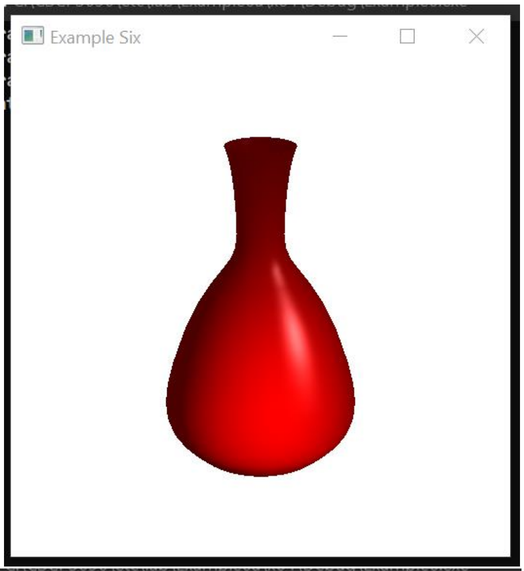
```
if (key == GLFW_KEY_L && action == GLFW_PRESS) {
    if (spotlight) {
        // change to point light
        light = {
            {500.0, 500.0, 800.0, 1.0},
            {1.0, 1.0, 1.0, 1.0},
            {0.0, 0.0, 0.0, 0.0},
            0.85, 200.0, 0.0, 0.0
        };

        spotlight = false;
    } else {
        // change to spotlight
        light = {
            {500.0, 500.0, 800.0, 1.0},
            {1.0, 1.0, 1.0, 1.0},
            {500.0, 500.0, 750.0, 1.0},
            0.85, 200.0, 0.0, 0.0
        };

        spotlight = true;
    }
}
```

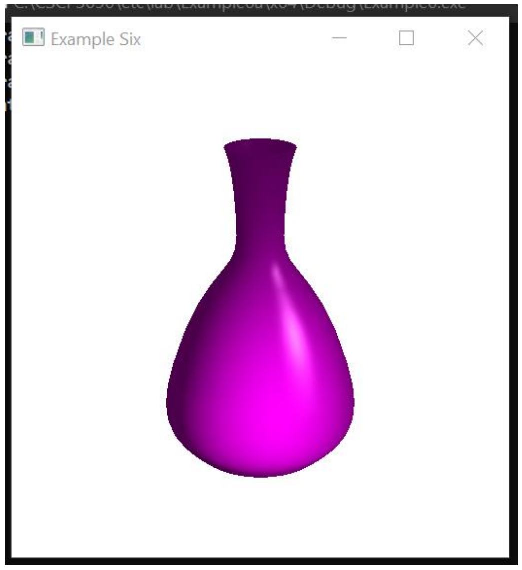
`example6d.fs`

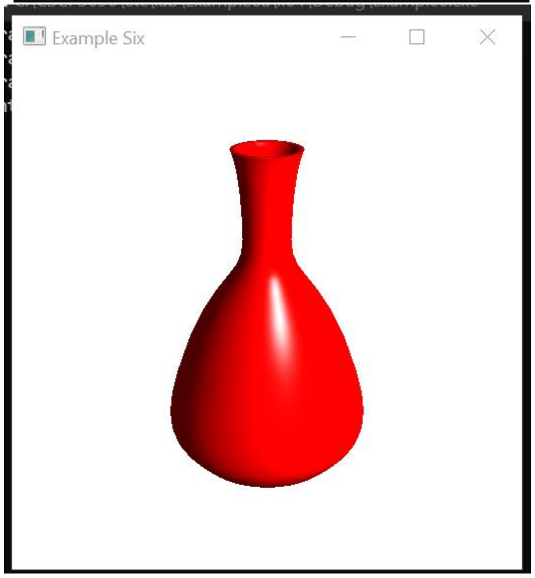


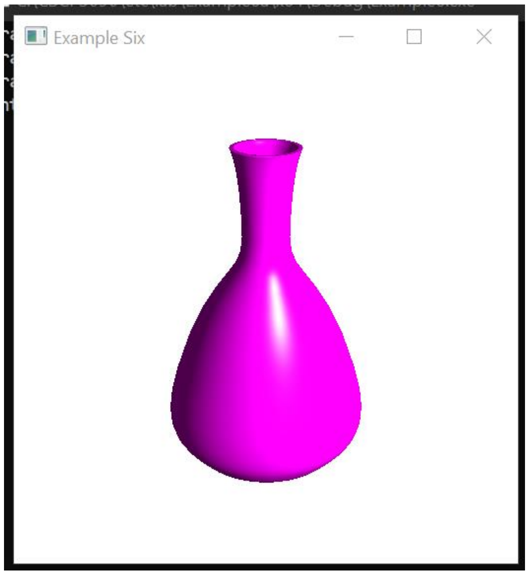
```
// point light
if (spotDirection == vec4(0.0, 0.0, 0.0, 0.0)) {
    N = normalize(normal);
    L = normalize(vec3(Lposition) - position.xyz);
    H = normalize(L + eye);
    L = normalize(L);

    diffuse = dot(N,L);
    if(diffuse < 0.0) {
        diffuse = 0.0;
        specular = 0.0;
    } else {
        specular = pow(max(0.0, dot(N,H)),n);
    }

// spotlight
} else {
    N = normalize(normal);
    L = vec3(Lposition) - position.xyz;
    H = normalize(L + eye);
    L = vec3(normalize(L));

    spotCos = dot(L, vec3(normalize(spotDirection)));
    if(spotCos < spotCutoff) {
        atten = 0;
    } else {
        atten = pow(spotCos,spotExp);
    }

    diffuse = dot(N, L) * atten;
    if(diffuse < 0.0) {
        diffuse = 0.0;
        specular = 0.0;
    } else {
        specular = pow(max(0.0, dot(N, H)), n) * atten;
    }
}
```

Screenshots


Example Six


Example Six


Example Six


Example Six