Illumination Methods

Knights Who Say Ni Group Project 2:

Christopher McDaniel

Dean Morgan-Brown

Daniel Feuling

Different Illumination Methods

- Ambient
- Diffuse
- Specular
- Phong (combined)

Ambient

- This is the general brightness level for a scene.
- Produces a uniform ambient lighting for all objects.
- Approximates the global diffuse reflections from the various lit objects.
- For monochromatic lighting, the level for ambient light within a scene is given the intensity parameter I_a.
- Reflections from ambient lighting are a simple form of diffuse reflection.

Diffuse

- Diffuse light is scattered, and doesn’t leave sharp shadows.
- Contrasts with ambient lighting, as diffuse lighting is dependant on the direction of incoming light rays

Specular

- Specular lighting, like ambient and diffuse lighting, considers strength of light and direction of light rays.
- Specular differs from the other two in that it also takes the position of the viewer into consideration.

Phong (combined)

- Phong lighting is a combination of ambient, diffuse, and specular lighting.

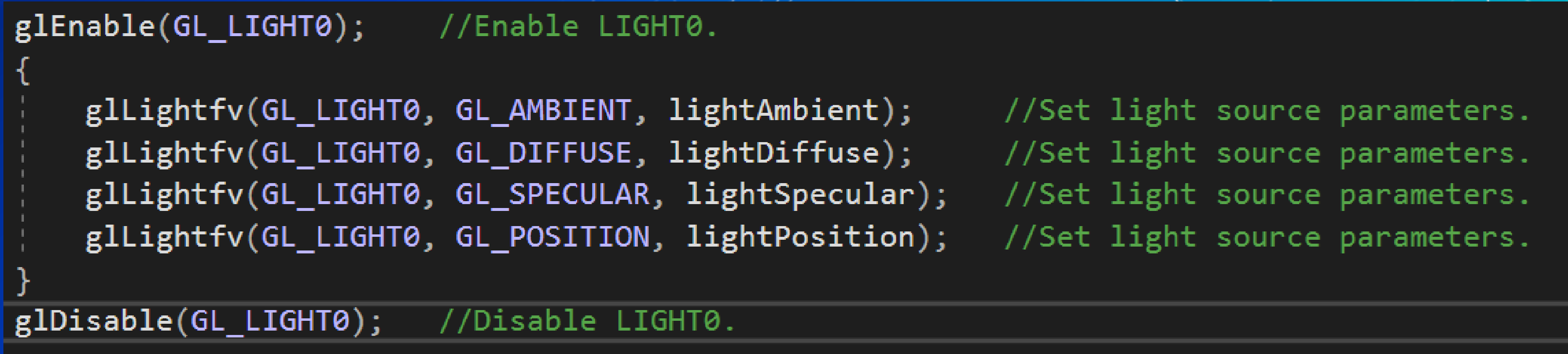
Lighting calculation

In order to enable and disable lighting calculations in the OpenGL render pipeline, one must call glEnable and glDisable, respectively, with the argument GL_LIGHTING.

Ex: glEnable(GL_LIGHTING) and glDisable(GL_LIGHTING)

Light implementation

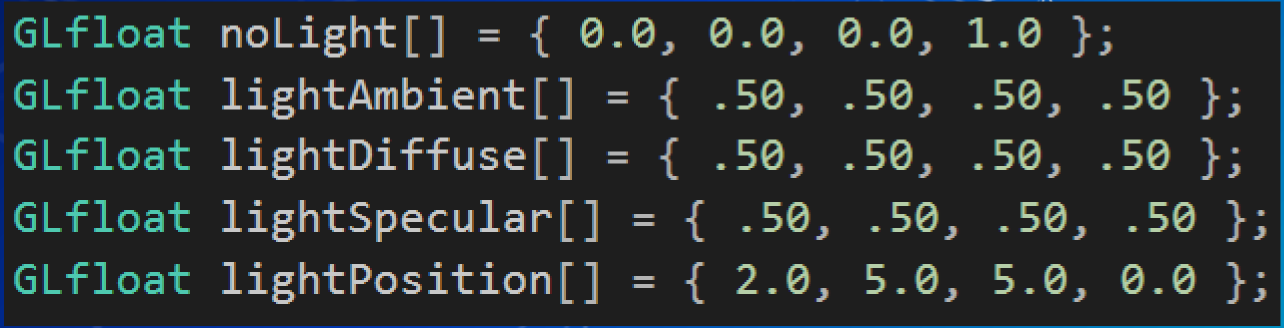
GL_LIGHT0 is a definition of the global light source implementation. I.e. type of light, position, color, attenuation, etc.



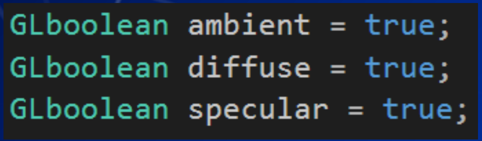
```
glEnable(GL_LIGHT0);    //Enable LIGHT0.
{
    glLightfv(GL_LIGHT0, GL_AMBIENT, lightAmbient);     //Set light source parameters.
    glLightfv(GL_LIGHT0, GL_DIFFUSE, lightDiffuse);     //Set light source parameters.
    glLightfv(GL_LIGHT0, GL_SPECULAR, lightSpecular);   //Set light source parameters.
    glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);   //Set light source parameters.
}
glDisable(GL_LIGHT0);   //Disable LIGHT0.
```

Lighting arrays

These are the parameters used for the glLight() command in the next slide. These arrays set the intensity of the light that will be generated.



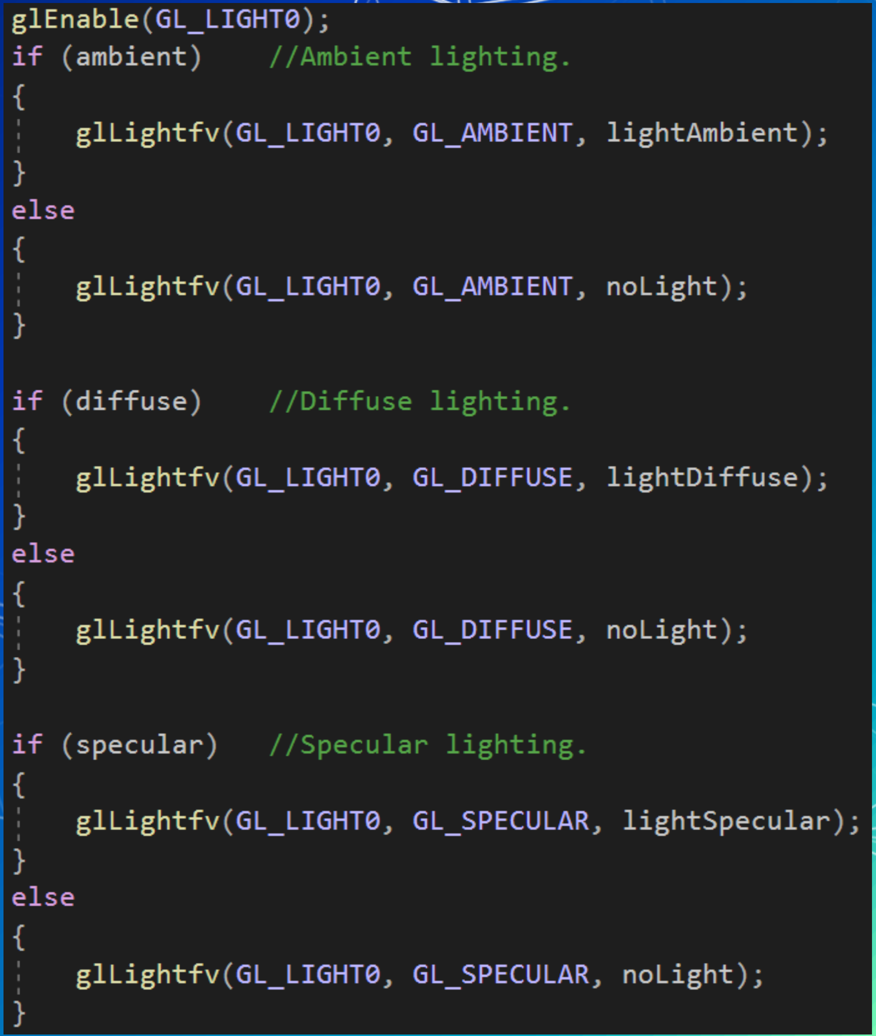
```
GLfloat noLight[] = { 0.0, 0.0, 0.0, 1.0 };
GLfloat lightAmbient[] = { .50, .50, .50, .50 };
GLfloat lightDiffuse[] = { .50, .50, .50, .50 };
GLfloat lightSpecular[] = { .50, .50, .50, .50 };
GLfloat lightPosition[] = { 2.0, 5.0, 5.0, 0.0 };
```



```
GLboolean ambient = true;
GLboolean diffuse = true;
GLboolean specular = true;
```

Lighting

- This is how the lights are set within the scene.
- glLight sets the light source parameters.



```
glEnable(GL_LIGHT0);
if (ambient)    //Ambient lighting.
{
    glLightfv(GL_LIGHT0, GL_AMBIENT, lightAmbient);
}
else
{
    glLightfv(GL_LIGHT0, GL_AMBIENT, noLight);
}

if (diffuse)    //Diffuse lighting.
{
    glLightfv(GL_LIGHT0, GL_DIFFUSE, lightDiffuse);
}
else
{
    glLightfv(GL_LIGHT0, GL_DIFFUSE, noLight);
}

if (specular)   //Specular lighting.
{
    glLightfv(GL_LIGHT0, GL_SPECULAR, lightSpecular);
}
else
{
    glLightfv(GL_LIGHT0, GL_SPECULAR, noLight);
}
```

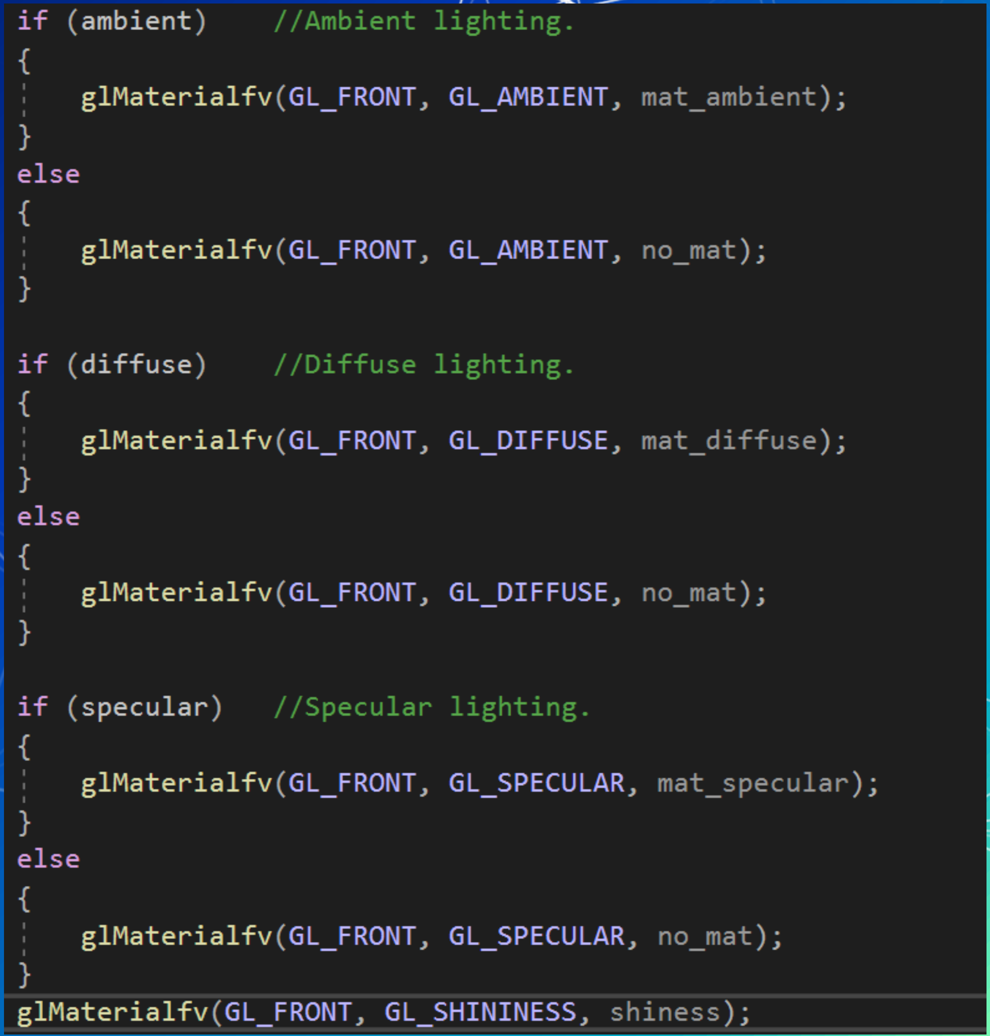
Lighting arrays cont.

These are the parameters used for the glMaterial() command in the next slide. These arrays set the intensity of the light that will be generated on the object itself.

```
GLfloat no_mat[] = { 0.0, 0.0, 0.0, 0.0 };
GLfloat mat_ambient[] = { 0.7, 0.7, 0.7, 1.0 };
GLfloat mat_diffuse[] = { 0.8, 0.8, 0.8, 1.0 };
GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat shiness[] = { 100.0 };
```

Lighting cont.

- This is how the lights are set for the object that is being lit.
- glMaterial specifies the material parameters for the lighting model.



```
if (ambient)    //Ambient lighting.
{
    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
}
else
{
    glMaterialfv(GL_FRONT, GL_AMBIENT, no_mat);
}

if (diffuse)    //Diffuse lighting.
{
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
}
else
{
    glMaterialfv(GL_FRONT, GL_DIFFUSE, no_mat);
}

if (specular)   //Specular lighting.
{
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
}
else
{
    glMaterialfv(GL_FRONT, GL_SPECULAR, no_mat);
}
glMaterialfv(GL_FRONT, GL_SHININESS, shiness);
```

Program Example