Name: Ralenski Doucet
Contents; Lighting

- **How I Explain what 3d Lighting:**

There are 4 types of Lights used in games. They include: Ambient, Point, Directional, and Spot lights.
The normal for a light is the direction of the surface a light is shining on is facing .
The direction of the normal is used to calculate the direction of the light.A normal is usually a normalised unit vector. To Be able to code Lighting you need to understand the formulas.
The elements used in the formula are as follows:
See Parts of the Formula below:
-K refers to the surface material property colours (ambient, diffuse, specular)
– I refers to the light properties (ambient, diffuse, specular)
– N is the surface normal vector
– Lm is the light direction, the Incident ray
– Rm is the light's Reflected ray
– V is a view direction that represents a ray from the surface to the camera
– A is a specular power used to control the sharpness of specular reflection

$$I_p = k_a i_a + \sum_{m \in lights} (k_d (L_m \cdot N) i_d + k_s (R_m \cdot V)^a i_s)$$

To setup lighting you want make sure you us the following GL functions in the defining of the draw function.These are the GL function needed in the draw function.
glUniformMatrix4fv(handle, 1, GL_FALSE, &mvp[0][0]);
glUniform3fv(light Color Handle, 1, & col[0]);
glUniform3fv(light Pos Handle, 1, & directional Light->pos[0]);
glUniform3fv(light Dir Handle, 1, & directional Light->direction[0]);
glUniform3fv(Camera Pos Handle, 1, & view[0]);
glUniform1fv(ambient Co Handle, 1, & a C);
glUniform1fv(diffuse Co Handle, 1, & d C);
glUniform1fv(specular Co Handle, 1, & s C);
All in all this is what the draw function for lighting should look like See image below:

Name: Ralenski Doucet
Contents; Lighting

```
void LightingApplication::draw()
{
    baseShader->bind();
    int handle = baseShader->getUniform("ProjectionViewWorld");
    int lightColorHandle = baseShader->getUniform("lightColor");
    int lightPosHandle = baseShader->getUniform("lightPos");
    int lightDirHandle = baseShader->getUniform("lightDir");
    int CameraPosHandle = baseShader->getUniform("cameraPos");
    int ambientCoHandle = baseShader->getUniform("ambientCo");
    int diffuseCoHandle = baseShader->getUniform("diffuseCo");
    int specularCoHandle = baseShader->getUniform("specularCo");
    glm::mat4 mvp = projection * view * model;

    glUniformMatrix4fv(handle, 1, GL_FALSE, &mvp[0][0]);
    glm::vec3 col = directionalLight->color;
    glUniform3fv(lightColorHandle, 1, &col[0]);
    glUniform3fv(lightPosHandle, 1, &directionalLight->pos[0]);
    glUniform3fv(lightDirHandle, 1, &directionalLight->direction[0]);
    glm::vec3 view = glm::vec3(0, -10, 20);
    glUniform3fv(CameraPosHandle, 1, &view[0]);
    glUniform1fv(ambientCoHandle, 1, &aC);
    glUniform1fv(diffuseCoHandle, 1, &dC);
    glUniform1fv(specularCoHandle, 1, &sC);
    mesh1->render();
    mesh2->render();
    Sphere->render();
    baseShader->unbind();
}
```

- **The difference between Blinn Phong and Phong.**

The only difference between Blinn-Phong and Phong specular reflection is that measurement in the angle between the normal and the halfway vector is compared to the angle between the view direction and the reflection vector. The Blinn-Phong specular exponent will be a bit sharper compared to Phong.

My Implementation of diffuse:

```
int diffuseCoHandle = baseShader->getUniform("diffuseCo");
```

My Implementation of Ambient:

```
int ambientCoHandle = baseShader->getUniform("ambientCo");
```

My Implementation of Specular:

```
int specularCoHandle = baseShader->getUniform("specularCo");
```