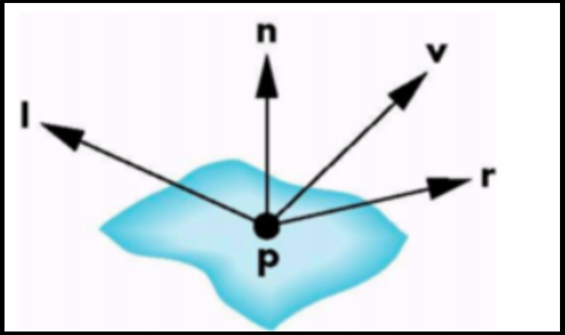
HW3

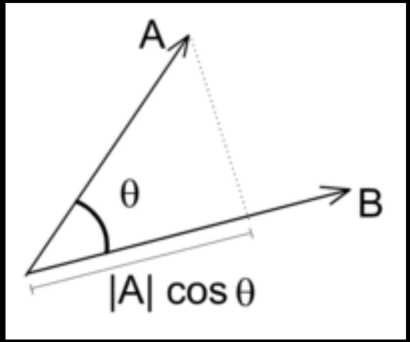
How to determine light intensity

- We can simply use the included angle of the reflection and view vectors.

- L is a vector towards the light source
- V is a vector towards the camera position
- R is a vector which is the reflection of L
- N is a vector which is the normal of the point P



How to determine light intensity



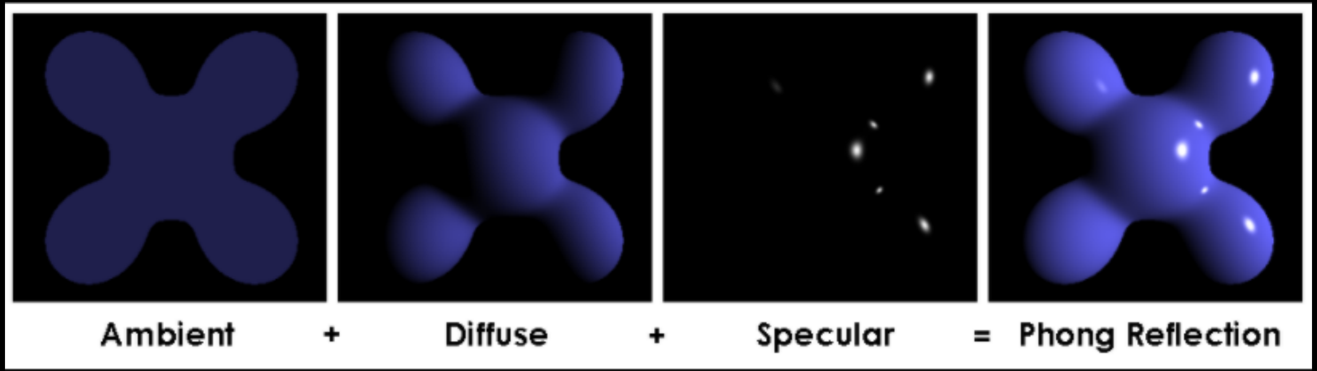
- If two vectors are unit vectors. Then we can get cosθ by doing dot products of the two vectors.

  $A \cdot B = |A||B| \cos \theta$

- The smaller θ is, the larger cos θ is. According to the Phong reflection model, we can determine the light intensity based on cos θ.

- If $\cos \theta < 0$, θ must bigger than 90°. In this case, this position cannot be illuminated.

Phong shading

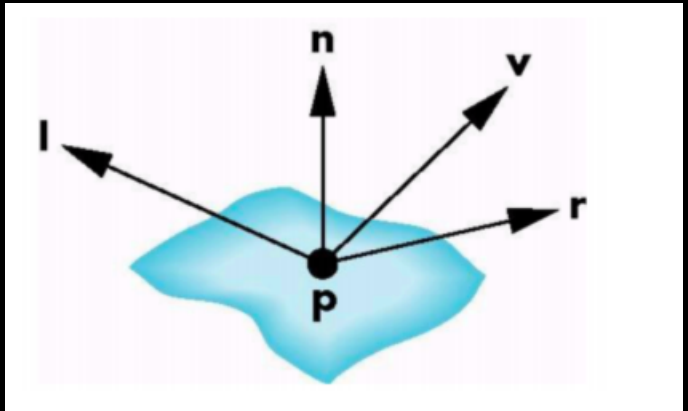


K is the reflectivity of each component of the material

- Parameters of model material:
  1. Ambient reflectivity (Ka) : 1.0 1.0 1.0
  2. Diffuse reflectivity (Kd) : 1.0 1.0 1.0
  3. Specular reflectivity (Ks) : 0.7 0.7 0.7
  4. Gloss (α) : 10.5

L is the intensity of each component of the light.

- Parameters of light:
  1. Ambient intensity (La) : 0.2 0.2 0.2
  2. Diffuse intensity (Ld) : 0.8 0.8 0.8
  3. Specular intensity (Ls) : 0.5 0.5 0.5
  4. Position : (20, 20, 0)



$$I = I_{ambient} + I_{diffuse} + I_{specular}$$
$$= k_a I_a + k_d I_d (\mathbf{l} \cdot \mathbf{n}) + k_s I_s (\mathbf{v} \cdot \mathbf{r})^{\alpha}$$

Phong shading - pseudocode

```
void main()
{
    object_color = texture2D(Texture, texcoord);

    ambient = La * Ka * object_color;
    diffuse = Ld * Kd * object_color * dot(L,N); // must > 0
    specular = Ls * Ks * pow(dot(V,R), gloss);

    color = ambient + diffuse + specular;
}
```

Gouraud shading - pseudocode

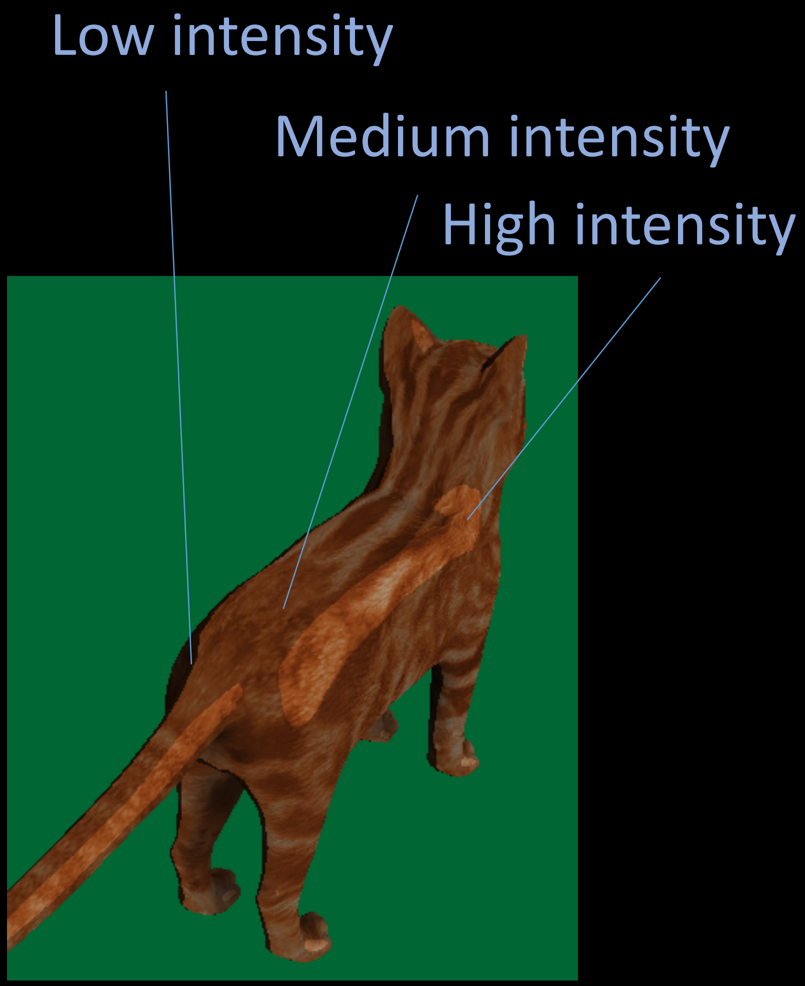
```
void main()
{
    // in vertex shader
    Calculate ambient, diffuse, specular on vertex.

    // in fragment shader
    Render the fragment with the interpolated light intensities.
}
```

Toon shading - pseudocode

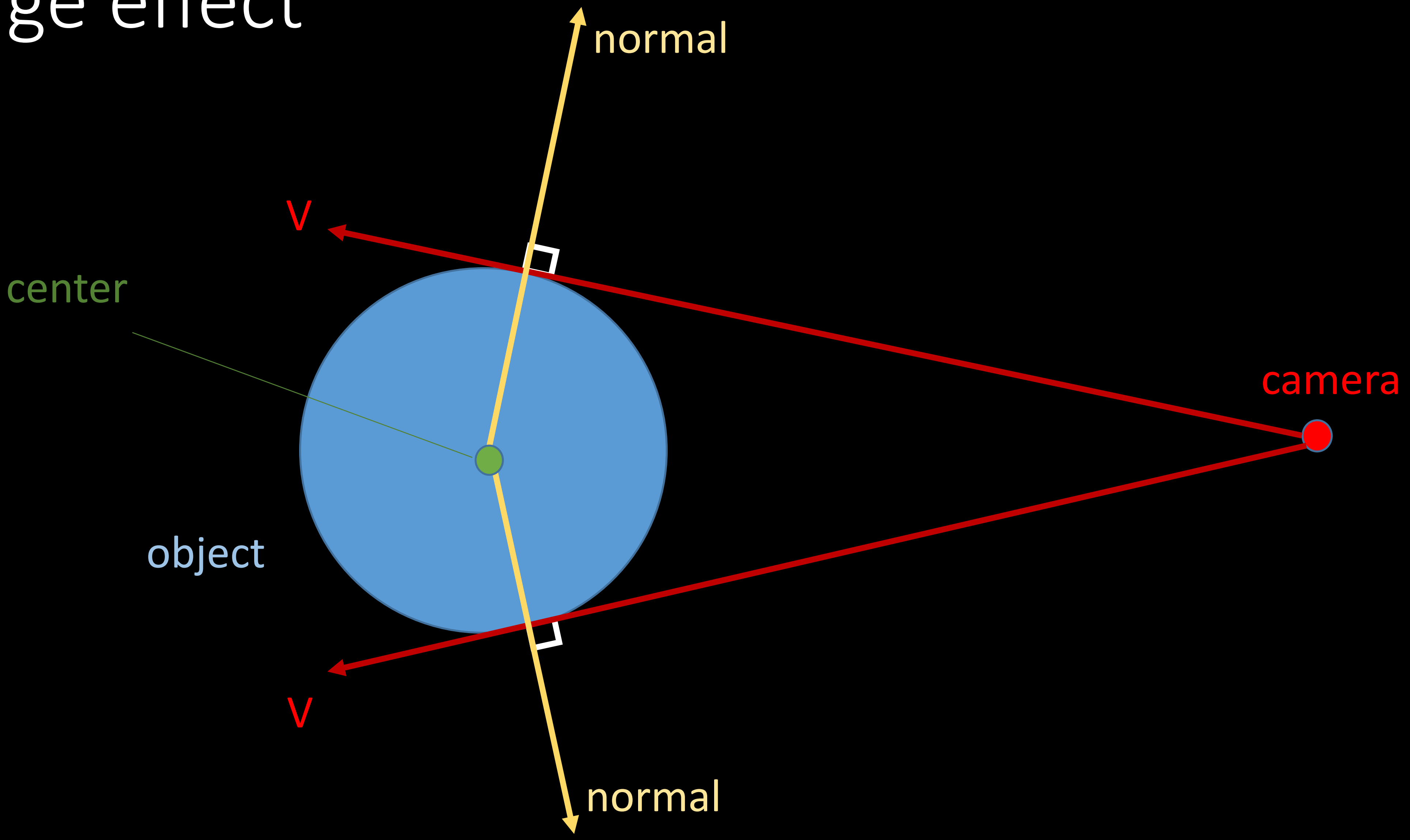
```
void main()
{
    object_color = texture2D(Texture, texcoord);
    Calculate the angle between the light and normal vector

    If not lighted (angle > 90), low intensity
    If strong specular, high intensity
    Else, medium intensity
    // you can decide the intensity and specular threshold
    Color = Kd * object_color * intensity ;
}
```



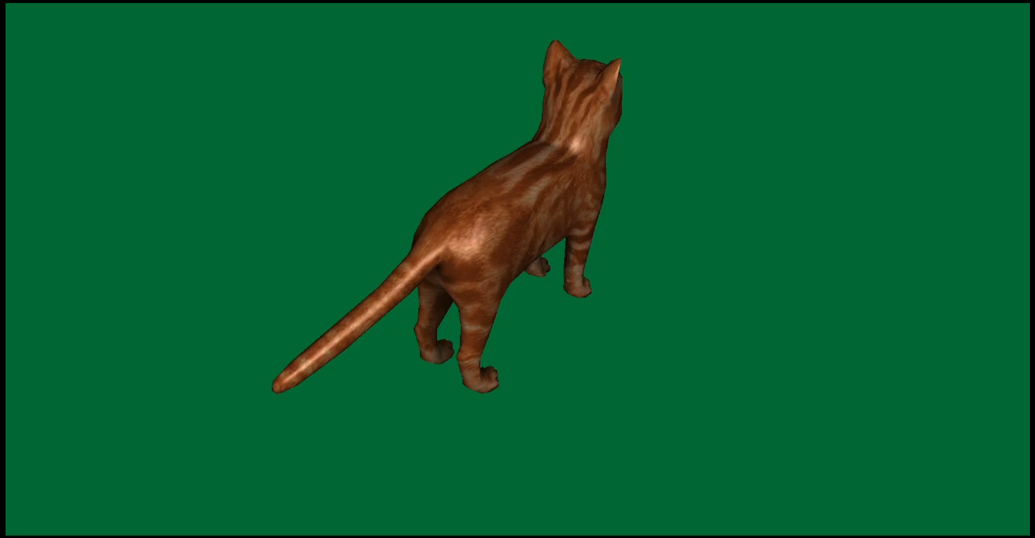
Green background color for display purpose, you can choose your own background color

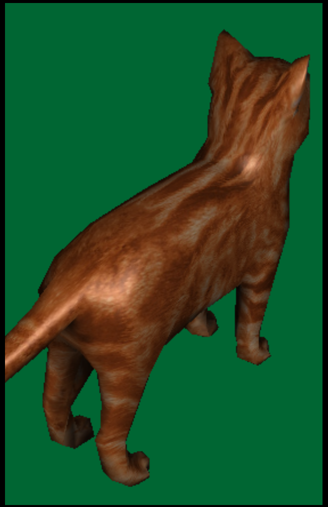
Edge effect

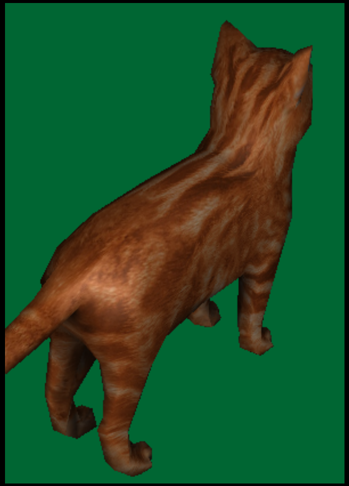


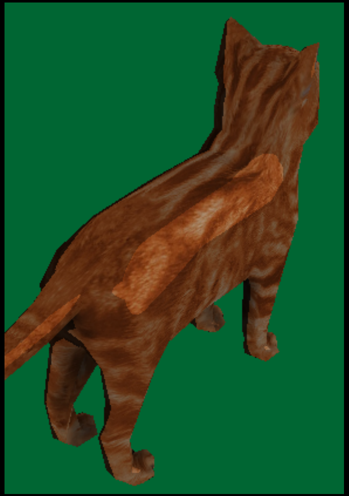
Homework 3

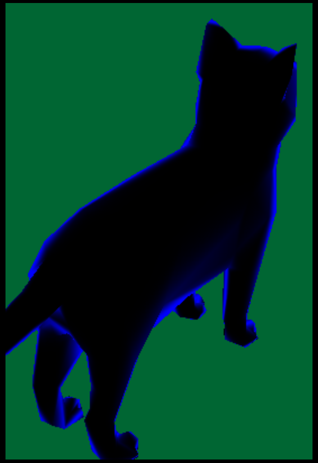
- Goal :
  1. Phong shading
  2. Gouraud shading
  3. Toon shading
  4. Edge effects




Phong shading


Gouraud shading


Toon shading


Edge effects

Homework 3 - score

1. create shaders and programs you need and can switch them correctly.(5%)
2. Create all variable and pass them to shaders through Uniform(5%)
3. Implement **Phong shading** via shader (25%)
4. Implement **Gouraud shading** via shader (20%)
5. Implement **Toon shading** via shader(15%)

   # at least define 3 levels.

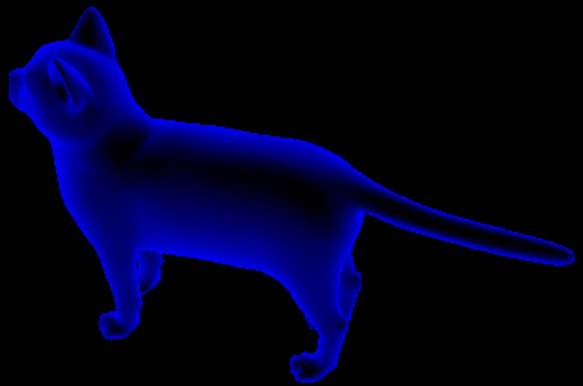
6. Implement **Edge effects** via shader(10%)

   # must clearly see the edge

   # The color of the edge is not specified

7. Report (20%)

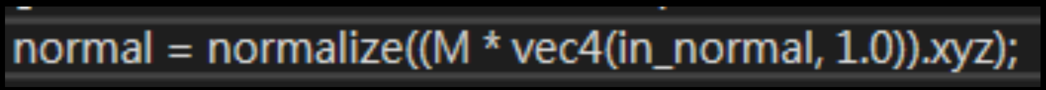
This is not allowed



Reminder

The cat is rotating about y axis 45 degree/second. Create the model transform matrix accordingly. You can get perspective and view matrices from getPerspective() and getView().

The light position, camera position and the model position might not be in the same space. You might need to deal with that.



```
normal = normalize((M * vec4(in_normal, 1.0)).xyz);
```

This will work in this homework, but it is the wrong way to do the space conversion.

Homework 3 (report)

- Please specify your name and student ID in the report.

- Explain how you implement the above shading/effects.
  (ex: how I get the vector L. I do dot(L, N) for what……etc.)

- Describe the problems you met and how you solved them.

Homework 3 - submission

- Deadline: 2022/12/6 23:59:59
- 10% penalty for each week late
- Pack your report and project in a zip file. File name: studentID_hw3.zip

Files to be implemented

- main.cpp
- shaders/Phong.vert
- shaders/Phong.frag
- shaders/Toon.vert
- shaders/Toon.frag
- shaders/Edge.vert
- shaders/Edge.frag
- shaders/Gouraud.vert
- shaders/Gouraud.frag