計算機圖學HW3 Report

determine light intensity

0611031 資工10 謝至恆

Notes

- 因為我覺得小視窗轉得有點快
  所以把　　angle += 0.5
  改成　　angle += 0.1
  比較好觀察

Edge effects

- 使用的參數: position , normal ,M ,V ,P （Transpose(inverse(M)) , BlueColor , BlackColor, vertex_to_camera
- 修正normal　　RGB(0,0,1) RBG(0,0,0) V向量
- 偷懶將normal和V向量轉到View space傳進fragment shader

vertex shader

```
void main() {
  gl_Position = P * V * M * vec4(in_position, 1.0);
  uv = texcoord;

  bcolor = BlueColor;
  blcolor = BlackColor;
  normal = ( V * Transpose_Inverse_M * vec4(in_normal,0.0)).xyz;
  vertex_to_camera = -1 * ( V * M * vec4(in_position, 1.0)).xyz;
```

Fragment shader (Edge effects)

- 將normal , V向量 normalize
  算出積，並用max函數使<0
  的數值直接轉成0，數字越小代表
  V與normal越接近90度，再用一個<0.3才
  顯示的過濾器來使外框看起來更明顯，
  另外發現沒把未定義的點給顏色有時候
  會有問題所以寫了一個else把其他點顏色
  塗黑
- dot(vc,n)
  V與normal越接近90度，藍色輪廓越明顯

```
void main()
{
    vec3 n = normalize(normal);
    vec3 vc = normalize(vertex_to_camera);

    float CameraPercentage;
    CameraPercentage = max(dot(vc,n),0);
    if(CameraPercentage < 0.3){
        color = bcolor * (1 - CameraPercentage);
    }else{
        color = blcolor;
    }
}
```

Problems I met

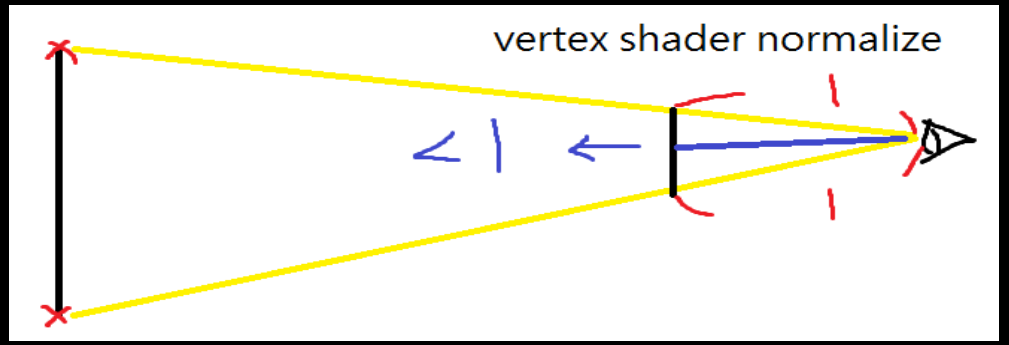
- space conversion

  剛開始寫的時候沒有注意到要轉成相同的space在做運算，跑出來的效果不是整片藍就是整片黑，重新檢查Code和複習MVP矩陣後發現問題

  然後因為一開始一直跑不出來就偷懶轉到View space(就不用管CamPos)

- Vertex shader normalize

  一開始跑出來的結果雖然邊緣有比較明顯但身體還是有一片數值較小的區域，發現在vertex shader裡面先算V會使得到fragment shader前內插出的position長度不為1，內積出來的結果長度會變小，應要到fragment shader再做normalize才會使每個點的V長度都為1，這時候算出的內積才正確的



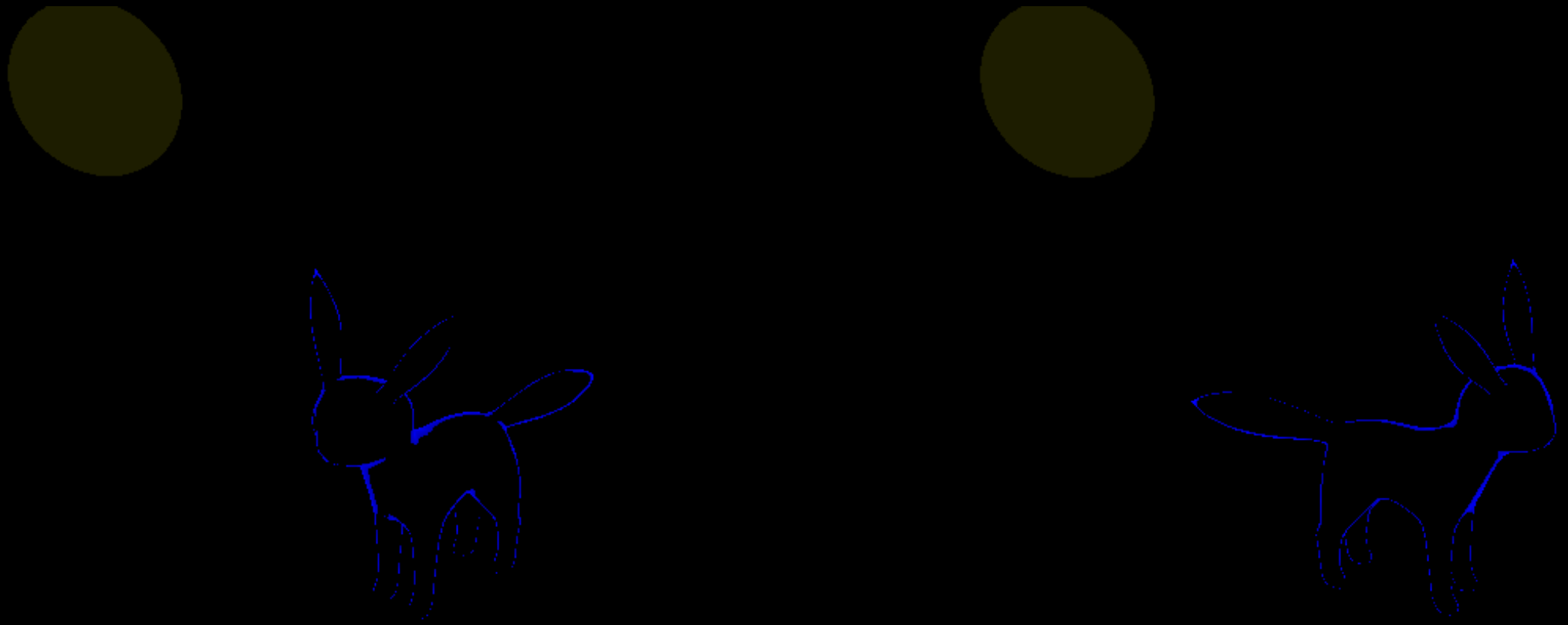
- Direction of V vector

  最後遇到的問題，我一開始是把V算成Viewer到vertex的向量，此時算出來的V拿去做內積正面是負的，背面是正的，而我以為算出來要顯示正的，所以我的畫面又變成黑的了，後來誤打誤撞發現color 設為 color = vec4(V,0.0f)結果全黑，但color = vec(-V,0.0f)卻有顏色才了解到內積方向的問題，將向量方向轉正確後，最後輸出正確的樣子

- retrieve inverse

  參考助教在討論區的文章，retrieve inverse = Transpose(inverse(M))

Edge effects result



Toon shading

- 使用的參數: position , normal ,M ,V ,P （Transpose(inverse(M)) , vertex_to_light , Kd
- 修正normal　　L向量
- 將 L 和 normal 轉到world space 傳給fragment shader

vertex shader

```
void main() {
  gl_Position = P * V * M * vec4(in_position, 1.0);
  uv= texcoord;

  Kd = Diffuse_reflectivity;
  normal = (Transpose_Inverse_M * vec4(in_normal,0.0)).xyz;
  vertex_to_light =  LightPos - ( M * vec4(in_position, 1.0)).xyz;
}
```

Fragment shader (Toon shading)

- 將normal , L向量(vl) normalize
  算出兩者的積，並用max函數使<0
  的數值直接轉成0，數字越大代表
  L與normal越接近0度，光應越亮
  參照助教給的pseudocode 寫出來
  結果就差不多了
- dot(vl,n)
  L與normal正好方向越一致，光應越亮

```
void main()
{
    vec4 object_color = texture2D(texture, uv);
    vec3 n = normalize(normal);
    vec3 vl = normalize(vertex_to_light);

    float intensity = 1;
    float level = max(dot(vl,n),0);

    if(level > 0.95) intensity = 1;
    else if(level>0.75) intensity = 0.8;
    else if(level>0.50) intensity = 0.6;
    else if(level>0.25) intensity = 0.4;
    else intensity = 0.2;


    color = Kd * object_color * intensity ;
}
```

Problems I met

- 因為Edge effect把該踩的雷都踩得差不多了，所以Toon shading意外地順利完成，沒出什麼問題

Toon shading result

Phong shading

- 使用的參數: position , normal ,M ,V ,P （Transpose(inverse(M)) , vertex_to_light , vertex_to_camera
- 修正normal　　L向量　　V向量
- 各種係數 Ka ,Kd ,Ks ,La ,Ld ,Ls ,gloss
- 將 V 和 L 和 normal 轉到world space 計算出結果傳給fragment shader

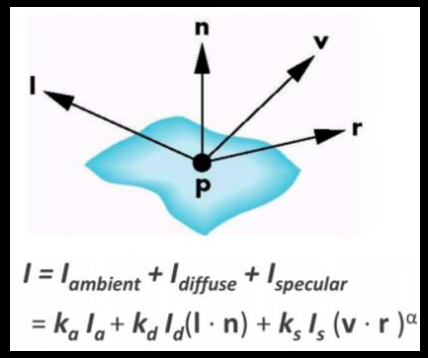
vertex shader

```
void main() {
  gl_Position = P * V * M * vec4(in_position, 1.0);
  uv= texcoord;

  Ka = Ambient_reflectivity;
  Kd = Diffuse_reflectivity;
  Ks = Specular_reflectivity;
  La = Ambient_intensity;
  Ld = Diffuse_intensity;
  Ls = Specular_intensity;
  a  =  gloss;
  normal = (Transpose_Inverse_M * vec4(in_normal,0.0f)).xyz;
  vertex_to_light =  LightPos - ( M * vec4(in_position, 1.0f)).xyz;

  vertex_to_camera = CamPos - ( M * vec4(in_position, 1.0f)).xyz;
}
```

Fragment shader (Phong shading)

- 將normal , L向量(vl),V向量(vc)做 normalize
  將所需要的r向量按照公式算出來
  再帶入Ambient, Diffuse, Specular 按照
  助教給的pseudocode計算就可以得到結果
- dot(vl,n)
  L與normal正好方向越一致，光應越亮
- dot(vc,reflect_vector)
  Viewer 和 reflect light
  方向越一致，內積越大，也就產生反射
  光的感覺

$$\mathbf{r} = 2\,(\mathbf{l} \cdot \mathbf{n})\,\mathbf{n} - \mathbf{l}$$




```
void main()
{
    vec4 object_color = texture2D(texture, uv);
    vec3 n = normalize(normal);
    vec3 vl = normalize(vertex_to_light);
    vec3 vc = normalize(vertex_to_camera);

    vec3 reflect_vector = ( 2 * max(dot(vl,n),0) * n) - vl;

    vec4 Ambient = Ka * La * object_color;
    vec4 Diffuse = Kd * Ld  * object_color * max(dot(vl,n),0);
    vec4 Specular = Ks * Ls  * pow(max(dot(vc,reflect_vector),0),a);

    color = Ambient + Diffuse + Specular;
}
```

Problems I met

- specular = Ls * Ks * pow(dot(V,R), gloss);

  一開始很雞婆的把specular還乘上了object color結果就是跑出來的反射光看起來很暗，然後又一直找不到bug，後來想說試試看跟著pseudocode照打就完成了

- Max(dot())

  原本前面兩個效果dot沒有判斷<0的情況也沒有問題，因為光都照不到所以不會顯示出差別，但phong shading在沒有把<0的dot值改成0的話結果就會怪怪的，像是會有一圈白色的外框，後來加上max函數就沒有問題了

Phong shading result