HW3_report_109550017_黃品云

在寫以下效果前，我創建了需要的shader, programs還有uniform變數，並正確地使用他們。

1. Implement Phong shading(按鍵1開啟)

在main當中傳入mateial的Ka, Kd, Ks, gloss及light的La, Ld, Ls與lightPos後，我在vertex shader用Uniform的M(model matrix), V(view matrix), P(perspective matrix)先使每個vertex作相對應的轉換，並把結果以gl_Position輸出，除此之外，也將vertex乘上V*M的結果存入worldPos，紀錄vertex在world coordinate上的位置，接著，我取出worldPos的x, y, z，轉成3維的vertPos，vertPos將用於fragment shader計算物體與觀察者間的向量使用。再來，我處理了材質的座標，將從layout得到的aTexCoord賦值至texCoord。最後，我對該vertex上的normal用V*M做轉換，避免向量因乘上V*M而失真，並把mormalize的結果放入normal變數中。先前得到的texCoord, vertPos, normal會被vertex shader傳到fragment shader中運用。

進到fragment shader後，用先前得來的normal, vertPos搭配lightPos計算向量n(垂直fragment), l(指向光源), r(反射方向), v(指向viewer)。n由normalize過的normal得來。l由lightPos與vertPos相減得來，並做normalization。r由l與n的反射角推算而來，可使用reflect函式計算，並做normalization。最後v由vertPos的反向得來，並做normalization。

得到向量後，利用透過Uniform傳入的ourTexture與textCoord得到fragment的顏色，並存入color中，並開始計算ambient, diffuse與specular。ambient以La*Ka*color得來，diffuse由Ld * Kd * color * dot(l, n)，其中l與n的內積需要大於0，否則設為0，dot(l, n)為計算光源與正交向量的內積，fragment愈是面向觀察者，diffuse光的值會愈大。specular由Ls * Ks * pow(dot(r, v), gloss)得來，dot(r, v)為計算反射光與觀察者的內積，觀察者愈是面向反射光，得到的specular光的值會愈大，此外，我限制specular需在dot(r, v)均大於0時才會計算Ls * Ks * pow(dot(r, v), gloss)，否則為0向量。

得到ambient, diffuse與specular後，將三種光相加，再利用FragColor輸出。

2. Implement Gouraud shading(按鍵2開啟)

與Phong shading類似，不同處在於計算ambient, diffuse與specular是在vertex shader當中做的，而fragment shader只負責將接收到的該fragment顏色輸出。

我在vertex shader用Uniform的M(model matrix), V(view matrix), P(perspective matrix)先使每個vertex作相對應的轉換，並把結果以gl_Position輸出，除此之外，也將vertex乘上V*M的結果存入worldPos，紀錄vertex在world coordinate上的位置。再來，我處理了材質的座標，將從layout得到的aTexCoord賦值至texCoord。最後我對該vertex上的normal用V*M做轉換，避免向量因乘上V*M而失真，並把normalize的結果放入normal變數中。接著用先前得來的normal, worldPos搭配lightPos計算向量n(垂直fragment), l(指向光源), r(反射方向), v(指向viewer)。n由normalize過的normal得來。l由lightPos與vertPos相減得來，並做normalization，其中vertPos為worldPos的x, y, z。r由l與n的反射角推算而來，可使用reflect函式計算，並做normalization。最後v由vertPos的反向得來，並做normalization。

得到向量後，開始計算每個vertex的ambient, diffuse與specular。ambient以La*Ka得來，diffuse由Ld * Kd * dot(l, n)，其中l與n的內積需要大於0，否則設為0，dot(l, n)為計算光源與正交向量的內積，vertex愈是面向觀察者，diffuse光的值會愈大。specular由Ls * Ks * pow(dot(r, v), gloss)得來，dot(r, v)為計算反射光與觀察者的內積，觀察者愈是面向反射光，得到的specular光的值會愈大，此外，我限制specular需在dot(r, v)均大於0時才會計算Ls * Ks * pow(dot(r, v), gloss)，否則為0向量。得到ambient, diffuse與specular後，將三種光傳給fragment shader。

Fragment shader得到每個fragment的光後，會再乘上Uniform傳入的ourTexture與textCoord得到fragment的texture顏色，最後再以FragColor = ambient + diffuse + specular輸出。

3. Implement Toon shading(按鍵3開啟)

因為toon shading只須在意fragment與光源之間的光強度關係，我在main當中只傳入mateial的gloss及light的lightPos。

我在vertex shader用Uniform的M(model matrix), V(view matrix), P(perspective matrix)先使每個vertex作相對應的轉換，並把結果以gl_Position輸出，除此之外，也將vertex乘上V*M的結果存入worldPos，紀錄vertex在world coordinate上的位置，接著，我取出worldPos的x, y, z，轉成3維的vertPos，vertPos將用於fragment shader計算物體與觀察間的向量使用。再來，我處理了材質的座標，將從layout得到的aTexCoord賦值至texCoord。最後我對該vertex上的normal用V*M做轉換，避免向量因乘上V*M而失真，並把mormalize的結果放入normal變數中。先前得到的texCoord, worldPos, normal會被vertex shader傳到fragment shader中運用。

進到fragment shader後，用先前得來的normal, worldPos搭配lightPos計算向量n(垂直fragment) 與 l(指向光源)。

得到向量後，設intensity以記錄 l 與 n 的內積大小，我設了3個threshold，分別是intensity大於0.7、intensity介於0.7與0.3、intensity小於0.3。intensity愈高，所需顯示的顏色會愈明亮，因此我將三個threshold由內積值高至低分別設color為(1.0, 1.0, 1.0, 1.0)、(0.6, 0.6, 0.6, 1.0)、(0.2, 0.2, 0.2, 1.0)。

得到color後，將透過Uniform傳入的ourTexture與textCoord得到fragment的顏色，並存入object_color中，並計算輸出的FragColor，其中FragColor = Kd * object_color * color，Kd為diffuse光的係數。

4. Implement Edge effect(按鍵4開啟)

Edge shading只須在意正交向量與從觀察者指向物體的向量，因此我在main當中並沒有傳入任何uniform變數。

我按照助教提供的ppt，在vertex shader中先將基本的動作設定好，用Uniform的M(model matrix), V(view matrix), P(perspective matrix)先使每個vertex作相對應的轉換，並把結果以gl_Position輸出，除此之外，也處理正交向量與頂點位置。我將vertex乘上V*M的結果存入worldPos，紀錄vertex在world coordinate上的位置，並取出worldPos的x, y, z，轉成3維的vertPos，將用於fragment shader計算物體與觀察間的向量使用。最後我對該vertex上的normal用V*M做轉換，避免向量因乘上V*M而失真，並把結果放入normal變數中。先前得到的vertPos, normal會被vertex shader傳到fragment shader中運用。

進到fragment shader後，用先前得來的normal做normalization得到n, vertPos做normalization得到v。接下來利用n與v的內積去選取貓咪的邊緣，我設定當內積值大於等於負0.2時顯示藍色，其餘為黑色。

5. Problems and Solutions

- 在實作edge effect時，儘管試過各種內積值，卻無法正確標出邊緣的形狀，常常會變成貓咪身上部分位置一起變色，後來發現是一開始實作時忘記乘上V matrix，導致glsl用錯誤的位置去計算，後來加上去就能正常運作了。
- 在實作gouraud shading時，一開始先在vertex shader中將ambient, diffuse, specular三種光相加再傳到fragment shader，再由fragment shader乘上texture color，但這樣做效果在光點上一直不明顯。之後才發現specular是不須乘上texture color的，前述的作法會出錯，因此改為將ambient, diffuse與specular由vertex shader傳到fragment shader當中，再將ambient, diffuse乘上texture color，最後與specular相加輸出。