Introduction to Multimedia Homework #4

103062318蔡尚倫

Q1. Bézier curve

1. Describe how you implement the Bézier curve

定義四個參數：Sample\_rate、LoD、ctrlPointList(由ginput(sample\_point)產生)，sample\_point = Sample\_rate-1

首先將outlineVertexList大小設定好，由於一個group的最後一個point是下一個group的頭，所以有sample\_point/3個group，每個group有1/level+1個point(要記得算頭尾所以要加1)，總大小是zeros((sample\_point/3)\*(1/level+1),2)

接著避免重複計算，先創一個T matrix用於紀錄所有的coefficient，跑for loop t = 0:LoD:1將講義中公式中的四個coefficient (1-t)^3、3\*t\*(1-t)^2、3\*t^2\*(1-t)、t^3算出並分別填入

最後跑雙層for loop v=1:3:sample\_point-2 for loop t=0:LoD:1將講義中的公式帶入得到P，接著依照順序填入outlineVertexList，Loop結束完後outlineVertexList就是Bézier curve所估計出來的點

1. Discuss the results between different sampling rates and different levels of detail.

different sampling rates：高sampling rates的圖片與原圖相比，能夠較精細描繪出原圖，尤其是在圖片彎曲處，反之低sampling rates的圖片則無法如此精細的畫出原圖，在圖片彎曲處沒辦法準確描繪出曲線，我想原因是，高sampling rates能夠在彎曲處取得較多點的資訊，所以自然能使用Bézier curve描繪出比較精細的曲線，而低sampling rates則必須使用較少的control point點出圖片的輪廓，所以在曲線上取得的資訊就少於高sampling rates，描繪出來就沒這麼精細

different levels of detail：高levels of detail的圖片與原圖相比，能夠在每個group中描繪出較平滑的線條，反之低levels of detail則無法描繪出如此平滑的線條，我想原因是，高levels of detail在每個group中能夠取得較多的點(因為t以每0.01遞進至1)，所以看起來會比較平滑，而低levels of detail則只能取得較少的點描述一個group(因為t以每0.2遞進至1)，相較起來就沒那麼平滑

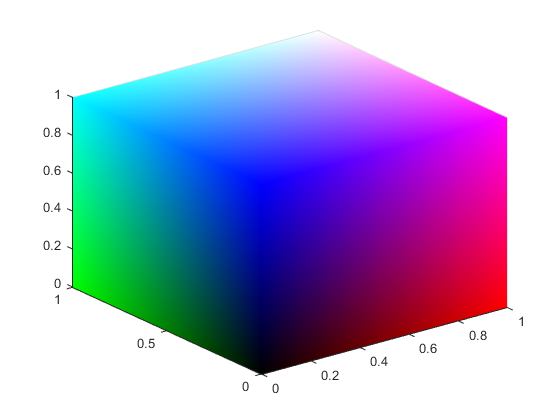
1. Compare results in (b) and discuss it.

結果如同講義所寫，如果使用Nearest-neighbor interpolation，周圍的顆粒感嚴重，反之使用Bézier curve(Scale the sampled control points by 4 times)則明顯看出周圍較平滑。我想原因是，Nearest-neighbor interpolation是取sample後的一個個pixel，每個pixel互相垂直與平行排列，所以周圍會呈現一個個pixel鋸齒狀的感覺，而Bézier curve是使用數學函式(Vector graph)描述周圍輪廓的點，圖片周圍的點是沿著輪廓function畫出來的，所以周圍會呈現較平順曲線的感覺。

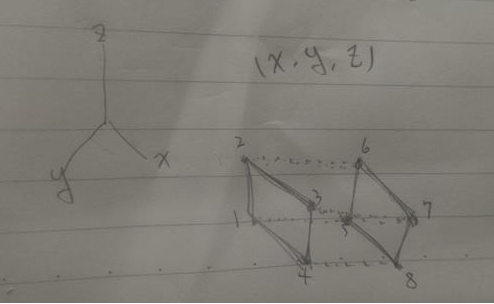
Q2. 3D Models(screenshot我直接將產生的.fig轉成.png貼上)

1. Show the screenshot of (2a) (2b) and discuss how to build these models.

(2a)



修改助教提供的makeRGBCube.m，因為CODE在填Side faces時，只填了其中一面的一半，所以只需要將另一半補回去就能完成，如下圖



EX：以最後那一面為例，助教給的code只填了三角形2 1 5，所以這次只要將2 6 5加進faces array即可

Code改寫成

for vertI = 1 : 4

faceVert1 = topVertIndex( mod(vertI,4)+1 ); %範例CODE提供

faceVert2 = topVertIndex( vertI ); %範例CODE提供

faceVert3 = botVertIndex( vertI ); %範例CODE提供

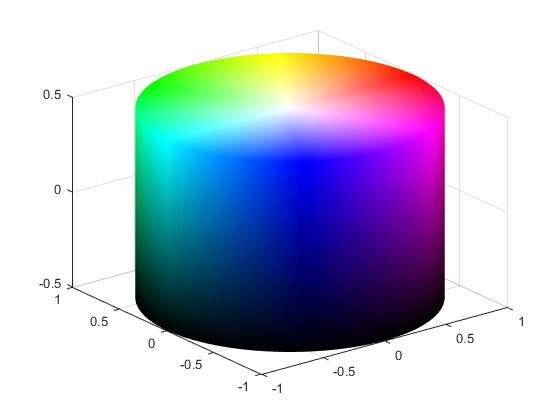
faces = [ faces ; faceVert1 faceVert2 faceVert3 ]; %範例CODE提供

faceVert2 = botVertIndex( mod(vertI,4)+1 ); %第二個點改取bot中與faceVert1同樣index的

faceVert3 = botVertIndex( vertI );

faces = [ faces ; faceVert1 faceVert2 faceVert3 ]; %將另一半三角形補齊

end

(2b)

由(2a)可以知道，順利將圖片畫出來需要faces、verts、vertColor這三個參數填入trisurf build function。

首先verts的座標在Supplement提到，

NumOfVert = 108; %取6的倍數 等一下在填Color時比較簡單

vertsPolarAngle = linspace(0,2\*pi,NumOfVert+1); %+1是為了讓間隔是NumOfVert數

vertsX = cos(vertsPolarAngle);

vertsY = sin(vertsPolarAngle);

這產生的是在X軸與Y軸上的座標，需要將Z軸上的資訊補齊，但要符合中心點在(0,0,0)，高為1的條件，所以topVert的Z座標為 0.5，而botVert的Z座標則為 -0.5，再來在畫top face與bot face時，需要中心點(0,0,0.5)與(0,0,-0.5)所以我分別塞在topVert與botVert的最後一個位置，最後將topVert與botVert合併，完成verts的設定

接著開始設定Color，由於botVert都是黑色的，所以最後全部設定為(0,0,0)即可，topVert Color變化由下面順序變化

R G B

1 0 0

1 1 0

0 1 0

0 1 1

0 0 1

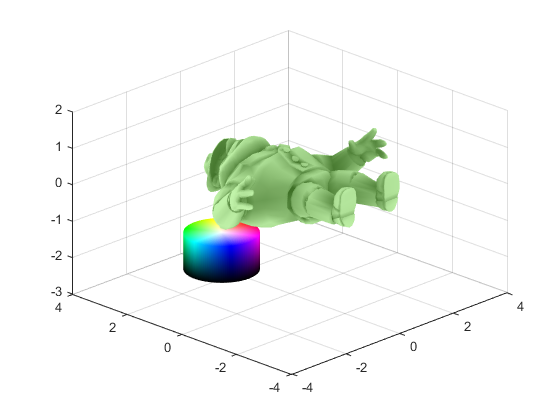
1 0 1

1 0 0

將0到1切成NumOfVert/6等份，照著RGB三色道變化分別填入相對應的數值，就能產生Supplement中Example的效果，最後記得top vert的中心要填(1,1,1)代表白色，如此一來就設定完vertColor

最後將faces的index填入，參考Supplement中的提示top與bot相同，將該point0、下一個point1與中心點NumOfVert+1填入(前面有提到我將中心點塞在最後一個)，最後要記得邊界處理，到pointLast時，下一個是point0，EX：(2,3,109)，比較難處理的是side face，看Supplement的example圖片發現，上方是top point0,bot point1,top point1，下方是bot point0,top point0,bot point1，照著這個順序就能順利完成face，最後一樣邊界要記得處理，如此一來就設定完faces

1. Show the screenshot of (2c) and discuss how to implement the transformation.



我直接修改readOBJFile.m來做修改，將(2b)的code複製一份到readOBJFile.m，接著只需要修改HSV cylinder中vert的座標即可，將所有座標位移(0,2.5,-2.5)，EX：top 與 bot的中心點則會跑到(0,0+2.5,0.5-2.5) (0,0+2.5,-0.5-2.5)，接著在ILMS上助教有說OBJ其實不在中心點，所以取出OBJ中的vert值(這裡以X軸當作範例講解)

max\_x = max(vertex(:,1)); %使用max找到最大值

min\_x = min(vertex(:,1)); %使用min找到最小值

x\_move = (max\_x + min\_x)/2; %算出位移量，是0的話自然不用位移

vertex (:,1) = vertex (:,1) - x\_move; %將X軸位移

Y軸與Z軸也能依樣畫葫蘆位移

將兩者的faces、verts、vertColor合併就能將3D Model合併

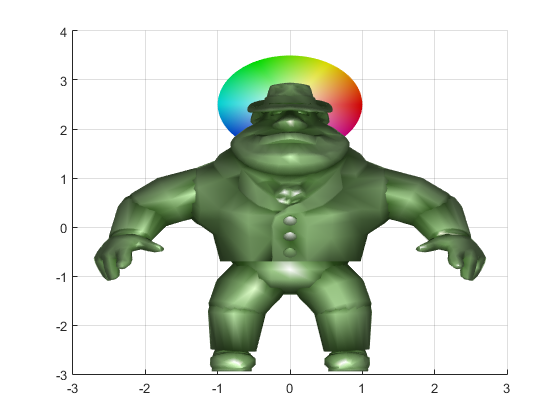
vertex = [vertex;verts]; %填在後面即可

outfaces = [outfaces;faces+3618]; %注意因為我將HSV cylinder填在後面，所以HSV cylinder中face的index要位移3618(OBJ的outface數量)

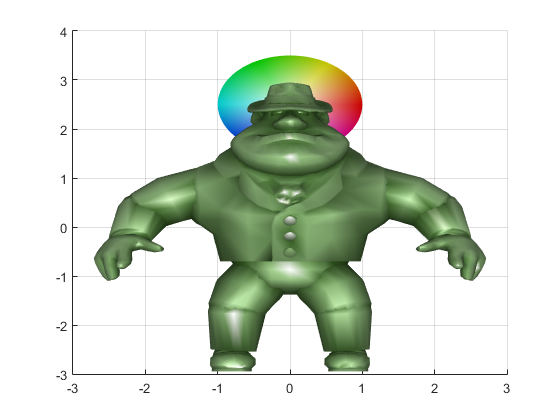
colors = [colors;vertColor]; %填在後面即可

1. Show the screenshot of (2d) and discuss the difference of different kind of light.( screenshot轉到正面作比較)

Postion light



Directional light



Postion light代表點光源，向四面八方四散，光源位置在(0,0,5)

Directional light代表平行光(類似太陽光概念)，平行照射，光源來自(0,0,5)

很明顯可以看出，Postion light的圖中，邊緣處比較暗，我想是因為點光源四散後，反射到OBJ到其他地方，反射光並沒有進入我們眼球，所以距離點光源越遠的地方，我們會覺得比較暗(陰影的感覺)，而Directional light則是平行光，OBJ的照到平行光後，大部分的光線都反射到我們眼球，所以我們看起來會比較亮

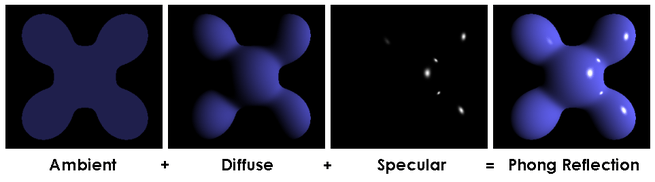
1. Show the screenshot of (2e) and discuss the difference from the results.

參考

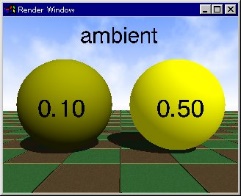
<https://cg2010studio.com/2011/03/27/specular/>

<http://www.arch.oita-u.ac.jp/a-kse/povjp/povjp/html/pov35ref_tex0c.html>

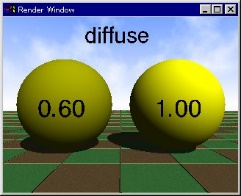
Ka，Kd為受到material color影響，Ks為受到light color影響



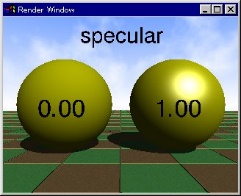
Ka代表本身OBJ的環境光程度參數(亮暗效果)

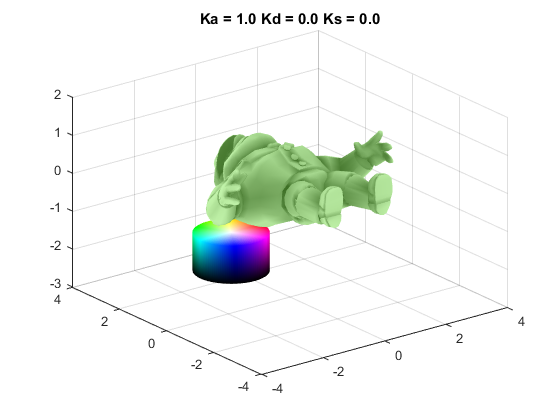


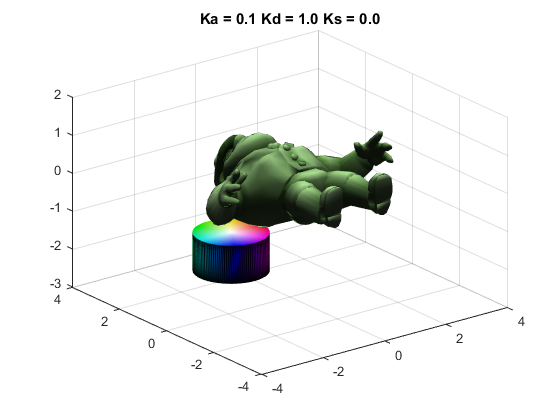
Kb代表本身OBJ的陰影程度參數(陰影效果)

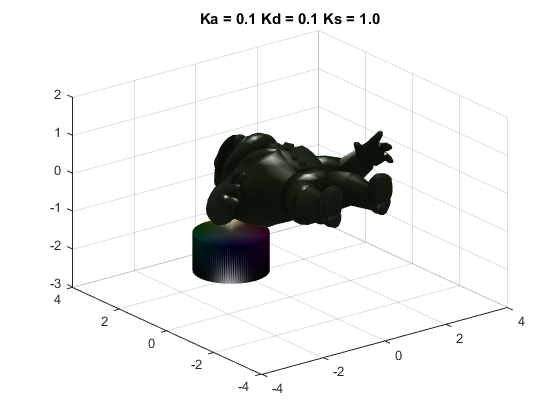


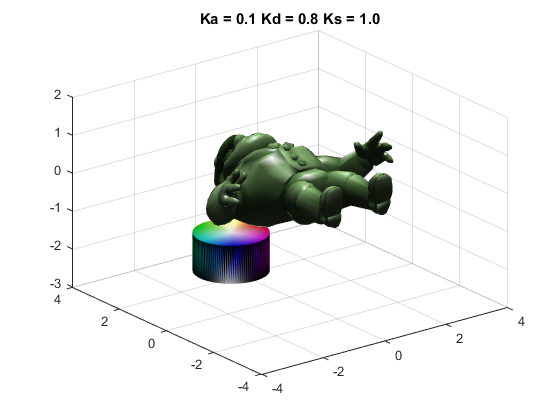
Ks代表本身OBJ的反光程度參數(反光效果 光澤)











Ka = 1.0 Kd = 0.0 Ks = 0.0圖片，由於Ka為最大值，所以會呈現很亮的OBJ，其餘都為0，沒有陰影效果與反光效果，所以相較於另外三張圖片，非常像(2c)的圖片。

Ka = 0.1 Kd = 1.0 Ks = 0.0圖片，參考資料中知道，Ka Kd受到material color影響，Ks受到light color影響，受material color影響會大於Ka = 0.1 Kd = 0.1 Ks = 1.0圖片，所以呈現較像material color，Ka很小，因此亮度較Ka = 1.0 Kd = 0.0 Ks = 0.0圖片小很多，Kd為1，OBJ會呈現比Ka = 0.1 Kd = 0.1 Ks = 1.0圖片強的陰影效果，Ks為0，沒有反光效果。

Ka = 0.1 Kd = 0.1 Ks = 1.0圖片，參考資料中知道，Ka Kd受到material color影響，Ks受到light color影響，Ka Kd都很小，亮度比Ka = 1.0 Kd = 0.0 Ks = 0.0圖片小，陰影程度比Ka = 0.1 Kd = 1.0 Ks = 0.0圖片小，受material color影響很小，所以呈現黑綠色，而Ks為1.0，有反光效果。

Ka = 0.1 Kd = 0.8 Ks = 1.0圖片，參考資料中知道，Ka Kd受到material color影響，Ks受到light color影響，受material color影響會大於Ka = 0.1 Kd = 0.1 Ks = 1.0圖片，所以呈現較像material color，Ka很小，因此亮度較Ka = 1.0 Kd = 0.0 Ks = 0.0圖片小很多，Kd為0.8，OBJ會呈現比Ka = 0.1 Kd = 0.1 Ks = 1.0圖片強的陰影效果，比Ka = 0.1 Kd = 1.0 Ks = 0.0圖片弱的陰影效果。