HW4_Report

1、

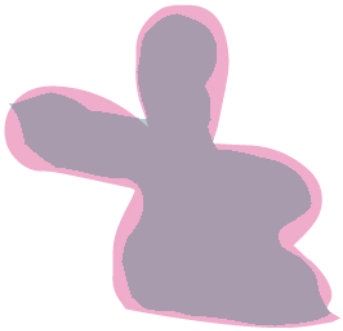
(a)

Low sampling rate:40points LoD t={0,0.2,0.4,…1}

Low sampling rate:40points LoD t={0,0.01,0.02,…1}

Low sampling rate:100points LoD t={0,0.2,0.4,…1}

Low sampling rate:100points LoD t={0,0.01,0.02,…1}



(b)

Scale the bitmap result:

Scale the sampled control points:

(c)

1、關於 Bezier curve 依照講義第 13-15 頁的公式，40points 代表要標 39 個點，因為上一個的尾是下一個的頭，所以實際上每次是多讀三個點而已，也就是如此取的點的個數希望盡量是 3 的倍數，唯一要注意的是最後四組的尾是最一開始的點，達到接起來的效果，將結果存進去 outlineVertexList 即可，code 如下。

```
for i=1:3:clickedN-1
    if(i+3>=clickedN+1)a=mod(i+3,clickedN);
    else a=i+3;
    end
    if (i+2>=clickedN+1)b=mod(i+2,clickedN);
    else b=i+2;
    end
    if(i+1>=clickedN+1)c=mod(i+1,clickedN);
    else c=i+1;
    end
    for t=0:level_rate:1

    outlineVertexList(index,1)=(1-t)^3*ctrlPointList(i,1)+3*t*(1-t)^2*ctrlPointList(c,1)+3*t^2*(1-t)*ctrlPointList(b,1)+t^3*ctrlPointList(a,1);
    outlineVertexList(index,2)=(1-t)^3*ctrlPointList(i,2)+3*t*(1-t)^2*ctrlPointList(c,2)+3*t^2*(1-t)*ctrlPointList(b,2)+t^3*ctrlPointList(a,2);
    index=index+1;
    end
end
```

2、明顯的可以從 a 小題發現 levels of details 影響曲線的曲度有很大的關聯，這是因為若今天 t=0.2 的間距時，等於我在這四個點(3rd order)中用六個點去逼近一個曲線的方程式，跟我在四個點中用 100 個點去逼近結果是顯而易見的，取的點越多越能逼近一條曲線。

至於 sampling rate 的數量對於曲線的曲度也是呈正向關係的，因為當取的點越多時，每兩點間的間距越短，在同樣的 levels of detail 下距離越短能產出的曲線就越曲。

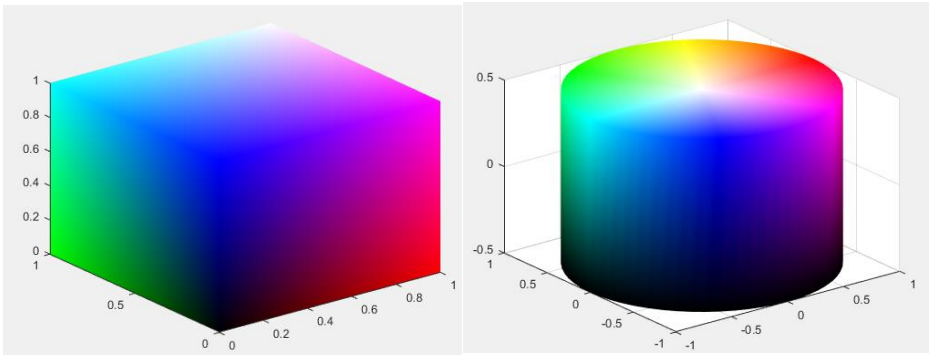
3、

這張是 b 小題 scale sampled control points 局部放大的效果，有鋸齒狀的是我單純去放大結果圖四倍的效果，而去放大 control points 的效果較單純放大四倍的效果好，這是因為 bitmap 跟 vector 的差異，從講義 4-7 頁我們可以發現若是以 bitmap 方式去放大的結果容易在邊界有鋸齒狀，vector 好很多，這是因為一個是以向量的方式，一個是以原本 x offset 跟 y offset 有幾個點的方式去做放大，所以 vector 做出來的效果會比 bitmap 好很多。

註:b scale sampled control points 的圖我是將放大四倍的圖當背景去畫曲線，所以顏色較深，並且疊在一起可以方便去比較。

2、

1、



2(a)這題比較單純，framework 中已經做出上半部分的三角形了，一個正方形是由兩個三角形拼湊而成的，所以我們只要在拚出下半部的三角形即可(兩點在下一點在上)，code 如下

```
for vertI = 1 : 4
    faceVert1 = topVertIndex( mod(vertI,4)+1 );
    faceVert2 = topVertIndex( vertI );
    faceVert3 = botVertIndex( vertI );
    faceVert11 = botVertIndex( mod(vertI,4)+1 );
    faceVert21 = botVertIndex( vertI );
    faceVert31 = topVertIndex( mod(vertI,4)+1 );
    faces = [ faces ; faceVert1 faceVert2 faceVert3 ; faceVert11,faceVert21,faceVert31 ];
end
```

至於圓柱體的話可以視為三個部分拼裝起來，上圓下圓跟側面，上圓的話將 360 度切成 60 個區塊(60 根據 supplement 能看起來有逼近圓面的效果)，視為 60 個小三角形的組成，每個三角形的三個點分別是圓心跟邊上兩個點，底部的圓面也如法炮製，至於側面的話就要應用上一題正方形拚側面的方式，上兩點配下一點跟下二點搭上一點的方式去拚進，可以這樣做的理由是因為若我們將圓柱的側面攤平其實就是一個長方形，所以可以用跟剛剛正方形的做法去實踐。

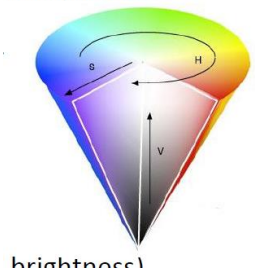
首先透過 linspace 去切點，之後透過 supplement 上給的公式去給出 x、y 點，之後開始對點，最後一個點等於第一個點，對於超出(numOfVertex=60+1)等於繞了一圈，這點要特別去做判斷，以免對到下面的點，code 如下。

```
topvert=[0,0,0.5];
botvert=[0,0,-0.5];
faces_c=[];
for i=1:numOfVert
    topvert=[topvert;vertsX(i,1),vertsY(i,1),0.5];
    if i+2>numOfVert+1
        a=2;
    else
        a=i+2;
    end
    faces_c=[faces_c;1,i+1,a];
end
[a,b]=size(topvert);

%Side face
    for v=1:numOfVert
       if v+2>numOfVert+1
          a=2;
       else
           a=v+2;
       end
        if numOfVert+v+3>numOfVert*2+2
           b=numOfVert+3;
       else
          b=numOfVert+v+3;
       end
```

```
      faceVert1=v+1;
      faceVert2=a;
      faceVert3=numOfVert+v+2;
      faceVert11=a;
      faceVert12=b;
      faceVert13=numOfVert+2+v;

      faces_c=[faces_c;faceVert1,faceVert2,faceVert3;faceVert11,faceVert12,faceVert13];
   end
% %Bottom faces

for i=1:numOfVert
    botvert=[botvert;vertsX(i),vertsY(i),-0.5];
    if numOfVert+i+3>numOfVert*2+2
        b=numOfVert+4;
    else
        b=numOfVert+i+3;
    end
    faces_c=[faces_c;numOfVert+2,numOfVert+i+2,b];
end
[a,b]=size(botvert);

verts=[topvert;botvert];
[a,b]=size(verts);
```

色彩部分的話對照下面這張圖，中心到邊界的 s 為 0-1，v 由上至下為 0-1，h 的部分看以角度去分。

0-100% in some



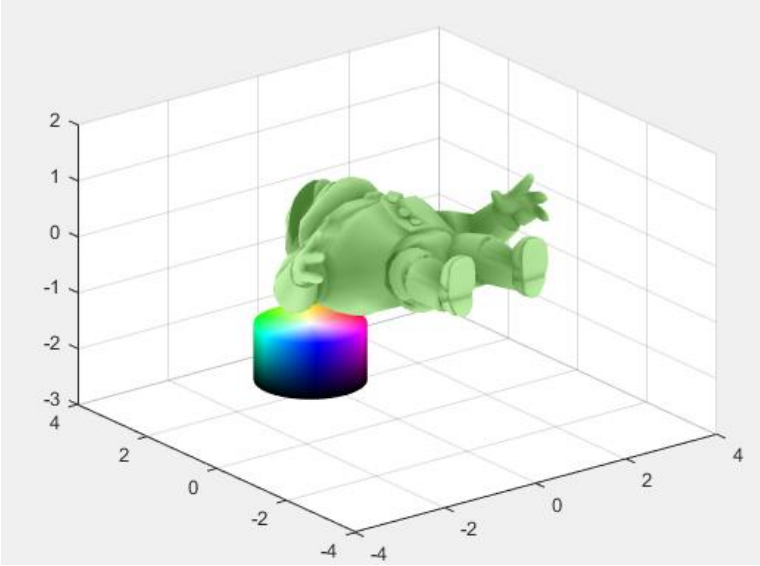
brightness)

```
topColor=[0,0,1];
botColor=[0,0,0];
for i=1:numOfVert
    topColor=[topColor;vertsPolarAngle(i)/(2*pi),1,1];
    botColor=[botColor;vertsPolarAngle(i)/(2*pi),1,0];
end
```

最後轉為 rgb 輸出。

```
vertColors = [ topColor; botColor ];
vertColors=hsv2rgb(vertColors);
```

2、



在做 translation 時必須先找到物體的中心點，並減掉(x,y,z)，然後加上期望位移的座標，因為 model 希望移到(0,0,0)所以可以不用加，至於 cylinder 因為中心點在(0,0,0)所以不用減去，中心點的求法是分別找出 x,y,z 的 max 和 min 然後除以 2 就是物體的中心點，code 如下。

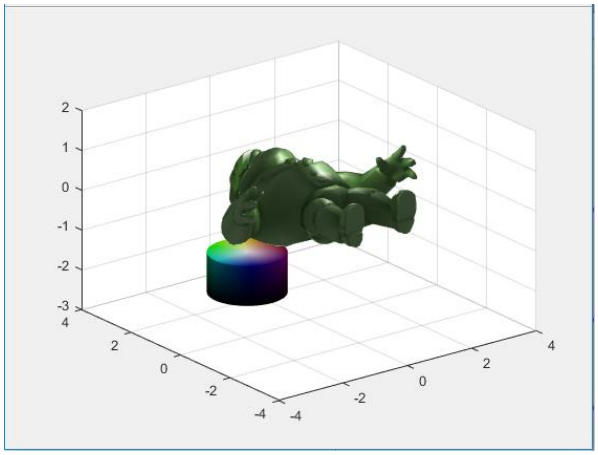
```
y_max=realmin;
z_max=realmin;
for i=1:m
    for j=1:n
        if j==1
            if vertex(i,j)>x_max
                x_max=vertex(i,j);
            end
        elseif j==2
            if vertex(i,j)>y_max
               y_max=vertex(i,j);
            end
        elseif j==3
            if vertex(i,j)>z_max
                z_max=vertex(i,j);
            end
        end
    end
end
```

```
x_min=realmax;
y_min=realmax;
z_min=realmax;
for i=1:m
    for j=1:n
        if j==1
            if vertex(i,j)<x_min
                x_min=vertex(i,j);
            end
        elseif j==2
            if vertex(i,j)<y_min
               y_min=vertex(i,j);
            end
        elseif j==3
            if vertex(i,j)<z_min
                z_min=vertex(i,j);
            end
        end
    end
end
```

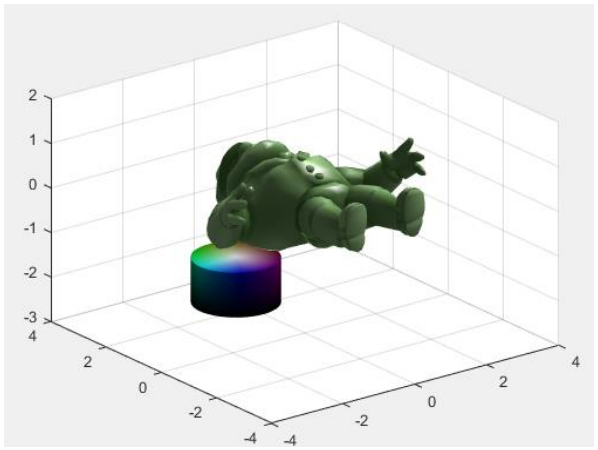
```
shift_vert(:,1)=verts(:,1);
shift_vert(:,2)=verts(:,2)+2.5;
shift_vert(:,3)=verts(:,3)-2.5;
vertex=[vertex;shift_vert(:,1),shift_vert(:,2),shift_vert(:,3)];
colors=[colors;vertColors(:,1),vertColors(:,2),vertColors(:,3)];
figure;trisurf(faces,vertex(:,1),vertex(:,2),vertex(:,3),'FaceVertexCData', colors,'FaceColor','interp', 'EdgeAlpha', 0);
```

3、

Points light



Directional light

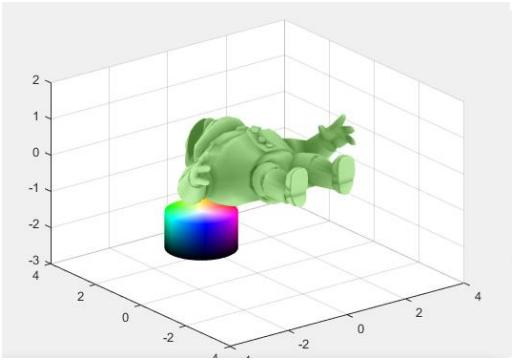


Directional light 跟 Point light 在 ambient、diffuse、specular 的實作其實是一樣的，只不過差在 point light 的 diffuse specular 會受到距離函數 fatt 的影響，也就是當距離光源越遠時，所受到的光照越弱，一般來說 fatt 可表示成 fatt(t)=1/a+b*t+c+t^2，t 代表與光源的距離。，透過內建函數 light 中將 style 設為 local or infinity 即可，code 如下。

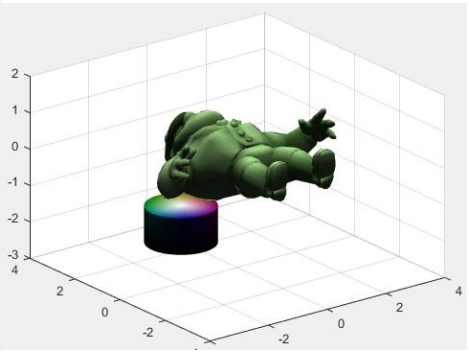
```
figure;trisurf(faces,vertex(:,1),vertex(:,2),vertex(:,3),'FaceVertexCData', colors,'FaceColor','interp', 'EdgeAlpha', 0);
l1 = light('Style', 'local', 'Visible', 'on');
lighting phong;


figure;trisurf(faces,vertex(:,1),vertex(:,2),vertex(:,3),'FaceVertexCData', colors,'FaceColor','interp', 'EdgeAlpha', 0);
l2 = light('Style', 'infinite', 'Visible', 'on');
lighting phong;
```

4、

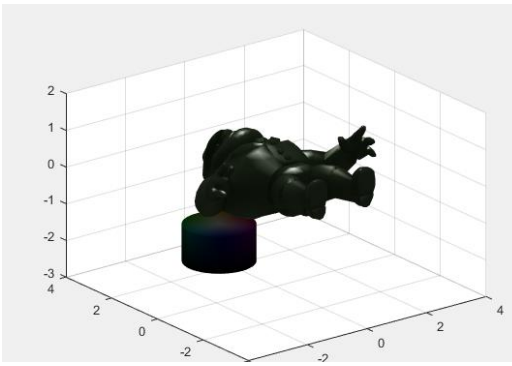
(ka kd ks)=(1.0,0.0,0.0)



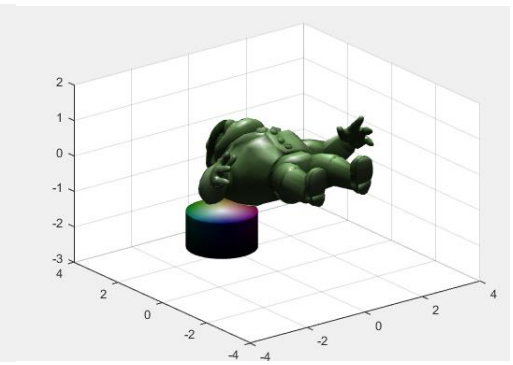
(ka kd ks)=(0.1,1.0,0.0)



(ka kd ks)=(0.1,0.1,1.0)



(ka kd ks)=(0.1,0.8,1.0)



ambient 代表環境光，diffuse 指的是光的漫射散射，反應在物體的顏色深淺上面，specular 管光的全反射，反應在物體的光澤上。

第一張圖因為只有 ambient 所以不受光的照射效果跟光澤影響，所以呈現的是物體本來的顏色，第二張圖因為沒有 specular 所以沒有光澤，第三圖因為 diffuse 很小所以顏色不明顯，但可以清楚地看到光澤，第四張圖就是綜合的效果。