Multimedia HW4

第一題：

第一題要根據提供的 vertex 座標，畫三次方的貝茲曲線。

在這裡根據講義第 15 頁的矩陣乘法，定義 T 矩陣、M 矩陣、G 矩陣。

$$P(t) = T * M * G$$

$T = [t^3\ t^2\ t\ 1]$, M is called the ***basis matrix***. G is called the ***geometry matrix***.

$$M = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \qquad G = \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

在 matlab 實作中的程式碼為如此：

```
M = [-1 3 -3 1 ; 3 -6 3 0 ; -3 3 0 0 ; 1 0 0 0];
T = zeros(1,4);
G_x = zeros(4,1);
G_y = zeros(4,1);
```

由於已經知道是要做三次方的貝茲曲線，所以 M 可以根據 blending function 算好，而講義 G 中的 p 為一個點的座標，這裡我將 x 和 y 拆成 G_x 和 G_y。至於 T 的部分，由於在算貝茲曲線的時候，t 的值會一直變動，所以我這裡先設他為 [0 0 0 0]。

這一題中要根據兩種不同的精細程度，來畫貝茲曲線。主要做法就是用一層迴圈去跑不同的 t，讓 t 從 0 跑到 1。

首先根據 low detail 實作：

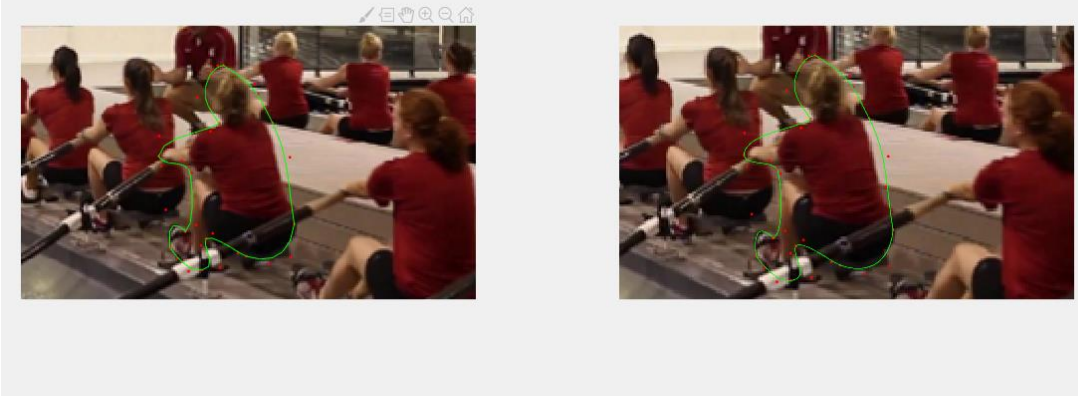
```
for i = 1:3:n-3
    for t = 0:0.2:1
       T = [t^3 t^2 t 1];
       P = T*M;
       G_x = [points(i,1) ; points(i+1,1) ; points(i+2,1) ; points(i+3,1)];
       resultl(index,1) = P*G_x;
       G_y = [points(i,2) ; points(i+1,2) ; points(i+2,2) ; points(i+3,2)];
       resultl(index,2) = P*G_y;
       index = index + 1;
    end
end
```

第一層迴圈是為了跑不同組的點來做貝茲曲線，可以看到每一組分別為 1~4，4~7，…，34~37。第二層就是去跑 t。中間做的就是矩陣的乘法，並把做完的結果存到 result1 中，並將 x 和 y 分開做。

接著根據 high detail 實作：

```
for i = 1:3:n-3
    for t = 0:0.01:1
        T = [t^3 t^2 t 1];
        P = T*M;
        G_x = [points(i,1) ; points(i+1,1) ; points(i+2,1) ; points(i+3,1)];
        result2(index,1) = P*G_x;
        G_y = [points(i,2) ; points(i+1,2) ; points(i+2,2) ; points(i+3,2)];
        result2(index,2) = P*G_y;
        index = index + 1;
    end
end
```

將做出來的結果 plot 出來：



可以觀察到，在 high detail（右方）情況下做出來的貝茲曲線較為圓滑。

接著，要將圖片 scale 成 4 倍，然後去做貝茲曲線。由於這裡是 scale 成 4 倍，首先要做的是將 vertex 的座標全部乘以 4。

```
points = points .* 4;
```

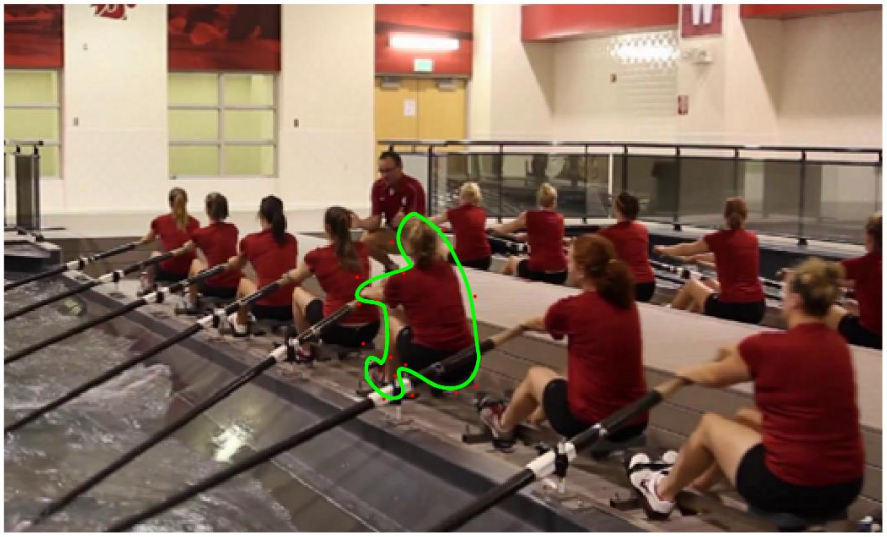
有了在 x 和 y 方向都乘 4 的座標後，使用這組座標做貝茲曲線（用上一題的 high detail 來做）。

```
index = 1;
for i = 1:3:n-3
    for t = 0:0.01:1
       T = [t^3 t^2 t 1];
       P = T*M;
       G_x = [points(i,1) ; points(i+1,1) ; points(i+2,1) ; points(i+3,1)];
       result3(index,1) = P*G_x;
       G_y = [points(i,2) ; points(i+1,2) ; points(i+2,2) ; points(i+3,2)];
       result3(index,2) = P*G_y;
       index = index + 1;
    end
end
```

同時，為了要讓做出來的貝茲曲線和圖片對齊，這裡呼叫 imgresize，也將圖片 scale 成 4 倍，使用的方式是 nearest neighbor。

```
f_2 = figure(2);
img_2 = imresize(img,4,'nearest');
imshow(img_2);
```

以下是實作出來的結果：



這裡可以看到，由於圖片 scale 成四倍，也要先將點的 x 和 y 座標 scale 成 4 倍。接著拿了 scale 成 4 倍的座標去做貝茲曲線，可以看到就算 scale 成 4 倍，貝茲曲線形狀看起來和原圖尺寸的時候差不多，有差別的就是曲線的大小，因為點跟點之間的距離拉開了，但可以看到貝茲曲線仍然圈住圖中的運動員，就和用原圖比例去做貝茲曲線的時候一樣。

第二題：

首先要把讀進來的 obj 的 center 移到(0,0,0)。

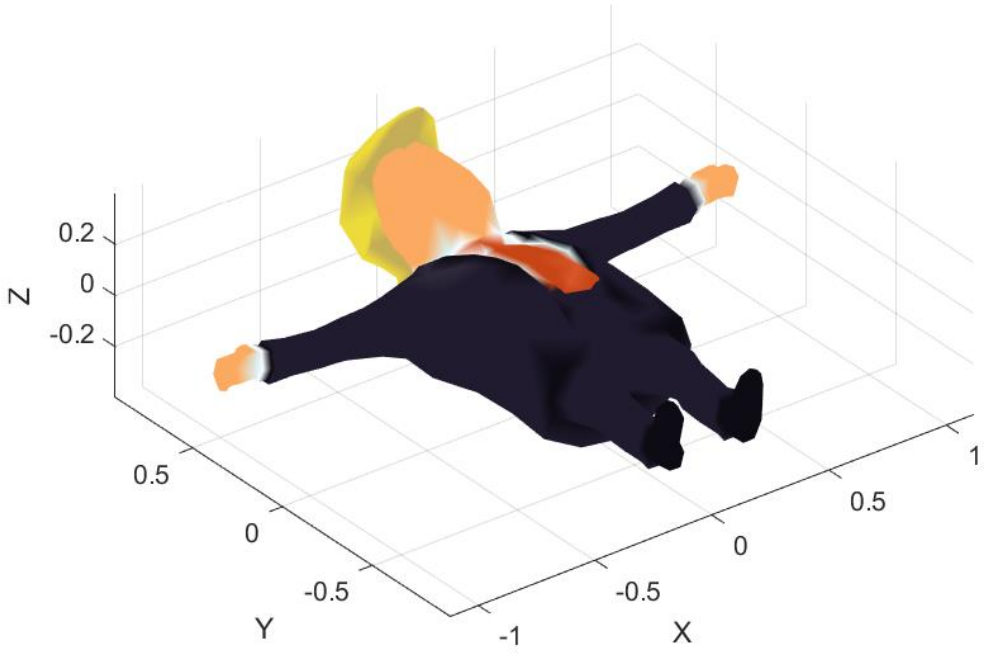
這裡先算出目前的 center 是多少：

```
Center = [(max(obj.v(:,1))+min(obj.v(:,1)))/2 ; (max(obj.v(:,2))+min(obj.v(:,2)))/2 ; (max(obj.v(:,3))+min(obj.v(:,3)))/2];
```

接著，根據目前的 center，讓每一個 vertex 的座標減掉目前 center 的座標，這樣就會讓 center 跑到(0,0,0)。

```
obj.v(:,1) = obj.v(:,1) - Center(1,1);
obj.v(:,2) = obj.v(:,2) - Center(2,1);
obj.v(:,3) = obj.v(:,3) - Center(3,1);
```

做出來的結果如下：



下一步要做的是畫出 HSV 的 cone。

這裡首先根據 HSV 的定義，將 H、S、V 三種值建成三個 100*100 的矩陣，並合起來組成一張 HSV 的圖，等等要貼到 cone 的表面。

```
H = repmat(linspace(0, 1, 100), 100, 1);
S = repmat([linspace(0, 1, 50) ...
            linspace(1, 0, 50)].', 1, 100);
V = repmat([ones(1, 50) ...
            linspace(1, 0, 50)].', 1, 100);
hsvImage = cat(3, H, S, V);
C = hsv2rgb(hsvImage);
```

根據 HSV color cone 的定義，H 為色相，S 為飽和度，V 為亮度。由於每一個色相，都包含了不同的飽和度和亮度，故讓每一個 H（0~1）都會有不同的飽和度（0~1）和亮度（0~1），透過將這三個 100*100 的矩陣組合成三個維度的矩陣，就形成了 HSV 的圖。接著將這張圖從 HSV 轉換成 RGB，呈現出色彩的樣貌。

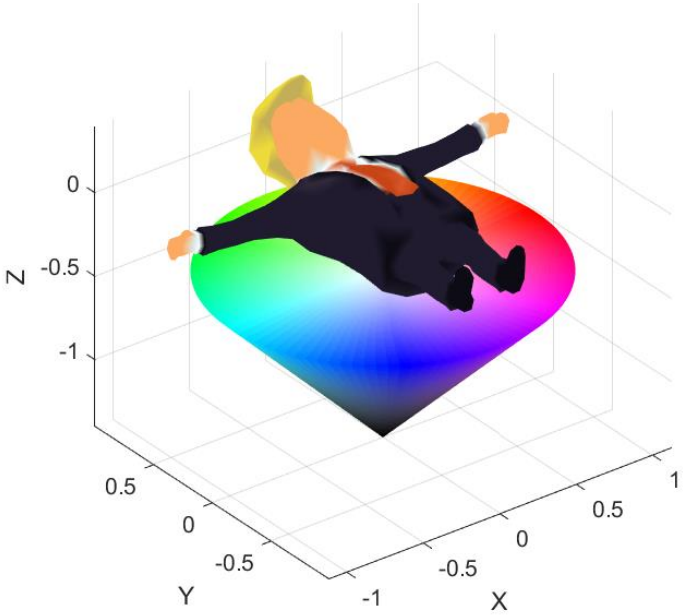
接著畫 cone。這裡將 x、y、z 的座標系統定義出來。由於是圓錐，在 xy 平面上用 cos 和 sin 來表示要怎麼呈現。另外，由於 peak 要在(0,0,-1.4)上，Z 設定好之後 peak 在(0,0,0)上，所以要再減 1.4。

```
theta = linspace(0, 2*pi, 100);
X = [zeros(1, 100); ...
     cos(theta); ...
     zeros(1, 100)];
Y = [zeros(1, 100); ...
     sin(theta); ...
     zeros(1, 100)];
Z = [ones(2, 100); ...
     zeros(1, 100)];
Z = Z - 1.4;
```

有了 cone 和 HSV 的圖之後，就可以畫出 cone 並把 HSV 的圖貼到這個 cone 上面。

```
surf(X, Y, Z, C, 'FaceColor', 'texturemap', 'EdgeColor', 'none');
```

做出來的結果如下：

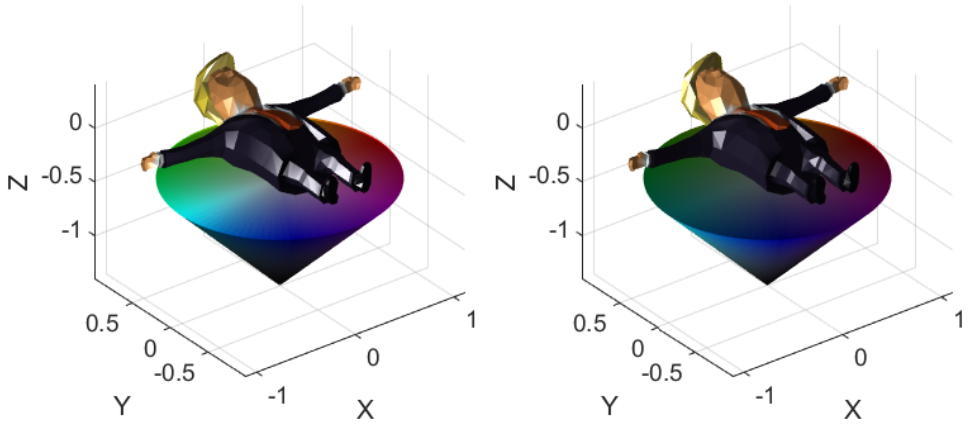


有了這個結果之後，接下來要做打光的動作。這裡使用 matlab 的函式 light。

```
light('Position', [-1 0 0], 'Style', 'local');
```

```
light('Position', [-1 0 0], 'Style', 'infinite');
```

由於題目要求有兩種光，分別為 directional 和 positional，這裡根據 matlab 的文件，將 style 設為 local 和 infinite。當 style 是 local 時，position 指的是光的位置，故為 positional light；當 style 是 infinite 時，position 指的是光的方向，故為 directional light。以下是做出來的結果(左邊為 positional light，右邊為 directional light)：

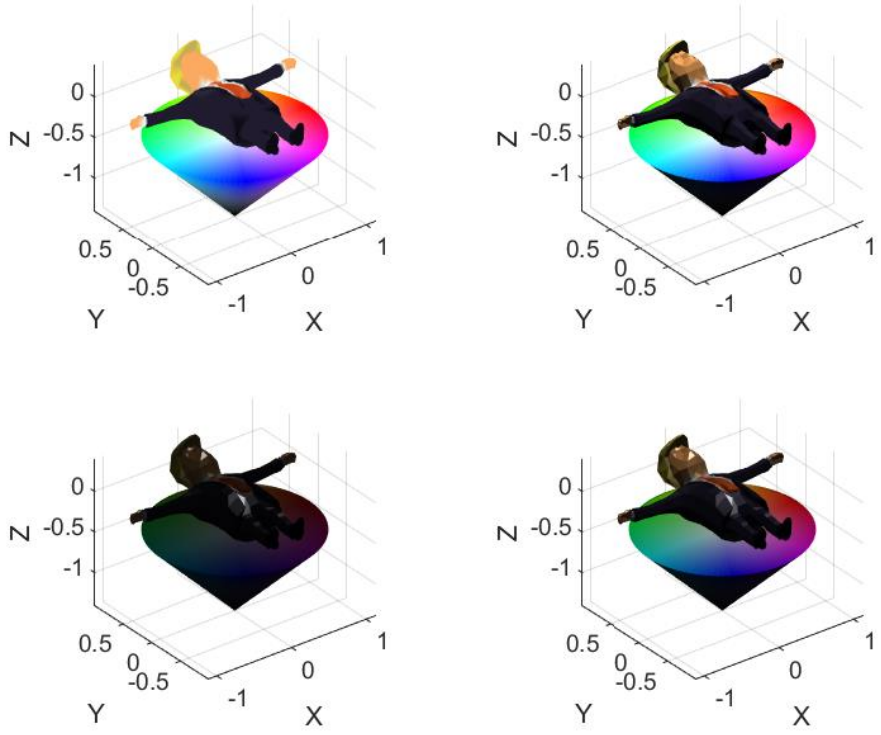


可以看到由於 positional light 的光源位置就在我設的(-1,0,0)上，所以打光的時候有點光源的性質，越接近的地方越亮（可以看到在 cone 平的地方，越接近光源的地方越亮），物體表面的 normal 和光的入射角也會影響做出來的視覺效果（可以觀察到川普的表面會因為 normal 跟光的入射夾角不同，造成效果不同）。而 directional light 的光源是 infinite，就像太陽光一樣，基本上對於 model 的方向的一樣（可以觀察到在 cone 的平的地方因為是 directional light，所以平的地方亮度幾乎一樣），但還是會根據表面的 normal 和光的入射角不同，而造成打光的效果不同（可以觀察到川普的表面會因為 normal 跟光的入射夾角不同，造成效果不同）。

最後是要去根據 d 小題的光源，去調整環境光和漫射強度以及全反射強度的係數。在這裡我是用 k1 和 k2 等於畫出來的川普和 HSV color cone，並用 k1 和 k2 去調整他們的環境光、漫射、全反射的係數。

```
k1 = trisurf(obj.f.v, obj.v(:,1), obj.v(:,2), obj.v(:,3), ...
    'FaceVertexCData', tval, 'FaceColor', 'interp', 'EdgeAlpha', 0);
hold on;
xlabel('X'); ylabel('Y'); zlabel('Z');
k2 = surf(X, Y, Z, C, 'FaceColor', 'texturemap', 'EdgeColor', 'none');
axis equal
light('Position', [0 0 1], 'Style', 'infinite');
k1.AmbientStrength = 1.0;
k1.DiffuseStrength = 0.0;
k1.SpecularStrength = 0.0;
k2.AmbientStrength = 1.0;
k2.DiffuseStrength = 0.0;
k2.SpecularStrength = 0.0;
```

以下是做出來的結果：



可以觀察到左上角那張因為環境光為 1(最高值)，所以整張圖就算沒有被光源照到的地方也是亮的，也因為漫射和全反射都是 0，看不出光源照在 model 上的效果。右上角的圖是環境光為 0.1，可以看到圓錐下方的部分因為沒有照到光，所以幾乎是暗的，而漫射的係數為 1，可以看到光照在川普以及圓錐平面部分

的漫射效果，基本上照到的地方都是亮的，而沒有照到的地方會有陰影。此部分可以和左上那張圖做明顯的比較，因為左上那張圖環境光為 1，所以看不到任何的陰影，而右上的圖因為環境光很暗，又漫射的係數為 1，故可以看到光打到的地方才亮，川普的 model 也有陰影產生。又因為右上的圖全反射係數為 0，所以看不到全反射。

左下的圖為環境光和漫射係數為 0.1，可以看到光沒照到的地方很暗，有照到的地方亮一點點，但也是很暗，然而這張圖因為全反射係數為 1，故在特定部分因為觀察角度，可以看到川普 model 有全反射的現象。

右下的圖為環境光為 0.1，漫射係數為 0.5，全反射係數為 1。拿他和左邊的圖做比較可以發現圓錐下方的部分一樣暗，因為都沒被光照到，然而被光照到而沒有全反射的部分，右邊的圖比左邊的亮，那是因為漫射係數高的原因。最後，可以看到全反射的部分由於兩張圖的係數一樣，故在同樣的地方有同樣強度的全反射。