Homework 4 104062241 莫雅雯

Part1

(a) Compute the Bézier curve from the sampled control points

1.做法

實做 ComputeBezier.m function 來計算 Bézier curve

(1) ComputeBezier.m

```
function [out]=ComputeBezier(ctrlPointList,N, LoD,c)
   for i=1:3:N
        for t=0:LoD:1
            p1=i;
            p2=i+1;
            p3=i+2;
            p4=i+3;
            if(p1>N)
                p1=rem(p1,N);
            end
            if(p2>N)
                p2=rem(p2,N);
            end
            if(p3>N)
                p3=rem(p3,N);
            end
            if(p4>N)
                p4=rem(p4,N);
            end
```

參考講義的公式計算 Bézier curve。

(1)從 part1.m 呼叫 ComputeBezier 傳入:

ctrlPointList : control points

N : control points 數

LoD : Levels of detail

c : 1

(2)須注意最後 4 組尾是一開始的點，才能達到聯接的效果，這裡我用超過 N 就取 rem 的方法。

```
temp(c,1)=(1-t)^3*ctrlPointList(p1,1)+3*t*(1-t)^2*ctrlPointList(p2,1)
          +3*t^2*(1-t)*ctrlPointList(p3,1)+t^3*ctrlPointList(p4,1);
temp(c,2)=(1-t)^3*ctrlPointList(p1,2)+3*t*(1-t)^2*ctrlPointList(p2,2)
          0+3*t^2*(1-t)*ctrlPointList(p3,2)+t^3*ctrlPointList(p4,2);
c=c+1;
```

依照講義的公式算 Bézier curve

(2) part1.m

```
%% Mouse input
xlabel ('Select at most 36 points along the outline', 'FontName', '微軟正黑體', 'FontSize', 14);
[ ctrlPointX, ctrlPointY ] = ginput(36);
ctrlPointList_36 = [ctrlPointX ctrlPointY];
clicked_N_36 = size(ctrlPointList_36,1);

promptStr = sprintf('%d points selected', clicked_N_36);
xlabel (promptStr, 'FontName', '微軟正黑體', 'FontSize', 14);

%-------------------------------------------------------
xlabel ('Select at most 72 points along the outline', 'FontName', '微軟正黑體', 'FontSize', 14);
[ ctrlPointX, ctrlPointY ] = ginput(72);
ctrlPointList_72 = [ctrlPointX ctrlPointY];
clicked_N_72 = size(ctrlPointList_72,1);

promptStr = sprintf('%d points selected', clicked_N_72);
xlabel (promptStr, 'FontName', '微軟正黑體', 'FontSize', 14);
```

做 2 個 mouse input ，一個做 Low sampling rate: 36 points.

，一個做 High sampling rate: 72 points。

```
%rate=36
LoD_low=0.2;
LoD_high=0.01;
outlineVertexList36_low=ComputeBezier(ctrlPointList_36, clicked_N_36, LoD_low,1);
outlineVertexList36_high=ComputeBezier(ctrlPointList_36,clicked_N_36,LoD_high,1);

%rate=72
outlineVertexList72_low=ComputeBezier(ctrlPointList_72, clicked_N_72, LoD_low,1);
outlineVertexList72_high=ComputeBezier(ctrlPointList_72,clicked_N_72,LoD_high,1);
```

(1)指定給 Low/High Levels of detail 值

(2)呼叫 ComputeBezier function 並傳入對應參數。

```
figure('name',X{1},'numbertitle','off');
subplot(2, 2, 1);
result36_low=drawAndFillPolygon(rbImage,ctrlPointList_36,outlineVertexList36_low,true,true,true);
title('sampling rate=36,level=Low');
```

用助教給的 drawAndFillPolygon.m function 畫 polygon.

(b) Using (sampling rate = High, levels of detail = High) in (a) to do scaling

1.做法

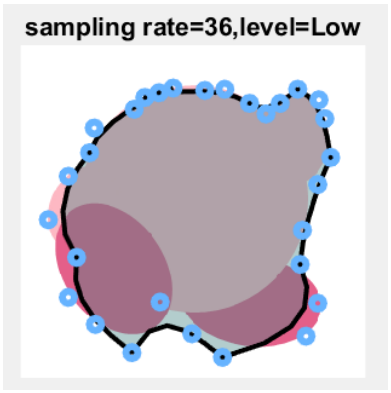
```
result_scaling=imresize(result72_high, 4, 'nearest');
 for i = 1 :clicked_N_72
        ctrlPointList_72(i,1)=4*ctrlPointList_72(i,1);
        ctrlPointList_72(i,2)=4*ctrlPointList_72(i,2);
 end
 outlineVertexList_s=ComputeBezier(ctrlPointList_72,clicked_N_72,LoD_high,1);
 figure('name',X{3},'numbertitle','off');
 result_s= drawAndFillPolygon(result_scaling,ctrlPointList_72,outlineVertexList_s,true,true,true);

 imwrite(result_s, 'partb_scaling_result_SamplingRate_72_high.png');
```

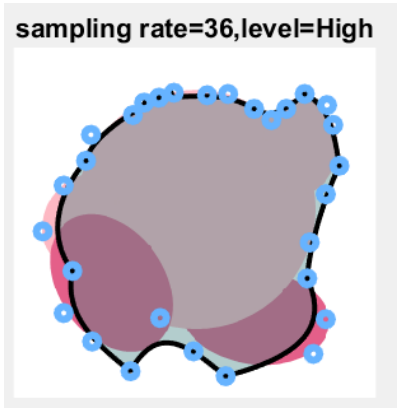
(1)用 imresize 把原圖指定用 Nearest-neighbor interpolation 的方法放大 4 倍。
(2)把 control points 都乘以 4
(3)呼叫 ComputeBezier function 並傳入對應參數。
(4)畫 polygon.

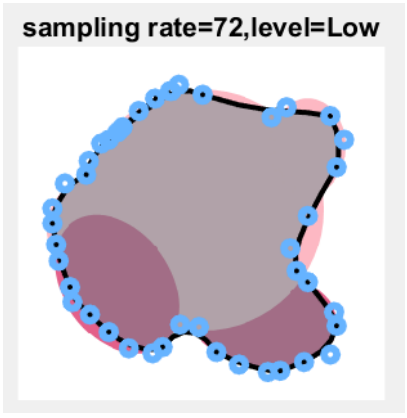
(c)

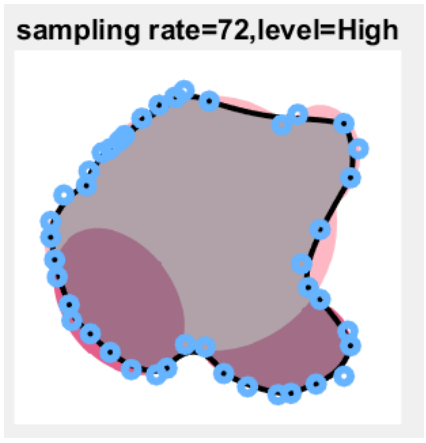
1. Discuss the results between different sampling rates and different levels of detail.

(1)結果





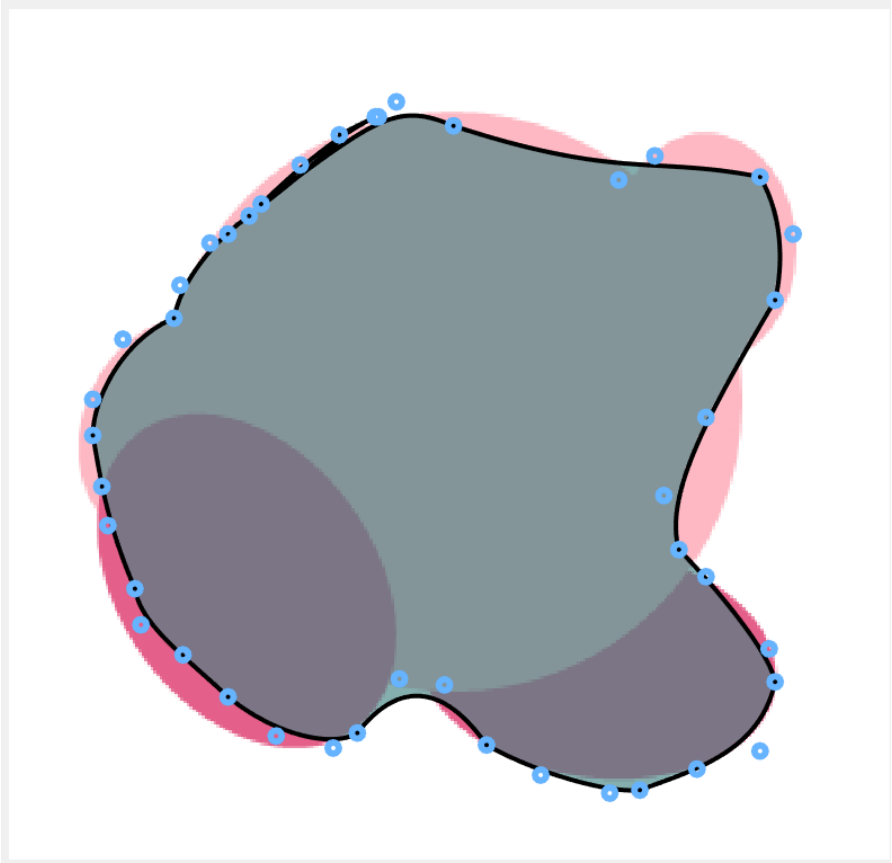




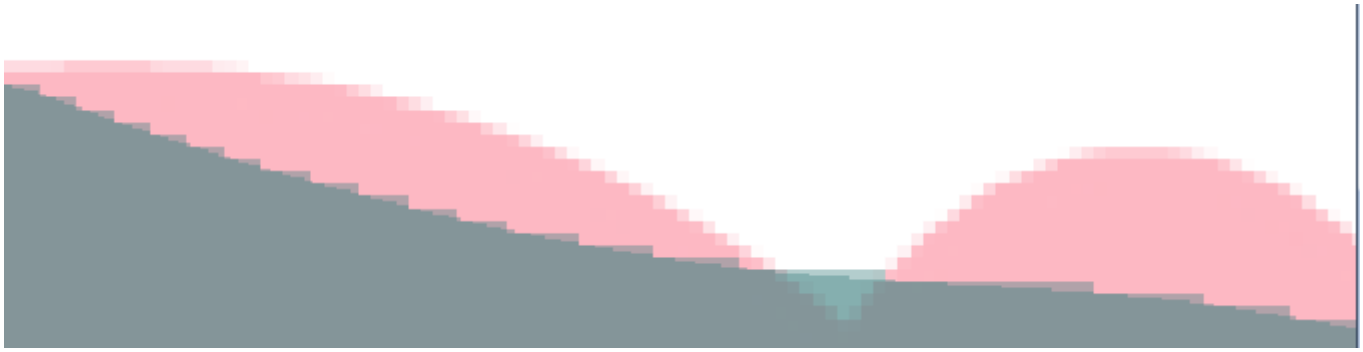
(1)Sampling rate 數量越多，每 2 點間距越短→同 Levels of detail 下，Sampling rate 越多，曲線越曲。
(2) Levels of detail 越小，代表在 4 個點中用較多點去逼近→同 Sampling rate 下，Levels of detail 越多，越逼近一條曲線。

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

圖一是放大 4 倍後的結果
圖二是擷取局部的圖



圖一



圖二

放大 control points 的效果比單純把圖放大 4 倍好。

因為以 bitmap 來放大→邊界易有鋸齒狀；用 vector 鋸齒狀就比較少。

單傳把圖放大相當於使用 bitmap 放大的方法，而放大 control points 相當於用 vector 的方法。

Part2

1.做法

(1)part2_makeRGBCube.m

```
%% Side faces (Your efforts here)
for vertI = 1 : 4
    faceVert1 = topVertIndex( mod(vertI,4)+1 );
    faceVert2 = topVertIndex( vertI );
    faceVert3 = botVertIndex( vertI );

    faceVert4 = botVertIndex( mod(vertI,4)+1);
    faceVert5 = botVertIndex( vertI);
    faceVert6 = topVertIndex( mod(vertI,4)+1);
    faces = [ faces ; faceVert1 faceVert2 faceVert3; faceVert4 faceVert5 faceVert6];
end
```

原本已經有上半部的三角形，這裡在拼出下半部的。(2 點在下 1 點在上)

(2)part2_readOBJFile.m

照 spec 的步驟做

```
modelV(:, 1)=vertex(:, 1)-center(1);
modelV(:, 2)=vertex(:, 2)-center(2);
modelV(:, 3)=vertex(:, 3)-center(3);
```

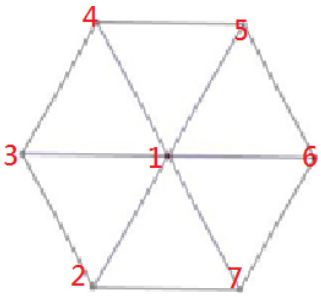
(b) shift this object's center to (0, 0, 0).

```
vertsX=cosd(30);
vertsY=sind(30);

topVs = [ vertsY 1 vertsX ;0 1 0; -vertsY 1 vertsX; 0 1 vertsX*2; 1 1 vertsX*2; 1.5 1 vertsX; 1 1 0 ];
botVs = [vertsY 0 vertsX ;  0 0 0; -vertsY 0 vertsX; 0 0 vertsX*2; 1 0 vertsX*2; 1.5 0 vertsX; 1 0 0];
```

(c)拼出六角柱
先把六角柱 top 的 vertex 跟 bottom 的位置指定好

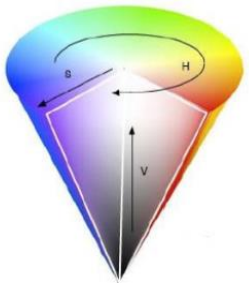
```
%top
hf = [];
f1 = topVInd(1);
f2 = topVInd(2);
f3 = topVInd(3);
hf = [ hf ; f1 f2 f3 ];
f1 = topVInd(1);
f2 = topVInd(3);
f3 = topVInd(4);
hf = [ hf ; f1 f2 f3 ];
f1 = topVInd(1);
f2 = topVInd(4);
f3 = topVInd(5);
hf = [ hf ; f1 f2 f3 ];
f1 = topVInd(1);
f2 = topVInd(5);
f3 = topVInd(6);
hf = [ hf ; f1 f2 f3 ];

f1 = topVInd(1);
f2 = topVInd(6);
f3 = topVInd(7);
hf = [ hf ; f1 f2 f3 ];
hf = [ hf ; f1 f2 f3 ];
f1 = topVInd(1);
f2 = topVInd(7);
f3 = topVInd(2);
hf = [ hf ; f1 f2 f3 ];
hf = [ hf ; f1 f2 f3 ];
```

畫出 top 的六邊形



(1)由 6 個三角形組成，並指定給每個 vertex 一個 index(如上圖)
(2)用 3 個 vertexs 組成一個三角形並將三角形放入 hf。
(3)側邊跟 bottom 的六邊形做法概念差不多。(就不放 code 占版面了)

```
Angle=linspace(0,2*pi,numVert+1)';
topColor=[0,0,1];
botColor=[0,0,0];
for i=1:numVert
    topColor=[topColor;Angle(i)/(2*pi),1,1];
    botColor=[botColor;Angle(i)/(2*pi),1,0];
end
vertColors = [ topColor; botColor ];
vertColors_final=hsv2rgb(vertColors);
```



跟據 HSV 的色彩圖，S 為 0-1，v 為 0-1，h 以角度分，最後轉成 RGB 輸出。

```
cen= [(max(verts(:, 1))+min(verts(:, 1)))/2, (max(verts(:, 2))+min(verts(:, 2)))/2, (max(verts(:, 3))+min(verts(:, 3)))/2];
v(:, 1)=verts(:, 1)-cen(1);
v(:, 2)=verts(:, 2)-cen(2)-1.4;
v(:, 3)=verts(:, 3)-cen(3);
```

(d)Shift the center of hexagonal prism to (0, -1.4, 0)，
並把 2 個 models 合在一起顯示

```
s=size(tval);
figure;
model = [model_vertex; v];
f = [faces;hf+s(1)];
c = [colors; vertColors_final];
trisurf(f,model(:,1),model(:,2),model(:,3),'FaceVertexCData', c,'FaceColor','interp', 'EdgeAlpha', 0);
xlabel('X'); ylabel('Y'); zlabel('Z');
title('(2d) result');
```

把六角柱+(川普模型的頂點數)後用 trisurf 顯示。

*tval=639*3(頂點數=639)

```
figure;
direction = light('Position',[0.0,0.0,5.0]);
lighting phong;
trisurf(f,model(:,1),model(:,2),model(:,3),'FaceVertexCData', c,'FaceColor','interp', 'EdgeAlpha', 0);
xlabel('X'); ylabel('Y'); zlabel('Z');
title('direction light');
```

(e) Adding different light sources (positional light and directional light)

左邊為 directional light。

Light Objects

You create a light object using the `light` function. Three important light object properties are

- `Color` — Color of the light cast by the light object
- `Style` — Either infinitely far away (the default) or local
- `Position` — Direction (for infinite light sources) or the location (for local light sources)

Style 部分若不指定為 local，default 是 infinitely far away 光(就是 directional light)

```
figure;
trisurf(f,model(:,1),model(:,2),model(:,3),'FaceVertexCData', c,'FaceColor','interp', 'EdgeAlpha', 0);
light('Position',[3.0,0.0,2.0],'Style','local');
xlabel('X'); ylabel('Y'); zlabel('Z');
title('position-light');
```

(e) Adding different light sources (positional light and directional light)

左邊為 positional light。

Style 指定為 local，Position 表示 the location of the light source。

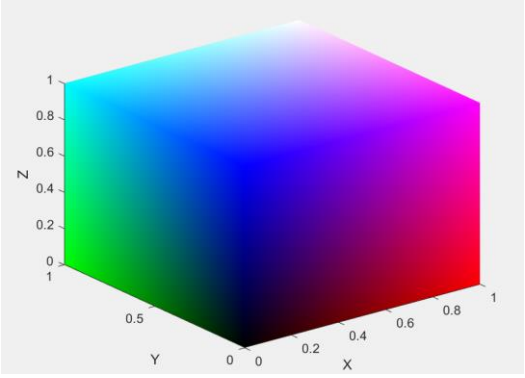
```
%I. (ka , kd , ks) = (1.0, 0.0, 0.0)
figure;
trisurf(f,model(:,1),model(:,2),model(:,3),'FaceVertexCData', c,'FaceColor','interp', 'EdgeAlpha', 0,'AmbientStrength',1.0,'DiffuseStrength',0.0,'SpecularStrength',0.0);
light('Position',[0.0,0.0,4.0],'Style','local');
xlabel('X'); ylabel('Y'); zlabel('Z');
title('I. (ka , kd , ks) = (1.0, 0.0, 0.0)');
```

(f) Also adjusting different ambient strength ka, diffuse strength kd, specular strength ks.

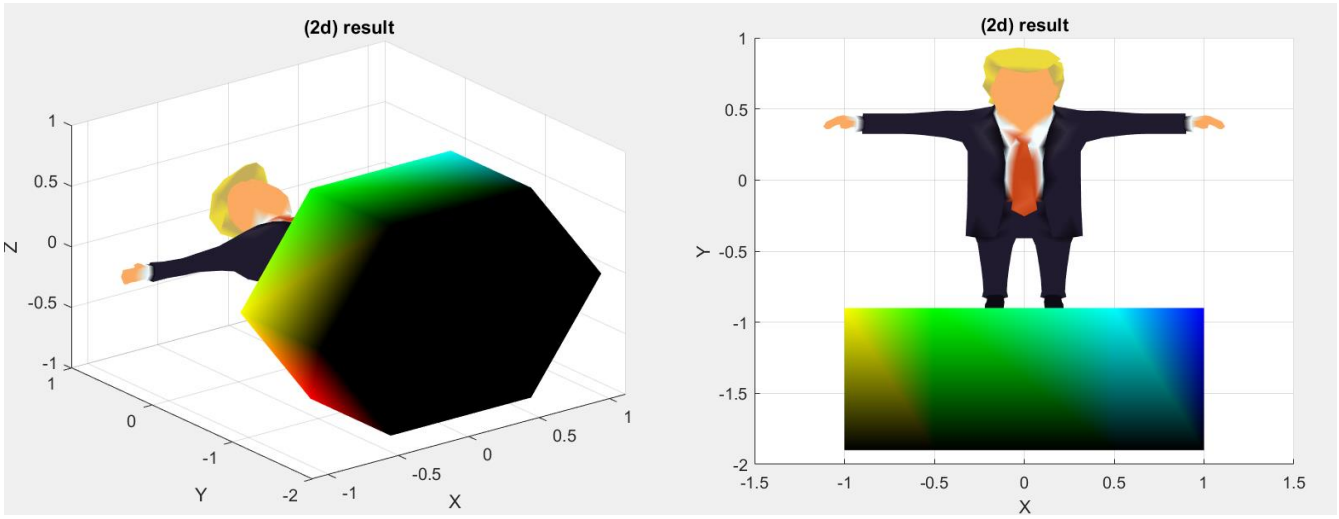
在 trisurf 裡加入'AmbientStrength',1.0,'DiffuseStrength',0.0,'SpecularStrength',0.0 即可。

2.結果

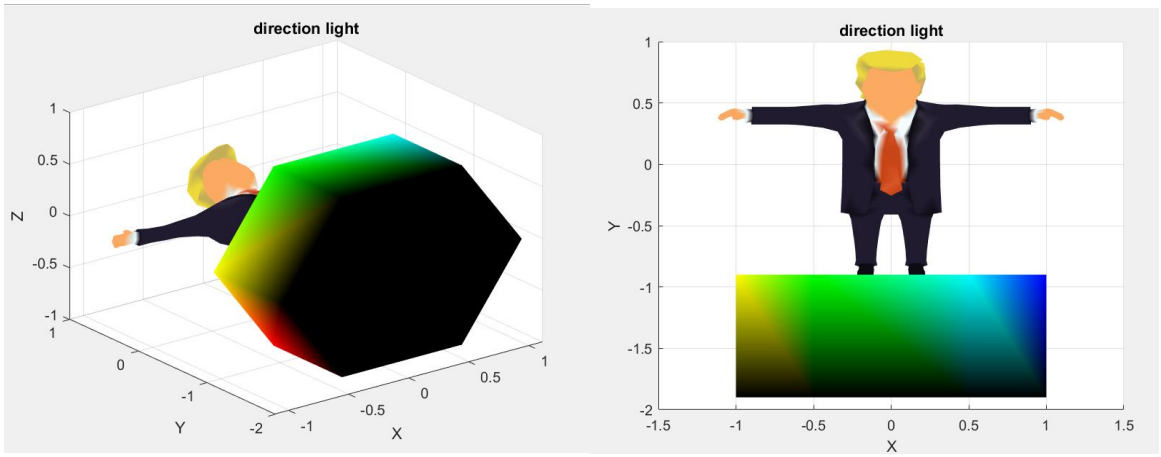
(a)

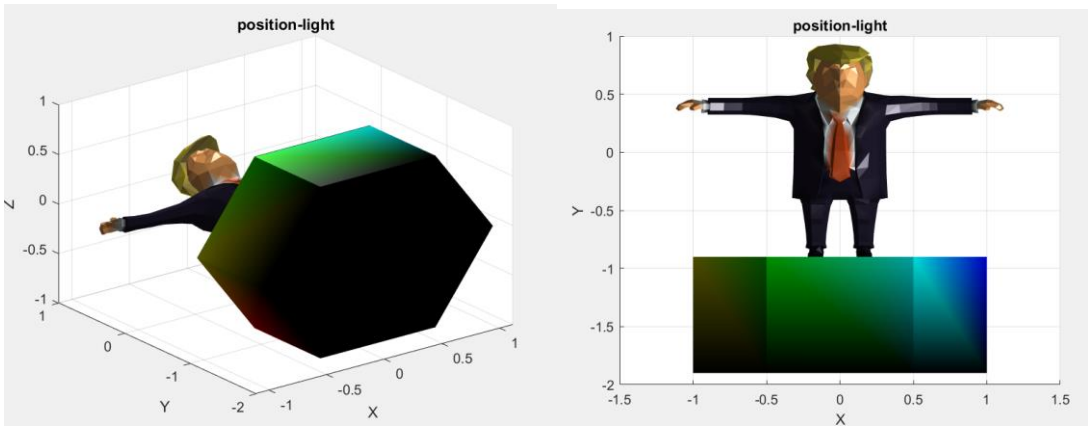


(d)

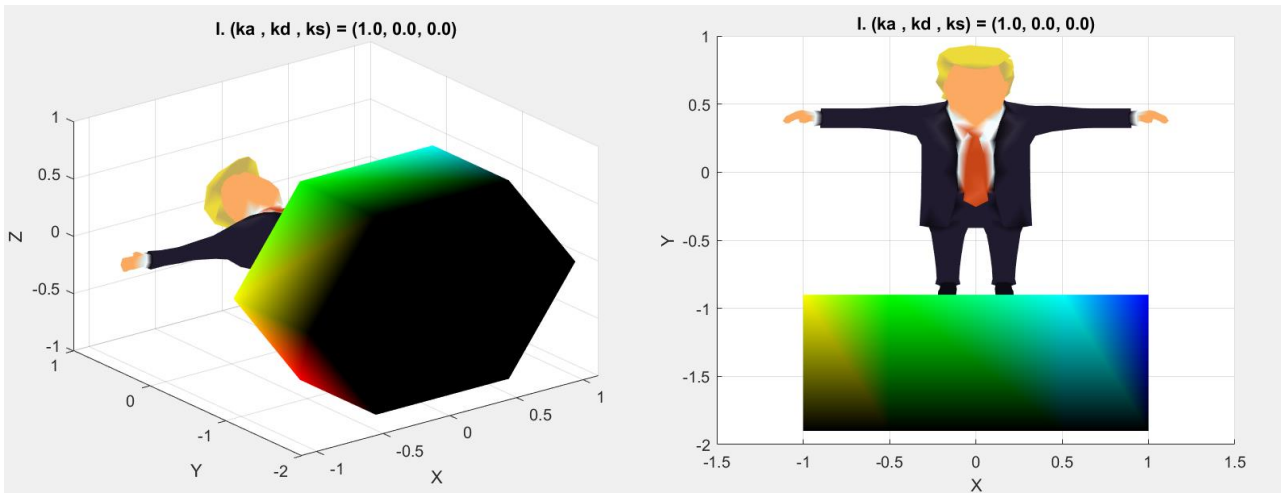

(2d) result
(2d) result

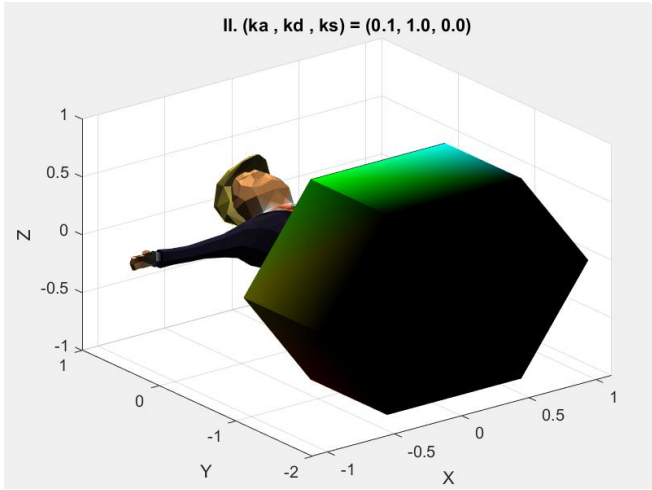
(e)

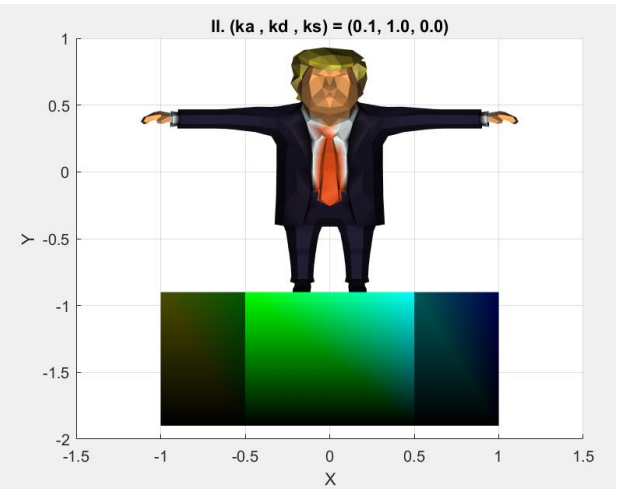

direction light
direction light

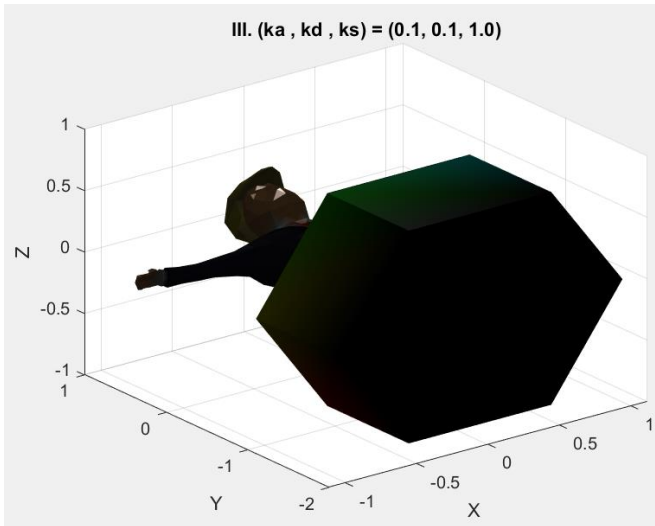

position-light
position-light

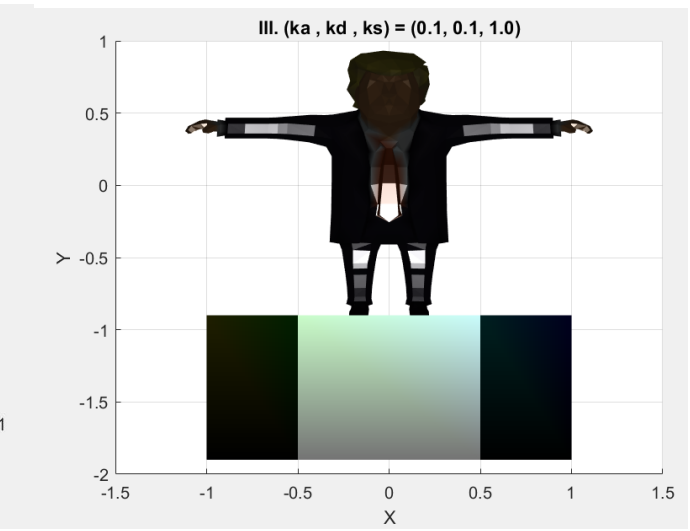
(f)

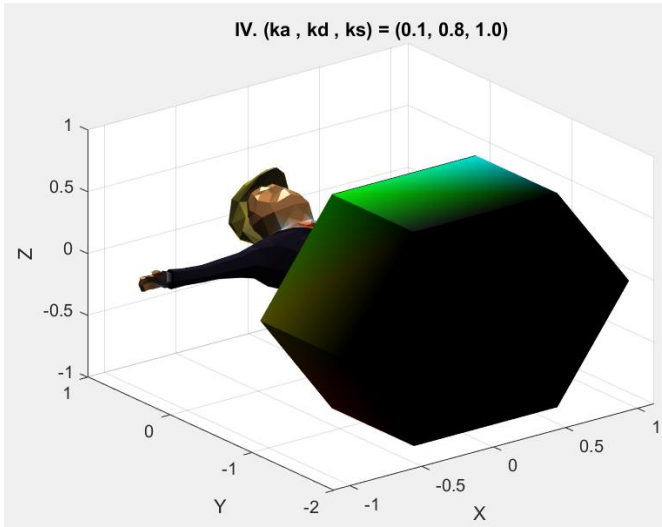

I. (ka , kd , ks) = (1.0, 0.0, 0.0)
I. (ka , kd , ks) = (1.0, 0.0, 0.0)

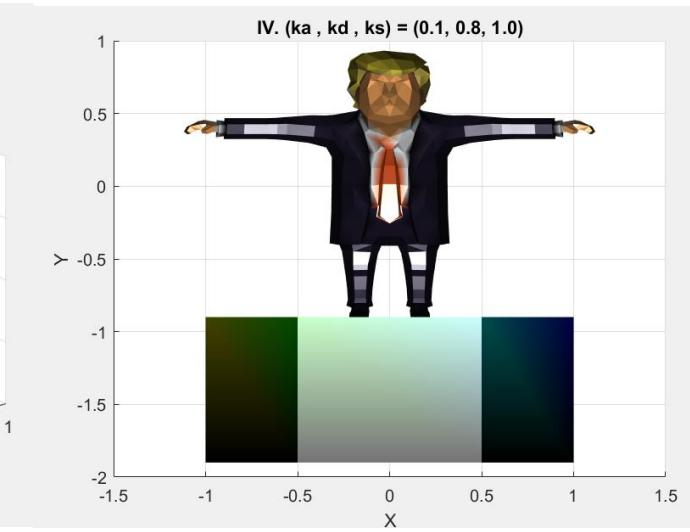

II. (ka , kd , ks) = (0.1, 1.0, 0.0)
Z
Y
X


II. (ka , kd , ks) = (0.1, 1.0, 0.0)
Y
X


III. (ka , kd , ks) = (0.1, 0.1, 1.0)
Z
Y
X


III. (ka , kd , ks) = (0.1, 0.1, 1.0)
Y
X


IV. (ka , kd , ks) = (0.1, 0.8, 1.0)
Z
Y
X


IV. (ka , kd , ks) = (0.1, 0.8, 1.0)
Y
X