Computer Vision
Homework 01: Radiometry, light and color
CMP SCI 670, Fall 2019

Name: Kunjal Panchal
Student ID: 32126469
Email: kpanchal@umass.edu

September 16, 2019

Contents

1 Problem 1

1.1 Definitions

Ambient Component: accounts for the small amount of light that is scattered about the entire scene. [1]

Diffuse Component: light is reflected equally in all directions. [2]

Specular Component: responsible for sharp mirror-like reflection from a surface. Some fraction is absorbed, some reflected. On real surfaces, energy usually goes into a lobe of directions, which is mathematically depcited by the angle θ between viewing and radiance vector and the amount of power/intensity of the lobe, which is depicted by the exponent n [2]

Phong Model: is an empirical model of local illumination. It describes the way a surface reflects light as a combination of the diffuse reflection of rough surfaces with the specular reflection of shiny surfaces. It is based on Phong's informal observation that shiny surfaces have small intense specular highlights, while dull surfaces have large highlights that fall off more gradually. [1]

Orthographic Projection: projection of a single view of an object (such as a view of the front) onto a drawing surface in which the lines of projection are perpendicular to the drawing surface **or** the representation of related views of an object as if they were all in the same plane and projected by orthographic projection. [3]

Albedo: is the measure of the diffuse reflection of solar radiation out of the total solar radiation received by an astronomical body. It is dimensionless and measured on a scale from 0 to 1. Surface albedo is defined as the ratio of radiosity to the irradiance received by a surface. [4]

1.2 The Setup

This section talks about the world. It includes, what constraints are used to create the solution of the given problem and how they are dealt with:

1.2.1 Light Source

These are some properties of the light source:

- STYLE Point source
- POSITION Infinitely away from sphere
- COLOR Red (for clear distinguishen between object surface material color and light color)
- INTENSITY 1

We implemeted this in Matlab code* as follows:

```
%% Light Source
l1 = light;
l1.Color = 'red';
l1.Style = 'infinite';

%% An alternative
% light 1
l2 = light('Style', 'infinite', 'Color', lightColor);
```

* complete code in Appendix A

1.2.2 Sphere

These are some properties of the sphere:

- RADIUS 1 meter
- POSITION At (0, 0, 0) of the co-ordinate system
- COLOR Blue (for clear distinguishen between object surface material color and light color)
- ALBEDO 1

We implemeted this in Matlab code* as follows:

```
%% Sphere
[X, Y, Z] = sphere;

%% Pure lambertian Surface
surf(X, Y, Z, 'EdgeColor','none', 'FaceColor', 'blue',
    'SpecularColorReflectance',0.0,'SpecularExponent'
    ,1);

%% An alternative for Phong model parameters
s1 = surf(X, Y, Z);
set(s1, 'FaceLighting','phong', 'FaceColor',
   surfaceType, 'EdgeColor','none', 'AmbientStrength',
   ka, 'DiffuseStrength',kd, 'SpecularStrength',ks, '
   SpecularExponent',ke, 'SpecularColorReflectance',
   scr, 'BackFaceLighting','unlit');
```

* complete code in Appendix A

NOTE: Matlab implementation of the sphere is generating faceted planes to make a sphere, this will result in the whole of tiny constituent plane having the same specularity. But from the neighboring planes, we can get idea of how actually a smoother implementation of the sphere will look.

Now, we have to render the sphere with 1 meter radius, away from a point light source at infinity. The assumption is that this is orthographic projection.

We implemeted rendering properties in Matlab code* as follows:

```
% graphics rendering set up
set(gcf, 'RendererMode', 'manual');
set(gcf, 'Renderer', 'zbuffer');
% set(gcf, 'Renderer', 'OpenGL');
```

* complete code in Appendix A

We will see two cases of suface properties and their effect on sphere rendering in the next following two subsections:

1.3 (a) Pure Lambertian Surface

For this kind of surface, the Specular Reflectance of the object surface material will be 0.0 (this component can range from 0.0 to 1.0). Thus, this surface is entirely dull or matte and doesn't have any shiny component in it.

```
surf(X, Y, Z, 'EdgeColor','none', 'FaceColor', 'blue',
    'SpecularColorReflectance',0.0,'SpecularExponent'
    ,1);
```

This shouldn't reflect light at all, as we can see from the results in Figure 1 (a), the orthographic rendering is shown in Figure 2(a), Figure 3(a) and Figure 4(a) too

1.4 (b) Specular surface rendered using the Phong reflection model with varying exponents (n in $\cos^n(\theta)$)

When we changed specular reflectance to 0.5 and then to 1.0 (any value (0.0, 1.0] will shine/reflect some light); we could see that the point light source at infinity was getting reflected according to the n of its $\cos^n(\theta)$) term, see Figures 1,2, 3, 4(b)-(h) and Figures 5, 6, 7, 8

1.4.1 Phong model

Let's see how we actually implemented the Phong Model:

The Matlab code* is as follows:

```
%% Surface properties
s1 = surf(X, Y, Z); % object handle for sphere 1
set(s1, 'FaceLighting','phong', 'FaceColor',
    surfaceType, 'EdgeColor','none', 'AmbientStrength',
    ka, 'DiffuseStrength',kd, 'SpecularStrength',ks, '
    SpecularExponent',ke, 'SpecularColorReflectance',
    scr, 'BackFaceLighting','unlit');
```

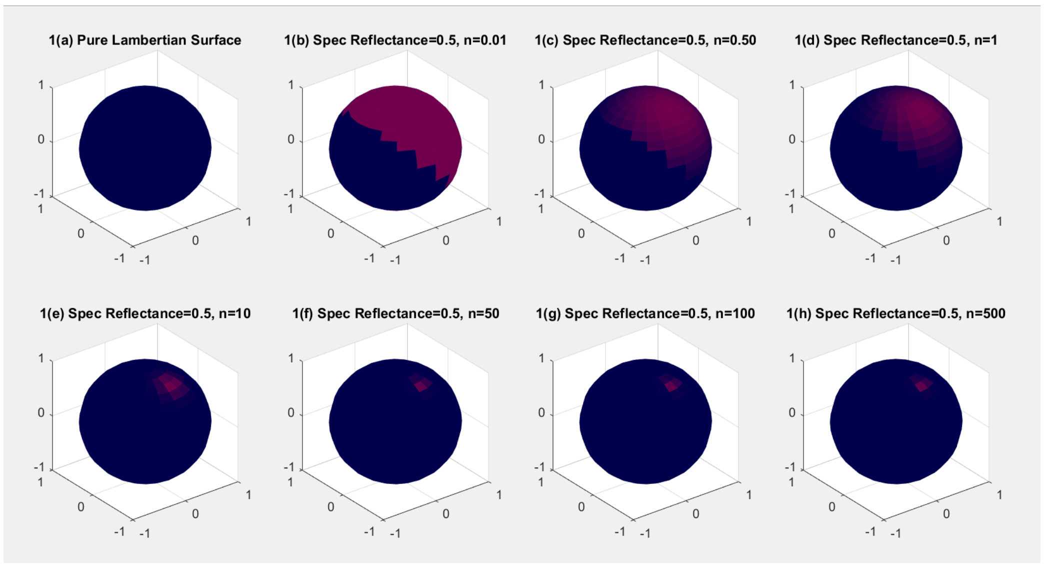

1(a) Pure Lambertian Surface
1(b) Spec Reflectance=0.5, n=0.01
1(c) Spec Reflectance=0.5, n=0.50
1(d) Spec Reflectance=0.5, n=1
1(e) Spec Reflectance=0.5, n=10
1(f) Spec Reflectance=0.5, n=50
1(g) Spec Reflectance=0.5, n=100
1(h) Spec Reflectance=0.5, n=500

Figure 1: 3D rendering of (a) Pure Lamberitan Surface with specular reflectance of 0.0; (b)-(h) Specular surfaces with reflectance 0.5 and varying n exponent of $\cos^n(\theta)$ where θ is the angle between relfected light vector and viewing vector, both are used to measure specularity of a surface material

```

%% Actual Equations
% N is vertex normal vector at each point on surface
% L is light vector
% R is reflected light vector
% V is view vector
% ka = ambient strength/ constant
% kd = diffuse strength/ constant
% ks = specular strength/ constant
% ke = cosine exponent

amb = ka * colorMaterial;
diff = sum(kd * (N.*L) * colorMaterial * colorLight,
    lights);
spec = sum(ks * (R.*V)^ke * colorSurface * colorLight,
     lights);

color = amb + diff + spec;
```

* complete code in Appendix A

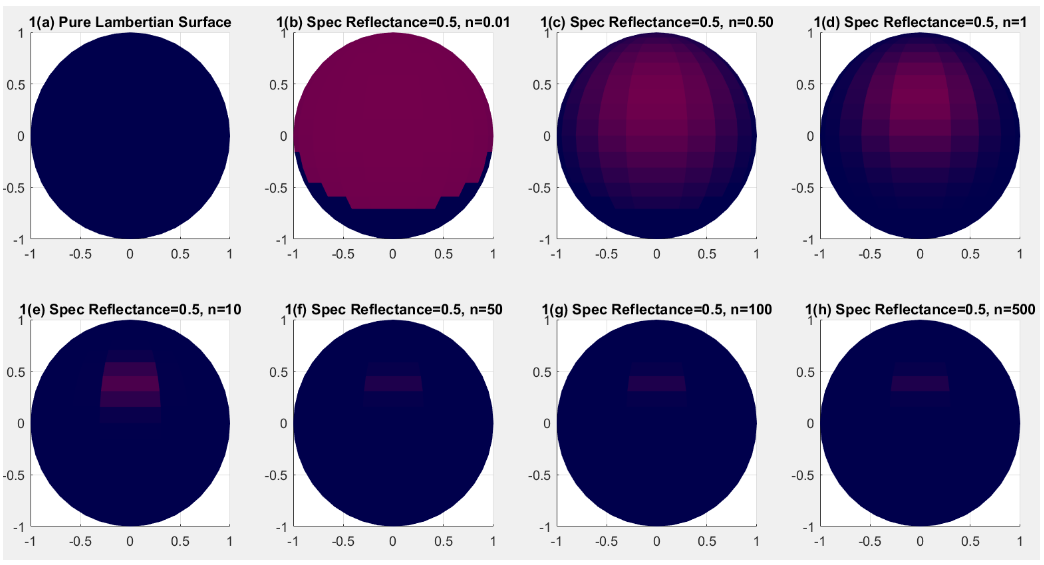

1(a) Pure Lambertian Surface
1(b) Spec Reflectance=0.5, n=0.01
1(c) Spec Reflectance=0.5, n=0.50
1(d) Spec Reflectance=0.5, n=1
1(e) Spec Reflectance=0.5, n=10
1(f) Spec Reflectance=0.5, n=50
1(g) Spec Reflectance=0.5, n=100
1(h) Spec Reflectance=0.5, n=500
1
0.5
0
-0.5
-1

Figure 2: 2D orthographic rendering of [camera near light source; showing Y-Z plane] (a) Pure Lamberitan Surface with specular reflectance of 0.0; (b)-(h) Specular surfaces with reflectance 0.5 and varying n exponent of $\cos^n(\theta)$ where θ is the angle between relfected light vector and viewing vector, both are used to measure specularity of a surface material

Note that, the 'phong' value has been deprecated, we should 'gouraud' instead. 'gouraud' varies the light across the faces, calculates the light at the vertices and then linearly interpolates the light across the faces. We can use this value to view curved surfaces. [8]

- On line 15 in the above code, ambient component just takes into account the color of the surface.

- On line 16, we have used "sum" for the cases where we have to deal with more than one light sources. It takes into account, the direction of irradience and the surface normal vector; alongside color of light and the surface material.

- On line 17, we have used "sum" for the same purpose mentioned above. The specular component takes into account the direction

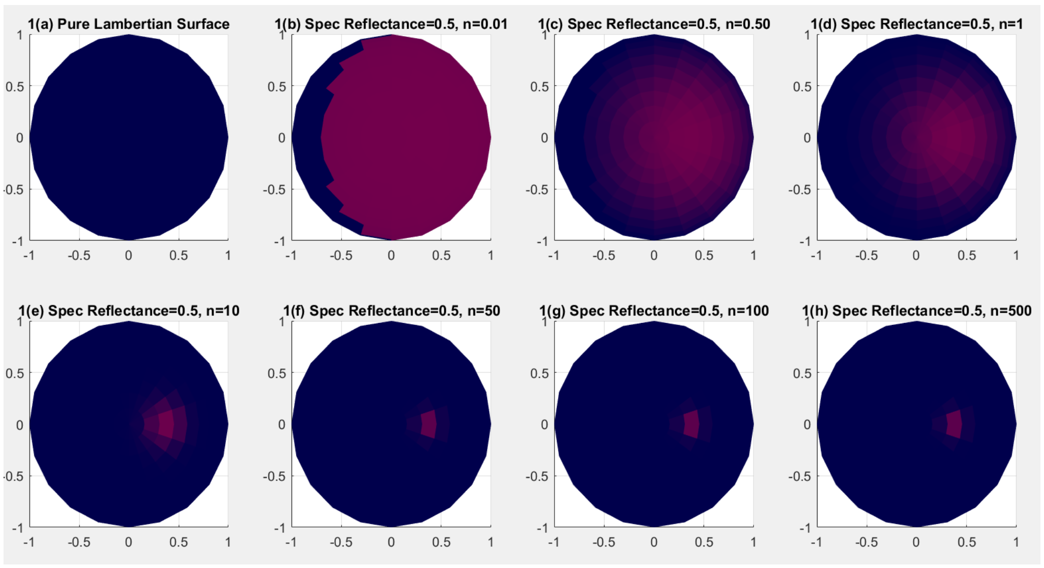


Figure 3: 2D orthographic rendering of [showing X-Y plane] (a) Pure Lamberitan Surface with specular reflectance of 0.0; (b)-(h) Specular surfaces with reflectance 0.5 and varying n exponent of $\cos^n(\theta)$ where θ is the angle between relfected light vector and viewing vector, both are used to measure specularity of a surface material

of radiance and the viewing direction. the θ we saw in $\cos^n(\theta)$ is the angle between these two vaectors. And the exponent ke is the n here.

Phong model is the addition of ambient, diffuse and specular components, hence, line 19. [6]

1.4.2 Observations

The observations are as follows:

- Naturally, as you go from 0.0 to 1.0 in value of specular component, the "shinyness" of the surface increases. See in Figure 1 and in Figure 5 how at the same n, the surface will reflect more light if the reflectance component is greater.
- n is in exponent of a cosine. Thus, the initial slight change in n will make drastic changes in the intensity of reflected light, but as the n reach higher values (n=100 from n=500), we won't be able to that much difference because of the inherent characteristics of the exponent.

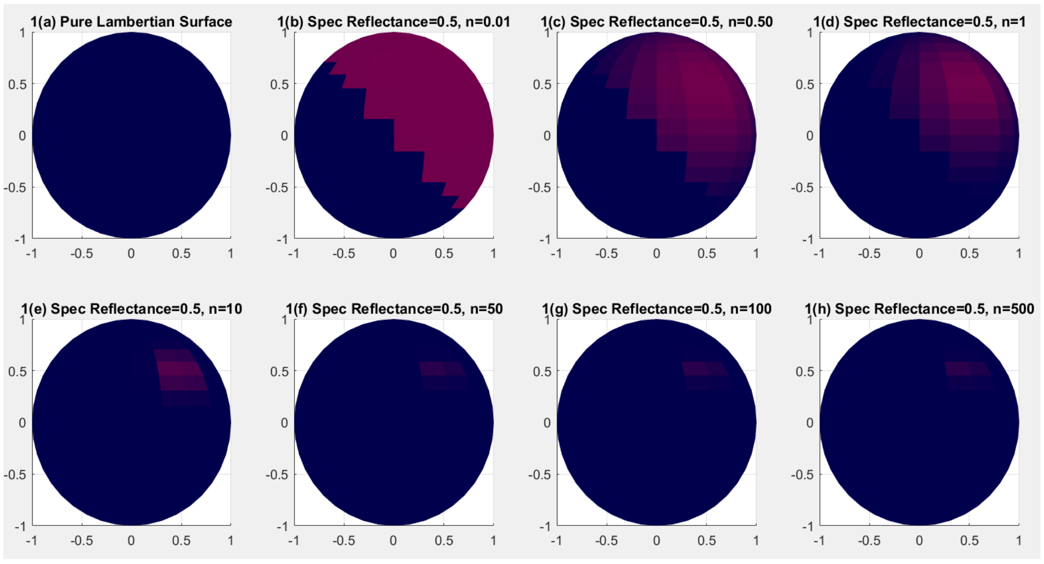

1(a) Pure Lambertian Surface
1(b) Spec Reflectance=0.5, n=0.01
1(c) Spec Reflectance=0.5, n=0.50
1(d) Spec Reflectance=0.5, n=1
1(e) Spec Reflectance=0.5, n=10
1(f) Spec Reflectance=0.5, n=50
1(g) Spec Reflectance=0.5, n=100
1(h) Spec Reflectance=0.5, n=500

Figure 4: 2D orthographic rendering of [showing X-Z plane] (a) Pure Lamberitan Surface with specular reflectance of 0.0; (b)-(h) Specular surfaces with reflectance 0.5 and varying n exponent of $\cos^n(\theta)$ where θ is the angle between relfected light vector and viewing vector, both are used to measure specularity of a surface material

- Value of $n < 1.0$ looks rather unnatural. As half of the sphere is shiny and the other half is not at all shining.

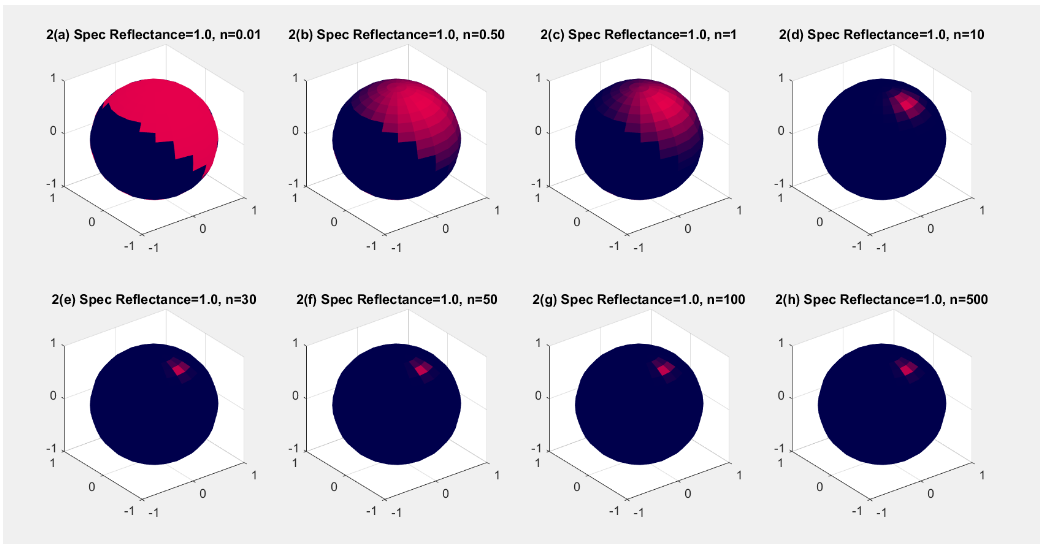

2(a) Spec Reflectance=1.0, n=0.01
2(b) Spec Reflectance=1.0, n=0.50
2(c) Spec Reflectance=1.0, n=1
2(d) Spec Reflectance=1.0, n=10
2(e) Spec Reflectance=1.0, n=30
2(f) Spec Reflectance=1.0, n=50
2(g) Spec Reflectance=1.0, n=100
2(h) Spec Reflectance=1.0, n=500

Figure 5: (a)-(h) Specular surfaces with reflectance 1.0 and varying n exponent of $\cos^n(\theta)$ where θ is the angle between relfected light vector and viewing vector, both are used to measure specularity of a surface material

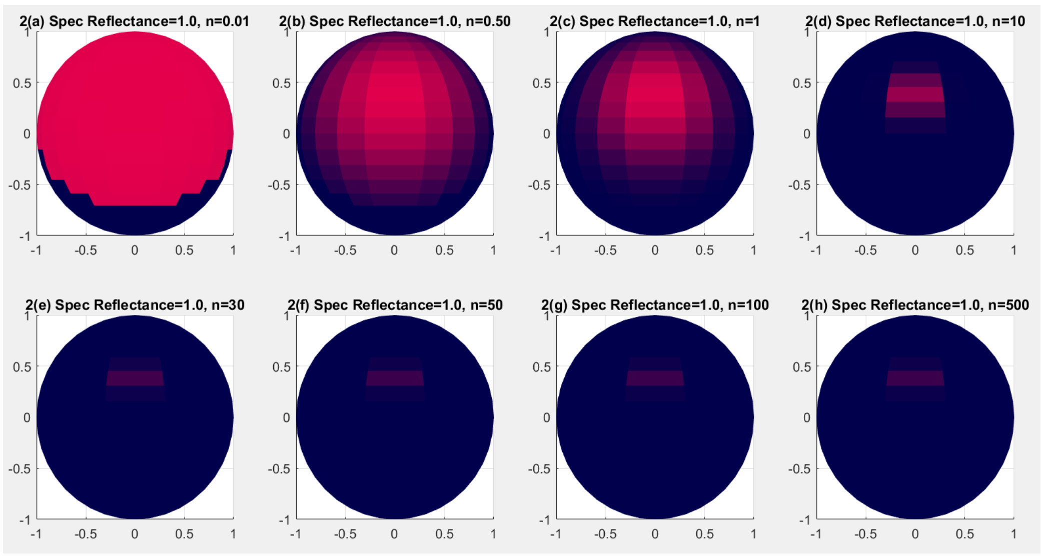

2(a) Spec Reflectance=1.0, n=0.01
2(b) Spec Reflectance=1.0, n=0.50
2(c) Spec Reflectance=1.0, n=1
2(d) Spec Reflectance=1.0, n=10
2(e) Spec Reflectance=1.0, n=30
2(f) Spec Reflectance=1.0, n=50
2(g) Spec Reflectance=1.0, n=100
2(h) Spec Reflectance=1.0, n=500

Figure 6: 2D orthographic rendering of [camera near light source; showing Y-Z plane](a)-(h) Specular surfaces with reflectance 1.0 and varying n exponent of $\cos^n(\theta)$ where θ is the angle between relfected light vector and viewing vector, both are used to measure specularity of a surface material

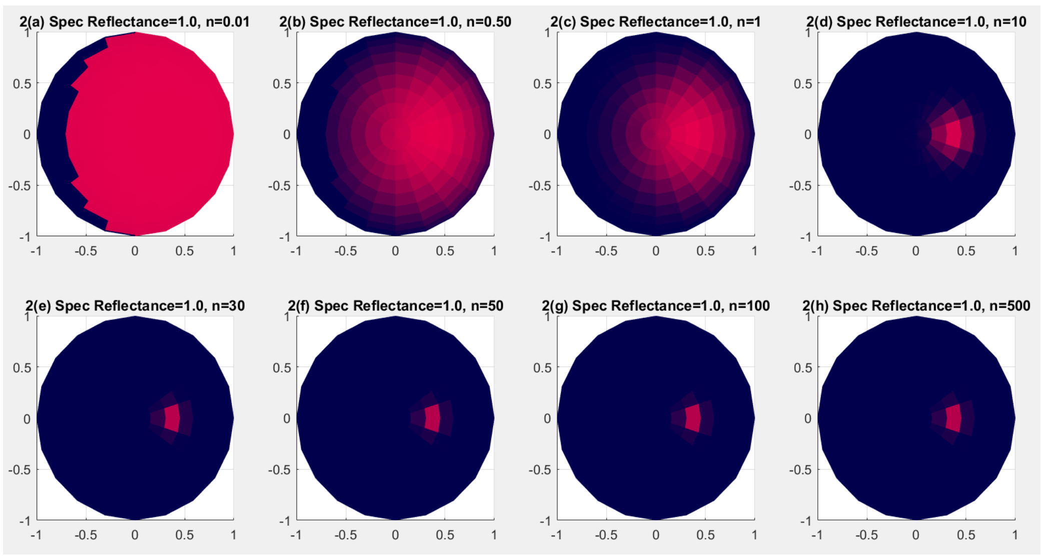


Figure 7: 2D orthographic rendering of [camera near light source; showing X-Y plane](a)-(h) Specular surfaces with reflectance 1.0 and varying n exponent of $\cos^n(\theta)$ where θ is the angle between relfected light vector and viewing vector, both are used to measure specularity of a surface material

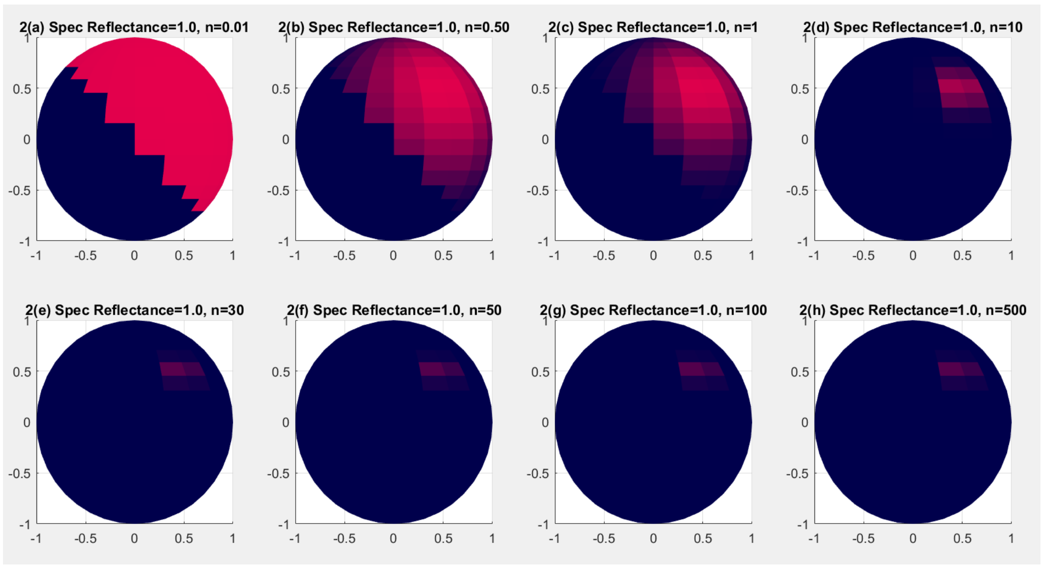

2(a) Spec Reflectance=1.0, n=0.01
2(b) Spec Reflectance=1.0, n=0.50
2(c) Spec Reflectance=1.0, n=1
2(d) Spec Reflectance=1.0, n=10
2(e) Spec Reflectance=1.0, n=30
2(f) Spec Reflectance=1.0, n=50
2(g) Spec Reflectance=1.0, n=100
2(h) Spec Reflectance=1.0, n=500
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1

Figure 8: 2D orthographic rendering of [camera near light source; showing X-Z plane](a)-(h) Specular surfaces with reflectance 1.0 and varying n exponent of $\cos^n(\theta)$ where θ is the angle between relfected light vector and viewing vector, both are used to measure specularity of a surface material

2 Problem 2

2.1 Definitions

Additive Colors: Additive color mixing explains how the eye interprets light wavelengths in the perception of color. It describes the color structure of light perception from four cardinal lights: red orange, middle green and blue violet, plus the white light (or white point) defined by mixing the three colored lights together. [5]

It predicts the resultant color by simply summing the numeric value of each individual color coinciding on others.

Lights work this way.

Subtractive Colors: Subtractive color mixing is, in comparison to additive color mixing, a flawed attempt to describe the colors that result when light absorbing substances are mixed. [5]

It gives the resultant value of power/energy distribution after light consecutively passes through layers of media which can absorb it relatively.

As stated above, "light absorbing substances" work this way. e.g Dyes, Inks, Paints.

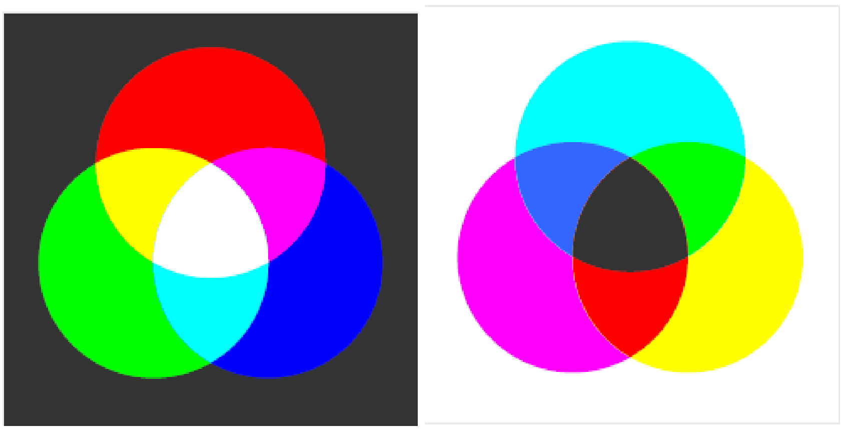
2.2 Colors

RGB color space: Red, Green and Blue; as shown in Figure 10, when added or "superimposed" together, results in addition of their wavelegth quantities and produces **white**.

This has nothing to do with wavelengths physically mixing up to create a new wavelength. In fact, this is the result of tri-stimulus theory where human vision has three kinds of cones which react to different wavelengths and intensities; producing an end result which is a color that can be described by its three components: RGB. Thus, any color can be reproduced by changing intensities of these three primary components.

CMYK color space: This color space can be useful in explaining the subtractive mixing we see with inks and paints.

These materials reflect light off of the paper, instead of absorbing and adding them. Thus, red colored surface would absorb green and blue lights, but incident red light is not absorbed and reflected back. [7]



(a) (b)

Figure 9: (a) Additive Colors (Primary Colors of RGB color space: Red, Green & Blue). (b) Subtractive Colors (Primary Colors of CMYK color space: Cyan, Magenta & Yellow). [6]

2.3 Answer to our question

When you are painting some material, with say, red, you are actually "painting" or "inking" it with a pigment or substance that **absorbs all other lights** except a red light. The substance will reflect only red light, thus you see that object or material as red colored.

Now, when you paint a green material with red, it will **not absorb red** light because of its green pigment and **not absorb green** light because of its red pigment. Thus, the name "subtrative inks". If you paint it with red, green and blue; the material or the paint layers end up absorbing **all** the colors, resulting in no light getting reflected, hence, material looking some dark/black colored.

2.4 Bottom line

In case of light, we are combining colors, not eliminating them. Mix of all additive light colors will be perceived white by eyes, because all lights are getting reflected.

In case of pigments, we are "removing" colors by putting layers of them on each other. Mix of all subtractive pigment colors will be perceived as some dark-ish(brownish-gray) or black color by eyes, because no light is getting reflected.

Paints do not reflect light, it particularly chooses a color to reflect. They act as filters.

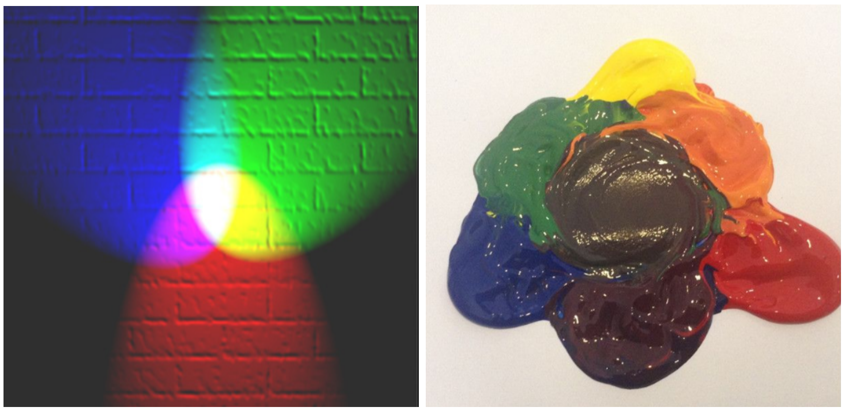

(a) (b)

Figure 10: (a) Mixing Red, Green and Blue lights; the produced result is a white light. [9] (b) Mixing Red, Green and Blue pigments; the produced result is a dark pigment.

References

[1] https://en.wikipedia.org/wiki/Phong_reflection_model

[2] Subhransu Maji *Computer Vision Lecture 02: Radiometry* https://www.dropbox.com/s/0rtuxli868hcwq9/lec02_radiometry.pdf?dl=0

[3] Merriam-Webster Dictionary https://www.merriam-webster.com/dictionary/orthographic\%20projection

[4] Pharr & Humphreys *"Physically Based Rendering": Fundametals of Rendering - Radiometry / Photometry*

[5] Bruce MacEvoy, *color vision:additive & subtractive color mixing* https://www.handprint.com/HP/WCL/color5.html

[6] Richard Szeliski, *Computer Vision: Algorithms and Applications* ISBN 978-1-84882-935-0

[7] David A. Forsyth, Jean Ponce, *Computer Vision: A Modern Approach (2nd Edition)* ISBN 978-0-13608-592-8

[8] Mathworks Documentation of 'surf', https://www.mathworks.com/help/matlab/ref/surf.html

[9] RGB Mix Paint https://www.pinterest.com/pin/28077197654741105/

A
Matlab code for Problem 1

A.1 Applying phong model

```matlab
function color: PhongModel(colorMaterial, colorLight,
   colorSurface, ka, kd, ks, ke, lights, N, L, R, V)

% N is vertex normal vector at each point on surface
% L is light vector
% R is reflected light vector
% V is view vector

amb = ka * colorMaterial;
diff = sum(kd * (N.*L) * colorMaterial * colorLight,
   lights);
spec = sum(ks * (R.*V)^ke * colorSurface * colorLight,
    lights);

color = amb + diff + spec;
end
```

A.2 Pure Lambertian and Specular Surfaces generated with properties of Sphere

```matlab
clc; clear all; close all;

subplot(241);
%%Sphere rendering
[X, Y, Z] = sphere;

%%Pure lambertian Surface
surf(X, Y, Z, 'EdgeColor','none', 'FaceColor', 'blue',
    'SpecularColorReflectance',0.0,'SpecularExponent'
   ,1);
title('1(a) Pure Lambertian Surface');

%%Light Source
l = light; l.Color = 'red'; l.Style = 'infinite';

```

```
%%Specular Reflectance = 0.5, varying n
subplot(242);
surf(X, Y, Z, 'EdgeColor','none', 'FaceColor', 'blue',
     'SpecularColorReflectance',0.5,'SpecularExponent'
    ,0.01);
title('1(b) Spec Reflectance=0.5, n=0.01');

%%Light Source
l = light; l.Color = 'red'; l.Style = 'infinite';

subplot(243);
surf(X, Y, Z, 'EdgeColor','none', 'FaceColor', 'blue',
     'SpecularColorReflectance',0.5,'SpecularExponent'
    ,0.50);
title('1(c) Spec Reflectance=0.5, n=0.50');

%%Light Source
l = light; l.Color = 'red'; l.Style = 'infinite';

subplot(244);
surf(X, Y, Z, 'EdgeColor','none', 'FaceColor', 'blue',
     'SpecularColorReflectance',0.5,'SpecularExponent'
    ,1);
title('1(d) Spec Reflectance=0.5, n=1');

%%Light Source
l = light; l.Color = 'red'; l.Style = 'infinite';

subplot(245);
surf(X, Y, Z, 'EdgeColor','none', 'FaceColor', 'blue',
     'SpecularColorReflectance',0.5,'SpecularExponent'
    ,10);
title('1(e) Spec Reflectance=0.5, n=10');

%%Light Source
l = light; l.Color = 'red'; l.Style = 'infinite';

subplot(246);
surf(X, Y, Z, 'EdgeColor','none', 'FaceColor', 'blue',
     'SpecularColorReflectance',0.5,'SpecularExponent'
    ,50);
```

```
title('1(f) Spec Reflectance=0.5, n=50');

%%Light Source
l = light; l.Color = 'red'; l.Style = 'infinite';

subplot(247);
surf(X, Y, Z, 'EdgeColor','none', 'FaceColor', 'blue',
    'SpecularColorReflectance',0.5,'SpecularExponent'
   ,100);
title('1(g) Spec Reflectance=0.5, n=100');

%%Light Source
l = light; l.Color = 'red'; l.Style = 'infinite';

subplot(248);
surf(X, Y, Z, 'EdgeColor','none', 'FaceColor', 'blue',
    'SpecularColorReflectance',0.5,'SpecularExponent'
   ,500);
title('1(h) Spec Reflectance=0.5, n=500');

%%Light Source
l = light; l.Color = 'red'; l.Style = 'infinite';

figure;
%%Specular Reflectance = 1.0, varying n
subplot(241);
surf(X, Y, Z, 'EdgeColor','none', 'FaceColor', 'blue',
    'SpecularColorReflectance',1.0,'SpecularExponent'
   ,0.01);
title('2(a) Spec Reflectance=1.0, n=0.01');

%%Light Source
l = light; l.Color = 'red'; l.Style = 'infinite';

subplot(242);
surf(X, Y, Z, 'EdgeColor','none', 'FaceColor', 'blue',
    'SpecularColorReflectance',1.0,'SpecularExponent'
   ,0.50);
title('2(b) Spec Reflectance=1.0, n=0.50');

%%Light Source
```

```
l = light; l.Color = 'red'; l.Style = 'infinite';

subplot(243);
surf(X, Y, Z, 'EdgeColor','none', 'FaceColor', 'blue',
    'SpecularColorReflectance',1.0,'SpecularExponent'
    ,1);
title('2(c) Spec Reflectance=1.0, n=1');

%%Light Source
l = light; l.Color = 'red'; l.Style = 'infinite';

subplot(244);
surf(X, Y, Z, 'EdgeColor','none', 'FaceColor', 'blue',
    'SpecularColorReflectance',1.0,'SpecularExponent'
    ,10);
title('2(d) Spec Reflectance=1.0, n=10');

%%Light Source
l = light; l.Color = 'red'; l.Style = 'infinite';

subplot(245);
surf(X, Y, Z, 'EdgeColor','none', 'FaceColor', 'blue',
    'SpecularColorReflectance',1.0,'SpecularExponent'
    ,30);
title('2(e) Spec Reflectance=1.0, n=30');

%%Light Source
l = light; l.Color = 'red'; l.Style = 'infinite';

subplot(246);
surf(X, Y, Z, 'EdgeColor','none', 'FaceColor', 'blue',
    'SpecularColorReflectance',1.0,'SpecularExponent'
    ,50);
title('2(f) Spec Reflectance=1.0, n=50');

%%Light Source
l = light; l.Color = 'red'; l.Style = 'infinite';

subplot(247);
surf(X, Y, Z, 'EdgeColor','none', 'FaceColor', 'blue',
    'SpecularColorReflectance',1.0,'SpecularExponent'
    ,100);
title('2(g) Spec Reflectance=1.0, n=100');

%%Light Source
l = light; l.Color = 'red'; l.Style = 'infinite';

subplot(248);
surf(X, Y, Z, 'EdgeColor','none', 'FaceColor', 'blue',
     'SpecularColorReflectance',1.0,'SpecularExponent'
    ,500);
title('2(h) Spec Reflectance=1.0, n=500');

%%Light Source
l = light; l.Color = 'red'; l.Style = 'infinite';
```

A.3 Tweaking more parameter of the phong model

```
function PhongShader(surfaceType, lightColor, ka, kd,
    ks, ke, scr)

figure;

[X, Y, Z] = sphere; % Sphere
s1 = surf(X, Y, Z); % object handle for sphere 1
s2 = surf(X-5, Y+5, Z); % object handle for sphere 2

% light 1
hL1 = light('Style', 'infinite', 'Color', lightColor);
% light 2
hL2 = light('Style', 'infinite', 'Color', lightColor);

set(s1, 'FaceLighting','phong', 'FaceColor',
    surfaceType, 'EdgeColor','none', 'AmbientStrength',
    ka, 'DiffuseStrength',kd, 'SpecularStrength',ks, '
    SpecularExponent',ke, 'SpecularColorReflectance',
    scr, 'BackFaceLighting','unlit');

set(s2, 'FaceLighting','phong', 'FaceColor','r', '
    EdgeColor','none', 'AmbientStrength',ka, '
    DiffuseStrength',kd, 'SpecularStrength',ks, '
    SpecularExponent',ke, 'SpecularColorReflectance',
    scr, 'BackFaceLighting','lit');

set(gca,'Fontsize',20);
title(sprintf('Surfaces: %s \n ka=%1.2f kd=%1.2f ks
    =%1.2f ke=%1.2f scr=%1.2f',surfaceType, ka, kd, ks,
     ke, scr));
```