Homework 2 – Image Formation

This homework purposes to create an image of hemisphere with Lambertian surface.

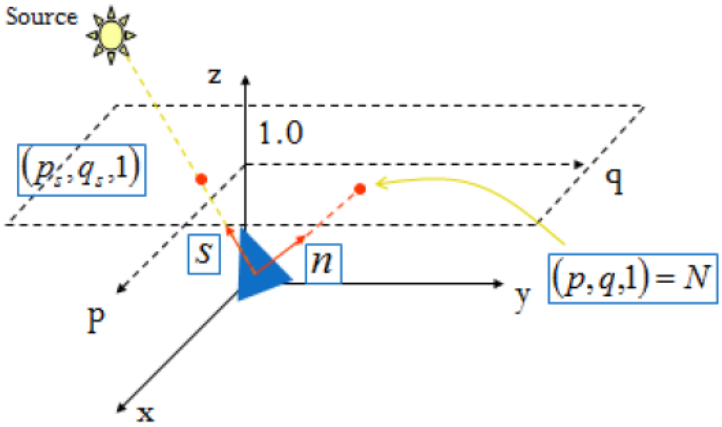
1. Lambertian Model

Lambertian Model is a basic model for generating images. The Lambertian Model is

$$I(x,y) = \rho\, \vec{n}\, \vec{s}$$

where $\vec{n}$ is the surface normal and $\vec{s}$ is the representation of light source direction and ρ is the albedo.

The model can be seen following image:



Surface normal can be represented as

$$n = \frac{N}{||N||} = \frac{(p,q,1)}{\sqrt{p^2+q^2+1}} \; where\; p = -\frac{\partial z}{\partial x} \; and \; q = -\frac{dz}{dy}$$

Source direction can be represented as

$$s = \frac{S}{||S||} = \frac{(p_s, q_s, 1)}{\sqrt{{p_s}^2+{q_s}^2+1}} \; where\; p_s = \frac{\partial z}{\partial x} \; at\; x = x_s \; and \; q_s = \frac{dz}{dy} \; at\; y = y_s$$

In the formula, x_s and y_s are the light sources coordinates.

For an image creating action, p_s and q_s is calculated once for a source location. Source direction is constant for entire scene is the assumption for the Lambertian Model.

2. Implementation of Lambertian Surface Model

Lambertian Surface Model is implemented for image creating by using "MATLAB".

i. Equation for creating an image of a hemisphere

$$z(x,y) = \sqrt{R^2 - (x - x_0)^2 - (y - y_0)^2}$$

Let's $x_0 = y_0 = 0$

Then equation will be as following:

$$z(x,y) = \sqrt{R^2 - x^2 - y^2}$$

MATLAB syntax for the equation is given as:

```
% Constants
R = 100;
% Variables
syms x y z
% Equation
z = sqrt(R^2-x^2-y^2);
```

ii. Calculating Source Direction

Unit vector of source direction is given as:

$$s = \frac{S}{||S||} = \frac{(p_s, q_s, 1)}{\sqrt{{p_s}^2 + {q_s}^2 + 1}} \; where \; p_s = \frac{\partial z}{\partial x} \; at \; x = x_s \; and \; q_s = \frac{dz}{dy} \; at \; y = y_s$$

Then calculation steps will be as following:

$p_s = \frac{\partial z}{\partial x}$

```
%dz/dx symbolic toolbox showing
ps_sym = (diff(z,'x'));
```

$q_s = \frac{dz}{dy}$

```
% dz/dy symbolic toolbox showing
qs_sym = (diff(z,'y'));%dz/dy
```

Assume $x = x_s = 0 \; and \;\; y = y_s = 0$

```
% Souce Location
x_light_source = 0;
y_light_source = 0;
```

Calculate s unit vector

```
%calculate ps and qs sybolic to numeric
ps = subs(ps_sym,[x,y],[x_light_source,y_light_source]);
qs = subs(qs_sym,[x,y],[x_light_source,y_light_source]);
% S vector
S = [ps,qs,1];
% s unit vector
s = S./ norm(S);
```

This vector will be calculated for once for an image

iii. Calculating Surface Normal Direction

$$n = \frac{N}{||N||} = \frac{(p,q,1)}{\sqrt{p^2+q^2+1}} \; where \; p = -\frac{\partial z}{\partial x} \; and \; q = -\frac{dz}{dy}$$

$$p = -\frac{\partial z}{\partial x}$$

```
%dz/dx symbolic toolbox showing
p_sym = -(diff(z,'x'));%dz/dx
```

$$q = -\frac{dz}{dy}$$

```
%dz/dx symbolic toolbox showing
q_sym = -(diff(z,'y'));%dz/dy
```

This vector must be calculated for every pixel of the image. Therefore image size is needed.

```
% Image Size
ImageSize = 2 * R;
```

Moreover, $I(x,y) = \rho \, \vec{n} \, \vec{s}$ vector must be calculated for every pixel of the image. Therefore I vector must be an array, so I vector is a row x column sized vector. The row and column sizes are assumed to be equal to Image Size.

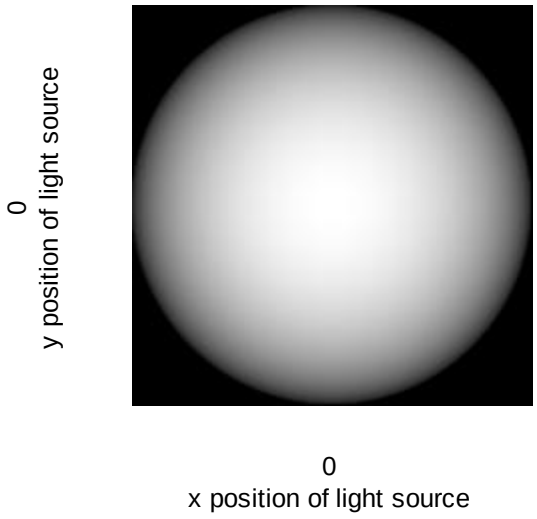
```
% Image Size
I = ones(ImageSize) * 255;
```

Calculating normal unit vector for every pixel of the image will be as following

```
% calculate normal unit vector for every pixel of the image
for row = 1:1:ImageSize
    for col = 1:1:ImageSize
        y_position = row - R;
        x_position = - col + R;
        if x_position * x_position + y_position * y_position < R * R
            p = subs(p_sym,[x,y],[x_position,y_position]);
            q = subs(q_sym,[x,y],[x_position,y_position]);
            n_xy = [p,q,1];
            n_xy = n_xy ./ norm(n_xy);
            I(row,col) = albedo * n_xy * s';
            if I(row, col) < 0
                I(row, col) = 0;
            end
        else
            I(row,col) = 0;
        end

    end
end
```

After $I(x,y) = \rho\, \vec{n}\, \vec{s}$ is calculated for all pixels while source locations are $x = x_s = 0$ *and* $y = y_s = 0$, by using following code, an image created:

```
figure;
imshow(I, []);
xlabel('x');
ylabel('y');
```



Symbolic Toolbox is not very effective to calculate $I(x,y) = \rho \vec{n} \vec{s}$

If symbolic calculations are removed, and values are implemented to the equations directly, another but more efficient code solution for creating image can be designed.

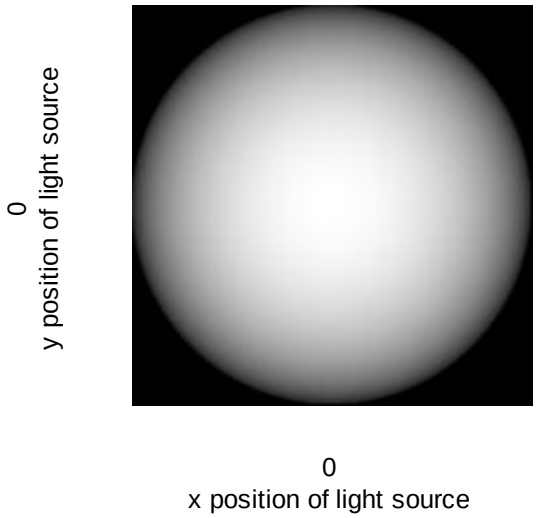
More effective software solution for creating image is given below:

```
% Constants
R = 100;
% souce Location
x_light_src = 0;
y_light_src = 0;

% Calculate ps and qs numeric
ps = -x_light_src/(- x_light_src^2 - y_light_src^2 +
R^2)^(1/2);%(diff(z,'x'));%dz/dx
qs = -y_light_src/(- x_light_src^2 - y_light_src^2 +
R^2)^(1/2);%(diff(z,'y'));%dz/dy

% S vector
S = [ps,qs,1];
% s unit vector
s = S./ norm(S);

% Image Size
ImageSize = 2 * R;

albedo = 255;

I = ones(ImageSize) * (albedo);

% Calculate normal unit vector for every pixel of the image
for row = 1:1:ImageSize
    for col = 1:1:ImageSize
        % calculate I(x,y)
        y_position = row - R;
        x_position = - col + R;
        % Shape boundaries
        if x_position * x_position + y_position * y_position < R * R
            p = x_position/(- x_position^2 - y_position^2 + R^2)^(1/2);%-
(diff(z,'x'));%dz/dx
            q = y_position/(- x_position^2 - y_position^2 + R^2)^(1/2);%-
(diff(z,'y'));%dz/dy
            n_xy = [p,q,1];
            n_xy = n_xy ./ norm(n_xy);
            I(row,col) = albedo * n_xy * s';
            % image is black if I(x,y) < 0
            if I(row, col) < 0
                I(row, col) = 0;
            end
        else
            I(row,col) = 0;
        end

    end
end
```

```
figure;
imshow(I, []);
xlabel('xeff');
ylabel('yeff');
```

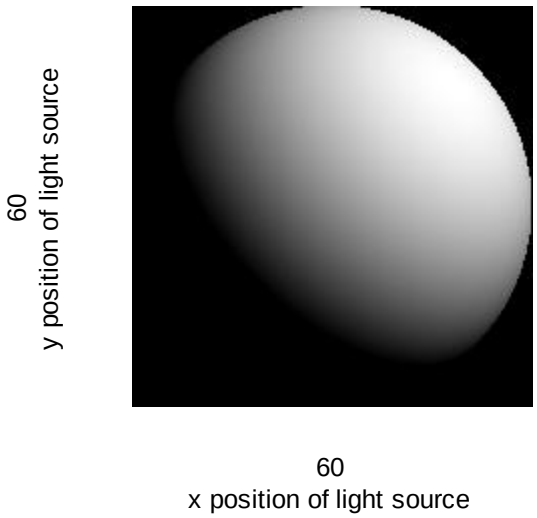
3. Results

i. Light Source x = 0; y = 0

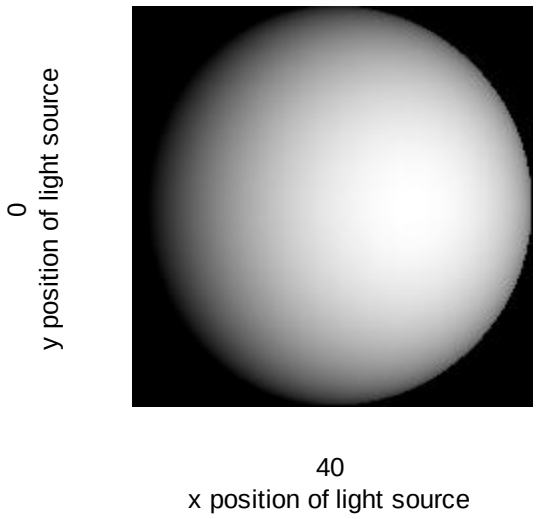


ii. Light Source x = 20; y = 20

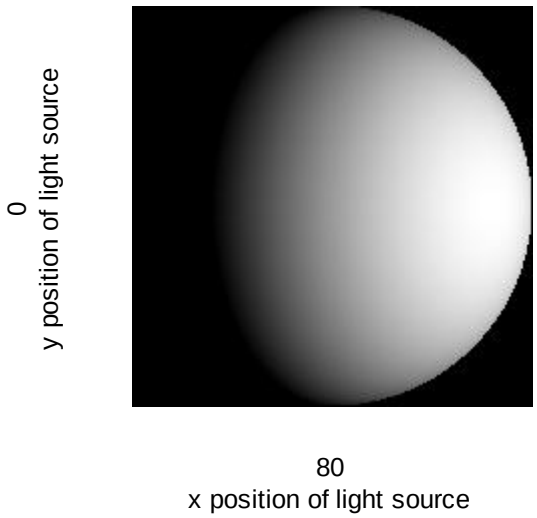
iii. Light Source x = 60; y = 60



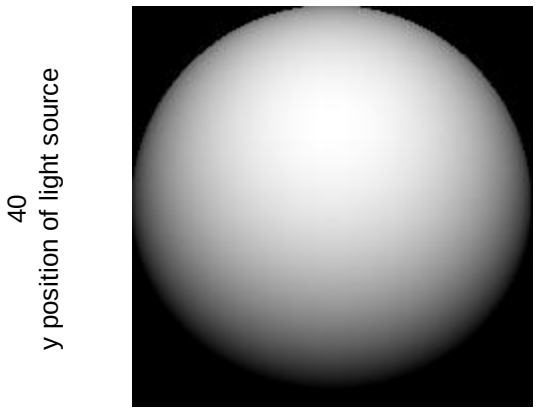
iv. Light Source x = 40; y = 0



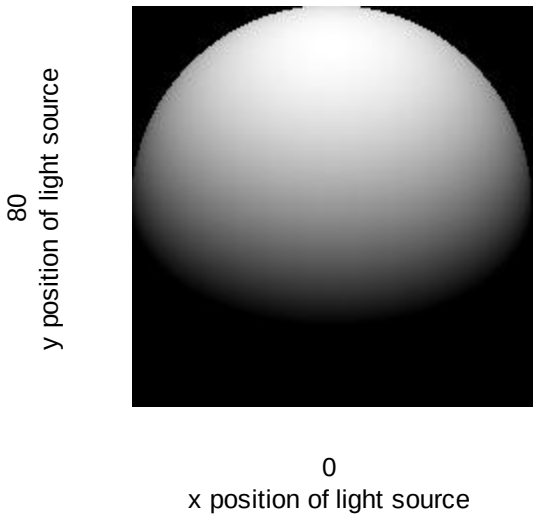
v. Light Source x = 80; y = 0



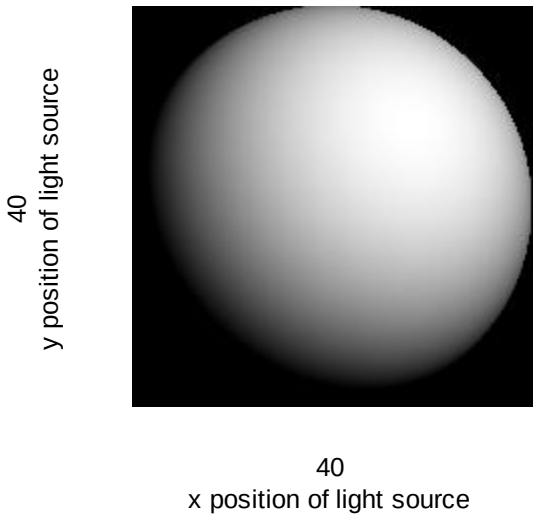
vi. Light Source x = 0; y = 40



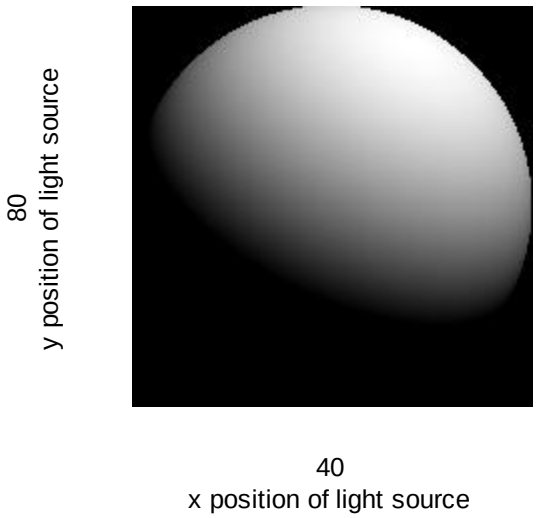
vii. Light Source x = 0; y = 80



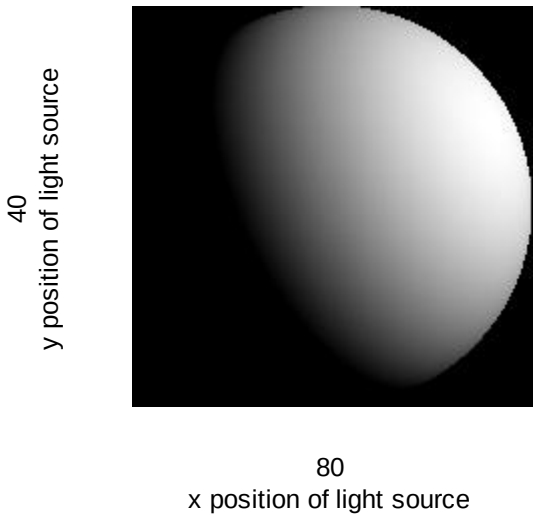
viii. Light Source x = 40; y = 40



ix. Light Source x = 40; y = 80



x. Light Source x = 80; y = 40



xi. Homework also contains 3 videos that

i. x location of light source is changing while y location of light source is constant.

x location is changing from "0" to "99", y location is constant and equals "0". This video is named as: "xs_is_changing_while_ys_is_constant.avi"

ii. y location of light source is changing while x location of light source is constant.

y location is changing from "0" to "99", x location is constant and equals "0". This video is named as: " ys_is_changing_while_xs_is_constant.avi"

iii. x location of light source and while y location of light source are changing together.

x and y location are changing from “0” to “60”. This video is named as: “xs_and_ys_are_changing_together.avi”