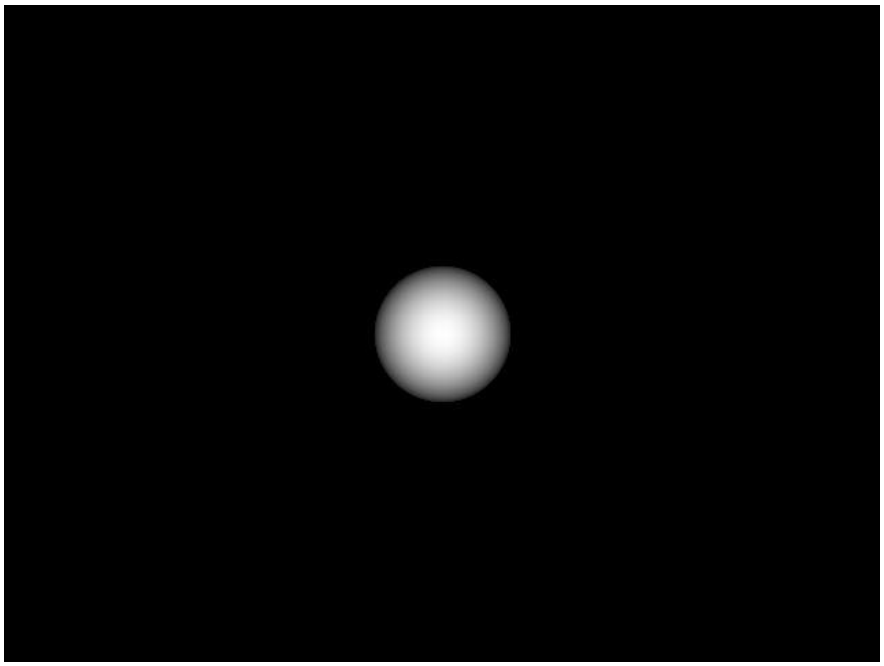
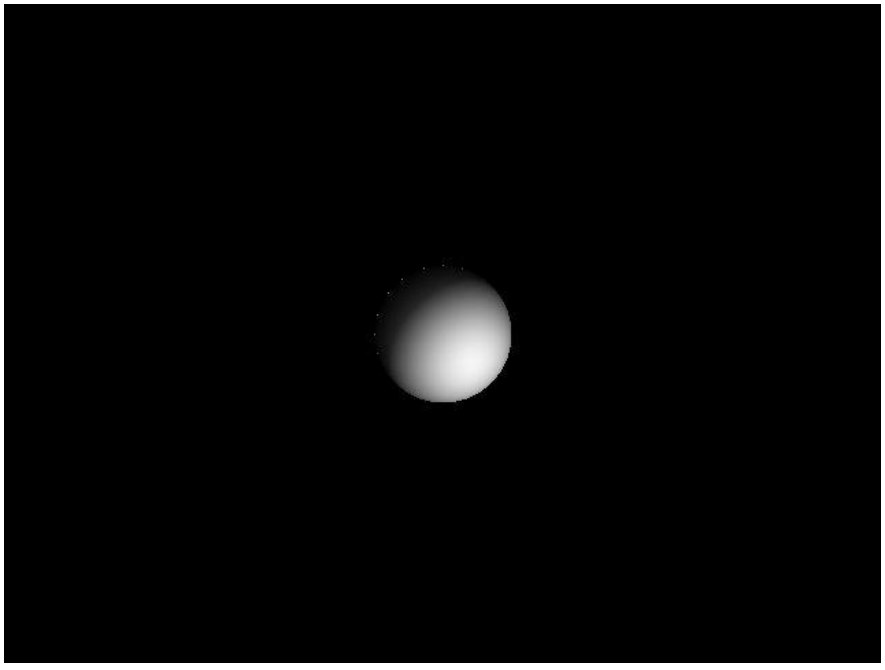
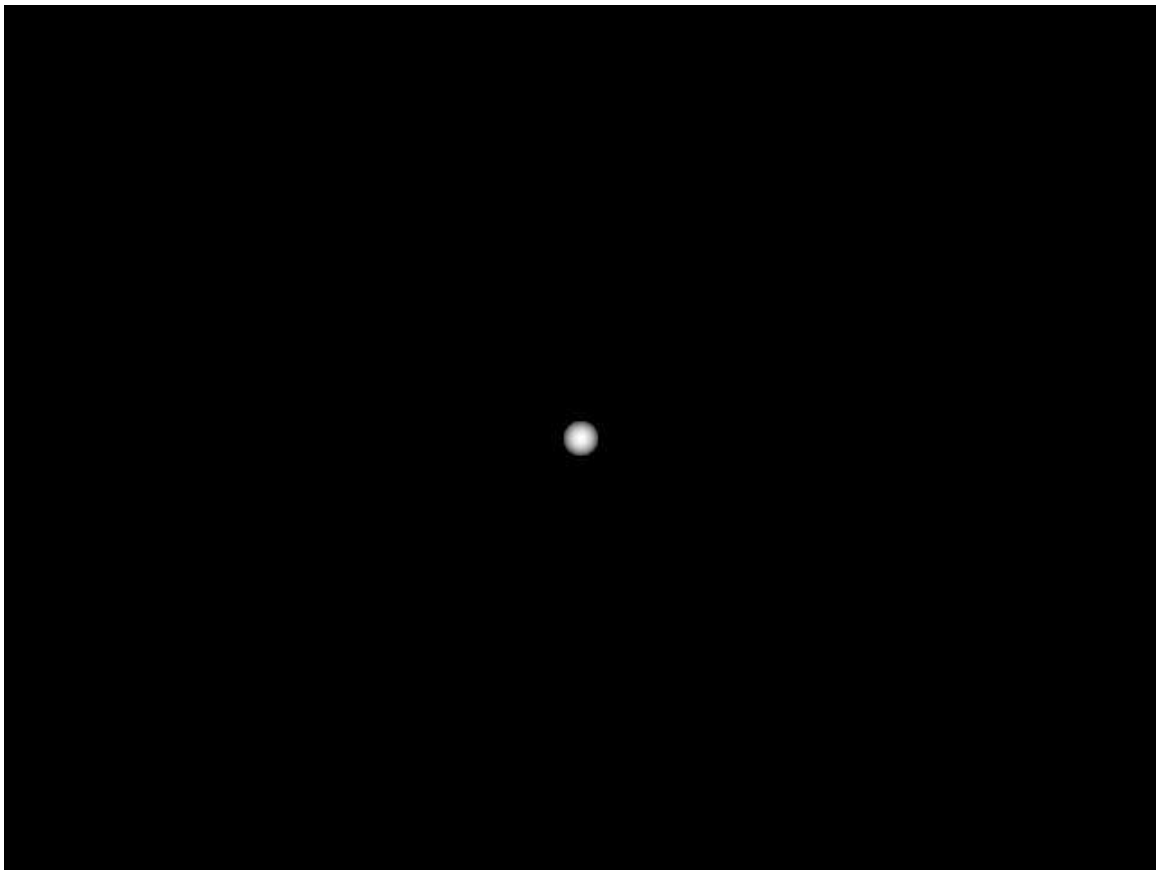
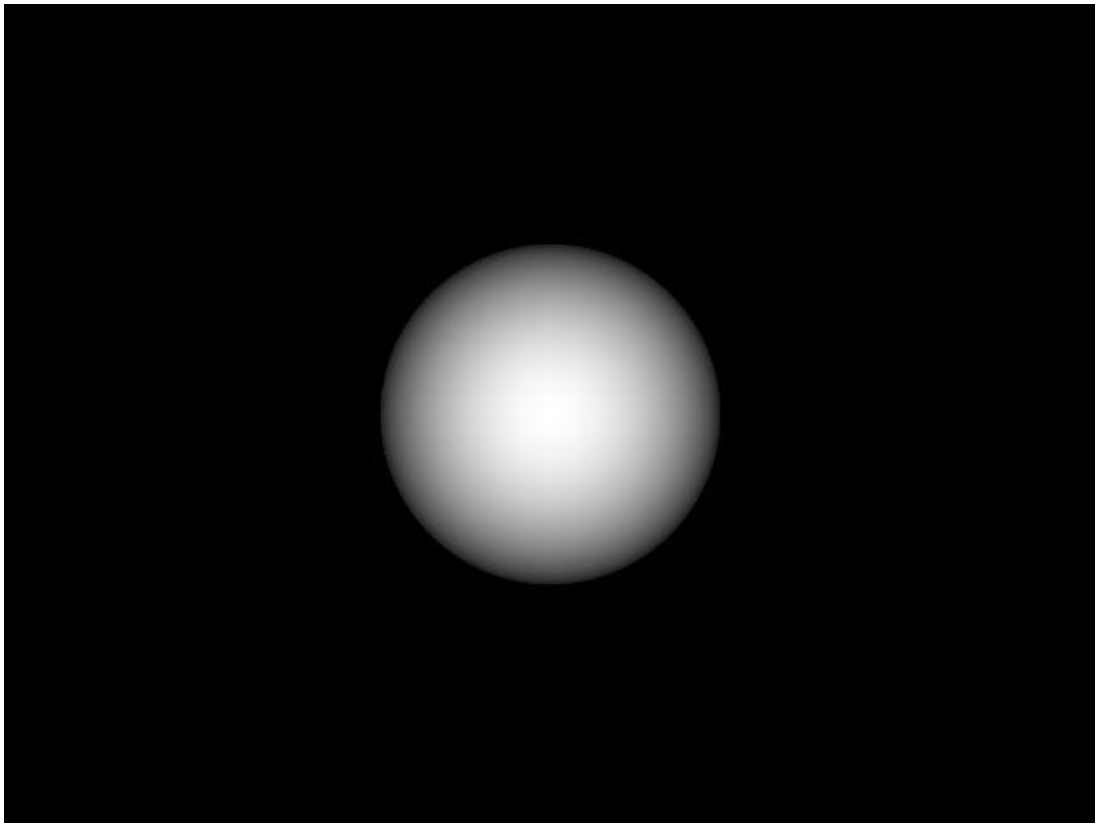
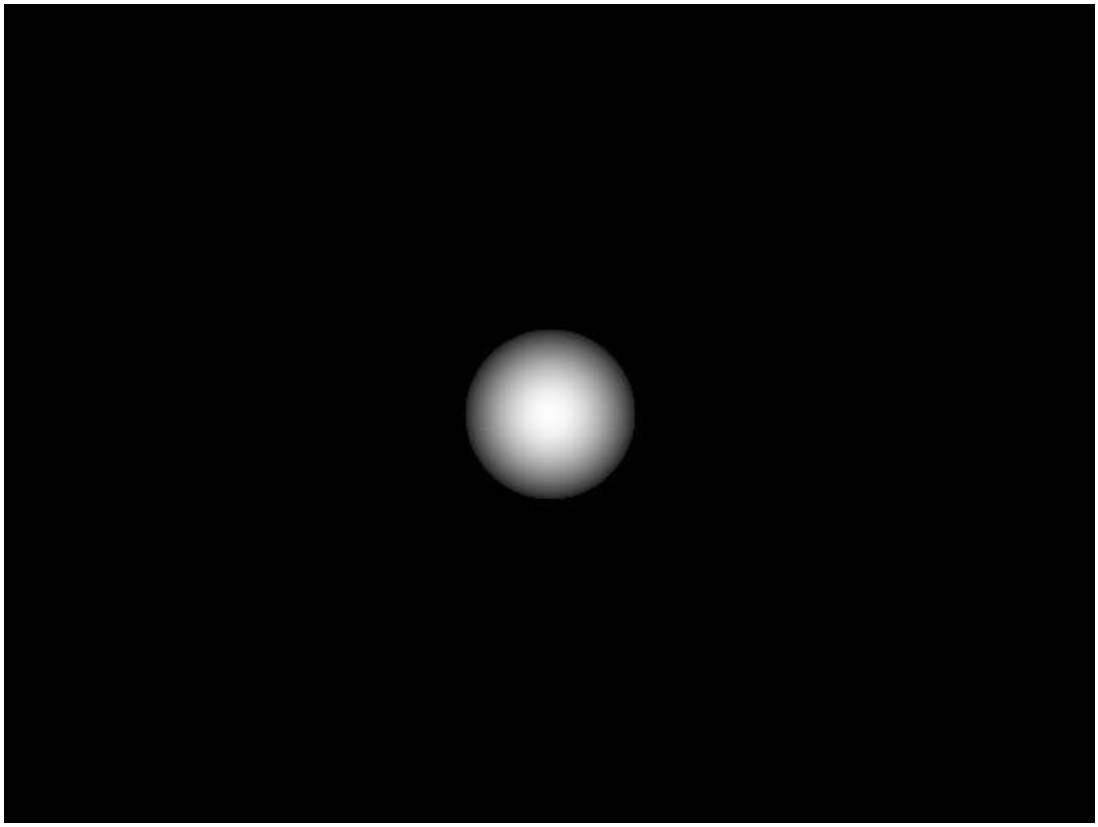
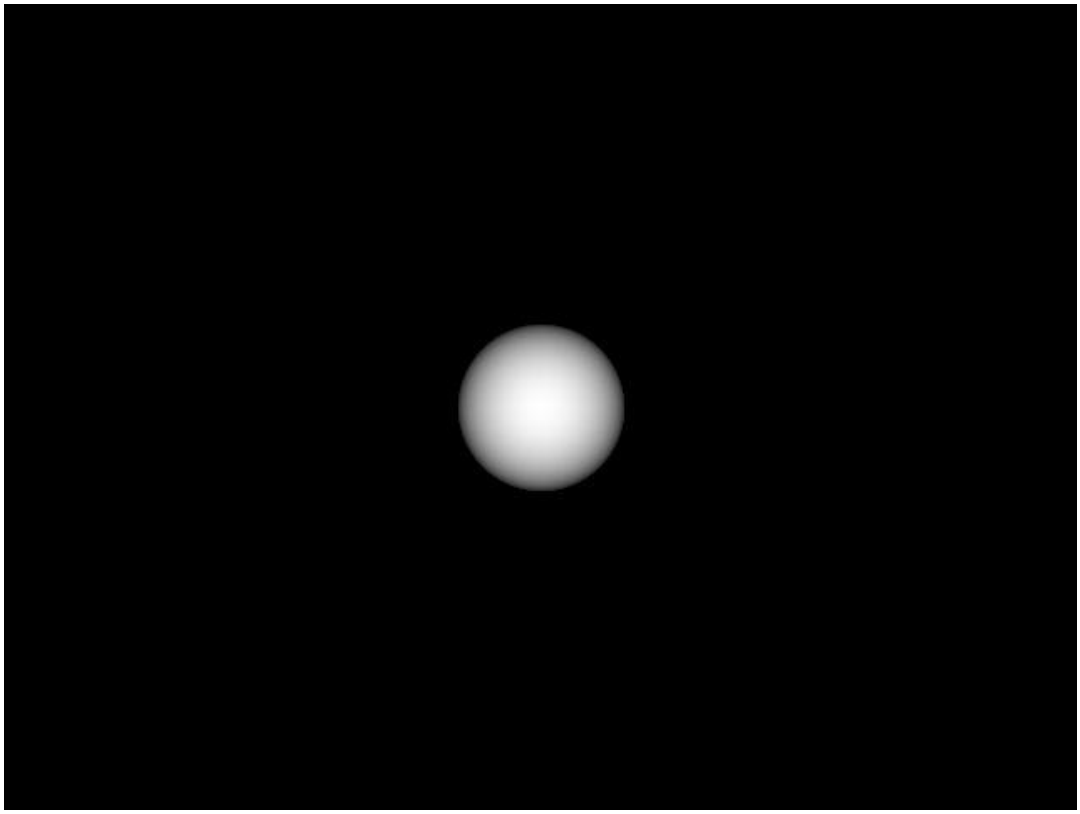
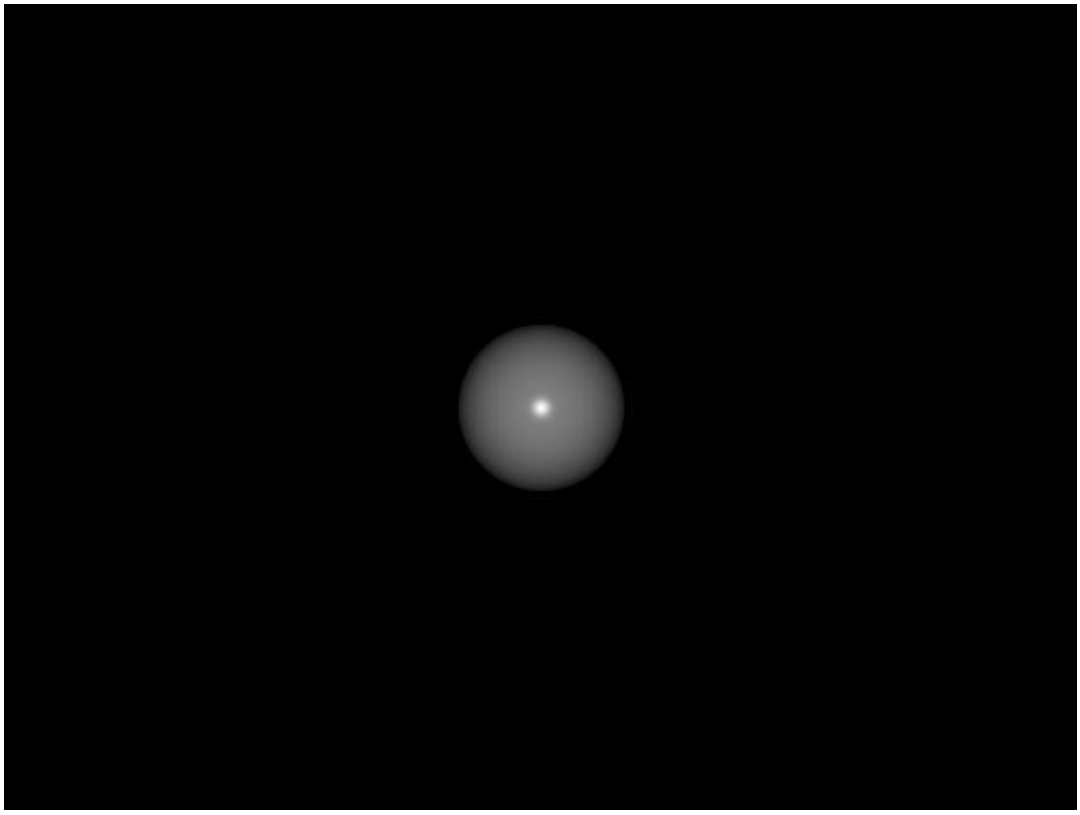
HW7

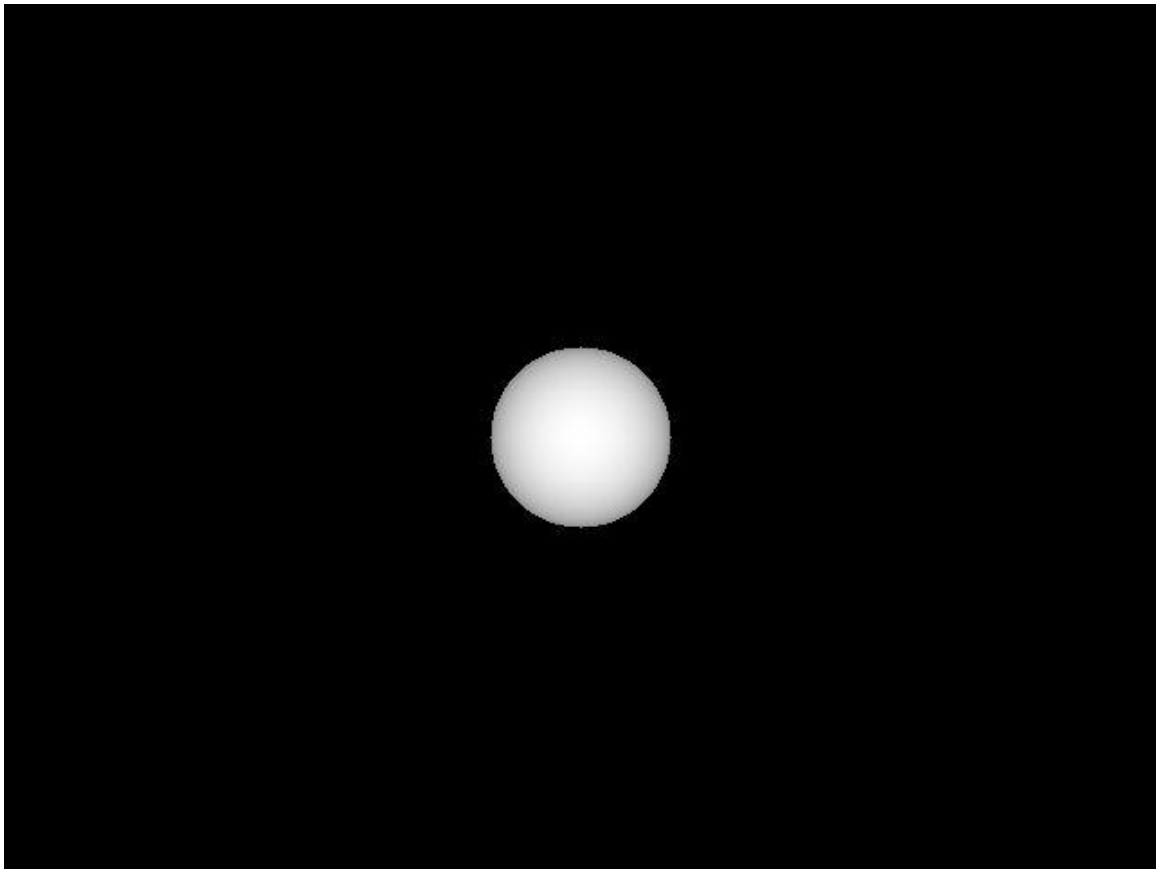
Images:

- Image A
- 
- Image B
- 

- Image C
- 
- Image D
- 

- Image E
- 
- Image F
- 

- Image G
- 
- Image H
- 

- Image I
- 

Finding normal:

P = $\frac{dz}{dx} = \frac{-x}{\sqrt{r^2-(x^2+y^2)}}$, q = $\frac{dz}{dy} = \frac{-y}{\sqrt{r^2-(x^2+y^2)}}$

N = $[\frac{-p,-q,1}{\sqrt{p^2+q^2+1)}}]$

Effects of each variable:

S is the location of the light source. If the light source is in the direction of the viewing position, in image a,d-i, then as a result the entire surface facing the viewer, is illuminated. In image b, c the light is shine from an angle then only part of the sphere is illuminated, on the same side with the light source.

R is the radius of the sphere, the larger the radius of the sphere, the larger the image, since the position of the sphere and the viewer is fixed. Therefore, when image = 10 in image D, results a smallest circle on the image plane.

M is the surface roughness. Larger m value will result a higher L value to 1 since large m value represents a matte surface as oppose to small m represent Lambertion surface. In the last image, a large m value made the whole image to have L close to 1 and almost evenly illuminated.

A is a weight of how much Lambertion or the Specular property of the image to dominate in an image. Factor A being large, will result the image to be more like an Lambertion surface while small A value will cause the surface to appear more Specular.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
//#include "StdAfx.h"

#define ROWS                            480
#define COLS                            640
#define LOGICAL_X_MIN                  -4.0
#define LOGICAL_X_MAX                   4.0
#define LOGICAL_Y_MIN                  -4.0
#define LOGICAL_Y_MAX                   4.0

void clear( unsigned char image[][COLS] );
int plot_logical_point( float x, float y, unsigned char image[][COLS] );
int plot_physical_point( int x, int y, unsigned char image[][COLS] );
int in_range( int x, int y );
void header( int row, int col, unsigned char head[32] );
void readFile (unsigned char image[][COLS],FILE *fp,char *ifile, char **argv);
void writeFile (unsigned char image[][COLS],FILE *fp, char *ofile, char **argv);
void dedy(int *a, int *b, unsigned char image[][COLS]);
void dedx(int *a, int *b, unsigned char image[][COLS]);
void sephereImage(float a,float m,double rA,unsigned char image[][COLS],float Si,float Sj,
float Sk);

int main( int argc, char **argv )
{
        int                             i, j;
        float                   a,m,Si,Sj,Sk;
        double                  x, y, rA,p,q,Ni,Nj,Nk,Ll,Ls,alpha,Hi,Hj,Hk,L,min,max;
        FILE                    *fp;
        unsigned char  image[ROWS][COLS];
        char *ifile, *ofileA, *ofileB,
*ofileC ,*ofileD,*ofileE,*ofileF,*ofileG,*ofileH,*ofileI;
        unsigned char head[32];

        fp = NULL;

        /* Example to show how to do format conversion */
        /* Input image file */
        //ifile = "image.raw";
        /* Output image file */
        ofileA = "imageA.ras";
        /* Output image file */
        ofileB = "imageB.ras";
        /* Output image file */
        ofileC = "imageC.ras";
        /* Output image file */
        ofileD = "imageD.ras";
        /* Output image file */
        ofileE = "imageE.ras";
        /* Output image file */
        ofileF = "imageF.ras";
```

```
        /* Output image file */
        ofileG = "imageG.ras";
        /* Output image file */
        ofileH = "imageH.ras";
        /* Output image file */
        ofileI = "imageI.ras";

        /* Clear image buffer */
        clear(image);

        /* Read in a raw image */
        //readFile (image,fp,ifile, argv);

        /* Create a header */
        header(ROWS, COLS, head);

        //Initialize target array
        clear(image);

       //-----------------------------------------------------Computing first case S= [0,0,1]
V = [0,0,1]
        //Initialize target array
        clear(image);
        rA = 50;
        a = 0.5;
        m =1;
        Si = 0;
        Sj = 0;
        Sk = 1;
        sephereImage(a,m,rA,image,Si,Sj,Sk);
        /* Save it into a ras image */
        writeFile (image,fp, ofileA, argv, head);

        //-----------------------------------------------------Computing 2nd case
        //Initialize target array
        clear(image);
        rA = 50;
        a = 0.5;
        m =1;
        Si = 1/sqrt(3.0);
        Sj = 1/sqrt(3.0);
        Sk = 1/sqrt(3.0);
        sephereImage(a,m,rA,image,Si,Sj,Sk);
        /* Save it into a ras image */
        writeFile (image,fp, ofileB, argv, head);

        //-----------------------------------------------------Computing 3rd case
        //Initialize target array
        clear(image);
        rA = 50;
        a = 0.5;
        m =1;
```

```
Si = 1;
Sj = 0;
Sk = 0;
sephereImage(a,m,rA,image,Si,Sj,Sk);
/* Save it into a ras image */
writeFile (image,fp, ofileC, argv, head);

//-------------------------------------------------Computing 4th case
//Initialize target array
clear(image);
rA = 10;
a = 0.5;
m =1;
Si = 0;
Sj = 0;
Sk = 1;
sephereImage(a,m,rA,image,Si,Sj,Sk);
/* Save it into a ras image */
writeFile (image,fp, ofileD, argv, head);

//-------------------------------------------------Computing 5th case
//Initialize target array
clear(image);
rA = 100;
a = 0.5;
m =1;
Si = 0;
Sj = 0;
Sk = 1;
sephereImage(a,m,rA,image,Si,Sj,Sk);
/* Save it into a ras image */
writeFile (image,fp, ofileE, argv, head);

//-------------------------------------------------Computing 6th case
//Initialize target array
clear(image);
rA = 50;
a = 0.1;
m =1;
Si = 0;
Sj = 0;
Sk = 1;
sephereImage(a,m,rA,image,Si,Sj,Sk);
/* Save it into a ras image */
writeFile (image,fp, ofileF, argv, head);

//-------------------------------------------------Computing 7th case
//Initialize target array
clear(image);
rA = 50;
a = 1;
m =1;
```

```
        Si = 0;
        Sj = 0;
        Sk = 1;
        sephereImage(a,m,rA,image,Si,Sj,Sk);
        /* Save it into a ras image */
        writeFile (image,fp, ofileG, argv, head);

        //-------------------------------------------------Computing 8th case
        //Initialize target array
        clear(image);
        rA = 50;
        a = 0.5;
        m =0.1;
        Si = 0;
        Sj = 0;
        Sk = 1;
        sephereImage(a,m,rA,image,Si,Sj,Sk);
        /* Save it into a ras image */
        writeFile (image,fp, ofileH, argv, head);

        //-------------------------------------------------Computing 9th case
        //Initialize target array
        clear(image);
        rA = 50;
        a = 0.5;
        m =10000;
        Si = 0;
        Sj = 0;
        Sk = 1;
        sephereImage(a,m,rA,image,Si,Sj,Sk);
        /* Save it into a ras image */
        writeFile (image,fp, ofileI, argv, head);

        return 0;
}

void sephereImage(float a,float m,double rA,unsigned char image[][COLS],float Si,float Sj,
float Sk)
{
        int                             i,j;
        //float                 Si,Sj,Sk;
        double                  x,  y,p,q,Ni,Nj,Nk,Ll,Ls,alpha,Hi,Hj,Hk,L;

        for(i=0; i<ROWS; i++)
        {
                for(j=0; j< COLS;j++)
                {
                        y = i-ROWS/2;
                        x = j-COLS/2;
                        if( (pow(x,2)+ pow(y,2))<= (pow(rA,2)))
                        {
                                if((pow(x,2)+ pow(y,2)) == (pow(rA,2)))
```

```
                                {
                                        Ni = -y/rA;
                                        Nj = -x/rA;
                                        Nk = 0;
                                }else
                                {
                                        p = -x/sqrt(pow(rA,2)-(pow(x,2)+ pow(y,2)));
                                        q = -y/sqrt(pow(rA,2)-(pow(x,2)+ pow(y,2)));
                                        Ni= -p/sqrt(pow(p,2)+pow(q,2)+1);
                                        Nj= -q/sqrt(pow(p,2)+pow(q,2)+1);
                                        Nk= 1/sqrt(pow(p,2)+pow(q,2)+1);
                                }

                                Ll =
(Si*Ni+Sj*Nj+Sk*Nk)/(sqrt(Si*Si+Sj*Sj+Sk*Sk)*sqrt(Ni*Ni+Nj*Nj+Nk*Nk));; //Ni*0+Nj*0+Nk*1
                                if(acos(Ll) >= (3.142/2))
                                        Ll=0;
                                Hi = (0+Si)/sqrt(pow(Si,2)+pow(Sj,2)+pow((Sk+1),2));
                                Hj = (0+Sj)/sqrt(pow(Si,2)+pow(Sj,2)+pow((Sk+1),2));
                                Hk = (Sk+1)/sqrt(pow(Si,2)+pow(Sj,2)+pow((Sk+1),2));
                                alpha =
acos((Ni*Hi+Nj*Hj+Nk*Hk)/sqrt(pow(Ni,2)+pow(Nj,2)+pow(Nk,2))/sqrt(pow(Hi,2)+pow(Hj,2)+pow(
Hk,2)));
                                Ls = exp(-pow((alpha/m),2));
                                L = a*Ll+(1-a)*Ls;
                                image[i][j] = 255*L;
                        }

                }
        }
        return;
}

void dedx(int *a, int *b, unsigned char image[][COLS])
{
        int i,temp,j;
        for ( i=1;i<ROWS-1;i++ )       //              de/dx
        {
                for (j=1;j<COLS-1;j++)
                {
                        //              de/dx
                        temp = abs((image[i-1][j+1]+2*image[i][j+1]+image[i+1][j+1])-
(image[i-1][j-1]+2*image[i][j-1]+image[i+1][j-1]));


                        if (temp > *a)
                        {
                                *a = temp;
                        }
                        if (temp < *b)
                        {
                                *b = temp;
```

```
                        }
                }
        }

        return;

}

void dedy(int *a, int *b, unsigned char image[][COLS])
{
        int i,temp,j;
        for ( i=1;i<ROWS-1;i++ )              //              de/dy
        {
                for (j=1;j<COLS-1;j++)
                {
                        //                      de/dy
                        temp=abs(image[i-1][j-1]+2*image[i-1][j]+image[i-1][j+1]-
(image[i+1][j-1] + 2*image[i+1][j]+image[i+1][j+1]));

                        if (temp > *a)
                        {
                                *a = temp;
                        }
                        if (temp < *b)
                        {
                                *b = temp;
                        }
                }
        }

        return;

}

void writeFile (unsigned char SGM[][COLS],FILE *fp, char *ofile, char **argv, unsigned
char head[32])
{
        int i;
        /* Save it into a ras image */
        /* Open the file */
        if (!( fp = fopen( ofile, "wb" )))
        fprintf( stderr, "error: could not open %s\n", argv[1] ), exit(1);

        /* Write the header */
        fwrite( head, 4, 8, fp );

        /* Write the image */
        for ( i = 0; i < ROWS; i++ )
                fwrite( SGM[i], 1, COLS, fp );

        /* Close the file */
        fclose( fp );
```

```
        return;
}
void readFile (unsigned char image[][COLS],FILE *fp,char *ifile, char **argv)
{
        int i = 0;
        /* Open the file */
        if (( fp = fopen( ifile, "rb" )) == NULL )
        {
                fprintf( stderr, "error: couldn't open %s\n", argv[1] );
                exit( 1 );
        }

        /* Read the file */
        for (i = 0; i < ROWS ; i++ )
                if ( fread( image[i], 1, COLS, fp ) != COLS )
                {
                        fprintf( stderr, "error: couldn't read enough stuff\n" );
                        exit( 1 );
                }

        /* Close the file */
        fclose( fp );
        return;
}
void clear( unsigned char image[][COLS] )
{
        int     i,j;
        for ( i = 0 ; i < ROWS ; i++ )
                for ( j = 0 ; j < COLS ; j++ ) image[i][j] = 0;
}

int plot_logical_point( float x, float y, unsigned char image[][COLS] )
{
        int     nx, ny;
        float   xc, yc;
        xc = COLS / ((float)LOGICAL_X_MAX - LOGICAL_X_MIN);
        yc = ROWS / ((float)LOGICAL_Y_MAX - LOGICAL_Y_MIN);
        nx = (x - LOGICAL_X_MIN) * xc;
        ny = (y - LOGICAL_Y_MIN) * yc;
        return plot_physical_point( nx, ny, image );
}

int plot_physical_point( int x, int y, unsigned char image[][COLS] )
{
        if (! in_range(x,y) ) return 0;
        return image[y][x] = 255;
}
int in_range( int x, int y )
        {
                return x >= 0 && x < COLS && y >= 0 && y < ROWS;
        }
```

```
        void header( int row, int col, unsigned char head[32] )
{
        int *p = (int *)head;
        char *ch;
        int num = row * col;

        /* Choose little-endian or big-endian header depending on the machine. Don't modify
this */
        /* Little-endian for PC */

        *p = 0x956aa659;
        *(p + 3) = 0x08000000;
        *(p + 5) = 0x01000000;
        *(p + 6) = 0x0;
        *(p + 7) = 0xf8000000;

        ch = (char*)&col;
        head[7] = *ch;
        ch ++;
        head[6] = *ch;
        ch ++;
        head[5] = *ch;
        ch ++;
        head[4] = *ch;

        ch = (char*)&row;
        head[11] = *ch;
        ch ++;
        head[10] = *ch;
        ch ++;
        head[9] = *ch;
        ch ++;
        head[8] = *ch;

        ch = (char*)&num;
        head[19] = *ch;
        ch ++;
        head[18] = *ch;
        ch ++;
        head[17] = *ch;
        ch ++;
        head[16] = *ch;

/*
        // Big-endian for unix
        *p = 0x59a66a95;
        *(p + 1) = col;
        *(p + 2) = row;
        *(p + 3) = 0x8;
        *(p + 4) = num;
        *(p + 5) = 0x1;
        *(p + 6) = 0x0;
```

```
        *(p + 7) = 0xf8;
*/
}
```