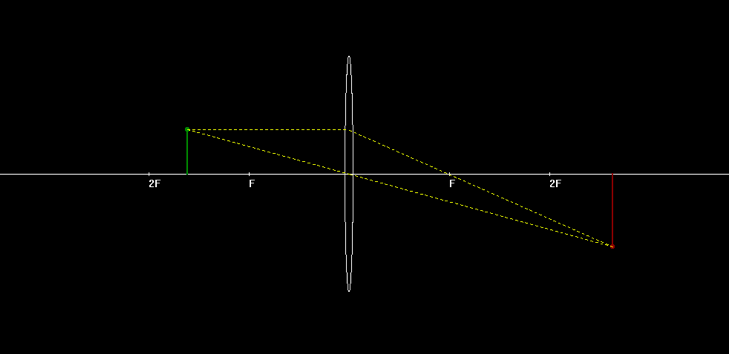
COMPUTER SCIENCE PROJECT  
2014-2015

SRI KUMARAN CHILDREN’S HOME C.B.S.E

SIMULATION OF IMAGE FORMATION  
IN LENS AND MIRROR



AKHIL U   
ANSUMAN PALO



# ACKNOWLEDGEMENTS

We would like to thank our Computer Science teachers, Mrs. Smitha and Mrs. Kavita, without whose invaluable knowledge and support, completing this project would have been an arduous task. We would also like to express our gratitude towards the school staff and principal for providing us the opportunity to access the laboratory.

Contents

[SYNOPSIS 4](#_Toc409439101)

[REASONS FOR PROJECT 5](#_Toc409439102)

[SYSTEM REQUIREMENTS 6](#_Toc409439103)

[CLASS DIAGRAM 6](#_Toc409439104)

[STANDARD HEADER FILES 7](#_Toc409439105)

[FLOWCHART 8](#_Toc409439106)

[SOURCE CODE 10](#_Toc409439107)

[SCREENSHOTS 11](#_Toc409439108)

[SCOPE FOR IMPROVEMENT 13](#_Toc409439109)

[BIBLOGRAPHY 14](#_Toc409439110)

# SYNOPSIS

The project is based on the reflection and refraction by different types of spherical mirrors and lenses respectively.

It uses a graphical representation of the image formation taking into account the necessary values like focal length, object distance and the object height to accurately calculate and plot the image.

The program uses an interactive interface allowing the user to:

* Feed values from a text window which opens a graphics window with required data
* Manipulate previous values in graphics window which gets refreshed
* Save the current data (in a data file).
* Open previously saved data.

# reasons for project

The project provides a graphical approach to the topic which is required for a better understanding. Many cases can be handled more accurately with the tool. Often visualisation of the cocept of image formation in ray optics can be quite an ordeal. This tool provides a medium to comprehend these concepts through pictorial representation.

# SYSTEM REQUIREMENTS

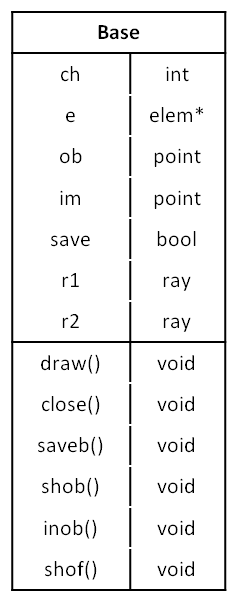
Requirements for program execution:

* + OS – Windows XP/ Appil or higher
  + RAM – 1GB or higher

Requirements for Compilation and debugging:

* + OS – Windows XP/ Appil or higher
  + RAM – 1GB or higher
  + C++ compiler with BGI graphics support (e.g. Bloodshed Dev C++, Visual Studio)

# CLASS DIAGRAM



Draw() – draws the principal axis, calls functions to draw lens/mirror, object image and the rays

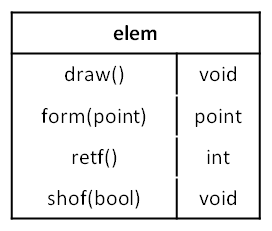
Close() – deletes dynamically allocated mirror/ lens and closes graphics window

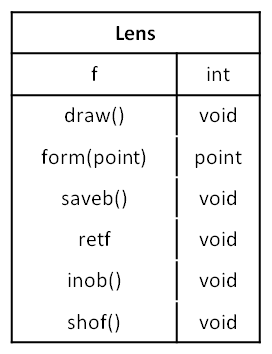
Saveb() – calls savebase() sending required parameters to save into text file

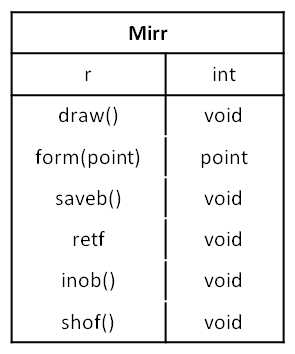
Shob() – shifts object by unit distance and redraws the layout

Inob() – increases object height by unit distance and redraws the layout

Shof() – shifts focal length by unit distance and redraws the layout





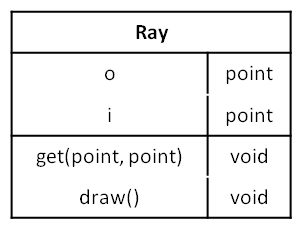


Draw() – draws mirror/lens

Form() – returns image coordinates using formula

Retf() – returns focal length for saving

Shof() – shifts focal length

Get() – initialises the ray with required coordinates

Draw() – Draws the rays

# STANDARD HEADER FILES

iostream

fstream

math.h

# FLOWCHART

# SOURCE CODE

Source.cpp

using namespace std;

#include<iostream>

#include<math.h>

#include "graphics.h"

#include<fstream>

#include "base.h"

void disp(base b);

extern void swindow(int);

extern void reset();

struct saveb{

int k;

float x, y;

bool operator==(saveb &t){if(k==t.k && x==t.x && y==t.y) return 1;}

};

void nwindow(){

int c;

cout << "1.Lens\n2.Mirror\n";

cin >> c;

base b(c - 1);

x = initwindow(1360, 650, "Draw");

b.draw();

disp(b);

closegraph(x);

}

void swindow(int k){

fstream fkey;

int num[2];

saveb b;

fkey.open("key.dat", ios::in|ios::binary);

fkey.read((char\*)num, sizeof(num));

fkey.close();

if(!num[k]){cout<<"\nNo Records Found\n"; system("pause");

return;}

ifstream f;

if(k)f.open("mirr.dat", ios::binary|ios::in);

else f.open("lens.dat", ios::in|ios::binary);

cout<<"Slno\t";

if(k)cout<<"Radius\t";

else cout<<"Focal l\t";

cout<<"Obj Dist\tObj Ht";

for(int i=0; i<num[k]; i++){

f.read((char\*)&b,sizeof(b));

cout<<endl<<i+1<<"\t"<<b.k<<"\t"<<b.x<<"\t\t"<<b.y;

}

cout<<"\nEnter Number:";

int x; cin>>x;

f.seekg((x-1)\*sizeof(b), ios::cur);

f.read((char\*)&b, sizeof(b));

base s(k, b.k, b.x, b.y);

x = initwindow(1360, 650, "Draw");

s.draw();

disp(s);

closegraph(x);

}

int main(){

int c;

do{

system("cls");

cout<<"1.Create new\n2.Open saved mirror\n3.Open saved lens\n4.Reset saved data\n0.Exit\nEnter Choice:";

cin>>c;

switch(c){

case 1:nwindow();break;

case 2:swindow(1);break;//MIRR

case 3:swindow(0);break;//LENS

case 4:reset();

}

}while(c);

}

void drawbox(bool c){

//Object dist

rectangle(65, 600, 90, 625);

outtextxy(75, 605, "-");

outtextxy(95, 605, "obj dis");

rectangle(140, 600, 165, 625);

outtextxy(150, 605, "+");

//Focal Length

rectangle(200, 600, 225, 625);

outtextxy(210, 605, "-");

outtextxy(230, 605, "focal l");

rectangle(275, 600, 300, 625);

outtextxy(285, 605, "+");

//Object height

rectangle(335, 600, 360, 625);

outtextxy(345, 605, "-");

outtextxy(365, 605, "obj h");

rectangle(410, 600, 435, 625);

outtextxy(420, 605, "+");

//Save

if(c)outtextxy(1096, 602, "D");

rectangle(1060,600,1109,625);

outtextxy(1063, 602, "SAVE");

//EXIT

rectangle(1130, 600, 1175, 625);

outtextxy(1135, 602, "EXIT");

}

void disp(base b){

int mx, my;

drawbox(b.save);

while (!kbhit()){

if (ismouseclick(WM\_LBUTTONUP)){

getmouseclick(WM\_LBUTTONUP, mx, my);

if (my > 600 && my < 625){

if (mx < 90 && mx>65) b.shob(0);

else if (mx > 140 && mx < 165) b.shob(1);

else if (mx > 200 && mx < 225) b.shof(0);

else if (mx > 275 && mx < 300) b.shof(1);

else if (mx > 335 && mx < 360) b.inob(0);

else if (mx > 410 && mx < 435) b.inob(1);

else if (mx > 1130 && mx < 1175){ b.close(); return;}

else if (!b.save) if (mx > 1060 && mx < 1105)b.saveb();

drawbox(b.save);

}

}

}

}

Base.h

using namespace std;

#include<iostream>

#include<math.h>

#include "graphics.h"

int xelem = 680, ypa = 325, s=15;

struct point { float x, y; };

int x;

class base;

extern int savebase(int, int, float, float);

class elem{

public:

virtual void draw() = 0;

virtual point form(point) = 0;

friend class base;

virtual int retf()=0;

virtual void shof(bool)=0;

};

class mirr : public elem{

int r;

friend class base;

public:

mirr(int rad){

r = rad;

}

mirr(){}

point form(point o){

point i;

if(o.x==r/2){i.x=i.y=0; return i;}

i.x = (r\*o.x) / ((2 \* o.x) - r);

i.y = -1.0\*(o.y\*i.x) / o.x;

return i;

}

void draw(){

line(r\*15+xelem,ypa-2,r\*15+xelem,ypa+2);

outtextxy(r\*15+xelem,ypa+5,"R");

line((r/2)\*15+xelem,ypa-2,(r/2)\*15+xelem,ypa+2);

outtextxy((r/2)\*15+xelem,ypa+5,"F");

if(r<0)

ellipse(xelem+r\*13/fabs(r),ypa,-87,87,13, 160);

else

ellipse(xelem+r\*13/fabs(r),ypa,87,-87,13, 160);

}

void shof(bool ty){

if(ty) r++;

else r--;

if(!r)if(ty) r++;

else r--;

}

int retf(){return r;}

};

class lens : public elem{

int f;

public:

lens(int foc){

f = foc;

}

lens(){}

point form(point o){

point i;

if(o.x==f){i.x=i.y=0; return i;}

i.x = (f\*o.x) / (o.x + f);

i.y = (o.y\*i.x) / o.x;

return i;

}

void draw(){

line(f\*15+xelem,ypa-2,f\*15+xelem,ypa+2);

line(-1\*f\*15+xelem,ypa-2,-1\*f\*15+xelem,ypa+2);

outtextxy(xelem+f\*15,ypa+5,"F");

outtextxy(xelem-f\*15,ypa+5,"F");

line(2\*f\*15+xelem,ypa-2,2\*f\*15+xelem,ypa+2);

line(-2\*f\*15+xelem,ypa-2,-2\*f\*15+xelem,ypa+2);

outtextxy(xelem+2\*f\*15,ypa+5,"2F");

outtextxy(xelem-2\*f\*15,ypa+5,"2F");

if(f>0)ellipse(xelem, ypa, 0, 360, 8 - 0.2\*f, 160);

else{

ellipse(xelem - 16, ypa, -87, 87, 13, 160);

ellipse(xelem + 16, ypa, 87, -87, 13, 160);

line(xelem - 16, ypa - 160, xelem + 16, ypa - 160);

line(xelem - 16, ypa + 160, xelem + 16, ypa + 160);

}

}

void shof(bool ty){

if(ty) f++;

else f--;

if(!f)if(ty) f++;

else f--;

}

int retf(){return f;}

};

class ray{

point o, i;

public:

void get(point ob, point im){

o = ob;

i = im;

}

void draw(){

setcolor(YELLOW);

setlinestyle(DOTTED\_LINE,0,1);

line(15\*o.x+xelem,-15\*o.y+ypa,xelem,ypa);//primary

line(15\*i.x+xelem,-15\*i.y+ypa,xelem,ypa);//primary

line(15\*o.x+xelem,-15\*o.y+ypa,xelem-2,-15\*o.y+ypa);//secondary

line(15\*i.x+xelem,-15\*i.y+ypa,xelem-2,-15\*o.y+ypa);//secondary

setcolor(WHITE);

setlinestyle(SOLID\_LINE,0,1);

}

};

class base{

int ch;//for lens/mir;

elem \*e;

point ob, im;

ray r1, r2;

public:

bool save;

base (int c, int k, float x, float y){

ch=c;

if(c)e=new mirr(k);

else e=new lens(k);

ob.x=x;

ob.y=y;

save = TRUE;

}

base(int c){

save = FALSE;

if (!c){

int f;

cout << "Enter focal length:";

cin >> f;

e = new lens(f);

ch=0; //FOR LENS 0

}

else{

int r;

cout << "Enter radius(less than 25):";

cin >> r;

e = new mirr(r);

ch=1; //FOR MIRROR 1

}

cout << "Enter object distance(preferably negative value; less than 25):";

cin >> ob.x;

cout << "Enter object height(less than 9):";

cin >> ob.y;

}

void close(){

delete e;

}

~base(){close();}

void draw(){

im = e->form(ob);

r1.get(ob, im);

line(0,325,1360,325);

e->draw();

setcolor(GREEN);

setlinestyle(SOLID\_LINE,0,2);

circle(15 \* ob.x + xelem, -15 \* ob.y + ypa,3);

line(15 \* ob.x + xelem, -15 \* ob.y + ypa, 15 \* ob.x + xelem, ypa);

setcolor(RED);

circle(15 \* im.x + xelem, -15 \* im.y + ypa,3);

line(15 \* im.x + xelem, -15 \* im.y + ypa, 15 \* im.x + xelem, ypa);

setlinestyle(SOLID\_LINE,0,1);

r1.draw();

}

void shob(bool ty){

if(ty)ob.x += 0.5;

else ob.x -= 0.5;

if(!ob.x)if(ty)ob.x += 0.5;

else ob.x -= 0.5;

cleardevice();

draw();

save=FALSE;

}

void inob(bool ty){

if(ty){

if(ob.y<9.5)ob.y += 0.25;

else { outtextxy(359, 570, "MAX HEIGHT!"); return;}

}

else ob.y -= 0.25;

cleardevice();

draw();

save=FALSE;

}

void shof(bool ty){

e->shof(ty);

cleardevice();

draw();

save=FALSE;

}

void saveb(){if(savebase(ch, e->retf(), ob.x, ob.y)) save=TRUE; draw();}

};

Save.cpp

#include<iostream>

#include<fstream>

using namespace std;

struct saveb{

int k;

float x, y;

bool operator==(saveb &t){if(k==t.k && x==t.x && y==t.y) return 1; else return 0;}

};

int savebase(int ch, int k, float x, float y){

fstream fkey;

int num[2];

saveb b = {k, x, y};

fkey.open("key.dat", ios::in|ios::binary);

fkey.read((char\*)num, sizeof(num));

fkey.close();

cout<<num[0]<<" "<<num[1]<<endl;

ofstream f;

if(ch)f.open("mirr.dat", ios::binary|ios::app);

else f.open("lens.dat", ios::app|ios::binary);

if(f)cout<<"YA!\n";

f.write((char\*)&b, sizeof(b));

if(f.good())num[ch]++;

cout<<f.fail();

f.close();

cout<<endl<<num[0]<<" "<<num[1]<<endl;

fkey.open("key.dat", ios::binary|ios::out);

fkey.write((char\*)num, sizeof(num));

fkey.close();

}

void reset(){

int num[2];

fstream fkey;

fstream f;

f.open("mirr.dat", ios::binary|ios::out);

f.close();

f.open("lens.dat", ios::binary|ios::out);

f.close();

num[0]=num[1]=0;

fkey.open("key.dat", ios::binary|ios::out);

fkey.write((char\*)num, sizeof(num));

fkey.close();

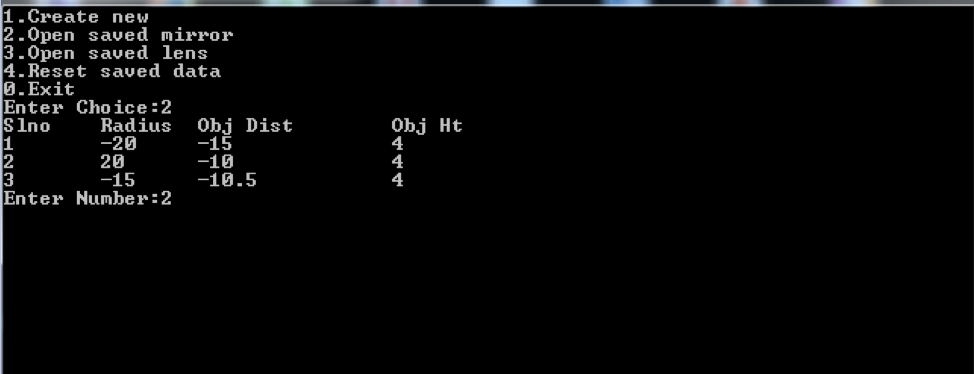
cout<<"\nReset successfully!\n";

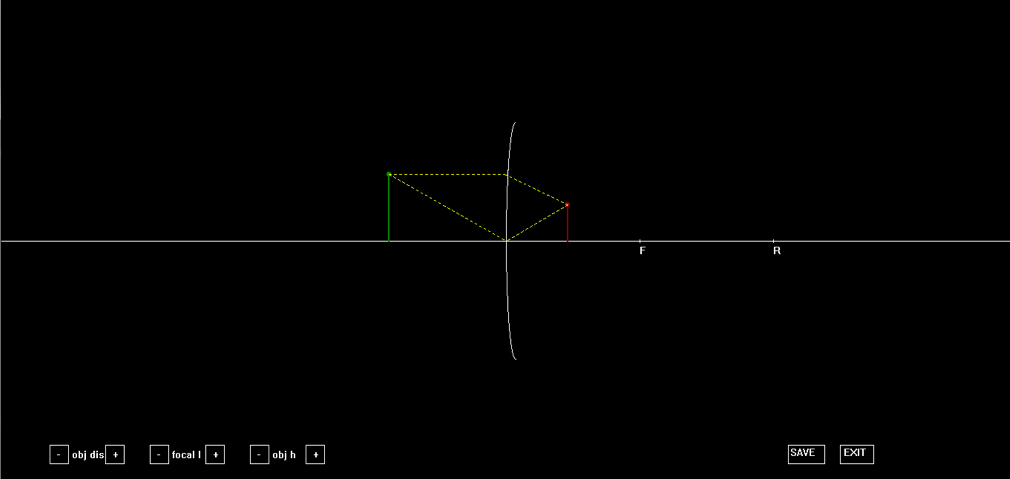
system("pause");

}

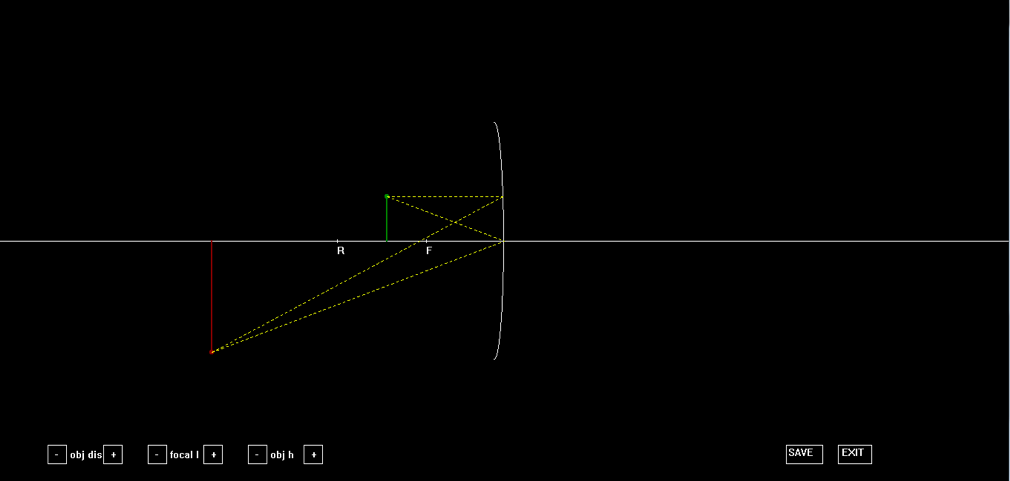
# SCREENSHOTS

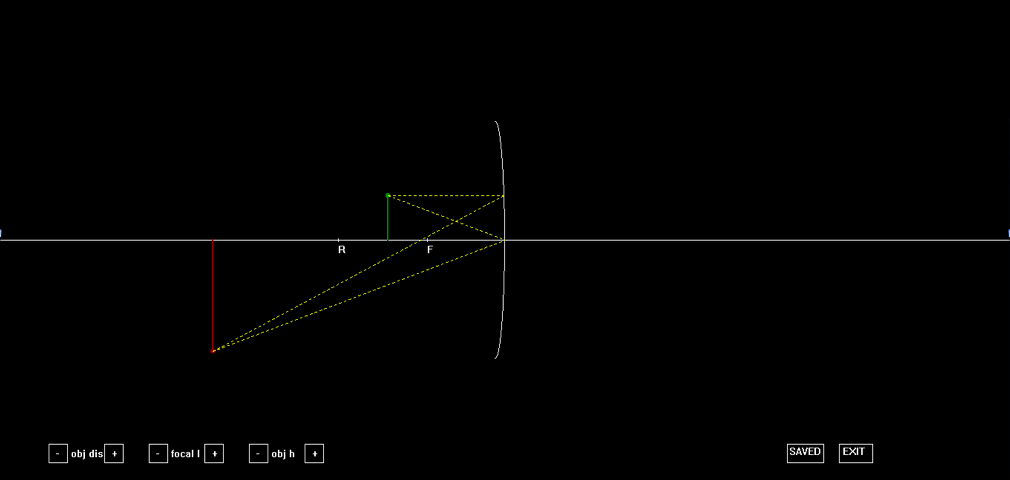
First menu:  
  


Opening a record from saved list:  
  




Saving a file :



click on the “SAVE” button  


Modifications can be made with the help of the buttons:



# SCOPE FOR IMPROVEMENT

The project is a basic version of the intended result and requires a lot of upgrades, patches and debugging to include more cases, add more features and improve its efficiency.

We find the following to be major areas which require attention:

Known Bugs and errors:

* Certain cases (e.g. image at infinity) cannot be displayed appropriately.
* Numeric details (distance, height, focal length) are not displayed in the graphics window correctly (feature removed).

Areas of Improvement:

* Upgrade to include comparison of two or more cases simultaneously.
* Improve the save function to remove duplicates and include a delete option. This can be done by upgrading to databases from flat files.
* Make the interface more user-friendly, e.g. by making the graphics more dynamic, adding more features like control of curvature, number of rays, etc.

# BIBLOGRAPHY

BOOKS:-

Computer Science for Class 12 by Sumita Arora

NCERT Class 12 Physics textbook (part 2)

WEBSITES:-

BGI Graphics – www.cs.colorado.edu