

**“OPTICAL VIRTUAL SIMULATOR”**

Report sumbitted in partial fulfillment of the requirment for the degree of

Bachelor of Technology

in

computer science & Engineering

By

**Prince raj (53), Jyotiprakash Nayak (26), Siddharth Phogat(56), Paras Sharma(44).**

To

Dr. Geetika dhand, Associate professor, CSE

Maharaja Surajmal institute of Technology

Affiliated to Guru Gobind singh Indraprastha University

Janakpuri, New Delhi-58.

2020-2024

**Introduction:**

Due to the Corona Virus the schools, colleges and other educational institutions have been shut down and are forced to take online classes. In some subjects teacher can tell and make a student understand the topic theoretically but in other subjects that require practical knowledge as well like Physics, the students no matter how hard they try, find it difficult to understand topics without actually seeing what is happening.

One of those topics is the Understanding about the working of mirrors, lens and prism about how when an object is placed at certain distance it can produce different types of images i.e., nature of image, magnification of image and distance of image.

In such cases an Online Simulator is really helpful to make students get an idea how Optical Devices function. A project in Computer Graphics was given to us. There were some restrictions like we cannot use premade code, images and drop function. Also, we were not allowed to use 3-D libraries, so we decided to use the good old graphics.h library and C++ language.

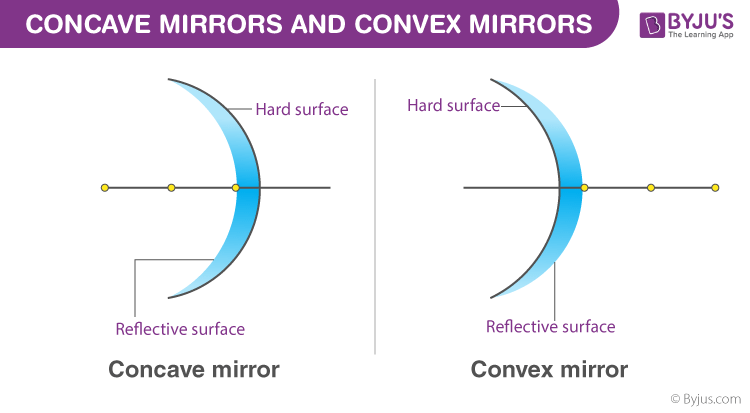
We started the project of an Online Optical Devices simulator that, when given some input, tells how the image of the object will be made using Ray Diagrams.

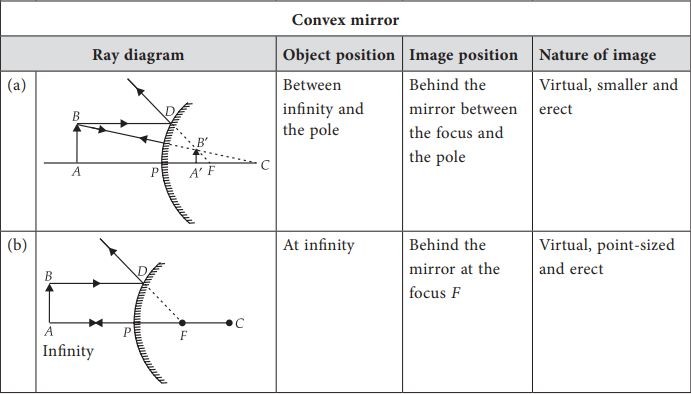
But before that I think we should once get a look on the different types of lens, mirrors and prism so that code is easily understood.

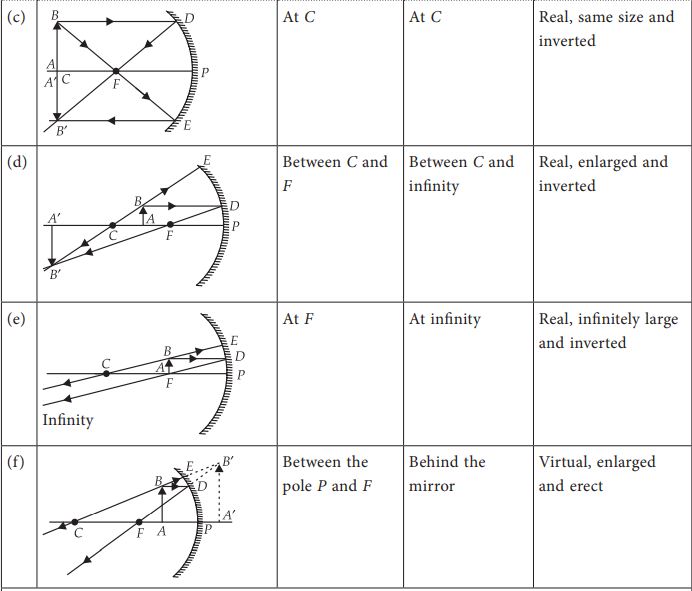
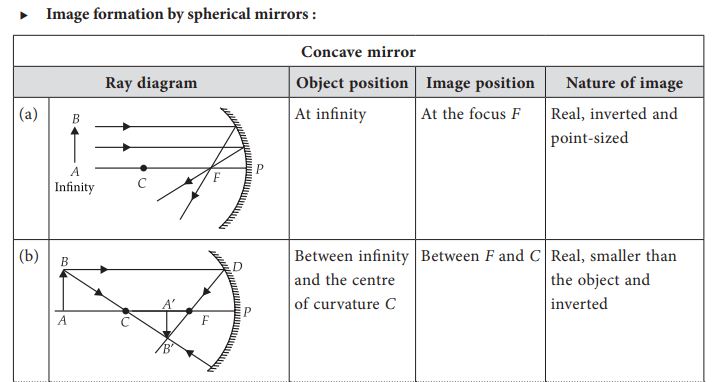
1.Mirrors:

A mirror is an object that reflects an image. Light that bounces off a mirror will show an image of whatever is in front of it, when focused through the lens of the eye or a camera. Mirrors reverse the direction of the image in an equal yet opposite angle from which the light shines upon it. Mirrors are of various types but the ones we used are : 1. Convex Mirrors and 2. Concave Mirrors.

Spherical mirrors in which inward surfaces are painted are known as **convex** mirrors, while the spherical mirrors in which outward surfaces are painted are known as **concave** mirrors.







2.Lens:

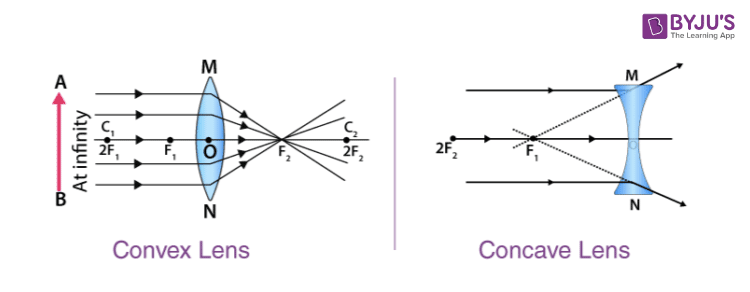
Lenses are basically magnifying glasses with curved sides. A lens is a piece of transparent glass which concentrates or disperses light rays when passes through them by refraction. Due to the magnifying property, lenses are used in telescopes and other magnifying devices. They are employed in cameras for gathering the light rays.

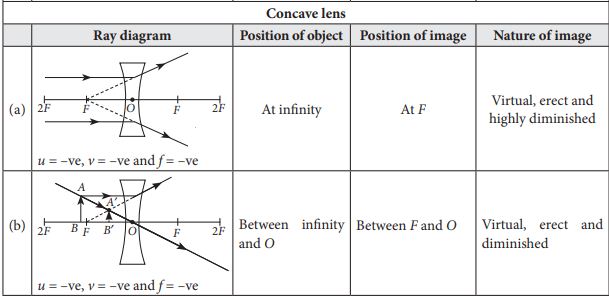
Lens are of many types but which we used in our project are:

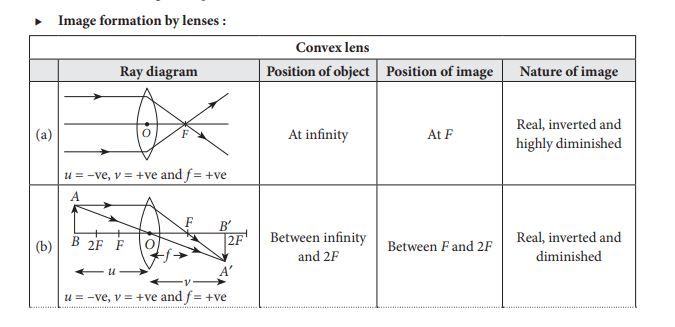
1. Convex lens and 2. Concave Lens

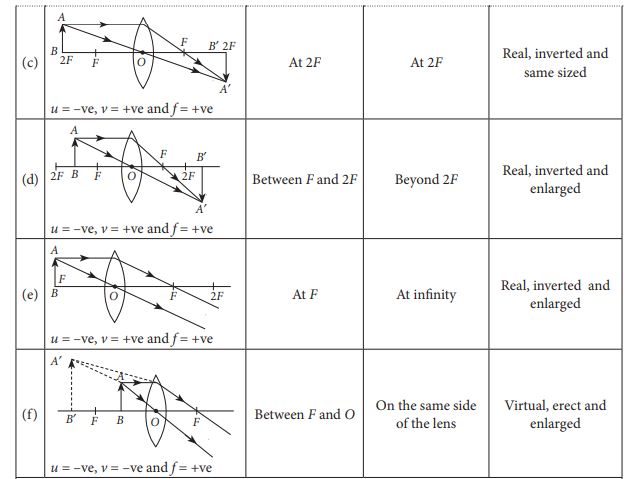
A convex lens is a lens with an outward curve. Unlike the concave lens, the thickness at the centre of a convex lens is more than the thickness at the edges of the lens. Convex lenses are converging lenses. They have the ability to converge a parallel beam of light into a point.

A **concave** lens is a type of lens with at least one side curved inwards. A concave lens with both sides curved inward is known as a biconcave lens. Concave lenses are diverging lenses, that is, they spread out light rays that have been refracted through it. They have the ability to diverge a parallel beam of light.









3.Prism:

Prism is a homogeneous solid transparent and refracting medium bounded by two plane surfaces inclined at an angle. The commonly used prism has two triangular faces that are parallel to each other and three rectangular surfaces. They are made from glass or other transparent material cut with precise angles. Two of the three rectangular faces are polished and are called refracting surfaces. The angle between these two rectangular faces is called the refractive angle of the prism.



**Making Of Simulator And Contribution Of Members**

In the 2D optical simulator, we had a team of four each contributing 4 distinct parts of the entire project. The members of team are Paras Sharma, Prince Raj, Siddharth Phogat and Jyotiprakash Nayak. As we are a team of four we decided to divide our work in four parts as mirror, convex lens, concave lens and prism.

**Mirror:** Mirror part was done by Paras Sharma. In mirror Paras made all the conditions on movement of light rays through mirror using predefined variables and functions. He used arc function to draw the mirrors, convex and concave respectively. Also he used line function to make a line that goes through the center of the mirror. Then he combined mirror formula and Bresenham’s algorithm to draw the rays of light. He also putpixel function to put colour to the rays. Again Paras made a Circles to define the source of the rays i.e object position and the image position after the reflection. Also he used outtextxy function to write about the positions of the rays and what type of image it is creating.

**Convex Lens**: Convex lens(Biconvex lens) part was made by Prince Raj. Again prince also used predefined functions to write his part of the code. He made the biconvex lens using arc function and the rays using the lens formula and Bresenham’s line drawing algorithm. Again as Paras has written his code Prince also used putpixel and outtextxy to define the object and image position as well as the path of the ray.

**Concave lens**: Concave lens(Biconcave lens) part was made by Siddharth Phogat. Siddharth also used predefined function but in a different way. He drew the structures and shapes inside his loops and statement which made it more user friendly and easy to use. As biconcave is a little more difficult to use and it has some limitations, he has given the users some range. The range given by him was to have precise and clear result from the algorithm in the graphic window.

**Prism**: Prism part was made by Jyotiprakash Nayak. In prism he used line functions to make the triangles and base of the prism. The rays in deviation part was made with putpixel and Bresenham’s algorithm. Written part was made by outtext and settextstyle functions. In the diffraction part again line was made using Bresenham’s algorithm and putpixel. The diffracted/multicolour part was made with ‘for’ loop and setcolor function.

**Merging and Final Touches**: Our goal was to make the project interactive so we used scanf functions to get the inputs from the user. After making our parts in the project now we have to merge our code. We decided that we will be using Switch case method to merge our code and make our menu.

**CODE**

#include <graphics.h>

#include <stdio.h>

#include <conio.h>

#include <math.h>

#include <dos.h>

      ////   OPTICS VIRTUAL SIMULATOR   ////

int main()

{

  int gd = DETECT, gm;

  int x1, y1, x2, x5, x6, y6, y5, y2, x3, y3, x4, y4, dx, dy, k, p, x, y, i, h, x7, y7, x8, y8;

  int q, f, v, u, hi, ho, m1, m2, choice1, choice2, choice, o = 360, num;

  char \*a = "C:\\MinGW\\mingw32\\lib";

  initgraph(&gd, &gm, a);

  outtextxy(250,20,"OPTICAL SIMULARTOR");

  printf("1. Mirror\n");

  printf("2. Convex Lens\n");

  printf("3. Concave Lens\n");

  printf("4. Prism\n");

  printf("5. Exit\n");

  printf("Choice:\n");

  scanf("%d", &choice);

  switch (choice)

  {

  case 1:

    printf("10. Concave Mirror\n");

    printf("20. Convex Mirror\n");

    printf("30. Exit\n");

    printf("Choice:\n");

    scanf("%d", &choice1);

    switch (choice1)

    {

    case 10:

      printf("enter the focal length of mirror :\n");

      scanf("%d", &h);

      printf("enter the distance of object from pole : \n");

      scanf("%d", &u);

      printf("enter the height of object(max. 45 units) : \n");

      scanf("%d", &ho);

      line(0, 250, 700, 250);

      arc(224, 250, 310, 50, 136);

      arc(224, 250, 310, 50, 136);

      outtextxy(360, 252, "O");

      outtextxy(360 - h, 252, "F");

      outtextxy(360 - 2 \* h, 252, "C");

      setcolor(YELLOW);

      line(360 - u, 250, 360 - u, 250 - ho \* 8 / 9);

      circle(360 - u, 250 - ho, ho / 9);

      u = u + 1;

      v = (u \* h) / (u - h);

      hi = (ho \* v) / u + 250;

      x1 = 360 - u;

      y1 = 250 - ho;

      x2 = 352;

      y2 = 250 - ho;

      dx = x2 - x1;

      dy = y2 - y1;

      p = 2 \* dy - dx;

      for (k = 0; k < dx; k++)

      {

        putpixel(x1, y1, 2);

        x1 = x1 + 1;

        y1 = y1;

        p = p + 2 \* abs(dy);

        delay(10);

      }

      if (h < u)

      {

        x3 = 352;

        y3 = 250 - ho;

        x4 = 360 - v;

        y4 = hi;

        dx = x3 - x4;

        dy = y3 - y4;

        p = 2 \* dy - dx;

        for (k = 0; k < dx; k++)

        {

          if (p < 0)

          {

            putpixel(x3, y3, 2);

            x3 = x3 - 1;

            y3 = y3;

            p = p + 2 \* abs(dy);

            delay(10);

          }

          else

          {

            putpixel(x3, y3, 2);

            x3 = x3 - 1;

            y3 = y3 + 1;

            p = p + 2 \* abs(dy) - 2 \* abs(dx);

            delay(10);

          }

        }

      }

      x6 = 360;

      y6 = 250;

      x5 = 360 - u;

      y5 = 250 - ho;

      dx = x6 - x5;

      dy = y6 - y5;

      p = 2 \* dy - dx;

      for (k = 0; k < dx; k++)

      {

        if (p < 0)

        {

          putpixel(x5, y5, 2);

          x5 = x5 + 1;

          y5 = y5;

          p = p + 2 \* abs(dy);

          delay(10);

        }

        else

        {

          putpixel(x5, y5, 2);

          x5 = x5 + 1;

          y5 = y5 + 1;

          p = p + 2 \* abs(dy) - 2 \* abs(dx);

          delay(10);

        }

      }

      x7 = 360;

      y7 = 250;

      x8 = 360 - v;

      y8 = hi;

      dx = x7 - x8;

      dy = y7 - y8;

      p = 2 \* dy - dx;

      for (k = 0; k < dx; k++)

      {

        if (p < 0)

        {

          putpixel(x7, y7, 2);

          x7 = x7 - 1;

          y7 = y7;

          p = p + 2 \* abs(dy);

          delay(10);

        }

        else

        {

          putpixel(x7, y7, 2);

          x7 = x7 - 1;

          y7 = y7 + 1;

          p = p + 2 \* abs(dy) - 2 \* abs(dx);

          delay(10);

        }

      }

      if (h > u)

      {

        x3 = 352;

        y3 = 250 - ho;

        x4 = 360 - v;

        y4 = hi;

        dx = x4 - x3;

        dy = y4 - y3;

        p = 2 \* dy - dx;

        for (k = 0; k < dx; k++)

        {

          if (p < 0)

          {

            putpixel(x3, y3, 2);

            x3 = x3 + 1;

            y3 = y3;

            p = p + 2 \* abs(dy);

            delay(10);

          }

          else

          {

            putpixel(x3, y3, 2);

            x3 = x3 + 1;

            y3 = y3 - 1;

            p = p + 2 \* abs(dy) - 2 \* abs(dx);

            delay(10);

          }

        }

        x7 = 360;

        y7 = 250;

        x8 = 360 - v;

        y8 = hi;

        dx = x8 - x7;

        dy = y8 - y7;

        p = 2 \* dy - dx;

        for (k = 0; k < dx; k++)

        {

          if (p < 0)

          {

            putpixel(x7, y7, 2);

            x7 = x7 + 1;

            y7 = y7;

            p = p + 2 \* abs(dy);

            delay(10);

          }

          else

          {

            putpixel(x7, y7, 2);

            x7 = x7 + 1;

            y7 = y7 - 1;

            p = p + 2 \* abs(dy) - 2 \* abs(dx);

            delay(10);

          }

        }

      }

      if (h > u)

      {

        setcolor(YELLOW);

        line(360 - v, 250, 360 - v, 250 - (250 - hi) \* 8 / 9);

        circle(360 - v, hi, (250 - hi) / 9);

        outtextxy(200, 400, "Virtual ,Erect and Enlarged Image");

      }

      if (h < u)

      {

        setcolor(YELLOW);

        line(359 - v, 250, 359 - v, (hi - 250) \* 8 / 9 + 250);

        circle(359 - v, hi, (hi - 250) / 9);

      }

      if (u > 2 \* h)

      {

        outtextxy(200, 400, "Real, Inverted and Diminished Image");

      }

      if (2 \* h > u && u > h)

      {

        outtextxy(200, 400, "Real, Inverted and Enlarged Image");

      }

      if (u == h)

      {

        outtextxy(200, 400, "Real, Inverted and Highly Enlarged Image");

      }

      if (u - 1 == 2 \* h)

      {

        outtextxy(200, 400, "Real, Inverted and Same Sized Image");

      }

      getch();

      closegraph();

      break;

    case 20:

      printf("enter the value of focal length of mirror(max. 300 units) :\n");

      scanf("%d", &h);

      printf("enter the distance of object from pole(max. 350 units) : \n");

      scanf("%d", &u);

      printf("enter the height of object(max. 45 units) : \n");

      scanf("%d", &ho);

      line(0, 250, 700, 250);

      arc(494, 250, 130, 230, 136);

      arc(495, 250, 130, 230, 136);

      outtextxy(365, 252, "O");

      outtextxy(360 + h, 252, "F");

      outtextxy(360 + 2 \* h, 252, "C");

      setcolor(YELLOW);

      line(360 - u, 250, 360 - u, 250 - ho \* 8 / 9);

      circle(360 - u, 250 - ho, ho / 9);

      u = u + 1;

      v = (u \* h) / (u + h);

      hi = 250 - (ho \* v) / u;

      x1 = 360 - u;

      y1 = 250 - ho;

      x2 = 364;

      y2 = 250 - ho;

      dx = x2 - x1;

      dy = y2 - y1;

      p = 2 \* dy - dx;

      for (k = 0; k < dx; k++)

      {

        putpixel(x1, y1, 2);

        x1 = x1 + 1;

        y1 = y1;

        p = p + 2 \* abs(dy);

        delay(10);

      }

      x7 = 364;

      y7 = 250 - ho;

      x8 = 250;

      y8 = (hi + ho - 250) \* 364 \* 250 / (724 + v) + 250 - ho - (hi + ho - 250) \* 364 / (724 + v);

      dx = x7 - x8;

      dy = y7 - y8;

      p = 2 \* dy - dx;

      for (k = 0; k < dx; k++)

      {

        if (p < 0)

        {

          putpixel(x7, y7, 2);

          x7 = x7 - 1;

          y7 = y7;

          p = p + 2 \* abs(dy);

          delay(10);

        }

        else

        {

          putpixel(x7, y7, 2);

          x7 = x7 - 1;

          y7 = y7 - 1;

          p = p + 2 \* abs(dy) - 2 \* abs(dx);

          delay(10);

        }

      }

      x6 = 360;

      y6 = 250;

      x5 = 360 - u;

      y5 = 250 - ho;

      dx = x6 - x5;

      dy = y6 - y5;

      p = 2 \* dy - dx;

      for (k = 0; k < dx; k++)

      {

        if (p < 0)

        {

          putpixel(x5, y5, 2);

          x5 = x5 + 1;

          y5 = y5;

          p = p + 2 \* abs(dy);

          delay(10);

        }

        else

        {

          putpixel(x5, y5, 2);

          x5 = x5 + 1;

          y5 = y5 + 1;

          p = p + 2 \* abs(dy) - 2 \* abs(dx);

          delay(10);

        }

      }

      x7 = 360;

      y7 = 250;

      x8 = 250;

      y8 = (250 - hi) \* 360 \* 250 / v + 250 - (250 - hi) \* 360 / v;

      dx = x7 - x8;

      dy = y7 - y8;

      p = 2 \* dy - dx;

      for (k = 0; k < dx; k++)

      {

        if (p < 0)

        {

          putpixel(x7, y7, 2);

          x7 = x7 - 1;

          y7 = y7;

          p = p + 2 \* abs(dy);

          delay(10);

        }

        else

        {

          putpixel(x7, y7, 2);

          x7 = x7 - 1;

          y7 = y7 + 1;

          p = p + 2 \* abs(dy) - 2 \* abs(dx);

          delay(10);

        }

      }

      x7 = 364;

      y7 = 250 - ho;

      x8 = 360 + v;

      y8 = hi;

      dx = x8 - x7;

      dy = y8 - y7;

      p = 2 \* dy - dx;

      for (k = 0; k < dx; k++)

      {

        if (p < 0)

        {

          putpixel(x7, y7, 2);

          x7 = x7 + 1;

          y7 = y7;

          p = p + 2 \* abs(dy);

          delay(10);

        }

        else

        {

          putpixel(x7, y7, 2);

          x7 = x7 + 1;

          y7 = y7 + 1;

          p = p + 2 \* abs(dy) - 2 \* abs(dx);

          delay(10);

        }

      }

      x7 = 360;

      y7 = 250;

      x8 = 360 + v;

      y8 = hi;

      dx = x8 - x7;

      dy = y8 - y7;

      p = 2 \* dy - dx;

      for (k = 0; k < dx; k++)

      {

        if (p < 0)

        {

          putpixel(x7, y7, 2);

          x7 = x7 + 1;

          y7 = y7;

          p = p + 2 \* abs(dy);

          delay(10);

        }

        else

        {

          putpixel(x7, y7, 2);

          x7 = x7 + 1;

          y7 = y7 - 1;

          p = p + 2 \* abs(dy) - 2 \* abs(dx);

          delay(10);

        }

      }

      setcolor(YELLOW);

      line(360 + v, 250, 360 + v, 250 - (250 - hi) \* 8 / 9);

      circle(360 + v, hi, (250 - hi) / 9);

      outtextxy(200, 400, "Virtual, Erect and Diminished Image");

      getch();

      closegraph();

      break;

    case 30:

      printf("\nBye\n");

      break;

    default:

      printf("\nInvalid Code\n");

    }

    break;

  case 2:

    printf("enter the value of focal length of the lens :");

    scanf("%d", &f);

    printf("enter the value of distance of object from the pole :");

    scanf("%d", &u);

    printf("enter the height of object and it would must be less than 45 cm : ");

    scanf("%d", &ho);

    u = u + 1;

    v = (u \* f) / (u - f); //lens formula

    hi = (ho \* v) / u;

    line(0, 250, 1000, 250);

    arc(300, 250, 310, 50, 90);

    arc(418, 250, 130, 230, 90);

    outtextxy(360 - f, 252, "F");

    outtextxy(360 - 2 \* f, 252, "2F");

    outtextxy(360, 252, "O");

    outtextxy(360 + f, 252, "F");

    outtextxy(360 + 2 \* f, 252, "2F");

    /\* \*\*\*\*\*\*\*\*\*

               convex lens all cases in form of graphic code written by prince raj

                                                 \*\*\*\*\*\*\*\*/

    if (u - 1 > f && u != f + 1)

    {

      setcolor(YELLOW);

      line(360 - u, 250, 360 - u, 250 - ho);

      circle(360 - u, 255 - ho, 5);

      x1 = 360 - u;

      y1 = 255 - ho;

      x2 = 360;

      y2 = 255 - ho;

      dx = x2 - x1;

      dy = y2 - y1;

      p = 2 \* dy - dx;

      for (k = 0; k < dx; k++)

      {

        if (p < 0)

        {

          putpixel(x1, y1, 2);

          x1 = x1 + 1;

          y1 = y1;

          p = p + 2 \* abs(dy);

          delay(10);

        }

        else

        {

          putpixel(x1, y1, 2);

          x1 = x1 + 1;

          y1 = y1 + 1;

          p = p + 2 \* abs(dy) - 2 \* abs(dx);

          delay(10);

        }

      }

      x3 = 360;

      y3 = 255 - ho;

      x4 = 360 + v;

      y4 = 252 + hi;

      dx = x4 - x3;

      dy = y4 - y3;

      p = 2 \* dy - dx;

      for (k = 0; k < dx; k++)

      {

        if (p < 0)

        {

          putpixel(x3, y3, 2);

          x3 = x3 + 1;

          y3 = y3;

          p = p + 2 \* abs(dy);

          delay(10);

        }

        else

        {

          putpixel(x3, y3, 2);

          x3 = x3 + 1;

          y3 = y3 + 1;

          p = p + 2 \* abs(dy) - 2 \* abs(dx);

          delay(10);

        }

      }

      // code for line passing through center of curverture

      x6 = 360 + v;

      y6 = 252 + hi;

      x5 = 360 - u;

      y5 = 255 - ho;

      dx = x6 - x5;

      dy = y6 - y5;

      p = 2 \* dy - dx;

      for (k = 0; k < dx; k++)

      {

        if (p < 0)

        {

          putpixel(x5, y5, 2);

          x5 = x5 + 1;

          y5 = y5;

          p = p + 2 \* abs(dy);

          delay(10);

        }

        else

        {

          putpixel(x5, y5, 2);

          x5 = x5 + 1;

          y5 = y5 + 1;

          p = p + 2 \* abs(dy) - 2 \* abs(dx);

          delay(10);

        }

      }

      line(360 + v, 247 + hi, 360 + v, 252);

      circle(360 + v, 252 + hi, 5);

      if (u < 2 \* f)

      {

        outtextxy(200, 400, "Real,Enlarge,inverted image");

      }

      else if (u > 2 \* f && u != 2 \* f + 1)

      {

        outtextxy(200, 400, "real,diminished,inverted image");

      }

      else if (2 \* f + 1 == u)

      {

        outtextxy(200, 400, "real,equal and inverted image");

      }

    }

    // code when the object at the focal legth

    else if (u == f + 1)

    {

      setcolor(YELLOW);

      line(360 - u, 250, 360 - u, 250 - ho);

      circle(360 - u, 255 - ho, 5);

      // line pass up to lens

      x1 = 360 - u;

      y1 = 255 - ho;

      x2 = 360;

      y2 = 255 - ho;

      dx = x2 - x1;

      dy = y2 - y1;

      p = 2 \* dy - dx;

      for (k = 0; k < dx; k++)

      {

        if (p < 0)

        {

          putpixel(x1, y1, 4);

          x1 = x1 + 1;

          y1 = y1;

          p = p + 2 \* abs(dy);

          delay(10);

        }

        else

        {

          putpixel(x1, y1, 4);

          x1 = x1 + 1;

          y1 = y1 + 1;

          p = p + 2 \* abs(dy) - 2 \* abs(dx);

          delay(10);

        }

      }

      // line from lens to focus

      x3 = 360;

      y3 = 255 - ho;

      x4 = 360 + f;

      y4 = 252;

      dx = x4 - x3;

      dy = y4 - y3;

      p = 2 \* dy - dx;

      for (k = 0; k < dx; k++)

      {

        if (p < 0)

        {

          putpixel(x3, y3, 4);

          x3 = x3 + 1;

          y3 = y3;

          p = p + 2 \* abs(dy);

          delay(10);

        }

        else

        {

          putpixel(x3, y3, 4);

          x3 = x3 + 1;

          y3 = y3 + 1;

          p = p + 2 \* abs(dy) - 2 \* abs(dx);

          delay(10);

        }

      }

      x6 = 580 - ho;

      y6 = (255 - ho) + ((ho - 3) \* (220 - ho)) / u;

      x5 = 360 + f;

      y5 = 252;

      dx = x6 - x5;

      dy = y6 - y5;

      p = 2 \* dy - dx;

      for (k = 0; k < dx; k++)

      {

        if (p < 0)

        {

          putpixel(x5, y5, 4);

          x5 = x5 + 1;

          y5 = y5;

          p = p + 2 \* abs(dy);

          delay(10);

        }

        else

        {

          putpixel(x5, y5, 4);

          x5 = x5 + 1;

          y5 = y5 + 1;

          p = p + 2 \* abs(dy) - 2 \* abs(dx);

          delay(10);

        }

      }

      x6 = 360;

      y6 = 252;

      x5 = 360 - u;

      y5 = 255 - ho;

      dx = x6 - x5;

      dy = y6 - y5;

      p = 2 \* dy - dx;

      for (k = 0; k < dx; k++)

      {

        if (p < 0)

        {

          putpixel(x5, y5, 4);

          x5 = x5 + 1;

          y5 = y5;

          p = p + 2 \* abs(dy);

          delay(10);

        }

        else

        {

          putpixel(x5, y5, 4);

          x5 = x5 + 1;

          y5 = y5 + 1;

          p = p + 2 \* abs(dy) - 2 \* abs(dx);

          delay(10);

        }

      }

      x5 = 360;

      y5 = 252;

      x6 = 575 - ho;

      y6 = 252 + ((ho - 3) \* (215 - ho)) / u;

      dx = x6 - x5;

      dy = y6 - y5;

      p = 2 \* dy - dx;

      for (k = 0; k < dx; k++)

      {

        if (p < 0)

        {

          putpixel(x5, y5, 4);

          x5 = x5 + 1;

          y5 = y5;

          p = p + 2 \* abs(dy);

          delay(10);

        }

        else

        {

          putpixel(x5, y5, 4);

          x5 = x5 + 1;

          y5 = y5 + 1;

          p = p + 2 \* abs(dy) - 2 \* abs(dx);

          delay(10);

        }

      }

      outtextxy(200, 400, "object at F image at infnity");

    }

    // for the virtual image in convex lens

    else if (u < f)

    {

      setcolor(YELLOW);

      line(360 - u, 250, 360 - u, 250 - ho);

      circle(360 - u, 255 - ho, 5);

      x1 = 360 - u;

      y1 = 255 - ho;

      x2 = 360;

      y2 = 255 - ho;

      dx = x2 - x1;

      dy = y2 - y1;

      p = 2 \* dy - dx;

      for (k = 0; k < dx; k++)

      {

        if (p < 0)

        {

          putpixel(x1, y1, 2);

          x1 = x1 + 1;

          y1 = y1;

          p = p + 2 \* abs(dy);

          delay(10);

        }

        else

        {

          putpixel(x1, y1, 2);

          x1 = x1 + 1;

          y1 = y1 + 1;

          p = p + 2 \* abs(dy) - 2 \* abs(dx);

          delay(10);

        }

      }

      x3 = 360;

      y3 = 255 - ho;

      x4 = 360 + f;

      y4 = 252;

      dx = x4 - x3;

      dy = y4 - y3;

      p = 2 \* dy - dx;

      for (k = 0; k < dx; k++)

      {

        if (p < 0)

        {

          putpixel(x3, y3, 2);

          x3 = x3 + 1;

          y3 = y3;

          p = p + 2 \* abs(dy);

          delay(10);

        }

        else

        {

          putpixel(x3, y3, 2);

          x3 = x3 + 1;

          y3 = y3 + 1;

          p = p + 2 \* abs(dy) - 2 \* abs(dx);

          delay(10);

        }

      }

      x6 = 580 - ho;

      y6 = (255 - ho) + ((ho - 3) \* (220 - ho)) / f;

      x5 = 360 + f;

      y5 = 252;

      dx = x6 - x5;

      dy = y6 - y5;

      p = 2 \* dy - dx;

      for (k = 0; k < dx; k++)

      {

        if (p < 0)

        {

          putpixel(x5, y5, 2);

          x5 = x5 + 1;

          y5 = y5;

          p = p + 2 \* abs(dy);

          delay(10);

        }

        else

        {

          putpixel(x5, y5, 2);

          x5 = x5 + 1;

          y5 = y5 + 1;

          p = p + 2 \* abs(dy) - 2 \* abs(dx);

          delay(10);

        }

      }

      //  equation of line passing through cneter of curvecture

      x6 = 360;

      y6 = 252;

      x5 = 360 - u;

      y5 = 255 - ho;

      dx = x6 - x5;

      dy = y6 - y5;

      p = 2 \* dy - dx;

      for (k = 0; k < dx; k++)

      {

        if (p < 0)

        {

          putpixel(x5, y5, 2);

          x5 = x5 + 1;

          y5 = y5;

          p = p + 2 \* abs(dy);

          delay(10);

        }

        else

        {

          putpixel(x5, y5, 2);

          x5 = x5 + 1;

          y5 = y5 + 1;

          p = p + 2 \* abs(dy) - 2 \* abs(dx);

          delay(10);

        }

      }

      x5 = 360;

      y5 = 252;

      x6 = 575 - ho;

      y6 = 252 + ((ho - 3) \* (215 - ho)) / u;

      dx = x6 - x5;

      dy = y6 - y5;

      p = 2 \* dy - dx;

      for (k = 0; k < dx; k++)

      {

        if (p < 0)

        {

          putpixel(x5, y5, 2);

          x5 = x5 + 1;

          y5 = y5;

          p = p + 2 \* abs(dy);

          delay(10);

        }

        else

        {

          putpixel(x5, y5, 2);

          x5 = x5 + 1;

          y5 = y5 + 1;

          p = p + 2 \* abs(dy) - 2 \* abs(dx);

          delay(10);

        }

      }

      //inverted part of image

      x4 = 360;

      y4 = 255 - ho;

      x3 = 360 + v;

      y3 = 252 + hi;

      dx = x4 - x3;

      dy = y4 - y3;

      p = 2 \* dy - dx;

      for (k = 0; k < dx; k++)

      {

        if (p < 0)

        {

          putpixel(x3, y3, 4);

          x3 = x3 + 1;

          y3 = y3;

          p = p + 2 \* abs(dy);

          delay(10);

        }

        else

        {

          putpixel(x3, y3, 4);

          x3 = x3 + 1;

          y3 = y3 + 1;

          p = p + 2 \* abs(dy) - 2 \* abs(dx);

          delay(10);

        }

      }

      x4 = 360 - u;

      y4 = 255 - ho;

      x3 = 360 + v;

      y3 = 252 + hi;

      dx = x4 - x3;

      dy = y4 - y3;

      p = 2 \* dy - dx;

      for (k = 0; k < dx; k++)

      {

        if (p < 0)

        {

          putpixel(x3, y3, 4);

          x3 = x3 + 1;

          y3 = y3;

          p = p + 2 \* abs(dy);

          delay(10);

        }

        else

        {

          putpixel(x3, y3, 4);

          x3 = x3 + 1;

          y3 = y3 + 1;

          p = p + 2 \* abs(dy) - 2 \* abs(dx);

          delay(10);

        }

      }

      line(360 + v, 252 + hi + 10, 360 + v, 250);

      circle(360 + v, 252 + hi, 5);

      outtextxy(200, 400, "virtual image, inverted, enlarge");

    }

    getch();

    closegraph();

    break;

  case 3: // initwindow(1000, 1000);

      // // // // // // // // // // // // // // // // // // // // //  Siddharth Phogat

    printf("For Object between O to infinity press 1.\nAnd for Object at infinity press 2\n");

    scanf("%d", &num);

    line(0, 240, 1000, 240);

    arc(265, 240, 310, 50, 90);

    arc(465, 240, 130, 230, 90);

    line(320, 308, 408, 308);

    line(320, 170, 408, 170);

    if (num == 1)

    {

      outtextxy(200, 100, "CASE : WHEN OBJECT IS BETWEEN O AND INFINITY");

      printf("IMPORTANT note:\n1. Take the height of object less than 50.\n2. To make the object appear at infnity put u=400\n");

      printf("Enter the value of focal length :\n");

      scanf("%d", &f);

      printf("Enter the value of distance of object : \n");

      scanf("%d", &u);

      printf("Enter the height of object : \n");

      scanf("%d", &ho);

      u = u + 1;

      v = (u \* f) / (u - f);

      hi = (ho \* v) / u;

      outtextxy(360 - f, 242, "F");

      outtextxy(360 - 2 \* f, 242, "2F");

      outtextxy(360, 242, "O");

      outtextxy(360 + f, 242, "F");

      outtextxy(360 + 2 \* f, 242, "2F");

      if (u - 1 < o)

      {

        setcolor(YELLOW);

        line(360 - u, 240, 360 - u, 242 - ho);

        circle(360 - u, 242 - ho, 5);

        x1 = 360 - u;

        y1 = 242 - ho;

        x2 = 360;

        y2 = 242 - ho;

        dx = x2 - x1;

        dy = y2 - y1;

        p = 2 \* dy - dx;

        for (k = 0; k < dx; k++)

        {

          if (p < 0)

          {

            putpixel(x1, y1, 4);

            x1 = x1 + 1;

            y1 = y1;

            p = p + 2 \* abs(dy);

            delay(10);

          }

        }

        x3 = 360;

        y3 = 242 - ho;

        x4 = 360 + v;

        y4 = 242 + hi;

        dx = x4 - x3;

        dy = y4 - y3;

        p = 2 \* dy - dx;

        for (k = 0; k < dx; k++)

        {

          if (p < 0)

          {

            putpixel(x3, y3, 4);

            x3 = x3 + 1;

            y3 = y3;

            p = p + 2 \* abs(dy);

            delay(10);

          }

          else

          {

            putpixel(x3, y3, 4);

            x3 = x3 + 1;

            y3 = y3 - 1;

            p = p + 2 \* abs(dy) - 2 \* abs(dx);

            delay(10);

          }

        }

        x6 = 360 + v;

        y6 = 242 + hi;

        x5 = 360 - u;

        y5 = 242 - ho;

        dx = x6 - x5;

        dy = y6 - y5;

        p = 2 \* dy - dx;

        for (k = 0; k < dx; k++)

        {

          if (p < 0)

          {

            putpixel(x5, y5, 4);

            x5 = x5 + 1;

            y5 = y5;

            p = p + 2 \* abs(dy);

            delay(10);

          }

          else

          {

            putpixel(x5, y5, 4);

            x5 = x5 + 1;

            y5 = y5 + 1;

            p = p + 2 \* abs(dy) - 2 \* abs(dx);

            delay(10);

          }

        }

        x3 = 360;

        y3 = 242 - ho;

        x4 = 360 + v;

        y4 = 242 + hi;

        dx = x4 - x3;

        dy = y4 - y3;

        p = 2 \* dy - dx;

        for (k = 0; k < dx; k++)

        {

          if (p < 0)

          {

            putpixel(x3, y3, 3);

            x3 = x3 - 1;

            y3 = y3;

            p = p + 2 \* abs(dy);

            delay(10);

          }

          else

          {

            putpixel(x3, y3, 3);

            x3 = x3 - 1;

            y3 = y3 + 1;

            p = p + 2 \* abs(dy) - 2 \* abs(dx);

            delay(10);

          }

        }

        setcolor(WHITE);

        line((735 - u) / 2, (488 - ho) / 2, (735 - u) / 2, 240);

        circle((735 - u) / 2, (484 - ho) / 2, 5);

        outtextxy(250, 285, "DIMINISHED, ERECT, VIRTUAL");

      }

      getch();

      closegraph();

    }

    else if (num == 2)

    {

      outtextxy(200, 100, "CASE : WHEN OBJECT IS AT INFINITY");

      printf("Enter the value of focal length :\n");

      scanf("%d", &f);

      printf("Enter the value of distance of object : \n");

      scanf("%d", &u);

      u = u + 1;

      v = (u \* f) / (u - f);

      hi = (ho \* v) / u;

      outtextxy(360 - f, 242, "F");

      outtextxy(360 - 2 \* f, 242, "2F");

      outtextxy(360, 242, "O");

      outtextxy(360 + f, 242, "F");

      outtextxy(360 + 2 \* f, 242, "2F");

      {

        //Above Ray

        x1 = -10;

        y1 = 205;

        x2 = 360;

        y2 = 205;

        dx = x2 - x1;

        dy = y2 - y1;

        p = 2 \* dy - dx;

        for (k = 0; k < dx; k++)

        {

          if (p < 0)

          {

            putpixel(x1, y1, 4);

            x1 = x1 + 1;

            y1 = y1;

            p = p + 2 \* abs(dy);

            delay(10);

          }

        }

        //Below Ray

        x3 = -10;

        y3 = 275;

        x4 = 360;

        y4 = 275;

        dx = x4 - x3;

        dy = y4 - y3;

        p = 2 \* dy - dx;

        for (k = 0; k < dx; k++)

        {

          if (p < 0)

          {

            putpixel(x3, y3, 4);

            x3 = x3 + 1;

            y3 = y3;

            p = p + 2 \* abs(dy);

            delay(10);

          }

        }

        //Above ray Refracted

        x3 = 360;

        y3 = 205;

        x4 = 360 + v;

        y4 = 242 + hi;

        dx = x4 - x3;

        dy = y4 - y3;

        p = 2 \* dy - dx;

        for (k = 0; k < dx; k++)

        {

          {

            putpixel(x3, y3, 4);

            x3 = x3 + 2.5;

            y3 = y3 - 1;

            p = p + 2 \* abs(dy) - 2 \* abs(dx);

            delay(10);

          }

        }

        //Below ray Refracted

        x3 = 360;

        y3 = 275;

        x4 = 360 + v;

        y4 = 242 + hi;

        dx = x4 - x3;

        dy = y4 - y3;

        p = 2 \* dy - dx;

        for (k = 0; k < dx; k++)

        {

          {

            putpixel(x3, y3, 4);

            x3 = x3 + 2;

            y3 = y3 + 1;

            p = p + 2 \* abs(dy) - 2 \* abs(dx);

            delay(10);

          }

        }

        //Above Ray Dotted

        x3 = 360 - f;

        y3 = 240;

        x4 = 360;

        y4 = 205;

        dx = x4 - x3;

        dy = y4 - y3;

        p = 2 \* dy - dx;

        for (k = 0; k < dx; k++)

        {

          if (p < 0)

          {

            putpixel(x3, y3, 3);

            x3 = x3 + 1;

            y3 = y3;

            p = p + 2 \* abs(dy);

            delay(10);

          }

          else

          {

            putpixel(x3, y3, 3);

            x3 = x3 + 1;

            y3 = y3 - 1;

            p = p + 2 \* abs(dy) - 2 \* abs(dx);

            delay(10);

          }

        }

        //Below Ray Dotted

        x3 = 360 - f;

        y3 = 240;

        x4 = 360;

        y4 = 275;

        dx = x4 - x3;

        dy = y4 - y3;

        p = 2 \* dy - dx;

        for (k = 0; k < dx; k++)

        {

          if (p < 0)

          {

            putpixel(x3, y3, 3);

            x3 = x3 + 1;

            y3 = y3;

            p = p + 2 \* abs(dy);

            delay(10);

          }

          else

          {

            putpixel(x3, y3, 3);

            x3 = x3 + 1;

            y3 = y3 + 1;

            p = p + 2 \* abs(dy) - 2 \* abs(dx);

            delay(10);

          }

        }

        setcolor(WHITE);

        outtextxy(200, 285, "POINT SIZED, ERECT, VIRTUAL");

      }

      getch();

      closegraph();

    }

    break;

  case 4:

    printf("10. Deviation\n");

    printf("20. Diffraction\n");

    printf("30. Exit\n");

    printf("Choice:\n");

    scanf("%d", &choice2);

    switch (choice2)

    {

    case 10:

      printf("Enter the x co-ordinate:\n");

      scanf("%d", &x);

      printf("Enter the y co-ordinate:\n");

      scanf("%d", &y);

      line(0, 250, 700, 250);

      line(320, 150, 400, 250);

      line(400, 250, 250, 250);

      line(250, 250, 320, 150);

      setcolor(14);

      outtextxy(285, 85, "Deviation");

      outtextxy(x - 15, y - 15, "source");

      outtextxy(320, 140, "A");

      outtextxy(410, 255, "C");

      outtextxy(240, 255, "B");

      if (x == 250 && y == 250)

      {

        setcolor(GREEN);

        line(x, y, 325, 145);

      }

      else if (y == 200)

      {

        setcolor(GREEN);

        line(x, y, 285, 200);

        line(285, 200, 370, 215);

        line(370, 215, 405, 255);

      }

      else

      {

        x1 = x;

        y1 = y;

        x2 = 285;

        y2 = 198;

        dx = x2 - x1;

        dy = y2 - y1;

        p = 2 \* dy - dx;

        for (k = 0; k < abs(dx); k++)

        {

          if (p < 0)

          {

            putpixel(x1, y1, 2);

            x1 = x1 + 1;

            y1 = y1;

            p = p + 2 \* abs(dy);

            delay(40);

          }

          else if (y < 200)

          {

            putpixel(x1, y1, 2);

            x1 = x1 + 1;

            y1 = y1 + 1;

            p = p + 2 \* abs(dy) - 2 \* abs(dx);

            delay(40);

          }

          else

          {

            putpixel(x1, y1, 2);

            x1 = x1 + 1;

            y1 = y1 - 1;

            p = p + 2 \* abs(dy) - 2 \* abs(dx);

            delay(40);

          }

        }

        x6 = x2;

        y6 = y2;

        x5 = 360;

        y5 = 200;

        dx = x6 - x5;

        for (k = 0; k < 75; k++)

        {

          putpixel(x6, y6, 4);

          x6 = x6 + 1;

          y6 = y6;

          delay(10);

        }

        x3 = x5;

        y3 = y5;

        x4 = 450;

        y4 = 250;

        dx = x4 - x3;

        dy = y4 - y3;

        p = 2 \* dy - dx;

        for (k = 0; k < dx; k++)

        {

          if (p < 0)

          {

            putpixel(x3, y3, 12);

            x3 = x3 + 1;

            y3 = y3;

            p = p + 2 \* abs(dy);

            delay(40);

          }

          else

          {

            putpixel(x3, y3, 2);

            x3 = x3 + 1;

            y3 = y3 + 1;

            p = p + 2 \* abs(dy) - 2 \* abs(dx);

            delay(40);

          }

        }

      }

      getch();

      closegraph();

      break;

    case 20:

      printf("Enter the x co-ordinates:\n");

      scanf("%d", &x);

      printf("Enter the y co-ordinates:\n");

      scanf("%d", &y);

      line(0, 250, 700, 250);

      line(320, 150, 400, 250);

      line(400, 250, 250, 250);

      line(250, 250, 320, 150);

      line(450, 150, 450, 250);

      setcolor(14);

      outtextxy(280, 85, "Diffraction");

      outtextxy(320, 140, "A");

      outtextxy(410, 255, "C");

      outtextxy(240, 255, "B");

      x1 = x;

      y1 = y;

      x2 = 285;

      y2 = 200;

      outtextxy(x1, y1 - 10, "source");

      dx = x2 - x1;

      dy = y2 - y1;

      p = 2 \* dy - dx;

      for (k = 0; k < abs(dx); k++)

      {

        if (p < 0)

        {

          putpixel(x1, y1, 2);

          x1 = x1 + 1;

          y1 = y1;

          p = p + 2 \* abs(dy);

          delay(40);

        }

        else if (p < 0 && y1 > 200)

        {

          putpixel(x1, y1, 2);

          x1 = x1 + 1;

          y1 = y1 - 1;

          p = p + 2 \* abs(dy) - 2 \* abs(dx);

          delay(40);

        }

        else if (y1 < 200)

        {

          putpixel(x1, y1, 2);

          x1 = x1 + 1;

          y1 = y1 + 1;

          p = p + 2 \* abs(dy) - 2 \* abs(dx);

          delay(40);

        }

        else

        {

          putpixel(x1, y1, 2);

          x1 = x1 + 1;

          y1 = y1 - 1;

          p = p + 2 \* abs(dy) - 2 \* abs(dx);

          delay(40);

        }

      }

      for (i = 1; i < 50; i++)

      {

        delay(40);

        setcolor(i / 6);

        line(x1, y1, 450, i + 200);

      }

      getch();

      closegraph;

      break;

    case 30:

      printf("\nBye\n");

      break;

    default:

      printf("\nInvalid Code\n");

    }

    break;

  case 5:

    printf("\nBye\n");

    break;

  default:

    printf("\nInvalid Code\n");

  getch();

  closegraph();

  }

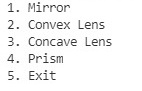
  return 0;

}

**Result & Discussion:**

The very first interface

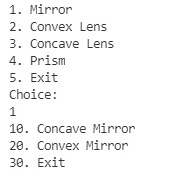
Select one from five shown options :



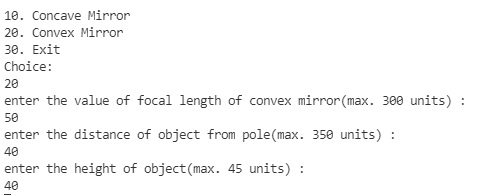
If we select choice 1.

**case 1: All mirror case.**

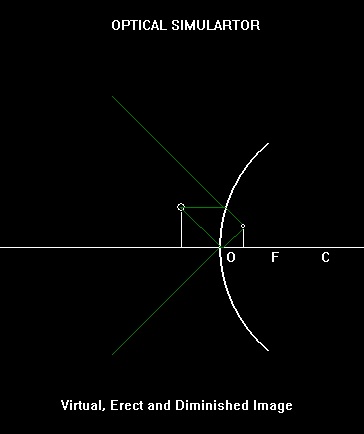
Mirror further have two more options :



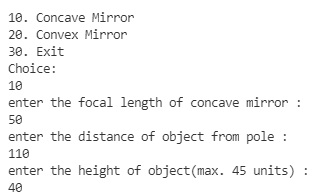
On selecting 20. Convex Mirror and entering required values :



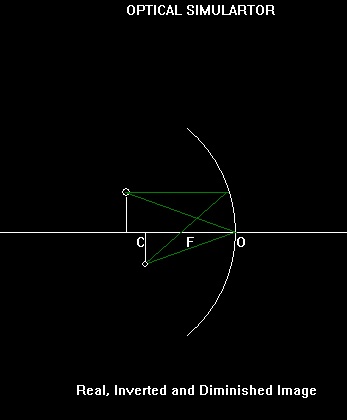
Result :



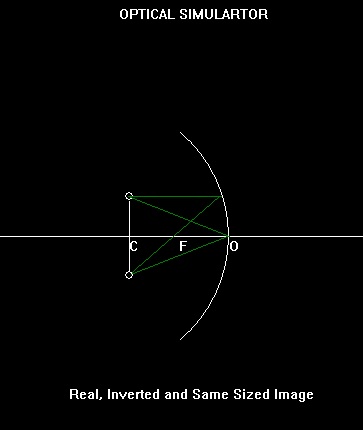
On selecting 10. Concave Mirror and entering required values :



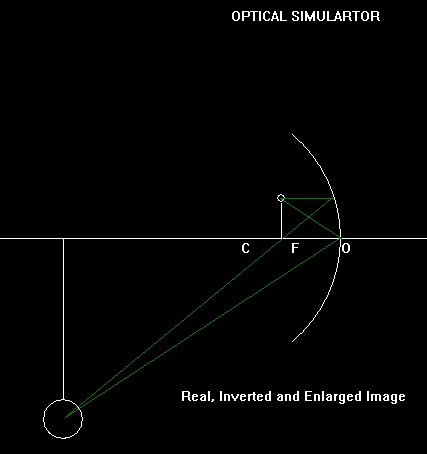
Result :



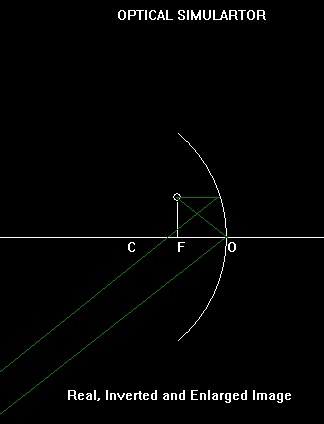
For Concave Mirror of focal length 50 units with object of height 40 units at a distance of 100 units from the pole(O) :



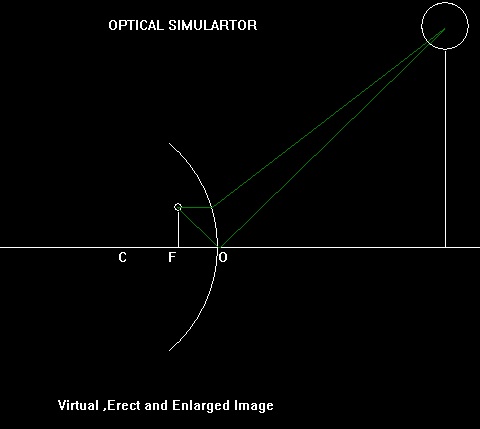
For Concave Mirror of focal length 50 units with object of height 40 units at a distance of 60 units from the pole(O) :



For Concave Mirror of focal length 50 units with object of height 40 units at a distance of 50 units from the pole(O) :



For Concave Mirror of focal length 50 units with object of height 40 units at a distance of 40 units from the pole(O) :



**case 2: All convex lens cases.**

If we select choice 2 mean convex lens.

\*After enter the choice 2 for convex lens we need to enter the value of focal length,distance of object and height of the object of convex lens.

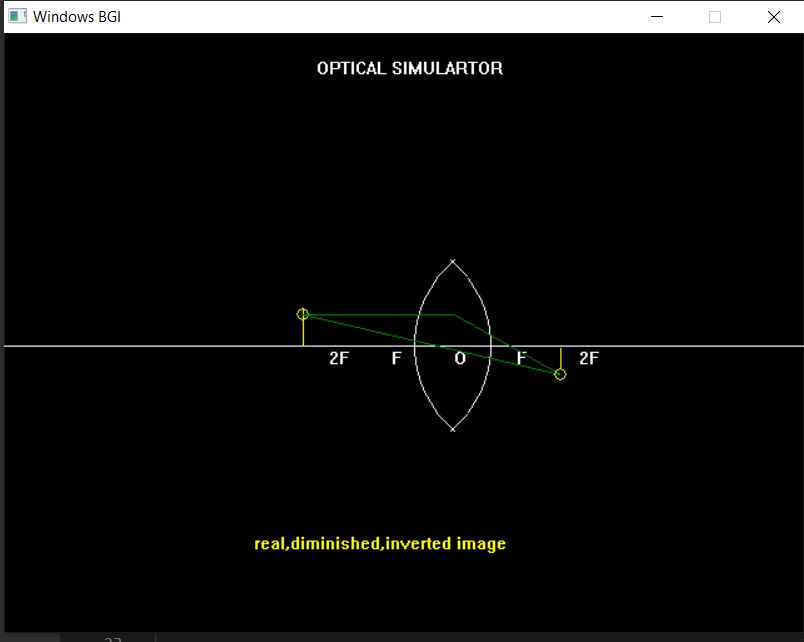
For convex lens of focal length 50 units with object of height 40 units at a distance of 120 units from the pole(O) :

enter the value of focal length of the lens :50

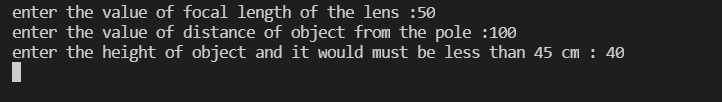
enter the value of distance of object from the pole :120

enter the height of object and it would must be less than 45 cm :40

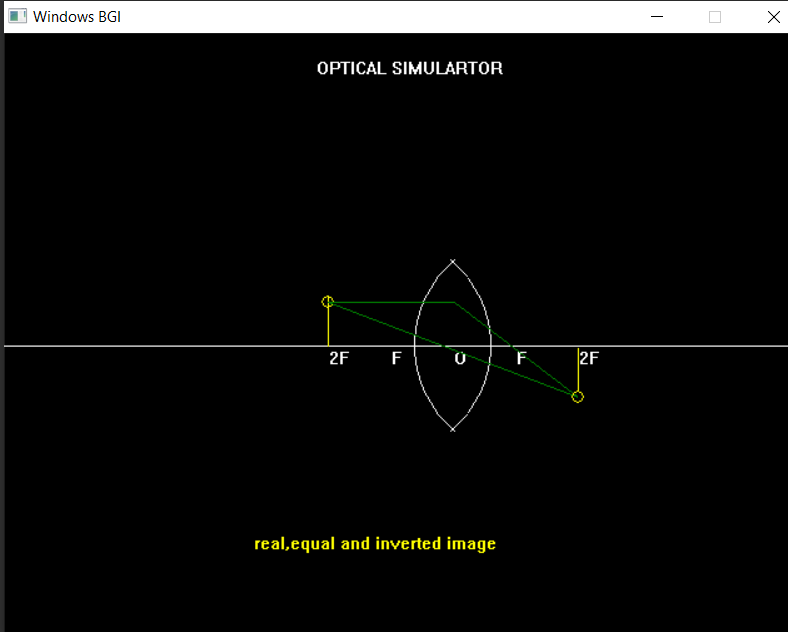
**Result;**



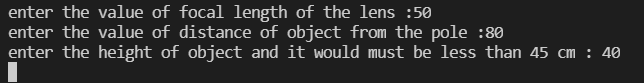
For convex lens of focal length 50 units with object of height 40 units at a distance of 100units from the pole(O) :



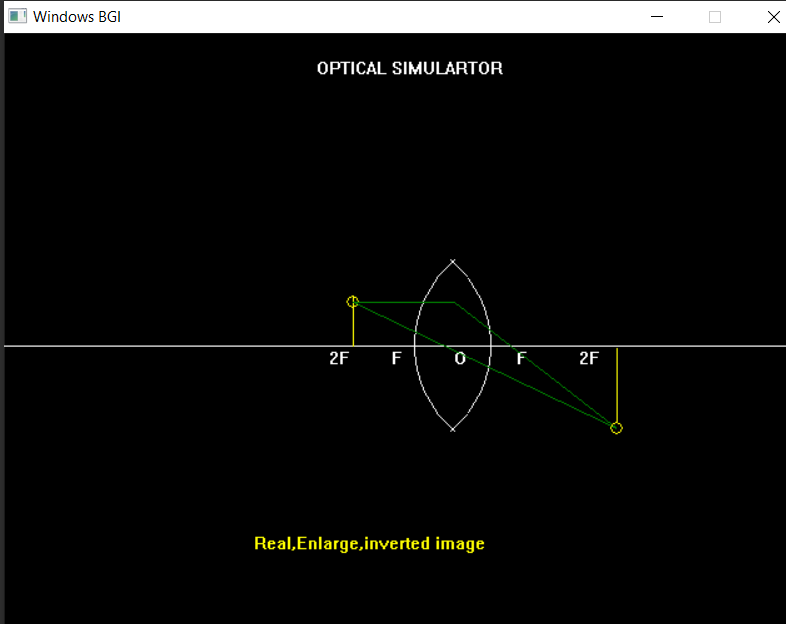
**Result:**



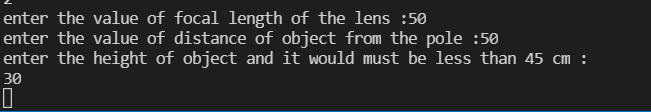
For convex lens of focal length 50 units with object of height 40 units at a distance of 80units from the pole(O) :



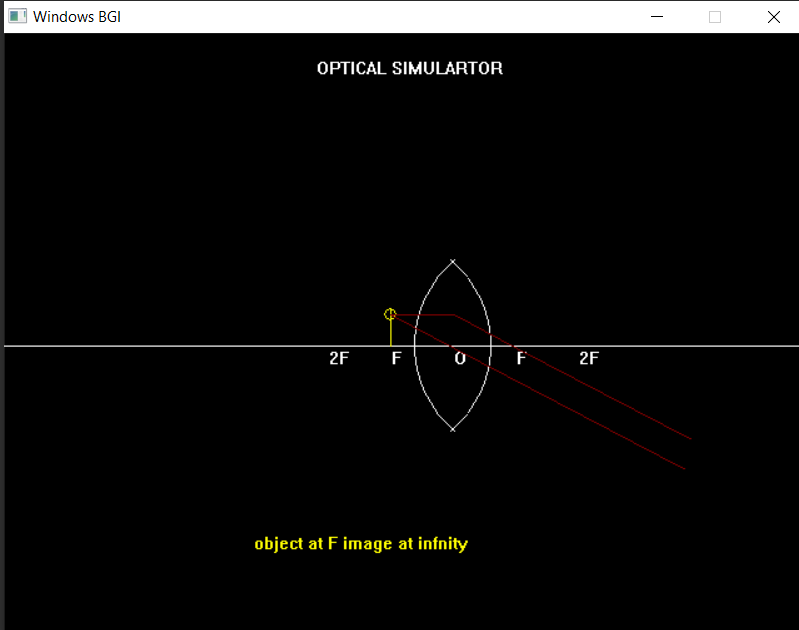
**Result:**



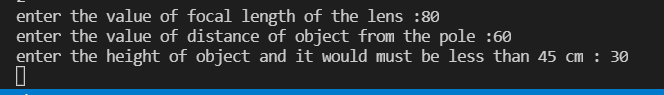
**for the object at focus:**



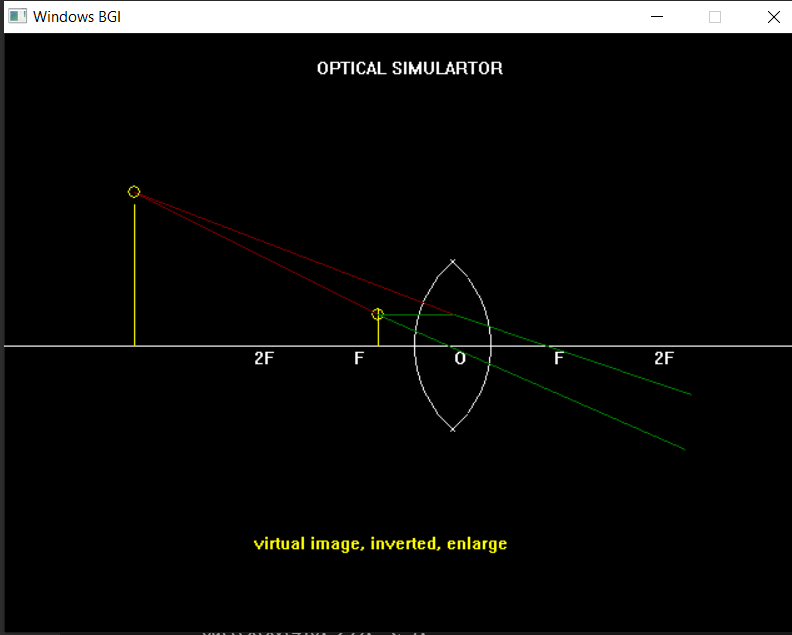
**RESULT:**



If the object is between focus and pole.



RESULT:



**CASE 3:**

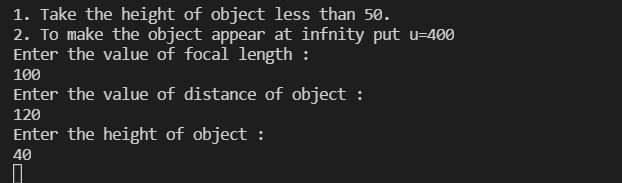
**for concave lens.**

\*If we select the choice 3 (concave lens)

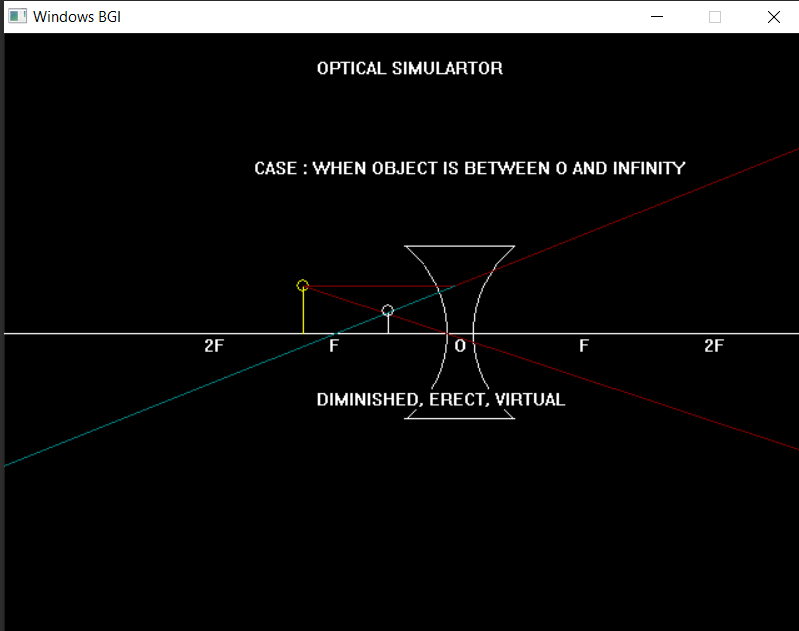
we get choices like,



after selecting 1.



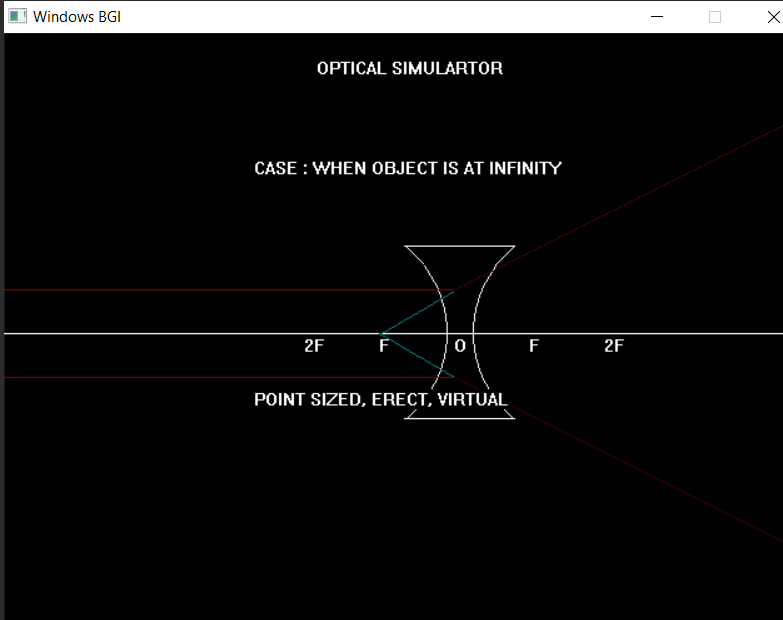
**RESULT:**



if we press 2.



RESULT:

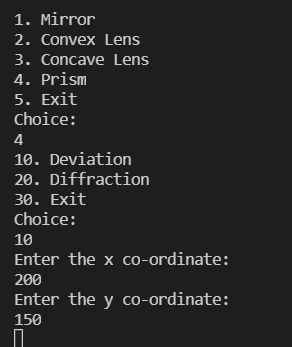


**CASE 4:**  PRISM

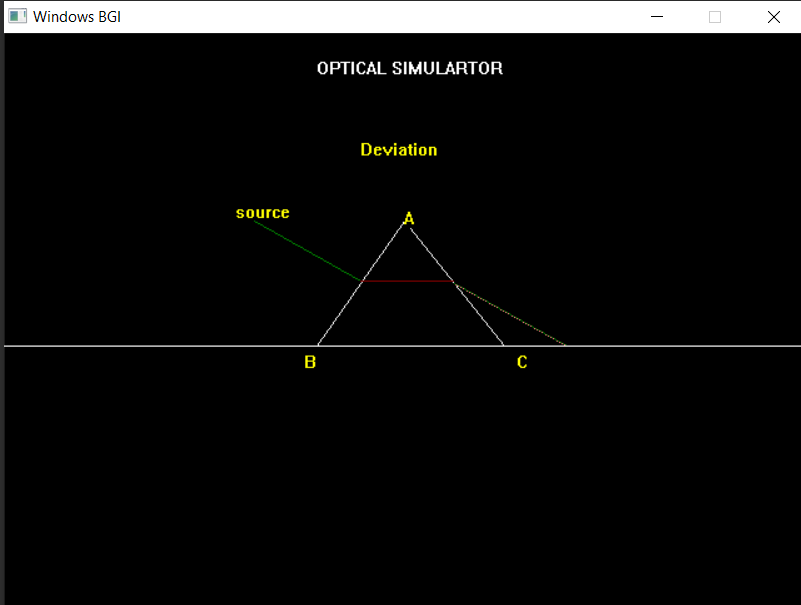
If we select the choice 4.

\*we get two option 10. deviation and 20. diffraction.

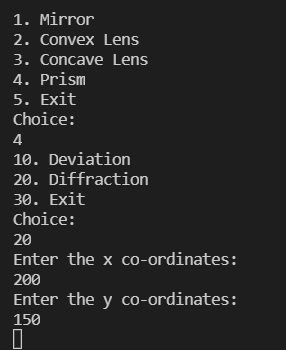
\*if we select 10. deviation, we need to enter two option x,y cordinate.



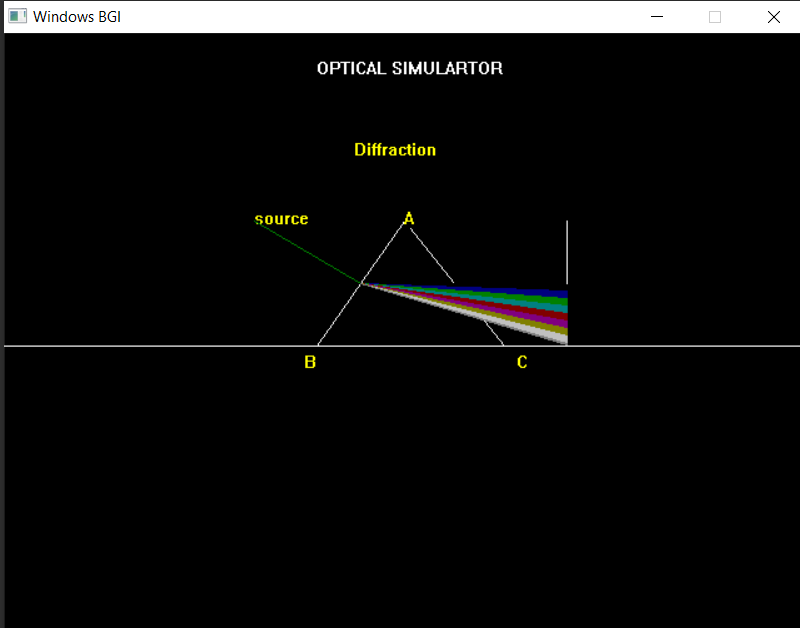
RESULT:



\*if we select 10. deviation, we need to enter two option x,y cordinate.



RESULT:



**Future Scope & Conclusion:**

This optical simulator was developed from a deep need to reduce the multiple trial and error iteration in the lab among scientists, researchers and engineers in the optics community. Currently, this project is a prototype which will be further enhanced with more features like 3D animation, Real Time simuluation, enhanced graphics, more optical instruments, etc. Then, the final product will be a free, cloud-based optical design and simulation platform with no installation is needed that runs on a standard browser. The platform will provide remote access to a virtual lab with optomechanics, lasers and detectors. Both professionals and students can run an optical simulation by customizing optical elements. It will be available across the globe change the way scientists, researchers, and engineers design and simulate the implementation of optical prototypes and setups. We are very excited to bring an innovative cloud platform with enhanced simulation features combined with accessibility and ease of use. Our users can run a complete optical simulation in a short time and share the design with their friends and colleagues, and make smart procurement decisions before they go into the lab.

**Reference:**

We had taken some of the idea from our classes that our teachers teach us about the graphics like: bresenham line drawing algorithm, outextxy fuction,line,arc,etc. some of the other idea we had taken from the intrenet like: youtube,wikipedia,quora,etc and books of “Michale Abrash” and “Pauline Baker”.