



**SAVEETHA
ENGINEERING COLLEGE**

AUTONOMOUS

Affiliated to Anna University | Approved by AICTE



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**19IT402 /
GAME DEVELOPMENT
TECHNOLOGIES**

LAB MANUAL

Vision of the Institute

To be and to be recognized for setting the standard of excellence in engineering education and high quality research in Science and Technology

Mission of the Institute

To promote academic excellence; widen intellectual horizon; self-discipline and high ideals for the total personality development of the individual

Vision of the Department

To uniquely position the department and to establish synergistic relationships across the entire spectrum of disciplines involved with computing by our faculty contributing to Computer Science and devoting themselves to take the maximal advantage of modern Computer Science to solve a wide range of complex, scientific, technological and social problems.

Mission of the Department

1. To pursue our vision by striving for excellence in creating, applying, and imparting knowledge in Computer Science and Engineering.
2. To pursue a comprehensive educational system, research in collaboration with industry and Government and to disseminate knowledge through scholarly publications.
3. To provide service through professional societies to the community, the state, and the nation.

GENERAL GUIDELINES / DOS AND DON'TS INSIDE LABORATORY

DO's:




1. **Follow Course Instructions:** Adhere to the guidelines provided by the instructor for assignments, projects, and software usage.
2. **Use Laptops Responsibly:** Handle your laptop with care and ensure it is in good working condition.
3. **Maintain a Clean Workspace:** Keep your study area tidy and free from food or drinks to prevent accidental damage.
4. **Ensure Regular Backups:** Save important files to cloud storage or external devices to avoid data loss.
5. **Keep Software Updated:** Use licensed software and keep your operating system and applications updated.
6. **Follow Cybersecurity Best Practices:** Use strong passwords, enable antivirus protection, and avoid suspicious links or downloads.
7. **Charge Your Laptop Before Class:** Ensure your device is fully charged to minimize disruptions.
8. **Respect Online and Offline Learning Spaces:** Be punctual for classes and follow online etiquette when attending virtual sessions.
9. **Collaborate Ethically:** Work with peers responsibly and follow academic integrity guidelines.
10. **Report Technical Issues:** Inform the instructor or IT support about any software or hardware problems promptly.

DON'Ts:

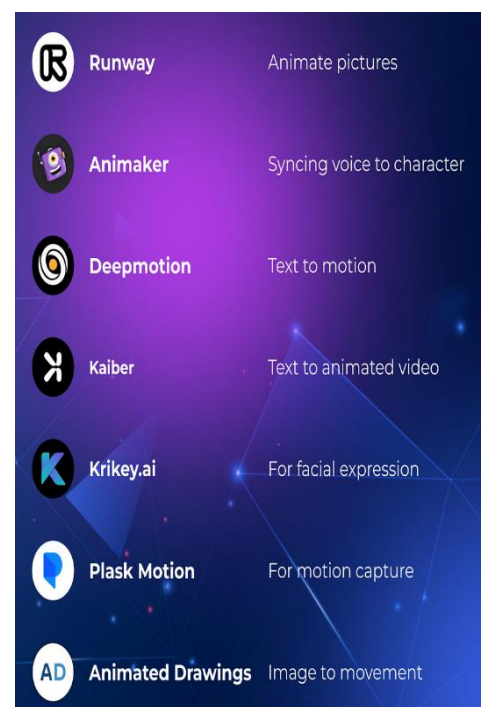
1. **Avoid Installing Unverified Software:** Do not download or install unauthorized applications that may compromise security.
2. **No Hacking or Unauthorized Access:** Never attempt to bypass security settings, access restricted areas, or tamper with system configurations.
3. **Refrain from Distracting Activities:** Avoid gaming, social media browsing, or non-academic use during class.
4. **Do Not Share Credentials:** Keep your login information private to prevent unauthorized access to your accounts.
5. **No Food or Drinks Near Your Laptop:** Spills can cause severe damage to your device.
6. **Avoid Overloading Power Sockets:** Do not connect multiple devices to a single outlet without proper surge protection.
7. **No Loud Conversations in Study Areas:** Maintain a quiet environment for focused learning.
8. **Do Not Leave Your Laptop Unattended:** Secure your device at all times to prevent theft or misuse.
9. **Avoid Plagiarism or Academic Dishonesty:** Ensure all work submitted is original and properly cited.
10. **Do Not Ignore Software or System Updates:** Keeping your system updated enhances security and performance.

LIST OF EXPERIMENTS

S. NO	NAME OF THE EXPERIMENT	QR CODE
1.	<p>Implementation of DDA Algorithm to draw a Line.</p> <p>https://github.com/ramyadevirsec/GameDevelopment/blob/main/Bresenham%E2%80%98s%20Line%20Drawing%20Algorithm</p>	
2.	<p>Implementation of Bresenham's Algorithm to draw a Line.</p> <p>https://github.com/ramyadevirsec/GameDevelopment/blob/main/Bresenham%E2%80%98s%20Line%20Drawing%20Algorithm</p>	
3.	<p>Implementation of Mid-Point Algorithm to draw a Circle.</p> <p>https://github.com/ramyadevirsec/GameDevelopment/blob/main/Circle%20Drawing%20Algorithm</p>	
4.	<p>Implementation of Mid-Point Algorithm to draw an Ellipse.</p> <p>https://github.com/ramyadevirsec/GameDevelopment/blob/main/ELLIPSE%20DRAWING%20ALGORITHM</p>	
5.	<p>Implementation of Two-Dimensional Transformations: To perform Translation, Rotation, Scaling, Reflection, Shear.</p> <p>https://github.com/ramyadevirsec/GameDevelopment/blob/main/Two-Dimensional%20Transformations</p>	
6.	<p>Implementation of Cohen Sutherland Algorithm to apply line clipping and Windowing.</p> <p>https://github.com/ramyadevirsec/GameDevelopment/blob/main/LINE%20CLIPPING%20ALGORITHM</p>	
7	<p>Implementation of 3D Transformations - Translation, Rotation, Scaling on a Cube.</p> <p>https://github.com/ramyadevirsec/GameDevelopment/blob/main/THREE%20DIMENSIONAL%20TRANSFORMATIONS</p>	

8	Implementation of 3D Transformations to create a Projection of a Scene. https://github.com/ramyadevirsec/GameDevelopment/blob/main/THREE%20DIMENSIONAL%20IMAGE%20PROJECTIONS	
9	Implementation of Motion Tweening, Shape Tweening, Guide layer and Masking using Flash. https://github.com/ramyadevirsec/GameDevelopment/blob/main/ANIMATION%20USING%20FLASH	
10	Implementation of editing an Image using an Image Editing Software – Photoshop. https://github.com/ramyadevirsec/GameDevelopment/blob/main/IMAGE%20EDITING	

Commonly used Tools for Editing Images and Video



Exp 1 :

DDA LINE DRAWING ALGORITHM

AIM :

To implement the DDA line drawing algorithm for line using a c coding.

EQUIPMENT REQUIRED:

- **Hardware:** Personal Computer (PC)
- **Software:** C Compiler

ALGORITHM :

Step 1 : Start.

Step 2 : Initialize the graphics header files and functions.

Step 3 : Declare the required variables and functions.

Step 4 : Get the four points for drawing a line namely x_1, x_2, y_1, y_2 .

Step 5 : Draw the line using the algorithm.

Step 6 : Display the output.

Step 7 : stop.

PROGRAM :

```
#include<stdio.h>

#include<conio.h>

#include<math.h>

#include<graphics.h>

main()

{

    int gd=DETECT,gm;

    int xa,xb,ya,yb;

    float step,dx,dy;

    float x,y,xinc,yinc;

    initgraph(&gd,&gm,"C://TURBOC3//BGI");

    printf("Enter The Two Left Endpoints(xa,ya):\n");

    scanf("%d%d",&xa,&ya);

    printf("Enter The Two Right Endpoints(xb,yb):\n");

    scanf("%d%d",&xb,&yb);

    dx=abs(xa-xb);

    dy=abs(ya-yb);

    if(dx >= dy)

        step=dx;

    else

        step =dy;

    x=xa; y=ya;

    putpixel(x, y, 10);

    xinc=dx/step;

    yinc=dy/step;

    while(x <= xb)

    {

        x = x + xinc;

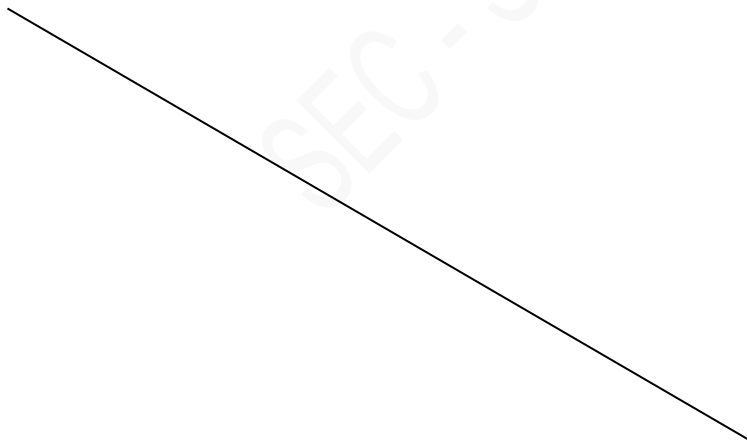
        y = y + yinc;
```

```
        putpixel(x, y, 10);  
    }  
  
    getch();  
    return(0);  
}
```

OUTPUT :

Enter The Two Left Endpoints(xa,ya): 250 140

Enter The Two Right Endpoints(xb,yb): 450 360



RESULT: Thus the DDA algorithm was executed and verified successfully.

Ex.no.: 2

BRESENHAM'S ALGORITHM

AIM :

To implement the Bresenham's algorithm for line using a c coding.

EQUIPMENT REQUIRED:

- **Hardware:** Personal Computer (PC)
- **Software:** C Compiler

ALGORITHM :

Step 1 : Start.

Step 2 : Initialize the graphics header files and functions.

Step 3 : Declare the required variables and functions.

Step 4 : Get the four points for drawing a line namely x1,x2,y1,y2.

Step 5 : Draw the line using the algorithm.

Step 6 : Display the output.

Step 7 : stop.

PROGRAM :

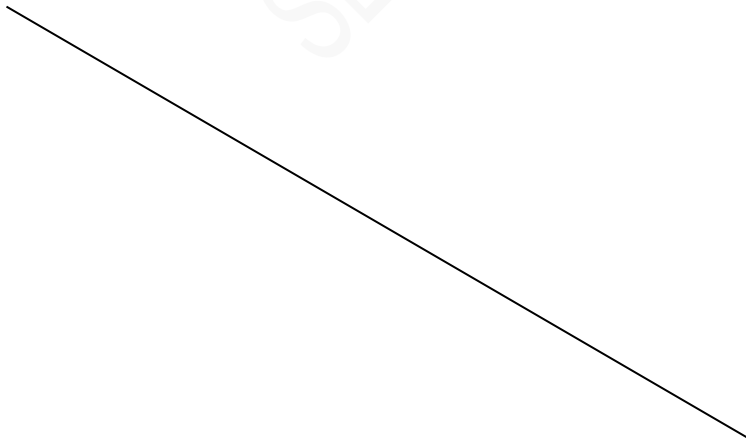
```
#include "stdio.h"
#include "conio.h"
#include "math.h"
#include "graphics.h"
main()
{
    int gd=DETECT,gm;
    int xa,xb,ya,yb;
    int dx,dy,x,y,xend,p;
    initgraph(&gd,&gm,"c:\\tc\\bgi");
    printf("Enter The Two Left Endpoints(xa,ya):\n");
    scanf("%d%d",&xa,&ya);
    printf("Enter The Two Right Endpoints(xb,yb):\n");
    scanf("%d%d",&xb,&yb);
    dx=abs(xa-xb);
    dy=abs(ya-yb);
    p=2*dy-dx;
    if(xa>xb)
    {
        x=xb;
        y=yb;
        xend=xa;
    }
else
{
    x=xa;
    y=ya;
    xend=xb;
}
    putpixel(x,y,6);
    while(x<xend)
    {
        x=x+1;
        if(p<0)
        {
            p=p+2*dy;
        }
```

```
else
{
y=y+1;
p=p+2*(dy-dx);
}
putpixel(x,y,6);
}
getch();
return(0);
}
```

OUTPUT :

Enter The Two Left Endpoints(xa,ya): 234 124

Enter The Two Right Endpoints(xb,yb): 578 321



RESULT: Thus the Bresenham's algorithm was executed and verified successfully.

Ex.no.: 3

MIDPOINT CIRCLE DRAWING ALGORITHM

AIM :

To implement the Bresenham's algorithm for circle using a c coding.

EQUIPMENT REQUIRED:

- **Hardware:** Personal Computer (PC)
- **Software:** C Compiler

ALGORITHM :

Step 1 : Start.

Step 2 : Initialize the graphics header files and functions.

Step 3 : Declare the required variables and functions.

Step 4 : Get the co-ordinates and radius of the circle.

Step 5 : Draw the circle using the algorithm.

Step 6 : Display the output.

Step 7 : stop.

PROGRAM :

```
#include "stdio.h"
#include "conio.h"
#include "math.h"
#include "graphics.h"
main()
{
    int gd=DETECT,gm;
    int xcenter,ycenter,radius;
```

```

int p,x,y;
initgraph(&gd,&gm,"c:\\tc\\bgi");
x=0;
printf("Enter The Radius Value:\n");
scanf("%d",&radius);
y=radius;
printf("Enter The xcenter and ycenter Values:\n");
scanf("%d%d",&xcenter,&ycenter);
plotpoints(xcenter,ycenter,x,y);
p=1-radius;

```

```

while(x<y)
{
    if(p<0)
        x=x+1;
    else
    {
        x=x+1;
        y=y-1;
    }
    if(p<0)
        p=p+2*x+1;
    else
        p=p+2*(x-y)+1;
    plotpoints(xcenter,ycenter,x,y);
}

```

```

getch();
return(0);

```

```

}

```

```

int plotpoints(int xcenter,int ycenter,int x,int y)

```

```

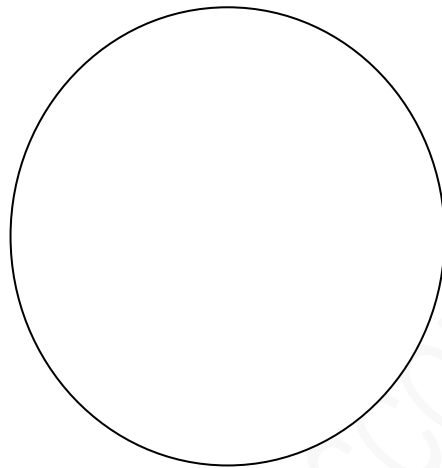
{
    putpixel(xcenter+x,ycenter+y,1);
    putpixel(xcenter-x,ycenter+y,1);
    putpixel(xcenter+x,ycenter-y,1);
    putpixel(xcenter-x,ycenter-y,1);
    putpixel(xcenter+y,ycenter+x,1);
    putpixel(xcenter-y,ycenter+x,1);
    putpixel(xcenter+y,ycenter-x,1);
}

```

OUTPUT :

Enter The Radius Value : 80

Enter The xcenter and ycenter Values : 230 260



RESULT:

The Bresenham's algorithm for circle was executed and verified successfully.

Ex.no.: 4

MIDPOINT ELLIPSE DRAWING ALGORITHM

AIM :

To implement the Bresenham's algorithm for ellipse using a c coding.

EQUIPMENT REQUIRED:

- **Hardware:** Personal Computer (PC)
- **Software:** C Compiler

ALGORITHM :

Step 1 : Start.

Step 2 : Initialize the graphics header files and functions.

Step 3 : Declare the required variables and functions.

Step 4 : Get the co-ordinates and radius of the ellipse.

Step 5 : Draw the ellipse using the algorithm.

Step 6 : Display the output.

Step 7 : stop.

PROGRAM :

```
#include "stdio.h"
#include "conio.h"
#include "math.h"
#include "graphics.h"
main()
{
    int gd=DETECT,gm;
    int xcenter,ycenter,rx,ry;
    int p,x,y,px,py,rx1,ry1,rx2,ry2;
    initgraph(&gd,&gm,"c:\\tc\\bgi");
    printf("Enter The Radius Value:\n");
    scanf("%d%d",&rx,&ry);
    printf("Enter The xcenter and ycenter Values:\n");
    scanf("%d%d",&xcenter,&ycenter);
    ry1=ry*ry;
    rx1=rx*rx;
    ry2=2*ry1;
    rx2=2*rx1;
x=0;
    y=ry;
    plotpoints(xcenter,ycenter,x,y);
    p=(ry1-rx1*ry+(0.25*rx1));
    px=0;
    py=rx2*y;
    while(px<py)
    {
        x=x+1;
        px=px+ry2;
```



```
if(p>=0)
    y=y-1;
    py=py-rx2;
```

```
if(p<0)
    p=p+ry1+px;
else
    p=p+ry1+px-py;
plotpoints(xcenter,ycenter,x,y);
```

```
p=(ry1*(x+0.5)*(x+0.5)+rx1*(y-1)*(y-1)-rx1*ry1);
while(y>0)
```

```
{
    y=y-1;
    py=py-rx2;
    if(p<=0)
    {
        x=x+1;
        px=px+ry2;
```

```
    }
    if(p>0)
        p=p+rx1-py;
    else
        p=p+rx1-py+px;
```

```
    plotpoints(xcenter,ycenter,x,y);
```

```
    }
}
```

```
    getch();
    return(0);
```

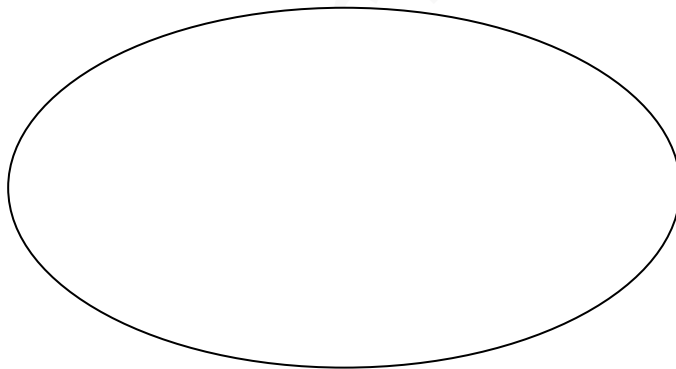
```
}
```

```
int plotpoints(int xcenter,int ycenter,int x,int y)
{
    putpixel(xcenter+x,ycenter+y,6);
    putpixel(xcenter-x,ycenter+y,6);
    putpixel(xcenter+x,ycenter-y,6);
    putpixel(xcenter-x,ycenter-y,6);
}
```

OUTPUT :

Enter The Radius Value(Rx,Ry) : 10 30

Enter The xcenter and ycenter Values : 300 150



RESULT:

The Midpoint Algorithm for Ellipse was executed and verified successfully.

EX.NO : 5

TWO DIMENSIONALTRANSFORMATION

AIM

To write a c program to implement 2D transformation of image.

EQUIPMENT REQUIRED:

- **Hardware:** Personal Computer (PC)
- **Software:** C Compiler

ALGORITHM

Step 1:Start the program.

Step 2:Draw the image with default parameters.

Step 3 : Get the choice from the user.

Step 4: Get the parameters for transformation.

Step 5 : Perform the transformation.

Step 6 : Draw the image.

Step 7 : Stop the program.

SOURCE CODE

```
# include<graphics.h>

# include<stdio.h>

# include<conio.h>

# include<math.h>


void main()

{

    int gd=DETECT,gm,i,j,k,ch;

    float tx,ty,x,y,ang,n,temp;

    float  a[5][3],si,co,b[5][3],c[5][3];


    initgraph(&gd,&gm,"e:\\tcpp\\bgi");

        n=4;

    a[0][0]=0;  a[0][1]=0;  a[1][0]=100; a[1][1]=0;

    a[2][0]=100; a[2][1]=100; a[3][0]=0;  a[3][1]=100;

    a[4][0]=0;  a[4][1]=0;

    while(1)

    {

        cleardevice();

        gotoxy(1,8);

        printf("\n\t***** Program to perform 2-D Transformations *****");

        printf("\n\t\t\t 1. Accept the polygon");

        printf("\n\t\t\t 2. Perform translation");

        printf("\n\t\t\t 3. Perform scaling");

        printf("\n\t\t\t 4. Perform rotation");

        printf("\n\t\t\t 5. Perform reflection");
```

```

printf("\n\t\t\t 6. Perform shearing");
printf("\n\t\t\t 7. Exit");
printf("\n\t\t\t Enter your choice::");
scanf("%d",&ch);
switch(ch)
{
case 1:
    cleardevice();
    gotoxy(1,1);
    printf("\n\tEnter no of points::");
    scanf("%f",&n);
    for(i=0;i<n;i++)
    {
        printf("\n\t Enter x,y co-ordinates for %d::",i+1);
        scanf("%f %f",&a[i][0],&a[i][1]);
    }
    a[i][0]=a[0][0];
    a[i][1]=a[0][1];
    cleardevice();
    for(i=0;i<n;i++)
        line(320+a[i][0],240-a[i][1],320+a[i+1][0],240-a[i+1][1]);
    line(0,240,639,240);
    line(320,0,320,479);
    getch();
    break;

case 2:
    cleardevice();

```

```

for(i=0;i<n;i++)
line(320+a[i][0],240-a[i][1],320+a[i+1][0],240-a[i+1][1]);

line(0,240,639,240);

line(320,0,320,479);

gotoxy(1,1);

printf("Enter translation vectors tx and ty\n\t");

scanf("%f %f",&x,&y);

cleardevice();

for(i=0;i<n;i++)
line(320+a[i][0]+x,240-(a[i][1]+y),320+a[i+1][0]+x,240-(a[i+1][1]+y));

line(0,240,639,240);

line(320,0,320,479);

getch();

break;

case 3:

cleardevice();

for(i=0;i<n;i++)

line(320+a[i][0],240-a[i][1],320+a[i+1][0],240-a[i+1][1]);

line(0,240,639,240);

line(320,0,320,479);

gotoxy(1,1);

printf("Enter scaling vectors tx and ty\n\t");

scanf("%f %f",&x,&y);

if(x==0)

x=1;

if(y==0)

```

```
y=1;
cleardevice();
for(i=0;i<n;i++)
line(320+(a[i][0]*x),240-(a[i][1]*y),320+(a[i+1][0]*x),240-(a[i+1][1]*y));
line(0,240,639,240);
line(320,0,320,479);
getch();
break;
```

case 4:

```
cleardevice();
for(i=0;i<n;i++)
line(320+a[i][0],240-a[i][1],320+a[i+1][0],240-a[i+1][1]);
line(0,240,639,240);
line(320,0,320,479);
gotoxy(1,1);
printf("Enter the angle of rotation\n\t");
scanf("%f",&ang);
ang=ang*0.01745;
gotoxy(1,3);
printf("Enter point of rotation\n\t");
scanf("%f %f",&x,&y);
gotoxy(1,5);
printf("1.clockwise 2.anticlockwise\n\t");
scanf("%d",&k);
si=sin(ang);
co=cos(ang);
```

```

    for(i=0;i<n+1;i++)
    {
        c[i][0]=a[i][0];
        c[i][1]=a[i][1];
        c[i][2]=1;
    }

    b[0][0]=co;
    b[0][1]=si;
    b[0][2]=0;
    b[1][0]=(-si);
    b[1][1]=co;
    b[1][2]=0;
    b[2][0]=(-x*co)+(y*si)+x;
    b[2][1]=(-x*si)-(y*co)+y;
    b[2][2]=1;
    if(k==1)
    {
        b[0][1]=(-si);
        b[1][0]=(si);
        b[2][0]=(-x*co)-(y*si)+x;
        b[2][1]=(-x*si)+(y*co)+y;
    }

```

```

for(i=0;i<n+1;i++)

```

```

{

```

```

    for(j=0;j<3;j++)

```

```

    {

```

```

        a[i][j] = 0 ;

```



```

        for(k=0;k<3;k++)
            a[i][j] = a[i][j] + c[i][k] * b[k][j] ;
    }
}

```

```

cleardevice();
for(i=0;i<n;i++)
    line(320+a[i][0],240-a[i][1],320+a[i+1][0],240-a[i+1][1]);
line(0,240,639,240);
line(320,0,320,479);
getch();
break;

```

case 5:

```

cleardevice();
for(i=0;i<n;i++)
    line(320+a[i][0],240-a[i][1],320+a[i+1][0],240-a[i+1][1]);
line(0,240,639,240);
line(320,0,320,479);
gotoxy(1,1);
printf("\n1.Reflection about Y-axis");
printf("\n2.Reflection about X-axis");
printf("\n3.Reflection about origin");
printf("\n4.Reflection about line y=x");
printf("\n5.Reflection about line y=-x");
printf("\nEnter your choice:");
scanf("%d",&ch);

```

```
switch(ch)
{
case 1:
    for(i=0;i<n+1;i++)
        a[i][0]=a[i][0]*(-1);
    /* Plot the polygon */
    cleardevice();
    for(i=0;i<n;i++)
        line(320+a[i][0],240-a[i][1],320+a[i+1][0],240-a[i+1][1]);
    line(0,240,639,240);
    line(320,0,320,479);
    getch();
    break;
```

```
case 2:
    for(i=0;i<n+1;i++)
        a[i][1]=a[i][1]*(-1);
        cleardevice();
    for(i=0;i<n;i++)
        line(320+a[i][0],240-a[i][1],320+a[i+1][0],240-a[i+1][1]);
    line(0,240,639,240);
    line(320,0,320,479);
    getch();
    break;
```

```
case 3:
    for(i=0;i<n+1;i++)
    {
```

```
a[i][1]=a[i][1]*(-1);
a[i][0]=a[i][0]*(-1);
}

cleardevice();

for(i=0;i<n;i++)
line(320+a[i][0],240-a[i][1],320+a[i+1][0],240-a[i+1][1]);

line(0,240,639,240);

line(320,0,320,479);

getch();
break;
```

case 4:

```
for(i=0;i<n+1;i++)
{
temp=a[i][0];
a[i][0]=a[i][1];
a[i][1]=temp;
}

cleardevice();

for(i=0;i<n;i++)
line(320+a[i][0],240-a[i][1],320+a[i+1][0],240-a[i+1][1]);

line(0,240,639,240);

line(320,0,320,479);

line(0,479,639,0);

getch();
break;
```

case 5:

```
for(i=0;i<n+1;i++)  
{  
temp=a[i][0];  
a[i][0]=a[i][1];  
a[i][1]=temp;  
}
```

```
for(i=0;i<n+1;i++)  
{  
a[i][1]=a[i][1]*(-1);  
a[i][0]=a[i][0]*(-1);  
}
```

```
cleardevice();
```

```
for(i=0;i<n;i++)
```

```
line(320+a[i][0],240-a[i][1],320+a[i+1][0],240-a[i+1][1]);
```

```
line(0,240,639,240);
```

```
line(320,0,320,479);
```

```
line(0,0,639,479);
```

```
getch();
```

```
break;
```

```
default:
```

```
break;
```

```
}
```

```
break;
```

```
case 6:
```

```
cleardevice();
```

```
for(i=0;i<n;i++)
```

```
line(320+a[i][0],240-a[i][1],320+a[i+1][0],240-a[i+1][1]);  
line(0,240,639,240);  
line(320,0,320,479);  
gotoxy(1,1);  
printf("\n1.X shear with y reference line");  
printf("\n2.Y shear with x reference line");
```

```
printf("\nEnter your choice:");  
scanf("%d",&ch);  
switch(ch)  
{  
  
    case 1:  
printf("\nEnter the x-shear parameter value:");  
scanf("%f",&temp);  
printf("\nEnter the yref line");  
scanf("%f",&ty);  
        b[0][0]=1;  
        b[0][1]=0;  
        b[0][2]=0;  
        b[1][0]=temp;  
        b[1][1]=1;  
        b[1][2]=0;  
        b[2][0]=(-temp)*(ty);  
        b[2][1]=0;  
        b[2][2]=1;
```

```
for(i=0;i<n+1;i++)
```

```
a[i][2]=1;
```

```
for(i=0;i<n+1;i++)
```

```
{
```

```
    for(j=0;j<3;j++)
```

```
        {
```

```
            c[i][j] = 0 ;
```

```
            for(k=0;k<3;k++)
```

```
                c[i][j] = c[i][j] + a[i][k] * b[k][j] ;
```

```
        }
```

```
}
```

```
cleardevice();
```

```
for(i=0;i<n;i++)
```

```
    line(320+c[i][0],240-c[i][1],320+c[i+1][0],240-c[i+1][1]);
```

```
    line(0,240,639,240);
```

```
    line(320,0,320,479);
```

```
    getch();
```

```
break;
```

```
case 2:
```

```
printf("\nEnter the y-shear parameter value:");
```

```
scanf("%f",&temp);
```

```
    printf("\nEnter the xref line");
```

```
scanf("%f",&tx);
```

```
    b[0][0]=1;
```

```
    b[0][1]=temp;
```

```

b[0][2]=0;
b[1][0]=0;
b[1][1]=1;
b[1][2]=0;
b[2][0]=0;
b[2][1]=(-temp)*(tx);
b[2][2]=0;
for(i=0;i<n+1;i++)
a[i][2]=1;

for(i=0;i<n+1;i++)
{
    for(j=0;j<3;j++)
    {
        c[i][j] = 0 ;
        for(k=0;k<3;k++)
            c[i][j] = c[i][j] + a[i][k] * b[k][j] ;
    }
}

cleardevice();
for(i=0;i<n;i++)
    line(320+c[i][0],240-c[i][1],320+c[i+1][0],240-c[i+1][1]);
    line(0,240,639,240);
    line(320,0,320,479);
    getch();
break;

```

```
default: break;
```

```
}
```

```
break;
```

```
case 7:
```

```
exit(1);
```

```
closegraph();
```

```
restorecrtmode();
```

```
break;
```

```
default: break;
```

```
}
```

```
}}
```

OUTPUT

1. Translation
2. Rotation
3. Scaling
4. Shearing
5. Reflection
6. Exit

TRANSLATION

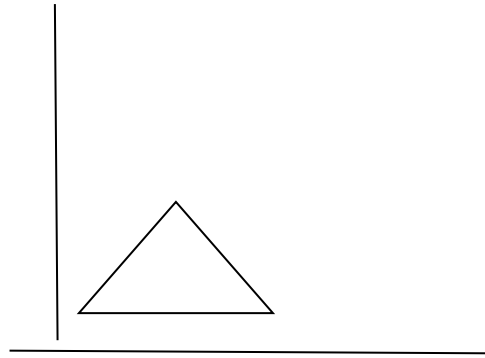
Enter the choice: 1

Enter the number of Vertices: 3

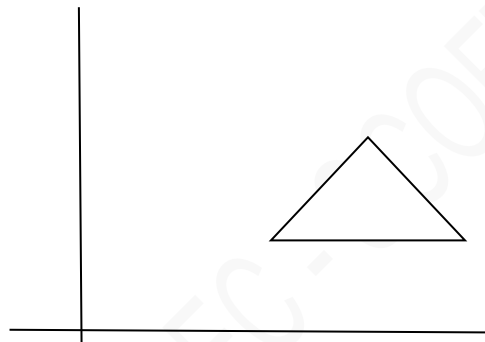
Enter the co-ordinates: 30 150 10 200

Enter the co-ordinates: 10 200 60 200

Enter the co-ordinates: 60 200 30 150



Enter the translation vector Tx, Ty : 90 60



ROTATION

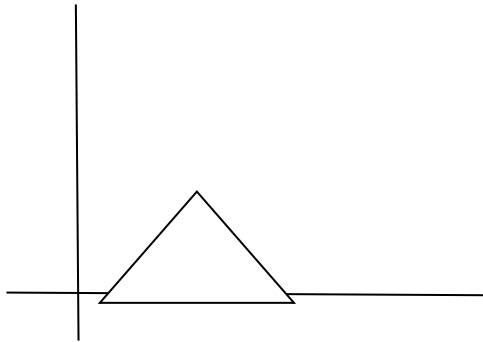
Enter the choice : 2

Enter the number of Vertices: 3

Enter the coordinates : 30 150 10 200

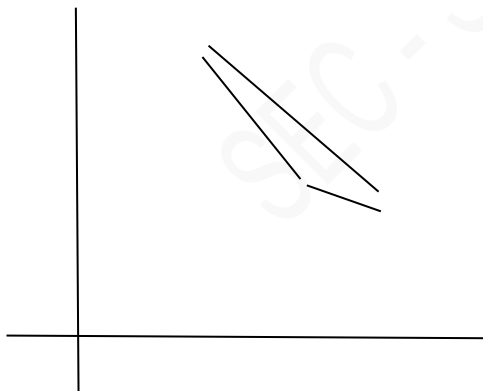
Enter the coordinates : 10 200 60 200

Enter the coordinates : 60 200 30 150



Enter the Rotating Angle : 90

Enter the Pivot Point : 100 200



SCALING

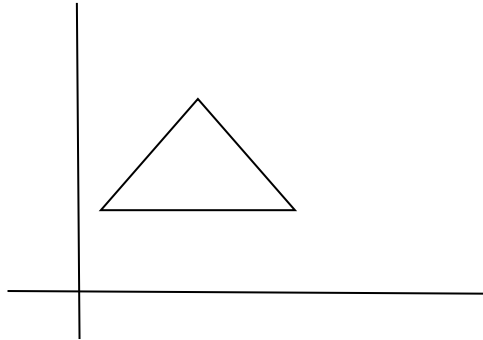
Enter the choice : 3

Enter the number of Vertices: 3

Enter the coordinates : 30 150 10 200

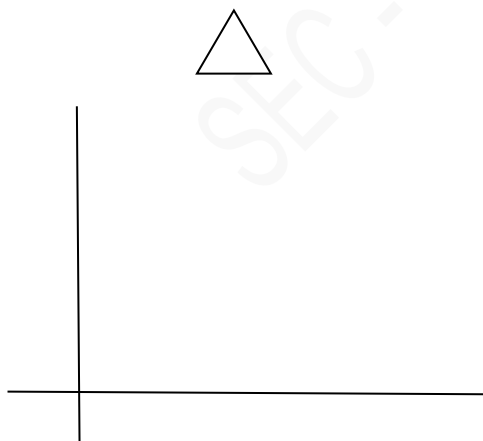
Enter the coordinates : 10 200 60 200

Enter the coordinates : 60 200 30 150



Enter the scaling Factor : 0.3 0.4

Enter the Fixed Point : 100 200



SHEARING

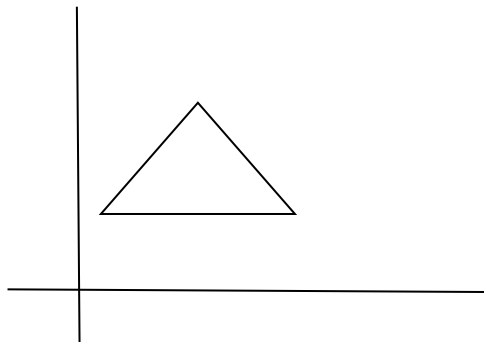
Enter the choice : 4

Enter the number of Vertices: 3

Enter the coordinates : 30 150 10 200

Enter the coordinates : 10 200 60 200

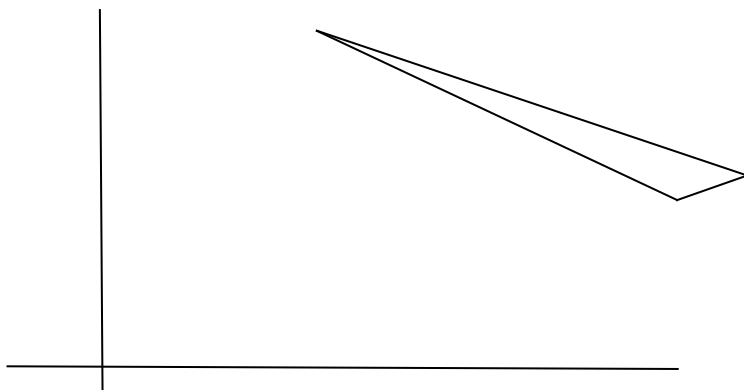
Enter the coordinates : 60 200 30 150



Enter the shear Value : 5

Enter the fixed point : 50 100

Enter the Axis for shearing if x-axis then 1
 if y-axis then 0



REFLECTION

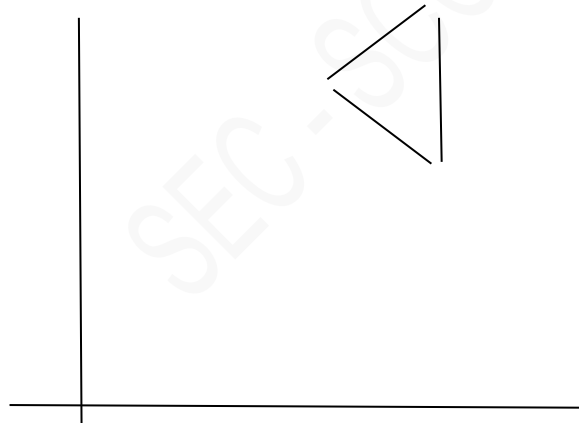
Enter the choice : 5

Enter the number of Vertices: 3

Enter the coordinates : 30 150 10 200

Enter the coordinates : 10 200 60 200

Enter the coordinates : 60 200 30 150



RESULT

Thus the program was completed and the output was obtained successfully.

EX. NO : 6

COHEN SUTHERLAND LINE CLIPPING ALGORITHM

AIM :

To write a C program to perform line clipping using Cohen Sutherland Algorithm.

EQUIPMENT REQUIRED:

- **Hardware:** Personal Computer (PC)
- **Software:** C Compiler

ALGORITHM :

Step 1: Start

Step 2: Get the bottom-left coordinate of view port from the user.

Step 3: Get the top-right coordinate of view port from the user.

Step 4: Get the coordinates of 1st end point of line from the user.

Step 5: Get the coordinates of 2nd endpoint of line from the user.

Step 6: Print the region code of the first and second point.

Step 7: If the points lie inside the view port, print "The line is totally visible".

Step 8: If the starting point lies outside the view port, print "The line is invisible"

Step 9: If the starting point lies inside the view port and the ending point lies outside the view port, print "The line is partially visible"

Step 10: Clip the line present outside the view port.

Step 11: Draw the clipped line present inside the view port.

Step 12: Stop

SOURCE CODE

```
#include<stdio.h>

#include<graphics.h>

//#include<process.h>

void main()

{

    int gd=DETECT, gm;

    float i,xmax,ymax,xmin,ymin,x11,y11,x22,y22,m;

    float a[4],b[4],c[4],x1,y1;

    clrscr();

    initgraph(&gd,&gm,"c:\\tc\\bgi");

    printf("\nEnter the bottom-left coordinate of viewport: ");

    scanf("%f %f",&xmin,&ymin);

    printf("\nEnter the top-right coordinate of viewport: ");

    scanf("%f %f",&xmax,&ymax);

    rectangle(xmin,ymin,xmax,ymax);

    printf("\nEnter the coordinates of 1st end point of line: ");

    scanf("%f %f",&x11,&y11);

    printf("\nEnter the coordinates of 2nd endpoint of line: ");

    scanf("%f %f",&x22,&y22);

    line(x11,y11,x22,y22);

    for(i=0;i<4;i++)

    {

        a[i]=0;

        b[i]=0;

    }

    m=(y22-y11)/(x22-x11);
```

```

if(x11<xmin) a[3]=1;
if(x11>xmax) a[2]=1;
if(y11<ymin) a[1]=1;
if(y11>ymax) a[0]=1;
if(x22<xmin) b[3]=1;
if(x22>xmax) b[2]=1;
if(y22<ymin) b[1]=1;
if(y22>ymax) b[0]=1;
printf("\nRegion code of 1st pt ");
for(i=0;i<4;i++)
{
printf("%f",a[i]);
}
printf("\nRegion code of 2nd pt ");
for(i=0;i<4;i++)
{
printf("%f",b[i]);
}
printf("\nAnding : ");
for(i=0;i<4;i++)
{
c[i]=a[i]&& b[i];
}
for(i=0;i<4;i++)
    printf("%f",c[i]);
getch();
if((c[0]==0)&&(c[1]==0)&&(c[2]==0)&&(c[3]==0))
{

```



```

        if((a[0]==0)&&(a[1]==0)&&(a[2]==0)&&(a[3]==0)&&
(b[0]==0)&&(b[1]==0)&&(b[2]==0)&&(b[3]==0))
    {
        clrscr();
        clearviewport();
        printf("\nThe line is totally visible\nand not a clipping candidate");
        rectangle(xmin,ymin,xmax,ymax);
        line(x11,y11,x22,y22);
        getch();
    }
else
{
    clrscr();
    clearviewport();
    printf("\nLine is partially visible");
    rectangle(xmin,ymin,xmax,ymax);
    line(x11,y11,x22,y22);
    getch();
    if((a[0]==0)&&(a[1]==1))
    {
        x1=x11+(ymin-y11)/m;
        x11=x1;
        y11=ymin;
    }
    else if((b[0]==0)&&(b[1]==1))
    {
        x1=x22+(ymin-y22)/m;
        x22=x1;

```

```

    y22=ymin;
}
if((a[0]==1)&&(a[1]==0))
{
    x1=x11+(ymax-y11)/m;
    x11=x1; y11=ymax;
}
else if((b[0]==1)&&(b[1]==0))
{
    x1=x22+(ymax-y22)/m;
    x22=x1; y22=ymax;
}
if((a[2]==0)&&(a[3]==1))
{
    y1=y11+m*(xmin-x11);
    y11=y1; x11=xmin;
}
else if((b[2]==0)&&(b[3]==1))
{
    y1=y22+m*(xmin-x22);
    y22=y1;
    x22=xmin;
}
if((a[2]==1)&&(a[3]==0))
{
    y1=y11+m*(xmax-x11);
    y11=y1; x11=xmax;
}

```

```

        else if((b[2]==1)&&(b[3]==0))
        {
            y1=y22+m*(xmax-x22);
            y22=y1;
            x22=xmax;
        }

        clrscr();

        clearviewport();

        printf("\nAfter clipping:");

        rectangle(xmin,ymin,xmax,ymax);

        line(x11,y11,x22,y22);

        getch();
    }
}

else
{
    clrscr();

    clearviewport();

    printf("\nLine is invisible");

    rectangle(xmin,ymin,xmax,ymax);

    getch();
}

closegraph();

getch();
}

```

SAMPLE OUTPUT

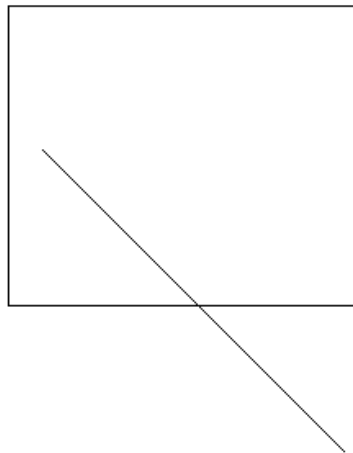
Enter the bottom-left coordinate of viewport: 100 100

Enter the top-right coordinate of viewport: 200 100

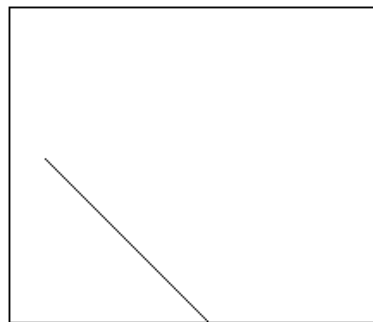
Enter the coordinates of 1st end point of line: 110 150

Enter the coordinates of 2nd endpoint of line: 200 250

The line is partially visible.



After clipping.



RESULT

Thus the program to perform line clipping by Cohen Sutherland algorithm was executed and the output was obtained.

Ex.no.: 7

Date :

THREE DIMENSIONAL TRANSFORMATIONS

AIM :

To implement the various transformations on three dimensional objects using a c coding.

EQUIPMENT REQUIRED:

- **Hardware:** Personal Computer (PC)
- **Software:** C Compiler

ALGORITHM :

Step 1 : Start.

Step 2 : Draw an image with default parameters.

Step 3 : Get the choice from user.

Step 4 : Get the parameters for transformation.

Step 5 : Perform the transformation.

Step 6 : Display the output.

Step 7 : Stop.

PROGRAM :

```
#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

int maxx,maxy,midx,midy;

void axis()

{

getch();

cleardevice();

line(midx,0,midx,maxy);

line(0,midy,maxx,midy);

}


void main()

{

int gd,gm,x,y,z,o,x1,x2,y1,y2;

detectgraph(&gd,&gm);

initgraph(&gd,&gm," ");

setfillstyle(0,getmaxcolor());

maxx=getmaxx();

maxy=getmaxy();

midx=maxx/2;

midy=maxy/2;

axis();

bar3d(midx+50,midy-100,midx+60,midy-90,5,1);

printf("Enter Translation Factor");

scanf("%d%d%d",&x,&y,&z);
```

```

axis();

printf("after translation");

bar3d(midx+(x+50),midy-(y+100),midx+x+60,midy-(y+90),5,1);

axis();

bar3d(midx+50,midy+100,midx+60,midy-90,5,1);

printf("Enter Scaling Factor");

scanf("%d%d%d",&x,&y,&z);

axis();

printf("After Scaling");

bar3d(midx+(x*50),midy-(y*100),midx+(x*60),midy-(y*90),5*z,1);

axis();

bar3d(midx+50,midy-100,midx+60,midy-90,5,1);

printf("Enter Rotating Angle");

scanf("%d",&o);

```

```

x1=50*cos(o*3.14/180)-100*sin(o*3.14/180);
y1=50*cos(o*3.14/180)+100*sin(o*3.14/180);
x2=60*sin(o*3.14/180)-90*cos(o*3.14/180);
y2=60*sin(o*3.14/180)+90*cos(o*3.14/180);

axis();

printf("After Rotation about Z Axis");

bar3d(midx+x1,midy-y1,midx+x2,midy-y2,5,1);

axis();

printf("After Rotation about X Axis");

bar3d(midx+50,midy-x1,midx+60,midy-x2,5,1);

axis();

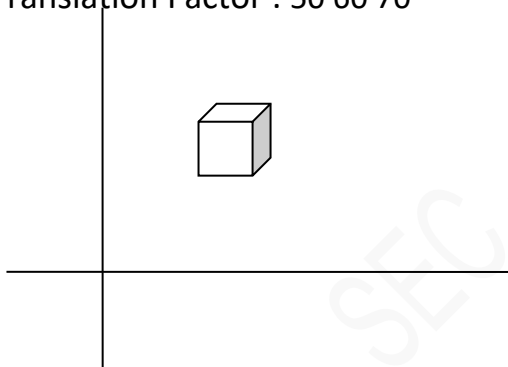
```

```
printf("After Rotation about Y Axis");  
bar3d(midx+x1,midy-100,midx+x2,midy-90,5,1);  
getch();  
closegraph();  
}
```

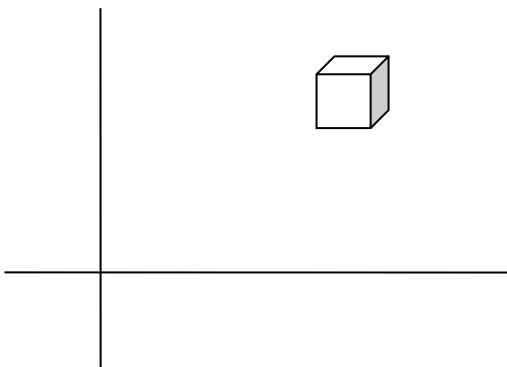
OUTPUT :

Translation

Enter Translation Factor : 50 60 70

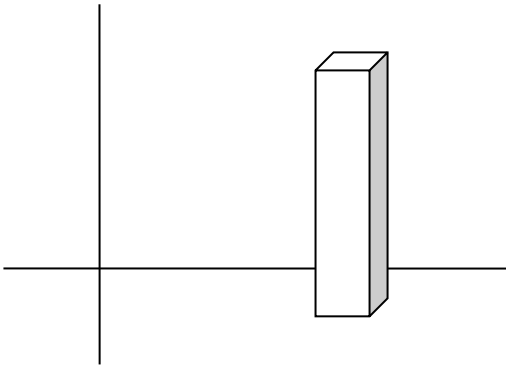


After Translation

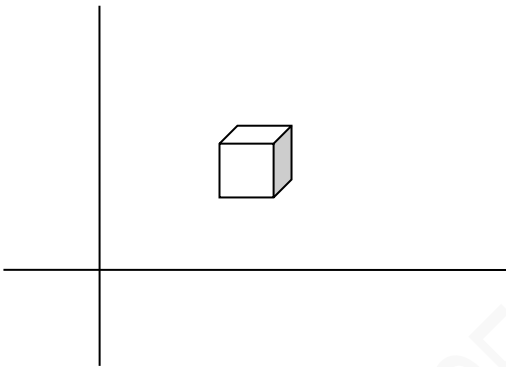


Scaling

Enter Scaling Factor : 80 90 95

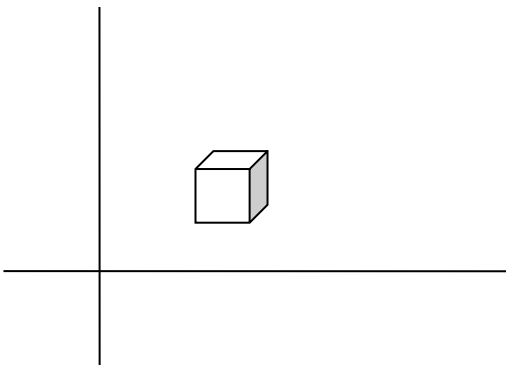


After Scaling

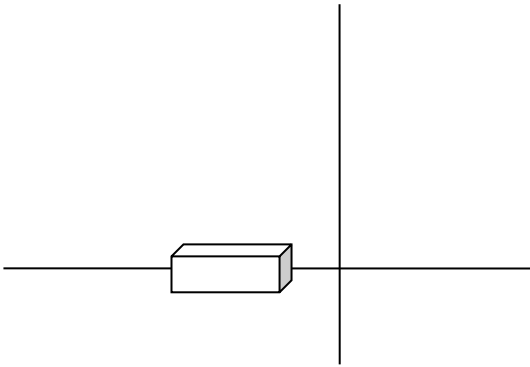


Rotation

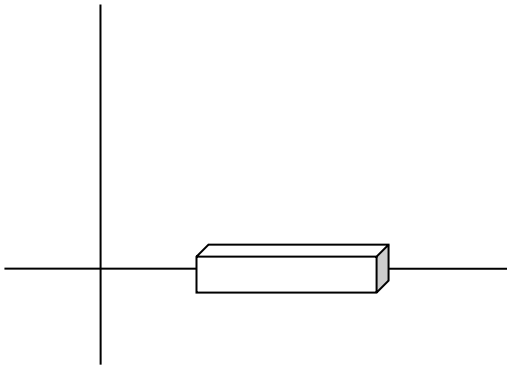
Enter Rotating Angle : 60



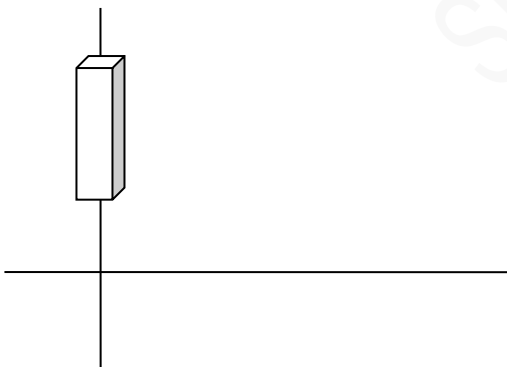
After Rotation about Z-Axis



After Rotation about X-Axis



After Rotation about Y-Axis :



RESULT :

Thus the program was executed and the output was obtained successfully.

Ex.no.: 8

Date :

THREE DIMENSIONAL IMAGE PROJECTIONS

AIM :

To implement the projections in three dimensional image using a C coding.

EQUIPMENT REQUIRED:

- **Hardware:** Personal Computer (PC)
- **Software:** C Compiler

ALGORITHM :

Step 1 : start.

Step 2 : Draw any image in the three dimensional plane.

Step 3 : Get the choice of axis as input from the user.

Step 4 : Perform the projection about the desired axis.

Step 5 : Display the projected image.

Step 6 : Stop.

PROGRAM CODE :

```
#include<stdio.h>

#include<math.h>

#include<conio.h>

#include<stdlib.h>

#include<graphics.h>

int gd=DETECT,gm;

double x1,x2,y1,y2;

void draw_cube(double edge[20][3])

{

int i;

initgraph(&gd,&gm,"..\bgi");

clearviewport();

for(i=0;i<19;i++)

    {

        x1=edge[i][0]+edge[i][2]*(cos(2.3562));

        y1=edge[i][1]-edge[i][2]*(sin(2.3562));

        x2=edge[i+1][0]+edge[i+1][2]*(cos(2.3562));

        y2=edge[i+1][1]-edge[i+1][2]*(sin(2.3562));

        line(x1+320,240-y1,x2+320,240-y2);

    }

line(320,240,320,25);

line(320,240,550,240);

line(320,240,150,410);

getch();

closegraph();

}

void perspect(double edge[20][3])
```

```

{
int ch;

int i;

float p,q,r;

clrscr();

printf("\n -=[ Perspective Projection About ]=-");

printf("\n 1:==>X-Axis ");

printf("\n 2:==>Y-Axis ");

printf("\n 3:==>Z-Axis ");

printf("\n Enter Your Choice :=");

scanf("%d",&ch);

switch(ch)
{
case 1:

printf("\n Enter P :=");

scanf("%f",&p);

for(i=0;i<20;i++)

{

edge[i][0]=edge[i][0]/(p*edge[i][0]+1);

edge[i][1]=edge[i][1]/(p*edge[i][0]+1);

edge[i][2]=edge[i][2]/(p*edge[i][0]+1);

}

draw_cube(edge);

break;

case 2:

printf("\n Enter Q :=");

scanf("%f",&q);

for(i=0;i<20;i++)

```

```

        {
            edge[i][1]=edge[i][1]/(edge[i][1]*q+1);
            edge[i][0]=edge[i][0]/(edge[i][1]*q+1);
            edge[i][2]=edge[i][2]/(edge[i][1]*q+1);
        }
        draw_cube(edge);
        break;
    case 3:
        printf("\n Enter R :=");
        scanf("%f",&r);
        for(i=0;i<20;i++)
        {
            edge[i][2]=edge[i][2]/(edge[i][2]*r+1);
            edge[i][0]=edge[i][0]/(edge[i][2]*r+1);
            edge[i][1]=edge[i][1]/(edge[i][2]*r+1);
        }
        draw_cube(edge);
        break;
    }
closegraph();
}

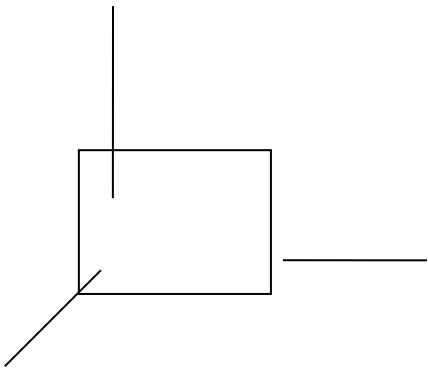
void main()    {
    int choice;

    double edge[20][3]= {
        100,0,0,100,100,0,0,100,0,0,100,100,0,0,100,0,0,0,
        100,0,0,100,0,100,100,75,100,75,100,100,100,100,75,
        100,100,0,100,100,75,100,75,100,75,100,100,0,100,100,
        0,100,0,0,0,0,0,0,100,100,0,100
    }
}

```

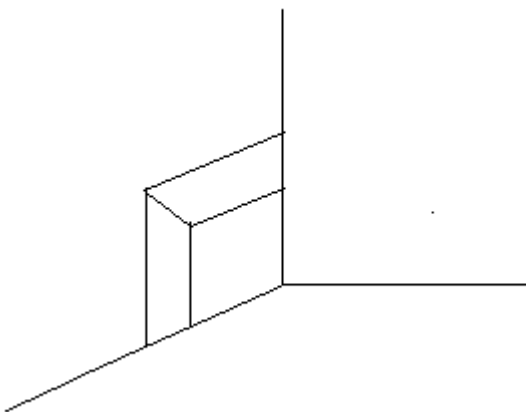
```
};  
clrscr();  
draw_cube(edge);  
perspect(edge);  
closegraph();  
}
```

OUTPUT :



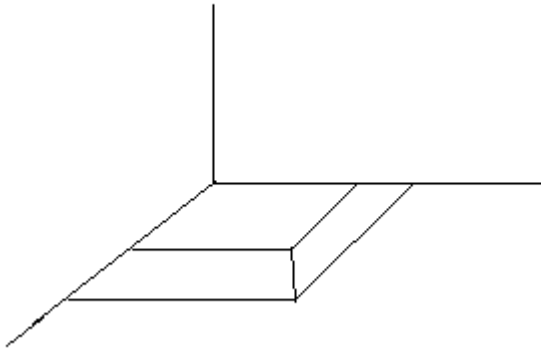
Projection about X-axis :

Enter the value of P : 45



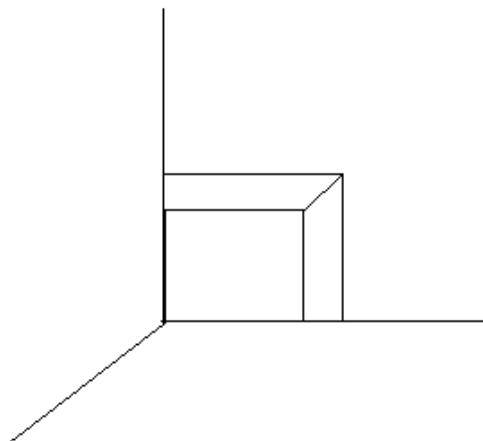
Projection about Y-axis :

Enter the value of Q : 45



Projection about Z-axis :

Enter the value of R : 45



RESULT :

Thus the projections on the three dimensional images was performed successfully and the output was verified.

Ex.No.: 9

ANIMATION USING FLASH

AIM

To create a motion tweening, shape tweening, guide layer and masking using Flash.

ALGORITHM

MOTION TWEENING

- Step 1: Create an object in the first layer of first key frame.
- Step 2: Create the last key frame and move the object to it.
- Step 3: Right click on the first key frame and select create motion tween.
- Step 4: Play the picture.

SHAPE TWEENING

- Step 1: Create an object in the first layer of first key frame.
- Step 2: Create the last key frame.
- Step 3: In first key frame's properties and select shape tween.
- Step 4: Change the shape of the object at the last frame.
- Step 5: Play the picture.

GUIDE LAYER

- Step 1: Create an object in the first layer of first key frame.
- Step 2: Create the last key frame and move the object to it.
- Step 3: Right click on the first key frame and select create motion tween.
- Step 4: Right click on the object's layer select add motion guide.
- Step 5: In motion guide draw your guide line.
- Step 6: Play the picture.

MASKING

Step 1: Create an object in the first layer of first key frame

Step 2: Add new layer and draw the shape of the view.

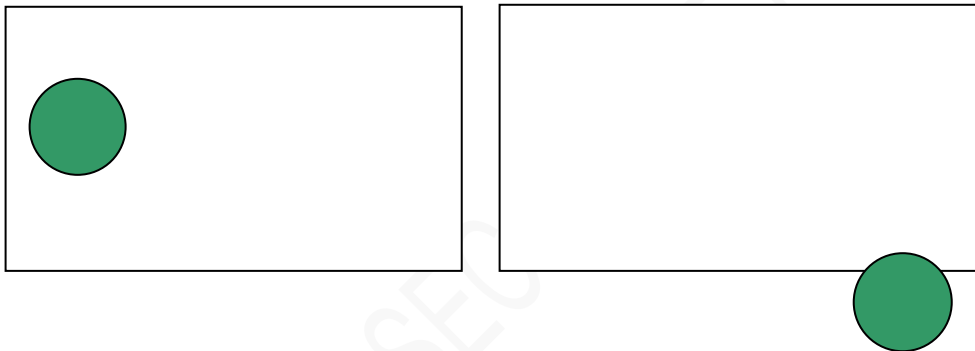
Step 3: Add the motion guide to the view tool.

Step 4: Right click on the second layer (view layer) and select mask.

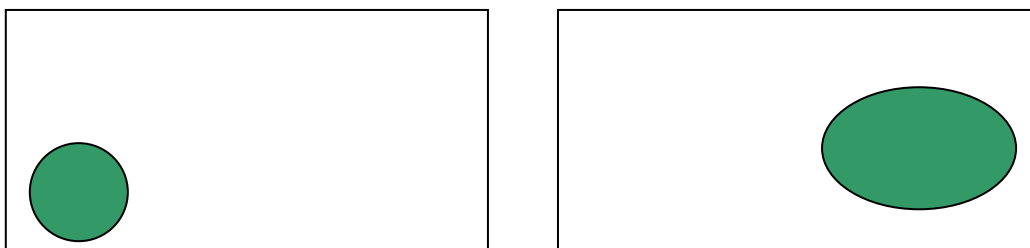
Step 5: Play the picture.

OUTPUT

MOTION TWEENING



SHAPE TWEENING



RESULT :

Thus the animation was done using flash and the output was verified.

Ex.No.:10

IMAGE EDITING

AIM:

To edit an image using an image editing software.

EDITING

- Open your project file and create a duplicate.
- Crop the image using crop tool.
- Change the image size using canvas technique.

BACK GROUND CHANGING

- Select the area to change the back ground using magic wand tool.
- Select the back ground image for your image.
- Move the shape of the back ground using marquee tool.
- Using selection tool, move the back ground.

CHANGING COLOUR

- Select the area using Lasso tool.
- Go to image tab and adjustments and select the Hue / saturation option.
- Change the colour using RGB mode.

CHANGING THE SIZE

- Select your image and go to edit and select transform.
- Change the size of the object.
- Press enter to set the size.

SAMPLE OUTPUT :



RESULT :

Thus the image was edited using the editing software and the output was verified.

PROGRAMME SPECIFIC OUTCOMES (PSOs)

1. **Professional Skills:** Design and analyze optimal artificial intelligence solutions to real-world problems in data analysis/ augmented reality/virtual reality/ robotics.
2. **Technical Skills:** Use modern AI/Data Science open-source software tools and hardware for product development.
3. **Entrepreneurship Skills:** Ability to lead a product development company/team.
4. **Research Skills:** Ability to use the acquired knowledge to identify real-world research problems.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

1. Graduates, within four years of graduation, should demonstrate peer-recognized expertise together with the ability to articulate that expertise and use it for contemporary problem-solving in the analysis, design, implementation and evaluation of artificial intelligence / data science systems.
2. Graduates, within four years of graduation, should demonstrate engagement in the engineering profession, locally and globally, by contributing to the ethical, competent, and creative practice of engineering or other professional careers.
3. Graduates, within four years of graduation, should demonstrate sustained learning and adapt to a constantly changing field through graduate work, professional development, and self-study.
4. Graduates, within four years of graduation, should demonstrate leadership and initiative to ethically advance professional and organizational goals, facilitate the achievements of others, and obtain substantive results.
5. Graduates, within four years of graduation, should demonstrate a commitment to teamwork while working with others of diverse cultural and interdisciplinary backgrounds.

PROGRAMME OUTCOMES (POS)

1. **Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to solve complex engineering problems.
2. **Problem Analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/Development of Solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for public health and safety, and cultural, societal, and environmental considerations.
4. **Conduct Investigations of Complex Problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern Tool Usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The Engineer and Society:** Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and Sustainability:** Understand the impact of professional engineering solutions in societal and environmental contexts and demonstrate the knowledge of and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and Team Work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project Management and Finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Lifelong Learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.