**EX NO: 1**  **TO IMPLEMENT LINE, CIRCLE AND ELLIPSE ATTRIBUTES**

**DATE:**

**Aim :**

To write a C program to draw the various attributes of line, circle and ellipse.

**Algorithm:**

**Step 1:** Start the program.

**Step 2:** Initialize the variables.

**Step 3:** Call the initgraph() function

**Step 4:** Set color for the output primitives.

**Step 5:** Using Outtextxy() display the chosen particular primitives.

**Step 6:** Include the various attributes of line, circle and ellipse.

**Step 7: C**lose the graph and run the program.

**Step 8: S**top the program

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<string.h>

void main()

{

char ch='y';

int gd=DETECT,gm,x1,y1,x2,y2,rad,sa,ea,xrad,yrad,i;

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

while(ch=='y')

{

cleardevice();

setbkcolor(9);

outtextxy(100,130,"Choose From The Following ");

outtextxy(100,150,"1. Line");

outtextxy(100,170,"2.Circle");

outtextxy(100,190,"3.Box");

outtextxy(100,210,"4.Arc");

outtextxy(100,230,"5.Ellipse");

outtextxy(100,250,"6.Rectangle");

outtextxy(100,270,"7.Exit");

ch=getch();

cleardevice();

switch(ch)

{

case '1':

line(100,200,300,400);

break;

case '2':

circle(200,200,100);

break;

case '3':

setfillstyle(5,4);

bar(100,300,200,100);

break;

case '4':

setfillstyle(5,4);

arc(200,200,100,300,100);

break;

case '5':

setfillstyle(5,4);

fillellipse(100,100,50,100);

break;

case '6':

settextstyle(DEFAULT\_FONT,0,2);

outtextxy(120,140,"AMSCOLLEGE");

line(100,100,100,300);

line(300,300,100,300);

line(100,100,300,100);

line(300,100,300,300);

break;

case '7':

closegraph();

return;

}

ch='y';

getch();

}

}

**OUTPUT:**

Choose from the following

1.Line

2. Circle

3.Box

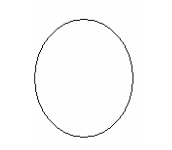
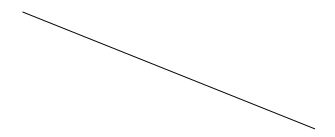
4.Arc

5.Ellipse

6.Rectangle

7.Exit

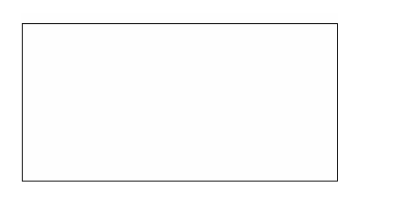
**LINE 2. CIRCLE**



**4. ARC 5. ELLIPSE**



**6. RECTANGLE**



**RESULT:**

Thus the above program has been executed and output is verified.

**EX NO: 2 A BRESENHAM’S LINE DRAWING ALGORITHM**

**DATE :**

**Aim:**

To write a C program to draw a line using Bresenham’s Algorithm.

**Algorithm:**

**Step 1:** Start the program.

**Step 2:** Input the two endpoints (x1,y1) and (x2,y2).

**Step 3:** Plot the pixel value (x1,y1) with a specified color.

**Step 4:** Calculate the value of dx and dy and find the starting value of decision parameter as **dp=2\*dy-dx.**

**Step 5:** Calculate the values of s1 and s2 depending on (x1,y1) and (x2,y2) values.

**Step 6:** If dp<0, the next point to plot is (x,y+s2) and dp=+2\*dy.

**Step 7:** Otherwise the next point to plot is (x+s1,y+s2) and

dp=dp+2\*dx-2\*dy.

**Step 8:** Repeat steps 5 and 6 dx times.

**Step 9:** Stop the program.

**PROGRAM :**

#include<stdio.h>

#include<math.h>

#include<conio.h>

#include<graphics.h>

void main()

{

int x1,x2,y1,y2;

int gd=DETECT,gm;

void linebres(int,int,int,int);

printf("Enter the two end points:");

scanf("%d%d%d%d",&x1,&x2,&y1,&y2);

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

cleardevice();

linebres(x1,y1,x2,y2);

getch();

line(x1,y1,x2,y2);

getch();

closegraph();

}

void linebres(int x1,int y1,int x2,int y2)

{

int dx=abs(x1-x2),dy=abs(y1-y2);

int p,x,y,i,xend,yend;

if(dx!=0)

{

p=2\*dy-dx;

if(x1>x2)

{

x=x2;

y=y2;

xend=x1;

}

else

{

x=x1;

y=y1;

xend=x2;

}

putpixel(x,y,2);

for(i=x;i<xend;i++)

{

x+=1;

if(p<0)

p+=2\*dy;

else

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

}

putpixel(x,y,2);

}

else

{

p=2\*dx-dy;

if(y1>y2)

{

x=x2;

y=y2;

yend=y2;

}

putpixel(x,y,2);

for(i=y;i<yend;i++)

{

y+=1;

if(p<0)

p+=2\*dx;

else

{

x+=1;

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

}

putpixel(x,y,2);

}

}

}

**OUTPUT :**

Bresenham line drawing algorithm

Enter the co-ordinates

150 100 250 300

**RESULT:**

Thus the above program has been executed and output is verified.

**EX NO: 2 B** **BRESENHAM’S CIRCLE DRAWING ALGORITHM**

**DATE:**

**Aim :**

To write a C program to draw a Circle using Bresenham’s Algorithm.

**Algorithm:**

**Step 1:** Start the program.

**Step 2:** Input radius r and the midpoint of the circle (x,y) and obtain the first point on the circumference for the circle as (0,r).

**Step 3:** Calculate the initial value of the decision parameter as p=1-r.

**Step 4:** At each position check the following conditions.

a) If p<0 then x=x+1 and p+=2\*x+1

b) Else y=y-1 and p+=2\*(x-y)+1.

**Step 5:** Determine symmetry points at the other seven octants.

**Step 6:** Move each calculated pixel position (x,y) onto the circular path centered on (xc,yc) and plot the coordinate value as x=x+xc and y=y+yc.

**Step 7:** Repeat step 3 until x<y.

**Step 8:** Stop the program.

**PROGRAM :**

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

void main()

{

int gd=DETECT,gm;

int x,y,r;

void cir(int,int,int);

printf("Enter the Mid points and Radious:");

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

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

cir(x,y,r);

getch();

closegraph();

}

void cir(int x1,int y1,int r)

{

int x=0,y=r,p=1-r;

void cliplot(int,int,int,int);

cliplot(x1,y1,x,y);

while(x<y)

{

x++;

if(p<0)

p+=2\*x+1;

else

{

y--;

p+=2\*(x-y)+1;

}

cliplot(x1,y1,x,y);

}

}

void cliplot(int xctr,int yctr,int x,int y)

{

putpixel(xctr +x,yctr +y,1);

putpixel(xctr -x,yctr +y,1);

putpixel(xctr +x,yctr -y,1);

putpixel(xctr -x,yctr -y,1);

putpixel(xctr +y,yctr +x,1);

putpixel(xctr -y,yctr +x,1);

putpixel(xctr +y,yctr -x,1);

putpixel(xctr -y,yctr -x,1);

getch();

}

**OUTPUT:**

Enter the Mid points and Radious:100 100 50

**RESULT:**

Thus the above program has been executed and output is verified.

**EX NO: 2 C BRESENHAM’S ELLIPSE DRAWING ALGORITHM**

**DATE:**

**Aim :**

To write a C program to draw a Ellipse using Bresenham’s Algorithm

**Algorithm:**

**Step 1:** Start the program.

**Step 2:** Input **rx , ry**and the center of the ellipse (**xc,yc**)and obtain the first point on the ellipse centered on the origin as (x0,y0) = (0,ry).

**Step 3:** Calculate the initial value of the decision parameter as **P10 = ry2 – rx2ry + ¼rx2**

**Step 4:** At each position k x in region 1, starting at k=0,perform the  following test. If p1k < 0 the next pt along the ellipse centered on (0,0) is **(xk+1,yk)** and  **p1k+1 = p1k + 2ry2xk+1 + ry2**

**Step 5:** Otherwise the next point along the ellipse is **(xk+1,yk-1)** and

**p1k+1 = p1k+2ry2 xk+1–2rx2yk+1+ry2** with

**2ry2xk+1 = 2ry2xk+2ry2 , 2rx2yk+1 = 2rx2yk – 2rx2**  and continue until **2ry2x ≥2rx2y**.

**Step 6:** Calculate the initial position of the decision parameter in region 2 as

**P20 = ry2  2 + ry2 ( y0 – 1 )2 - rx2 ry2**where (x0 ,y0) is the last position

**Step 7:** At each yk position in region 2, starting at k=0,perform the following test , if  p2k > 0 the next point along the ellipse centered on (0,0) is    **( xk , yk+1)** and

**p2k+1 = p2k – 2rx2yk+1 + ry2**.

**Step 8:** Otherwise the next point along the ellipse is ( xk + 1 ,yk -1) and

**P2k+1 = p2k – 2ry2 xk+1 – 2rx2yk+1 + rx2.**

**Step 9:** Using the same incremental values for x and y as in region 1 continue until y=0.

**Step 10:**For both regions determine symmetry points along the other three quadrants.

**Step 11:**Move each calculated pixel position (x,y) on to the elliptical path centered on

(xc  , yc  ) and plot the co-ordinates values x = x + xc , y = y + yc .

**Step 12:**Display the output points.

**Step 13:**Stop the program.

**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 : 10 30

Enter the X Center and Y Center: 300 150

**RESULT:**

Thus the above program has been executed and output is verified.

**EX NO: 3 A** **TWO – DIMENSIONAL TRANSFORMATION**

**DATE:**

**Aim :**

To write a C program to perform 2D transformations such as translation, rotation, scaling, reflection and shearing.

**Algorithm:**

**Step 1:** Start the program.

**Step 2:** Input the object coordinates

**Step 3:** For Translation

a) Enter the translation factors tx and ty.

b) Move the original coordinate position (x,y) to a new position (x1,y1).ie. x=x+x1, y=y+y1.

c) Display the object after translation

**Step 4:** For  Rotation

a) Enter the radian for rotation angle θ.

b) Rotate a point at position (x,y) through an angle θ about the origin x1=xcos θ - ysin θ , y1=ycos θ + xsin θ.

c) Display the object after rotation

**Step 5:** For  Scaling

a) Input the scaled factors sx and sy.

b) The transformed coordinates (x1,y1) , x1=x.sx and y1=y.sy.

c) Display the object after scaling

**Step 6:** Reflection can be performed about x axis and y axis.

a) Reflection about x axis : The transformed coordinates are x1=a and y1=-y.

b) Reflection about y axis : The transformed coordinates are  x1=x and  y1=y.

c) Display the object after reflection

**Step 7:** For  Shearing

a) Input the shearing factors shx and shy.

b) Shearing related to x axis : Transform coordinates     x1=x+shx\*y and y1=y.

c) Shearing related to y axis : Transform coordinates x1=x   and  y1=y+shy\*x.

d) Input the xref and yref values.

e) X axis shear related to the reference line y-yref is x1=x+shx(y-yref) and y1=y.

f) Y axis shear related to the reference line x=xref is x1=x

g) Display the object after shearing

**Step 8:** Stop the Program.

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

#include<math.h>

#include<stdlib.h>

void menu();

void input();

void output();

void translation();

void rotation();

void scaling();

void shearing();

void reflection();

int a[10][2],i,x,option,temp,angle,tx,ty,fx,fy,sh,k,n,axis,y;

float sx,sy;

void menu()

{

printf("menu\n");

printf("1.Translation\n");

printf("2.rotation\n");

printf("3.scaling\n");

printf("4.shearing\n");

printf("5.reflection\n");

printf("6.exit\n");

printf("enter the choice:");

scanf("%d",&option);

switch(option)

{

case  1:

input();

translation();

break;

case 2:

input();

rotation();

break;

case 3:

input();

scaling();

break;

case 4 :

input();

shearing();

break;

case 5:

input();

reflection();

break;

case 6:

exit(0);

break;

}

}

void input()

{

printf("enter the number of vertices:" );

scanf("%d",&n);

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

{

printf("enter the coordinates:");

scanf("%d%d%d%d",&a[i][0],&a[i][1],&a[i+1][0],&a[i+1][1]);

}

}

void output()

{

cleardevice();

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

{

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

}

}

void translation()

{

output();

printf("enter the tranformation vertex tx,ty:\n");

scanf("%d%d",&tx,&ty);

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

{

a[i][0]=a[i][0]+tx;

a[i][1]=a[i][1]+ty;

}

output();

delay(10);

menu();

}

void rotation()

{

output();

printf("enter the rotating angle:");

scanf("%d",&y);

printf("enter the pivot point:");

scanf("%d%d",&fx,&fy);

k=(y\*3.14)/180;

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

{

a[i][0]=fx+(a[i][0]-fx)\*cos(k)-(a[i][1]-fy)\*sin(k);

a[i][1]=fy+(a[i][0]-fx)\*sin(k)-(a[i][1]-fy)\*cos(k);

}

output();

delay(10);

menu();

}

void scaling()

{

output();

printf("enter the scaling factor\n");

scanf("%f%f",&sx,&sy);

printf("enter the fixed point:");

scanf("%d%d",&fx,&fy);

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

{

a[i][0]=a[i][0]\*sx+fy\*(1-sx);

a[i][1]=a[i][1]\*sy+fy\*(1-sy);

}

output();

delay(10);

menu();

}

void shearing()

{

output();

printf("enter the shear value:");

scanf("%d",&sh);

printf("enter the fixed point:");

scanf("%d%d",&fx,&fy);

printf("enter the axis for shearing if x-axis then 1 if y-axis the                  0:");

scanf("%d",&axis);

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

{

if(axis==1)

{

a[i][0]=a[i][0]+sh\*(a[i][1]-fy);

}

else

{

a[i][1]=a[i][1]+sh\*(a[i][0]-fx);

}

}

output();

delay(10);

menu();

}

void reflection()

{

output();

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

{

temp=a[i][0];

a[i][0]=a[i][1];

a[i][1]=temp;

}

output();

delay(10);

menu();

}

void main()

{

int gd=DETECT,gm;

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

menu();

getch();

}

**OUTPUT:**

**Menu**

Translation

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

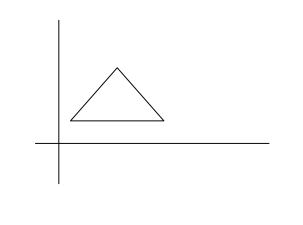
Enter the choice : 1

Enter the number of Vertices: 3

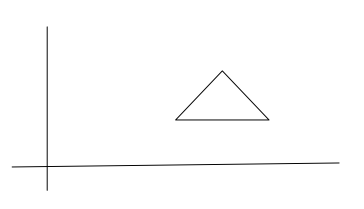
Enter the coordinates :        30        150      10        200

Enter the coordinates :        10        200      60        200

Enter the coordinates :        60        200      30        150



 Enter the translation vector Tx, Ty :       90        60



**ROTATION**

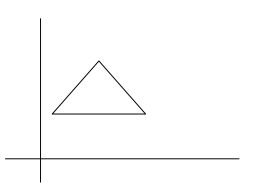
Enter the choice : 2

Enter the number of Vertices: 3

Enter the coordinates :        30        150      10        200

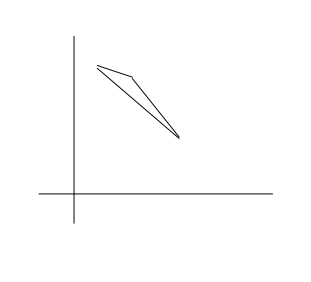
Enter the coordinates :        10        200      60        200

Enter the coordinates :        60        200      30        150



Enter the Rotating Angle :    90

Enter the Pivot Point            :          100      200



**SCALING**

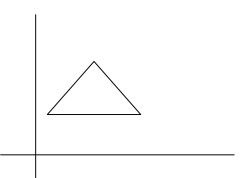
Enter the choice : 3

Enter the number of Vertices: 3

Enter the coordinates :        30        150      10        200

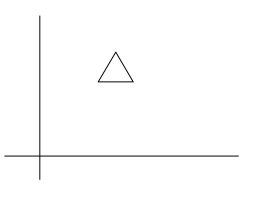
Enter the coordinates :        10        200      60        200

Enter the coordinates :        60        200      30        150



Enter the scaling Factor :     0.3       0.4

Enter the Fixed Point          :           100      200



**SHEARING**

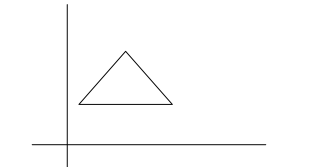
Enter the choice : 4

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

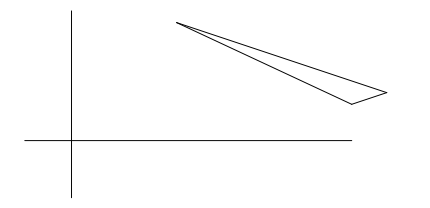


Enter the shear Value :        5

Enter the fixed point            : 50      100

Enter the Axis for shearing if  x-axis  then  1

if  y-axis  then  0



**REFLECTION**

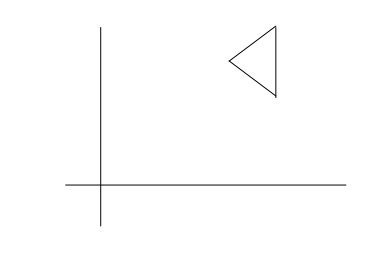
Enter the choice : 5

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 above program has been executed and output is verified.

**EX NO: 3 B THREE – DIMENSIONAL TRANSFORMATION**

**DATE:**

# AIM:

To implement the various 3D transformations like translation, scaling and rotation.

# ALGORITHM:

**To draw polygon**

1. Read the number of vertices.
2. Read the x&y coordinates of the vertices and store it in an array.
3. Draw the polygon using drawpoly().

**Translation**

It is applied to an object by repositioning it along a straight line path from one coordinate to another.

1. Read the translation distance tx & ty
2. Add tx&tyto the coordinates to move to the new position
3. Display the polygon

**Scaling**

It alters the size of an object, by multiplying the coordinate values of each vertex by scaling factors.

1. Read the scaling factors x
2. Multiply the scaling x with each coordinate vertex to alter the size.
3. Display the polygon.

**Rotation**

It is applied to an object by repositioning it a long a circular path in the xy plane.

1. Read the rotation fact or pivot point (a) and distance (xr, yr)from origin.
2. Polygon is rotated by displacing each vertex through the specified rotation angle(a).
3. Display the polygon.

# SOURCECODE:

#include<stdio.h> #include<conio.h> #include<graphics.h> #include<math.h>

void main()

{

int n,poly[20],poly1[20],poly2[10],sx,poly3[10];

inttx,ty,xr,yr,a,sy,poly4[10],poly5[10],poly6[10],poly7[10]; int i;

int gd=DETECT,gm; initgraph(&gd,&gm,"");

//STEPTO DRAW A POLYGAN

printf("\nEnterthenumberofverticesof thepolygan. \n");

scanf("%d",&n);

printf("\nEntertheX,Y co-ordinatesof eachvertex \n");

for(i=0;i<2\*n;i++)

{

scanf("%d",&poly[i]);

poly[i]=(poly[i])+40; poly[i+1]=(poly[i+1])+40;

poly1[i]=(poly[i]+50)+40;

poly1[i+1]=(poly[i+1]+50)+40;

}

poly[2\*n]=poly[0];

poly[2\*n+1]=poly[1];

poly1[2\*n]=poly1[0];

poly1[2\*n+1]=poly1[1];

outtextxy(70,70,"ORIGINALIMAGE");

drawpoly(n+1,poly);

drawpoly(n+1,poly1);

for(i=0;i<2\*n;i+=2)

line(poly[i],poly[i+1],poly1[i],poly1[i+1]);

getch();

cleardevice();

# //TRANSLATION

outtextxy(30,30,"ORIGINALIMAGE");

drawpoly(n+1,poly); drawpoly(n+1,poly1); for(i=0;i<2\*n;i+=2)

line(poly[i],poly[i+1],poly1[i],poly1[i+1]); getch();

outtextxy(10,10,"EntertheTranslationfactor:"); gotoxy(30,3);

scanf("%d%d",&tx,&ty);

for(i=0;i<2\*n;i+=2)

{

poly2[i]=poly[i]+tx; poly2[i+1]=poly[i+1]+ty; poly3[i]=poly1[i]+tx; poly3[i+1]=poly1[i+1]+ty;

}

poly2[2\*n]=poly2[0]; poly2[2\*n+1]=poly2[1]; poly3[2\*n]=poly3[0]; poly3[2\*n+1]=poly3[1]; drawpoly(n+1,poly2); drawpoly(n+1,poly3); for(i=0;i<2\*n;i+=2)

line(poly2[i],poly2[i+1],poly3[i],poly3[i+1]); getch();

cleardevice();

# //SCALING

printf("\n\n\toriginalimage!"); drawpoly(n+1,poly); drawpoly(n+1,poly1); gotoxy(300,100); for(i=0;i<2\*n;i+=2)

line(poly[i],poly[i+1],poly1[i],poly1[i+1]); getch();

outtextxy(10,10,"Enterthescalingfactor:"); gotoxy(30,3);

scanf("%d",&sx); for(i=0;i<2\*n;i+=2)

{

poly4[i]=poly[i]\*sx; poly4[i+1]=poly[i+1]\*sx; poly5[i]=poly1[i]\*sx; poly5[i+1]=poly1[i+1]\*sx;

}

poly4[2\*n]=poly4[0]; poly4[2\*n+1]=poly4[1]; poly5[2\*n]=poly5[0]; poly5[2\*n+1]=poly5[1]; drawpoly(n+1,poly4); drawpoly(n+1,poly5); for(i=0;i<2\*n;i+=2)

line(poly4[i],poly4[i+1],poly5[i],poly5[i+1]); getch();

cleardevice();

# //ROTATION

printf("\n\n\toriginalimage!"); drawpoly(n+1,poly); drawpoly(n+1,poly1); for(i=0;i<2\*n;i+=2)

line(poly[i],poly[i+1],poly1[i],poly1[i+1]); getch();

outtextxy(10,10,"Entertherotation factor");

gotoxy(30,3); scanf("%d%d%d",&xr,&yr,&a); for(i=0;i<2\*n;i+=2)

{

poly6[i]=xr+((poly[i]-xr)\*cos(a))-((poly[i+1]-yr)\*sin(a))+70;

poly6[i+1]=yr+((poly[i+1]-yr)\*cos(a))+((poly[i]-xr)\*sin(a))+70;

poly7[i]=xr+((poly[i]-xr)\*cos(a))-((poly1[i+1]-yr)\*sin(a))+70;

poly7[i+1]=yr+((poly[i+1]-yr)\*cos(a))+((poly1[i]-xr)\*sin(a))+70;

}

poly6[2\*n]=poly6[0]; poly6[2\*n+1]=poly6[1]; poly7[2\*n]=poly7[0]; poly7[2\*n+1]=poly7[1]; drawpoly(n+1,poly6); drawpoly(n+1,poly7); for(i=0;i<2\*n;i+=2)

line(poly6[i],poly6[i+1],poly7[i],poly7[i+1]); getch();

cleardevice();

}

**Input:**

Enter the number of vertices of the polygon: 4

Enter the (x,y) co-ordinates of the vertices: 100 100 150 100 150 150 100 150

**Output**

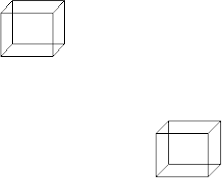
[](http://1.bp.blogspot.com/-hgpOx3uZqqc/UKiFP7Mb72I/AAAAAAAAAHY/Ln5iXlon1sA/s1600/origi.png)Original image

**Translation**

**Input:**

Enter the Translation Factor 200 200

**Output**

[](http://4.bp.blogspot.com/-t7j-Wzl_IJs/UKiFgDGcaYI/AAAAAAAAAHg/2-NIBnrFjC8/s1600/translation.png)Originalimage

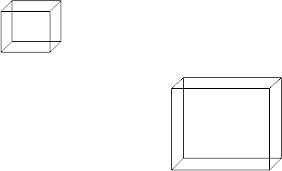
**Scaling**

**Input:**

Enter the scaling Factor 2

**Output**

Original image

[](http://2.bp.blogspot.com/-qTboE-uqJgU/UKiFtablT9I/AAAAAAAAAHo/-BrV2N7rF-s/s1600/scaling.png)

**Rotation**

**Input:**

Enter the Reflection Factor: 250 250 60

**Output**

Original image

[](http://2.bp.blogspot.com/--qDKjZdca2M/UKiF6S2r9TI/AAAAAAAAAHw/MpW23ngf9-M/s1600/rotation.png)

**RESULT:**

Thus generation of implementation of three dimensional objects **using** the above program has been executed and output is verified.

**EX NO: 4 TO PERFORM ANIMATION USING ANY ANIMATION SOFTWARE**

**DATE: -FRAME BY FRAME ANIMATIONS USING MULTIMEDIA**

**AUTHORINGTOOLS**

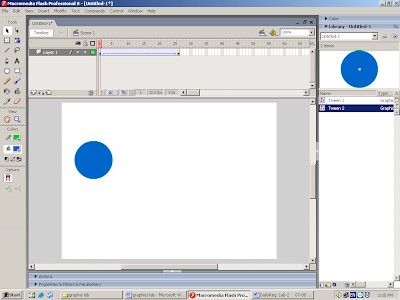
# MOTIONTWEENING

**AIM:**

To create motion tweening of an object.

# ALGORITHM:

1. Select the layer and place the ball by drawing with the help of tools.
2. Select the frames by pressing F6 or right click and select insert frame.
3. Click the layer and right click& create motion tween.
4. Move the ball in the screen to the required destination point.
5. Press ctrl+Enter
6. Enter to show it in full screen

[](http://4.bp.blogspot.com/-Dyt5H8Yc9a8/UKiHAHRUlVI/AAAAAAAAAIA/nmp2IKS2CkM/s1600/1.png)

# RESULT:

Thus the motion tweening of an object has been implemented and the output was verified.

# SHAPETWEENING–OBJECTANDTEXT

**AIM:**

To create shape tweening of an object and text**.**

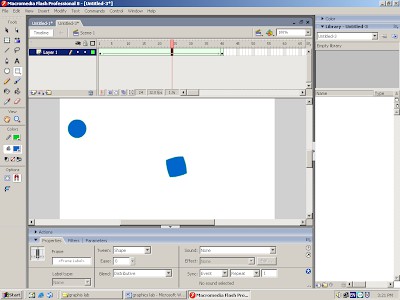
# ALGORITHM:

**Object:**

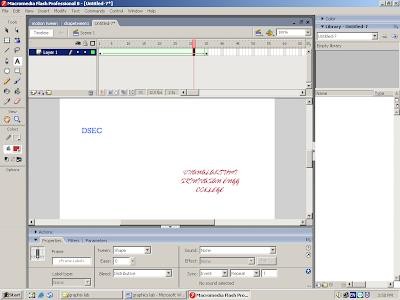
* 1. Select the layer and place a ball.
  2. Select the frame by pressing F6 (or) right click the mouse and select insert frame.
  3. In the same layer, create an other objects a rectangle.
  4. Click the layer and in properties change the tween to shape.
  5. Press ctrl+Enter to show it in full screen.

**Text:**

1. Select the text from the tool box and place it
2. Press ctrl twice a time.
3. Select the frame by pressing F6 or right click the mouse and select insert frame.
4. Select the layer and in properties change the tween as shape.
5. Press ctrl+enter to show it in full screen.

[](http://1.bp.blogspot.com/-BAhVcN7L7xk/UKiHRd0FW_I/AAAAAAAAAII/QyS8Rb2dTzY/s1600/2.png)

**Shape tweening of text:**

[](http://3.bp.blogspot.com/-24bm1SRyjX0/UKiHpfHEXAI/AAAAAAAAAIQ/lN8_fl9K3lo/s1600/3.png)

# RESULT:

Thus the shape tweening-text and object has been implemented and the output was verified.

**EX NO: 5** BASIC OPERATIONS ON IMAGE

**DATE:**

**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*.

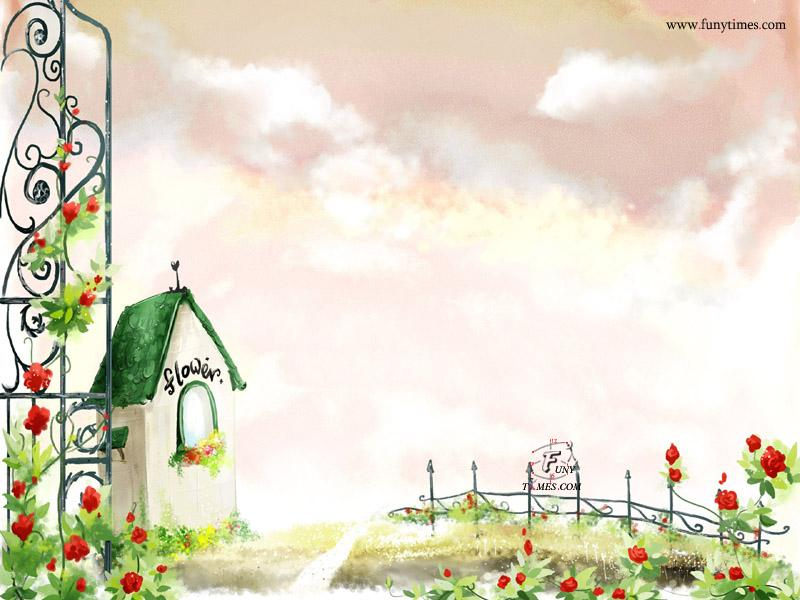
**CHANGING THE SIZE**

 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.

**EX NO: 6** PRESENTATION FOR PRODUCT USING TECHNIQUES

(GUIDE LAYER, MASKING, ONION SKIN)

**DATE:**

# GUIDELAYER

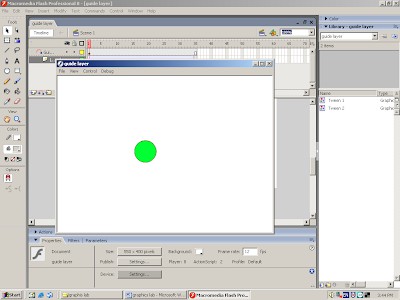
**AIM:**

To create an animation using guide Layer.

# ALGORITHM:

1. Select the layer and draw an object.
2. Select the frame bypressingF6.
3. Select the guide layer in the layer options.
4. With the help of pencil tool, draw the path.
5. Move the object over the path.
6. Select the first layer and select create motion tween by right clicking the mouse.
7. Press Ctrl+Ente rto show it in full screen.

# OUTPUT:

[](http://1.bp.blogspot.com/-TLVUEiE6fZ4/UKiH8lMvEQI/AAAAAAAAAIY/hA9v_6Yhoos/s1600/4.png)

**RESULT:**

Thus the guide layer has been implemented and the output was verified.

**MASKNG**

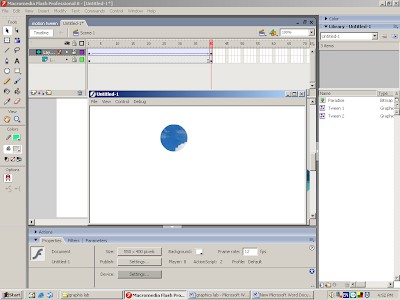
AIM:

To implement masking concept

# ALGORITHM:

1. Select the text.
2. Select the frame by pressing F6.
3. Insert an other new layer and select create motion tween.
4. Right click the frame and select create motion tween.
5. Move the ball over select the mark
6. Press Ctrl+Enter to show it in full screen.

# OUTPUT:

[](http://3.bp.blogspot.com/-cGDLyy05JT8/UKiIOT0fSjI/AAAAAAAAAIg/k8bJOvICUI0/s1600/5.png)

**RESULT:**

Thus the masking has been implemented and the output was verified.

**ONION SKIN**

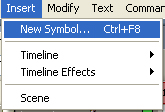
AIM:

To implement masking concept

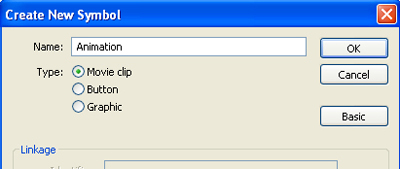
# ALGORITHM:

**1.** Create a new document in Macromedia Flash 8.

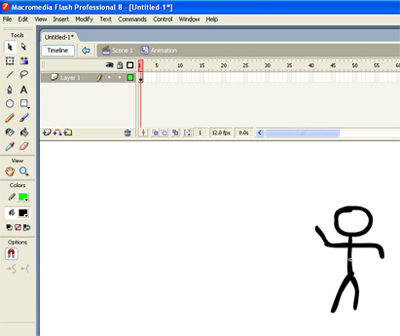
**2.** Go to "Insert - New symbol"(Ctrl-F8) to create a new symbol.



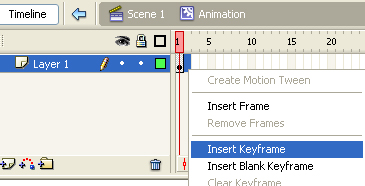
**3.** In the new pop up window type "Animation" in the name field and make sure to select "Movie clip" as type.



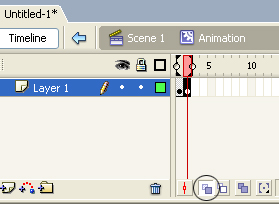
**4.** In the first frame of layer 1 just draw a random drawing. This is what we're going to animate.



**5.** Insert a new frame in Layer 1 (F6).



**6.** Turn on onion skin.



**7.** If you try to change the drawing you can see the traces of the last frame to help you animate.



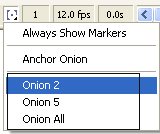
**8.** You can drag this marker to select how many frames you want to see in onion skin mode.

image 7

You can tweak the onion skin settings at the bottom of the timeline tab. The first icon toggles onion skin on or off. The second will set onion skin outlines only. If you set the third one you will be able to edit all frames you have selected at once.

image 8

The fourth option down there gives a few choices when selected. What this does is just to set the number of frames to "onion skin" to either the last 2, 5 or all.



That's basically onion skin in Flash 8.

**EX NO: 7** VARIOUS FEATURES OF AN IMAGE EDITING TOOL

**DATE:**

|  |
| --- |
|  |

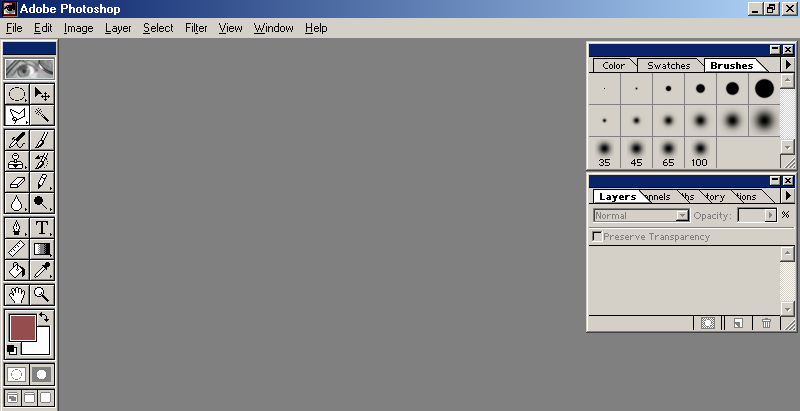
**How to open Photoshop**

Step 1: Click on START Step 2: Select Programs

Step 3: Select Adobe

Step 4: Select PhotoShop 5.x Step 5: Click on Photoshop 5.x

After finishing doing this you will get a screen like this (below):



The above screen provides you the tool bar and few other windows, but not the work area.

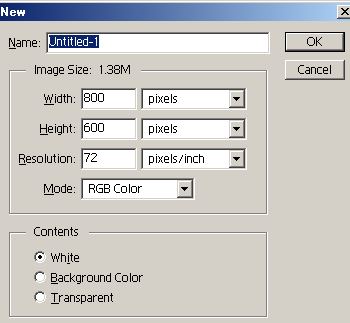
##### For work area;

Step 1: Click on FILE Step 2: Click on NEW

The next screen will appear, which will require few inputs from you, like

1. The height and width of the work area in pixel / inches / cm / points / picas / columns,
2. The resolution for the file
3. The image color mode Bitmap / Grayscale / RGB / CMYK / lab color
4. The last option for the type of background required white / background color / transparent.
5. Give a file name under “Name”
6. Click on OK

The Screen looks like as given below.



The normal size for work area is 640 pixels (width) x 480 pixels (height) with the resolution 72 pixels/inch.

Now as you have got the work area, you can start working on it.

With the help of the Photoshop tool bar you can create, edit, retouch, manipulate your drawing or imported image.

# TO SAVE YOUR WORK

Step 1: Click on File

Step 2: Click on Save / Save as

Step 3: Select the directory under which the file you want to save

Step 4: Provide a name (if not provided at the time of creating a new file) Step 5: Click on Save

###### Note: Photoshop saves the file in .psd file format

For saving your work either in .jpg or in .gif for your webpage do the following;

**For JPEG file format:**

Step 1: Click on File

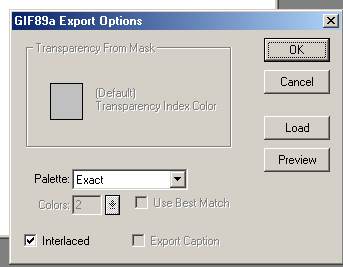
Step 2: Click on Save a copy…

Step 3: Select the directory under which the file you want to save Step 4: Provide a name

Step 5: Select JPEG from under the “Save as” option. Step 6: Click on Save

##### For GIF file format:

Step 1: Click on File Step 2: Click on Export

Step 3: Click on GIF89a export You will get the following screen

Step 4: without changing anything (as above) Click on OK

Step 5: Select the directory under which the file you want to save