M.A.M COLLEGE OF ENGINEERING AND TECHNOLOGY

SIRUGANUR, TRICHY

GRAPHICS AND MULTIMEDIA LAB MANUAL

IV B.Tech(IT) / VII SEMESTER

Prepared by,

P.MAthumitha

Curriculam and Syllabus
CS1360 – GRAPHICS AND MULTIMEDIA LABORATORY

S.No.	Subject Code	Subject
	Theory	
1	**IT1401**	Web Technology
2	**IT1402**	Middleware Technologies
3	**IT1403**	Mobile Computing
4	**CS1354**	Graphics and Multimedia
5	**CS1017**	TCP / IP DESIGN AND IMPLEMENTATION
6	**CS1021**	SOFTWARE PROJECT MANAGEMENT
	Practical	
7	**IT1404**	Middleware Technologies Laboratory
8	**CS1360**	**Graphics and Multimedia Laboratory**
9	**CS1403**	Software Development Laboratory

LIST OF EXPERIMENTS

1. To implement Bresenham's algorithms for line, circle and ellipse drawing

2. To perform 2D Transformations such as translation, rotation, scaling, reflection and sharing.

3. To implement Cohen–Sutherland 2D clipping and window–viewport mapping

4. To perform 3D Transformations such as translation, rotation and scaling.

5. To visualize projections of 3D images and Hidden Surface Elimination.

6. To convert between color models.

7. To implement text compression algorithm

8. To implement image compression algorithm

9. To perform animation using any Animation software

10. To perform basic operations on image using any image editing software

EX NO:1a **BRESENHAM'S LINE DRAWING ALGORITHM**

AIM:

To write a program for implementing bresenham's line drawing algorithm.

ALGORITHM:

- Declare the variables in the main function.
- Pass the parameter for the functions initgraph
- Get the values for the line to be drawn
- Compute the absolute value for the x & y co-ordinates.
- Compute p=2*dy-dx and x=xa and y=ya.
- Using for loop,pass the parameter for putpixel function.
- If p>0, Compute p=p+2*dy-2*dx and increament x value.
- Call the function close graph().

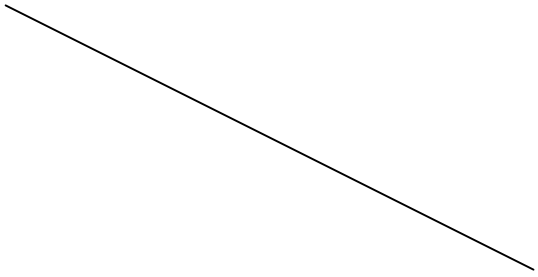
PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
Void main()
{
Int gd=DETECT,gm;
Int dx,dy,p,end;
float x1,x2,y1,y2,x,y;
initgraph(&gd,&gm,"c:\tc\bgi");
printf("\n enter the value of x1:");
scanf("%f",&x1);
printf("\n enter the value of x2:");
scanf("%f",&x2);
printf("enter the value for y1 and y2");
scanf("%f%f",&y1,&y2);
dx=abs(x1-x2);
dy=abs(y1-y2);
p=2*dy-dx;
if(x1>x2)
{
X=x2;
Y=y2;
end=x1;
```

```
}
Else{
X=x1;
Y=y1;
end=x2 ;
}
Putpixel(x,y,10);
While(x<end)
{
X=x+1;
If(p<10)
P=p+2*dy;
Else{
Y=y+1;
P=p+2*(dy-dx);
}
Putpixel(x,y,10);
}
Getch();
Closegraph();
}
```

OUTPUT:

```
Enter the value of x1:100
Enter the value of x2:200
Enter the value of y1& y2 : 300   400
```



RESULT:

The program to implement bresenham's line drawing algorithm is created and verified.

EX NO:1b **MIDPOINT CIRCLE ALGORITM**

AIM:

To draw a circle using midpoint circle algorithm.

ALGORITHM:

- Declare the header files and functions.
- We declare the functions initgraph(),
- Enter the value for circle to be drawn.
- In putpixel function enter the x and y value and constant value.
- Initialize xa=0 and ya=90 as the circle coordinates.
- In plotcircle() function, get the values for the x and y values for the circles.

PROGRAM:

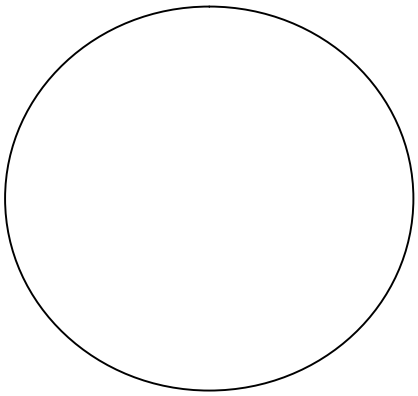
```
#include<stdio.h>

#include<conio.h>
#include<graphics.h>
#include<math.h>
Void circlept(int xc,int yc,int x,int y)
{
Putpixel(xc+x,yc+y,6);
Putpixel(xc-x,yc+y,6);
Putpixel(xc+x,yc-y,6);
Putpixel(xc-x,yc-y,6);
Putpixel(xc+y,yc+x,6);
Putpixel(xc-y,yc+x,6);
Putpixel(xc+y,yc-x,6);
Putpixel(xc-y,yc-x,6);
}
void main()
{
Int gd=DETECT,gm;
Int x,y,xc,yc,r,p;
Initgraph(&gd,&gm,"c:\tc\bgi");
Printf("enter the midpoint of a circle");
Scanf("%d%d",&xc,&yc);
printf("\n Enter the radius");
scanf("%d",&r);
P=3-2*r;
```

```
    x=0;
    y=r;
    circlept(xc,yc,x,y);
    while(x<y)
    {
    if(p<0)
    {
    p=p+4*x+6;
    x++;
    }
    else
    {
    p=p+4*x-4*y+10;
    x++;
    y--;
    }
    circlept(xc,yc,x,y);
    }
getch();
}
```

OUTPUT:

```
Enter the midpoint of circle :300      300
Enter the radius :100
```



RESULT:

Thus the program to implement bresenhams circle drawing algorithm is created and the output is verified.

EX NO: 1c

MIDPOINT ELLIPSE ALGORITHM

AIM:

To develop the midpoint ellipse algorithm using turboc3.

ALGORITHM:

- Declare all the variables that is needed for the program.
- Give the statement that is to be display in ouput stream.
- Get the input through the scanf statements.
- Give the condtions like while and do.
- Check the if condition & also else part.

PROGRAM:

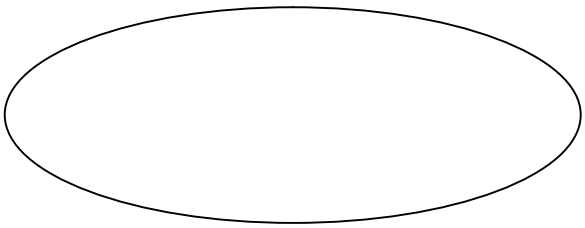
```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
#include<stdio.h>
void main()
{
int gd=DETECT,gm;
long d1,d2;
int x,y,i;
long rx,ry,rxsq,rysq,trxsq,trysq,dx,dy;
printf("\n Enter x and y radius:");
scanf("%ld%ld",&rx,&ry);
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"E:\\tc\\bgi");
rxsq=rx*rx;
rysq=ry*ry;
trxsq=2*rxsq;
trysq=2*rysq;
x=0;
y=ry;
d1=rysq-rxsq*ry+(0.25*rxsq);
dx=trysq*x;
dy=trxsq*y;
```

```
do
{
setcolor(25);
putpixel(200+x,200+y,12);
putpixel(200-x,200-y,12);
putpixel(200+x,200-y,10);
putpixel(200-x,200+y,10);
if(d1<10)
{
x=x+1;
y=y;
dx=dx+trysq;
d1=d1+dx+rysq;
}
else
{
x=x+1;
y=y-1;
dx=dx+trysq;
dy=dy-trxsq;
d1=d1+dx-dy+rysq;
}
}
while(dx<dy);
d2=rysq*(x+0.5)*(x+0.5)+rxsq*(y-1)*(y-1)-rxsq*rysq;
do
{
putpixel(200+x,200+y,12);
putpixel(200-x,200-y,12);
putpixel(200+x,200-y,10);
putpixel(200-x,200+y,10);
if(d2>0)
{
x=x;
y=y-1;
dy=dy-trxsq;
d2=d2-dy+rxsq;
}
else
{
x=x+1;
y=y-1;
dy=dy-trxsq;
dx=dx+trysq;
d2=d2+dx-dy+rxsq;
}
```

```
}
while(y>0);
getch();
closegraph();
}
```

OUTPUT:

Enter the x and y radius : 40 20



RESULT:

The above program have been created and the verified.

Ex no:2 **IMPLEMENTATION OF 2D TRANSFORMATION SUCH AS TRANSLATION,ROTATION,SCALING,REFLECTION & SHEARING**

AIM:

To write a c program for implementation of 2D transformation.

ALGORITHM:

- Get the Number of polygon & get (x,y) coordinates.
- Display the original image
- Get the translation factor & display translated image.
- Get the shearing factor & display the image
- Get the rotating factor & display the image
- Display the reflected image
- Get the shear factor & display the shear image.

PROGRAM:

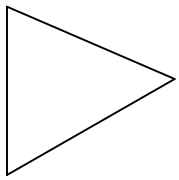
```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
#include<stdlib.h>
void main()
{
int i,poly1[10],poly2[10],poly3[10],poly4[10],tx,ty,sx;
int n,xr,yr,a;
int poly[10],poly5[10],sy;
int gd=DETECT,gm;
initgraph(&gd,&gm,"E:\\tc\\bgi");
printf("\Enter the no of verticles for a polygon");
scanf("%d",&n);
printf("\n Enter the (x,y)co-ordinates of verticles");
for(i=0;i<2*n;i++)
{
scanf("%d",&poly[i]);
}
poly[2*n]=poly[0];
poly[2*n+1]=poly[1];
outtextxy(70,70,"orginal image");
drawpoly(n+1,poly);
getch();
cleardevice();
outtextxy(30,30,"original image");
drawpoly(n+1,poly);
```

```
outtextxy(10,10,"Enter translation factor");
gotoxy(30,3);
scanf("%d%d",&tx,&ty);
for(i=0;i<2*n;i+=2)
{
poly1[i]=poly[i]+tx;
poly1[i+1]=poly[i+1]+ty;
}
poly1[2*n]=poly1[0];
poly1[2*n+1]=poly1[1];
drawpoly(n+1,poly1);
getch();
cleardevice();
outtextxy(30,30,"original image");
drawpoly(n+1,poly);
outtextxy(10,10,"Enter the scaling Factor");
gotoxy(30,3);
scanf("%d",&sx);
for(i=0;i<2*n;i+=2)
{
poly2[i]=poly[i]*sx;
poly2[i+1]=poly[i+1]*sx;
}
poly2[2*n]=poly2[0];
poly2[2*n+1]=poly2[1];
drawpoly(n+1,poly2);
getch();
cleardevice();
outtextxy(30,30,"original image");
drawpoly(n+1,poly);
outtextxy(10,10,"Enter The Rotation Factor");
gotoxy(30,3);
scanf("%d%d%d",&xr,&yr,&a);
for(i=0;i<2*n;i+=2);
{
poly3[i]=xr+((poly[i]-xr)*cos(a))-((poly[i+1]-yr)*sin(a));
poly3[i+1]=yr+((poly[i+1]-yr)*cos(a))-((poly[i]-xr)*sin(a));
}
poly3[2*n]=poly3[0];
poly3[2*n+1]=poly3[1];
drawpoly(n+1,poly3);
getch();
cleardevice();
outtextxy(30,30,"original image");
drawpoly(n+1,poly);
outtextxy(10,10,"reflected image");
```

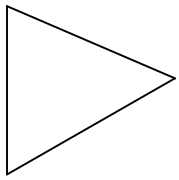
```
gotoxy(30,3);
for(i=0;i<2*n;i+=2)
{
poly4[i]=640-poly[i];
poly4[i+1]=poly[i+1];
}
poly4[2*n]=poly4[0];
poly4[2*n+1]=poly4[1];
drawpoly(n+1,poly4);
getch();
cleardevice();
outtextxy(30,30,"original image");
drawpoly(n+1,poly);
outtextxy(10,10,"Enter the shear factor");
gotoxy(30,3);
scanf("%d",&sy);
for(i=0;i<2*n;i+=2)
{
poly5[i]=poly[i]+sy*poly[i+1];
poly5[i+1]=poly[i];
}
poly5[2*n]=poly5[0];
poly5[2*n+1]=poly5[1];
drawpoly(n+1,poly5);
getch();
cleardevice();
}
```

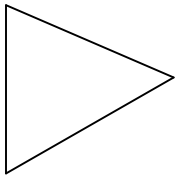
OUTPUT:

```
Enter the no. of vertices of polygon : 3
Enter  (x,y) coordinates of vertex
100
100
100
200
150
150
```



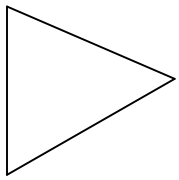
Enter translation factor
100
100
Original image

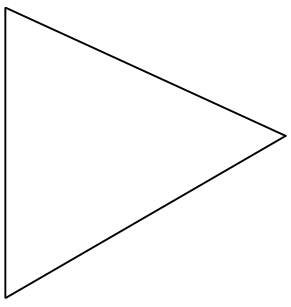




Enter the scaling factor: 2

Original image





Enter the rotation factor

250

250

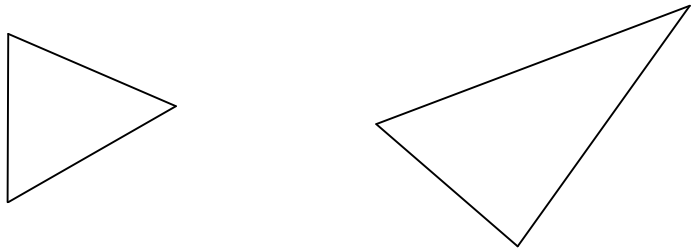
90

Original image

Reflected Image:

Original Image

Enter the Shear factor : 2



RESULT:

Thus the above program implemented successfully and the output was verified.

Ex no:3a **IMPLEMENTATION OF 2D CLIPPING**

AIM:

To write a program to implement 2D clipping.

ALGORITHM:

- Enter the window & line coordinates .
- With their values ,the window & corresponding line is drawn.
- If the coordinates of the line is not within the window it is clipped.
- Thus the clipped line within the window is drawn as a result of clipping.

PROGRAM:

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
int cliptest(float p,float q,float *u1,float *u2)
{
```

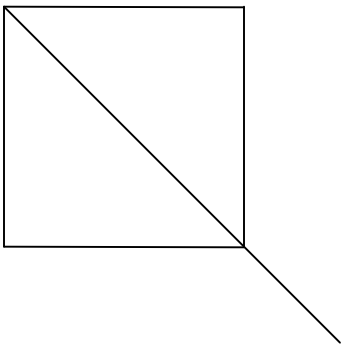
```
float r;
int g;
g=1;
if(p<0.0)
{
r=q/p;
if(r>*u2)
g=0;
else
if(r>*u1)
*u1=r;
}
else
if(p>0.0)
{
r=q/p;
if(r<*u1)
g=0;
else
if(r<*u2)
*u2=r;
}
else
if(q<0.0)
g=0;
return(g);
}
void main()
{
int wx1,wx2,wy1,wy2,lx1,lx2,ly1,ly2;
int d=DETECT,m;
float dx,dy,u1,u2;
initgraph(&d,&m,"E:\\tc\\bgi");
printf("\n enter the window coordinates");
scanf("%d%d%d%d",&wx1,&wy1,&wx2,&wy2);
printf("\n enter the line coordinates");
scanf("%d%d%d%d",&lx1,&ly1,&lx2,&ly2);
printf("\n before clipping");
rectangle(wx1,wy1,wx2,wy2);
line(lx1,ly1,lx2,ly2);
getch();
u1=0.0;
u2=1.0;
dx=lx2-lx1;
if(cliptest(-dx,lx1-wx1,&u1,&u2))
if(cliptest(dx,wx2-lx1,&u1,&u2))
```

```
{
dy=ly2-ly1;
if(cliptest(-dy,ly1-wy1,&u1,&u2))
if(cliptest(dy,wy2-ly1,&u1,&u2))
{
if(u1<1.0)
lx2=lx1+u2*dx;
ly2=ly1+u2*dy;
}
if(u1>0.0)
{
lx1=lx1+u1*dx;
ly1=ly1+u1*dy;
}
cleardevice();
printf("\n after clipping");
rectangle(wx1,wy1,wx2,wy2);
line(lx1,ly1,lx2,ly2);
}
getch();
}
```

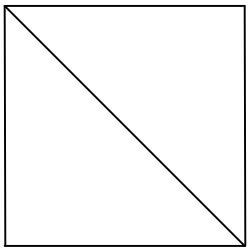
OUPUT:

```
Enter the window coordinates:
100
100
200
200
Enter the line coordinates:
100
100
300
300

Before Clipping
```



After Clipping:



RESULT:

Thus the program to implement 2D clipping have been successfully executed and output is verified.

Ex No: 3b **WINDOW VIEW PORT**

AIM

To develop a program for the window viewport using turbo c

ALGORITHM

- Declare the values that is used in the program
- Give the function that is required for the program
- Give the statements to the display and the maximum value for X and Y
- Check the condition for the S value & print the line
- After completing the process, close the graph

PROGRAM

```
#include<stdio.h>

#include<conio.h>

#include<graphics.h>

void main()

{

int gd=DETECT,gm,s,i;
float xw1,xw2,xv1,xv2,yw1,yw2;
float yv1,yv2,x[20],y[20],xv[20],yv[20],sx,sy;
initgraph(&gd,&gm,"e:\\tc\\bgi");
cleardevice();
printf("\n Enter the number of sides:");
scanf("%d",&s);
printf("\n Enter the coordinates:");
for(i=0;i<s;i++)
scanf("%f%f",&x[i],&y[i]);
printf("\nEnter the view port values:");
scanf("%f%f%f%f",&xv1,&yv1,&xv2,&yv2);
xw1=0;
yw1=0;
xw2=getmaxx();
yw2=getmaxy();
cleardevice();
rectangle(xw1,yw1,xw2,yw2);
for(i=0;i<(s-1);i++)
line(x[i],y[i],x[i+1],y[i+1]);
line(x[i],y[i],x[0],y[0]);
outtextxy(xw1+10,yw1+10,"window port");
```

```
sx=(xv2-xv1)/(xw2-xw1);
sy=(yv2-yv1)/(yw2-yw1);
for(i=0;i<s;i++)
{
xv[i]=xv1+(x[i]-xw1)*sx;
yv[i]=yv1+(y[i]-yw1)*sy;
}
outtextxy(xw1+10,yw2-10,"press any key");
getch();
cleardevice();
rectangle(xv1,yv1,xv,yv2);
outtextxy(xv1+10,yv1+10,"view");
for(i=0;i<(s-1);i++)
line(xv[i],yv[i],xv[i+1],yv[i+1]);
line(xv[i],yv[i],xv[0],yv[0]);
getch();
closegraph();
}
```

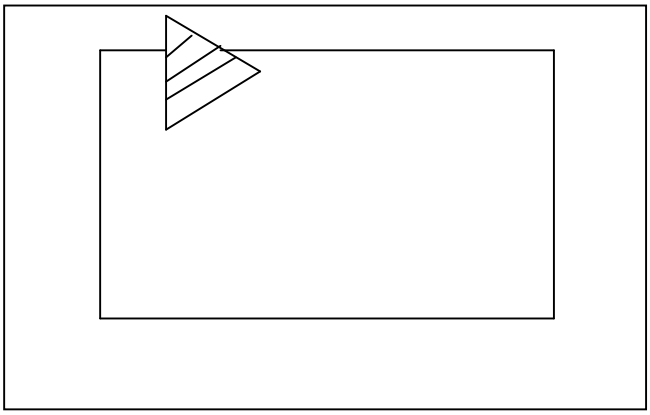
OUTPUT:

Enter the window coordinates : 10 10 100 100

Enter the view port coordinates : 20 40 80 80

Enter the number of vertices : 3

Enter the 6 coordinates : 20 30 40 60 40 80



RESULT Thus the program to create the window viewport using turbo c was created, executed and the output verified.

Exno:4 **IMPLEMENTATION OF 3D TRANSFORMATION SUCH AS TRANSLATION,ROTATION & SCALING.**

AIM:

To implement the 3D-tranformation like translation,rotation & scaling usng ‘c’ program.

ALGORITHM:

- First get the no. of vertices for the polygon & get the array test of polygon.
- Calculate the vertices of polygon & assign it to the coordinates vertices of polygon.
- Using draw polygon function obtain the original image.
- Get the translation factor of the polygon.
- Calculate factor and add the translation factor.

PROGRAM:

```
#include<math.h>
#include<iostream.h>
#include<dos.h>
#include<process.h>
#include<conio.h>
#include<graphics.h>
int gd=DETECT,gm;
int x1,y1,x2,y2;
void draw_cube(double edge[20][3])
{
initgraph(&gd,&gm,"E:\\tc\\bgi");
int i;
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 scale(double edge[20][3])
{
```

```
double a,b,c;
int i;
cout<<"enter the scaling factor";
cin>>a>>b>>c;
initgraph(&gd,&gm,"E:\\tc\\bgi");
clearviewport();
for(i=0;i<20;i++)
{
edge[i][0]=edge[i][0]*a;
edge[i][1]=edge[i][1]*b;
edge[i][2]=edge[i][2]*c;
}
draw_cube(edge);
closegraph();
}
void translate(double edge[20][3])
{
int a,b,c;
int i;
cout<<"enter the translate factor";
cin>>a>>b>>c;
initgraph(&gd,&gm,"E:\\tc\\bgi");
clearviewport();
for(i=0;i<20;i++)
{
edge[i][0]+=3;
}
draw_cube(edge);
closegraph();
}
void rotate(double edge[20][3])
{
int ch,i;
double temp,temp1,theta;
clrscr();
cout<<"\n Rotation about:";
cout<<"1.->x-axis"<<"2:->y-axis"<<"3:->z-axis";
cout<<"\nEnter ur choice";
cin>>ch;
switch(ch)
{
case 1:
cout<<"\nEnter the angles:";
cin>>theta;
theta=(theta*3.14)/180;
for(i=0;i<20;i++)
```

```
{
edge[i][0]=edge[i][0];
temp=edge[i][1];
temp1=edge[i][2];
edge[i][1]=temp*cos(theta)-temp1*sin(theta);
edge[i][2]=temp*sin(theta)+temp1*cos(theta);
}
draw_cube(edge);
break;
case 2:
cout<<"\nEnter the angles";
cin>>theta;
theta=(theta*3.14)/180;
for(i=0;i<20;i++)
{
edge[i][1]=edge[i][1];
temp=edge[i][0];
temp1=edge[i][2];
edge[i][0]=temp*cos(theta)-temp1*sin(theta);
edge[i][2]=temp*sin(theta)+temp1*cos(theta);
}
draw_cube(edge);
break;
case3:
cout<<"\n Enter the angle:";
cin>>theta;
theta=(theta*3.14)/180;
for(i=0;i<20;i++)
{
edge[i][2]=edge[i][2];
temp=edge[i][0];
temp1=edge[i][1];
edge[i][1]=temp*cos(theta)-temp1*sin(theta);
edge[i][2]=temp*sin(theta)+temp1*cos(theta);
}
draw_cube(edge);
break;
}
}
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};
while(1)
{
clrscr();
cout<<"\n3D Transformation :";
cout<<"\n1:draw table\n2:transformation\n3:rotation";
cout<<"\n4:scaling\n5:exit";
cout<<"\nEnter your choice:";
cin>>choice;
switch(choice)
{
case 1:
draw_cube(edge);
break;
case 2:
translate(edge);
break;
case 3:
rotate(edge);
break;
case 4:
scale(edge);
break;
case 5:
exit(0);
break;
default:
cout<<"Enter your valid key";
getch();
break;
}
```

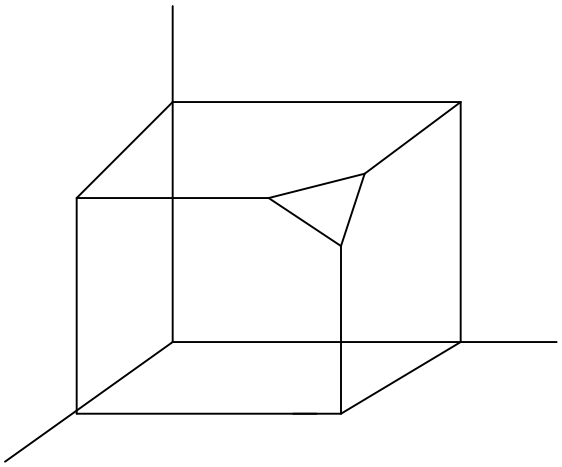
```
closegraph();
}
}
```

OUTPUT:

3D Transformation

1.Draw cube
2.Tranformation
3.Rotation
4.Scaling
5.Exit

Enter your choice : 1



3D Transformation

1.Draw cube
2.Tranformation
3.Rotation
4.Scaling
5.Exit

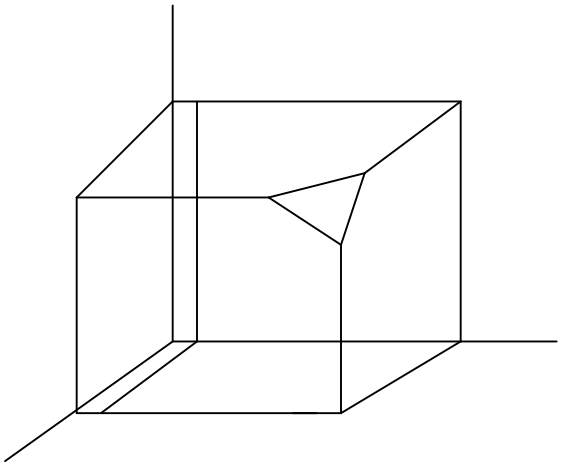
Enter your choice : 2

Enter the Translation factor:

100

100

100



3D Transformation

1.Draw cube

2.Tranformation

3.Rotation

4.Scaling

5.Exit

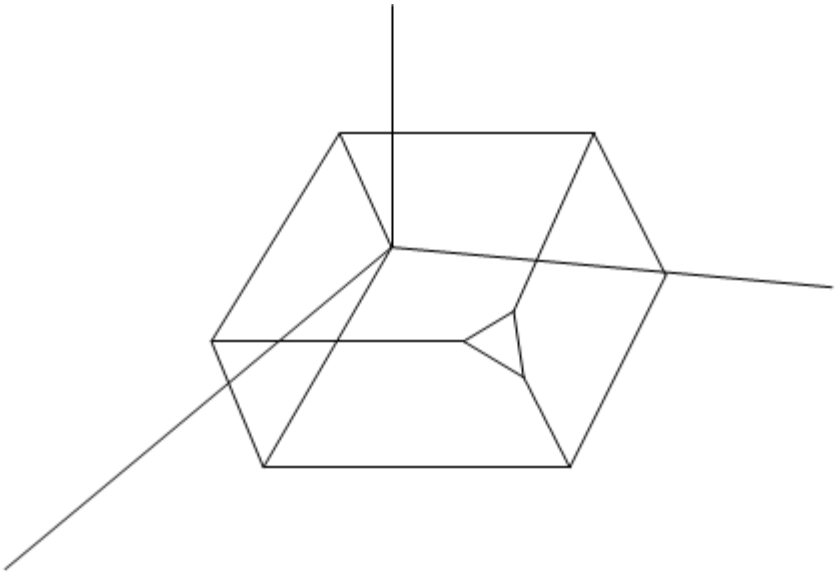
Enter your choice : 3

Rotation about..

1.X-axis 2.Y-axis 3.Z-axis

Enter your choice :1

Enter the angle : 30



3D Transformation

1.Draw cube

2.Tranformation

3.Rotation

4.Scaling

5.Exit

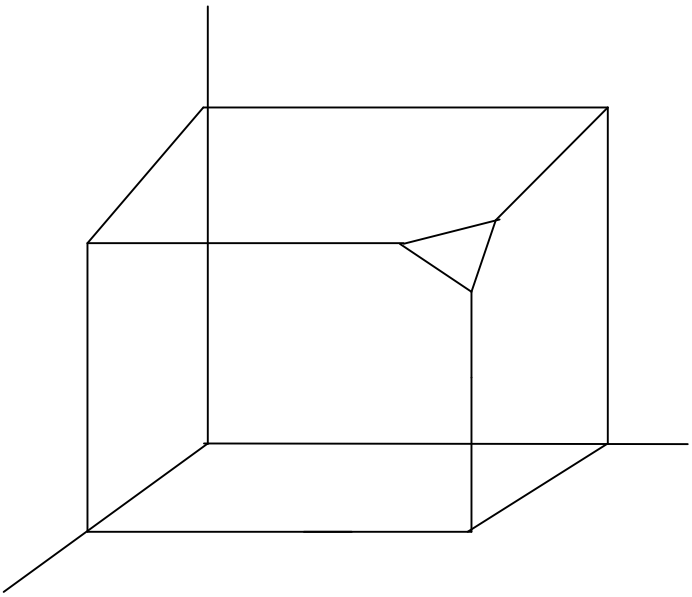
Enter your choice : 4

Enter the scaling factor

2

2

2



3D Transformation

1.Draw cube
2.Tranformation
3.Rotation
4.Scaling
5.Exit

Enter your choice : 5

RESULT:

Thus the program to implement 3D transformation have been created successfully and the output was verified.

Exno:5 **IMPLEMENTATION OF 3D PROJECTION**

AIM:

To write a c program for implementing the 3D projection.

ALGORITHM:

- Start the program.
- Initialize the graphics system
- Draw the 3D figure “cube” using the function “Draw code”.
- Read the axis & reference point about which the perspection projection is done.
- Draw the 3D projectional object.
- Stop the program.

PROGRAM

```
#include<process.h>
#include<dos.h>
#include<math.h>
#include<conio.h>
#include<iostream.h>
#include<graphics.h>
int gd=DETECT,gm;
int x1,x2,y1,y2;
void drawcube(double edge[20][3])
{
initgraph(&gd,&gm,"e:\\tc\\bgi");
int i;
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);
```

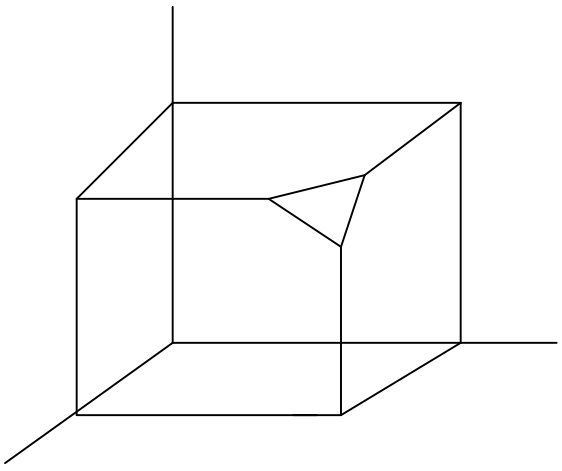
```
getch();
closegraph();
}
void perspective(double edge[20][3])
{
int ch,i;
double p,q,r;
clrscr();
cout<<"perspective projection about\n1.x-axis\n2.y-axis\n3.z-axis";
cin>>ch;
switch(ch)
{
case 1:
cout<<"enter the angle";
cin>>p;
for(i=0;i<20;i++)
{
edge[i][0]=(edge[i][0]/(p*edge[i][0]+1));
edge[i][1]=(edge[i][1]/(p*edge[i][0]+1));
edge[i][2]=(edge[i][2]/(p*edge[i][0]+1));
}
drawcube(edge);
break;
case 2:
cout<<"enter the angle";
cin>>q;
for(i=0;i<20;i++)
{
edge[i][1]=(edge[i][1]/(q*edge[i][1]+1));
edge[i][2]=(edge[i][2]/(q*edge[i][1]+1));
edge[i][0]=(edge[i][0]/(q*edge[i][1]+1));
}
drawcube(edge);
break;
case 3:
cout<<"enter the angle";
cin>>r;
for(i=0;i<20;i++)
{
edge[i][2]=(edge[i][2]/(q*edge[i][2]+1));
```

```
edge[i][0]=(edge[i][0]/(q*edge[i][2]+1));
edge[i][1]=(edge[i][1]/(q*edge[i][2]+1));
}
drawcube(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};
while(1)
{
clrscr();
cout<<"3d perspective \t\t\n";
cout<<"\n1.drawcube\n2.perspective\n3.exit\n";
cout<<"enter the choice";
cin>>choice;
switch(choice)
{
```

```
case 1:
drawcube(edge);
break;
case 2:
perspective(edge);
break;
case 3:
exit(0);
default:
cout<<"enter the valid choice";
break;
}
getch();
closegraph();
}
}
```

OUTPUT:

- 3D perspective projection:
- 1.Draw cube
- 2.Perspective
- 3.Exit
- Enter your choice : 1



3D perspective projection:

1.Draw cube

2.Perspective

3.Exit

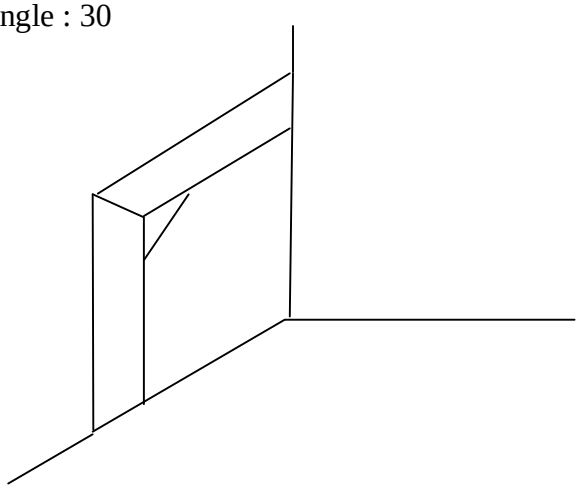
Enter your choice : 2

perspective about..

1.X-axis 2.Y-axis 3.Z-axis

Enter ur choice : 1

Enter the angle : 30



3D perspective projection:

1.Draw cube

2.Perspective

3.Exit

Enter your choice : 3

RESULT:

Thus the program to implement 3D projection is created and verified.

Ex no:6(a) IMPLEMENTATION OF COLOR MODEL HSV TO RGB

AIM:

To convert the HSV values to RGB values using C program.

ALGORITHM :

- Start the program
- Input the HSV values in the range of 0 to 1
- If the values of s is zero then it is gray.
- Scale & values of RGB become value of 1.
- Otherwise calculate the value of RGB with HSV values.
- Display RGB values & stop the execution.

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void hsvtorgb(float h,float s,float v)
{
int i;
float r,g,b,aa,bb,cc,f;
if(s==0)
r=g=b=r;
else
{
if(h==1.0)
h=0;
h*=6.0;
i=floor(h);
f=h-i;
aa=v*(1-s);
bb=v*(1-(s*f));
cc=v*(1-(s*(1-f)));
switch(i)
{
case 0:
r=v;
g=cc;
b=aa;
```

```
break;
case 1:
r=bb;
g=v;
b=cc;
break;
case 2:
r=aa;
g=bb;
b=v;
break;
case 3:
r=aa;
g=bb;
b=v;
break;
case 4:
r=cc;
g=aa;
b=v;
break;
case 5:
r=v;
g=aa;
b=bb;
break;
}
}
printf("\n the RGB values are:");
printf("\n r=%f,g=%f,b=%f",r,g,b);
}
void main()
{
float h,s,v;
clrscr();
printf("\n Enter the HSV values(0-1):");
scanf("%f%f%f",&h,&s,&v);
hsvtorgb(h,s,v);
getch();
}
```

OUTPUT:

Enter the HSV values (0-1):
0.2
0.4
0.6

The RGB values are:
r=0.552000,g=0.600000,b=0.408000

RESULT:

Thus the above program have been created successfully and output is verified.

Ex no:6(b) IMPLEMENTATION OF COLOR MODEL RGB TO HSV

AIM:

To convert the RGB values to HSV values using C program.

ALGORITHM :

- Start the program
- Input the RGB values .
- The RGB colors can be changed to HSV color using rgb to hsv() function.
- Display HSV values
- Stop the execution.

PROGRAM

```
const NO_hue=1;
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define MIN(a,b)(a<b?a:b)
#define MAX(a,b)(a>b?a:b)
void rgbtohsv(float r,float g,float b)
{
float h,s,v;
float max=MAX(r,MAX(g,b)),min=MIN(r,MIN(g,b));
float delta=max-min;
v=max;
```

```
if(max!=0.0)
s=delta/max;
else
s=0.0;
if(r==max)
h=(g,b)/delta;
else if(g==max)
h=2+(b-r)/delta;
else if(b==max)
h=4+(r-9)/delta;
h=60.0;
if(h<0)
h+=360.0;
h/=360.0;
printf("\n The HSV values are :");
printf("\n h=%f,s=%f,v=%f",h,s,v);
}

void main()
{
float r,g,b;
clrscr();
printf("Enter the RGB value(enter the values from 0-1);");
scanf("%f%f%f",&r,&g,&b);
rgbtohsv(r,g,b);
getch();
}
```

OUTPUT:

```
Enter the RGB value(enter the values from 0-1);
0.5
0.6
0.4

The HSV values are :
h=0.166667,s=0.333333,v=0.600000
```

RESULT:

Thus the above program have been created successfully and output is verified.

Ex no:7 **IMPLEMENTATION OF TEXT COMPRESSION**

AIM:

To write a program to perform text compression algorithm.

ALGORITHM:

- Start the program.
- Write some text in the input file
- Using the text in input file & compress the content.
- Write the compressed text (i.e.) translated into another form into output file.
- Stop the execution.

PROGRAM

```
#include<stdio.h>
#include<conio.h>
void main()
{
FILE *fp1,*fp2;
int i=1;
char a,c;
clrscr();
fp1=fopen("a.txt","r");
fp2=fopen("b.txt","w");
a=getc(fp1);
while(!feof(fp1))
{
c=fgetc(fp1);
if(c==a)
{
a=c;
i++;
}
else
{
fprintf(fp2,"%d%c",i,a);
a=c;
i=1;
}
}
printf("successfully compressed");
fclose(fp1);
fclose(fp2);
getch();
}
```

INPUT FILE:
a.txt

aaa
b.txt

3a2b

OUTPUT:

successfully compressed

RESULT:

Thus the above program performed successfully and the output is verified.

Ex no:8 **IMPLEMENTATION OF IMAGE COMPRESSION**

AIM:

To write a c program to perform image compression algorithm.

ALGORITHM:

- Start the program.
- Open the bmp file in read mode.
- Compress the content of the bmp file.
- Print the compressed text another file.
- Stop the execution.

PROGRAM

```
#include<stdio.h>
#include<conio.h>
void main()
{
FILE *fp1,*fp2;
int i=1;
char a,c;
clrscr();
```

```
fp1=fopen("w.jpg","r");
fp2=fopen("bb.jpg","w");
a=fgetc(fp1);
while(!feof(fp1))
{
c=fgetc(fp1);
if(c==a)
{
a=c;
i++;
break;
}
else
{
fprintf(fp2,"%d%c",i,a);
a=c;
i=1;
}
}
printf("\n successfully compressed");
fclose(fp1);
fclose(fp2);
getch();
}
```

INPUT FILE:
w.jpg
bb.jpg

OUTPUT:

successfully compressed

RESULT:

Thus the above program performed successfully and the output is verified.

Ex.no 9a **TEXT TWEENING**

AIM

To convert a set of character to another set of character using tweening

ALGORITHM

Open a macromedia flash's from start option

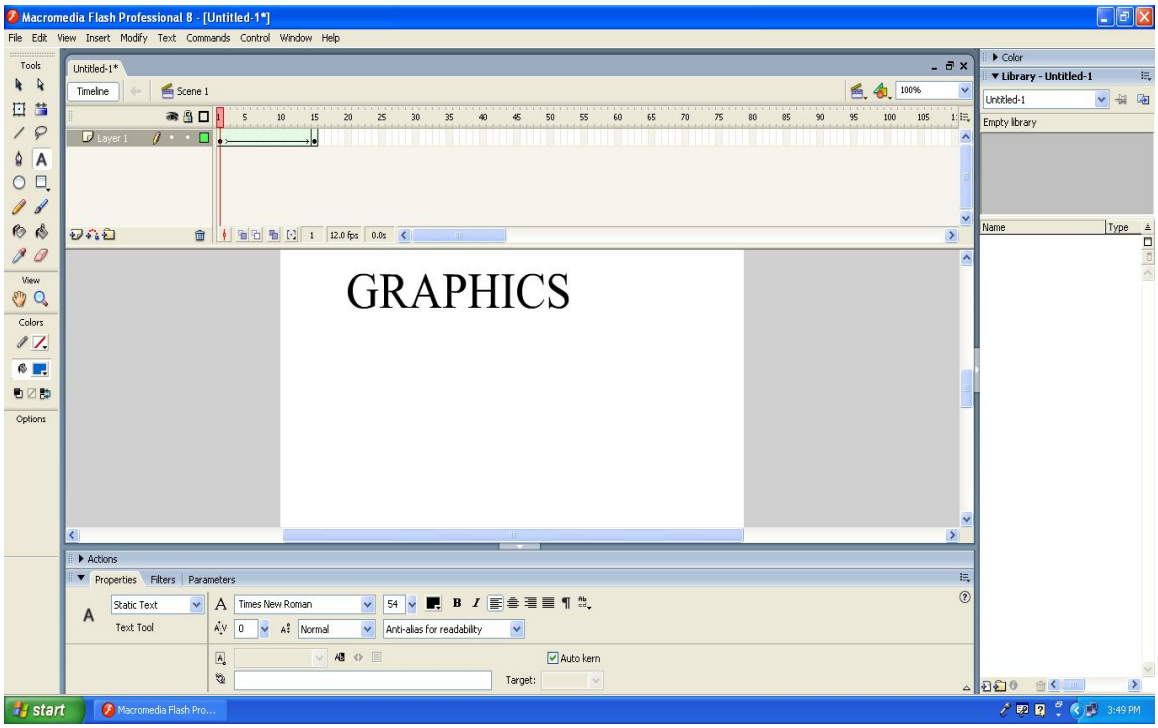
Select a text bar from the tool bar and place it on the play area then type some text, then select some text and press ctrl +B twice

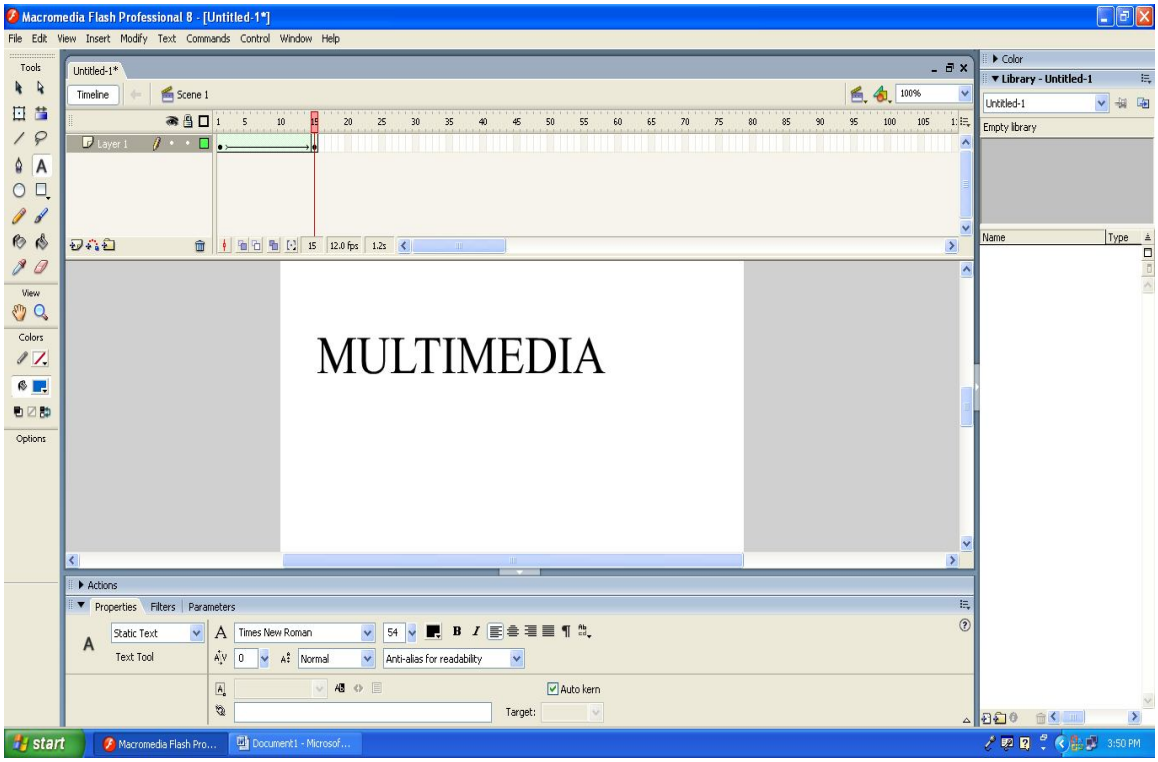
Select a frame, right click and select insert key frame and perform step 2

Select the first text, effect window -> frame and select tween as a shape and do the same for the second object

Press ctrl + enter to test the tweening

OUTPUT:





RESULT:

Thus the program for implementing the text tweening was created, executed and the output was verified successfully.

Ex No:9b **SHAPE TWEENING**

AIM

To convert the object from one shape to another shape

ALGORITHM

Open a macromedia flash from the start menu

From the toolbar select an object and place it on the play area

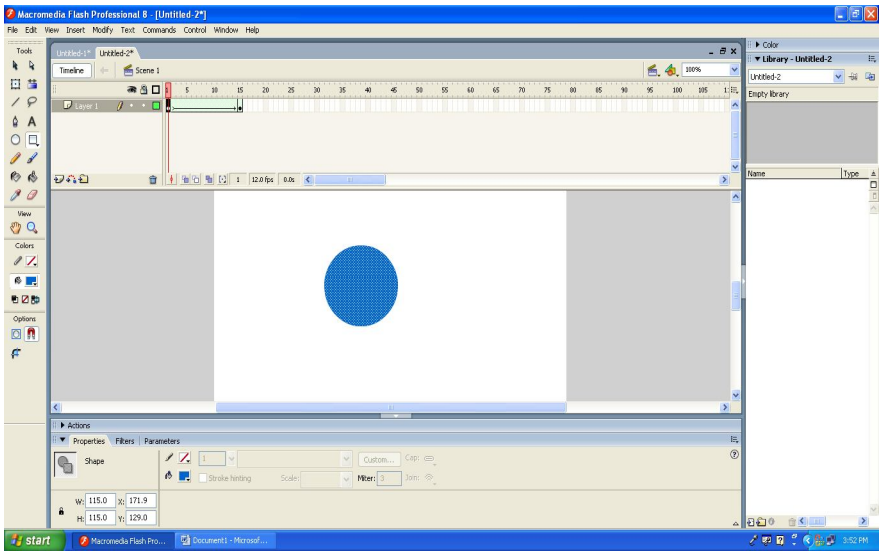
Select the frame, select the old object and delete it and draw the new object with new different colour and shape

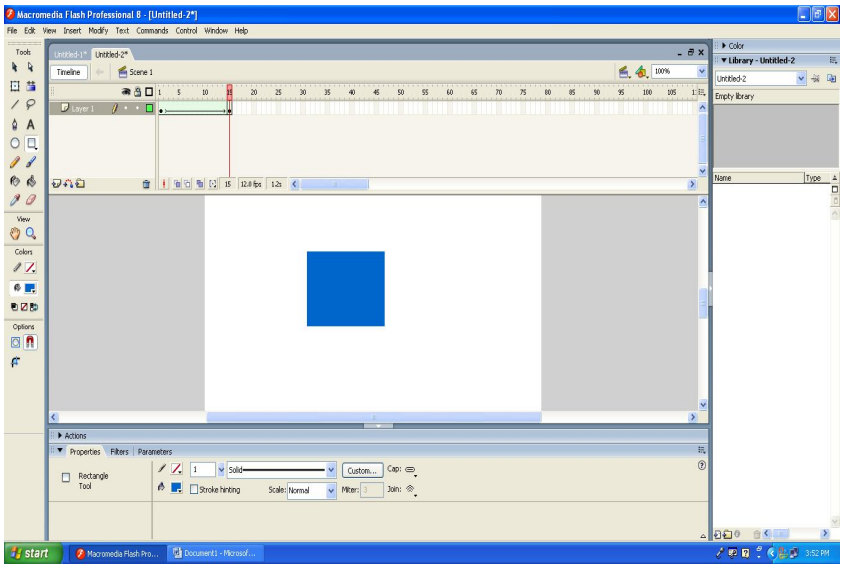
In the frame, right click and select insert key frame

Select the first object, offset window -> select tween as shape and same for the second object

Press ctrl + enter to test the movie.

OUTPUT:





RESULT:

Thus the program for implementing the shape tweening was created, executed and the output was verified successfully

EX NO:10a **IMAGE EDITING USING ADOBE PHOTOSHOP(FILTER)**

AIM:

To develop a filter image using adobe photoshop

ALGORITHM:

Open adobe photoshop 1.0.

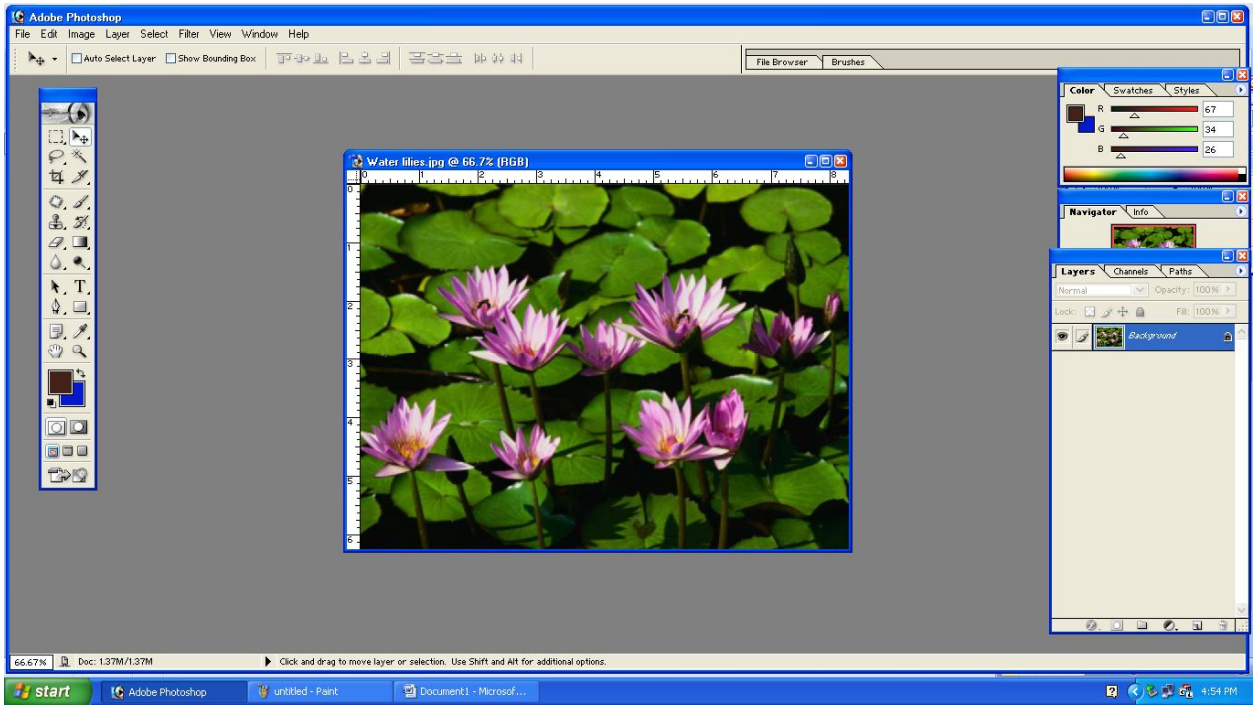
Select the file menu -> click open -> browse a picture

Select filter menu from the title bar

Select filter option -> blur -> radial blur

Set the appropriate radial to blur the picture and click ok

OUTPUT: PATTERN



Result:

Thus the program for generating filtered image was created, executed and the output was verified successfully.

Ex No:!0b **IMAGE EDITING USING ADOBE PHOTOSHOP(PATTERN)**

AIM

To generate a repeated pattern of image using adobe photoshop

ALGORITHM

Select the file menu and then click open for a sample picture

Select filter menu from the menu bar

Choose pattern maker from the filter menu

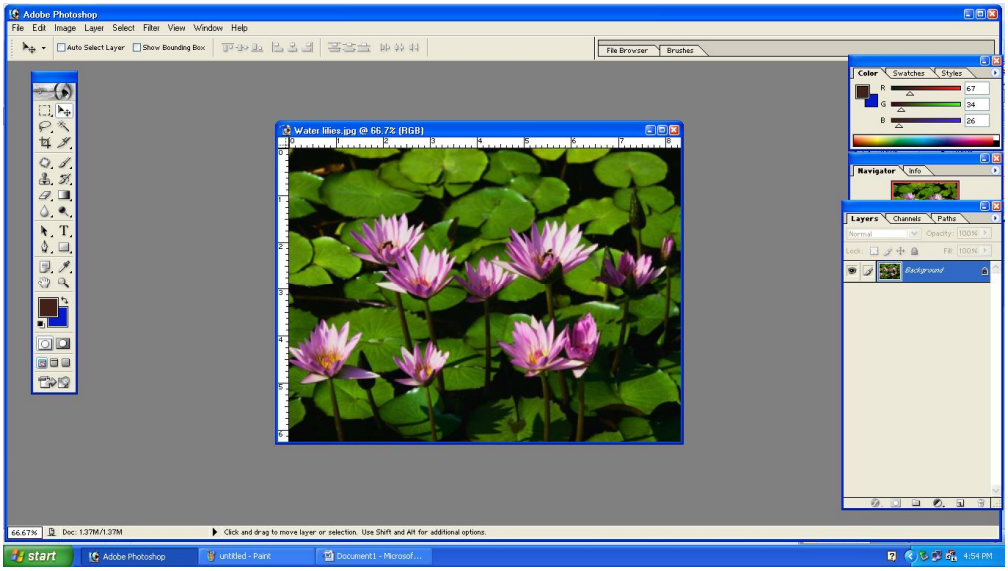
New window will be opened and the browsed image will be present

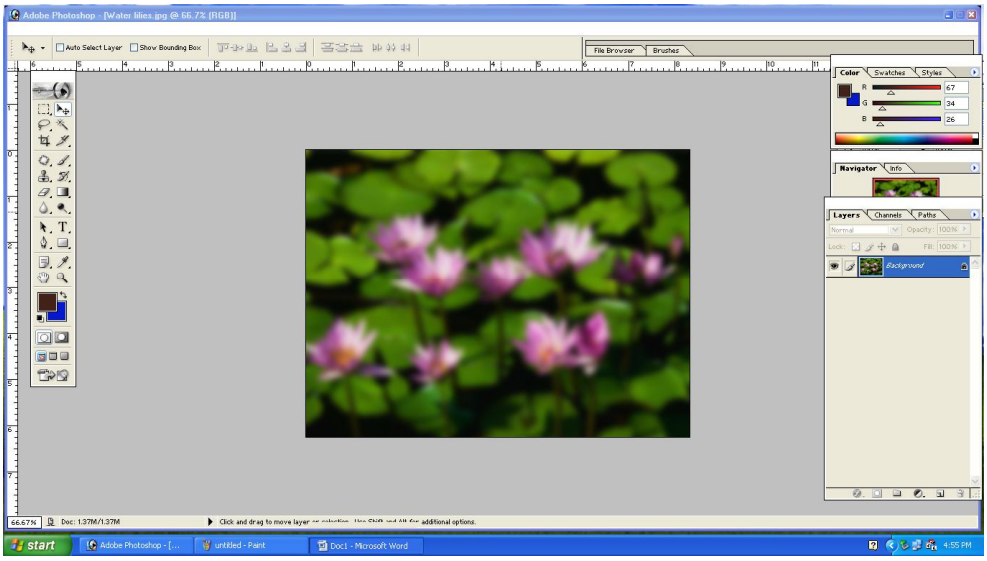
Select a pattern of the picture and click generate

Finally a pattern of image will be displayed.

OUTPUT:

BLUR





Result:

Thus the program for generating a repeated pattern of image was created, executed and the output was verified successfully.