ANNAMALAI UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

B.E. [COMPUTER SCIENCE AND ENGINEERING]

SEMESTER - V

CSCP507 - COMPUTER GRAPHICS AND MULTIMEDIA LAB

LABORATORY MANUAL

(July 2023 – November 2023)

TABLE OF CONTENTS

GRAPHICS

GIMP

AUDACITY

WINDOWS MOVIE MAKER

SWISH

FLASH

PHOTO IMPACT

LAB INCHARGES

A1 Batch – Dr. G. ARULSELVI

A2 Batch – Dr. R. SAMINATHAN

Ex. No. 1 (A)	IMPLEMENTATION OF BRESENHAM'S LINE DRAWING ALGORITHM

AIM

To draw a line using Bresenham's line drawing algorithm using C program.

ALGORITHM

Step 1: Start the program.
Step 2: Declare the necessary variables.
Step 3: Initialize the graph using dx, dy, gd, gm.
Step 4: Assign the values of x1, y1 to x, y respectively.
Step 5: Similarly, absolute values to dx, dy.
Step 6: put pixel and set 15 to pixel position.
Step 7: Using do-while loop, put e,x,y,I values.
Step 8: Stop the program.

PROGRAM

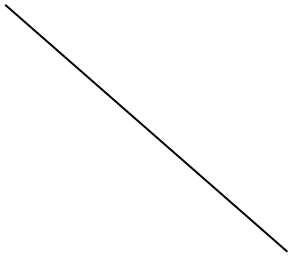
```
#include<stdio.h>
#include<graphics.h>
void drawline(int x0, int y0, int x1, int y1)
{
int dx, dy, p, x, y;
 dx=x1-x0;
dy=y1-y0;
x=x0;
y=y0;
p=2*dy-dx;
while((x<=x1)&&(y<=y1))
{
if(p>=0)
{
putpixel(x,y,7);
If(y!=y1) y=y+1;
p=p+2*dy-2*dx;
}
else
{
putpixel(x,y,7);
p=p+2*dy;
}
if(x!=x1) x=x+1;
if((x==x1)&&(y==y1))
break;
}
}
```

```
int main()
{
int gdriver=DETECT, gmode, error, x0, y0, x1, y1;
initgraph(&gdriver, &gmode, "c:\\turboc3\\bgi");
printf("Enter first co-ordinates point: ");
scanf("%d%d", &x0, &y0);
printf("Enter second co-ordinates point: ");
scanf("%d%d", &x1, &y1);
drawline(x0, y0, x1, y1);
return 0;
}
```

OUTPUT

Enter first co-ordinates point: 100 100

Enter the second co-ordinates point: 300 300



RESULT

Thus, the program to draw a line using Bresenham's algorithm is implemented and executed successfully.

Ex. No. 1(B)	IMPLEMENTATION OF BRESENHAM'S CIRCLE DRAWING ALGORITHM

AIM

To draw a circle using Bresenham's circle drawing algorithm using C program.

ALGORITHM

Step 1: Start the program.
Step 2: Declare the necessary variables.
Step 3: Create the function for Circle.
Step 4: Enter the radius and center values.
Step 5: Initialize the graph with gd, gm and assign y<-radius.
Step 6: Start the circle function and p<- 1- radius.
Step 7: Check the while loop until the condition is satisfied.
Step 8: Check the if –else condition until the condition is satisfied.
Step 9: Assign all operation for circle function and the values.
Step 10: Stop the Program.

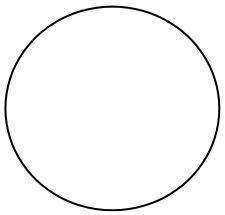
PROGRAM

```
#include<stdio.h>
#include<graphics.h>
void main()
{
int gd=DETECT,gm=0,xa,ya,r,p,k,x,y;
initgraph(&gd,&gm,"D:\\TC\\BGI");
printf("Enter the coordinates");
scanf("%d%d",&xa,&ya,);
printf("Enter the radius");
scanf("%d",&r);
p=1-r;
x=0;
y=r;
for(k=0;x<y;k++)
{
if(p<0)
{
p=p+(2*x)+2+1;
x=x+1;
}
else
{
p=p+(2*x)+2+1-(2*y)+2;
x=x+1;
y=y-1;
```

```
}
putpixel(xa+x,ya+y,1);
putpixel(xa-x,ya+y,2);
putpixel(xa+x,ya-y,3);
putpixel(xa-x,ya-y,4);
putpixel(xa+y,ya+x,5);
putpixel(xa-y,ya+x,6);
putpixel(xa+y,ya-x,7);
putpixel(xa-y,ya-x,8);
}}
```

OUTPUT

```
Enter the coordinates: 75 50
Enter the radius: 40
```



RESULT

Thus, the C program to draw a circle using Bresenham's algorithm is implemented and executed successfully.

Ex. No. 2	IMPLEMENTATION OF BRESENHAM'S ELLIPSE DRAWING ALGORITHM

AIM

To draw an ellipse using Bresenham's ellipse drawing algorithm using C progam.

ALGORITHM

Step 1: Start
Step 2: Declare the necessary variables .
Step 3: Initialize the gd, gm.
Step 4: Get the input, X radius of the ellipse and Y radius of the ellipse.
Step 5: Calculate rxsq,rysq, tworxsq, tworxsq values.
Step 6: Initialize the x value to 0 and y to ry.
Step 7 : Calculate the dx and dy values.
Step 8: In do loop, create a function to draw ellipse.
Step 9: Check the if –else conditions, and assign all the values to draw the ellipse.
Step 10: Stop the program.

PROGRAM

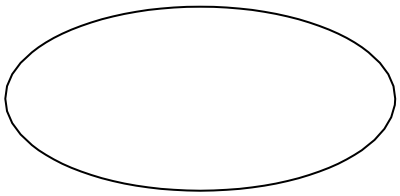
```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>
#include <dos.h>
void main()
{
long int d1,d2;
int i,gd=DETECT ,gm,x,y;
long int rx,ry,rxsq,rysq,tworxsq,tworysq,dx,dy;
printf("Enter the x Radius of the ellipse");
 scanf("%ld",&rx);
printf("Enter the y Radius of the ellipse");
scanf("%ld",&ry);
initgraph(&gd,&gm," ");
rxsq=rx*rx;
rysq=ry*ry;
tworxsq=2*rxsq;
tworysq=2*rysq;
x=0;
y=ry;
d1=rysq - (rxsq * ry) + (0.25 * rxsq);
dx= tworysq * x;
dy= tworxsq * y;
do
```

```
{
putpixel(200+x,200+y,15);
putpixel(200-x,200-y,15);
putpixel(200+x,200-y,15);
putpixel(200-x,200+y,15);
if (d1 < 0)
{
x=x+1;
y=y;
dx=dx + tworysq;
d1=d1 + dx + rysq;
}
else
{
x=x+1;
y=y-1;
dx= dx + tworysq;
dy= dy - tworxsq;
d1= d1 + dx - dy + rysq;
}
delay(50);
}
while (dx < dy);
d2 = rysq * ( x + 0.5) * ( x + 0.5 ) + rxsq * (y - 1) * (y-1) - rxsq * rysq;
do
{
putpixel(200+x,200+y,15);
putpixel(200-x,200-y,15);
putpixel(200+x,200-y,15);
putpixel(200-x,200+y,15);
if (d2 >0)
{
x=x;
y=y-1;
dy = dy - tworxsq;
d2 = d2 - dy + rxsq;
}
else
{
x= x+1;
y=y-1;
dy=dy - tworxsq;
dx= dx + tworysq;
d2 = d2 + dx -dy + rxsq;
```

```
}
delay(50);
} while ( y> 0);
getch();
closegraph();
}
```

OUTPUT

```
Enter the x Radius of the ellipse: 100
Enter the y Radius of the ellipse: 50
```



RESULT

Thus, the program to draw an ellipse using Bresenham's algorithm is implemented and executed successfully.

Ex. No. 3	GENERATION OF LINE, CIRCLE AND ELLIPSE ATTRIBUTES

AIM

To draw line, circle and ellipse attributes using C program.

ALGORITHM

Step 1: Start

Step 2: Declare the necessary variables and functions to draw line, circle and ellipse.

Step 3: Initialize the gd, gm.

Step 4: Create a while and get the input choice form the user.

Step 5: Using switch case statement, call the required function to draw line, circle and ellipse attributes.

Step 6: If input choice is 1, Get the input co-ordinates points to draw a line using DDA algorithm. Get the input for the line style and fill style.

Step 7: Draw the line using the line style and fill style and exit.

Step 8: If input choice is 2, Get the radius to draw the circle. Get the centre co-ordinates, number for color, and number for fill style.

Step 9: Draw the circle using the number of color and fill style and exit.

Step 10: If the input choice is 3, Get the X radius and Y radius of the ellipse, get the centre co- ordinates, number for colour, and number for fill style.

Step 11: Stop the program.

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void lineatt();
void ciratt();
void ellatt();
void main()
{
int gd=DETECT,gm,f=1,ch;
initgraph(&gd,&gm,"..//bgi");
while(f==1)
{
clrscr();
cleardevice();
printf("\n **********MENU**********");
printf("1:Line attributes\n");
printf("2:Circle attributes\n");
printf("3:Ellipse attributes\n");
printf("4:Exit\n");
printf("\n *************************");
```

```
printf("\n \n Enter your choice : ");
scanf("%d",&ch);
switch(ch)
{
case 1:
lineatt();
break;
case 2:
ciratt();
break;
case 3:
ellatt();
break;
case 4:
f=0;
exit(0);
}}
getch();
closegraph();
}
void lineatt()
{
/* the names of the line styles supported
{0->"SOLID_LINE",1->"DOTTED_LINE",2->"CENTER_LINE",3-
>"DASHED_LINE"}*/
int x1,y1,x2,y2,sl,lt,lc;
printf("\n Enter line co-ordinate points:");
scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
printf("\n Enter a number for line style: \n");
scanf("%d",&sl);
printf("\n Enter for 1/2/3 for thickness:\n");
scanf("%d",&lt);
printf("\n Enter a number for color: \n");
scanf("%d",&lc);
clrscr();
cleardevice();
setcolor(lc);
setlinestyle(sl,1,lt);
line(x1,y1,x2,y2);
 getch();
}
void ciratt()
{
int x,y,c,cc,cf;
```

```
/* Fillstyle: { "EMPTY_FILL","SOLID_FILL","LINE_FILL","LTSLASH_FILL",
"SLASH_FILL","BKSLASH_FILL","LTBKSLASH_FILL",
"HATCH_FILL","XHATCH_FILL","INTERLEAVE_FILL", "WIDE_DOT_FILL",
"CLOSE_DOT_FILL", "USER_FILL"} */
printf("\n Enter radius:");
scanf("%d",&c);
printf("\n Enter center co-ordinates:");
scanf("%d%d",&x,&y);
printf("\n Enter a number for color:");
scanf("%d",&cc);
printf("\n Enter a number for fill style:");
scanf("%d",&cf);
clrscr();
cleardevice();
setcolor(cc);
setfillstyle(cf,cc);
circle(x,y,c);
floodfill(x,y,cc);
getch();
}
void ellatt()
{
int x,y,xc,yc,ec,ef;
printf("\n Enter X radius:");
scanf("%d",&xc);
printf("\n Enter Y radius:");
scanf("%d",&yc);
printf("\n Enter center co-ordinates:");
scanf("%d%d",&x,&y);
printf("\n Enter a number for color:");
scanf("%d",&ec);
printf("\n Enter a number for fill style:");
scanf("%d",&ef);
clrscr();
cleardevice();
setcolor(ec);
setfillstyle(ef,ec);
ellipse(x,y,0,360,xc,yc);
fillellipse(x,y,xc,yc);
getch();
}
```

OUTPUT

********MENU*********

1. Line attributes
2. Circle attributes
3. Ellipse attributes
4. Exit

****

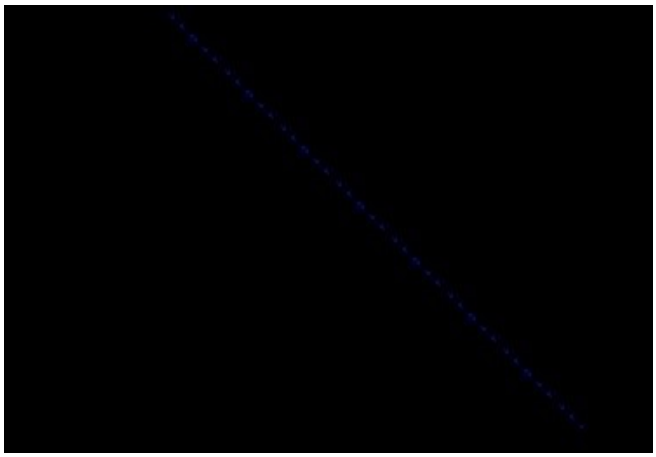
Enter your choice: 1

Enter line co-ordinate points: 100 100 250 250

Enter a number for line style : 1

Enter for 1/2/3 for thickeness : 1

Enter a number for color: 1



********MENU*********

1. Line attributes
2. Circle attributes
3. Ellipse attributes
4. Exit

****

Enter your choice: 1

Enter line co-ordinate points: 100 100 250 250

Enter a number for line style: 2

Enter for 1/2/3 for thickness: 2

Enter a number for color: 2

********MENU*********

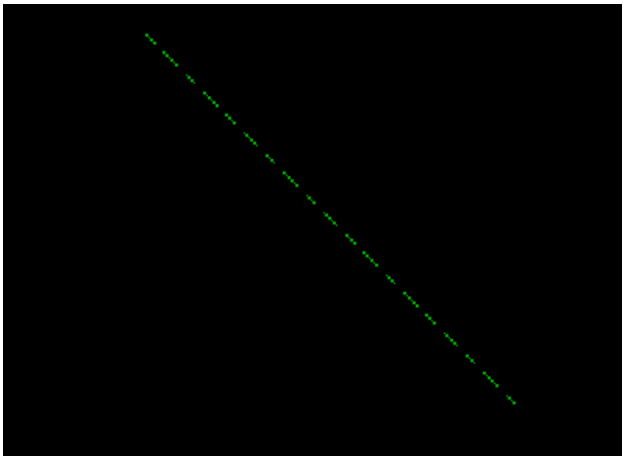
1. Line attributes
2. Circle attributes
3. Ellipse attributes
4. Exit

****

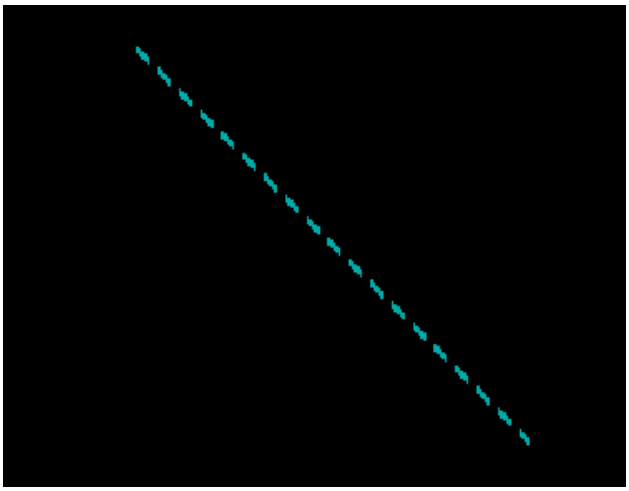
Enter your choice: 1

Enter line co-ordinate points: 100 100 250 250

Enter a number for line style: 2
Enter for 1/2/3 for thickness: 2
Enter a number for color: 2

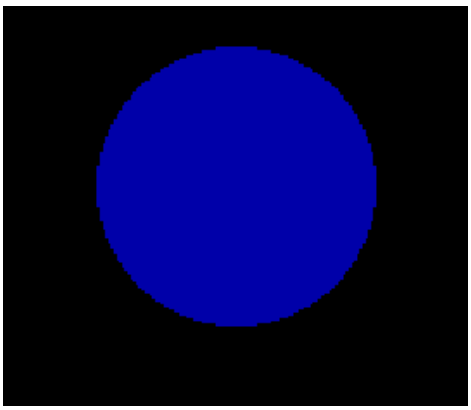


********MENU*********
1. Line attributes
2. Circle attributes
3. Ellipse attributes
4. Exit
****
Enter your choice: 1
Enter line co-ordinate points: 100 100 250 250
Enter a number for line style: 3
Enter for 1/2/3 for thickness: 3
Enter a number for color: 3

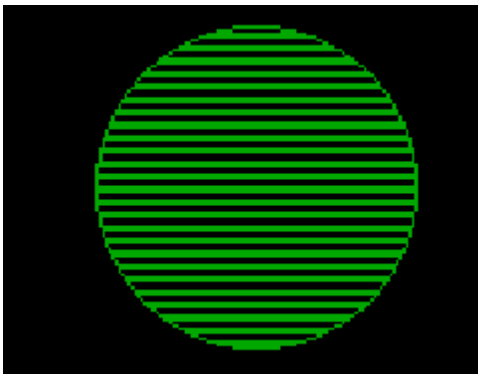


********MENU*********
1. Line attributes
2. Circle attributes
3. Ellipse attributes
4. Exit
****

Enter your choice: 2
Enter radius: 50
Enter centre co-ordinates: 100 100
Enter a number for color:1
Enter a number for fillstyle: 1

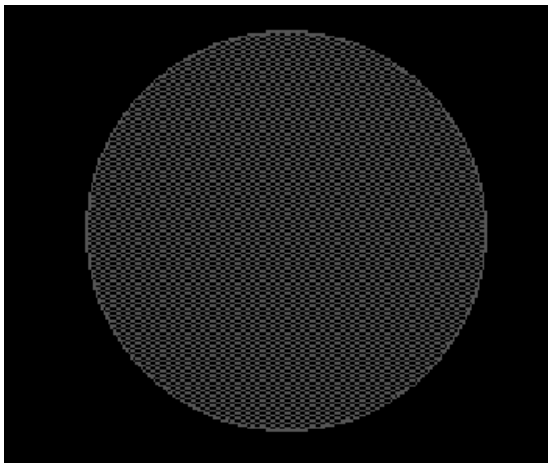


********MENU*********
1. Line attributes
2. Circle attributes
3. Ellipse attributes
4. Exit
****
Enter your choice: 2
Enter radius: 50
Enter centre co-ordinates: 100 100
Enter a number for color:2
Enter a number for fillstyle: 2

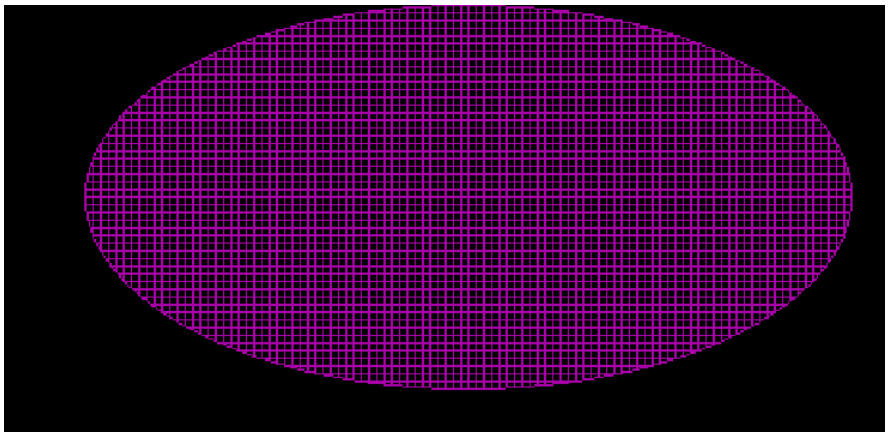


********MENU*********
1. Line attributes
2. Circle attributes
3. Ellipse attributes
4. Exit
****
Enter your choice: 2

Enter radius: 80
Enter centre co-ordinates: 1580 150
Enter a number for color:
Enter a number for fillstyle: 9



********MENU*********
1. Line attributes
2. Circle attributes
3. Ellipse attributes
4. Exit
****
Enter your choice: 3
Enter x radius: 200
Enter y radius: 100
Enter center co-ordinates: 200 100
Enter a number for color: 5
Enter a number for fill style: 7



RESULT

Thus, the program to draw line, circle and ellipse attributes is implemented and executed successfully.

Ex. No. 4	TWO DIMENSIONAL TRANSFORMATIONS - TRANSLATION, ROTATION, SCALING, REFLECTION AND SHEAR

AIM

To implement 2D transformations using C program.

i) Translation
ii) Rotation
iii) Scaling
iv) Reflection
v) Shear

ALGORITHM

Step 1: Start the program.
Step 2: Declare the necessary variables and initialize the graph, gd, gm.
Step 3: Use do-while loop and declare the function clear device.
Step 4: Create four cases translation, scaling, rotation and exit.
Step 5: In case 1 enter the translation values and print the translation object.
Step 6: In case 2 enter the scaling values and print the scaling object.
Step 7: In case 3 enter the rotation values and print rotation object.
Step 8: Clockwise rotation and counter clockwise rotation use the same equation.
Step 9: Stop the program.

PROGRAM

```
#include<stdio.h>
#include<conio.h>
 #include<math.h>
#include<graphics.h>
int ch,x,y,az,i,w,ch1,ch2,xa,ya,ra,a[10],b[10],da,db;
float x1,y1,az1,w1,dx,dy,theta,x1s,y1s,sx,sy,a1[10],b1[10];
void main()
{
int gm, gr;
clrscr();
detectgraph(&gm,&gr);
initgraph(&gm,&gr,"d:\\tc\\BGI");
printf("Enter the upper left corner of the rectangle:\n");
scanf("%d%d",&x,&y);
printf("Enter the lower right corner of the rectangle:\n");
scanf("%d%d",&az,&w);
rectangle(x,y,az,w);
da=az-x;
db=w-y;
a[0]=x;
```

```
b[0]=y;
a[1]=x+da;
b[1]=y;
a[2]=x+da;
b[2]=y+db;
a[3]=x;
b[3]=y+db;
while(1)
{
printf("******2DTransformations*******\n");
printf("1.Translation\n 2.Rotation\n 3.Scaling\n 4.Reflection\n 5.Shearing\n 6.Exit\n
Enter your choice:\n");
scanf("%d",&ch);
switch(ch)
{
case 1:
detectgraph(&gm,&gr);
initgraph(&gm,&gr,"d:\\tc\\BGI");
rectangle(x,y,az,w);
printf("*******Translation*******\n\n");
printf("Enter the value of shift vector:\n");
scanf("%f%f",&dx,&dy);
 x1=x+dx;
y1=y+dy;
az1=az+dx;
w1=w+dy;
rectangle(x1,y1,az1,w1);
break;
case 2:
detectgraph(&gm,&gr);
initgraph(&gm,&gr,"d:\\tc\\BGI");
rectangle(x,y,az,w);
printf("*******Rotation*******\n\n");
printf("Enter the value of fixed point and angle of rotation:Enter the value of fixed point
and angle of rotation:\n");
scanf("%d%d%d",&xa,&ya,&ra);
theta=(float)(ra*(3.14/180));
for(i=0;i<4;i++)
{
a1[i]=(xa+((a[i]-xa)*cos(theta)-(b[i]-ya)*sin(theta)));
b1[i]=(ya+((a[i]-xa)*sin(theta)+(b[i]-ya)*cos(theta)));
}
for(i=0;i<4;i++)
{
```

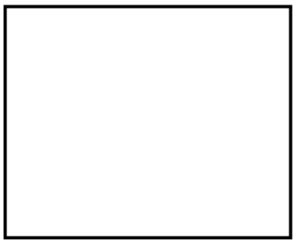
```
if(i!=3) line(a1[i],b1[i],a1[i+1],b1[i+1]);
else
line(a1[i],b1[i],a1[0],b1[0]);
}
break;
case 3:
detectgraph(&gm,&gr);
initgraph(&gm,&gr,"d:\\tc\\BGI");
rectangle(x,y,az,w);
printf("********Scaling*******\n\n");
printf("Enter the value of scaling factor:\n");
scanf("%f%f",&sx,&sy);
x1=x*sx;
y1=y*sy;
az1=az*sx;
w1=w*sy;
rectangle(x1,y1,az1,w1);
break;
case 4:
detectgraph(&gm,&gr);
initgraph(&gm,&gr,"d:\\tc\\BGI");
rectangle(x,y,az,w);
printf("*******Reflection*********\n");
printf("1.About x-axis\n2.About y-axis\n3.About both axis\nEnter your choice:\n");
scanf("%d",&ch1);
switch(ch1)
{
case 1:
printf("Enter the fixed point\n");
scanf("%d%d",&xa,&ya);
theta=(float)(90*(3.14/180));
for(i=0;i<4;i++)
{
a1[i]=(xa+((a[i]-xa)*cos(theta)-(-b[i]-ya)*sin(theta)));
b1[i]=(ya+((a[i]-xa)*sin(theta)+(-b[i]-ya)*cos(theta)));
}
for(i=0;i<4;i++)
{
 if(i!=3)
line(a1[i],b1[i],a1[i+1],b1[i+1]);
else line(a1[i],b1[i],a1[0],b1[0]);
}
break;
case 2:
```

```
printf("Enter the fixed point\n");
scanf("%d%d",&xa,&ya);
theta=(float)(270*(3.14/180));
for(i=0;i<4;i++)
{
a1[i]=(xa+((-a[i]-xa)*cos(theta)-(b[i]-ya)*sin(theta)));
b1[i]=(ya+((-a[i]-xa)*sin(theta)+(b[i]-ya)*cos(theta)));
}
for(i=0;i<4;i++)
{
if(i!=3) line(a1[i],b1[i],a1[i+1],b1[i+1]);
else
line(a1[i],b1[i],a1[0],b1[0]);
}
break;
case 3:
printf("Enter the fixed point\n");
scanf("%d%d",&xa,&ya);
theta=(float)(180*(3.14/180));
for(i=0;i<4;i++)
{
a1[i]=(xa+((-a[i]-xa)*cos(theta)-(-b[i]-ya)*sin(theta)));
b1[i]=(ya+((-a[i]-xa)*sin(theta)+(-b[i]-ya)*cos(theta)));
 }
for(i=0;i<4;i++)
{
if(i!=3) line(a1[i],b1[i],a1[i+1],b1[i+1]);
else
line(a1[i],b1[i],a1[0],b1[0]);
}
break;
}
break;
case 5:
detectgraph(&gm,&gr);
initgraph(&gm,&gr,"d:\\tc\\BGI");
rectangle(x,y,az,w);
printf("*******Shearing******\n\n");
printf("1.x-direction shear\n2.y-direction shear\nEnter your choice:\n");
scanf("%d",&ch2);
switch(ch2)
{
case 1:
printf("Enter the value of shear:\n");
```

```
scanf("%f",&x1s);
x1=x+(y*x1s);
y1=y;
az1=az+(w*x1s);
w1=w;
rectangle(x1,y1,az1,w1);
break;
case 2:
printf("Enter the value of shear:\n");
scanf("%f",&y1s);
x1=x;
y1=y+(x*y1s);
az1=az;
w1=w+(az*y1s);
rectangle(x1,y1,az1,w1);
break;
}
break;
case 6:
exit(0);
}}
getch();
}
```

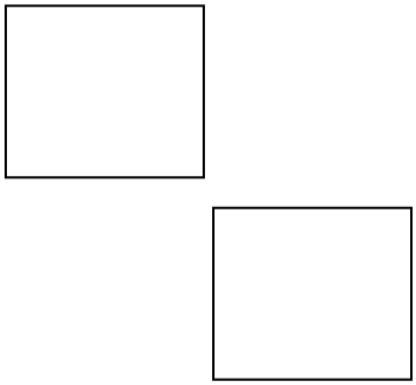
OUTPUT

```
Enter the upper left corner of the rectangle: 210 210
Enter the bottom left corner of the rectangle: 240 240
```



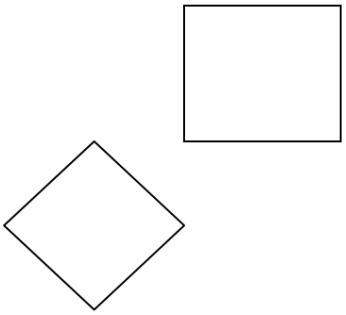
```
******* 2D Transformation *********
1.      Translation
2.      Rotation
3.      Scaling
4.      Reflection
5.      Shearing
6.      Exit
Enter your choice : 1
******** Translation *******
```

Enter the value of shift vector: 30 40



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

Enter your choice: 2
Enter the fixed point and angle of rotation: 30 30 30



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

Enter your choice: 3
********* Scaling ***********
Enter the value of scaling factor:
2
1

******* 2D Transformation **********

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

Enter your choice: 4

******** Reflection **********

1. About x-axis
2. About y-axis
3. About both axis

Enter your choice:1

Enter the fixed point 40 40

******* 2D Transformation **********

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

Enter your choice: 4

******** Reflection **********

1. About x-axis
2. About y-axis
3. About both axis

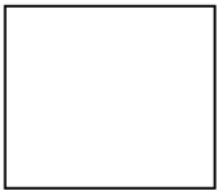
Enter your choice: 2

Enter the fixed point 40 40

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

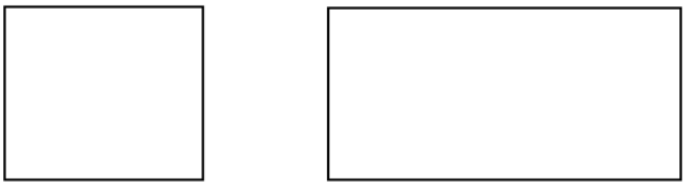
Enter your choice: 4
******** Reflection **********
1. About x-axis
2. About y-axis
3. About both axis

Enter your choice: 3
Enter the fixed point 40 40

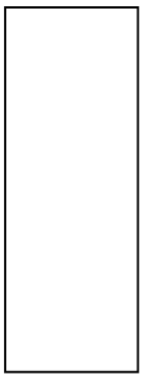


******* 2D Transformation **********
1. Translation
2. Rotation
3. Scaling
4. Reflection
5. Shearing
6. Exit

Enter your choice: 5
********* Shearing *********
1. X-direction shear
2. Y-direction shear
Enter your choice: 1
Enter the value of shear: 0.5



******* 2D Transformation ***********
1. Translation
2. Rotation
3. Scaling
4. Reflection
5. Shearing
6. Exit
Enter your choice: 5
********* Shearing *********
1. X-direction shear
2. Y-direction shear
Enter your choice: 2
Enter the value of shear: 0.5



RESULT

Thus, the program to implement 2D transformations is executed successfully.

Ex. No. 5 (A)	COHEN - SUTHERLAND 2D LINE CLIPPING ALGORITHM

AIM

To implement Cohen - Sutherland 2D line clipping algorithm using C program.

ALGORITHM

Step 1: Start.

Step 2: Get the co-ordinates of line (x11,y11) & (x22,y22)

Step 3: Get the co-ordinate of clipping window (xmin,ymin,xmax,ymax)

Step 4: Get the code region for line

(a)if(x11<xmin) a[3]=1

(b)if(x11>xmax) a[2]=1

(c)if(y11<ymin) a[1]=1;

(d)if(y11>ymax) a[0]=1;

(e)if(x22<xmin) b[3]=1;

(f)if(x22>xmax) b[2]=1;

(g)if(y22<ymin) b[1]=1;

(h)if(y22>ymax) b[0]=1;

Step 5: Calculate slope of line i.e m using following formula: m=(y22-y11)/(x22-x11)

Step 6: If a[3],a[2],a[1],a[0],b[3],b[2],b[1],b[0] all are zero Then the line is totally visible and not a clipping candidate

Step 7: Draw line with coordinates (x11,y11) & (x22,y22) and window with coordinates (xmin,ymin,xmax,ymax)

Step 8: Line will partially visible if any of following conditions is satisfied:

(i) if((a[0]=0)&(a[1]=1))

x11=x11+(ymin-y11)/m

y11=ymin

else if((b[0]=0)&(b[1]=1))

x22=x22+(ymin-y22)/m

y22=ymin

(ii) if((a[0]==1)&(a[1]==0))

x11=x11+(ymax-y11)/m

y11=ymax

else if((b[0]=1)&(b[1]=0))

x22=x22+(ymax-y22)/m;

y22=ymax;

(iii)if((a[2]=0)&(a[3]==1))

y11=

} }

else

{

```
clrscr();
clearviewport();
printf("\nLine is invisible");
rectangle(xmin,ymin, y11+m*(xmin-x11)
x11=xmin
else if((b[2]=0)&(b[3]=1))
y22=y22+m*(xmin-x22) x22=xmin

(iv)if((a[2]=1)&(a[3]=0))
y11=y11+m*(xmax-x11) x11=xmax
else if((b[2]=1)&(b[3]=0))
y22=y22+m*(xmax-x22)
x22=xmax
```

Step 9: Draw line with coordinates (x11,y11) & (x22,y22) and window with coordinates (xmin,ymin,xmax,ymax)

Step 10: If any condtion described in step-5 and step-7 is not satisfied, then line is invisible i.e outside of clipping window.

Step 11: Stop

PROGRAM

```
#include<stdio.h>
#include<conio.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:\\dosapp~1\\tc\\bgi");
printf("\nEnter the 1st co-ordinate of line: ");
scanf("%f %f ",&x11,&y11);
printf("\nEnter the 2nd co-ordinate of line: ");
scanf("%f %f ",&x22,&y22);
printf("\nEnter the co-ordinates of clipping window: ");
scanf("%f %f %f %f",&xmin,&ymin,&xmax,&ymax);
rectangle(xmin,ymin,xmax,ymax);
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;
for(i=0;i<4;i++)
{
c[i]=a[i]&&b[i];
}
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("\n The line is totally visible\n and 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))
```

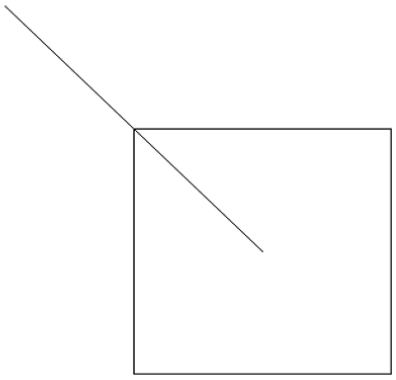
```
{
x1=x11+(ymax-y11)/m;
x11=x1;
y11=ymax;
}
else if((b[0]==1)&&(b[1]==0))
{
x1=x22+(ymax-y22)/m;
x22=x1;
y22=ymax;
}
if((a[2]==0)&&(a[3]==1))
{
y1=y11+m*(xmin-x11);
y11=y1;
x11=xmin;
}
else if((b[2]==0)&&(b[3]==1))
{
y1=y22+m*(xmin-x22);
y22=y1;
x22=xmin;
}
if((a[2]==1)&&(a[3]==0))
{
y1=y11+m*(xmax-x11);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

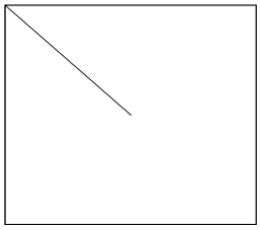
OUTPUT

Enter the 1-st co-ordinates of line: 150 300
Enter the 2nd co-ordinates of line: 450 300
Enter the co-ordinates of clipping window: 200 200 400 400

Line is partially visible

After clippling:

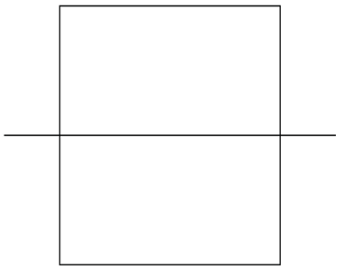


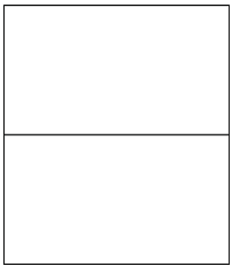


Enter the 1-st co-ordinates of line : 150 300
Enter the 2nd co-ordinates of line: 450 300
Enter the co-ordinates of clipping window: 200 200 400 400

Line is partially visible

After clippling:





RESULT

Thus, the program to implement Cohen-Sutherland 2D line clipping is executed successfully.

Ex. No. 5 (B)	WINDOWING TRANSFORMATION

AIM

To implement windowing transformation using C program.

ALGORITHM

Step 1: Start the program.
Step 2: Initialize the variables gd=DETECT, gm for graphics mode.
Step 3: Get the co-ordinate points to draw the object.
Step 4: Assign the co-ordinates for the world port.
Step 5: Using rectangle function for drawing the world port.
Step 6: Using line function draw the object.
Step 7: Assign the co-ordinates for the window port.
Step 8: Using rectangle function for drawing the window port.
Step 9: Apply scaling factor for the object in world port.
Step 10: Using line function draw the object.
Step 11: Stop the program.

PROGRAM

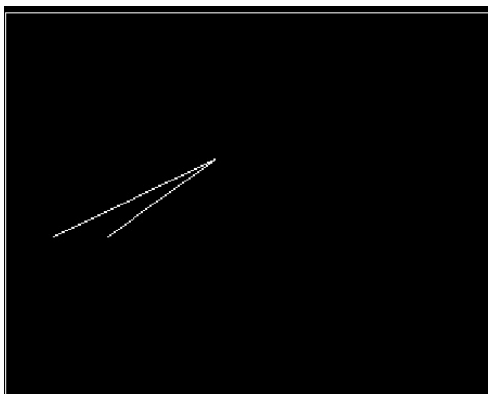
```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
main()
{
float sx,sy;
int w1,w2,w3,w4,x1,x2,x3,x4,y1,y2,y3,y4,v1,v2,v3,v4;
int gd=DETECT,gm;
initgraph(&gd,&gm,"..//bgi");
printf("Enter the Co-ordinates x1,y1,x2,y2,x3,y3\n");
scanf("%d%d%d%d%d%d",&x1,&y1,&x2,&y2,&x3,&y3);
cleardevice();
w1=5; w2=5; w3=635; w4=465;
rectangle(w1,w2,w3,w4);
line(x1,y1,x2,y2);
line(x2,y2,x3,y3);
getch();
v1=425; v2=75; v3=550; v4=250;
sx=(float)(v3-v1)/(w3-w1);
sy=(float)(v4-v2)/(w4-w2);
rectangle(v1,v2,v3,v4);
x1=v1+floor(((float)(x1-w1)*sx)+0.5);
x2=v1+floor(((float)(x2-w1)*sx)+0.5);
x3=v1+floor(((float)(x3-w1)*sx)+0.5);
y1=v2+floor(((float)(y1-w2)*sy)+0.5);
```

```
y2=v2+floor(((float)(y2-w2)*sy)+0.5);
y3=v2+floor(((float)(y3-w2)*sy)+0.5);
line(x1,y1,x2,y2);
line(x2,y2,x3,y3);
getch();
return 0;   }
```

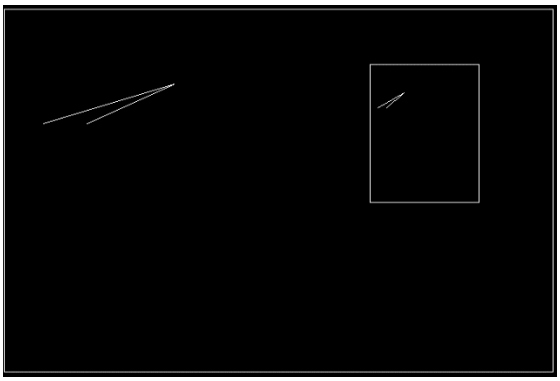
OUTPUT

```
Enter the Co-ordinates x1,y1,x2,y2,x3,y3 100
150
200
100
50
150
```

Before Transformation



After Transforation



RESULT

Thus, the C program to implement windowing transformation is implemented successfully.

Ex. No. 6	SUTHERLAND - HODGEMAN POLYGON CLIPPING ALGORITHM

AIM

To implement Sutherland - Hodgeman polygon clipping algorithm using C program.

ALGORITHM

Step 1: Start.

Step 2: Let p(1), p(2). p(n) be the vector list of the polygon to be clipped.

Step 3: Let edge E determined by end points A and B be any edge of a position counted convex clipping algorithm.

Step 4: we clip each edge of the polygon in turn against the edge of the clipping polygon forming a new polygon whose vertices are determined as follows consider the edge p(-1).

If both p (i-1) are to be left of edge vector, p(i) is placed on vertex o/p list of clipped algorithm.

If both p (i-1) and p(i) are the right of edge vertex, nothing is placed on the vertex o/p list.

If p (i-1) is the left and p(i) is to right of edge E, the intersection point i of the line segment p(i-1) p(i) with the extended edge E is calculated and placed on the vertex o/p.

Step 5: Stop.

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<stdlib.h>
#include<graphics.h>
#include<dos.h>
int code (int,int);
void dda(int,int,int,int);
int xt=100,yb=200,xr=400,yt=400,xi,yi;
void main()
{
int visible(int,int,int);
void rec(int,int,int,int);
int gd=DETECT,gm,n,i,m=0,c1,c2,ct=0,v,e,s;
int nx[20],ny[20],c,x1,x2,y1,y2,x[20],y[20];
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"e:\\tcplus\\bgi");
printf("\n Enter the number of verties:");
scanf("%d",&n);
for(i=0;i<n;i++)
```

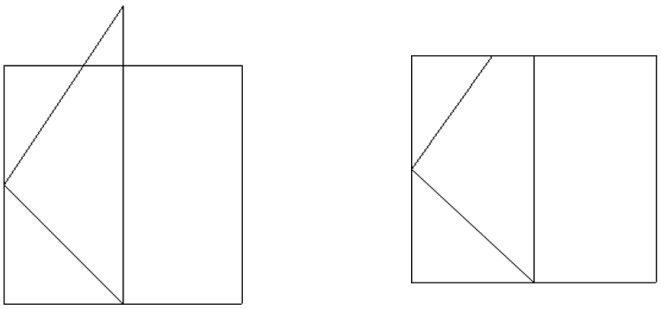
```
{
printf("\n Enter window co-ordinates:");
scanf("%d%d",&x[i],&y[i]);
} n++;
x[n]=x[1]; y[n]=y[1];
printf("\n Enter window coordinates");
scanf("%d%d%d%d",&xt,&yb,&xr,&yt);
cleardevice();
rec(xt,yb,xr,yt);
moveto(x[1],y[1]);
for(i=2;i<=n;i++)
lineto(x[i],y[i]);
getch();
cleardevice();
do{
m=0;
for(i=1;i<=n;i++)
{
if((i==1)&&(ct==0))
goto pro;
else
{
c2=code(x[i],y[i]);
c=(int)pow((float)2,(float)ct);
if(((c1&c)||(c2&c))!=0)
{
x1=x[s];x2=x[i]; y1=y[s];y2=y[i]; e=ct;
if(e==0){
xi=xt;
yi=y2-(float)(y2-y1)/(x2-x1)*(x2-xt);
}
if(e==1){
xi=xr;
yi=y2-(float)(y2-y1)/(x2-x1)*(x2-xr);
}
if(e==2){
yi=yt;
xi=x2-(float)(x2-x1)/(y2-y1)*(y2-yt);
}
if(e==3){
yi=yb;
xi=x2-(float)(x2-x1)/(y2-y1)*(y2-yb);
}
m++;
```

```
nx[m]=xi;
ny[m]=yi;
}}
pro: s=i;
c1=code(x[i],y[i]);
v=visible(x[i],y[i],ct);
if(v==1){
m++;
nx[m]=x[i]; ny[m]=y[i];
}}
ct++;
n=m;
for(i=1;i<=n;i++)
{
x[i]=nx[i]; y[i]=ny[i];
}}
while(ct<4);
i=1;
moveto(x[i],y[i]);
for(i=2;i<=n;i++)
lineto(x[i],y[i]);
lineto(x[1],y[1]);
rec(xt,yb,xr,yt);
getch();
closegraph();
}
int code(int a,int b)
{
int c=0; if(a<xt) c+=1;
if(a>xr) c+=2;
if(b<yb) c+=8;
if(b>yt) c+=4;
return(c);
}
int visible(int a,int b,int c)
{
int code(int,int);
int d,e;
d=code(a,b);
e=(int)pow((float)2,(float)c);
if((d&e)==0)
return(1);
else
return(0);
```

```
}
void rec(int xt,int yb,int xr,int yt)
{
dda(xt,yb,xr,yb);
dda(xr,yb,xr,yt);
dda(xr,yt,xt,yt);
dda(xt,yt,xt,yb);
}
void dda(int x1,int y1,int x2,int y2)
{
float x,y,dy,dx,len;
int i;
if(abs(y2-y1)>=abs(x2-x1)) len=abs(y2-y1);
else
len=abs(x2-x1);
dx=(x2-x1)/len;
dy=(y2-y1)/len;
x=x1+0.5*dx;
y=y1+0.5*dy;
for(i=1;i<=len;i++)
{
putpixel(x,y,getmaxcolor());
x+=dx; y+=dy;
}}
```

OUTPUT

```
Enter the no of vertices : 3
Enter the polygon coordinate :200 300
Enter the polygon coordinate :300 400
Enter the polygon coordinate :300 150
Enter the window coordinate : 200 200 400 400
```



RESULT

Thus, Sutherland - Hodgeman Polygon Clipping algorithm is implemented and output is verified successfully.

Ex. No. 7	THREE DIMENSIONAL TRANSFORMATIONS – TRANSLATION, ROTATION AND SCALING

AIM

To implement 3D transformations using C program.

ALGORITHM

Step 1: Start
Step 2: initialize ch,t,ty,x,sx,sy,sz,m=9;
Step 3: Get choice
Create translation
Get the translation vector t[i].x1=a[i].x1+tx t[i].y1=a[i].y1+ty t[i].z1=a[i].z1+tz
Scaling
Get the Scaling vector f[i].x1=a[i].x1*sx f[i].x1=a[i].x1*sx f[i].x1=a[i].x1*sx
Rotation
Get axis,angle theta
Rotation about x-axis x1= x
y1= ycos(theta)+zsin(theta) z1= ysin(theta)+zcos(theta)
Rotation about y-axis x1= xcos(theta)+ysin(theta) y1= y
z1= xsin(theta)+ycos(theta)
Rotation about z-axis x1= xcos(theta)+ysin(theta)
y1= -xsin(theta)+ycos(theta) z1= z
Step 4: Draw cube.
Step 5: Stop.

PROGRAM

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<graphics.h>
#include<process.h>
#include<dos.h>
#include<math.h>
int x,y,z;
struct cube
{
int x1,y1,z1;
};
struct cube1
{
int x2,y2,z2;
};
struct cube a[8]={0},t[8]={0},f[8]={0},g[8]={0},k[8]={0},l[8]={0};
struct cube1 b[8]={0},c[8]={0},d[8]={0},e[8]={0},h[8]={0},j[8]={0};
```

```
void init()
{
int gd=DETECT,gm;
initgraph(&gd,&gm,"e:\\tcplus\\bgi");
}
void create();
void drawcube(struct cube1 v[]);
void rotate();
void main()
{
int ch,tx,ty,tz,i,sx,sy,sz,m=9;
init();
do
{
flushall();
gotoxy(1,1);
cleardevice();
printf("MENU ");
printf("**********************");
printf("\n1.Create \n2.Translation\n3.Scaling\n4.Rotation\n5.Exit");
printf("\n\n Enter your choice:");
scanf("%d",&ch);
setcolor(m++);
switch(ch)
{
case 1:
cleardevice();
init();
create();
getch();
break;
case 2:
init();
flushall();
cleardevice();
printf("\n Enter Translation vactor:");
scanf("%d%d%d",&tx,&ty,&tz);
init();
drawcube(b);
for(i=0;i<8;i++)
{
t[i].x1=a[i].x1+tx;
t[i].y1=a[i].y1+ty;
t[i].z1=a[i].z1+tz;
```

```
}
for(i=0;i<8;i++)
{
c[i].x2=t[i].x1+t[i].z1/2;
c[i].y2=t[i].y1+t[i].z1/2;
}
drawcube(c);
break;
case 3:
cleardevice();
init();
flushall();
printf("Enter the scaling vector:");
scanf("%d%d%d",&sx,&sy,&sz);
init();
drawcube(b);
for(i=0;i<8;i++)
{
f[i].x1=a[i].x1*sx;
f[i].y1=a[i].y1*sy;
f[i].z1=a[i].z1*sz;
}
for(i=0;i<8;i++)
{
d[i].x2=f[i].x1+f[i].z1/2;
d[i].y2=f[i].y1+f[i].z1/2;
}
drawcube(d);
break;
case 4:
cleardevice();
init();
rotate();
break;
case 5:
closegraph();
exit(0);
default:
continue;
}}
while(1);
}
void create()
{
```

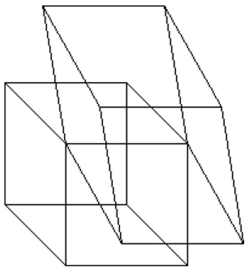
```
int d,i;
flushall();
printf("\n Enter co-ordinates:");
scanf("%d%d%d",&x,&y,&z);
printf("\n Enter the side of cube");
scanf("%d",&d);
cleardevice();
setcolor(9);
settextstyle(0,0,5);
outtextxy(10,30,"cube");
a[0].x1=x; a[0].y1=y; a[0].z1=z;
a[1].x1=x+d; a[1].y1=y; a[1].z1=z;
a[2].x1=x+d; a[2].y1=y+d; a[2].z1=z;
a[3].x1=x; a[3].y1=y+d; a[3].z1=z;
a[4].x1=x; a[4].y1=y; a[4].z1=z-d;
a[5].x1=x+d; a[5].y1=y; a[5].z1=z-d;
a[6].x1=x+d; a[6].y1=y+d; a[6].z1=z-d;
a[7].x1=x; a[7].y1=y+d; a[7].z1=z-d;
for(i=0;i<8;i++)
{
b[i].x2=a[i].x1+a[i].z1/2;
b[i].y2=a[i].y1+a[i].z1/2;
}
cleardevice();
drawcube(b);
getch();
}
void drawcube(struct cube1 v[])
{
gotoxy(3,4);
line(v[0].x2,v[0].y2,v[1].x2,v[1].y2);
line(v[1].x2,v[1].y2,v[2].x2,v[2].y2);
line(v[2].x2,v[2].y2,v[3].x2,v[3].y2);
line(v[3].x2,v[3].y2,v[0].x2,v[0].y2);
line(v[4].x2,v[4].y2,v[5].x2,v[5].y2);
line(v[5].x2,v[5].y2,v[6].x2,v[6].y2);
line(v[6].x2,v[6].y2,v[7].x2,v[7].y2);
line(v[7].x2,v[7].y2,v[7].x2,v[7].y2);
line(v[0].x2,v[0].y2,v[4].x2,v[4].y2);
line(v[1].x2,v[1].y2,v[5].x2,v[5].y2);
line(v[2].x2,v[2].y2,v[6].x2,v[6].y2);
line(v[3].x2,v[3].y2,v[7].x2,v[7].y2);
line(v[4].x2,v[4].y2,v[7].x2,v[7].y2);
getch();
```

```
delay(5);
}
void rotate()
{
cleardevice();
int cho,i,n;
float theta;
flushall();
printf("Rotation with respect to axis:");
printf("\n\t1.x_axis\n\t2.y_axis\n\t3.z_axis\n\t4.exit");
printf("\n Enter ur choice");
scanf("%d",&cho);
printf("\n Enter rotation angle");
scanf("%f",&theta);
theta=(theta)*180/3.14;
switch(cho)
{
case 1:
cleardevice();
init();
drawcube(b);
for(i=0;i<8;i++)
{
g[i].x1=a[i].x1;
g[i].y1=(a[i].y1-y)*cos(theta)+(a[i].z1-z)*sin(theta)+y;
g[i].z1=(a[i].y1-y)*sin(theta)+(a[i].z1-z)*cos(theta)+z;
}
for(i=0;i<8;i++)
{
e[i].x2=g[i].x1+g[i].z1/2;
e[i].y2=g[i].y1+g[i].z1/2;
}
drawcube(e);
settextstyle(0,0,8);
setcolor(14);
outtextxy(20,360,"stop");
getch();
cleardevice();
break;
case 2:
cleardevice();
init();
drawcube(b);
for(i=0;i<8;i++)
```

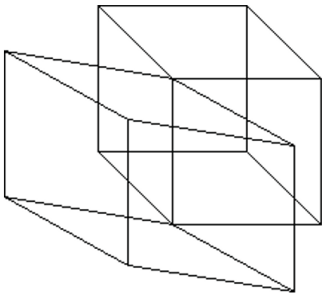
```
{
k[i].y1=(a[i].x1-x)*cos(theta)+(a[i].z1-z)*sin(theta)+x;
k[i].y1=a[i].y1;
k[i].y1=(a[i].x1-x)*sin(theta)+(a[i].z1-z)*cos(theta)+z;
}
for(i=0;i<8;i++)
{
h[i].x2=k[i].x1+k[i].z1/2;
h[i].y2=k[i].y1+k[i].z1/2;
}
drawcube(h);
settextstyle(0,0,8);
setcolor(14);
outtextxy(20,360,"stop");
getch();
cleardevice();
break;
case 3:
cleardevice();
init();
drawcube(b);
for(i=0;i<8;i++)
{
l[i].x1=(a[i].x1-x)*cos(theta)+(a[i].y1-y)*sin(theta)+x;
l[i].z1=(a[i].x1-x)*sin(theta)+(a[i].y1-y)*cos(theta)+y;
l[i].y1=a[i].z1;
}
for(i=0;i<8;i++)
{
j[i].x2=l[i].x1+l[i].z1/2;
j[i].y2=l[i].y1+l[i].z1/2;
}
drawcube(j);
cleardevice();
settextstyle(0,0,8);
setcolor(14);
outtextxy(20,360,"stop");
getch();
cleardevice();
break;
default:
puts("error");
}}
```

OUTPUT

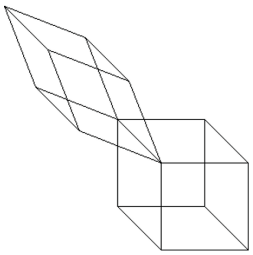
Menu
1. Create
2. Translation
3. Scaling
4. Rotation
5. Exit
Enter your choice : 4
Rotation with respect to axis
1. X-axis
2. Y-axis
3. Z-axis
4. Exit
Enter your Choice : 1
Enter the Rotation angle : 45



Menu
1. Create
2. Translation
3. Scaling
4. Rotation
5. Exit
Enter your choice : 4
Rotation with respect to axis
1. X-axis
2. Y-axis
3. Z-axis
4. Exit
Enter your Choice : 2
Enter the Rotation angle : 45



Menu
1. Create
2. Translation
3. Scaling
4. Rotation
5. Exit
Enter your choice : 4
Rotation with respect to axis
1. X-axis
2. Y-axis
3. Z-axis
4. Exit
Enter your Choice : 3
Enter the Rotation angle : 45



Menu
1. Create
2. Translation
3. Scaling
4. Rotation
5. Exit
Enter your choice : 5

RESULT

Thus, the 3D Transformations on a cube are implemented and the output is verified.

Ex. No. 8	GENERATING FRACTAL IMAGES

AIM

To generate fractal images using C program.

ALGORITHM

Step 1: Start.

Step 2: Initialise the necessary variable gd, gm.

Step 3: Create a function drawst_f() to draw the serpinski triangle.

Step 4: Using for loop, iterate the loop for 10000 times and put pixel is set to WHITE colour.

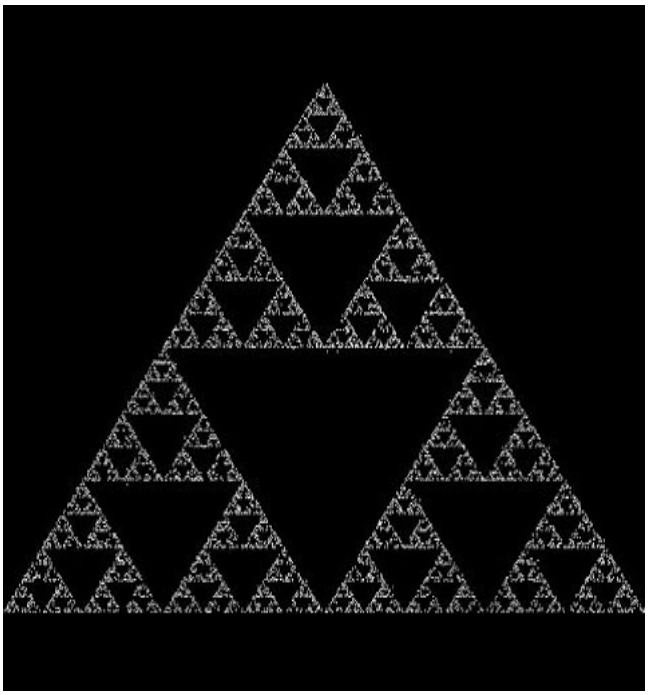
Step 5: Stop.

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
#include<stdlib.h>
void drawst_f()
{
char ch;
unsigned int x1, x2, y1, y2;
int i;
x1=x2=320;
y1=y2=0;
for(i=0;i<10000;i++)
{
ch=random(3);
if(ch==0)
{
x1=(x2+320)/3; y1=(y2+0)/2;
}
else if(ch==1)
{
x1=(x2+0)/2;
y1=(y2+480)/2;
}
else if(ch==2)
{
x1=(x2+640)/2;
y1=(y2+480)/2;
}
putpixel(x1,y1,WHITE);
x2=x1;
```

```
y2=y1;
}
}
void main()
{
int gd=DETECT, gm;
initgraph(&gd,&gm,"..//BGI");
drawst_f();
getch();
}
```

OUTPUT



RESULT

Thus, the above C program to generate fractal images is executed.

Ex. No. 9	DRAWING THREE DIMENSIONAL OBJECTS AND SCENES USING OPENGL

AIM

To generate 3D objects and scenes using OpenGL functions.

ALGORITHM

Step 1: Start the program.
Step 2: Include GL/glut.h.
Step 3: Initialize the glut functions.
Step 4: Enable the necessary GL-flags.
Step 5: Draw two icosahedrons.
Step 6: Apply rendering for icosahedrons.
Step 7: Apply blending effects to icosahedrons.
Step 8: Stop the program.

PROGRAM

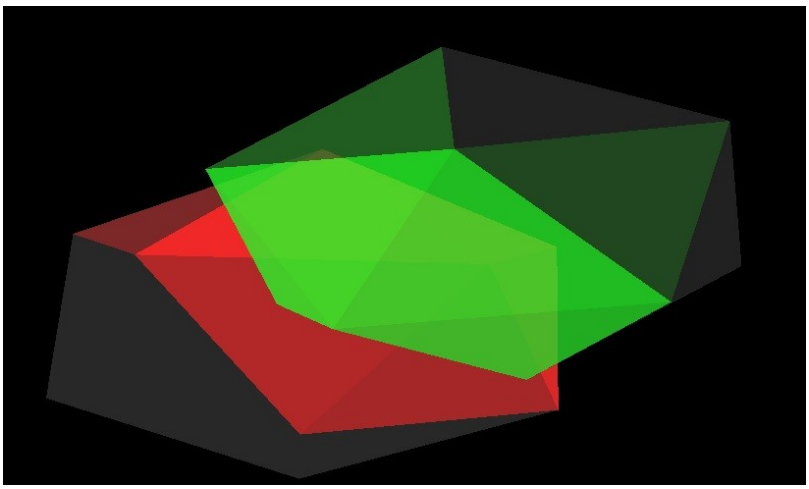
```
#include <GL/glut.h>
#include <stdio.h>
#include <math.h>
GLfloat light0_ambient[] ={0.2, 0.2, 0.2, 1.0};
GLfloat light0_diffuse[] ={0.0, 0.0, 0.0, 1.0};
GLfloat light1_diffuse[] ={1.0, 0.0, 0.0, 1.0};
GLfloat light1_position[] ={1.0, 1.0, 1.0, 0.0};
GLfloat light2_diffuse[] ={0.0, 1.0, 0.0, 1.0};
GLfloat light2_position[] ={-1.0, -1.0, 1.0, 0.0};
float s = 0.0;
GLfloat angle1 = 0.0, angle2 = 0.0;
void output(GLfloat x, GLfloat y, char *text)
{
char *p;
glPushMatrix();
glTranslatef(x, y, 0);
for (p = text; *p; p++)
glutStrokeCharacter(GLUT_STROKE_ROMAN, *p);
glPopMatrix();
}
Void display(void)
{
static GLfloat amb[] ={0.4, 0.4, 0.4, 0.0};
static GLfloat dif[] ={1.0, 1.0, 1.0, 0.0};
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glEnable(GL_LIGHT1);
glDisable(GL_LIGHT2);
```

```
amb[3] = dif[3] = cos(s) / 2.0 + 0.5;
glMaterialfv(GL_FRONT, GL_AMBIENT, amb);
glMaterialfv(GL_FRONT, GL_DIFFUSE, dif);
glPushMatrix();
glTranslatef(-0.3, -0.3, 0.0);
glRotatef(angle1, 1.0, 5.0, 0.0);
glCallList(1);
/* render ico display list */
glPopMatrix();
glClear(GL_DEPTH_BUFFER_BIT);
glEnable(GL_LIGHT2);
glDisable(GL_LIGHT1);
amb[3] = dif[3] = 0.5 - cos(s * .95) / 2.0;
glMaterialfv(GL_FRONT, GL_AMBIENT, amb);
glMaterialfv(GL_FRONT, GL_DIFFUSE, dif);
glPushMatrix();
glTranslatef(0.3, 0.3, 0.0);
glRotatef(angle2, 1.0, 0.0, 5.0);
glCallList(1);
/* render ico display list */
glPopMatrix();
glPushAttrib(GL_ENABLE_BIT);
glDisable(GL_DEPTH_TEST);
glDisable(GL_LIGHTING);
glMatrixMode(GL_PROJECTION);
glPushMatrix();
glLoadIdentity();
gluOrtho2D(0, 1500, 0, 1500);
glMatrixMode(GL_MODELVIEW);
glPushMatrix();
glLoadIdentity();
/* Rotate text slightly to help show jaggies. */
glRotatef(4, 0.0, 0.0, 1.0);
output(200, 225, "This is antialiased.");
glDisable(GL_LINE_SMOOTH);
glDisable(GL_BLEND);
output(160, 100, "This text is not.");
glPopMatrix();
glMatrixMode(GL_PROJECTION);
glPopMatrix();
glPopAttrib();
glMatrixMode(GL_MODELVIEW);
glutSwapBuffers();
}
```

```
void idle(void)
{
angle1 = (GLfloat) fmod(angle1 + 0.8, 360.0);
angle2 = (GLfloat) fmod(angle2 + 1.1, 360.0);
s += 0.05;
glutPostRedisplay();
}
void visible(int vis)
{
if (vis == GLUT_VISIBLE)
glutIdleFunc(idle); else
glutIdleFunc(NULL);
}
int main(int argc, char **argv)
{
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
glutCreateWindow("blender");
glutDisplayFunc(display);
glutVisibilityFunc(visible);
glNewList(1, GL_COMPILE);
/* create ico display list */
glutSolidIcosahedron();
glEndList();
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glLightfv(GL_LIGHT0, GL_AMBIENT, light0_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light0_diffuse);
glLightfv(GL_LIGHT1, GL_DIFFUSE, light1_diffuse);
glLightfv(GL_LIGHT1, GL_POSITION, light1_position);
glLightfv(GL_LIGHT2, GL_DIFFUSE, light2_diffuse);
glLightfv(GL_LIGHT2, GL_POSITION, light2_position);
glEnable(GL_DEPTH_TEST);
glEnable(GL_CULL_FACE);
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
glEnable(GL_LINE_SMOOTH);
glLineWidth(2.0);
glMatrixMode(GL_PROJECTION);
gluPerspective( /* field of view in degree */ 40.0,/* aspect ratio */ 1.0,/* Z near */ 1.0,
/* Z far*/ 10.0);
glMatrixMode(GL_MODELVIEW);
gluLookAt(0.0, 0.0, 5.0, /* eye is at (0,0,5) */0.0, 0.0, 0.0, /* center is at (0,0,0) */0.0,
1.0, 0.);
```

```
/* up is in positive Y direction */
glTranslatef(0.0, 0.6, -1.0);
glutMainLoop();
return 0;          /* ANSI C requires main to return int. */
}
```

OUTPUT



RESULT

Thus, the 3D objects and scenes are successfully generated using openGL functions.

Ex. No. 10	CREATING CHESS BOARD USING OPENGL

AIM

To create a Chess Board using OpenGL functions.

ALGORITHM

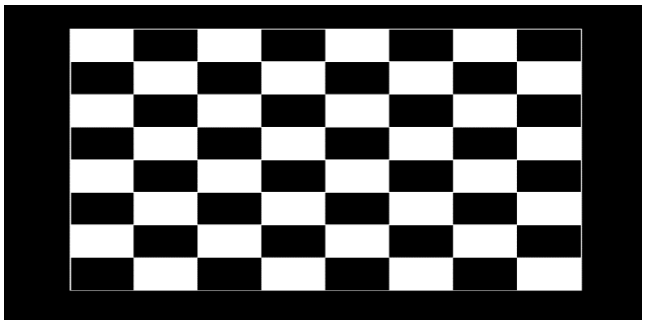
Step 1: Start the program.
Step 2: Include GL/glut.h.
Step 3: Initialize the glut functions.
Step 4: Enable the necessary GL-flags.
Step 5: Draw a square box.
Step 6: Repeat step5 to form squares in 8 rows and 8 columns constituting a chessboard.
Step 7: Stop.

PROGRAM

```
#include<windows.h>
#include<glut.h>
int c = 0;
void init(){
// For displaying the window
color glClearColor(0, 1, 1, 0);
// Choosing the type of projection
glMatrixMode(GL_PROJECTION);
// for setting the transformation which here is 2D
gluOrtho2D(0, 800, 0,600);
}
void drawSquare(GLint x1, GLint y1, GLint x2, GLint y2, GLint x3, GLint y3, GLint
x4, GLint y4){
// if color is 0 then draw white box and change value of color = 1
if (c == 0){
glColor3f(1, 1, 1); // white color value is 1 1 1
c = 1;
}
// if color is 1 then draw black box and change value of color = 0
else{
glColor3f(0, 0, 0); // black color value is 0 0 0
c = 0;
}
// Draw Square
glBegin(GL_POLYGON);
glVertex2i(x1, y1); glVertex2i(x2, y2);  glVertex2i(x3, y3); glVertex2i(x4, y4);
glEnd();
}
void chessboard()
```

```
{
glClear(GL_COLOR_BUFFER_BIT); // Clear display window
GLint x, y;
for (x = 0; x <= 800; x += 100)
{
for (y = 0; y <= 600; y += 75)
{
drawSquare(x, y + 75, x + 100, y + 75, x + 100, y, x, y);
}}
// Process all OpenGL routine s as quickly as possible
glFlush();
}
int main(int agrc, char ** argv)
{
// Initialize GLUT
glutInit(&agrc, argv);
// Set display mode
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
// Set top - left display window position.
glutInitWindowPosition(100, 100);
// Set display window width and height
glutInitWindowSize(800, 600);
// Create display window with the given title
glutCreateWindow("Chess Board using OpenGL in C++");
// Execute initialization procedure
init();
// Send graphics to display window
glutDisplayFunc(chessboard);
// Display everything and wait.
glutMainLoop();
}
```

OUTPUT



RESULT

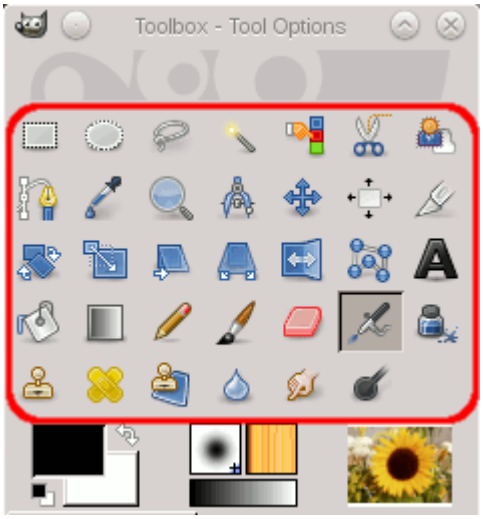
Thus, the program to create a chess board using OpenGL functions is executed successfully.

INTRODUCTION TO GIMP

GIMP provides a comprehensive toolbox in order to quickly perform basic tasks such as making selections or drawing paths. The many tools contained within GIMP's toolbox are discussed in detail here.

GIMP has a diverse assortment of tools that let you perform a large variety of tasks. The toolscan be thought of as falling into five categories:

- Selection tools, which specify or modify the portion of the image that will be affected by subsequent actions;
- Paint tools, which alter the colors in some part of the image;
- Transform tools, which alter the geometry of the image;
- Color tools, which alter the distribution of colors across the entire image;
- Other tools, which don't fall into the other four categories.



Most tools can be activated by clicking on an icon in the Toolbox. By default, some tools are accessible only via the menus (namely the Color tools are accessible only either as Colors or as Tools → Colors). Every tool, in fact, can be activated from the Tools menu; also, every tool can be activated from the keyboard using an accelerator key.

In the default setup, created when GIMP is first installed, not all tools show icons in the Toolbox: the Color tools are omitted. You can customize the set of tools that are shown in the Toolbox through Edit → Preferences → Toolbox. There are two reasons you might want to do this: first, if you only rarely use a tool, it might be easier to find the tools you want if the distracting icon is removed; second, if you use the Color tools a lot, you might find it convenient to have icons for them easily available. In any case, regardless of the Toolbox, you can always access any tool at any time using the Tools menu from an image menu bar.

The shape of the cursor changes when it is inside an image, to one that indicates which tool is active (if in Preferences you have set Image Windows → Mouse Pointers → Pointer mode → Tool icon).

Ex. No. 1	LOGO CREATION

AIM

To create a logo using Gimp tool.

ALGORITHM

Step 1: Open GIMP icon.

Step 2: Select file -> new.

Step 3: Set height, width and resolution of your logo.

Step 4: Select Bucket fill tool -> select pattern fill->select any pattern and click on your workspace to set background of your logo.

Step 5: Select text tool and type text for your logo.

Step 6: Select color icon to change color of your text.

Step 7: To insert an image select file -> open as layers and select your image and use scale tool to resize image.

Step 8: If you want to add more effects on your logo select ->filters ->select any options.

Step 9: After designing your logo select file ->export ->select file format and location and save your logo.

OUTPUT

RESULT

Thus, the logo is created successfully using GIMP.

Ex. No. 2	TEXT ANIMATION

AIM

To create text animation using GIMP.

ALGORITHM

Step 1: Open GIMP icon.

Step 2: Select file -> new.

Step 3: Set height, width and resolution of your workspace.

Step 4: Select Bucket fill tool -> select pattern fill->select any pattern and click on your workspace to set background.

Step 5: Select text tool and type texts.

Step 6: Select color icon to change color of your text.

Step 7: Select filter ->animation->select any option(blend, spinning globe, waves, rippling, burn-in) to animate text.

Step 8: Select filter ->play back.

Step 9: To save file->export->select file format->save. Applying spinning globe.

OUTPUT

RESULT

Thus, text animation is created successfully using GIMP.

INTRODUCTION TO AUDACITY

Control Tool Bar

Editing tools

Selection tool – For selecting the range of audio you want to edit or listento

Envelope tool – For changing the volume over driver

Draw tool – For modifying individual samples

Zoom tool– for zooming in and out.

Time shift tool – For sliding tracks left or right

Multi tool – lets you access all of these tools at once depending on thelocation of the mouse and the keys you are holding down.

Audio control button

Skip to Start – moves the cursor to time 0. If you press Play at this point, you project will play from the beginning.

Play – starts playing audio at the cursor position. If some audio is selected, only the selection is played.

Loop –if you hold down the Shift key, the Play button changes to a Loop button, which lets you keep playing the selection over

Record – starts recording audio at the project sample rate (the sample rate in the lower–left corner of the window). The new track begin at the current cursor position, so click the "Skip to Start" button first if you want thetrack to begin at time 0.

Pause – temporarily stops playback or recording until you press pause again.

Stop – stops recording or playing. You must do this before applying effects, saving or exporting.

Skip to End – moves the cursor to the end of the last track.

Mixer tool bar

The Mixer Toolbar has three controls, used to set the voh1me levels of your audio device and choose the input source. The leftmost slider controls the output voh1me, the other slider controls the recording voh1me, and the control on the right lets you choose the input source (such as "Microphone", "Line In", "Audio CD", etc.). Use the Record Level Meter to set the correct level

Changing these controls has no effect on the audio data in you· project - in other words it doesn't matter what the output volume level is when you Export or Save a project - the end result is the same.

Edit toolbar

All of the buttons on this toolbar perform action - and with couple of exceptions, they're on just shortcuts of existing menu items to save you

Holding the mouse over a tool will show a “tooltip” in case you forgot which one is which.

 Cut.

 Copy.

 Paste.

 Trim away the audio outside the selection.

 Silence the selected audio.

 Undo.

 Redo.

 Zoom In.

 Zoom out.

 Fit selection in window - zooms until the selection just fits inside the window.

 Fit project in window - zooms until all of the audio just fits inside the window.

Ex. No. 1	SILENCING TRIMMING AND DUPLICATING AUDIO SIGNAL

AIM

To apply silence, trim and duplicate effects for a selected audio signal.

ALGORITHM

A. Silencing

- Open Audacity and open a new audio file.
- Select a particular region to be silence and go to Edit -› Silence or Ctrl + L.
- To duplicate a particular part, select the part and go to Edit -›Copy and paste it in any place of the audio track.
- Play the track using the play button in the toolbar.
- Save the audio file.

B. Trimming

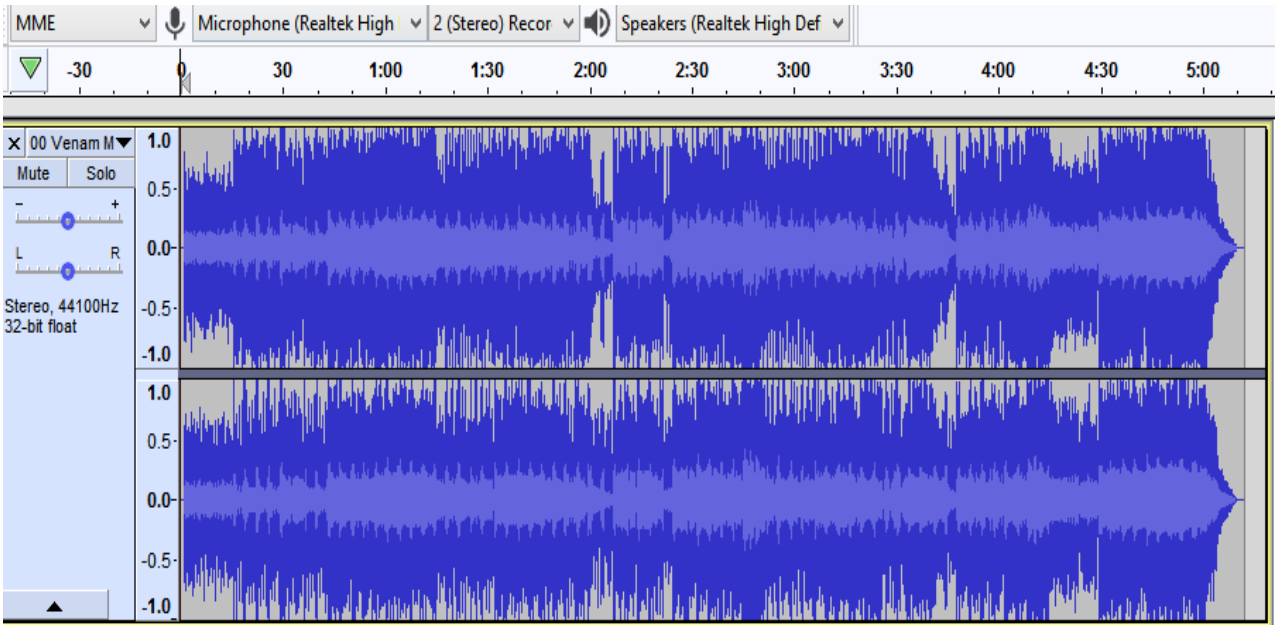
- Open Audacity and open a new audio file.
- To zoom in to get a closer look at the waveform, first choose the Selection Tool  , then click near the point you're interested in, then click the Zoom In button to see the detail you need.
- Select a particular region to be trimmed using button.
- Select  cut icon from the toolbar.
- Play the track using the play button in the toolbar.
- Save the audio signal.

C. Duplicating

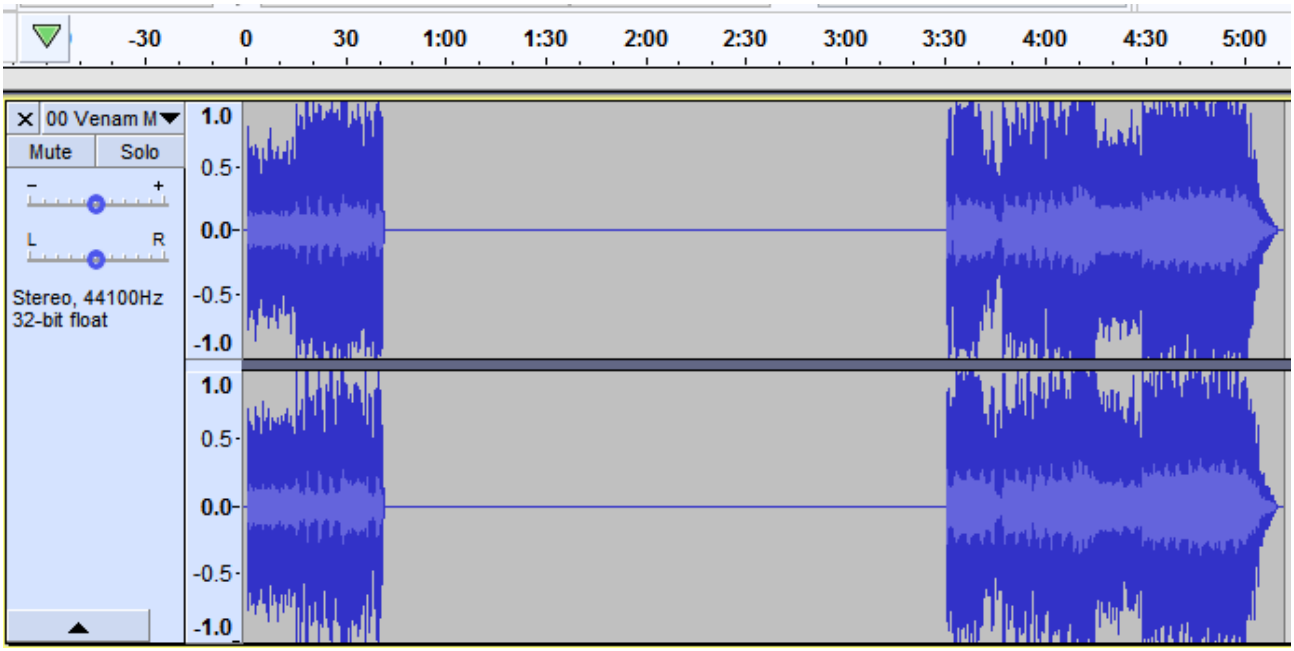
- Open Audacity and open a new audio file.
- To zoom in to get a closer look at the waveform, first choose the Selection Tool  , then click near the point you're interested in, then click the Zoom In button to see the detail you need.
- Select a particular region to be duplicated using  button.
- Play the track using the play button in the toolbar.
- Save the audio signal.

OUTPUT

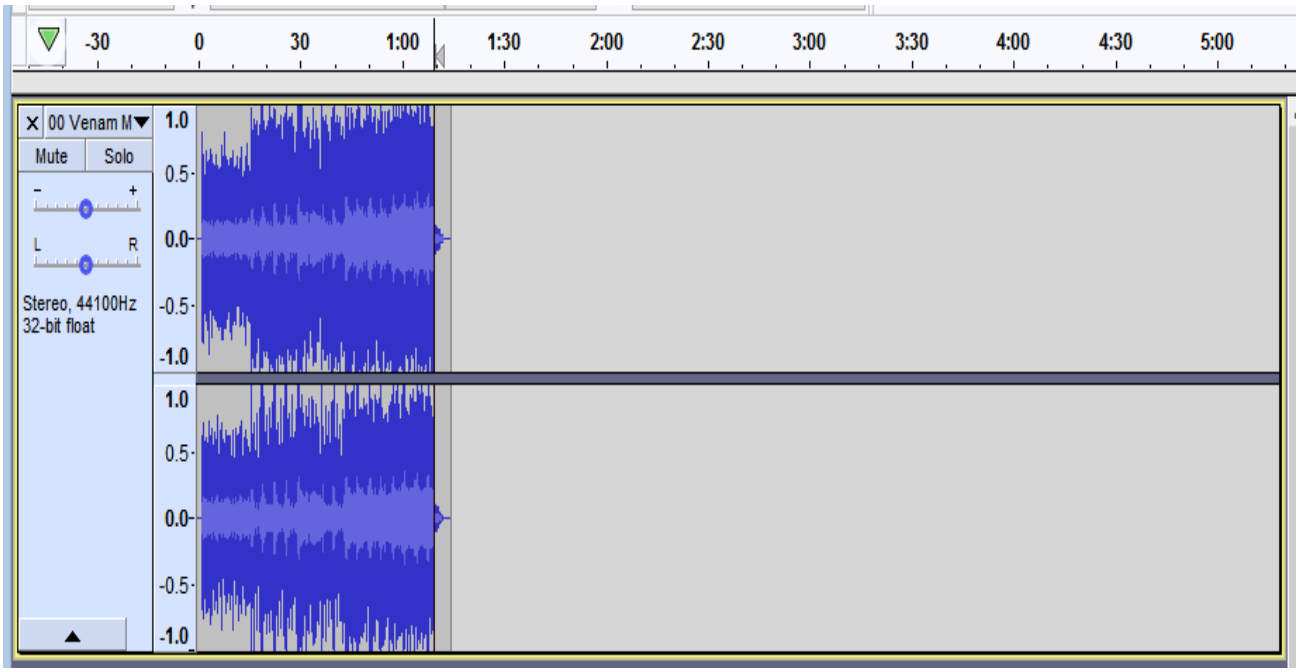
i) Original audio signal



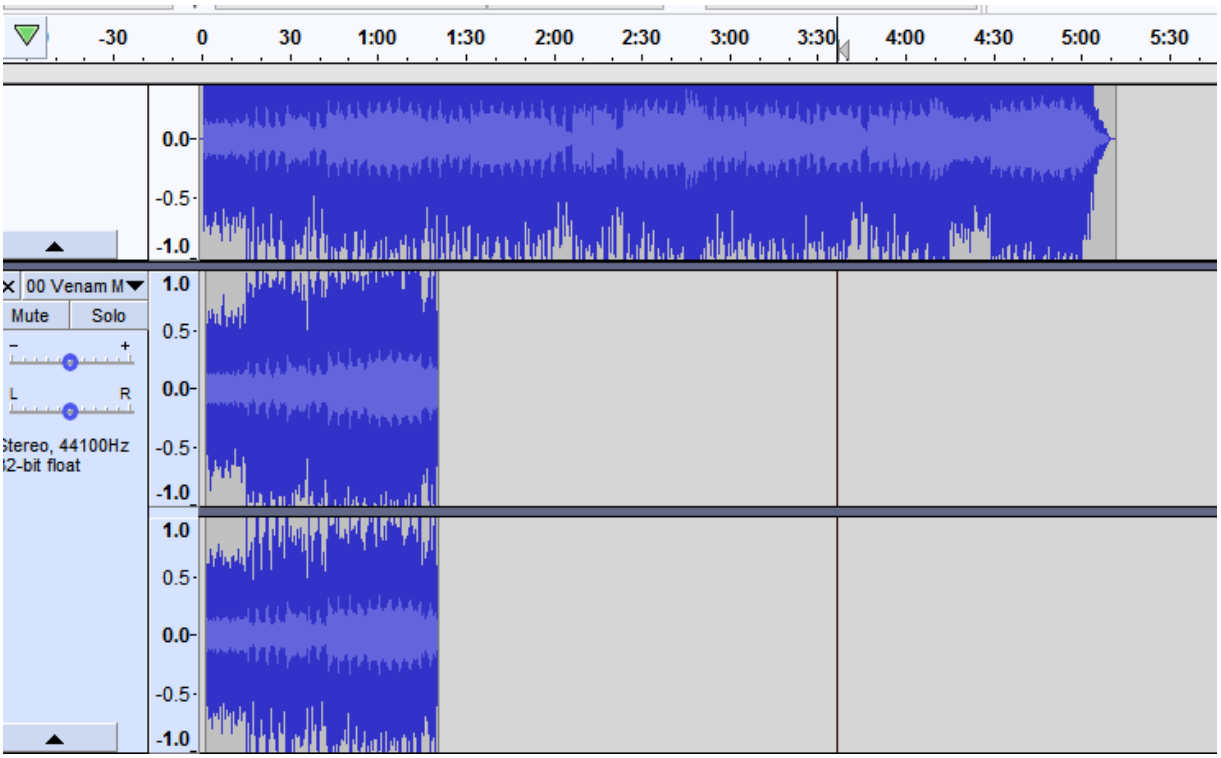
ii) After Silencing



iii) After Trimming



iv) Duplicating



RESULT

Thus, an audio source has been duplicated and silenced at a particular part using Audacity.

Ex. No. 2	ADVANCE EFFECTS TO AUDIO SIGNAL

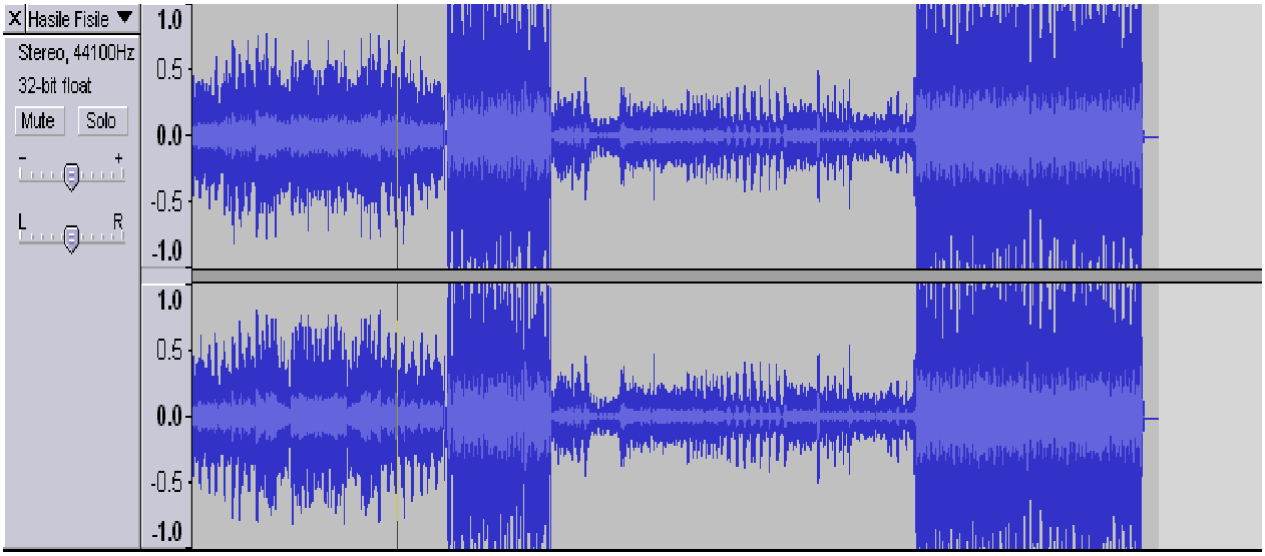
AIM

To apply various audio transitions to an audio signal.

ALGORITHM

- Open Audacity and open a new audio track.
- Select a part of it and go to Effect-›Reverse. Play to hear the effect.
- Select another part of the track and go to Effect -› WahWah and set the value as required. Play to hear the effect.
- Apply various effects of user defined.
- Audio effects have now been created.

OUTPUT



RESULT

Thus, various effects have been applied to an audio signal.

WINDOWS MOVIE MAKER

Ex. No. 1	APPLYING EFFECTS TO VIDEO

AIM

To apply some effects to video using windows movie maker.

ALGORITHM

Step 1: Open movie maker

Step 2: To add video and images select add videos and photos icon.

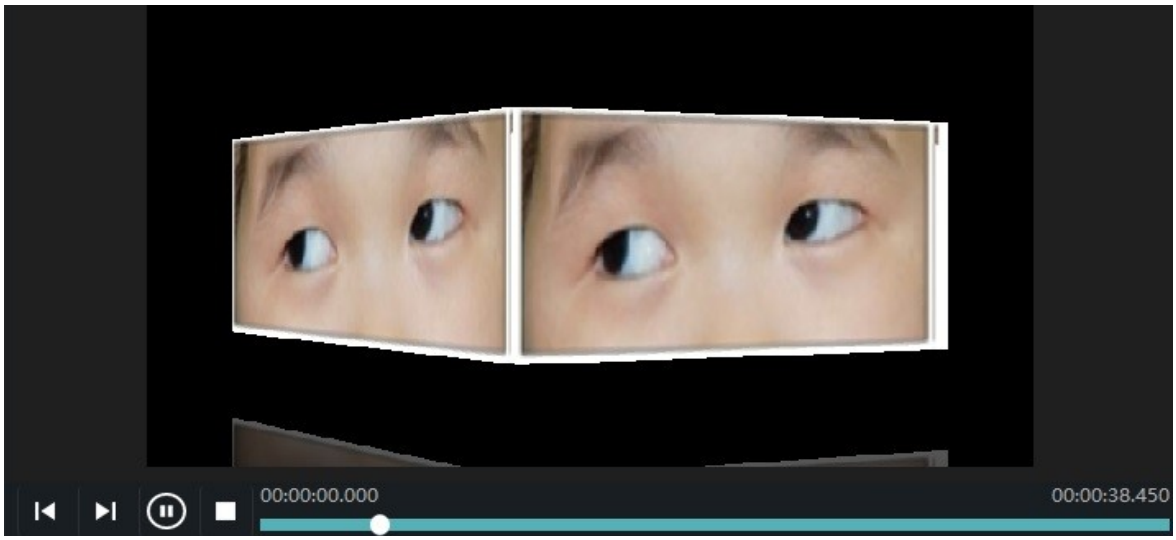
Step 3: To add music select add music icon.

Step 4: Your selected images/videos and music displayed on bottom of your window. here you can manage your movie.

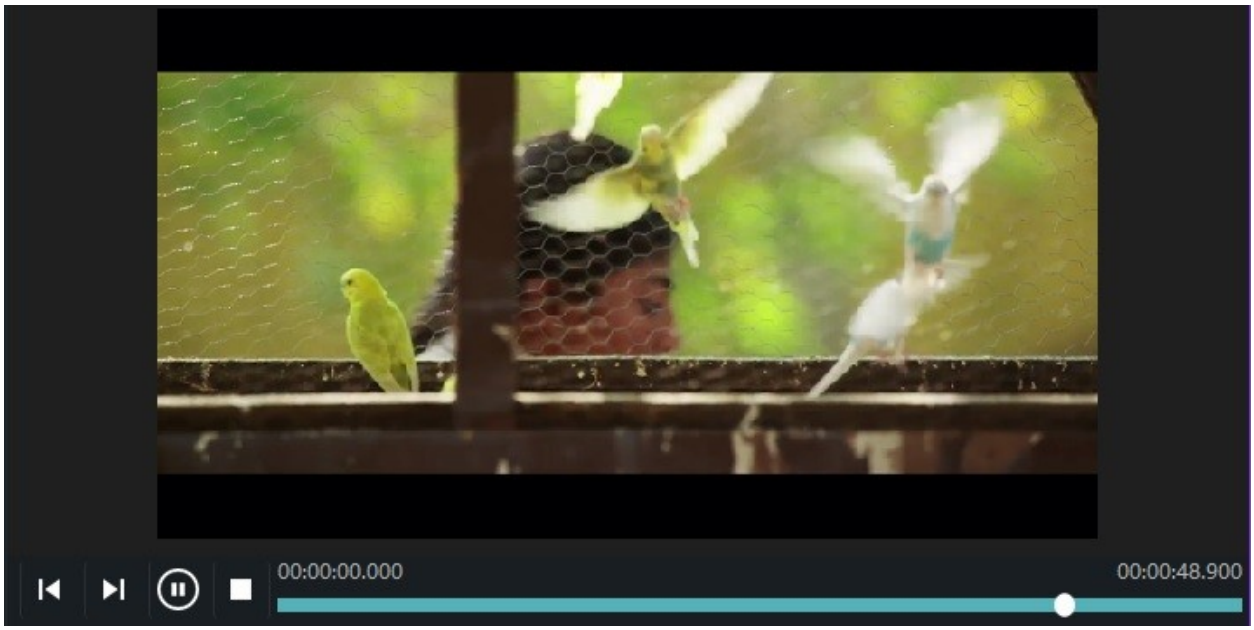
Step 5: To apply effects click animation and visual effects icon.

Step 6: After applying all effects select export to save.

OUTPUT



Original video



After applying grayscale effect

RESULT

Thus, the effect to video is implemented successfully using windows movie maker.

Ex. No. 2	CREATING TITLES IN VIDEO

AIM

To create titles in videos using windows movie maker.

ALGORITHM

Step 1: Open movie maker

Step 2: To add video and images select add videos and photos icon.

Step 3: To add effects on your video refer the procedure of previous exercise.

Step 4: Select text/credit icon->select title->add title for your movie

Step 5: To save click export from file and name your movie and then save.

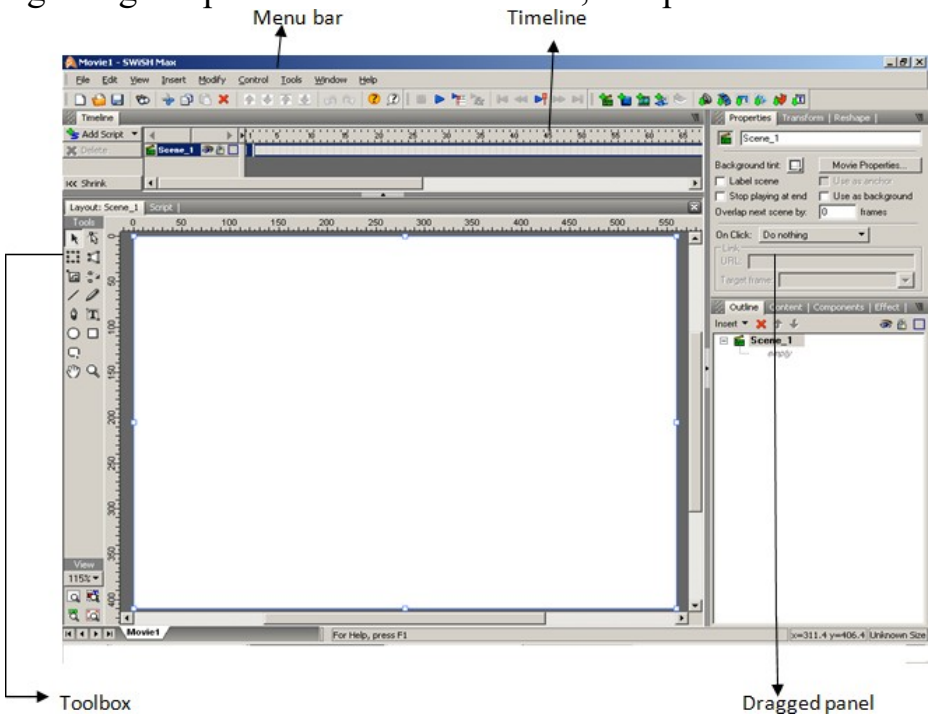
OUTPUT

RESULT

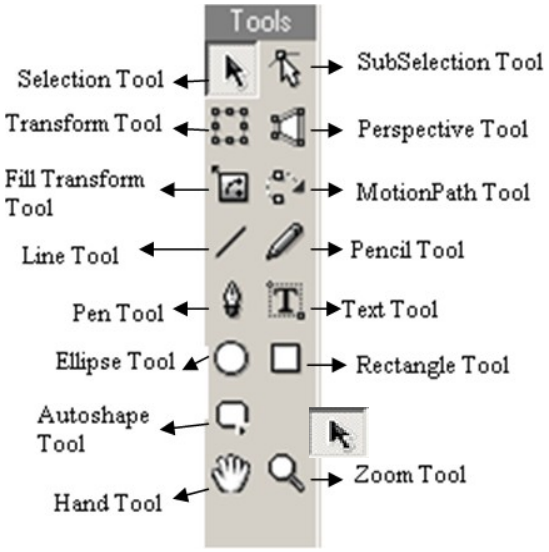
Thus, titles in videos is created successfully using windows movie maker.

INTRODUCTION TO SWISH MAX 2

Swish Max 2 is easy to use and produces complex animations with text, images, graphics, video and sound. Swish Max has tools for creating lines, rectangles, ellipses, vector and freehand curves, motion paths, movie clips, rollover buttons, and input forms all in an intuitive easy-to-use interface. Swish Max also includes a large range of preset multimedia effects, components and vector art.



Tools



Tools are modal commands that determine what Swish Max does when youclick and drag the mouse on the workspace. You can select a tool by clicking one of the options in the Tool box, which is located in thetop-left of the Layout panel. You can onlyselect one tool at a time.

Selection Tool (V)

The selection tool is used for selecting entire Objects and transforming graphic objects by moving or reshaping the object.

Subselection Tool  **(A)**

It reshapes graphic Objects by clicking and dragging the vertices the object outline path.

Transform Tool  **(Q)**

It transforms graphic Objects by clicking and dragging.

Perspective Tool (I) 

Dragging the handles provided modifies the Perspective of the Object.

Fill Transform (F) 

It transforms an Object's gradient or image fill without transforming the Object.

Motion Path (M) 

It plots a Motion Path for the selected Object by clicking and dragging.

Line Tool (L) 

It draws a line.

Pencil Tool (N) 

It draws a freehand line object.

Pen Tool (P) 

The Pen tool draws a set of connected curves or line segments.

Text Tool (T) 

The Text tool draws a Text Object.

Ellipse Tool (E) 

It draws an ellipse or circle.

Rectangle Tool (R) 

It draws a rectangle or square.

AutoShapes 

AutoShapes allows you to draw a number of predetermined shapes.

Hand Tool (H) 

Pans around the workspace in the Layout pannel by clicking and dragging.

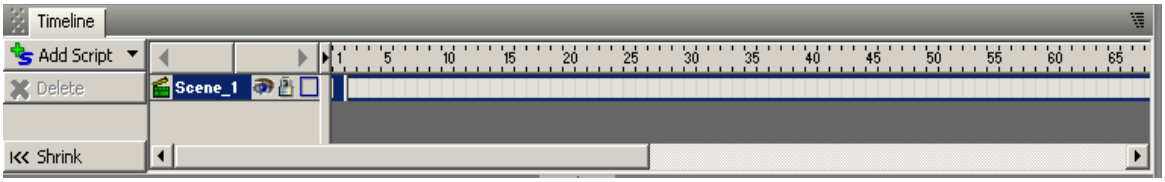
Zoom Tool (Z) 

It Zooms in/out of the workspace in the Layout pannel using the mouse.

Timeline

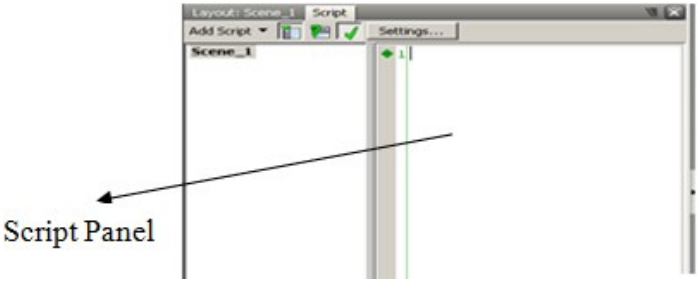
The 'Timeline' Panel contains time-based properties for the current Scene. The Scene is made up of a series of Frames, in the same way that a motion picture is made up of Frames. The Timeline is a visual representation of the Frames with the first Frame at the left and last Frame at the right

Scene: A Scene is a collection of Objects that are animated over a number of Frames.



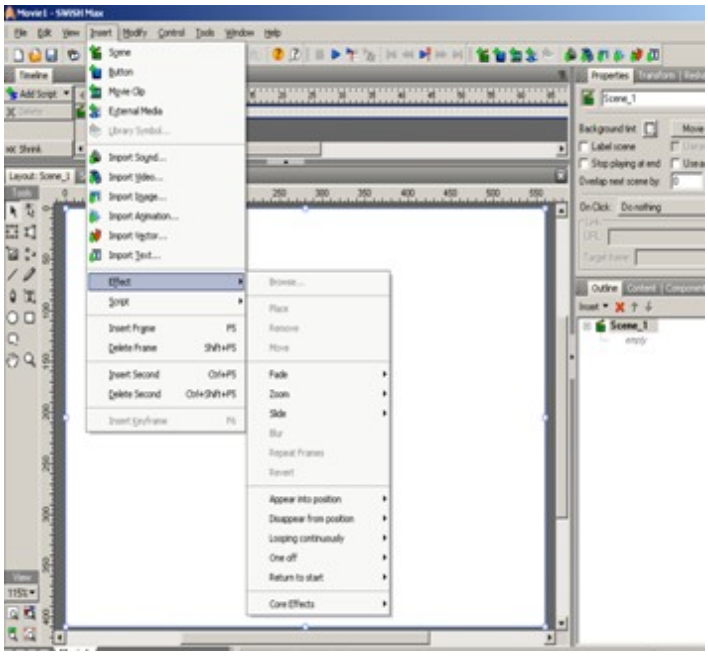
Scripting

Scripting can be used to define Actions that will occur at a specific Frame, when two Objects collide, or when someother external input is applied to the Movie.



Effects

Effects are animations that change the appearance of an Object over time. Youcan add, modify and coordinate Effects using the Timeline panel.

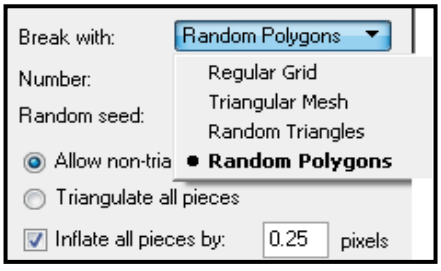


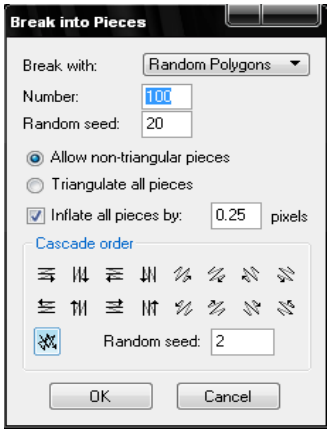
Events

All Actions are triggered in response to an Event. Any Scripting object can have events associated with it. When an event occurs the Event handling routine executes any Actions that are defined for that Event.

There are three types of events:

- Frame
- Button
- Self





Ex. No. 1	TEXT EFFECTS

AIM

To apply various text effects to the predefined formats.

ALGORITHM

- Create a Static text object and place it on the Stage.
- In the Outline Panel, right-click on the text object and select Copy Object.
- Then, use the Edit menu and select Paste in Place. You should now have two text objects at the exact same location on the stage.
- Select the bottom text object and use Modify -› Break -› Break into pieces.
- In the Break into Pieces settings window, use the drop-down menu next to
- Break With: ′ and select Random Polygons.
- Allow non-triangular pieces or Triangulate all pieces… again, this is up to you as it is difficult to tell any difference at such small sizes.
- Inflate all pieces by 0.25 pixels. This helps seal the scenes in between in each piece.
- When SWiSHmax has finished breaking the object into pieces, you should end up with a Group.
- In the Timeline Panel, right-click at Frame 1 on the row for the top text object and select the Place effect.
- Next, right-click at Frame 1 on the row for the Group object and select the
- Remove effect.
- Next, right-click at Frame 20 on the row for the top text object and select the
- Remove effect.
- Next, right-click at Frame 20 on the row for the Group object and select
- Disappear from Position | Wild – Splatter.

OUTPUT

RESULT

Thus, the dust effect has been applied to the custom text using Swish.

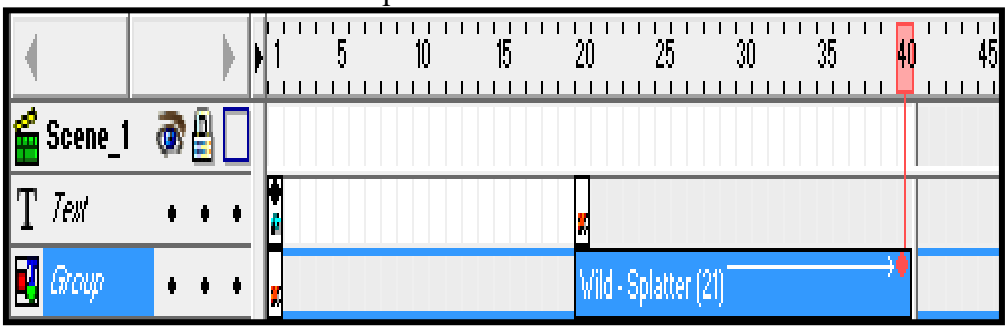
Ex. No. 2	PRE - LOADER

AIM

To create a pre - loader effect manually using Swish.

ALGORITHM

- Create a Dynamic TextBox name it as myMsgObj and name its variable myMsg.
- Create a rectangle for 20x20 pixels.
- Then convert it into movie clip.



- Name the movie clip as ‘ progBar ’.
- Then type the following coding on Script -> Scene_1:
- onFrame(1)
- {
- stop();
- }
- onSelfEvent(enterFrame)
- {
- pLoaded=this.percentLoaded(); tBytes=this.getBytesTotal(); myMsg=pLoaded+"%Loaded of "+tBytes+" bytes"; progBar_width=pLoaded;
- if(pLoaded==100) play();
- }
- The preloader is added.

OUTPUT

100% Loaded of 19861 bytes

RESULT

Thus, the pre-loader effect has been created using Swish.

FLASH

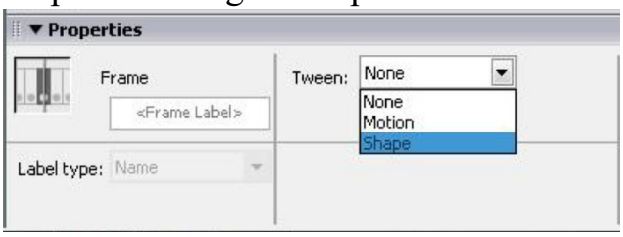
Ex. No. 1	CHANGING THE SHAPE OF THE OBJECT

AIM

To create Shape Tweening using Flash.

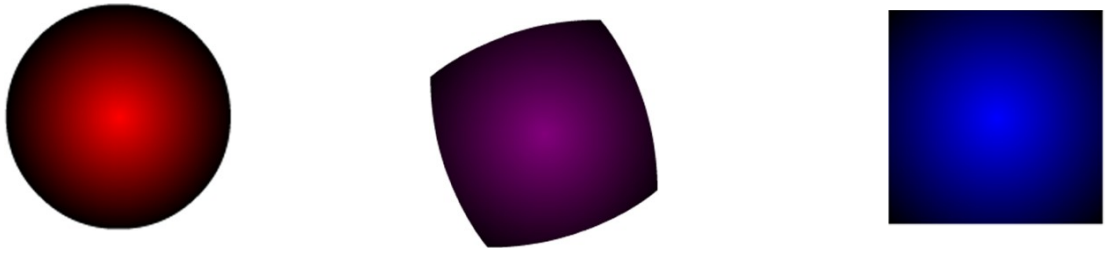
ALGORITHM

1. Create a New Flash Document.
2. Click the Oval Tool on the Tools.
3. Using the Fill Color Option select the attractive color to the Shape.
4. Create a Circle Object on Frame 1.
5. Click the 40th Frame and press F6.
6. Now Delete the Circle Object then Create a new Rectangle Object by using Rectangle Tool. Then change the color of the Rectangle Object using Fill Color.
7. Select the 20th Frame on Timeline, then Change the Tween option to the Shape on the Properties window.
8. Now Shape Tweening is completed. Press Ctrl + Enter.



OUTPUT

Shape tweening



RESULT

Thus, the shape tweening has been created using Flash.

Ex. No. 2	IMAGE VIEWING USING MASK

AIM

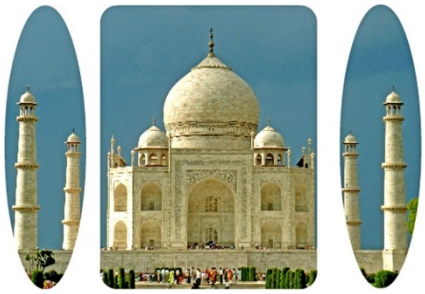
To create Masking using Flash.

ALGORITHM

- Create a New Flash Document.
- Import any Image on Frame 1.
- Create Motion Tweening for that Image. So that the Image can move from left to right side of the screen.
- Insert keyFrame.
- Now Lock the Layer 1 using Lock.
- Insert another Layer 2, and create any Objects (Ovel/ Rectangle/Any Shapes) for Object Masking , Or Type any Text with different Font & size for Text Masking.
- Right click on Layer 2 then select Mask.
- Now the Masking is created. Run the animation using Ctrl + Enter.

OUTPUT

Object Masking…



Text Masking…

RESULT

Thus, the Masking effecting has been created using Flash.

PHOTO IMPACT

Ex. No. 1	TEXT EFFECTS

AIM

To create custom effects to the pre defined font styles.

ALGORITHM

- Open a new image file.
- Create any text with user defined font style and size.
- Select the text and Select Object -> convert Object Type ->From Text/Image to Path.
- Switch to the Path Edit Tool  on the Tool Panel, you will then see from the wireframe structure that the text has become a path and can be
- edited as such.
- Use the dots to create custom design.

OUTPUT

Before:

After:

RESULT

Thus, custom effect has been applied to the pre defined font using Ulead Photo Impact.

Ex. No. 2	IMAGE SLICING

AIM

To slice a portion from a picture and merge it into another using Ulead Photo Impact.

ALGORITHM

- Open an image from which a specific portion has to be sliced.
- Select the image and go to Object -> Extract Object.
- By custom brush size and color, sketch out the portion to be parted and click next.
- Click extract and select the portion which should not to be sliced and click next.
- Adjust transparency and other details and click next.
- Now the portion to be sliced alone is selected.
- Import and background image and drag the selected image to the background image.
- Now the user defined image is ready.

OUTPUT

BEFORE

AFTER

RESULT

Thus, photo slicing and merging has been done using Ulead Photo Impact.