Lab Assignment - 01

AIM :- To study basics of computer graphics & its functions .Also discuss application of computer graphics .

INTRODUCTION

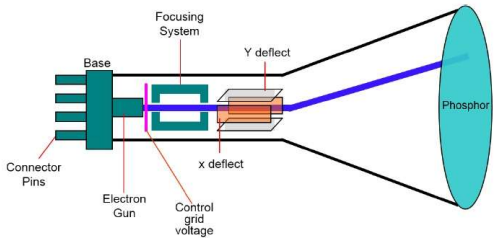
Computer graphics is an art of drawing pictures on computer screens with the help of programming. It involves computations, creation, and manipulation of data. In other words, we can say that computer graphics is a rendering tool for the generation and manipulation of images.

Cathode Ray Tube

The primary output device in a graphical system is the video monitor. The main element of a video monitor is the **Cathode Ray Tube (CRT),** shown in the following illustration.

The operation of CRT is very simple −

The electron gun emits a beam of electrons (cathode rays).
The electron beam passes through focusing and deflection systems that direct it towards specific positions on the phosphor-coated screen.
When the beam hits the screen, the phosphor emits a small spot of light at each position contacted by the electron beam.
It redraws the picture by directing the electron beam back over the same screen points quickly.



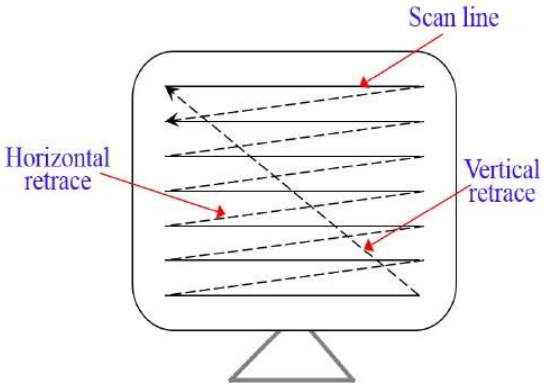
There are two ways (Random scan and Raster scan) by which we can display an object on the screen.

Raster Scan

In a raster scan system, the electron beam is swept across the screen, one row at a time from top to bottom. As the electron beam moves across each row, the beam intensity is turned on and off to create a pattern of illuminated spots.

Picture definition is stored in memory area called the **Refresh Buffer** or **Frame Buffer**. This memory area holds the set of intensity values for all the screen points. Stored intensity values are then retrieved from the refresh buffer and “painted” on the screen one row (scan line) at a time as shown in the following illustration.

Each screen point is referred to as a **pixel (picture element)** or **pel**. At the end of each scan line, the electron beam returns to the left side of the screen to begin displaying the next scan line.

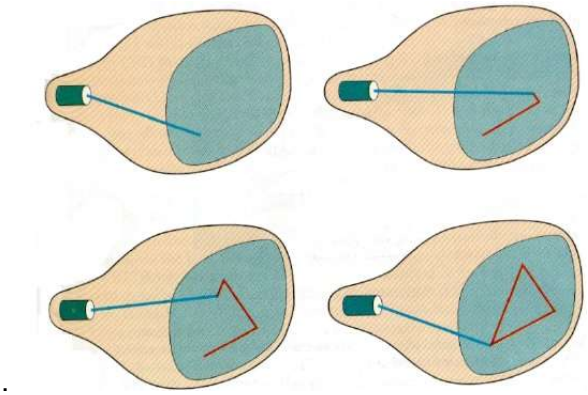


Random Scan (Vector Scan)

In this technique, the electron beam is directed only to the part of the screen where the picture is to be drawn rather than scanning from left to right and top to bottom as in raster scan. It is also called **vector display, stroke-writing display,** or **calligraphic display**.

Picture definition is stored as a set of line-drawing commands in an area of memory referred to as the **refresh display file**. To display a specified picture, the system cycles through the set of commands in the display file, drawing each component line in turn. After all the line-drawing commands are processed, the system cycles back to the first line command in the list.

Random-scan displays are designed to draw all the component lines of a picture 30 to 60 times each second



.

Application of Computer Graphics

Computer Graphics has numerous applications, some of which are listed below −

Computer graphics user interfaces (GUIs) − A graphic, mouse-oriented paradigm which allows the user to interact with a computer.
Business presentation graphics − "A picture is worth a thousand words".
Cartography − Drawing maps.
Weather Maps − Real-time mapping, symbolic representations.
Satellite Imaging − Geodesic images.
Photo Enhancement − Sharpening blurred photos.
Medical imaging − MRIs, CAT scans, etc. - Non-invasive internal examination.
Engineering drawings − mechanical, electrical, civil, etc. - Replacing the blueprints of the past.
Typography − The use of character images in publishing - replacing the hard type of the past.
Architecture − Construction plans, exterior sketches - replacing the blueprints and hand drawings of the past.
Art − Computers provide a new medium for artists.
Training − Flight simulators, computer aided instruction, etc.
Entertainment − Movies and games.
Simulation and modeling − Replacing physical modeling and enactments

Graphics.h

C graphics using graphics.h functions or WinBGIM (Windows 7) can be used to draw different shapes, display text in different fonts, change colors and many more. Using functions of graphics.h in Turbo C compiler you can make graphics programs, animations, projects, and games. You can draw circles, lines, rectangles, bars and many other geometrical figures. You can change their colors using the available functions and fill them. Following is a list of functions

of graphics.h header file. Every function is discussed with the arguments it needs, its description, possible errors while using that function and a sample C graphics program with its output.

initgraph

void initgraph(int *graphdriver, int *graphmode, char *pathtodriver);
initgraph initializes the graphics system by loading a graphics driver from disk (or validating a registered driver), and putting the system into graphics mode.To start the graphics system, first call the initgraph function. initgraph loads the graphics driver and puts the system into graphics mode.

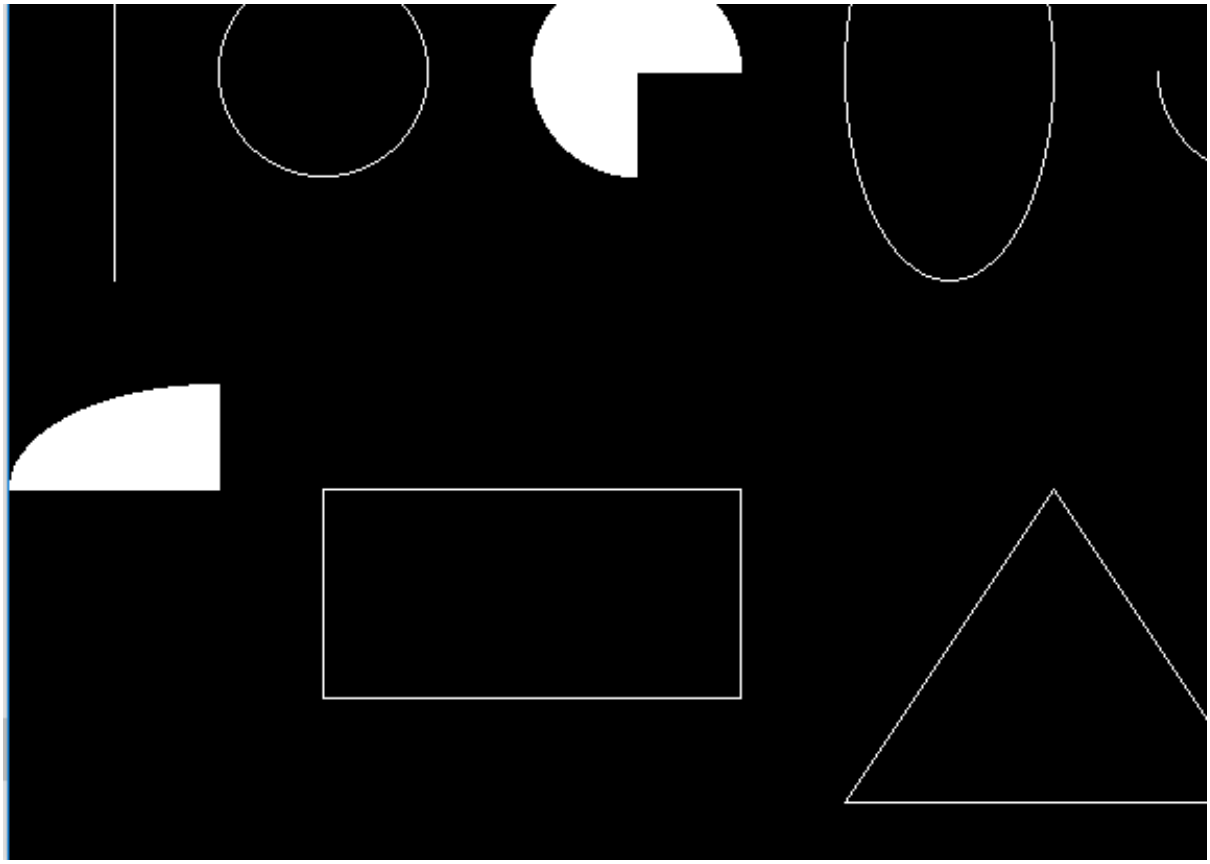
Program- 01

AIM :- Write a program in C++ to draw following Shapes Circle,Ellipse,Arc,Sector,Pie Slice,Line,Rectangle,Triangle .

Program:-

```
#include<conio.h>
#include<graphics.h>   `
void main() {
        clrscr();
        int gd = DETECT, gm;
        initgraph( & gd, & gm, "C:\\TURBOC3\\BGI");
        line(50, 50, 50, 200);
        circle(150, 100, 50);
        pieslice(300, 100, 0, 270, 50);
        ellipse(450, 100, 0, 360, 50, 100);
        arc(600, 100, 180, 270, 50);
        sector(100, 300, 90, 180, 100, 50);
        rectangle(150, 300, 350, 400);
        int poly[] = {
        400,
        450,
        600,
        450,
        500,
        300,
        400,
        450
        };
        drawpoly(4, poly);
        getch();
}
```

Output :-



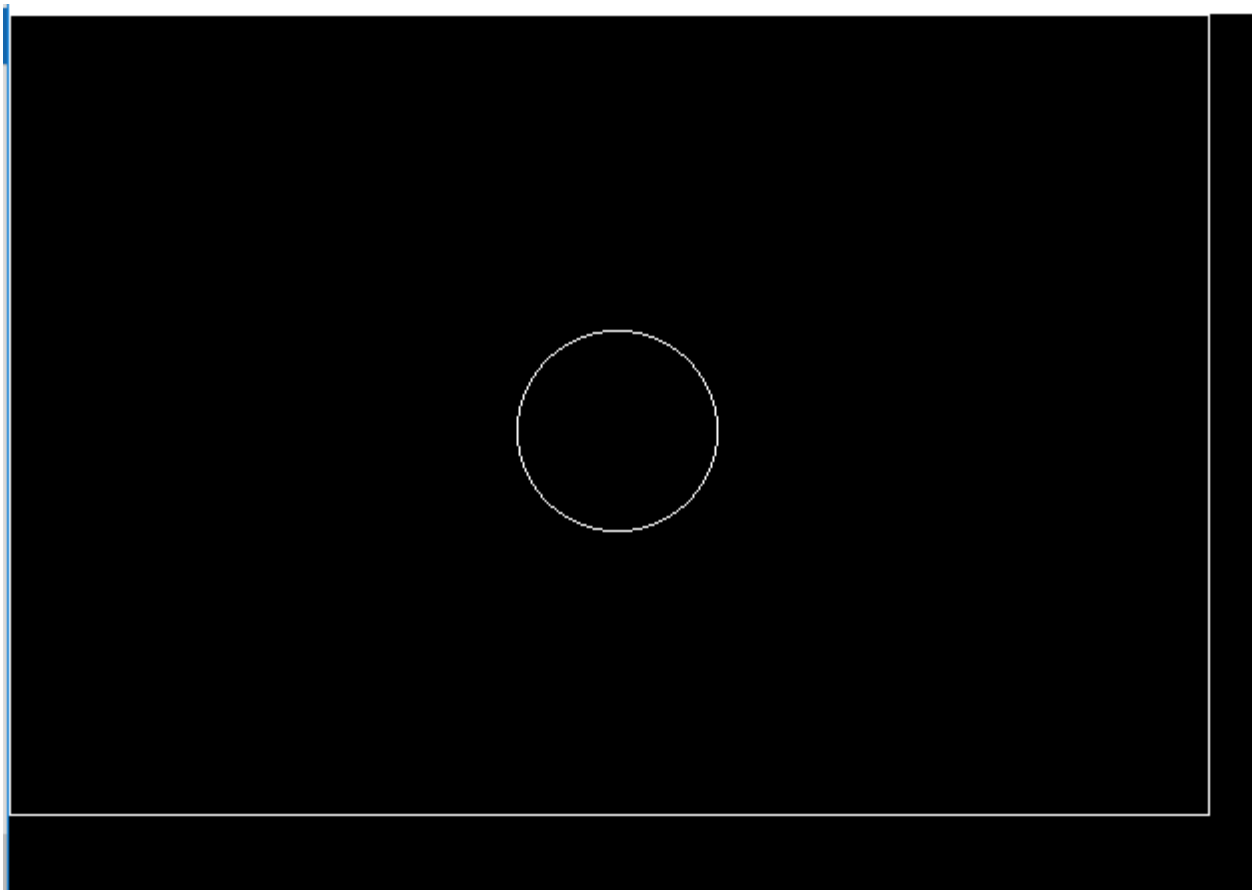
Program- 03

AIM :- Write a Program to Draw a bouncing Ball.

Program:-

```
#include<conio.h>
#include<graphics.h>  `
#include<dos.h>
void main() {
        int i = 0, x = 200, y = 200, xa = 2, ya = -2;
        int gd = DETECT, gm;
        initgraph( & gd, & gm, "C:\\TURBOC3\\BGI");
        for (i = 0; i < 300; i++) {
        cleardevice();
        rectangle(0, 0, 600, 400);
        if (x > (600 - (50)) || x < (50)) {
        xa = -xa;
        }
        if (y > (400 - (50)) || y < 50) {
        ya = -ya;
        }
        y = y + ya;
        x = x + xa;
        ellipse(x, y, 0, 360, 50, 50);
        delay(30);
        }
        getch();
}
```

Output:-



Program- 04

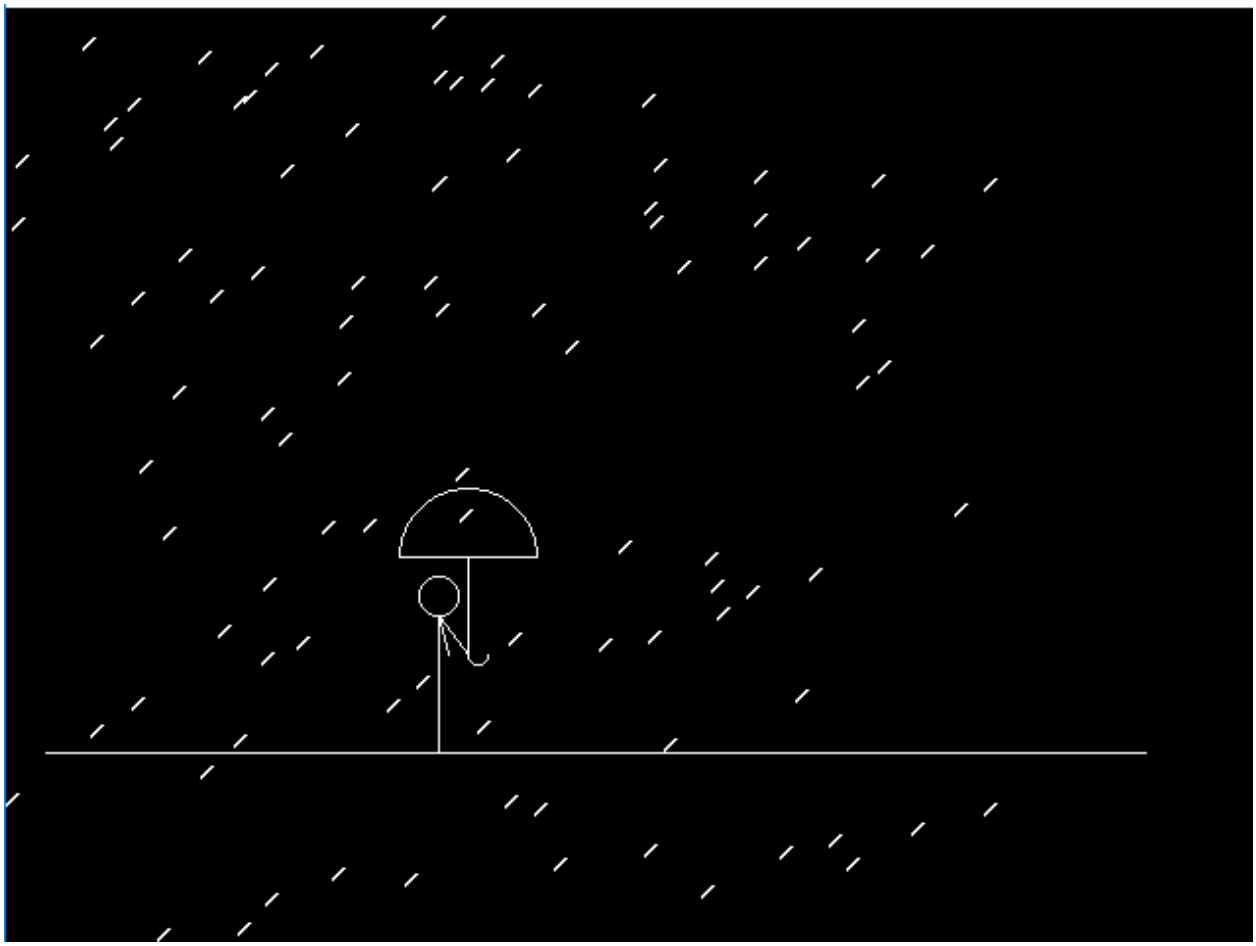
AIM :- Write a Program to Draw a Human Moving With Umbrella in Rain.

Program:-

```
#include<conio.h>
#include<graphics.h>
#include<stdlib.h>
#include<dos.h>
void main() {
        int gd = DETECT, gm;
        int j, i, k, rx, ry;
        clrscr();
        initgraph( & gd, & gm, "C:\\TURBOC3\\BGI");
        for (i = 0; i < 500; i += 5) {
        line(20, 380, 580, 380);
        if (i % 2 == 0) {
        line(25 + i, 380, 35 + i, 340);
        line(45 + i, 380, 35 + i, 340);
        line(35 + i, 310, 25 + i, 330);
        delay(20);
        } else {
        line(35 + i, 380, 35 + i, 340);
        line(35 + i, 310, 40 + i, 330);
        delay(20);
        }
        line(35 + i, 340, 35 + i, 310);
        circle(35 + i, 300, 10);
        line(35 + i, 310, 50 + i, 330);
        line(50 + i, 330, 50 + i, 280);
        line(15 + i, 280, 85 + i, 280);
        arc(50 + i, 280, 0, 180, 35);
        arc(55 + i, 330, 180, 360, 5);
        for (j = 0; j < 100; j++) {
        outtextxy(random(500), random(500), "/");
        }
        delay(150);
        cleardevice();
        }
```

```
        getch();}
```

Output:-



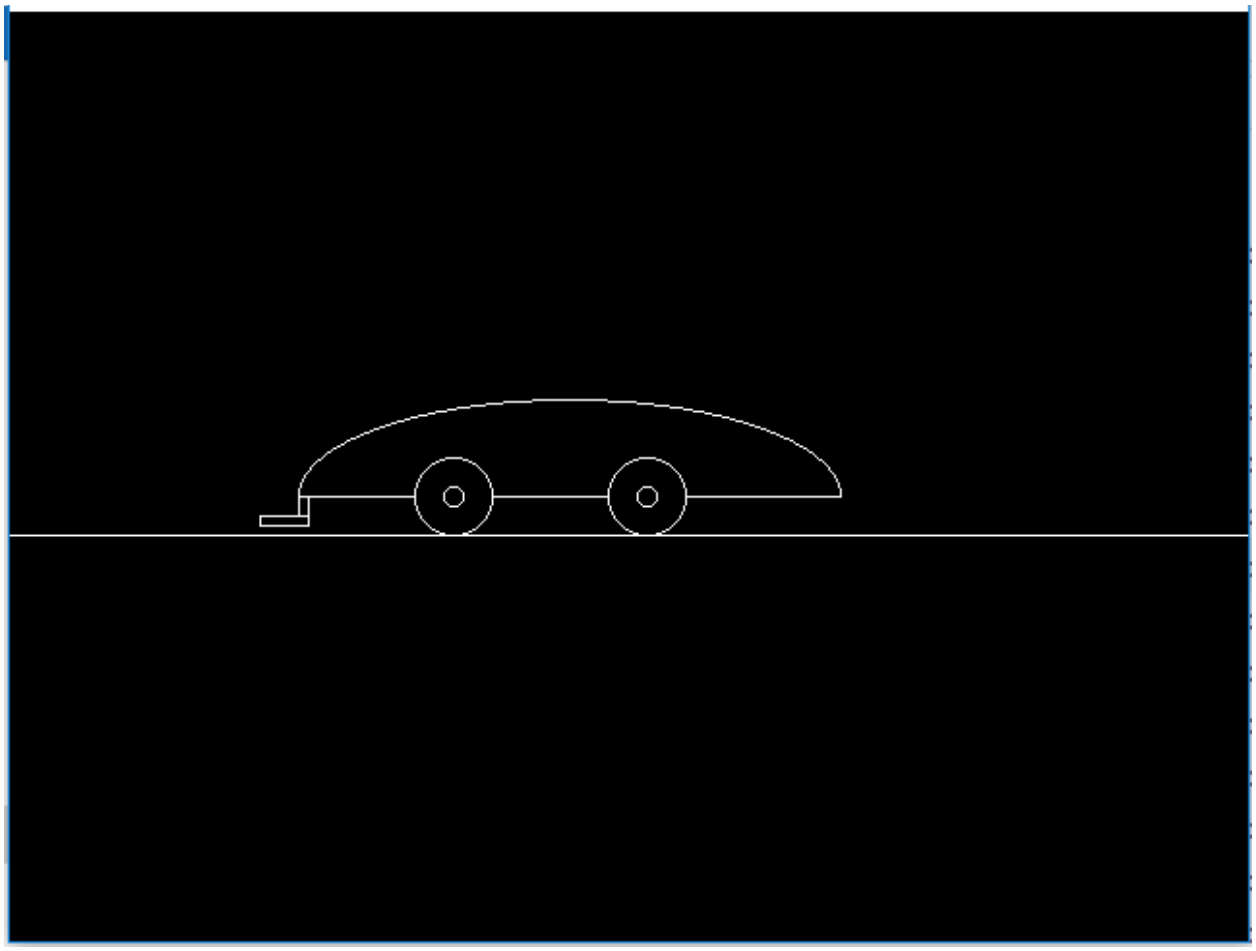
Program- 02

AIM :- Write a Program to Draw a moving Car.

Program:-

```
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void main() {
        clrscr();
        int gd = DETECT, gm, i;
        initgraph( & gd, & gm, "C:\\TURBOC3\\BGI");
        for (i = 0; i < 300; i++) {
        cleardevice();
        circle(100 + i, 250, 20);
        circle(200 + i, 250, 20);
        circle(100 + i, 250, 5);
        circle(200 + i, 250, 5);
        line(20 + i, 250, 80 + i, 250);
        line(120 + i, 250, 180 + i, 250);
        line(220 + i, 250, 300 + i, 250);
        ellipse(160 + i, 250, 0, 180, 140, 50);
        rectangle(25 + i, 250, 20 + i, 260);
        rectangle(25 + i, 260, 0 + i, 265);
        line(0, 270, 800, 270);
        delay(30);
        }
        getch();
}
```

Output:-



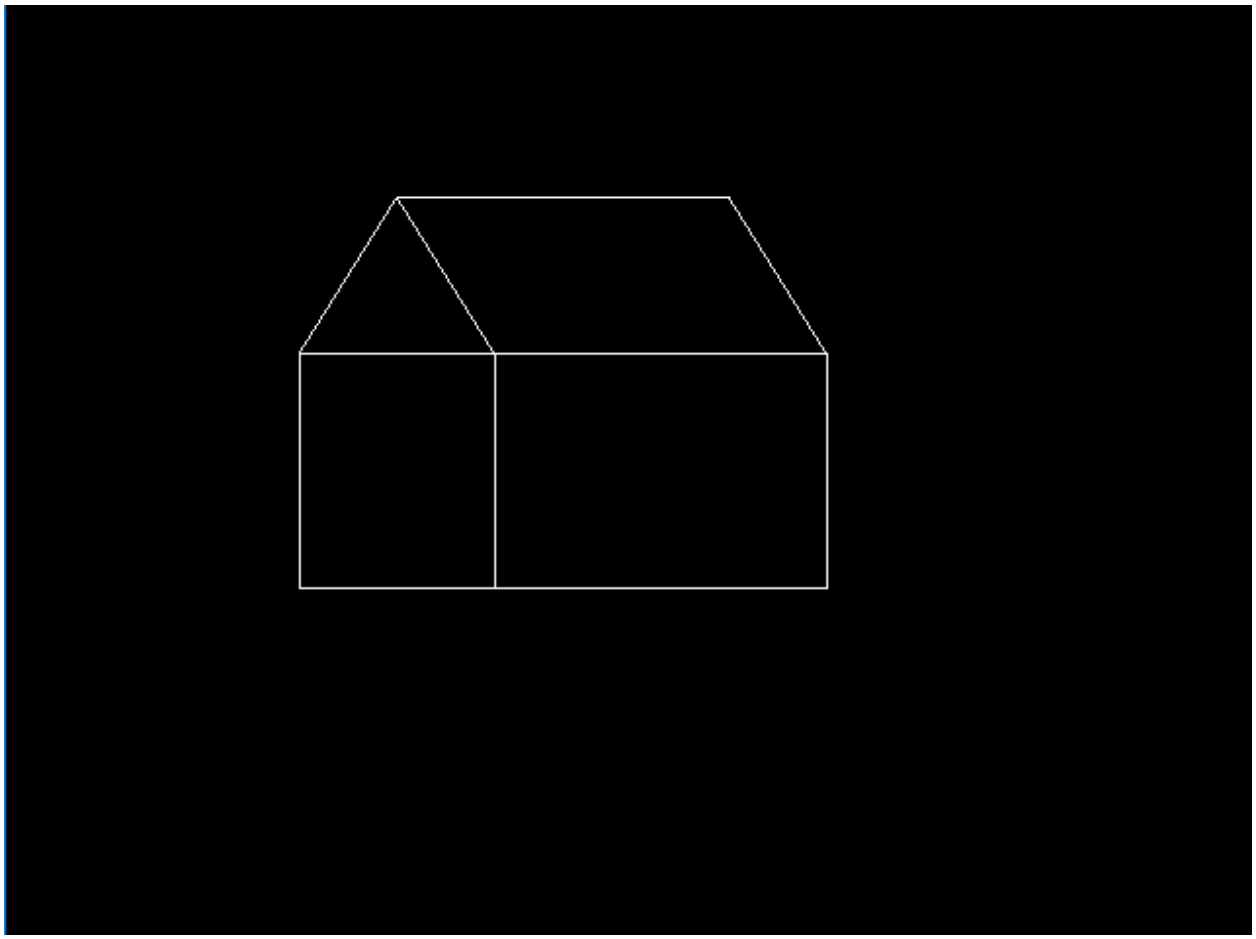
Program- 05

AIM :- Write a Program to Draw a House using DDA line drawing algorithm.

Program:-

```
#include<stdio.h>
#include<graphics.h>
#include<dos.h>
#include<conio.h>
int abs(int n) {
  return ((n > 0) ? n : (n * (-1)));
}
void DDA(int X0, int Y0, int X1, int Y1) {
  int dx = X1 - X0;
  int dy = Y1 - Y0;
  int steps = abs(dx) > abs(dy) ? abs(dx) : abs(dy);
  float Xinc = dx / (float) steps;
  float Yinc = dy / (float) steps;
  float X = X0;
  float Y = Y0;
  for (int i = 0; i <= steps; i++) {
          putpixel(X, Y, RED);
          X += Xinc;
          Y += Yinc;
  }
}
int main() {
  clrscr();
  int gd = DETECT, gm;
  initgraph( & gd, & gm, "C:\\TURBOC3\\BGI");
  DDA(200, 100, 150, 180);
  DDA(200, 100, 250, 180);
  DDA(200, 100, 370, 100);
  DDA(370, 100, 420, 180);
  DDA(150, 180, 250, 180);
  DDA(250, 180, 420, 180);
  DDA(250, 180, 250, 300);
  DDA(150, 180, 150, 300);
  DDA(420, 180, 420, 300);
  DDA(150, 300, 420, 300);
  getch();
```

```
  return 0;
}
```
Output:-



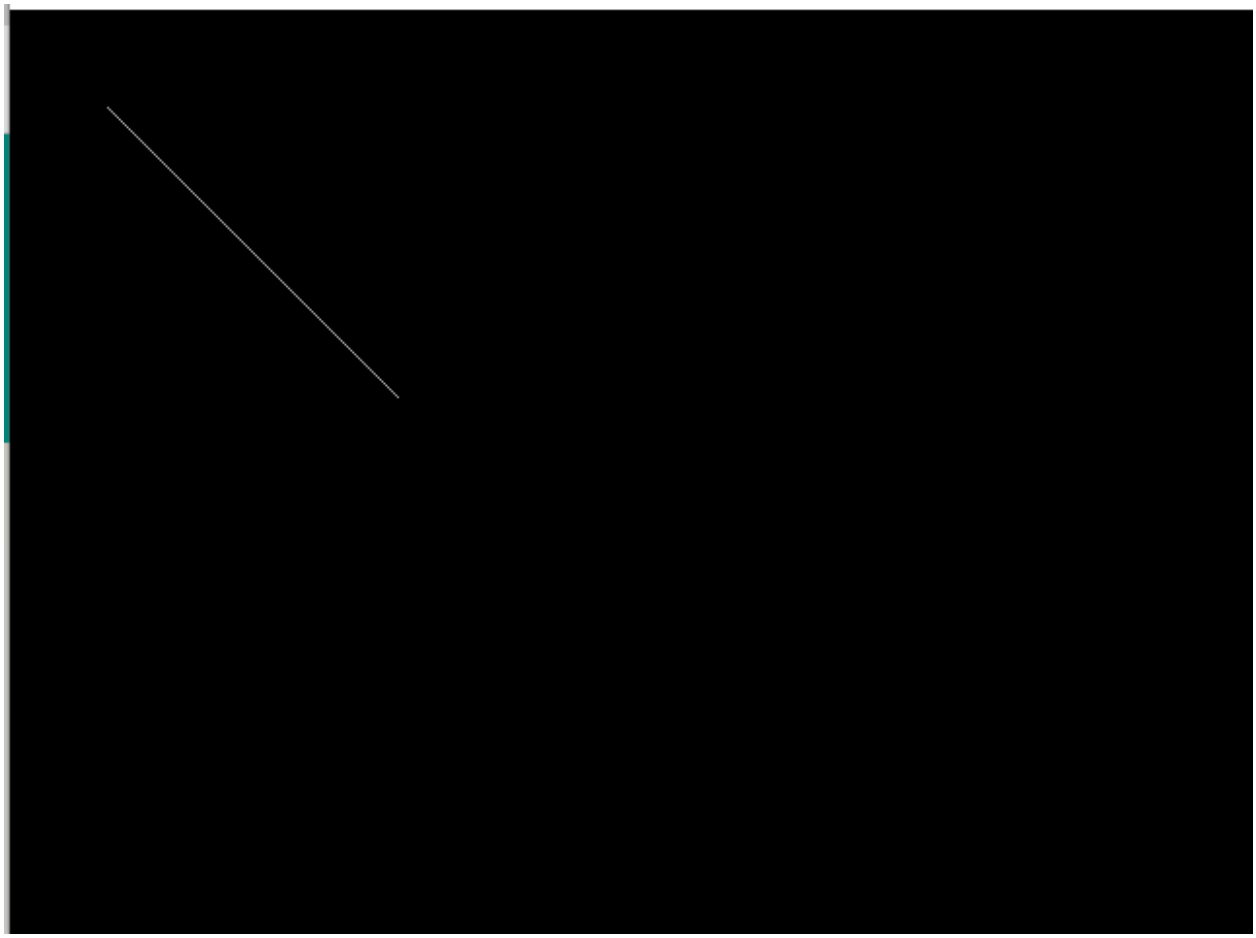
Program- 06

AIM :- Write a Program to Implement Bresenham's Line drawing algorithm.

Program:-

```
#include<stdio.h>
#include<graphics.h>
#include<conio.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) {
          if (p >= 0) {
          putpixel(x, y, 7);
          y = y + 1;
          p = p + 2 * dy - 2 * dx;
          } else {
          putpixel(x, y, 7);
          p = p + 2 * dy;
          }
          x = x + 1;
  }
}
int main() {
  clrscr();
  int gdriver = DETECT, gmode;
  initgraph( & gdriver, & gmode, "c:\\TURBOC3\\BGI");
  drawline(50, 50, 200, 200);
  getch();
  return 0;
}
```

Output:-



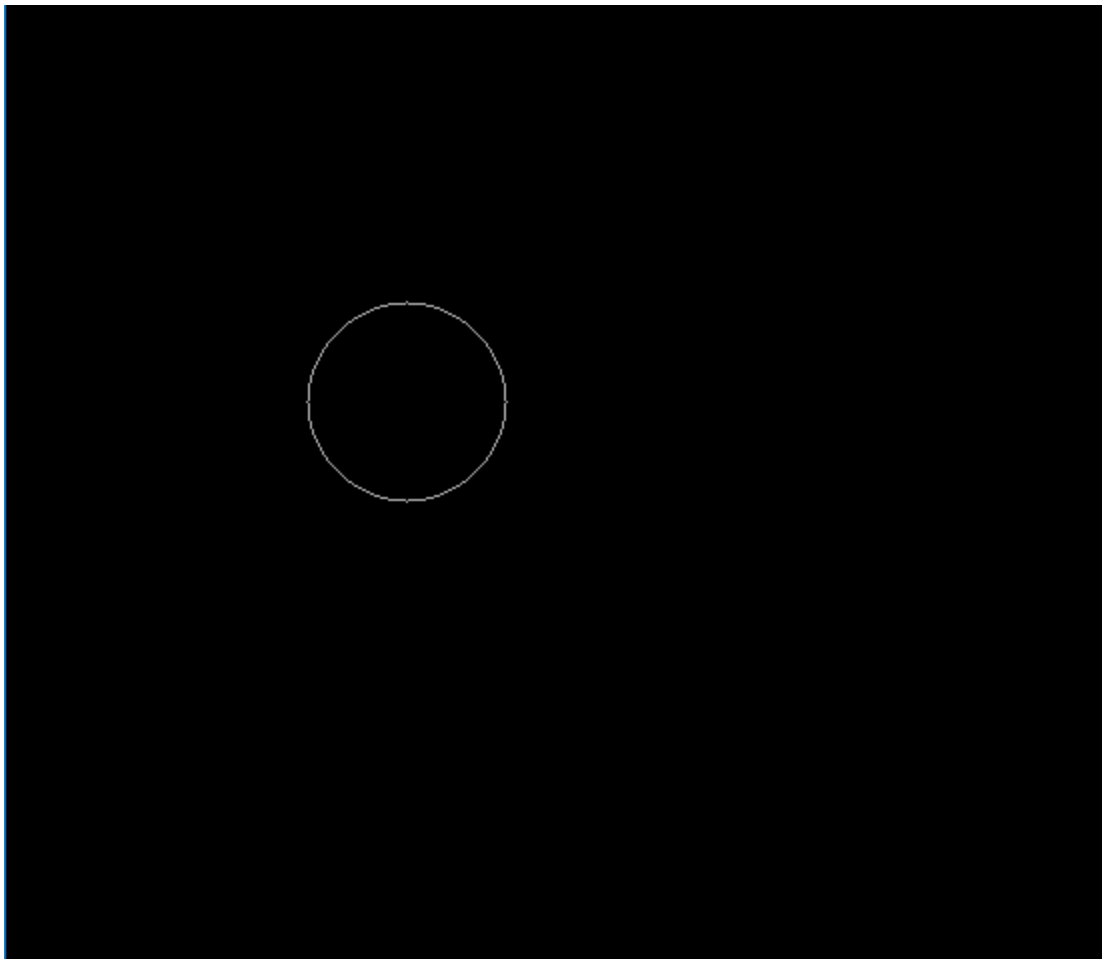
Program- 07

AIM :- Write a Program to Implement Midpoint circle drawing algorithm.

Program:-

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
void drawcircle(int x0, int y0, int radius) {
  int x = radius;
  int y = 0;
  int err = 0;
  while (x >= y) {
          putpixel(x0 + x, y0 + y, 7);
          putpixel(x0 + y, y0 + x, 7);
          putpixel(x0 - y, y0 + x, 7);
          putpixel(x0 - x, y0 + y, 7);
          putpixel(x0 - x, y0 - y, 7);
          putpixel(x0 - y, y0 - x, 7);
          putpixel(x0 + y, y0 - x, 7);
          putpixel(x0 + x, y0 - y, 7);
          if (err <= 0) {
          y += 1;
          err += 2 * y + 1;
          }
          if (err > 0) {
          x -= 1;
          err -= 2 * x + 1;
          }
  }
}
int main() {
  clrscr();
  int gdriver = DETECT, gmode, error, x, y, r;
  initgraph( & gdriver, & gmode, "c:\\turboc3\\bgi");
  drawcircle(200, 200, 50);
  getch();
  return 0;
}
```

Output:-



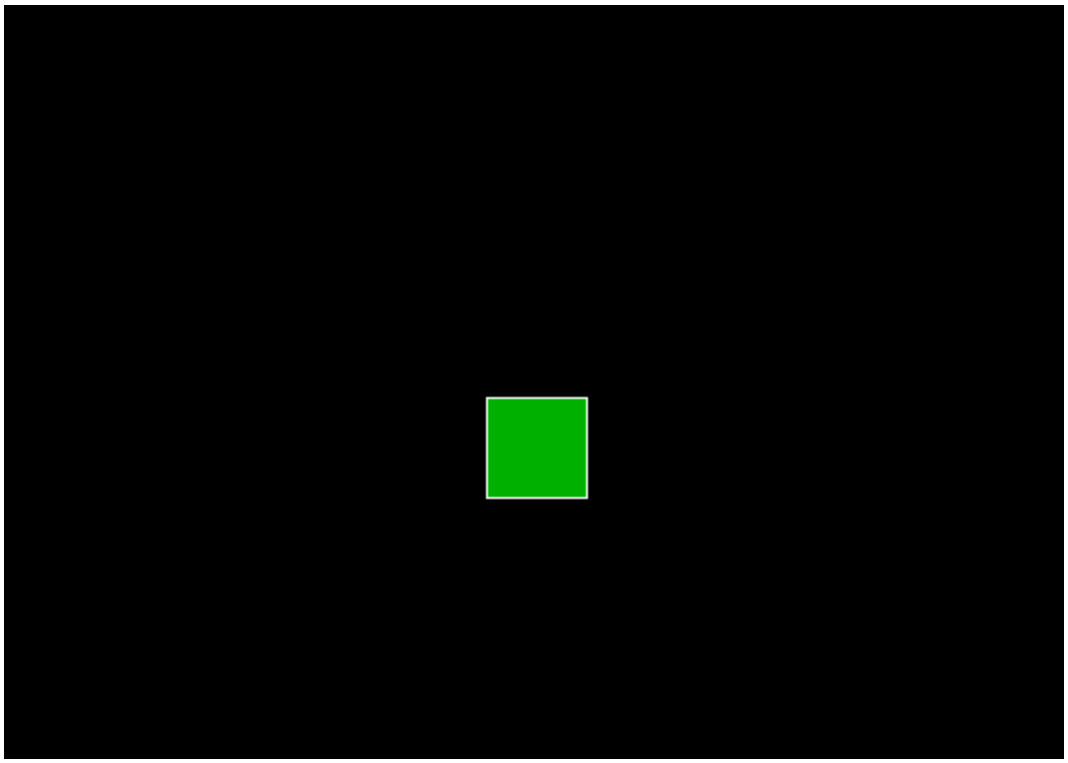
Program- 08

AIM :- Write a Program to Boundary Fill algorithm.

Program:-

```
#include <graphics.h>
#include<iostream.h>
#include <dos.h>
#include<conio.h>
void boundaryFill4(int x, int y, int fill_color,int boundary_color)
{       delay(1)  ;
         if(getpixel(x, y) != boundary_color &&
         getpixel(x, y) != fill_color)
         {
                  putpixel(x, y, fill_color);
                  boundaryFill4(x + 1, y, fill_color, boundary_color);
                  boundaryFill4(x, y + 1, fill_color, boundary_color);
                  boundaryFill4(x - 1, y, fill_color, boundary_color);
                  boundaryFill4(x, y - 1, fill_color, boundary_color);
         }
}

int main()
{
         clrscr();
         int gd = DETECT, gm;
         initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
         rectangle(250, 200, 300,300);
         boundaryFill4(255, 205, 6, 15);
         getch();

         return 0;
}
```

Output:-



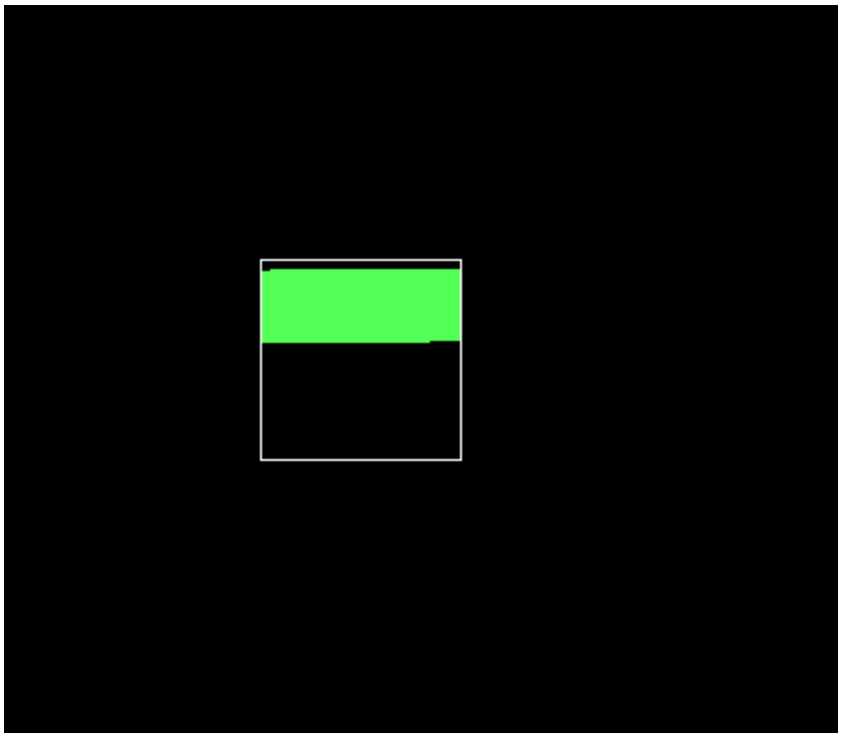
Program- 09

AIM :- Write a Program to Flood Fill algorithm.

Program:-

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void flood(int,int,int,int);
void main()
{
   intgd=DETECT,gm;
   initgraph(&gd,&gm,"C:/TURBOC3/bgi");
   rectangle(50,50,250,250);
   flood(55,55,10,0);
   getch();
}
void flood(intx,inty,intfillColor, intdefaultColor)
{
   if(getpixel(x,y)==defaultColor)
   {
      delay(1);
      putpixel(x,y,fillColor);
      flood(x+1,y,fillColor,defaultColor);
      flood(x-1,y,fillColor,defaultColor);
      flood(x,y+1,fillColor,defaultColor);
      flood(x,y-1,fillColor,defaultColor);
   }
}
```

Output:-



PROGRAM-10

AIM:-

Write a program in C++ to display major 2D Transformations – Translation,Rotation and Scaling.

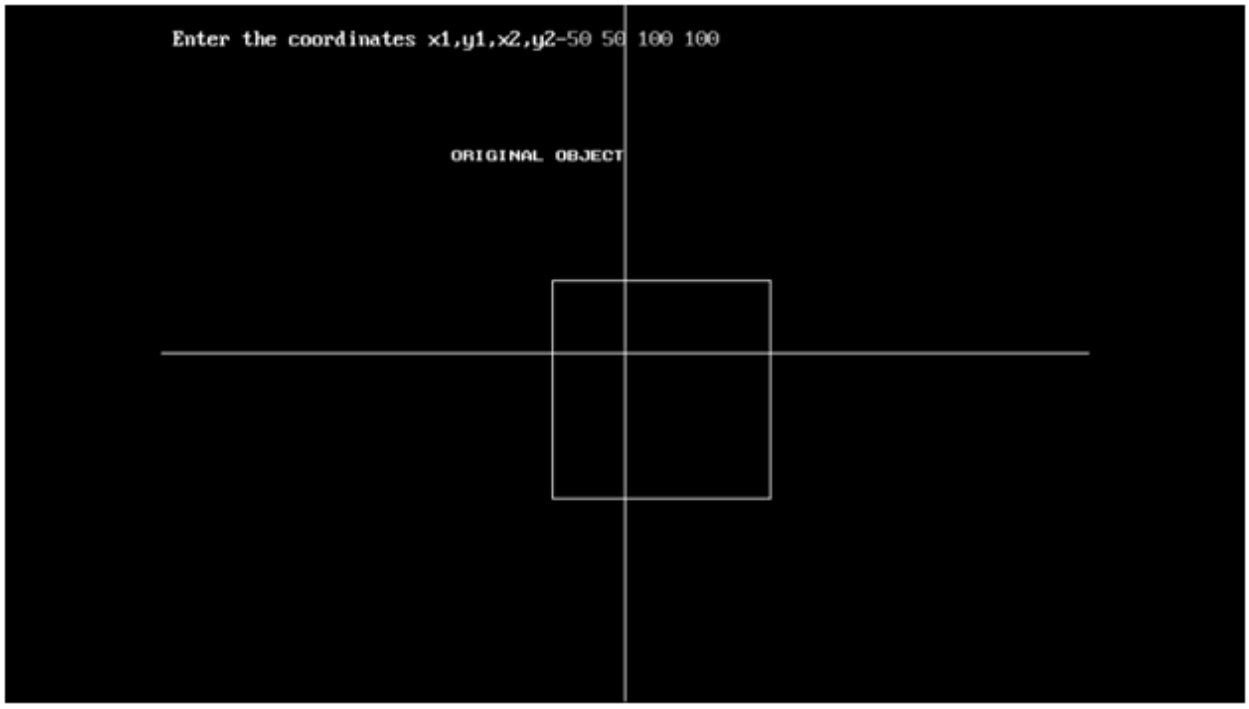
PROGRAM :-

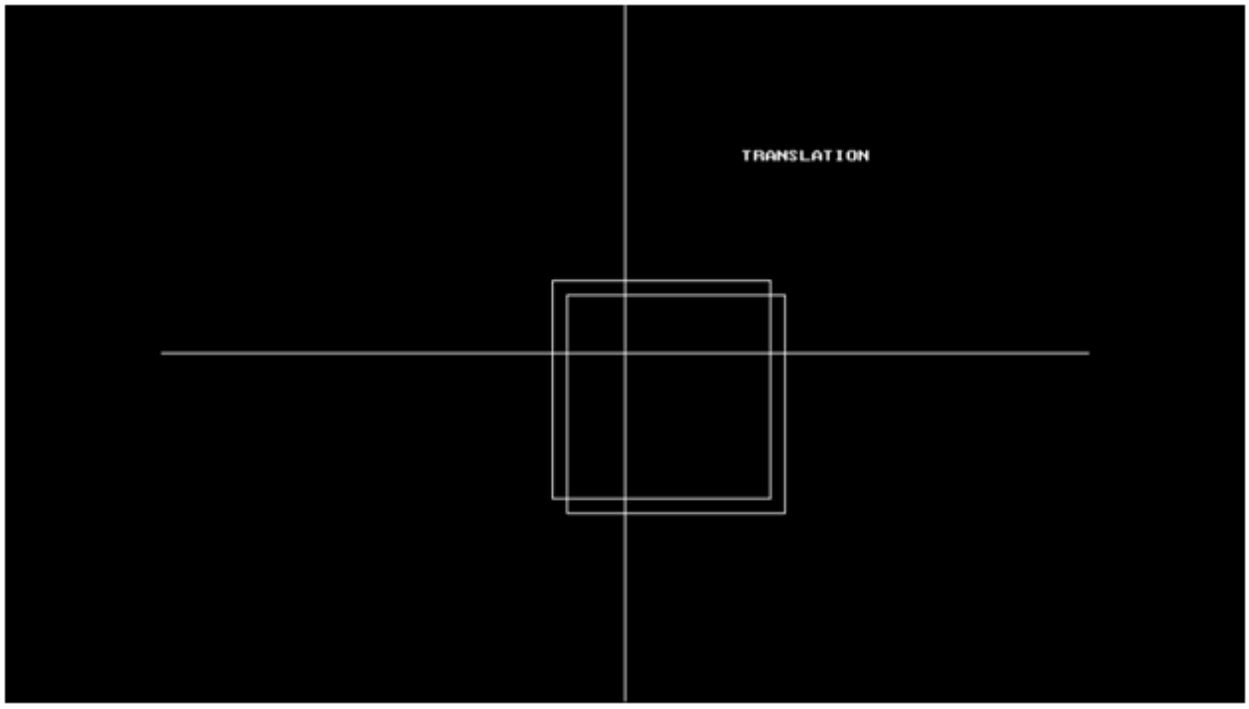
```
#include<stdio.h>
#include<graphics.h>
#include<stdlib.h>
#include<math.h>
#include<conio.h>
int x1,y1,x2,y2,midx,midy;

void axis()
{
midx=getmaxx() / 2;
midy=getmaxy() / 2;
line(0,midy,midx*2,midy);
line(midx,0,midx,midy*2);
rectangle(x1,y1,x2,y2);
}

void translation()
{
int tx,ty,xn1,yn1,xn2,yn2;
printf("\n\n\n Enter the translation:\n");
scanf("%d%d",&tx,&ty);
cleardevice();
outtextxy(400,100,"TRANSLATION");
xn1=x1+tx;
yn1=y1+ty;
xn2=x2+tx;
yn2=y2+ty;
axis();
rectangle(xn1,yn1,xn2,yn2);
getch();
}

void scaling()
{
float xn1,yn1,xn2,yn2;
```

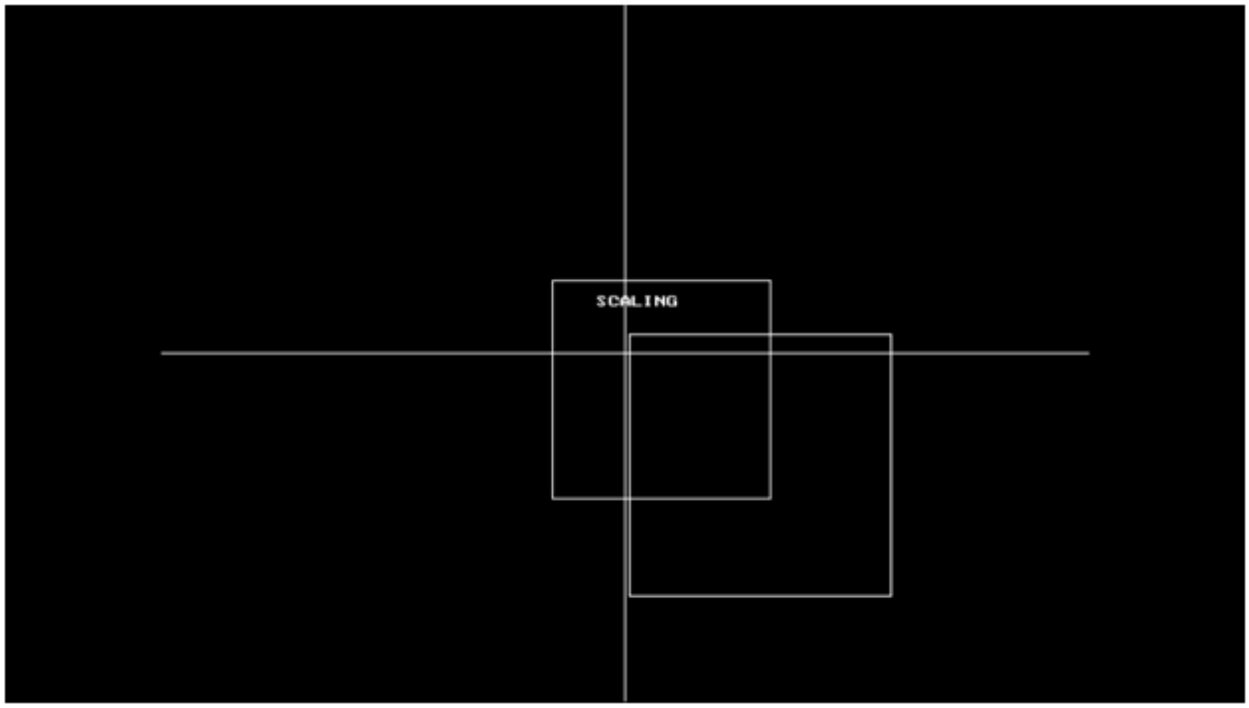
```
float sx,sy;
printf("\n\n\nEnter the scaling factor");
scanf("%f%f",&sx,&sy);
cleardevice();
outtextxy(300,200,"SCALING");
xn1=x1*sx;
yn1=y1*sy;
xn2=x2*sx;
yn2=y2*sy;
axis();
rectangle(xn1,yn1,xn2,yn2);
getch();
}

void rotation()
{
int ang;
float rx,xn1,yn1,xn2,yn2,x1n1,y1n1,x2n2,y2n2;
printf("\n\n\nEnter the angle for rotation:\n");
scanf("%d",&ang);
cleardevice();
outtextxy(500,200,"ROTATION");
rx=(ang*3.14)/180;
xn1=x1*cos(rx)-y1*sin(rx);
yn1=y1*cos(rx)+x1*sin(rx);
xn2=x2*cos(rx)-y2*sin(rx);
yn2=y2*cos(rx)+x2*sin(rx);
x1n1=x2*cos(rx)-y1*sin(rx);
y1n1=y1*cos(rx)+x2*sin(rx);
x2n2=x1*cos(rx)-y2*sin(rx);
y2n2=y2*cos(rx)+x1*sin(rx);
axis();
line(xn1,yn1,x1n1,y1n1);
line(x1n1,y1n1,xn2,yn2);
line(xn2,yn2,x2n2,y2n2);
line(x2n2,y2n2,xn1,yn1);
getch();
}

void get()
{
```

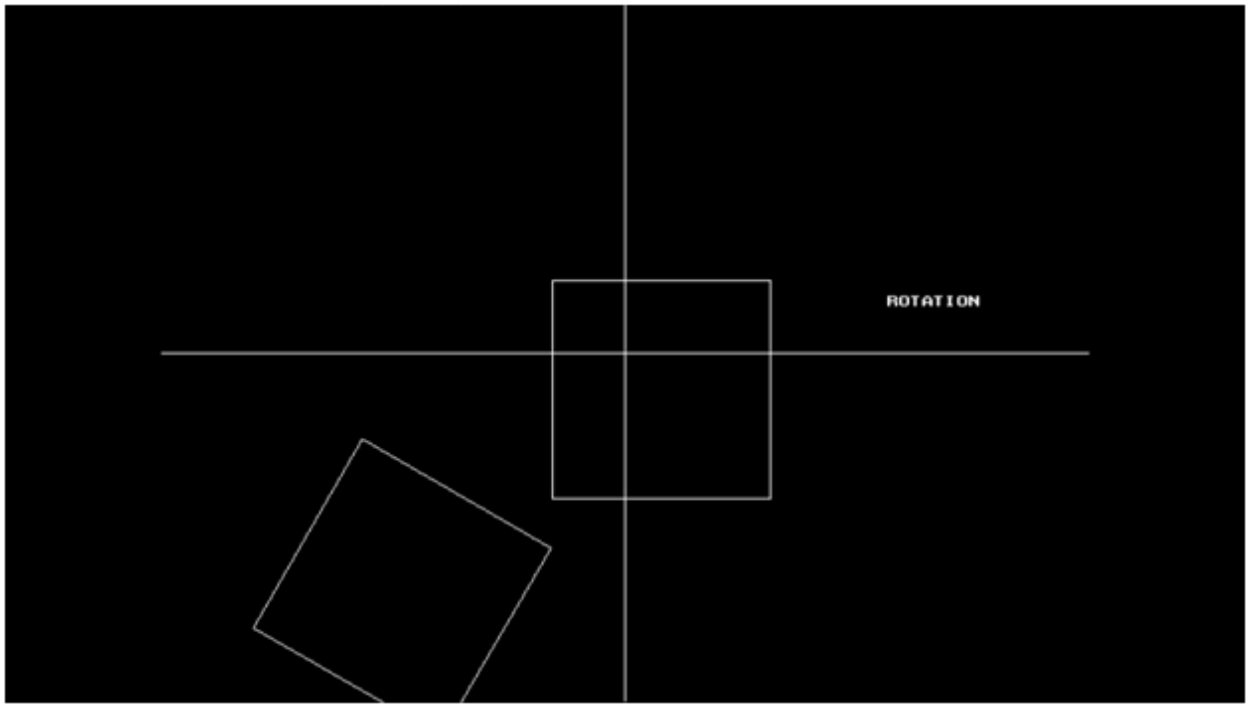
```
printf("\n Enter the coordinates x1,y1,x2,y2-");
scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
outtextxy(200,100,"ORIGINAL OBJECT");
x1= getmaxx() / 2-x1;
y1= getmaxy() / 2-y1;
x2 = getmaxx() / 2+x2;
y2 = getmaxy() / 2+y2;
axis();
getch();
}

void main()
{
int ch,gd=DETECT,gm;
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
get();
do
{
cleardevice();
outtextxy(10,10,"1)TRANSLATION");
outtextxy(10,20,"2)SCALING");
outtextxy(10,30,"3)ROTATION");
outtextxy(10,60,"4)EXIT");
outtextxy(10,70,"ENTER YOUR CHOICE:");
scanf("%d",&ch);
switch(ch)
{
case 1:
translation();
break;
case 2:
scaling();
break;
case 3:
rotation();
break;
default:
exit(0);
}
}while(ch<4);
}
```

OUTPUT:-


Enter the coordinates x1,y1,x2,y2-50 50 100 100
ORIGINAL OBJECT


TRANSLATION


SCALING


ROTATION

PROGRAM-11

AIM:-

Write a program in C++ to display Cohen-Sutherland Line Clipping Algorithm.

PROGRAM :-

```
#include<iostream.h>
#include<stdlib.h>
#include<math.h>
#include<graphics.h>
#include<dos.h>

typedef struct coordinate
{
        int x,y;
        char code[4];
}PT;

void drawwindow();
void drawline(PT p1,PT p2);
PT setcode(PT p);
int visibility(PT p1,PT p2);
PT resetendpt(PT p1,PT p2);

void main()
{
        int gd=DETECT,v,gm;
        PT p1,p2,p3,p4,ptemp;

        cout<<"\nEnter x1 and y1\n";
        cin>>p1.x>>p1.y;
        cout<<"\nEnter x2 and y2\n";
        cin>>p2.x>>p2.y;

        initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
        drawwindow();
        delay(500);

        drawline(p1,p2);
        delay(500);
        cleardevice();
```

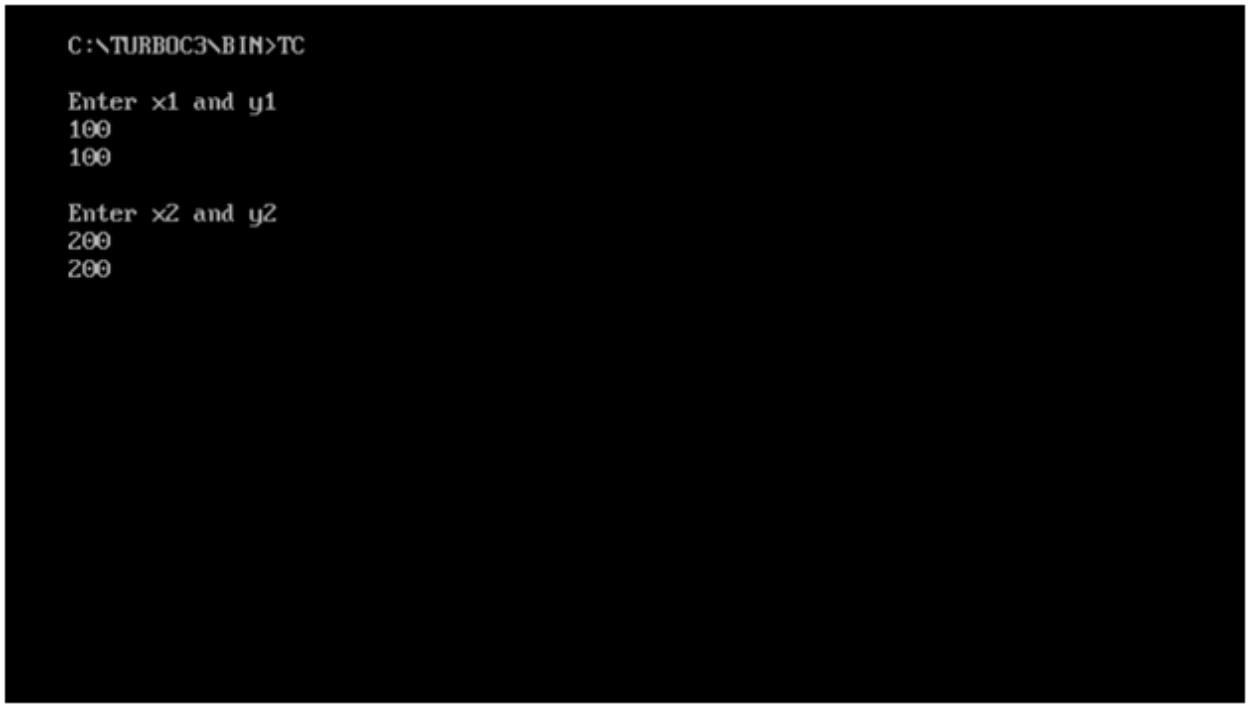
```
        delay(500);
        p1=setcode(p1);
        p2=setcode(p2);
        v=visibility(p1,p2);
        delay(500);

        switch(v)
        {
        case 0: drawwindow();
                        delay(500);
                        drawline(p1,p2);
                        break;
        case 1:drawwindow();
                        delay(500);
                        break;
        case 2:p3=resetendpt(p1,p2);
                        p4=resetendpt(p2,p1);
                        drawwindow();
                        delay(500);
                        drawline(p3,p4);
                        break;
        }

        delay(5000);
        closegraph();
}

void drawwindow()
{
        line(150,100,450,100);
        line(450,100,450,350);
        line(450,350,150,350);
        line(150,350,150,100);
}

void drawline(PT p1,PT p2)
{
        line(p1.x,p1.y,p2.x,p2.y);
}
```

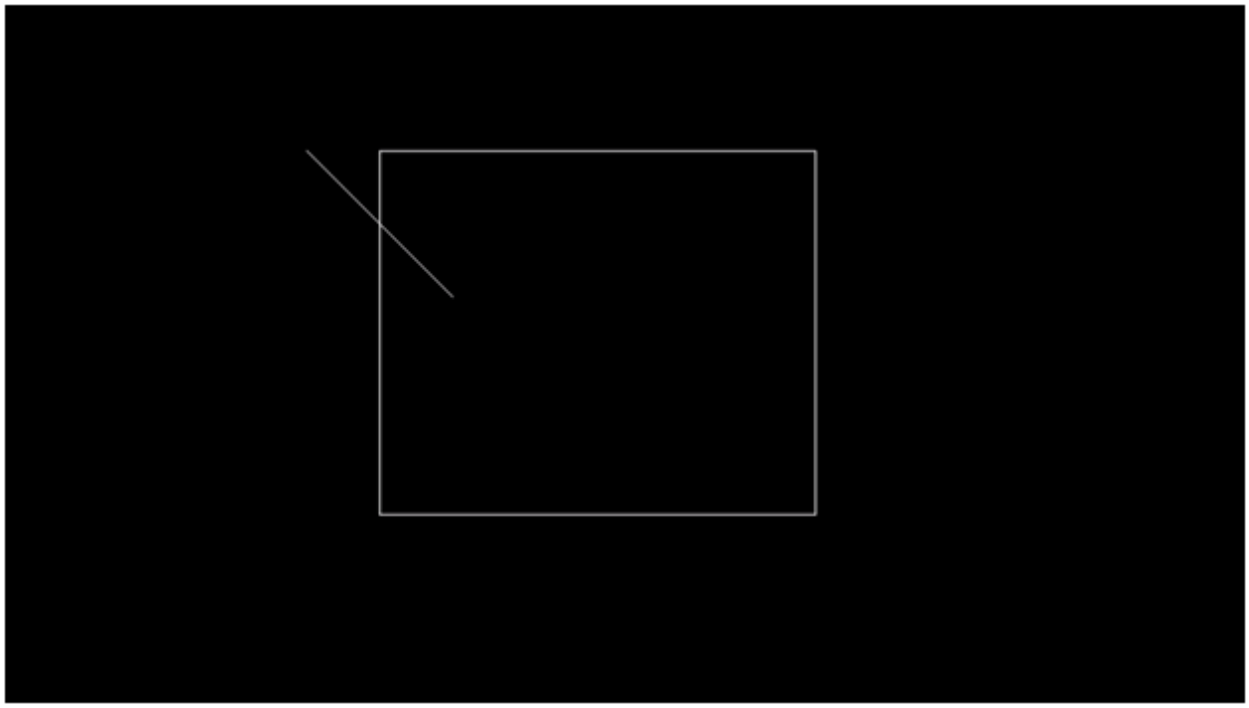
```
PT setcode(PT p)
{
	PT ptemp;

	if(p.y<100)
		ptemp.code[0]='1';
	else
		ptemp.code[0]='0';

	if(p.y>350)
		ptemp.code[1]='1';
	else
		ptemp.code[1]='0';

	if(p.x>450)
		ptemp.code[2]='1';
	else
		ptemp.code[2]='0';

	if(p.x<150)
		ptemp.code[3]='1';
	else
		ptemp.code[3]='0';

	ptemp.x=p.x;
	ptemp.y=p.y;

	return(ptemp);
}

int visibility(PT p1,PT p2)
{
	int i,flag=0;

	for(i=0;i<4;i++)
	{
		if((p1.code[i]!='0') || (p2.code[i]!='0'))
			flag=1;
	}

	if(flag==0)
```

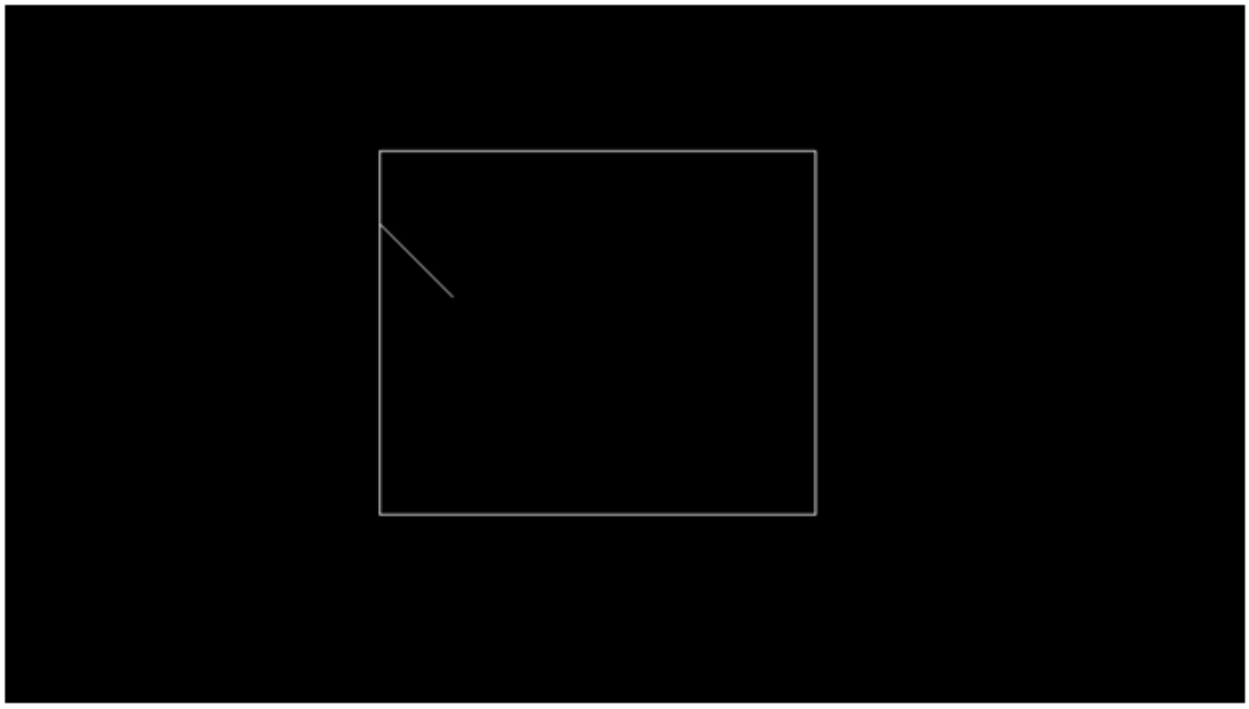
```
                return(0);

        for(i=0;i<4;i++)
        {
                if((p1.code[i]==p2.code[i]) && (p1.code[i]=='1'))
                        flag='0';
        }

        if(flag==0)
                return(1);

        return(2);
}

PT resetendpt(PT p1,PT p2)
{
        PT temp;
        int x,y,i;
        float m,k;

        if(p1.code[3]=='1')
                x=150;

        if(p1.code[2]=='1')
                x=450;

        if((p1.code[3]=='1') || (p1.code[2]=='1'))
        {
                m=(float)(p2.y-p1.y)/(p2.x-p1.x);
                k=(p1.y+(m*(x-p1.x)));
                temp.y=k;
                temp.x=x;

                for(i=0;i<4;i++)
                        temp.code[i]=p1.code[i];

                if(temp.y<=350 && temp.y>=100)
                        return (temp);
        }

        if(p1.code[0]=='1')
```

```
            y=100;

        if(p1.code[1]=='1')
            y=350;

        if((p1.code[0]=='1') || (p1.code[1]=='1'))
        {
            m=(float)(p2.y-p1.y)/(p2.x-p1.x);
            k=(float)p1.x+(float)(y-p1.y)/m;
            temp.x=k;
            temp.y=y;

            for(i=0;i<4;i++)
                temp.code[i]=p1.code[i];

            return(temp);
        }
        else
            return(p1);
}
```

OUTPUT:-







PROGRAM-12

AIM:-

Write a program in C++ to display Sutherland-Hodgman Polygon Clipping Algorithm.

PROGRAM:-

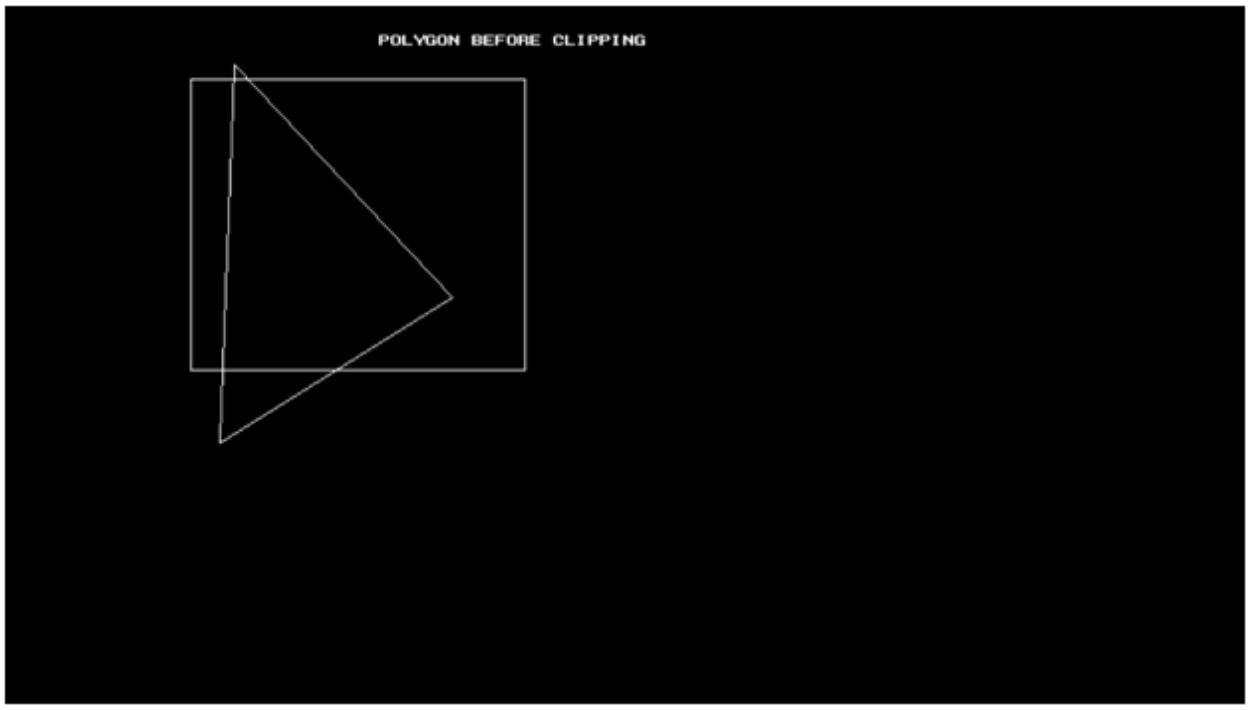
```
#include <stdio.h>
#include <graphics.h>
#include <conio.h>
#include <math.h>
#include <process.h>
#define TRUE 1
#define FALSE 0

typedef unsigned int outcode;
outcode CompOutCode(float x,float y);
enum  {  TOP = 0x1,
BOTTOM = 0x2,
RIGHT = 0x4,
LEFT = 0x8
};
float xmin,xmax,ymin,ymax;
void clip(float x0,float y0,float x1,float y1)
{
outcode outcode0,outcode1,outcodeOut;
int accept = FALSE,done = FALSE;
outcode0 = CompOutCode(x0,y0);
outcode1 = CompOutCode(x1,y1);
do
{
   if(!(outcode0|outcode1))
   {
         accept = TRUE;
         done = TRUE;
   }
   else
   if(outcode0 & outcode1)
         done = TRUE;
   else
   {
         float x,y;
```

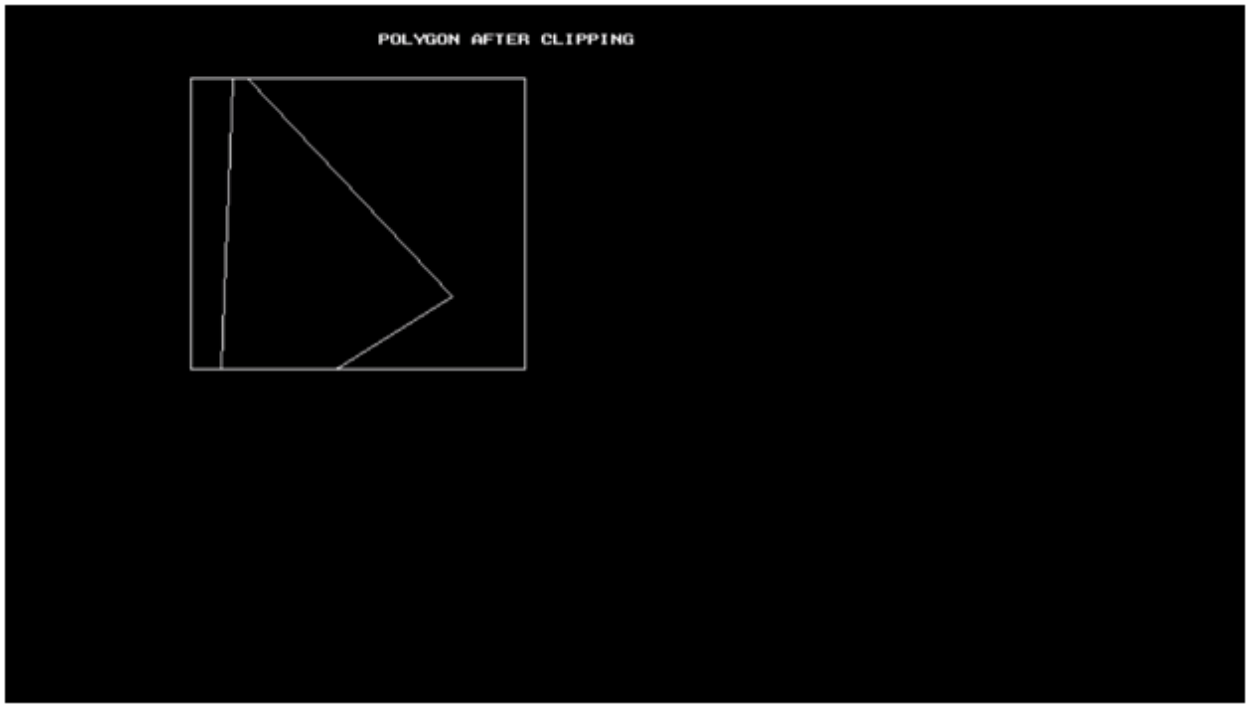
```
        outcodeOut = outcode0?outcode0:outcode1;
        if(outcodeOut & TOP)
        {
           x = x0+(x1-x0)*(ymax-y0)/(y1-y0);
           y = ymax;
        }
        else if(outcodeOut & BOTTOM)
        {
           x = x0+(x1-x0)*(ymin-y0)/(y1-y0);
           y = ymin;
        }
        else if(outcodeOut & RIGHT)
        {
           y = y0+(y1-y0)*(xmax-x0)/(x1-x0);
           x = xmax;
        }
        else
        {
           y = y0+(y1-y0)*(xmin-x0)/(x1-x0);
           x = xmin;
        }
        if(outcodeOut==outcode0)
        {
           x0 = x;
           y0 = y;
           outcode0 = CompOutCode(x0,y0);
        }
        else
        {
           x1 = x;
           y1 = y;
           outcode1 = CompOutCode(x1,y1);
         }
   }
}while(done==FALSE);
if(accept)
   line(x0,y0,x1,y1);
outtextxy(150,20,"POLYGON AFTER CLIPPING");
rectangle(xmin,ymin,xmax,ymax);
}
outcode CompOutCode(float x,float y)
```

```
{
   outcode code = 0;
   if(y>ymax)
         code|=TOP;
   else if(y<ymin)
            code|=BOTTOM;
   if(x>xmax)
         code|=RIGHT;
   else if(x<xmin)
         code|=LEFT;
   return code;
}
void main( )
{
float x1,y1,x2,y2;

int gdriver = DETECT, gmode, n,poly[14],i;
clrscr( );
printf("Enter the no of sides of polygon:");
scanf("%d",&n);
printf("\nEnter the coordinates of polygon\n");
for(i=0;i<2*n;i++)
{
   scanf("%d",&poly[i]);
}
poly[2*n]=poly[0];
poly[2*n+1]=poly[1];
printf("Enter the rectangular coordinates of clipping window\n");
scanf("%f%f%f%f",&xmin,&ymin,&xmax,&ymax);

initgraph(&gdriver, &gmode, "C://TURBOC3//BGI");

outtextxy(150,20,"POLYGON BEFORE CLIPPING");
drawpoly(n+1,poly);
rectangle(xmin,ymin,xmax,ymax);
getch( );
cleardevice( );
for(i=0;i<n;i++)
clip(poly[2*i],poly[(2*i)+1],poly[(2*i)+2],poly[(2*i)+3]);
getch( );
restorecrtmode( );  }
```

OUTPUT:-

```
Enter the no of sides of polygon:3

Enter the coordinates of polygon
50
40
200
200
40
300
Enter the rectangular coordinates of clipping window
20
50
250
250
```




POLYGON AFTER CLIPPING

Case Study- 1

Aim:- **Study of various file formats.Like text, image, audio, video.**

FILE FORMAT: The specific format in which an file is saved. The format is identified by the three letter extension at the end of the file name. Every format has its own characteristics, advantages and disadvantages. By defining the file format it may be possible to determine the number of bits per pixel and additional information.
File Format A file format is a standard way that information is encoded for storage in a computer file. A file format specifies how bits are used to encode information in a digital storage medium. File formats may be either proprietary or free and may be either unpublished or open. For example : image file format , audio file format , video file format and Text Formats etc.

Image File Formats

Image File Format Image Formats bring different ways to overcome the problem of delivering an image with reduced file size and minimum download time Image file formats are standardized means of organizing and storing digital images Image files are composed of digital data in one of these formats that can be rasterized for use on a computer display or printer An image file format may store data in uncompressed, compressed, or vector formats. Once rasterized, an image becomes a grid of pixels, each of which has a number of bits to designate its color equal to the color depth of the device displaying it.

GIF Stands for Graphic Interchange Format. Uses lossless compression technique Supports 8 bit colors Supported by all browsers Suitable for text , artwork , icons and cartoons Large File Size Extension is .gif

JPEG Stands for Joint Photographic Experts Group Uses lossy compression technique Supports 24-bit colors Supported by all browser Suitable for photographs Smaller File Size as compared to GIF Extension is .jpg or .jpeg

PNG Stands for Portable Network Graphics Uses lossless compression technique which supports 24-bits colors but not supported by all browsers Suitable for text , icons etc. Smaller File Size as compared to GIF and JPEG Extension is .png

Sound File Formats

An audio file format is a file format for storing digital audio data on a computer system. This data can be stored uncompressed, or compressed to reduce the file size. It can be a raw bit stream, but it is usually a container format or an audio data format with defined storage layer.

MPEG Stands for Moving Picture Experts Group MPEG files employ loss data compression MPEG files use the .mp3 filename extension Digital music , pod casts and audio books are often saved as MPEG3 files Older MPEG files are usually designated with an extension of .mp2 Older MPEG files are usually designated with an extension of .mp2 One minute of music takes up approximately 1MB of storage space

WAVE Stands for Resource Interchange File Format Waveform Wave files employ an uncompressed audio format WAVE uses .wav filename extensions This file format is a proprietary format that was sponsored jointly by Microsoft and IBM The .wav format can support both monaural (single-channel) and stereo (multichannel) audio One minute of stereo music takes up approximately 10MB of storage space

MIDI Stands for Musical Instrument Digital Interface MIDI files are very small, but are not compressed They use .mid or .midi filename extensions A MIDI file is not a recording of music being played; it is a description of how to create the sound based on predefined sounds, like a 6-string guitar or pipe organ A MIDI recording never contains the human voice A 10KB (10,000 storage locations) MIDI file could easily hold more than a minute of music

WMA Stands for Windows Media Audio WMA file formats use a loss compression developed by Microsoft and is widely recognized by a variety of players and jukeboxes, like Winamp and Music Match Windows Media Audio files use a .wma filename extension It is non-propriety MP3 format for saving and storing music files

AIFF Stands for Audio Integrated File Format AIFF uses the .aiff file extension AIFF is the native audio file format developed by Apple for the Macintosh computer platform It is an uncompressed audio format This means that it is much larger in file size than MP3 but can support the highest possible audio recording quality as well as lower quality settings AIFF can support music from the highest quality 48K recording through to lower quality recordings

AAC Stands for Advanced Audio Coding AAC is a standardized loss compression It uses the .aac file extension Intended to be the successor to MP3 format, AAC generally has better audio quality and is the default format for many digital audio players like the iPod, iPhone, iPad, Nintendo DS and others

Video File Formats

Video File Format is consist of two ways: Analog Digital Examples of these formats is: MPEG MOV AVI WMV Real Video SWF

Types of Video File Format

MPEG Stands for Motion Picture Experts Group MPEG files are also a common format for digital videos and movies It use the filename extensions of .mpg or .mpeg The latest MPEG version, MPEG4, uses the .mp4 filename extension

MOV Stands for QuickTime Movie The QuickTime video and movie file format was originally developed for the Apple Macintosh But MOV is now recognized by all personal computers QuickTime movies use the .mov filename extension .qt filename extension used as an alternative

AVI Stands for Audio/Video Interleave The AVI video and movie file format was originally developed by Microsoft for Windows-based personal computers It uses the .avi filename extension It is the nominal standard for personal computers using Windows

WMV Stands for Windows Media Video WMV file formats are propriety to Microsoft licensed products WMV are not widely recognized by non-Windows players Windows Media Video files use a .wmv filename extension Files stored in this format are intended to be played, not edited

RealVideo is a proprietary file format It uses .rm, .ram, .ra as file extensions Used mainly for real-time streaming of audio and video it requires RealPlayer (Windows and Mac) software

SWF Stands for Originally ShockWave Flash, now means Small Web Format SWF files use the .swf file extension It is a format for multimedia, vector graphics and ActionScript in the Adobe Flash environment SWF files can contain animations or applets of varying degrees of interactivity and function Currently, SWF functions are the dominant format for displaying "animated" vector graphics on the Web

Text File Formats

Text Format (Con’t) Most of these formats represent rows of data on different lines of the file using different strategies to separated data values within each row. 'Fixed-width' formats place each data entry in a separate column and therefore limit the size of the data entries. 'Separated' formats use a special character or character sequence to separate entries. For instance, the comma separated value, the tab separated value formats and the space separated value formats use commas, tabs, and spaces respectively to separate the data fields.

DOC/DOCX Microsoft's Word (word processing) software saves documents using the .doc filename extension These files contain special formatting codes that identify how the text with look (bold, italic, color, typeface, etc.) as well as how the page lays out (margins, indentation, pagination, etc.) This file format was superceded in Word 2007 with the .docx filename extension DOCX files incorporate XML (EXtensible Markup Language) coding rules that help integrate a document with Internet applications As a result, earlier versions of Word cannot read DOCX documents, but Microsoft does provide software that converts DOC documents into a DOCX format Word 2007 can read DOC documents and is able to save new documents in a DOC format when using the Save As option

RTF Stands for Rich Text Format RTF documents are designed to transfer documents between word processing software These files use .rtf filename extensions While the text formatting options are as "rich" as those used by Word, RTF files have limited page layout options For example, you cannot create columns, add page numbers, headers, or footers The WordPad word processor included with Windows defaults to creating RTF documents

TXT TXT documents only contain text Any computer can read a TXT file, but don't expect it to look pretty The Notepad text editor included with Windows defaults to creating TXT documents The individual characters in the document (letters, punctuation, newlines etc.) are each encoded into bytes using the ASCII encoding (or another character encoding such as UTF8 or iso8859-1, particularly if the document is not in English), and stored in a simple sequence This format only stores the text itself, with no information about formatting, fonts, page size, or anything like that It is portable across all computer systems and can be read and modified by a huge range of software applications The details of the format are freely available and standardized If the storage media are damaged, any undamaged sections can be recovered without problems

HTML Stands for HyperText Markup Language It use either .htm or .html filename extensions HTML files contain codes that browsers, like Internet Explorer or Safari, translate into Web pages The text, plus simple formatting, is stored in a simple encoding that is based on

the plain text file format above, with plain text markup interspersed with the text This format is freely available and controlled by a public-interest standards body The document can be viewed in any web browser It can be edited in a text editor by someone who knows HTML, or in any number of “rich text” editors, word processors, HTML editors and so on

PDF Stands for Portable Document Format PDF files use a .pdf filename extension These files are created using a software package from Adobe called Acrobat This software must be purchased and converts files created by other softwares, like Microsoft's Word, into a read-only PDF file In this format case, the text plus formatting, page size and similar information are stored in a moderately complex encoding While the details of this encoding are freely available, the format is owned by Adobe and can be changed by them at any time, for any reason The document can be viewed and printed on all major platforms, using free software provided by Adobe (or others) PDF documents cannot be readily edited

ZIP ZIP files are compressed data files Files of ZIP format use the .zip filename extension A popular shareware program called WinZip originally used this format At one time you needed to use WinZip to compress (zip) and uncompress (unzip) ZIP files, but many personal computers now recognize this file format and will unzip the files ZIP files can contain several compressed files under one filename, called an archive, when using the WinZip software As a result you could unzip a file (archive) named testbank.zip and find it contains 12 unique files for a 12chapter textbook

Case Study- 2

AIM:- Study of animation principles and formats.

Animation means giving life to any object in computer graphics. It has the power of injecting energy and emotions into the most seemingly inanimate objects. Computer-assisted animation and computer-generated animation are two categories of computer animation. It can be presented via film or video.

The basic idea behind animation is to play back the recorded images at the rates fast enough to fool the human eye into interpreting them as continuous motion. Animation can make a series of dead images come alive. Animation can be used in many areas like entertainment, computer aided-design, scientific visualization, training, education, e-commerce, and computer art.

In the last century Disney animators invented 12 basic laws and principles of animation. Knowing and practicing them will not only help you to create animation, but will also make your animation more appealing and alive.

12 principles of animation

1. Squash and Stretch

Squash and stretch makes an illusion of character's elasticity, volume and flexibility. Squash and stretch is also helpful in facial animation. The extent of squash and stretch depends on scene requirements and animation stylistics. More often squash and stretch is extremely strong in animation films and feeble in feature films when realistic characters are used. Squash and stretch is used in all kinds of character animation from bouncing ball to moving person.

2. Anticipation

This motion prepares a viewer for the main action, which the character intends to do. For instance, starting to run, jump or make a dash. To jump up, first you need to squat - this is preparation or anticipation. Comic effect can be achieved without preparation or anticipation after you used it several times. Virtually all real motions to a greater or lesser

extent contain preparation or anticipation - a sweeping motion with the bandy before a strike, squat before a jump, swinging your arm back before throwing a stone, etc.

3. Staging

Poses and actions, arrangement of cameras, background and stage elements shall clearly demonstrate character's temper, reaction, character's attitude to a story and continuity of the plotline. Effective use of close-ups, medium shots and master shots as well as camera angles help to narrate the story. Film duration is limited therefore each succession, each scene, each film frame shall be relevant to the whole story. Do not confuse the viewer with too many simultaneous actions, use one clear action at a time to convey the idea.

4. Straight Ahead Action and Pose to Pose

Straight ahead animation starts with the first picture (in hand-drawn animation) or character pose (in 3D animation) and gradually picture by picture (or pose by pose) is brought to the end of the scene. Using this method you may lose the size, volume and proportions of the character. This method helps to achieve more spontaneity in animation, but makes it difficult to control its duration. It is more often used in hand-drawn animation creating quick chaotic scenes.

Pose to pose method is more planned with clearly arranged key pictures/poses throughout the whole scene. Using this method means better control of a size, volume, proportions of the character as well as his actions. Lead animator may set only key poses of animation and pass the rest of the work to be finished by assistant animators. In this case work resources are being used more efficiently as lead animator shall not worry about each animation frame, he may concentrate on acting and in turn assistant animators shall not worry about keyframes.

Sometimes both Straight ahead and Pose to pose methods are used together complementing each other.

5. Follow Through and Overlapping Action

Let's have a look at a running squirrel, we will see that its tail reminds us the wave motion. Besides, when squirrel’s body goes down, the tip of its tail goes up. Or imagine a whip, the motion starts with a handle then passing to the middle part of the whip and afterwards to the end of the whip. These examples vividly demonstrate what an overlapping action is. Similar is follow through motion. An example of this motion might be a longhaired running girl wearing a dress. If she suddenly stops, her dress, hair,

hands and to some extent her body will continue to move for a while. Technically follow through motion is happening when one or several body parts have stopped, and the rest of the body is still in motion. Practically nothing comes to a stop at the same time.

6. Slow-in and Slow-out

Another name for it might be speedup and slowdown. Basically nothing moves with constant speed. Imagine that you sat on a bicycle and all of a sudden you ride with a speed of 40 miles per hour. You arrived at your destination and at the same very moment your speed from 40 miles per hour drops to 0 (zero) - a full stop. Something is missing, isn't it? Well, you sat on a bicycle, then you speed up, then you reach the speed of 40 miles per hour and having reached your destination you slow down to a full stop. The same applies to almost everything - you throw a ball, start to run or jump, start driving a car, fly on a plane - abatement of start and end of the motion. Simply put, everything starts with a speed-up and ends with a slowdown.

7. Arcs

Motions of all living beings (people, animals, birds, fish, etc.) and many other objects do not happen in straight lines, but in arcs. Imagine a pendulum, its motion reflects an exact arc. The same applies to hands, legs, head and body as a whole. A perfect example is walking. Pay attention to how you move your feet. You simultaneously start to raise and move the foot and end up with lowering and a full stop. Your foot made an arc motion. Your pelvis moved in arcs as well. You may try moving your feet in a straight line for that you just need to drag them without lifting from the ground. Your pelvis most probably will further continue to move in arcs. When throwing a ball your arm will move along in an arc and flying ball will also make the same arc motion. In animation arc motion will appear more natural and appealing.

8. Secondary Action

Secondary actions are intended to complement and intensify the main action or with intent to distract or direct spectator's attention to other actions so enriching the animation and making it more appealing and solid. Imagine a student reaching for a test paper during examination, he is viewing them with uncertainty, shifting from foot to foot in doubt, his eyes wondering - this is the main action. Now imagine the same scene when a schoolboy is fidgeting with a pen, what is a secondary action. This makes the whole scene deeper and more attractive.

Secondary actions may become main actions. In an example with a schoolboy we can switch spectator's attention from the main action. For instance if the schoolboy

unintentionally bends the pen and breaks it, so our secondary action becomes the main action because the spectator switches his attention from test-papers to reaction to a broken pen. So secondary action becomes the main action.

9. Timing

This is time or number of frames you use to demonstrate an action or motion. Use less frames and your motion will be sharp and quick, use more frames and your motion will be smooth and slow. For example you are working on an animation of striking a ball. Give 4 frames for this animation and you will have a sharp and very quick strike:

- 1st frame - a foot is lifted up
- 5th frame - the foot strikes a ball
- 2nd, 3rd and 4th frames are in between frames, where the foot goes all the way from a swing to striking a ball.

Now let us consider the same animation but with different timing:

- 1st frame - a foot is lifted up
- 50th frame - the foot strikes a ball
- In frames from the 2nd to 49th the foot goes all the way from the swing to striking a ball.

With a speed of 25 frames per second the first version will take 1/5 (one fifth) of a second while duration of the second version will be 2 seconds. Correspondingly the action in the second version will be much slower and smoother.

Timing is in charge not only for speed, but also for size, weight and even character's temper. Imagine a little ant, who within 2 seconds will run a distance of about 2 inches and make about 50 steps. And now think how many steps can an elephant make in 2 seconds? Or how long does it take for an ant and an elephant to make a U-turn? The time will not be the same - this exactly is timing.

There might be variations of quick and slow timing what gives unique rhythm and appeal to an action.

10. Exaggeration

Animation is limitless and allows showing things as we want them to show different from reality. By means of exaggeration we can achieve greater expression, precision, more dynamic poses and motions. Not only primary lines of a character can be exaggerated, but also his personal traits, his behavior, condition, his motions, etc. Let us compare two boxing punches. First example is a realistic one when during the swing the character makes a slight turn taking his body into a "charge". In animation this motion will not be dynamic enough and appealing. Another example is animation with exaggeration when during a swing our character turns his body to 3/4 of a circle - what is a powerful charge, dynamic and appealing pose.

11. Solid Drawing and Solid Posing

Your character poses shall be clear and expressive, the silhouette easily read. Stick to clear shapes, watch the center of gravity, weight should be evenly distributed. Poses shall clearly express thoughts, intentions, condition, wishes and feelings of a character.

12. Appeal

Here we do not deal with cover girls or a fluffy kitten with a pink ribbon. All characters may and shall to a greater or lesser degree be appealing whether they are heroes, villains, mammoths, dinosaurs or an object. This refers to their type, nature, background and behavior. Even villain-like characters shall be charismatic and might be liked by spectators. Spectators more easily accept and understand appealing characters, they show them empathy. Even a little mouse may be so appealing that became a legend - a Mickey Mouse.

Animation Formats

The format used for the animation is entirely based around how complex the animation is, where the animation is designed to be used; such as a webpage, and how large the file is for the animation.Different formats for animations are as follows:-

1. Dynamic HTML

Dynamic HTML is a type of HTML, which is used along side client scripting languages, such as Java Script. It can be used to create simple animations such as a button and drop down menus. Although a massive downside to using Dynamic HTML is since there are various types of browsers such as Google

Chrome and FireFox, information is displayed differently meaning Dynamic HTML may not always be effective.

2. **Flash**

   Flash allows users to create animations frame by frame, including using certain materials such as bitmaps and scripting such as in ActionScript. Flash is also useful for animations since it can include the use of sound and video. It is also possible to create a website using Flash, however since the entire website uses flash it may require high internet speeds in order to load. Flash also needs a Flash player installed onto the computer in order to show content, although most computers already tend to have this installed. Flash files (FLA) have to be converted into SWF format before they can be used on the internet. The SWF format was originally composed to store animations in, including sounds, however it can also be used for other purposes such as website building and process forms.

3. **Shockwave**

   Shockwave is used more for 3D graphics and streaming videos. It is designed to work with a Director application, which can then compile various types of assets into a multimedia product, on a much larger scale then Flash.

4. **Animated GIF's**

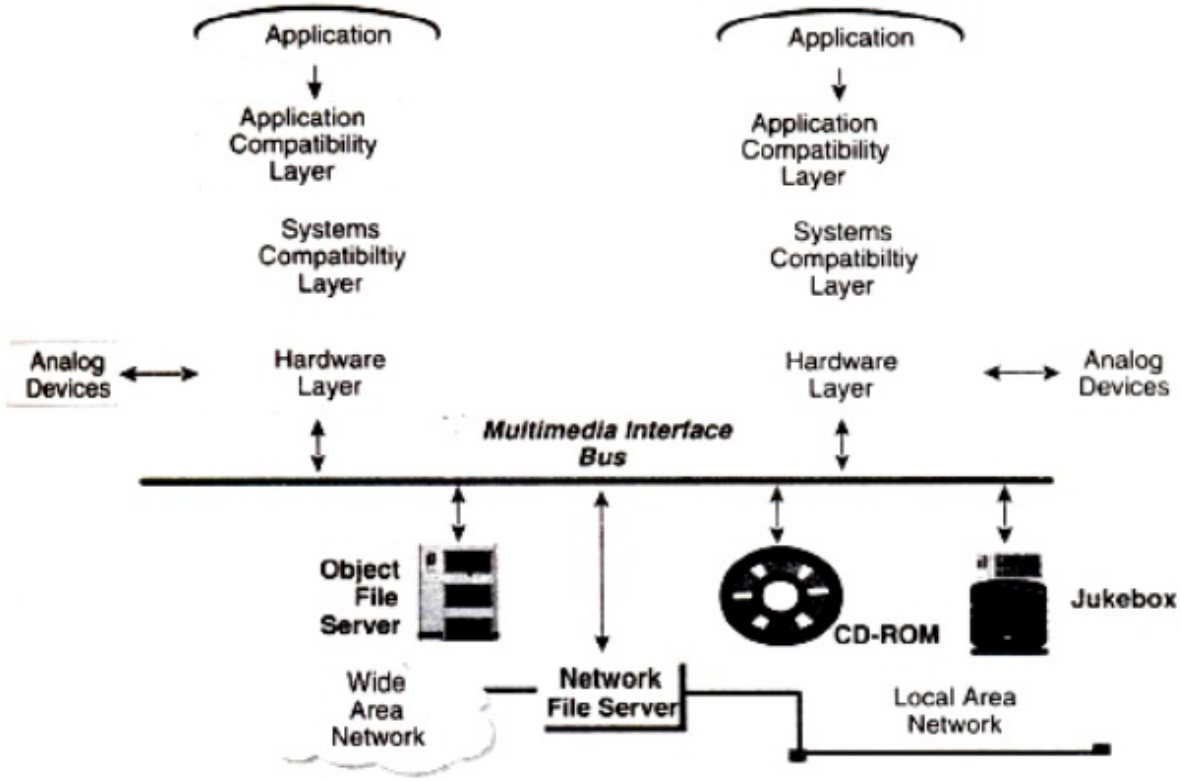
   GIF images which are animated normally have various images combined into a single GIF file. Applications, such as GIF89A, cycle through the various images in the GIF file in order to create an animation. While GIFs have a limited amount of flexibility and less amount of control compared to other formats, although since it is supported by almost every single web browser, it has become extremely popular. GIF files also tend to be smaller then other animation files.

Case Study- 3

AIM:- Study of Multimedia Architecture.

Multimedia encompasses a large variety of technologies and integration of multiple architectures interacting in real time. All of these multimedia capabilities must integrate with the standard user interfaces such as Microsoft Windows.

The following figure describes the architecture of a multimedia workstation environment.



The right side shows the new architectural entities required for supporting multimedia applications.

For each special devices such as scanners, video cameras, VCRs and sound equipment-, a software device driver is need to provide the interface from an application to the device. The GUI require control extensions to support applications such as full motion video

High Resolution Graphics Display

The various graphics standards such as MCA, GGA and XGA have demonstrated the increasing demands for higher resolutions for GUIs.

Combined graphics and imaging applications require functionality at three levels. They are provided by three classes of single-monitor architecture.

(i) VGA mixing: In VGA mixing, the image acquisition memory serves as the display source memory, thereby fixing its position and size on screen:

(ii) VGA mixing with scaling: Use of scalar ICs allows sizing and positioning of images in pre-defined windows.

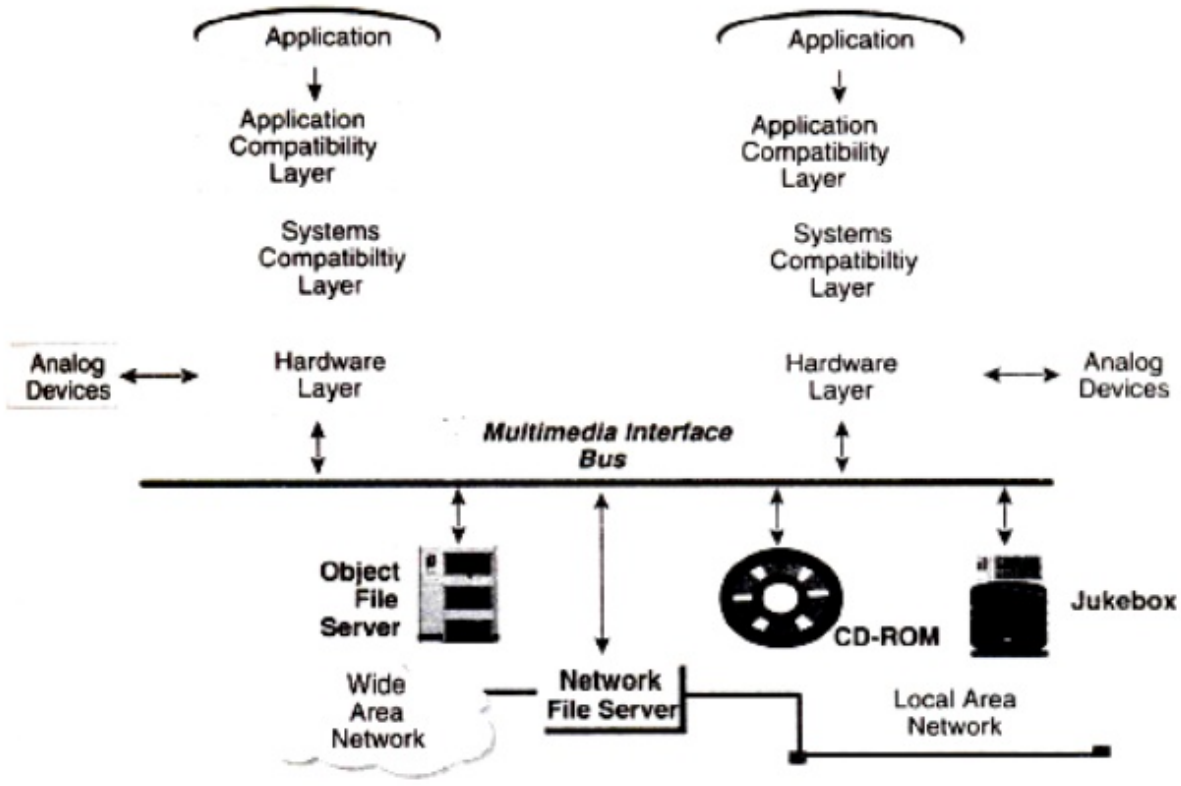
Resizing the window causes the things to be retrieved again.

(iii) Dual-buffered VGA/Mixing/Scaling: Double buffer schemes maintain the original images in a decompression buffer and the resized image in a display buffer.

The IMA Architectural Framework

The Interactive Multimedia Association has a task group to define the architectural framework for multimedia to provide interoperability. The task group has concentrated on the desktops and the servers. Desktop focus is to define the interchange formats. This format allows multimedia objects to be displayed on any work station.

The architectural approach taken by IMA is based on defining interfaces to a multimedia interface bus. This bus would be the interface between systems and multimedia sources. It provides streaming I/O service"s, including filters and translators **figure** describes the generalized architectural approach



Network Architecture for Multimedia Systems:

Multimedia systems need special networks. Because large volumes of images and video messages are being transmitted.

Asynchronous Transfer Mode technology (ATM) simplifies transfers across LANs and WANs.

Task based Multi level networking

Higher classes of service require more expensive components in the' workstations as well as in the servers supporting the workstation applications.

Rather than impose this cost on all work stations, an alternate approach is to adjust the class of service to the specific requirement for the user. This approach is to adjust the class of services according to the type of data being handled at a time also.

We call this approach task-based multilevel networking.

High speed server to server Links

1. **Duplication:** It is the process of duplicating an object that the user can manipulate. There is no requirement for the duplicated object to remain synchronized with the source (or master) object.
2. **Replication:** Replication is defined as the process of maintaining two or more copies of the same object in a network that periodically re-synchronize to provide the user faster and more reliable access to the data Replication is a complex process.
3. **Networking Standards:** The two well-known networking standards are Ethernet and token ring. ATM and FDDI are the two technologies which we are going to discuss in detail.
    - **ATM:** ATM is a acronym for Asynchronous Transfer Mode. It's topology was originally designed for broadband applications in public networks.

      ATM is a method of multiplexing and relaying (cell-switching) 53 byte cells. (48 bytes of user information and 5 bits of header information).

      Cell Switching: It is a form of fast packet switching based on the use of cells. **Cells:** Short, fixed length packets are called cells.

      ATM provides high capacity, low-latency switching fabric for data. It is independent of protocol and distances. ATM effectively manage a mix of data types, including text data, voice, images and full motion video. ATM was proposed as a means of transmitting multimedia applications over asynchronous networks.

    - **FDDI:** FDDI is an acronym of Fiber Distributed Data Interface. This FDDI network is an excellent candidate to act as the hub in a network configuration, or as a backbone that interconnects different types of LANs.

      FDDI presents a potential for standardization for high speed networks.

      The ANSI standard for FDDI allows large-distance networking. It can be used as high-performance backbone networks to complement and extend current LANs.