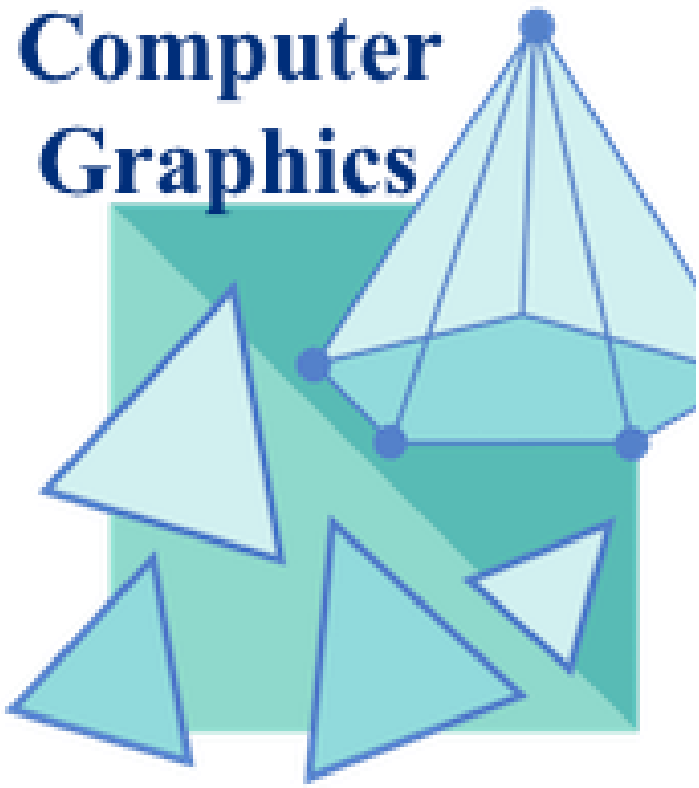
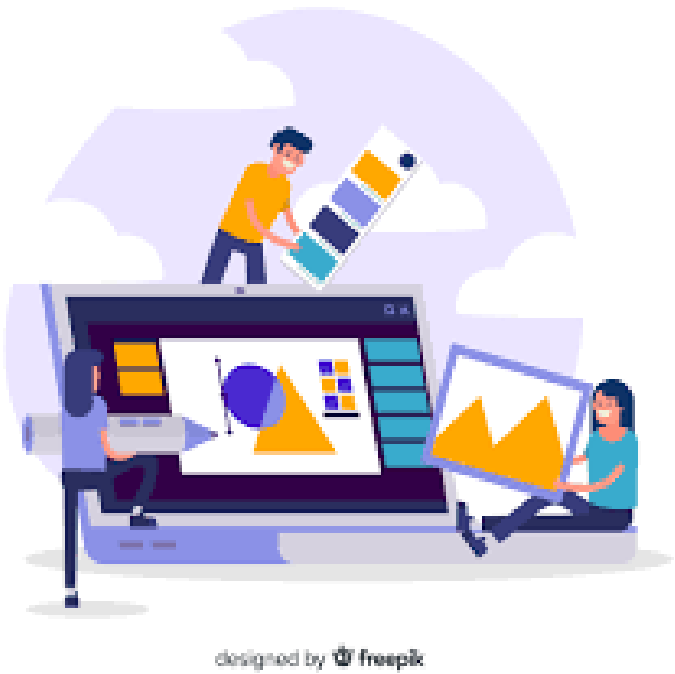
PATAN MULTIPLE CAMPUS
Patan Dhoka , Lalitpur

Lab Report

CACS 305 Computer Graphics & Animation

Submitted by
SAPHAL GHORASAINI





Tribhuvan University
Faculty of Humanities and Social Sciences

LAB REPORT
Computer Graphics & Animation

Submitted to
Ramesh Singh Saud
Department of Computer Application
Patan Multiple Campus

Submitted by

Saphal Ghorasaini
BCA 5th Semester
Roll No :-21/076

ACKNOWLEDGEMENT

We express our deep appreciation to our Teacher *(Mr. Ramesh Singh Saud)* who gave us the golden opportunity for doing this wonderful lab Report as well as helped us a lot in finalizing this Report within the limited time frame.

We would like to express our special thanks of gratitude to our Campus Chief (*Dr. Laxman Singh Kunwar)* who gave us permission for doing this wonderful lab Report. We would also like to express our hearty gratitude to all those people who directly and indirectly helped us to prepare this Report.

In spite of our herculean efforts, there might remain some errors in this Lab Report. So, we welcome critical feedback on this Report and look forward to constructive suggestions for improving the lab Report. We assure you that your invaluable comments and suggestions find their place in upcoming Reports.

Yours sincerely,

Saphal Ghorasaini

Preface

Computer and telecommunication technology that combine together has change three way we talk, word, work or deal. The world has been transformed into a small global village, where everyone on any comer is within the reach of each other. Information and communication technology (ICT) has emerged as the key ingredient of every activity.

Information is considered as the strategic resource on same manner as the capital, human resource or the machine. Every single, bunch of information is being generated, processed and stored communication. IT has been change in the world economy into knowledge of based economy rather than industry.

I have taken efforts in this lab Report. However, it would not been have been possible without the kind supports of our C-Programming sir *Mr. Ramesh Singh Saud*. I would like to extend my sincere thanks to him.

I am highly indebted to Patan Multiple Campus for their guidance and constant supervision as well as for providing necessary information regarding the project and support in the completion of the lab Report.

I would like to express my gratitude towards library and member of Patan Multiple Campus for their kind co-operation and encouragement which help me in completion of this Report

I would like to express my special gratitude and thanks to our *Co-ordinator Sir Mr. Bhoj Raj Joshi* for his support and help for our personnel development and mainly for the completion of this reports work of our Computer Graphic and Animation.

Table Of Contents

DDA line Drawing Algorithm

DDA stands for Digital Differential Analyzer. It is an incremental method of scan conversion of line. In this method calculation is performed at each step but by using results of previous steps.

DDA Algorithm:

Step1: Start

Step2: Declare x1,y1,x2,y2,dx,dy,x,y as integer variables.

Step3: Enter value of x1,y1,x2,y2.

Step4: Calculate dx = x2-x1

Step5: Calculate dy = y2-y1

Step6: If ABS (dx) > ABS (dy)
Then step = abs (dx)
Else

Step7: xinc=dx/step
yinc=dy/step
assign x = x1
assign y = y1

Step8: Set pixel (x, y)

Step9: x = x + xinc
y = y + yinc
Set pixels (Round (x), Round (y))

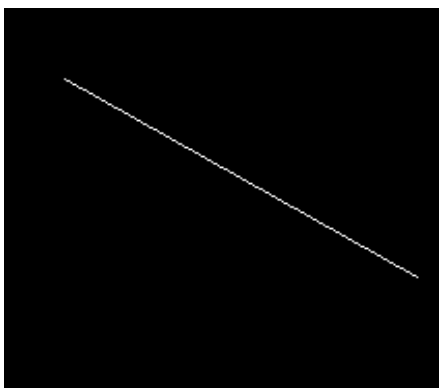
Step10: Repeat step 9 until x = x2

Step11: Stop

Lab no 1. Write a Program to implement the DDA line drawing algorithm.

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
#include<math.h>
void DDAline(int x1,int y1,int x2,int y2);
int main()
 {
   int gd=DETECT,gm;
   int x1,y1,x2,y2;
   initgraph(&gd,&gm,"");
   DDAline(30,40,200,150);
   getch();
   closegraph();
   return 0; }
void DDAline(int x1,int y1,int x2,int y2)
{ int dx, dy, steps, i;
   float xinc, yinc, x, y;
   dx=x2-x1;
   dy=y2-y1;
   if(abs(dx)>abs(dy))
      steps=abs(dy);
   else if(abs(dx)<=0)
      steps=abs(dx);
   xinc=dx/steps;
   yinc=dy/steps;
   x=x1;
   y=y1;
   putpixel(x,y,15);
   for(i=0;i<steps;i++)
{ x=x+xinc;
   y=y+yinc;
   putpixel(round(x),round(y),15);
   }
 }
```

OUTPUT:



Bresenham's Line Drawing Algorithm

This algorithm is used for scan converting a line. It was developed by Bresenham. It is an efficient method because it involves only integer addition, subtractions, and multiplication operations. These operations can be performed very rapidly so lines can be generated quickly.

Bresenham's Line Algorithm:

Step 1: Start Algorithm

Step 2: Declare variable x1,x2,y1,y2,d,i1,i2,dx,dy

Step 3: Enter value of x1,y1,x2,y2
Where x1,y1are coordinates of starting point
And x2,y2 are coordinates of Ending point

Step 4: Calculate dx = x2-x1
Calculate dy = y2-y1
Calculate i1=2*dy
Calculate i2=2*(dy-dx)
Calculate d=i1-dx

Step 5: Consider (x, y) as starting point and xendas maximum possible value of x.
If dx < 0
Then x = x2
y = y2
xend=x1
If dx > 0
Then x = x1
y = y1
xend=x2

Step 6: Generate point at (x,y)coordinates.

Step 7: Check if whole line is generated.
If x > = xend
Stop.

Step 8: Calculate co-ordinates of the next pixel
If d < 0
Then d = d + i1
If d ≥ 0
Then d = d + i2
Increment y = y + 1

Step 9: Increment x = x + 1

Step 10: Draw a point of latest (x, y) coordinates

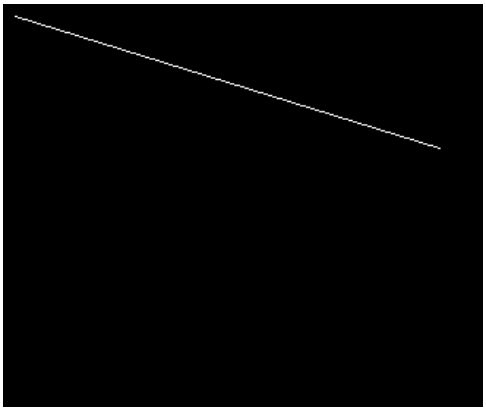
Step 11: Go to step 7

Step 12: End of Algorithm

Lab no 2. Write a Program to implement Bresenham's line drawing algorithm.

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<graphics.h>
int main()
{
   int x,y,x1,y1,x2,y2,p,dx,dy;
   int gd = DETECT,gm;
   initgraph(&gd,&gm," ");
   printf("enter the value of x1 and y1:");
   scanf("%d%d",&x1,&y1);
   printf("enter the value of x2 and y2:");
   scanf("%d%d",&x2,&y2);
   x = x1;
   y = y1;
   dx = x2-x1;
   dy = y2- y1;
   putpixel(x,y,5);
   p = 2*dy -dx;
   while(x<=x2){
      if(p<0){
      x = x+1;
      p = p+ 2*dy;
   }
   else{
      x = x+1;
      y= y+1;
      p = p+(2*dy)-(2*dx);
   }
   putpixel(x,y,5);
   }
   getch();
   closegraph();
   return 0;
}
```

OUTPUT:



Midpoint Circle Drawing Algorithm

It is based on the following function for testing the spatial relationship between the arbitrary point (x, y) and a circle of radius r centered at the origin.

Algorithm of Midpoint Circle Drawing

Step 1: Start.

Step 2: First, we allot the center coordinates (p0, q0) as follows-

P0 = 0

q0 =r

Step 3: Now, we calculate the initial decision parameter d0 –

d0 = 1 – r

Step 4: Assume, the starting coordinates = (pk, qk)

The next coordinates will be (pk+1, qk+1)

Now, we find the next point of the first octant according to the value of the decision parameter (dk).

Step 5: Now, we follow two cases-

Case 1: If

dk < 0

then

pk+1 =pk + 1

qk+1 =qk

dk+1 = dk + 2 pk+1 + 1

Case 2: If

dk >= 0

then

pk+1 =pk + 1

qk+1 =qk –1

dk+1 = dk – 2 (qk+1 + 2 pk+1)+ 1

Step 6: If the center coordinate point (p0, q0) is not at the origin (0, 0) then we will draw the points as follow-

For x coordinate = xc + p0

For y coordinate = yc + q0

{xc andyc containsthe current value of x and y coordinate}

Step 7: We repeat step 5 and 6 until we get x>=y.

Step 8: Stop.

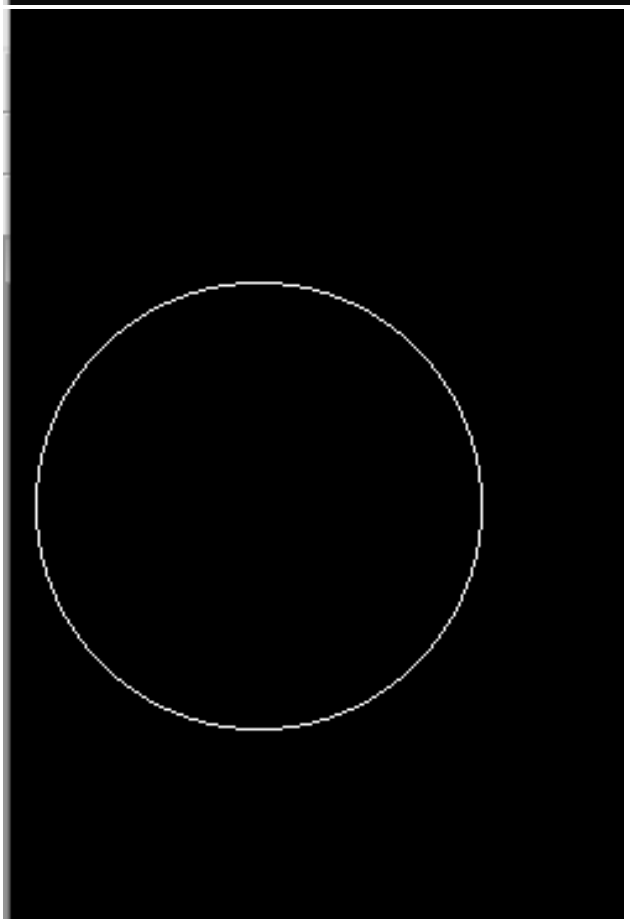
Lab no 3. Write a Program to implement mid-point circle drawing algorithm.

```
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
void drawcircle(int,int,int);
int main()
{
/* request auto detection */
int gdriver = DETECT, gmode;
int xc,yc,r;
printf("Enter the center co-ordinates:");
scanf("%d%d",&xc,&yc);
printf("Enter the radius");
scanf("%d",&r);
/* initialize graphics and local variables */
initgraph(&gdriver, &gmode, "");
drawcircle(xc,yc,r);
getch();
closegraph();
}
void drawcircle(int xc,int yc,int r) {
int p,x,y;
x=0;
y=r;
putpixel(xc+x,yc+y,15);
putpixel(xc-x,yc+y,15);
putpixel(xc+x,yc-y,15);
putpixel(xc-x,yc-y,15);
putpixel(xc+y,yc+x,15);
putpixel(xc-y,yc+x,15);
putpixel(xc+y,yc-x,15);
putpixel(xc-y,yc-x,15);
p=1-r;
while(x<y)
//because symmetry in octants
{ if(p<0) {
x=x+1;
p=p+2*x+1; }
 else
    { x=x+1;
 y=y-1;
 p=p+2*(x-y)+1;
}
putpixel(xc+x,yc+y,15);
```

```
putpixel(xc-x,yc+y,15);
putpixel(xc+x,yc-y,15);
putpixel(xc-x,yc-y,15);
putpixel(xc+y,yc+x,15);
putpixel(xc-y,yc+x,15);
putpixel(xc+y,yc-x,15);
putpixel(xc-y,yc-x,15); } }
```

OUTPUT:

```
Enter the center co-ordinates:100
200
Enter the radius90

Process returned 0 (0x0)   execution time : 16.371 s
Press any key to continue.
```



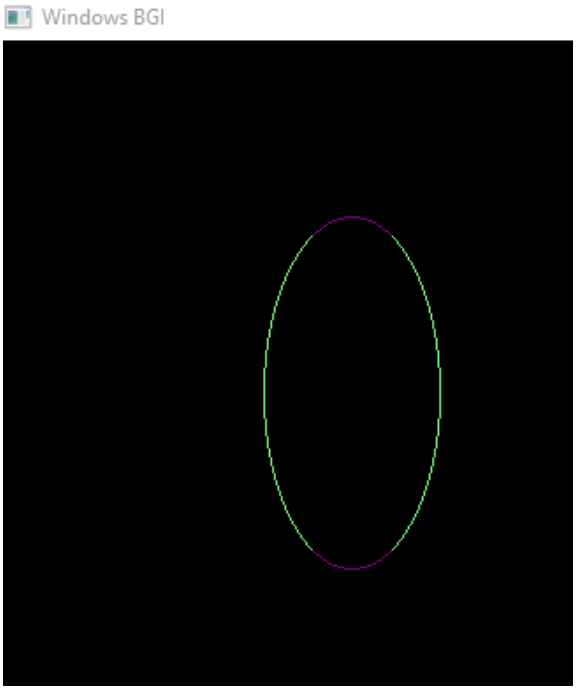
Lab no 4. Write a program to implement mid-point ellipse drawing algorithm.

```
#include<graphics.h>
#include<stdio.h>
#include<conio.h>
 void ellipse(int h,int k,int a,int b);
int main()
{
/* request auto detection */
int gdriver = DETECT, gmode;
int h,k,a,b;
   printf("Enter (h,k)");
   scanf("%d%d",&h,&k);
   printf("Enter (a,b)");
   scanf("%d%d",&a,&b);
/* initialize graphics and local variables */
initgraph(&gdriver, &gmode, "");
ellipse(h,k,a,b);
getch();
closegraph();
}
 void ellipse(int h,int k,int a,int b)
 {
   int x, y, p;
    x=0;
   y=b;
   p=(b*b)-(a*a*b)+((a*a)/4);
   while((2*x*b*b)<(2*y*a*a))
   {
       putpixel(h+x,k-y,5);
       putpixel(h-x,k+y,5);
       putpixel(h+x,k+y,5);
       putpixel(h-x,k-y,5);
       if(p<0)
       {
     x=x+1;
     p=p+b*b*(2*x+1);
       }
       else
       {
     x=x+1;
     y=y-1;
     p=p+b*b*(2*x+1)-(a*a*2*y);
       }
   }
   p=((float)x+0.5)*((float)x+0.5)*b*b+(y-1)*(y-1)*a*a-a*a*b*b;
```

```
      while(y>=0)
  {
     putpixel(h+x,k-y,10);
     putpixel(h-x,k+y,10);
     putpixel(h+x,k+y,10);
     putpixel(h-x,k-y,10);

     if(p>0)
     {
    y=y-1;
    p=p-a*a*(2*y-1);

     }
     else
     {
    y=y-1;
    x=x+1;
   p=p-a*a*(2*y-1)+b*b*2*x;
     }
  }
  getch();
  closegraph();
}
```

OUTPUT:

```
Enter (h,k)200 200 50 100
Enter (a,b)
Process returned 0 (0x0)   execution time : 10.100 s
Press any key to continue.
```

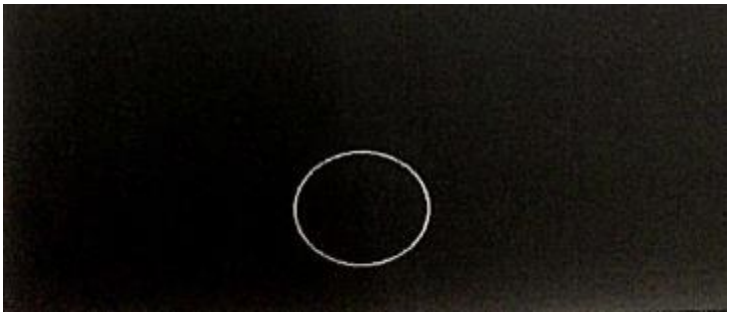


Lab no 5. Write a Program to implement the Circle drawing algorithm in polar co-ordinate form.

```
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#define color 10
void eightWaySymmetricPlot(int xc,int yc,int x,int y)
{
   putpixel(x+xc,y+yc,color);
   putpixel(x+xc,-y+yc,color);
   putpixel(-x+xc,-y+yc,color);
   putpixel(-x+xc,y+yc,color);
   putpixel(y+xc,x+yc,color);
   putpixel(y+xc,-x+yc,color);
   putpixel(-y+xc,-x+yc,color);
   putpixel(-y+xc,x+yc,color);
}
void PolarCircle(int xc,int yc,int r)
{
   int x,y,d;
   x=0;
   y=r;
   d=3-2*r;
   eightWaySymmetricPlot(xc,yc,x,y);
   while(x<=y)
 {
      if(d<=0)
{
         d=d+4*x+6;
  }
      else
 {
         d=d+4*x-4*y+10;
         y=y-1;
  }
      x=x+1;
      eightWaySymmetricPlot(xc,yc,x,y);
  }
 }
int main(void)
 {
   int gdriver = DETECT, gmode, errorcode;
   int xc,yc,r;
   initgraph(&gdriver, &gmode, "c:\\turboc3\\bgi");
errorcode = graphresult();
```

```
if (errorcode != grOk)
 {
       printf("Graphics error: %s\n", grapherrormsg(errorcode));
       printf("Press any key to halt:");
       getch();
       exit(1);
  }
printf("Enter the values of xc and yc ,that is center points of circle : ");
    scanf("%d%d",&xc,&yc);
      printf("Enter the radius of circle : ");
    scanf("%d",&r);
    PolarCircle(xc,yc,r);
    getch();
    closegraph();
  return 0;
    }
```

Output:



Lab no 6. Write a program to implement the translation of line?

```
#include<graphics.h>
#include<stdlib.h>
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
int graphdriver=DETECT,graphmode,errorcode;
int i;
int x2,y2,x1,y1,x,y;
printf("Enter the 2 line end points:");
printf("x1,y1,x2,y2");
scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
initgraph(&graphdriver,&graphmode,"c:\\tc\\bgi");
line(x1,y1,x2,y2);
printf("Enter translation co-ordinates ");
printf("x,y");
scanf("%d%d",&x,&y);
x1=x1+x;
y1=y1+y;
x2=x2+x;
y2=y2+y;
printf("Line after translation");
line(x1,y1,x2,y2);
getch();
closegraph();
}
```

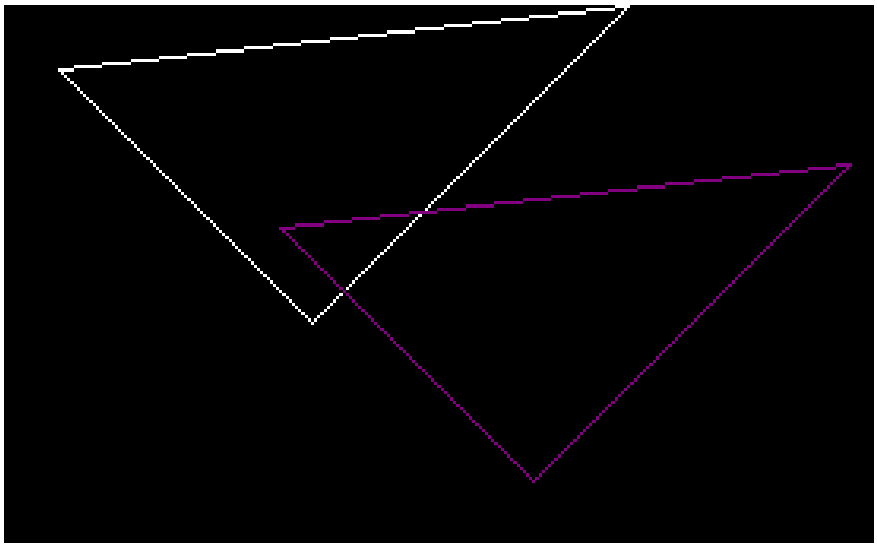
OUTPUT:

Lab no 8. Write a program to perform 2D translation.

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#define NOV 3
void drawpolygon(int VT[NOV][2], int color);
void translate(int VT[NOV][2],int IT[NOV][2], int tx,int ty);
int main()
{
   int gd=DETECT,gm;
   int tx=100,ty=100;
   int VT[NOV][2]/*={{100,150},{150,100},{200,200} }*/ ,IT[NOV][2],i;
//NO.OF VERTICES//
   printf("please enter x & y co-ordinates \n");
   for(i=0;i<NOV;i++)
   {
      printf("x%d=",i);
      scanf("%d",&VT[i][0]);
      printf("y%d=",i);
      scanf("%d",&VT[i][1]);
   }
   printf("please enter tx:");
   scanf("%d",&tx);
   printf("please enter ty:");
   scanf("%d",&ty);
   initgraph(&gd,&gm,"");
   drawpolygon(VT,15);
   getch();
   translate(VT,IT,tx,ty);
   drawpolygon(IT,5);
   getch();
   closegraph();
   return 0;
}
void drawpolygon(int VT[NOV][2], int color)
{
   int i;
   for(i=0;i<(NOV-1);i++)
   {
      setcolor(color);
      line(VT[i][0],VT[i][1],VT[i+1][0],VT[i+1][1]);
   }
   setcolor(color);
   line(VT[i][0],VT[i][1],VT[0][0],VT[0][1]);
   }
```

```
   void translate(int VT[NOV][2],int IT[NOV][2],int tx,int ty)
   {
 int i;
       for(i=0;i<NOV;i++)
       {
IT[i][0]=VT[i][0]+tx;
          IT[i][1]=VT[i][1]+ty;
       }}
```

OUTPUT:

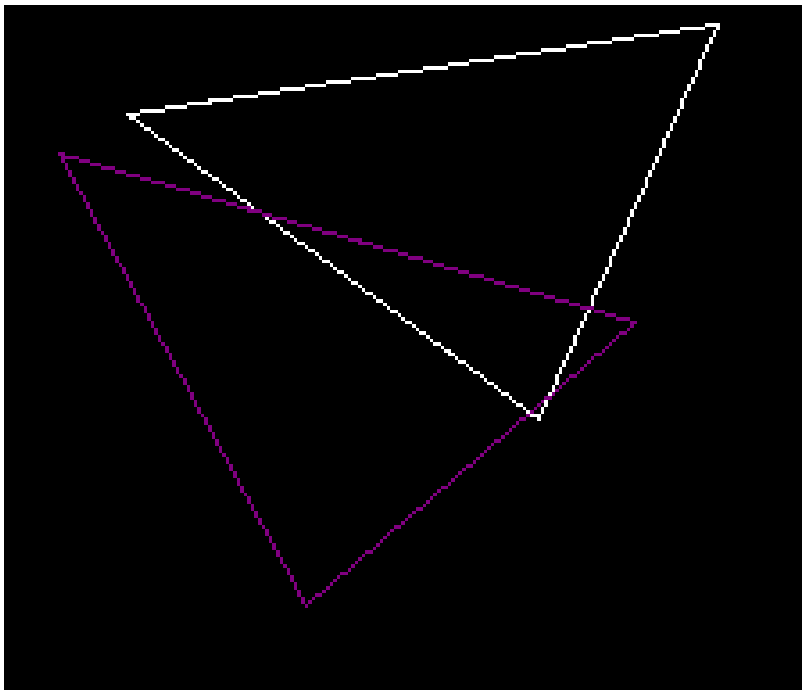


Lab no 9. Write a program to perform 2D rotation.

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>
#define NOV 3
void drawpolygon(int VT[NOV][2], int color);
void rotation(int VT[NOV][2],int IT[NOV][2],  float deg);
int main()
{
   int gd=DETECT,gm;
   float deg;
   int VT[NOV][2]/*={{100,150},{150,100},{200,200} }*/ ,IT[NOV][2],i; //NO.OF
VERTICES//
   printf("please enter x & y co-ordinates \n");
   for(i=0;i<NOV;i++)
   {
      printf("x%d=",i);
      scanf("%d",&VT[i][0]);
      printf("y%d=",i);
      scanf("%d",&VT[i][1]);
   }
   printf("please enter angle in degree:");
   scanf("%f",&deg);
   initgraph(&gd,&gm,"");
   drawpolygon(VT,15);
   getch();
   rotation(VT,IT,deg);
   drawpolygon(IT,5);
   getch();
   closegraph();
   return 0;
}
void drawpolygon(int VT[NOV][2], int color)
{
   int i;
   for(i=0;i<(NOV-1);i++)
   {
      setcolor(color);
      line(VT[i][0],VT[i][1],VT[i+1][0],VT[i+1][1]);
   }
   setcolor(color);
   line(VT[i][0],VT[i][1],VT[0][0],VT[0][1]);
   }
   void rotation(int VT[NOV][2],int IT[NOV][2],  float deg)
```

```
   {
float rad = (deg*3.14)/180;
      int i;
for(i=0;i<NOV;i++)
      {
IT[i][0]=VT[i][0]*cos(rad) - VT[i][1]*sin(rad);
         IT[i][1]=VT[i][0]*sin(rad) + VT[i][1]*cos(rad);
      }
}
```

OUTPUT:

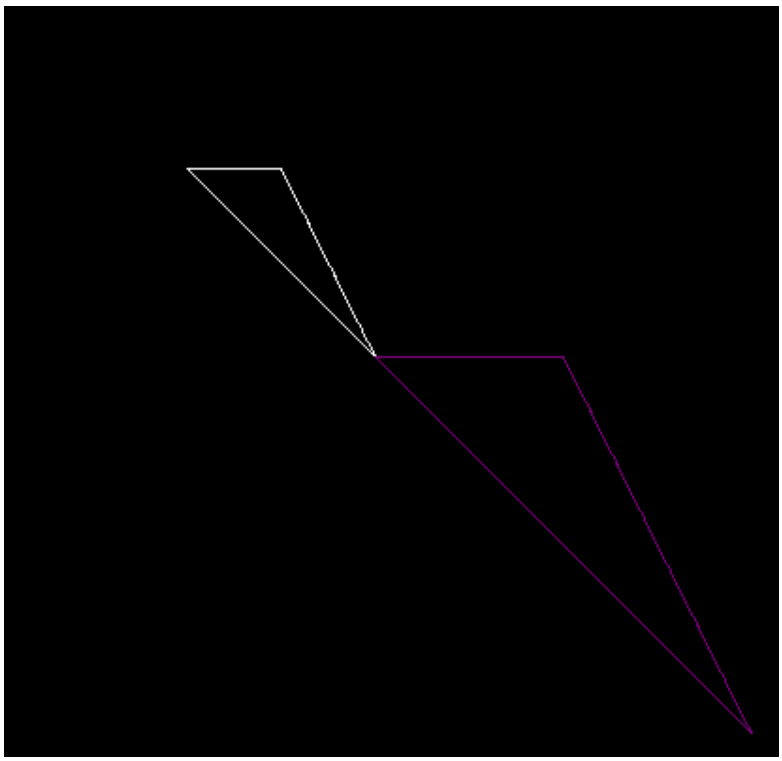


Lab no 10. Write a program to perform 2D scaling.

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#define NOV 3
void drawpolygon(int VT[NOV][2], int color);
void scaling(int VT[NOV][2],int IT[NOV][2],  int sx,int sy);
int main()
{
   int gd=DETECT,gm;
   int sx=2,sy=2;
   int VT[NOV][2]/*={{100,150},{150,100},{200,200} }*/ ,IT[NOV][2],i; //NO.OF
VERTICES//
   printf("please enter x & y co-ordinates \n");
   for(i=0;i<NOV;i++)
   {
      printf("x%d=",i);
      scanf("%d",&VT[i][0]);
      printf("y%d=",i);
      scanf("%d",&VT[i][1]);
   }
   printf("please enter sx:");
   scanf("%d",&sx);
   printf("please enter sy:");
   scanf("%d",&sy);
   initgraph(&gd,&gm,"");
   drawpolygon(VT,15);
   getch();
   scaling(VT,IT,sx,sy);
   drawpolygon(IT,5);
   getch();
   closegraph();
   return 0;
}
void drawpolygon(int VT[NOV][2], int color)
{
   int i;
   for(i=0;i<(NOV-1);i++)
   {
      setcolor(color);
      line(VT[i][0],VT[i][1],VT[i+1][0],VT[i+1][1]);
   }
   setcolor(color);
   line(VT[i][0],VT[i][1],VT[0][0],VT[0][1]);
   }
```

```
   void scaling(int VT[NOV][2],int IT[NOV][2],int sx,int sy)
   {
int i;
       for(i=0;i<NOV;i++)
       {
 IT[i][0]=VT[i][0]*sx;
          IT[i][1]=VT[i][1]*sy;
       }
}
```

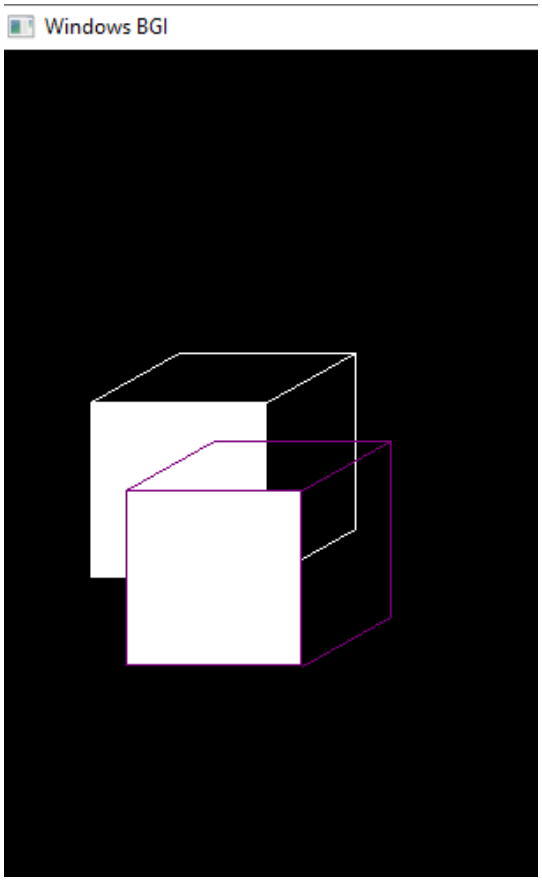
OUTPUT:



Lab no 11. Write a Program to perform 3D basic transformation.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
int main()
{
  int tx=20,ty=50,tz=0;
  int gd=DETECT, gm;
int left=50,top=200,right=150,bottom=300,depth=50;
/*printf("enter the coordinates of left,top,right,bottom and depth \n");
scanf("%d%d%d%d%d",&left,&top,&right,&bottom, &depth);
printf("enter the value of tx,ty \n");
scanf("%d%d",&tx, &ty);*/
initgraph(&gd,&gm,"");
bar3d(left,top,right,bottom,depth,1);
getch();
setcolor(5);
bar3d(left+tx,top+ty,right+tx,bottom+ty,depth,1);
getch();
return 0;
}
```

OUTPUT:



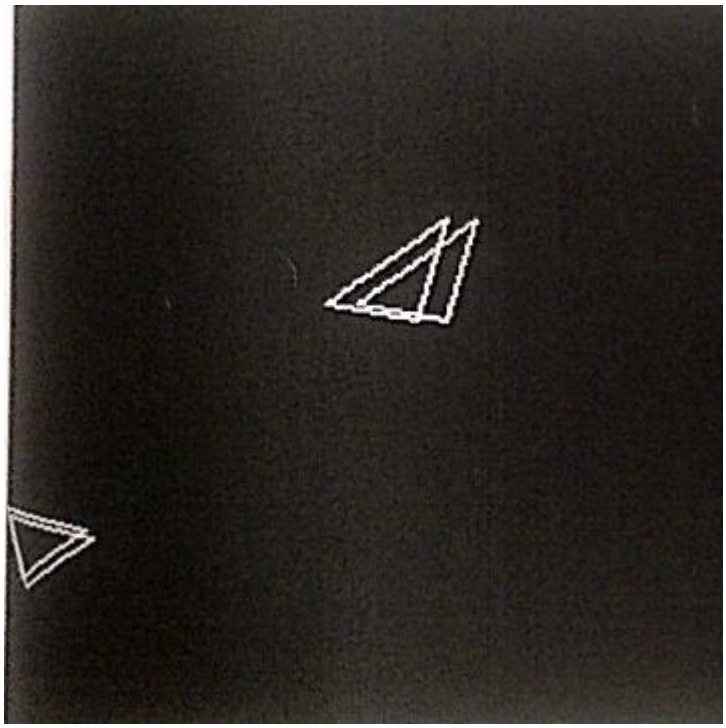
Lab no 7. Write a Program to implement composite Transformation?

```
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
int xa,xb,xc,ya,yb,yc,y1a,y1b,y1c,x1a,x1b,x1c,x2a,x2b,x2c,y2a,y2b,y2c;
int x3a,x3b,x3c,y3a,y3b,y3c,x4a,x4b,x4c,y4a,y4b,y4c,x5a,x5b,x5c,y5a,y5b,y5c;
int tx,shx,t,ch,shy;
float ang,theta,sx,sy;
int main(void)
{
int gdriver = DETECT, gmode, errorcode;
initgraph(&gdriver, &gmode,"C:\\TC\\BGI"); /* request for auto detection*/
printf("\n\t\t\t 2D Composite Transformations");
printf("\n\n Enter all coordinates values :");
scanf("%d %d %d %d %d %d",&xa,&ya,&xb,&yb,&xc,&yc);
printf("\n\n The original Image"); /* get the coordinates for the original image*/
line(xa,ya,xb,yb); /* draw the original image*/
line(xb,yb,xc,yc);
line(xc,yc,xa,ya);
printf("\n\n Enter the value tranlsation factor :"); /* get the translation factor*/
scanf("%d",&tx);
printf("\n\n After Translation ");
x1a=xa+tx;
x1b=xb+tx;
x1c=xc+tx;
y1a=ya;
y1b=yb;
y1c=yc;
line(x1a,y1a,x1b,y1b); /* image after translation*/
line(x1b,y1b,x1c,y1c);
line(x1c,y1c,x1a,y1a);
delay(1);
printf("\n\n Next Operation is Rotation");
printf("\n\n Enter the rotation angle :"); /* get the angle of rotation*/
scanf("%f",&ang);
theta=((ang*3.14)/180); /* convert the angle*/
x2a=x1a*cos(theta)-y1a*sin(theta);
y2a=x1a*sin(theta)+y1a*cos(theta);
x2b=x1b*cos(theta)-y1b*sin(theta);
y2b=x1b*sin(theta)+y1b*cos(theta);
x2c=x1c*cos(theta)-y1c*sin(theta);
y2c=x1c*sin(theta)+y1c*cos(theta);
printf("\n\n After Rotation "); /* the rotated object*/
```

```
line(x2a,y2a,x2b,y2b);
line(x2b,y2b,x2c,y2c);
line(x2c,y2c,x2a,y2a);
delay(1);
printf("\n\n Next Operation is Scaling"); /* get the scale factor*/
printf("\n\n Enter the Scale factor :");
scanf("%f %f",&sx,&sy);
x3a=x2a+sx; /* modify the objects coordinates based on the scale factor*/
y3a=y2a+sy;
x3b=x2b+sx;
y3b=y2b+sy;
x3c=x2c+sx;
y3c=y2c+sy;
printf("\n\n After Scaling ");
line(x3a,y3a,x3b,y3b);
line(x3b,y3b,x3c,y3c);
line(x3c,y3c,x3a,y3a);
delay(1);
printf("\n\n Next Operation is Shearing");
printf("\n\n Enter 1 for x-axis \n 2 for y-axis: "); /* get the choice of shearing in the x
or y axis*/
scanf("%d",&ch);
if(ch==1) {
printf("\n\n Enter the x-SHEAR (^.^) Value: ");
scanf("%d",&shx);}
else{
printf("\n\n Enter the y-SHEAR (^.^) Value: ");
scanf("%d",&shy);}
if(ch==1){
x3a=x3a+shx*y3a;
y4a=y3a;
x3b=x3a+shx*y3a;
y4b=y3b;
x3c=x3a+shx*y3a;
y4c=y3c;}
else{
x4a=x3a;
y3a=y3a+shy*x3a;
x4b=x3b;
y3b=y3b+shy*x3b;
x4c=x3c;
y3c=y3c+shy*x3c;}
printf("\n\n After Shearing "); /* draw the final object after shearing*/
line(x3a,y3a,x3b,y3b);
line(x3b,y3b,x3c,y3c);
```

```
line(x3c,y3c,x3a,y3a);
delay(1);
printf("\n\n Next Operation is Reflection");
t=abs(y3a-y3c); /* calculate the value for reflection*/
x5a=x3a;
x5b=x3b;
x5c=x3c;
y5a=y3a+10+(2*t);
y5b=y3b+10;
y5c=y3c+10;
printf("\n\n After Reflection "); /* the final object after all the transformations*/
line(x5a,y5a,x5b,y5b);
line(x5b,y5b,x5c,y5c);
line(x5c,y5c,x5a,y5a);
getch();
closegraph();
return 0;
}
```

OUTPUT:

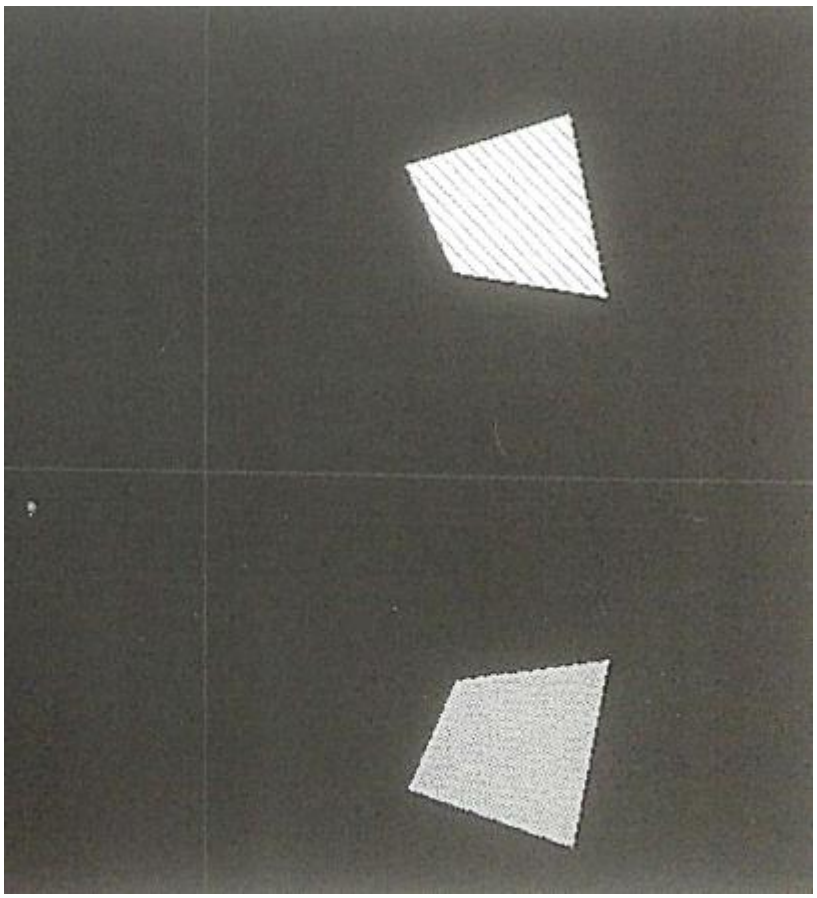


Lab no 12. Write a Program to implement the reflection transformation?

```
#include <conio.h>
#include <graphics.h>
#include <stdio.h>
void main(){
    int gm, gd = DETECT, ax, x1 = 100;
   int x2 = 100, x3 = 200, y1 = 100;
   int y2 = 200, y3 = 100;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
     line(getmaxx() / 2, 0, getmaxx() / 2,
        getmaxy());
   line(0, getmaxy() / 2, getmaxx(),
        getmaxy() / 2);
     printf("Before Reflection Object  in 2nd Quadrant");
     setcolor(14);
   line(x1, y1, x2, y2);
   line(x2, y2, x3, y3);
   line(x3, y3, x1, y1);
   getch();

   // After reflection
   printf("\nAfter Reflection");
     setcolor(4);
   line(getmaxx() - x1, getmaxy() - y1,
        getmaxx() - x2, getmaxy() - y2);
     line(getmaxx() - x2, getmaxy() - y2,
        getmaxx() - x3, getmaxy() - y3);
     line(getmaxx() - x3, getmaxy() - y3,
        getmaxx() - x1, getmaxy() - y1);
     // Reflection along x-axis i.e.,

   // in 1st quadrant
   setcolor(3);
   line(getmaxx() - x1, y1,
        getmaxx() - x2, y2);
   line(getmaxx() - x2, y2,
        getmaxx() - x3, y3);
   line(getmaxx() - x3, y3,
        getmaxx() - x1, y1);
     // Reflection along y-axis i.e.,

   // in 3rd quadrant
   setcolor(2);
   line(x1, getmaxy() - y1, x2,
        getmaxy() - y2);
```

```
    line(x2, getmaxy() - y2, x3,
        getmaxy() - y3);
    line(x3, getmaxy() - y3, x1,
        getmaxy() - y1);
    getch();
      // Close the graphics
    closegraph();
}
```

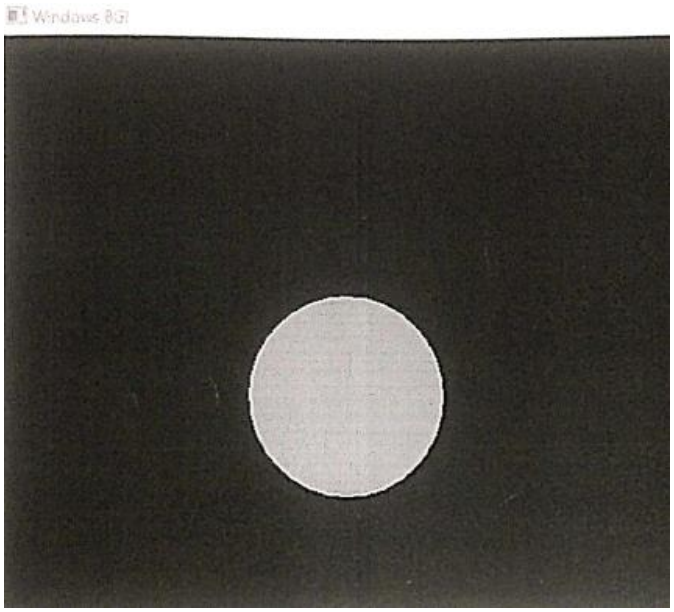
OUTPUT:



Lab no 13. Write a Program to implement the Boundary fill algorithm?

```
#include <graphics.h>
// Function for 4 connected Pixel
void boundaryFill4(int x, int y, int fill_color,int boundary_color){
   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);    }}
//driver code
int main(){
//gm is graphisc mode which is a computer display mode that generates image using
pixels.
//DETECT is a macro define in “graphic.h” header file
   int gd = DETECT, gm;
//initgraph initializes the graphics system by loading a graphics driver from disk
   initgraph(&gd, &gm, "");
   int x = 250, y = 200, radius = 50;
//circle function
   circle(x, y, radius);
//Function calling
  boundaryFill4(x, y, 6, 15);
   delay(10000);
   getch();
//closegraph function the graphics mode and deallocates
   closegraph();
return 0;}
```

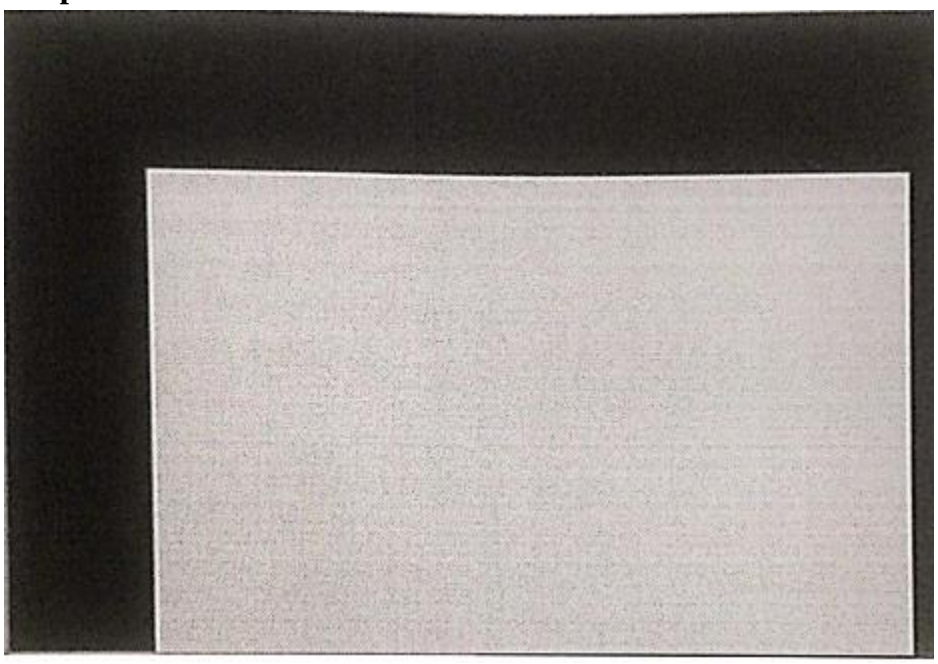
Output:



Lab no 14. Write a Program to imple1nent the Flood fill algorithm?

```
#include <graphics.h>
#include <stdio.h>
void flood(int x, int y, int new_col, int old_col){
    if (getpixel(x, y) == old_col) {
        putpixel(x, y, new_col);
        flood(x + 1, y, new_col, old_col);
        flood(x - 1, y, new_col, old_col);
        flood(x, y + 1, new_col, old_col);
        flood(x, y - 1, new_col, old_col);   }}
int main(){
    int gd, gm = DETECT;
    initgraph(&gd, &gm, "");
    int top, left, bottom, right;
    top = left = 50;
    bottom = right = 300;
    rectangle(left, top, right, bottom);
    int x = 51;
    int y = 51;
    int newcolor = 12;
    int oldcolor = 0;
    flood(x, y, newcolor, oldcolor);
    getch();
    return 0;}
```

Output:

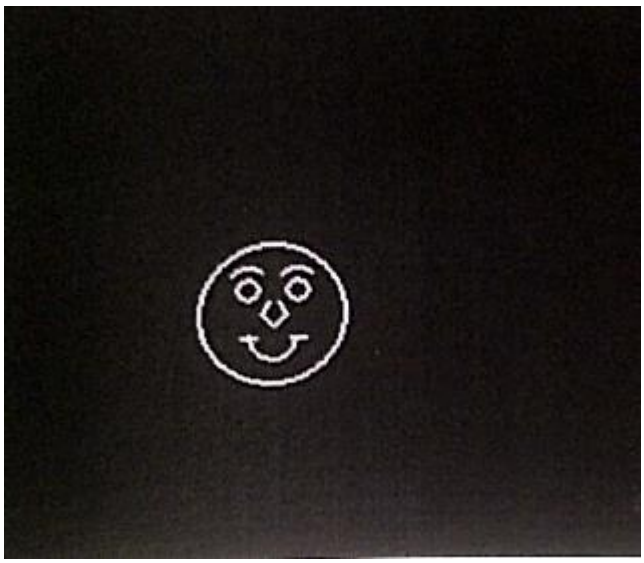


Lab no 15. Write a Program to animate the text?

```
#include<stdio.h>
#include<math.h>
#include<conio.h>
#include<graphics.h>
#define round(val) (int)(val+0.5)
void main() {
  int gd = DETECT, gm, sx, sy, tx, ty;
  char text[50];
  void move(int, int, int, int, char[]);
  printf("Enter the text:");
  scanf("%s", text);
  printf("Enter the initial points:");
  scanf("%d%d", &sx, &sy);
  printf("Enter the TARGET points:");
  scanf("%d%d", &tx, &ty);
  initgraph(&gd, &gm, "");
  outtextxy(sx, sy, text);
  move(sx, sy, tx, ty, text);
  getch();
  closegraph();
}
void move(int sx, int sy, int tx, int ty, char text[50]) {
  int dx = tx - sx, dy = ty - sy, steps, k;
  float xin, yin, x = sx, y = sy;
  getch();
  if (abs(dx) > abs(dy))
    steps = abs(dy);
  else
    steps = abs(dy);
  xin = dx / (float) steps;
  yin = dy / (float) steps;
  for (k = 0; k < steps; k++) {
    cleardevice();
    x += xin;
    y += yin;
    setcolor(15);
    outtextxy(round(x), round(y), text);
    delay(50);
  }
}
```

Lab no 16. Write a Program to implement Smiley face cartoon?

```
#include <conio.h>
#include <graphics.h>
#include <stdio.h>
int main()
{
   int gr = DETECT, gm;
   initgraph(&gr, &gm, "C:\\Turboc3\\BGI");
   setcolor(YELLOW);
   circle(300, 100, 40);
   setfillstyle(SOLID_FILL, YELLOW);
   floodfill(300, 100, YELLOW);
   setcolor(BLACK);
   setfillstyle(SOLID_FILL, BLACK);
   fillellipse(310, 85, 2, 6);
   fillellipse(290, 85, 2, 6);
   ellipse(300, 100, 205, 335, 20, 9);
   ellipse(300, 100, 205, 335, 20, 10);
   ellipse(300, 100, 205, 335, 20, 11);
   getch();
   closegraph();
   return 0;
}
```



OpenGL Basics

OpenGL is a software API to graphics hardware. It is designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms. It has no windowing commands, as well as no high-level commands for describing models of three-dimensional objects. The OpenGL Utility Library (GLU) provides many of the modeling features, such as quadric surfaces and NURBS (Non-uniform rational Basis Spline) curves and surfaces.

It is easy to use and close enough to the hardware to get excellent performance. It focusses on rendering

OpenGL functions are of two types

Primitive generating:

It can cause output if primitive is visible. The vertices are processed and appearance of primitive are controlled by the state

State changing:

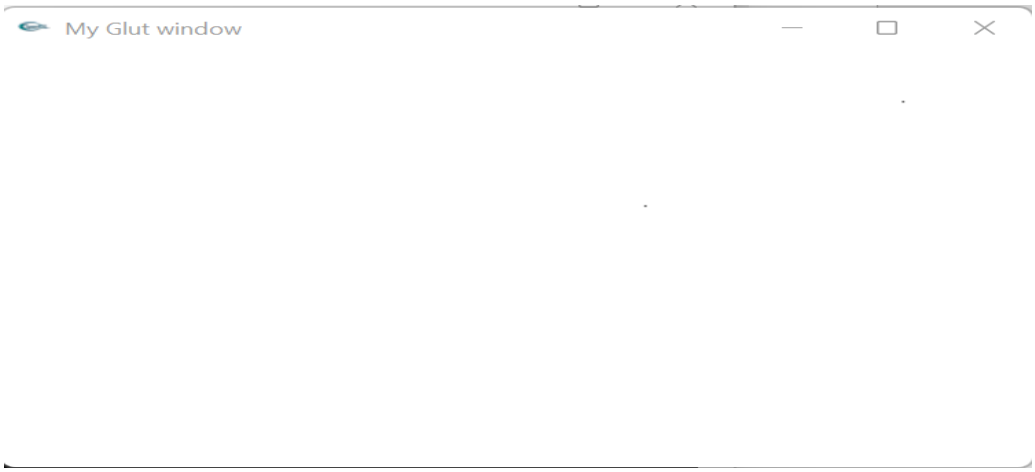
The OpenGL works as state machine once the state is set the system works unless the state is not altered. There are two types of state functions

- Transformation functions
- Attribute functions

Lab no 17. Write a program to implement Point drawing technique in OpenGl

```
#include<GL/glut.h>
#include<stdio.h>
void display()
{
glClear(GL_COLOR_BUFFER_BIT);
glBegin(GL_POINTS);
glColor3f(0.0,0.0,0.0);
glVertex3f(0.25,0.25,0.0);
glVertex3f(0.75,0.75,0.0);
glEnd();
glutSwapBuffers();
}
int main(int argc, char** argv)
{
   glutInit(&argc,argv);
   glutInitDisplayMode(GLUT_RGBA|GLUT_DOUBLE);
   glutInitWindowPosition(600,100);
   glutInitWindowSize(500,300);
   glutCreateWindow(" My Glut window");
  glClearColor(1.0,1.0,1.0,1.0);
   glutDisplayFunc(display);
   glutMainLoop();
}
```

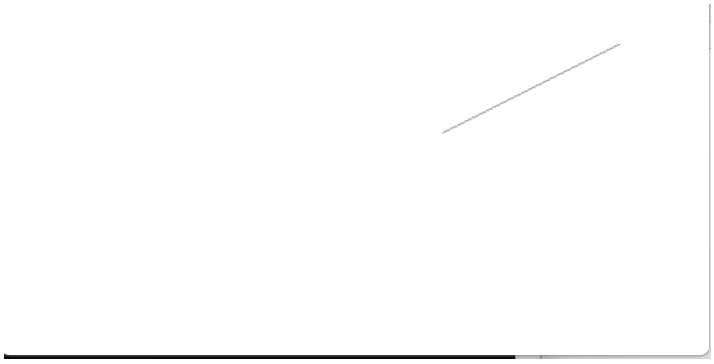
OUTPUT



Lab no 18. Write a program to implement Line drawing technique in OpenGl

```
#include<GL/glut.h>
#include<stdio.h>
void display()
{
glClear(GL_COLOR_BUFFER_BIT);
glBegin(GL_LINES);
glColor3f(.5,0.5,0.5);
glVertex3f(0.25,0.25,0.0);
glVertex3f(0.75,0.75,0.0);
glEnd();
glutSwapBuffers();
}
int main(int argc, char** argv)
{
   glutInit(&argc,argv);
   glutInitDisplayMode(GLUT_RGBA|GLUT_DOUBLE);
   glutInitWindowPosition(600,100);
   glutInitWindowSize(500,300);
   glutCreateWindow(" My Glut window");
  glClearColor(1.0,1.0,1.0,1.0);
   glutDisplayFunc(display);
   glutMainLoop();
}
```

OUTPUT:

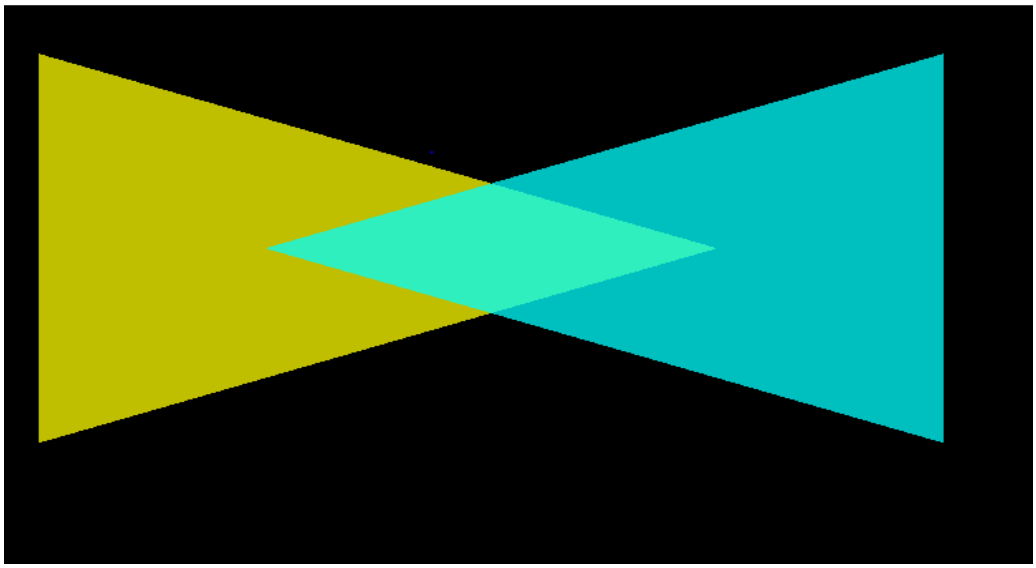


Lab no 19. Write a program implement Triangle drawing technique in OpenGl

```
#include <GL/glut.h>
#include <stdlib.h>
static int leftFirst = GL_TRUE;
 static void init(void)
{
  glEnable (GL_BLEND);
  glBlendFunc (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
  glShadeModel (GL_FLAT);
  glClearColor (0.0, 0.0, 0.0, 0.0);
}
static void drawLeftTriangle(void)
{
  glBegin (GL_TRIANGLES);
     glColor4f(1.0, 1.0, 0.0, 0.75);
     glVertex3f(0.1, 0.9, 0.0);
     glVertex3f(0.1, 0.1, 0.0);
     glVertex3f(0.7, 0.5, 0.0);
  glEnd();
}
static void drawRightTriangle(void)
{
  glBegin (GL_TRIANGLES);
     glColor4f(0.0, 1.0, 1.0, 0.75);
     glVertex3f(0.9, 0.9, 0.0);
     glVertex3f(0.3, 0.5, 0.0);
     glVertex3f(0.9, 0.1, 0.0);
  glEnd();
}
void display(void)
{
  glClear(GL_COLOR_BUFFER_BIT);

  if (leftFirst) {
```

```
      drawLeftTriangle();
      drawRightTriangle();
    }
    else {
      drawRightTriangle();
      drawLeftTriangle();
    }
  glFlush();
}
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (200, 200);
   glutCreateWindow (argv[0]);
   init();
   glutDisplayFunc (display);
   glutMainLoop();
   return 0;
}
```

OUTPUT:



Lab no 20. Write a program implement Triangle-Fan drawing technique in OpenGl

```
#include<GL/glut.h>
#include<stdio.h>
void display()
{
glClear(GL_COLOR_BUFFER_BIT);
glBegin(GL_TRIANGLE_FAN);
glColor3f(1.0,0.5,0.5);
glVertex3f(-0.5,-0.5,0.0);
glVertex3f(0.0,-0.5,0.0);
glVertex3f(0.75,0.75,0.0);
glColor3f(0.5,1.0,0.5);
glVertex3f(1.0,0.5,0.5);
glEnd();
glutSwapBuffers();
}
int main(int argc, char** argv)
{
   glutInit(&argc,argv);
   glutInitDisplayMode(GLUT_RGBA|GLUT_DOUBLE);
   glutInitWindowPosition(600,100);
   glutInitWindowSize(500,300);
   glutCreateWindow(" My Glut window");
  glClearColor(0.5,0.5,1.0,1.0);
   glutDisplayFunc(display);
   glutMainLoop();
}
```

OUTPUT: