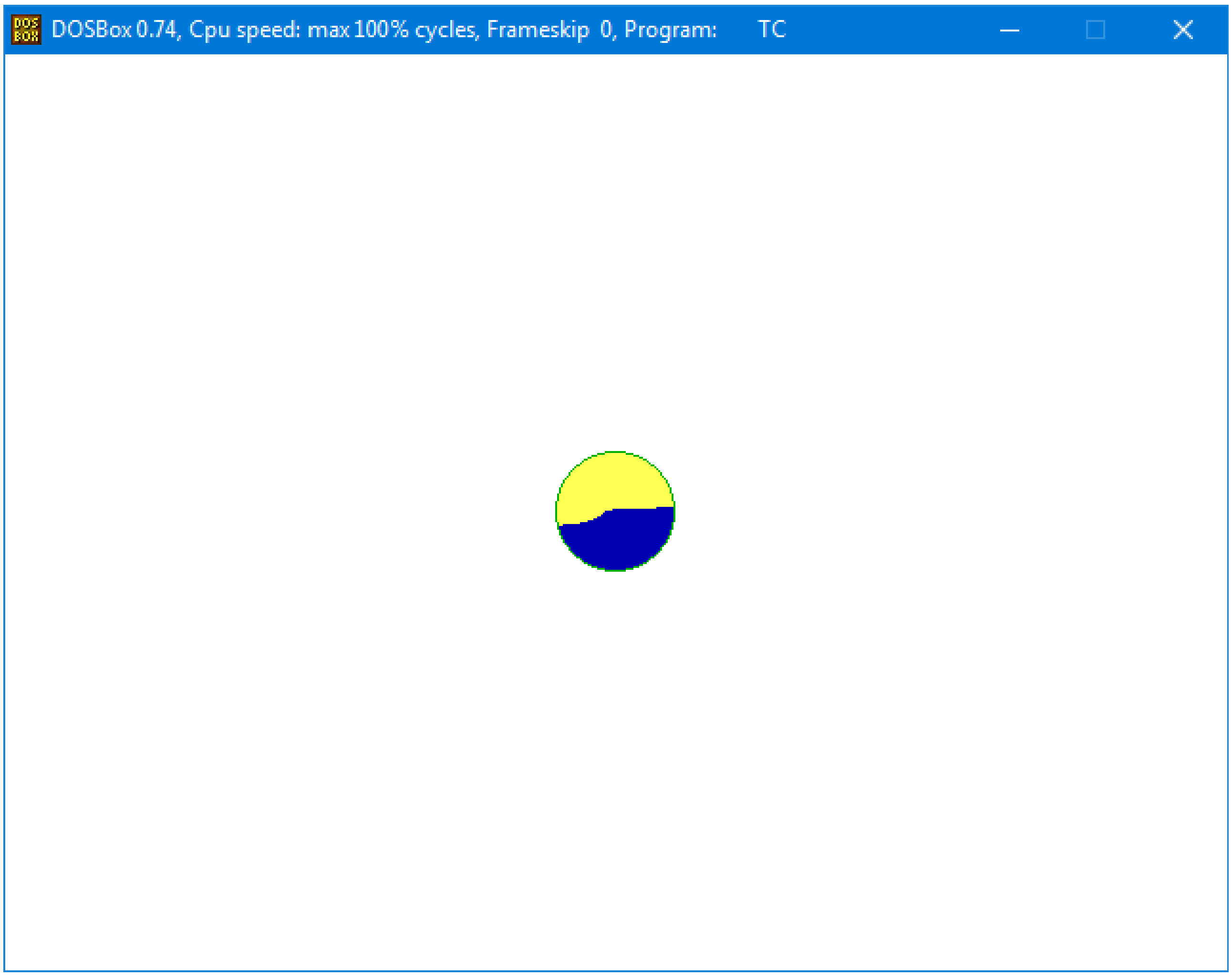
EXPERIMENT 5:BoundaryFill and FloodFill Algorithm & ScanLine Algorithm

CODE(BoundaryFill and FloodFill Algorithm):

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<time.h>

void BOUNDRYFILL(int x,int y,int boundry_color,int new_color)
{
   if((getpixel(x,y)!=boundry_color)&&(getpixel(x,y)!=new_color))
   {
      putpixel(x,y,new_color);
      delay(10);
      BOUNDRYFILL(x+1,y,boundry_color,new_color);
      BOUNDRYFILL(x,y+1,boundry_color,new_color);
      BOUNDRYFILL(x-1,y,boundry_color,new_color);
      BOUNDRYFILL(x,y-1,boundry_color,new_color);
   }
}

void FLOODFILL(int x,int y,int old_color,int new_color)
{
   if(getpixel(x,y)==old_color)
   {
      putpixel(x,y,new_color);
      delay(10);
      FLOODFILL(x+1,y,old_color,new_color);
      FLOODFILL(x,y+1,old_color,new_color);
      FLOODFILL(x-1,y,old_color,new_color);
      FLOODFILL(x,y-1,old_color,new_color);
   }
}

void main()
{
    int gd=DETECT,gm;
    int new_color;
    int option,midx,midy,rx;

    initgraph(&gd,&gm,"C:/TC/BGI");
    setbkcolor(WHITE);
    setcolor(GREEN);

    midx=getmaxx()/2;
    midy=getmaxy()/2;
    rx=midx/10;
    circle(midx,midy,rx);
    BOUNDRYFILL(midx,midy,GREEN,YELLOW);
    FLOODFILL(midx,midy,YELLOW,BLUE);

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

OUTPUT(<u>BoundaryFill and FloodFill Algorithm)</u>:

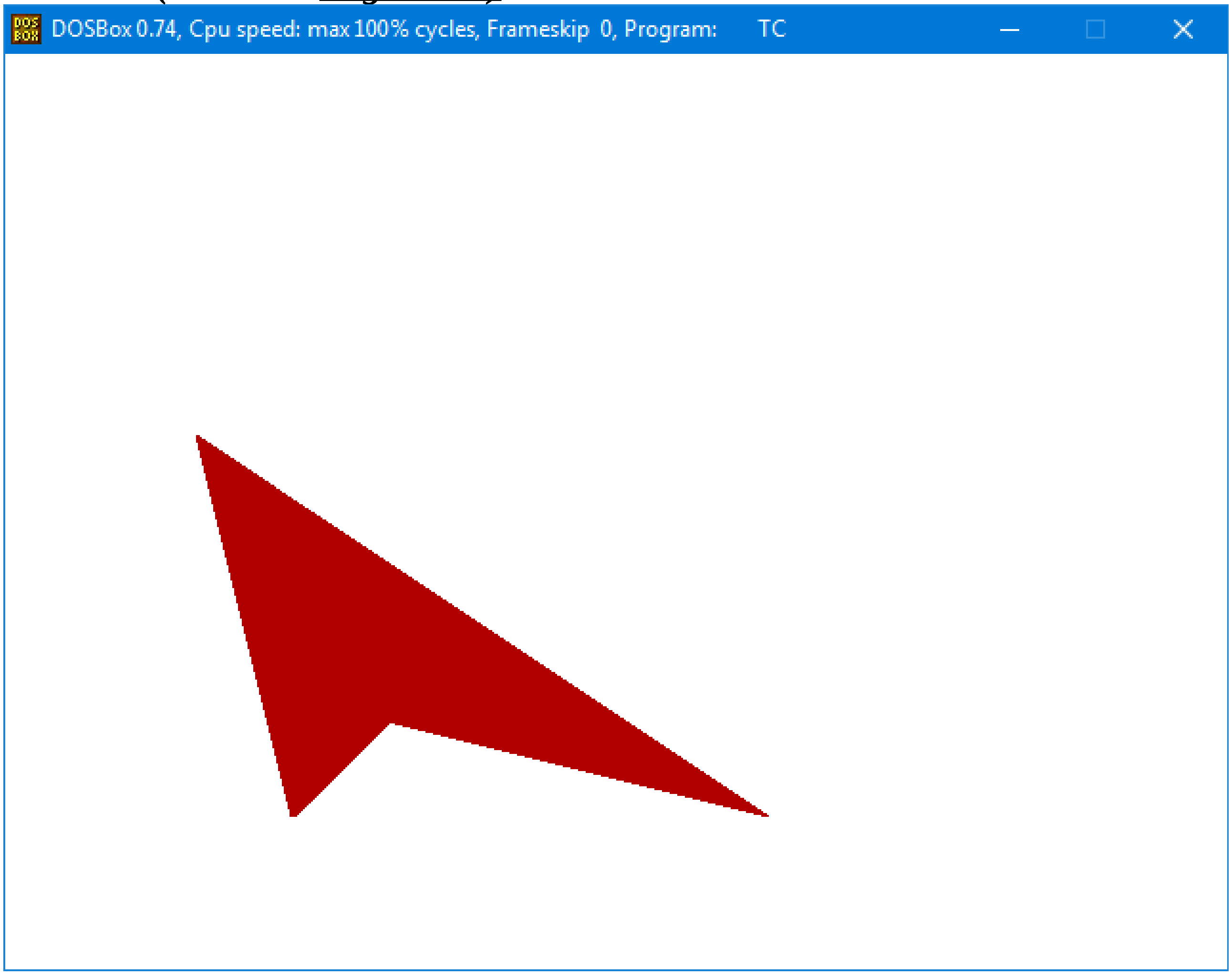

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

CODE(ScanLine <u>Algorithm)</u>:

```
#include <stdio.h>
#include<time.h>
#include <conio.h>
#include <graphics.h>
void main()
{
   int n,i,j,k,gd,gm,dy,dx;
   int x,y,temp;
   int a[20][2],xi[20];
   float slope[20];

   clrscr();
```

```
printf("\n\n\tEnter the no. of edges of polygon : ");
scanf("%d",&n);
printf("\n\n\tEnter the cordinates of polygon :\n\n\n ");

for(i=0;i<n;i++)
{
   printf("\tX%d Y%d : ",i,i);
   scanf("%d %d",&a[i][0],&a[i][1]);
}

a[n][0]=a[0][0];
a[n][1]=a[0][1];

detectgraph(&gd,&gm);
initgraph(&gd,&gm,"C:/TC/BGI");
setbkcolor(WHITE);
/*- draw polygon -*/

for(i=0;i<n;i++)
   line(a[i][0],a[i][1],a[i+1][0],a[i+1][1]);
getch();

for(i=0;i<n;i++)
{
   dy=a[i+1][1]-a[i][1];
   dx=a[i+1][0]-a[i][0];
   if(dy==0)
      slope[i]=1.0;
   if(dx==0)
      slope[i]=0.0;
   if((dy!=0)&&(dx!=0)) /*- calculate inverse slope -*/
      slope[i]=(float) dx/dy;
}
for(y=0;y<480;y++)
{
   k=0;
   for(i=0;i<n;i++)
   {
      if(((a[i][1]<=y)&&(a[i+1][1]>y))||((a[i][1]>y)&&(a[i+1][1]<=y)))
      {
         xi[k]=(int)(a[i][0]+slope[i]*(y-a[i][1]));
         k++;
      }
   }
   for(j=0;j<k-1;j++) /*- Arrange x-intersections in order -*/
   {
      for(i=0;i<k-1;i++)
      {
```

```
                if(xi[i]>xi[i+1])
                {
                    temp=xi[i];
                    xi[i]=xi[i+1];
                    xi[i+1]=temp;
                }
            }
        }
        setcolor(RED);
        for(i=0;i<k;i+=2)
        {
            line(xi[i],y,xi[i+1]+1,y);
            delay(16);
        }
    }
    getch();
}
```
OUTPUT(ScanLine Algorithm):


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC