INDEX

S.No.	Experiment	Date	Sign
1	To create a custom header file to be used in future programs		
2	To draw a line using DDA Algorithm		
3	To draw a line using Midpoint Line Drawing Algorithm		
4	To draw a line using Bresenhams Line Drawing Algorithm		
5	To draw name using lines		
6	To draw a circle using Bresenhams Circle Drawing Algorithm		
7	To draw an ellipse using Bresenhams Ellipse drawing algorithm		
8	To animate a spoked circle revolving around a circle		
9	To animate an ellipse revolving around a circle		
10	To clip a line using cohen sutherland algorithm for a rectangular window		
11	To clip a line for a non rectangular window		
12	To use sutherland hodgman polygon clipping algorithm		
13	To use scan line polygon filling algorithm		
14	To use Flood fill algorithm to fill a polygon		
15	To animate 2D transformation (scaling , translation and rotation)		
16	To display different projections of a 3D figure		

1. Custom Header File

dcgraphics.h:

```
#ifndef __DCGRAPHICS_H_INCLUDED__
#define __DCGRAPHICS_H_INCLUDED__
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include<DOS.h>
void drawCircle(int xc, int yc , int R , int ccode=15){
  int x=0,y=R;
  int d = 2 - 2*R;
  putpixel(320+xc+x,240-yc-y,ccode);
        putpixel(320+xc+x,240-yc+y,ccode);
        putpixel(320+xc-x,240-yc-y,ccode);
        putpixel(320+xc-x,240-yc+y,ccode);
        putpixel(320+xc+y,240-yc-x,ccode);
        putpixel(320+xc+y,240-yc+x,ccode);
        putpixel(320+xc-y,240-yc-x,ccode);
        putpixel(320+xc-y,240-yc+x,ccode);
  while(x<=y){
        if(d <0){
                d+= 2*x +3;
                x++;
                }
        else if(d==0){
                d+=2*x - 2*y + 6 ;
                y--;
                x++;
                }
        else{
                d+=-2*y + 3;
                y--;
        }
        //delay(10); used for animating line drawing (in name drawing program)
        putpixel(320+xc+x,240-yc-y,ccode);
        putpixel(320+xc+x,240-yc+y,ccode);
        putpixel(320+xc-x,240-yc-y,ccode);
        putpixel(320+xc-x,240-yc+y,ccode);
        putpixel(320+xc+y,240-yc-x,ccode);
        putpixel(320+xc+y,240-yc+x,ccode);
        putpixel(320+xc-y,240-yc-x,ccode);
        putpixel(320+xc-y,240-yc+x,ccode);
  }
}
void ltrd(int x1 , int x2 , int y1 , int y2,int c){
    int dx=x2-x1,dy=y1-y2,d,x=x1,y=y1;
  putpixel(320+x,240-y,c);
  if(dy<=dx){ d=2*dy - dx ;
    d=2*dy - dx ;
         while(x<=x2){
           if(d<=0){
           d+=2*dy;          }
           else     {
           d+=2*(dy-dx);
           y--;   }
           x++;
           putpixel(320+x,240-y,c);
  }
   }
```

```
   else{ d=2*dx-dy;
         while(y>=y2){
           if(d<=0){
                 d+=2*dx;              }
           else      {
                 d+=2*(dx-dy);
                 x++;      }
           y--;
           putpixel(320+x,240-y,c);
           }
    }
}
void ltru(int x1 , int x2 , int y1 , int y2,int c){
     int dx=x2-x1;
   int dy=y2-y1;
   int d=2*dy - dx ;
   int x=x1,y=y1;
   putpixel(320+x,240-y,c);
    if(dy<=dx){ d=2*dy - dx ;
         while(x<=x2){
         if(d<=0){
         d+=2*dy;
         }
         else      {
         d+=2*(dy-dx);
         y++;}
         x++;
         putpixel(320+x,240-y,c);
         }
    }
   else{ d=2*dx-dy;
         while(y<=y2){
           if(d<=0){
           d+=2*dx;
           }
           else      {
           d+=2*(dx-dy);
           x++;}
           y++;
           putpixel(320+x,240-y,c);
           }
    }
}

void rtld(int x1 , int x2 , int y1 , int y2,int c){
     int dx=x1-x2;
   int dy=y1-y2;
   int d=2*dy- dx ;
   int x=x1,y=y1;
   putpixel(320+x,240-y,c);
   if(dy<=dx){ d=2*dy - dx ;
         while(x>=x2){
           if(d<=0){
           d+=2*dy;
           }
           else
           {
           d+=2*(dy-dx);
           y--;}
           x--;
           putpixel(320+x,240-y,c);
   }
```

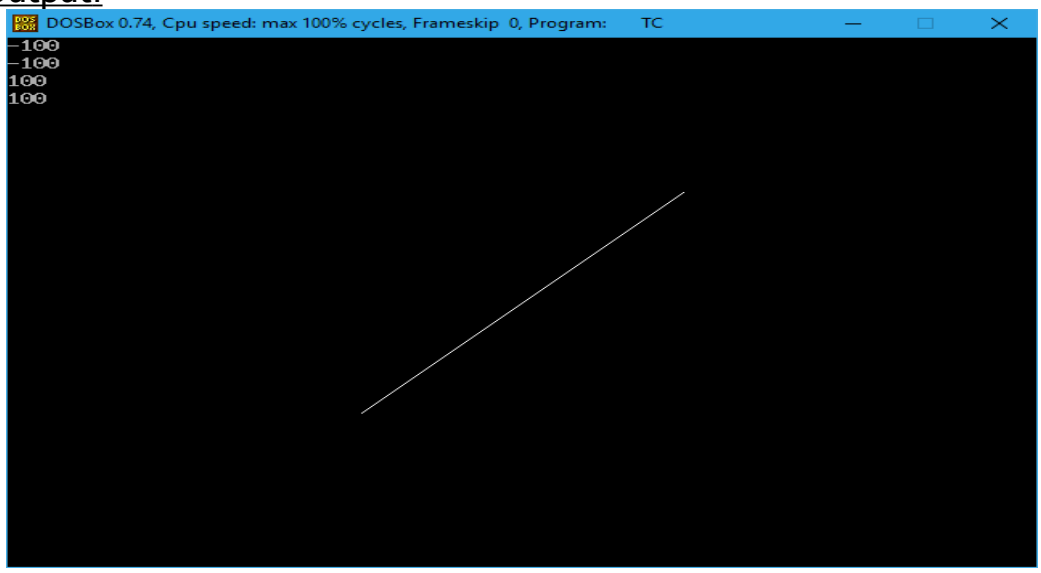
```
    }
   else{ d=2*dx-dy;
          while(y>=y2){
            if(d<=0){
            d+=2*dx;
            }
            else      {
            d+=2*(dx-dy);
            x--;}
            y--;
            putpixel(320+x,240-y,c);
            }
    }
}
void rtlu(int x1 , int x2 , int y1 , int y2,int c){
     int dx=x1-x2;
   int dy=y2-y1;
   int d=2*dy- dx ;
   int x=x1,y=y1;
   putpixel(320+x,240-y,c);
   if(dy<=dx){ d=2*dy - dx ;
          while(x>=x2){
            if(d<=0){
            d+=2*dy;
            }
            else
            {
            d+=2*(dy-dx);
            y++;}
            x--;
            putpixel(320+x,240-y,c);
   }
    }
   else{ d=2*dx-dy;
          while(y<=y2){
            if(d<=0){
            d+=2*dx;
            }
            else
            {
            d+=2*(dx-dy);
            x--;}
            y++;
            putpixel(320+x,240-y,c);
            }
    }
}
void drawline(int x1,int x2,int y1,int y2,int c=15){
    if(x1>x2){if(y1>y2){
            rtld(x1,x2,y1,y2,c);
         }
         else{
            rtlu(x1,x2,y1,y2,c);
         }
    }
    else{if(y1>y2){
            ltrd(x1,x2,y1,y2,c);
         }
         else{
            ltru(x1,x2,y1,y2,c);     }    }
}
#endif
```

2. DDA Line Drawing

dda.cpp:

```
#include<graphics.h>
#include<stdio.h>
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
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(320+X,240-Y,15);
                X += Xinc;
                Y += Yinc;
        }
}
 int main(void){
   int gdriver = DETECT, gmode, errorcode;
   initgraph(&gdriver, &gmode, "C:\\TC\\BGI");
   errorcode = graphresult();
   if (errorcode != grOk)   {
     printf("Graphics error: %s\n", grapherrormsg(errorcode));
     printf("Press any key to halt:");
     getch();
     exit(1);
   }
   setcolor(getmaxcolor());
        int x1=100,y1=200,x2=200,y2=300;
        cin>>x1>>y1>>x2>>y2;
        DDA(x1,y1,x2,y2);
   getch();
   closegraph();
   return 0;
}
```

Output:

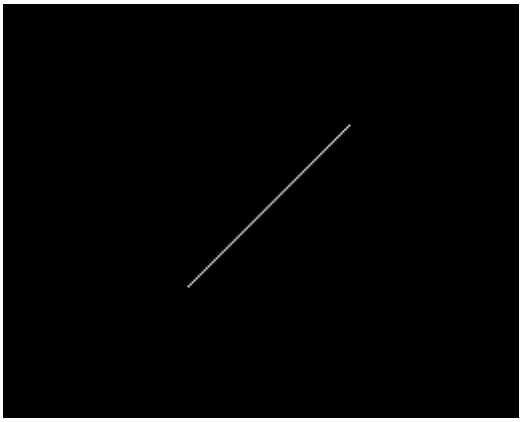


3. Mid Point Line Drawing Algorithm

mp.cpp:

```
#include<graphics.h>
#include<stdio.h>
#include<conio.h>
void midPoint(int X1, int Y1, int X2, int Y2) {
 int dx = X2 - X1;
 int dy = Y2 - Y1;
 int d = dy - (dx/2);
 int x = X1, y = Y1;
 putpixel(320+x,240-y, WHITE);
 while (x < X2)
 {
 x++;
 if (d < 0)
 d = d + dy;
 else{
 d += (dy - dx);
 y++;
 }
 putpixel(320+x,240-y,WHITE);
 }
}
int main(){
 int gd = DETECT, gm;
 initgraph (&gd, &gm, "C:\\TC\\BGI");
 int X1 = 20, Y1 = 20, X2 = 100, Y2 = 100;
 midPoint(X1, Y1, X2, Y2);
 getch();
 closegraph();
 return 0;
}
```

Output:



4. Bresenhams Line Drawing

dda.cpp:

```
 #include<graphics.h>
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#include<stdio.h>
void drawline(float xa,float ya,float xb,float yb){
   float x=xa,y=ya,dx=xb-xa,dy=yb-ya;
   if(abs(dy)<abs(dx)){       float d=2*dy-dx;
      putpixel(x,y,15);
      while(x<=xb){
         if(d<0){              d+=2*dy;          }
         else{
            d+=2*(dy-dx);
            y++;
         }
         x++;
         putpixel(x,y,15);
      }   }
   else{
      float d=2*dx-dy;
      putpixel(x,y,15);
      while(y<=yb){
         if(d<0){              d+=dx;          }
         else{
            d+=2*(dx-dy);
            x++;
         }
         y++;
         putpixel(x,y,15);
      }   }       }
int main(void){
  int gdriver = DETECT, gmode, errorcode;
  initgraph(&gdriver, &gmode, "C:\\TC\\BGI");
  errorcode = graphresult();
  if (errorcode != grOk)   {
     printf("Graphics error: %s\n", grapherrormsg(errorcode));
     printf("Press any key to halt:");
     getch();
     exit(1);
  }
  setcolor(getmaxcolor());
        int x1,y1,x2,y2
        cin>>x1>>y1>>x2>>y2;
        drawline(x1,y1,x2,y2);
  getch();
  closegraph();
  return 0;
}
```

Output:

5. Name Drawing Animation

final2.cpp:

```
 #include "dcgraphics.h"
int main(void)
{
  int gdriver = DETECT, gmode, errorcode;
  initgraph(&gdriver, &gmode, "C:\\TC\\BGI");
  errorcode = graphresult();
  if (errorcode != grOk)  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Press any key to halt:");
    getch();
    exit(1);
  }
  setcolor(getmaxcolor());
  //D
  drawline(-310,-310,-30,30);
  drawline(-315,-300,30,30);
  drawline(-300,-280,30,15);
  drawline(-280,-280,15,-15);
  drawline(-280,-300,-15,-30);
  drawline(-300,-315,-30,-30);
  //E
  drawline(-240,-270,29,29);
  drawline(-270,-270,30,0);
  drawline(-270,-240,0,0);
  drawline(-240,-270,-1,-1);
  drawline(-270,-270,-1,-30);
  drawline(-270,-240,-30,-30);
  //E
  drawline(-200,-230,29,29);
  drawline(-230,-230,30,0);
  drawline(-230,-200,0,0);
  drawline(-200,-230,-1,-1);
  drawline(-230,-230,-1,-30);
  drawline(-230,-200,-30,-30);
  //P
  drawline(-190,-190,-30,30);
  drawline(-195,-180,30,30);
  drawline(-180,-160,30,20);
  drawline(-160,-160,20,10);
  drawline(-160,-180,10,0);
  drawline(-180,-190,0,0);
  //E
  drawline(-120,-150,29,29);
  drawline(-150,-150,30,0);
  drawline(-150,-120,0,0);
  drawline(-120,-150,-1,-1);
  drawline(-150,-150,-1,-30);
  drawline(-150,-120,-30,-30);
  //S
  drawline(-80,-100,30,30);
  drawline(-100,-110,30,15);
  drawline(-110,-100,15,0);
  drawline(-100,-90,0,0);
  drawline(-90,-80,0,-15);
  drawline(-80,-90,-15,-30);
  drawline(-90,-110,-30,-30);
  //H
  drawline(-70,-70,30,-30);
```

```
   drawline(-70,-40,0,0);
   drawline(-40,-40,30,-30);
   getch();
   closegraph();
   return 0;
}
```

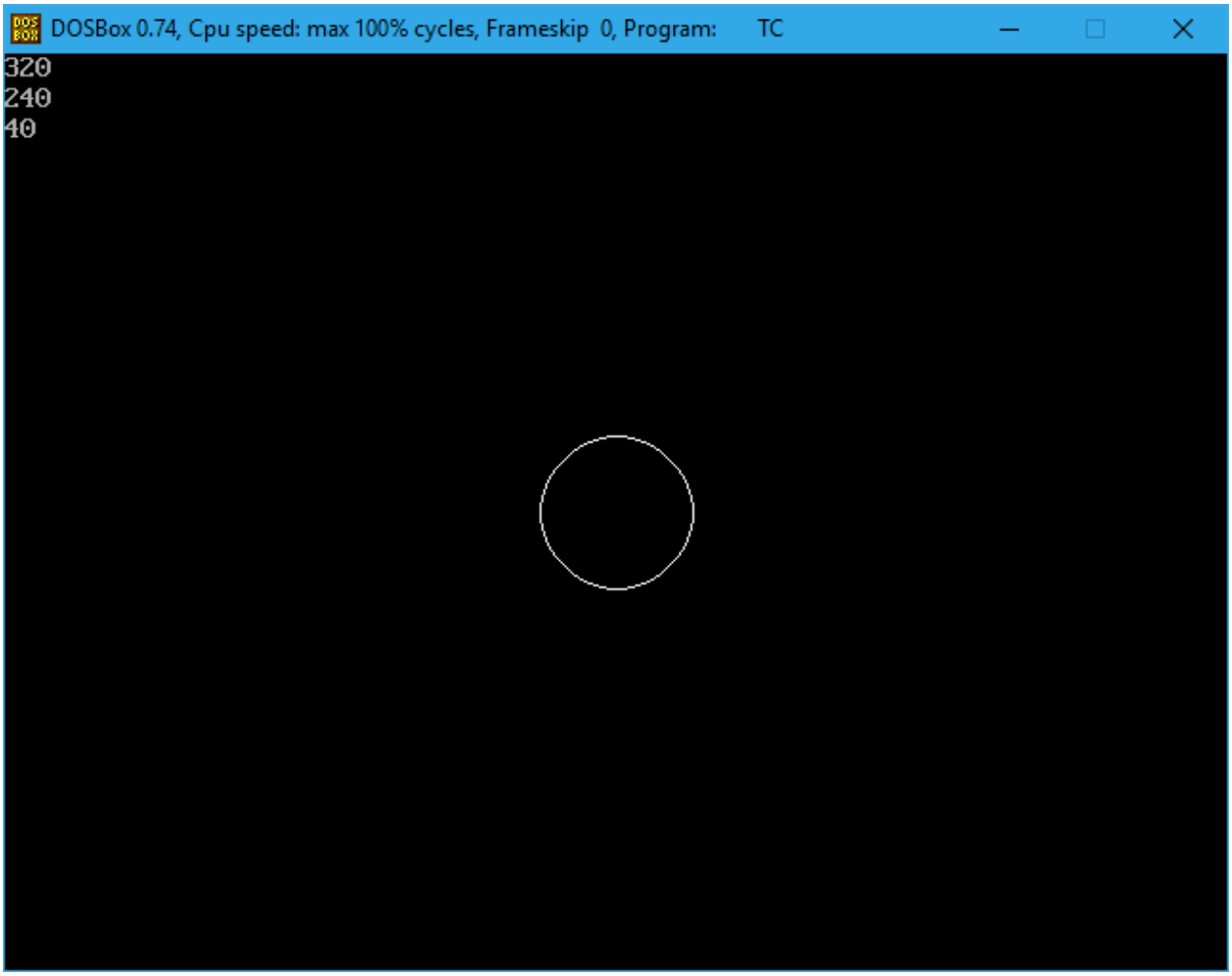
Output:

6. Bresenhams Circle Drawing

bcirc.cpp:

```
 #include<graphics.h>
#include<iostream.h>
#include<stdlib.h>
#include<dos.h>
#include<stdio.h>
#include<conio.h>
void drawCircle(int xc, int yc, int x, int y){
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);
}
void bresenham(int xc, int yc, int r){
int x = 0, y = r;
int d = 3 - 2 * r;
drawCircle(xc, yc, x, y);
while (y >= x){
x++;
if (d > 0){
y--;
d = d + 4 * (x - y) + 10;
}
else
  d = d + 4 * x + 6;
drawCircle(xc, yc, x, y);
}}
 int main(void){
  int gdriver = DETECT, gmode, errorcode;
  initgraph(&gdriver, &gmode, "C:\\TC\\BGI");
  errorcode = graphresult();
  if (errorcode != grOk)  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Press any key to halt:");
    getch();
    exit(1);
  }
  setcolor(getmaxcolor());
  int xc,yc,r;
  cin>>xc>>yc>>r;
  bresenham(xc,yc,r);
     getch();
  closegraph();
  return 0;
}
```

Output:


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
320
240
40

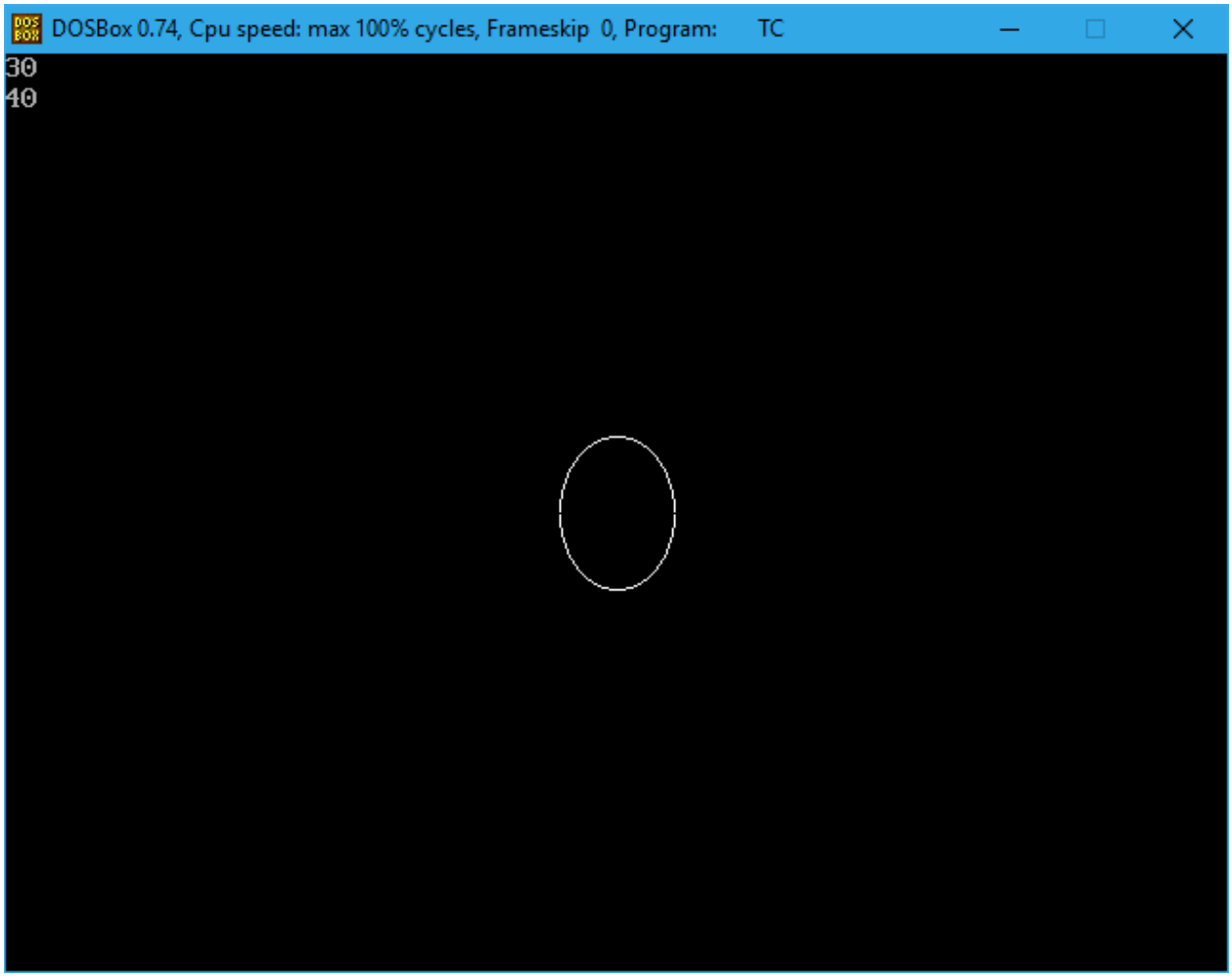
7. Bresenhams Ellipse Drawing

bellipse.cpp:

```
  #include<graphics.h>
#include<iostream.h>
#include<stdlib.h>
#include<dos.h>
#include<stdio.h>
#include<conio.h>
int main(void){
  int gdriver = DETECT, gmode, errorcode;
  initgraph(&gdriver, &gmode, "C:\\TC\\BGI");
  errorcode = graphresult();
  if (errorcode != grOk)  {
     printf("Graphics error: %s\n", grapherrormsg(errorcode));
     printf("Press any key to halt:");
     getch();
     exit(1);
  }
  setcolor(getmaxcolor());
 long int d1,d2;
int i,x=0,y;
long int rx,ry,rxsq,rysq,tworxsq,tworysq,dx,dy;
cin>>rx>>ry;
rxsq=rx*rx;
rysq=ry*ry;
tworxsq=2*rxsq;
tworysq=2*rysq;
y=ry;
d1=rysq - (rxsq * ry) + (0.25 * rxsq);
dx= tworysq * x;
dy= tworxsq * y;
do{
putpixel(320+x,240+y,15);
putpixel(320-x,240-y,15);
putpixel(320+x,240-y,15);
putpixel(320-x,240+y,15);
if (d1 < 0){
x=x+1;
y=y;
dx=dx + tworysq;
d1=d1 + dx + rysq;
}
else {
x=x+1;
y=y-1;
dx= dx + tworysq;
dy= dy - tworxsq;
d1= d1 + dx - dy + rysq;
}
delay(50);
}while (dx < dy);
d2 = rysq * ( x + 0.5) * ( x + 0.5 ) + rxsq * (y - 1) * (y-1) - rxsq * rysq;
do{
putpixel(320+x,240+y,15);
putpixel(320-x,240-y,15);
putpixel(320+x,240-y,15);
putpixel(320-x,240+y,15);
if (d2 >0){
x=x;
y=y-1;
```

```
dy = dy - tworxsq;
d2 = d2 - dy + rxsq;
}
else{
   x+=1;
   y-=1;
   dy-=tworxsq;
   dx+=tworysq;
   d2+=dx-dy+rxsq;
}
}while(y>0);
    getch();
   closegraph();
   return 0;
}
```

Output:



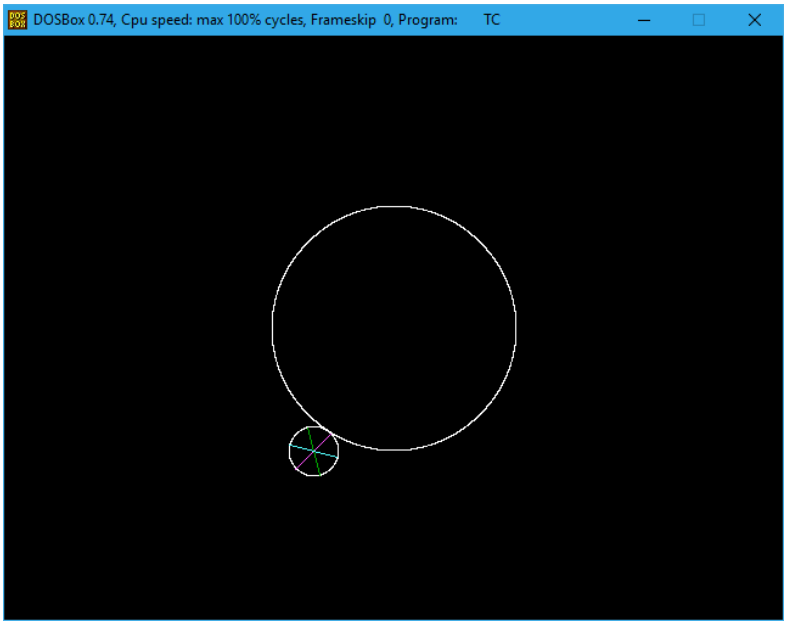
8. Circle With Spokes Animation

project2.cpp:

```
//A and D to move S and D to change speed X to exit

 #include<graphics.h>
 #include<stdio.h>
 #include<conio.h>
 #include<stdlib.h>
 #include<DOS.h>
 #include<math.h>
 #include"dcgraphics.h"

int round(float n){
    return floor((n+0.5));
}

 int main(void)
{
   int gdriver = DETECT, gmode, errorcode;
   initgraph(&gdriver, &gmode, "C:\\TC\\BGI");
   errorcode = graphresult();
   if (errorcode != grOk)
   {
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      printf("Press any key to halt:");
      getch();
      exit(1);
   }
   setcolor(getmaxcolor());
    int theta = 0;
    int x=0,y=120,x1,y1,x2,y2,x3,y3,R=120,i=1,o=3,b=0;
    char c;
    drawCircle(0,0,100);
    while(1>0){

          x = round((float)R * cos(theta* 3.14 / 180));
          y = round((float)R * sin(theta* 3.14 / 180));
          x1 = round((float)20 * cos(theta* 3.14 / 36));
          y1 = round((float)20 * sin(theta* 3.14 / 36));
          x2 = round((float)20 * cos((theta+60)* 3.14 / 36));
          y2 = round((float)20 * sin((theta+60)* 3.14 / 36));
          x3 = round((float)20 * cos((theta+120)* 3.14 / 36));
          y3 = round((float)20 * sin((theta+120)* 3.14 / 36));
          drawCircle(x,y,20,15);
          drawline(x+x1,x-x1,y+y1,y-y1,2);
          drawline(x+x2,x-x2,y+y2,y-y2,13);
          drawline(x+x3,x-x3,y+y3,y-y3,11);
          delay(15);
          c=getch();
          switch(c) {
                                case 'a' : i = 1*o;
                                           b=1;
                                           break;
                                case 'd': i=-1 * o;
                                           b=1;
                                           break;
                                case 'w' : o++;
                                           b=0;
                                                      break;
```

```
                        case 's' : o--;
                                   b=0;
                                   break;
                        case 'x' : return 0 ;
                        default:b=0;
        }

        if(b==1){
        drawline(x+x1,x-x1,y+y1,y-y1,0);
        drawline(x+x2,x-x2,y+y2,y-y2,0);
        drawline(x+x3,x-x3,y+y3,y-y3,0);
        drawCircle(x,y,20,0);
        drawCircle(0,0,100);

        theta+=i;
        }
   }

    drawline(x+x1,x-x1,y+y1,y-y1,2);
   drawline(x+x2,x-x2,y+y2,y-y2,13);
   drawline(x+x3,x-x3,y+y3,y-y3,11);
   drawCircle(x,y,20,15);
  getch();
  closegraph();
  return 0;
}
```

Output:



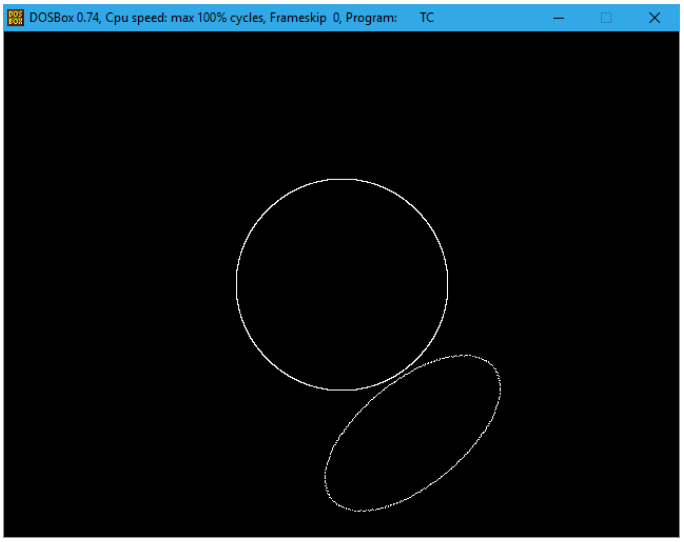
9. Ellipse Rotation And Revolution Animation

project4.cpp:

```
//ellipse rotating and revolving about circle
#include<graphics.h>
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<DOS.h>
#include<math.h>
#include"dcgraphics.h"
void drawellipse(int xc, int yc , int a , int b , int ccode=15,int theta = 0){
        int x, y,xdash,ydash;
        float d1, d2 = 0, dx, dy;
        x = 0;
        y = b;
        d1 = pow(b, 2) - (pow(a, 2) * b) + (0.25 * pow(a, 2));
        dx = 2 * pow(b, 2) * x;
        dy = 2 * pow(a, 2) * y;
        do      {
        xdash = (x)*cos(theta *3.14 / 90) - (y) *sin(theta *3.14/90);
        ydash = (x)*sin(theta *3.14 / 90) + (y) *cos(theta *3.14/90);
        putpixel(320+xc+ xdash, 240-yc- ydash, ccode);
        xdash = (x)*cos(theta *3.14 / 90) - (-y) *sin(theta *3.14/90);
        ydash = (x)*sin(theta *3.14 / 90) + (-y) *cos(theta *3.14/90);
        putpixel(320+xc+ xdash, 240-yc- ydash, ccode);
        xdash = (-x)*cos(theta *3.14 / 90) - (y) *sin(theta *3.14/90);
        ydash = (-x)*sin(theta *3.14 / 90) + (y) *cos(theta *3.14/90);
        putpixel(320+xc+ xdash, 240-yc- ydash, ccode);
        xdash = (-x)*cos(theta *3.14 / 90) - (-y) *sin(theta *3.14/90);
        ydash = (-x)*sin(theta *3.14 / 90) + (-y) *cos(theta *3.14/90);
        putpixel(320+xc+ xdash, 240-yc- ydash, ccode);
                if (d1 < 0)                     {
                        x++;
                        dx = dx + (2 * (pow(b, 2)));
                        d1 = d1 + dx + (pow(b, 2));
                }                       else                    {
                        x++;
                        y--;
                        dx = dx + (2 * (pow(b, 2)));
                        dy = dy - (2 * (pow(a, 2)));
                        d1 = d1 + dx - dy + (pow(b, 2));
                }               } while (dx < dy);
        do      {
        xdash = (x)*cos(theta *3.14 / 90) - (y) *sin(theta *3.14/90);
        ydash = (x)*sin(theta *3.14 / 90) + (y) *cos(theta *3.14/90);
        putpixel(320+xc+ xdash, 240-yc- ydash, ccode);
        xdash = (x)*cos(theta *3.14 / 90) - (-y) *sin(theta *3.14/90);
        ydash = (x)*sin(theta *3.14 / 90) + (-y) *cos(theta *3.14/90);
        putpixel(320+xc+ xdash, 240-yc- ydash, ccode);
        xdash = (-x)*cos(theta *3.14 / 90) - (y) *sin(theta *3.14/90);
        ydash = (-x)*sin(theta *3.14 / 90) + (y) *cos(theta *3.14/90);
        putpixel(320+xc+ xdash, 240-yc- ydash, ccode);
        xdash = (-x)*cos(theta *3.14 / 90) - (-y) *sin(theta *3.14/90);
        ydash = (-x)*sin(theta *3.14 / 90) + (-y) *cos(theta *3.14/90);
        putpixel(320+xc+ xdash, 240-yc- ydash, ccode);
                if (d2 > 0)                     {
                        x = x;
                        y--;
                        dy = dy - (2 * (pow(a, 2)));
```

```
                d2 = d2 - dy + pow(a, 2);
            }
            else            {
                x++;
                y--;
                dy = dy - (2 * (pow(a, 2)));
                dx = dx + (2 * (pow(b, 2)));
                d2 = d2 + dx - dy + pow(a, 2);
            }
        } while (y > 0);
}
int round(float n){
    return floor((n+0.5));
}
 int main(void){
   int gdriver = DETECT, gmode, errorcode;
   initgraph(&   gdriver, &gmode, "C:\\TC\\BGI");
   errorcode = graphresult();
   if (errorcode != grOk){
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      printf("Press any key to halt:");
      getch();
      exit(1);
   }
   setcolor(getmaxcolor());
    int theta = 0;
    int x=0,y=120,a=100,b=50,R=200;
    char c;
    drawCircle(0,0,100);
    while(theta<1080){
          x = round((float)100 * cos(theta* 3.14 / 180) + (float)a*cos(theta* 3.14 / 180));
          y = round((float)100 * sin(theta* 3.14 / 180) + (float)b*sin(theta* 3.14 / 180));
          drawellipse(x,y,a,b,15,theta);
          delay(100);
          drawellipse(x,y,a,b,0,theta);
          drawCircle(0,0,100);
          theta+=5;
    }
   getch();
   closegraph();
   return 0;
}
```

Output:



10. Cohen Sutherland Clipping(Rectangular Window)

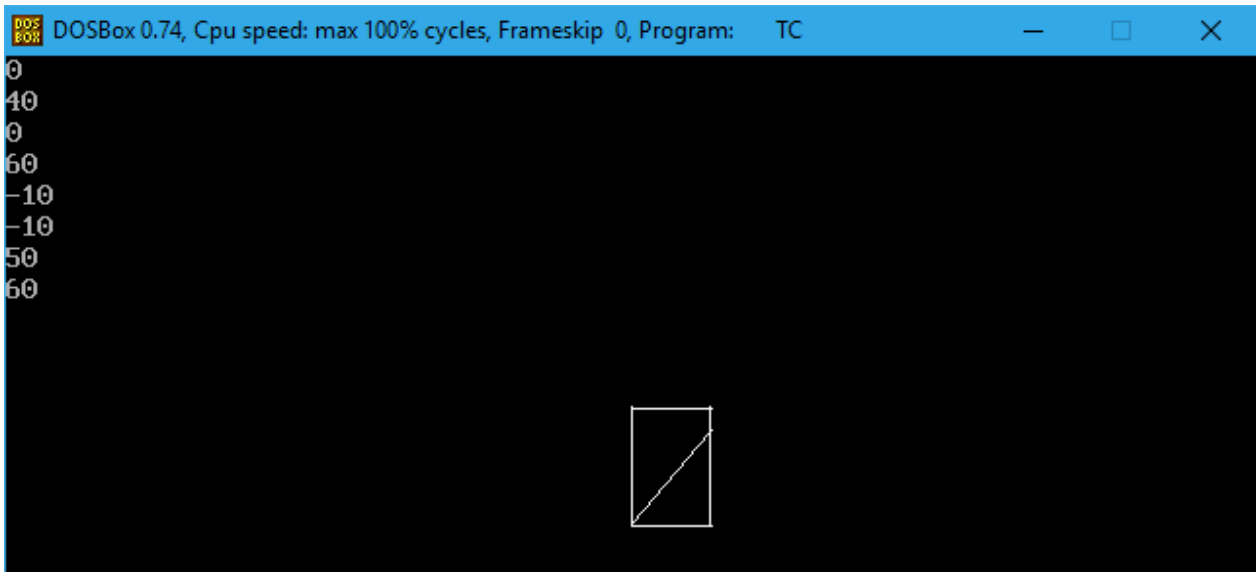
clip.cpp:

```
 #include<graphics.h>
#include<iostream.h>
#include<stdlib.h>
#include<dos.h>
#include<stdio.h>
#include<conio.h>
#include "dcgraphics.h"

 const int INSIDE = 0; // 0000
const int LEFT = 1;   // 0001
const int RIGHT = 2;  // 0010
const int BOTTOM = 4; // 0100
const int TOP = 8;    // 1000
  int codegen(double x, double y,int x_min,int x_max,int y_min, int y_max)
{
    int code = INSIDE;
    if (x < x_min)
        code |= LEFT;
    else if (x > x_max)
        code |= RIGHT;
    if (y < y_min)
        code |= BOTTOM;
    else if (y > y_max)
        code |= TOP;
    return code;
}
void cohenSutherlandClip(double x1, double y1,
                         double x2, double y2,int x_min , int x_max , int y_min , int y_max)
{
    int code1 = codegen(x1,y1,x_min,x_max,y_min,y_max);
    int code2 = codegen(x2,y2,x_min,x_max,y_min,y_max);
    int accept = 0;

    while (1)
    {
        if ((code1 == 0) && (code2 == 0))
        {
            accept = 1;
            break;
        }
        else if (code1 & code2)
        {
            break;
        }
        else
        {
            int code_out;
            double x, y;
            if (code1 != 0)
                code_out = code1;
            else
                code_out = code2;
            if (code_out & TOP)
            {
                x = x1 + (x2 - x1) * (y_max - y1) / (y2 - y1);
                y = y_max;
            }
```

```
            else if (code_out & BOTTOM)
            {
                  x = x1 + (x2 - x1) * (y_min - y1) / (y2 - y1);
                  y = y_min;
            }
            else if (code_out & RIGHT)
            {
                  y = y1 + (y2 - y1) * (x_max - x1) / (x2 - x1);
                  x = x_max;
            }
            else if (code_out & LEFT)
            {
                  y = y1 + (y2 - y1) * (x_min - x1) / (x2 - x1);
                  x = x_min;
            }
            if (code_out == code1)
            {
                  x1 = x;
                  y1 = y;
                  code1 = codegen(x1,y1,x_min,x_max,y_min,y_max);
            }
            else
            {
                  x2 = x;
                  y2 = y;
                  code2 = codegen(x2,y2,x_min,x_max,y_min,y_max);
            }
          }
    }
     if (accept)
    {
          drawline(x_min,x_min,y_min,y_max);
          drawline(x_min,x_max,y_min,y_min);
          drawline(x_max,x_max,y_min,y_max);
          drawline(x_min,x_max,y_max,y_max);

          drawline(x1,x2,y1,y2,15);
    }
    else
          cout << "Line is out of bounds" << endl;
}


 int main(void)
{
   int gdriver = DETECT, gmode, errorcode;
   initgraph(&gdriver, &gmode, "C:\\TC\\BGI");
   errorcode = graphresult();
   if (errorcode != grOk)
   {
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      printf("Press any key to halt:");
      getch();
      exit(1);
   }
   setcolor(getmaxcolor());
    int x_min,y_min,x_max,y_max;
    double x1,y1,x2,y2;
    cin>>x_min>>x_max>>y_min>>y_max>>x1>>y1>>x2>>y2;
    cohenSutherlandClip( x1,  y1, x2,  y2, x_min ,  x_max ,  y_min ,  y_max);
```

```
    getch();
    closegraph();
    return 0;
}
```

Output:



11. Line Clipping(Non Rectangular Window)

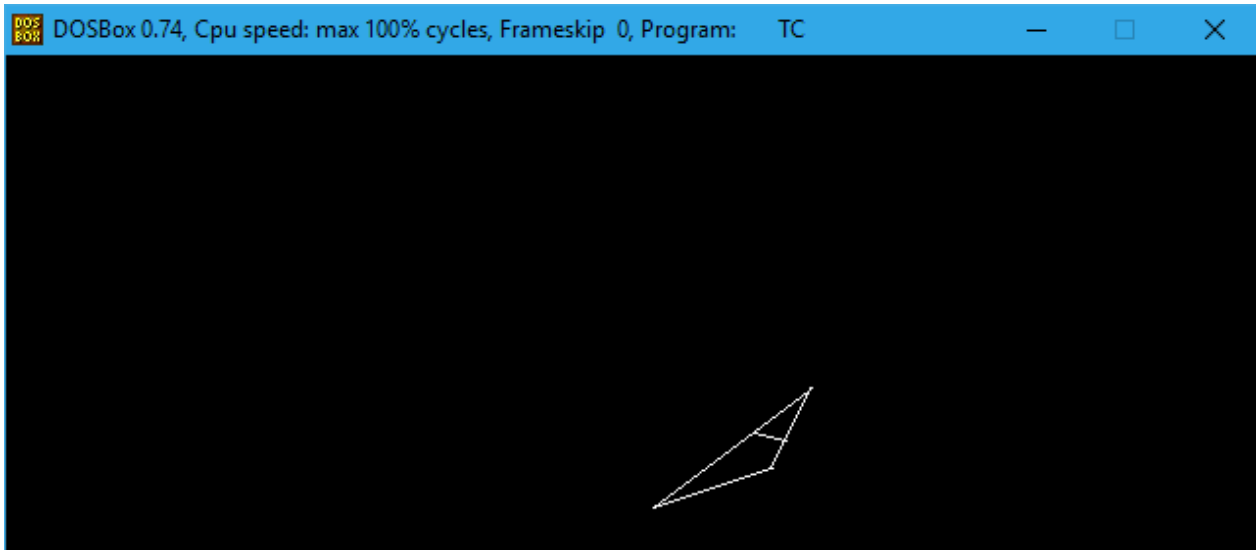
project5.cpp:

```
 //line clipping eg
#include<graphics.h>
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<DOS.h>
#include<math.h>
#include"dcgraphics.h"
#define AC(x,y) 4*y-3*x-10
#define AB(x,y) -3*y+x+20
#define BC(x,y) -y+2*x-110
#define UP 0x1
#define DOWN 0x2
#define RIGHT 0x4
int gencode(int x , int y){
        int code = 0x0;
        if(AC(x,y) >0){
                code |= UP;
        }
        if(AB(x,y) >0){
                code |= DOWN;
        }
        if(BC(x,y) >0){
                code |= RIGHT;
        }
        return code;
}
int round(float n){
   return floor((n+0.5));
}
void cliptest(int &x1, int &y1 , int &x2 , int &y2){
        int code1 = gencode(x1,y1) ;
        int code2 = gencode(x2,y2);
        int xdash , ydash;
        float m = (y2-y1)/(float)(x2-x1);
        float c = y1 - (float)m*x1;
        float m2 = 1/(float)m;
        float c2 = x1 - (float)m2*y1;
        char msg[128];
        int i =0 ;
        if((code1 & UP) == 1){
          if(abs((x2-x1)) > abs((y2-y1))){
                while(AC(x1,y1)  > 0){
                        x1=x1+abs((x2-x1))/(x2-x1);
                        y1 = (float)m*x1 + c;
                        }
                }
          else{
                        while(AC(x1,y1)  > 0){
                        y1=y1+abs(y2-y1)/(y2-y1);
                        x1 = (float)m2*y1+c2;
                        }
                }
        }
          else   if((code1 & DOWN) == 2){
          if(abs((x2-x1)) > abs((y2-y1))){
                while(AB(x1,y1)  > 0){
```

```
                        x1=x1+abs((x2-x1))/(x2-x1);
                y1 = (float)m*x1 + c;
                }
        }
          else{
                while(AB(x1,y1)  > 0){
                        y1=y1+abs(y2-y1)/(y2-y1);
                x1 = (float)m2*y1+c2;
                }
        }
       }
         else if((code1 & RIGHT) == 4){
                if(abs((x2-x1)) > abs((y2-y1))){
          while(AB(x1,y1)  > 0){
                        x1=x1+abs((x2-x1))/(x2-x1);
                y1 = (float)m*x1 + c;
                }
        }
          else{
                while(AB(x1,y1)  > 0){
                        y1=y1+abs(y2-y1)/(y2-y1);
                x1 = (float)m2*y1+c2;
                }               }       }
       else if((code2 & UP) == 1){
          if(abs((x2-x1)) > abs((y2-y1))){
                while(AC(x2,y2)  > 0){
                x2=x2+abs(x1-x2)/(x1-x2);
                y2 = (float)m*x2+c;
        }         }
          else{
                while(AC(x2,y2)  > 0){
                y2=y2+abs(y1-y2)/(y1-y2);
                x2 = (float)m2*y2+c2;
        }       }       }
          else   if((code2 & DOWN) == 2){
          if(abs((x2-x1)) > abs((y2-y1))){
                while(AB(x2,y2)  > 0){
                x2=x2+abs(x1-x2)/(x1-x2);
                y2 = (float)m*x2+c;
        }         }
          else{
                while(AB(x2,y2)  > 0){
                y2=y2+abs(y1-y2)/(y1-y2);
                x2 = (float)m2*y2+c2;
        }       }       }
         else     if((code2 & RIGHT) == 4){
          if(abs((x2-x1)) > abs((y2-y1))){
                while(BC(x2,y2)  > 0){
                x2=x2+abs(x1-x2)/(x1-x2);
                y2 = (float)m*x2+c;
        }         }
          else{
                while(BC(x2,y2)  > 0){
                y2=y2+abs(y1-y2)/(y1-y2);
                x2 = (float)m2*y2+c2;
                }               }       }       }
int main(void){
  int gdriver = DETECT, gmode, errorcode;
  initgraph(&
  gdriver, &gmode, "C:\\TC\\BGI");
  errorcode = graphresult();
  if (errorcode != grOk)   {
```

```
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Press any key to halt:");
    getch();
    exit(1);
  }
  setcolor(getmaxcolor());
   int done = 0;
   int flag = 0;
   int i = 0;
   int x1=15,y1=60,x2=100,y2=40;
   drawline(10,70,10,30);
   drawline(10,90,10,70);
   drawline(70,90,30,70);
   int code1,code2;
   do{
        code1 = gencode(x1,y1);
        code2 = gencode(x2,y2);
        if((code1 | code2) == 0 ){
                done = 1;
                flag = 1;
        }
        else if((code1 & code2) !=0){
                done = 1;
                flag = 0;
                }
        else {
                cliptest(x1,y1,x2,y2);
        }
        }while(done!=1) ;
        if(flag == 1)
        {
        drawline(x1,x2,y1,y2);
        }

  getch();
  closegraph();
  return 0;
}
```

Output:



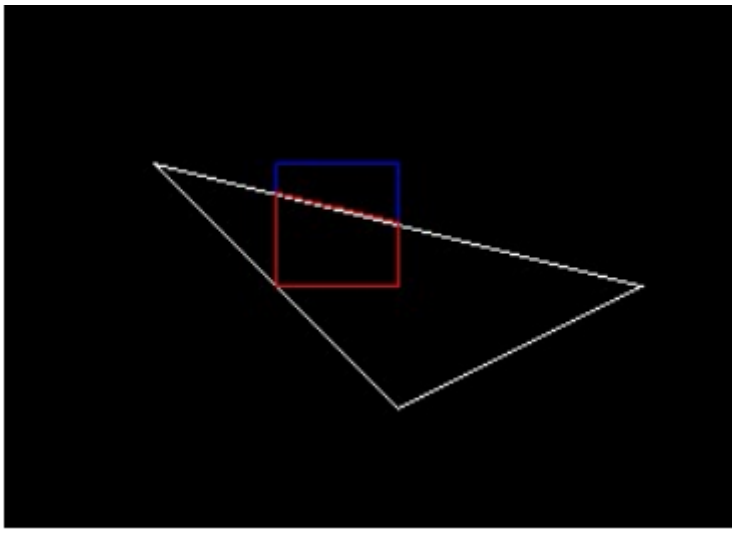
12. Polygon Clipping Using Cohen Sutherland

poly.cpp:

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

const int MAX_POINTS = 20;
int x_intersect(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4) {
int num = (x1*y2 - y1 * x2) * (x3 - x4) -  (x1 - x2) * (x3*y4 - y3 * x4);
int den = (x1 - x2) * (y3 - y4) - (y1 - y2) * (x3 - x4);
return num / den;
}
int y_intersect(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4) {
int num = (x1*y2 - y1 * x2) * (y3 - y4) -       (y1 - y2) * (x3*y4 - y3 * x4);
int den = (x1 - x2) * (y3 - y4) - (y1 - y2) * (x3 - x4);
return num / den;
}
void clip(int poly_points[][2], int &poly_size, int x1, int y1, int x2, int y2) {
int new_points[MAX_POINTS][2], new_poly_size = 0;
for (int i = 0; i < poly_size; i++) {
 int k = (i + 1) % poly_size;
 int ix = poly_points[i][0], iy = poly_points[i][1];
 int kx = poly_points[k][0], ky = poly_points[k][1];
 int i_pos = (x2 - x1) * (iy - y1) - (y2 - y1) * (ix - x1);
 int k_pos = (x2 - x1) * (ky - y1) - (y2 - y1) * (kx - x1);
 if (i_pos < 0 && k_pos < 0) {
 new_points[new_poly_size][0] = kx;
 new_points[new_poly_size][1] = ky;
 new_poly_size++;
 }
 else if (i_pos >= 0 && k_pos < 0) {
 new_points[new_poly_size][0] = x_intersect(x1, y1, x2, y2, ix, iy, kx, ky);
 new_points[new_poly_size][1] = y_intersect(x1, y1, x2, y2, ix, iy, kx, ky);
 new_poly_size++;
 new_points[new_poly_size][0] = kx;
 new_points[new_poly_size][1] = ky;
 new_poly_size++;
 }
 else if (i_pos < 0 && k_pos >= 0) {
 new_points[new_poly_size][0] = x_intersect(x1, y1, x2, y2, ix, iy, kx, ky);
 new_points[new_poly_size][1] = y_intersect(x1, y1, x2, y2, ix, iy, kx, ky);
 new_poly_size++;
 }
 }
poly_size = new_poly_size;
for ( i = 0; i < poly_size; i++) {
 poly_points[i][0] = new_points[i][0];
 poly_points[i][1] = new_points[i][1];
 }
}
void suthHodgClip(int poly_points[][2], int poly_size, int clipper_points[][2], int clipper_size) {
for (int i = 0; i<clipper_size; i++) {
 int k = (i + 1) % clipper_size;
 clip(poly_points, poly_size, clipper_points[i][0], clipper_points[i][1], clipper_points[k][0],  clipper_points[k]
[1]);
 }
 setcolor(RED);
for ( i = 1; i < poly_size; i++)
```

```
 line(poly_points[i][0], poly_points[i][1], poly_points[i - 1][0],poly_points[i - 1][1]);
 line(poly_points[0][0], poly_points[0][1], poly_points[poly_size - 1][0],poly_points[poly_size - 1][1]);
}
int main() {
int poly_size = 3;
int poly_points[20][2] = { { 100,150 },{ 200,250 },{ 300,200 } };
int clipper_size = 4;
int clipper_points[][2] = { { 150,150 },{ 150,200 },{ 200,200 },{ 200,150 } };
int gdriver = DETECT, gmode, errorcode;
   initgraph(&gdriver, &gmode, "C:\\TC\\BGI");

 setcolor(WHITE);
for (int i = 1; i < poly_size; i++)
 line(poly_points[i][0], poly_points[i][1], poly_points[i - 1][0],
poly_points[i - 1][1]);
 line(poly_points[0][0], poly_points[0][1], poly_points[poly_size - 1][0],
poly_points[poly_size - 1][1]);
 setcolor(BLUE);
for (i = 1; i < clipper_size; i++)
 line(clipper_points[i][0], clipper_points[i][1], clipper_points[i - 1][0],clipper_points[i - 1][1]);
 line(clipper_points[0][0], clipper_points[0][1], clipper_points[clipper_size-1][0], clipper_points[clipper_size
- 1][1]);
 suthHodgClip(poly_points, poly_size, clipper_points, clipper_size);
 getch();
 closegraph();
return 0;
}
```

Output:



13. Scanline Polygon Filling

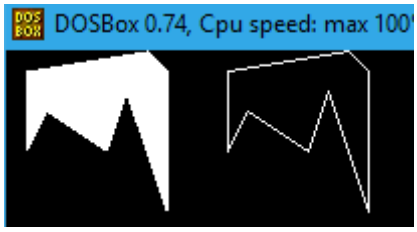
poly.cpp:

```
#include <stdio.h>
#include <math.h>
#include<graphics.h>
#define maxHt 100
#define maxWd 100
#define maxVer 10
typedef struct edgebucket {
int ymax;
float xofymin;
float slopeinverse;
}EdgeBucket;
typedef struct edgetabletup {
int countEdgeBucket;
EdgeBucket buckets[maxVer];
}EdgeTableTuple;
EdgeTableTuple EdgeTable[maxHt], ActiveEdgeTuple;
void initEdgeTable() {
int i;
for (i = 0; i<maxHt; i++) {
 EdgeTable[i].countEdgeBucket = 0;
 }
 ActiveEdgeTuple.countEdgeBucket = 0;
}
void insertionSort(EdgeTableTuple *ett) {
int i, j;
EdgeBucket temp;
for (i = 1; i < ett->countEdgeBucket; i++) {
 temp.ymax = ett->buckets[i].ymax;
 temp.xofymin = ett->buckets[i].xofymin;
 temp.slopeinverse = ett->buckets[i].slopeinverse;
 j = i - 1;
 while ((temp.xofymin < ett->buckets[j].xofymin) && (j >= 0)) {
 ett->buckets[j + 1].ymax = ett->buckets[j].ymax;
 ett->buckets[j + 1].xofymin = ett->buckets[j].xofymin;
 ett->buckets[j + 1].slopeinverse = ett->buckets[j].slopeinverse;
 j = j - 1;
 }
 ett->buckets[j + 1].ymax = temp.ymax;
 ett->buckets[j + 1].xofymin = temp.xofymin;
 ett->buckets[j + 1].slopeinverse = temp.slopeinverse;
 }
}
void storeEdgeInTuple(EdgeTableTuple *receiver, int ym, int xm, float slopInv) {
 (receiver->buckets[(receiver)->countEdgeBucket]).ymax = ym;
 (receiver->buckets[(receiver)->countEdgeBucket]).xofymin = (float)xm;
 (receiver->buckets[(receiver)->countEdgeBucket]).slopeinverse = slopInv;
 insertionSort(receiver);
 (receiver->countEdgeBucket)++;
}
void storeEdgeInTable(int x1, int y1, int x2, int y2) {
float m, minv;
int ymaxTS, xwithyminTS, scanline;
if (x2 == x1)
 minv = 0.000000;
else {
 m = ((float)(y2 - y1)) / ((float)(x2 - x1));
```

```
 if (y2 == y1)
 return;
 minv = (float)1.0 / m;
 }
if (y1>y2) {
 scanline = y2;
 ymaxTS = y1;
 xwithyminTS = x2;
 }
else {
 scanline = y1;
 ymaxTS = y2;
 xwithyminTS = x1;
 }
 storeEdgeInTuple(&EdgeTable[scanline], ymaxTS, xwithyminTS, minv);
}
void removeEdgeByYmax(EdgeTableTuple *Tup, int yy) {
int i, j;
for (i = 0; i< Tup->countEdgeBucket; i++) {
 if (Tup->buckets[i].ymax == yy) {
 for (j = i; j < Tup->countEdgeBucket - 1; j++) {
 Tup->buckets[j].ymax = Tup->buckets[j + 1].ymax;
Tup->buckets[j].xofymin = Tup->buckets[j + 1].xofymin;
Tup->buckets[j].slopeinverse = Tup->buckets[j + 1].slopeinverse;
 }
 Tup->countEdgeBucket--;
 i--;
 } }}
void updatexbyslopeinv(EdgeTableTuple *Tup) {
int i;
for (i = 0; i<Tup->countEdgeBucket; i++)
 (Tup->buckets[i]).xofymin = (Tup->buckets[i]).xofymin +(Tup->buckets[i]).slopeinverse;
}
void ScanlineFill() {
int i, j, x1, ymax1, x2, ymax2, FillFlag = 0, coordCount;
for (i = 0; i<maxHt; i++) {
 for (j = 0; j<EdgeTable[i].countEdgeBucket; j++) {
 storeEdgeInTuple(&ActiveEdgeTuple, EdgeTable[i].buckets[j]. ymax, EdgeTable[i].buckets[j].xofymin,
 EdgeTable[i].buckets[j].slopeinverse);
 }
 removeEdgeByYmax(&ActiveEdgeTuple, i);
 insertionSort(&ActiveEdgeTuple);
 j = 0;
 FillFlag = 0;
 coordCount = 0;
 x1 = 0;
 x2 = 0;
 ymax1 = 0;
 ymax2 = 0;
 while (j<ActiveEdgeTuple.countEdgeBucket)
 {
 if (coordCount % 2 == 0)
 {
 x1 = (int)(ActiveEdgeTuple.buckets[j].xofymin);
 ymax1 = ActiveEdgeTuple.buckets[j].ymax;
 if (x1 == x2) {
 if (((x1 == ymax1) && (x2 != ymax2)) || ((x1 != ymax1) &&(x2 == ymax2))) {
 x2 = x1;
 ymax2 = ymax1;
 }
 else {
```

```
 CoordCount++; } }
 else
 coordCount++; }
 else {
 x2 = (int)ActiveEdgeTuple.buckets[j].xofymin;
 ymax2 = ActiveEdgeTuple.buckets[j].ymax;
 FillFlag = 0;
 if (x1 == x2) {
 if (((x1 == ymax1) && (x2 != ymax2)) || ((x1 != ymax1) &&(x2 == ymax2))) {
 x1 = x2;
 ymax1 = ymax2;
 }
 else {
 coordCount++;
 FillFlag = 1;
 } }
 else {
 coordCount++;
 FillFlag = 1;
 }
 if (FillFlag)
 line(x1, i, x2, i);
 }
 j++;
 }
 updatexbyslopeinv(&ActiveEdgeTuple);
 }}
void drawPoly() {
int points[8][2] = { { 10, 10 },{ 10, 50 },{ 20, 30 },{ 50, 50 },{ 60, 20 },{ 80, 80},{ 80, 10 },{ 70, 0 } };
for (int i = 1; i < 8; i++)
 storeEdgeInTable(points[i][0], points[i][1], points[i - 1][0], points[i -1][1]);
 storeEdgeInTable(points[0][0], points[0][0], points[7][0], points[7][1]);
for ( i = 1; i < 8; i++)
 line(100 + points[i][0], points[i][1], 100 + points[i - 1][0], points[i - 1][1]);
 line(100 + points[0][0], points[0][0], 100 + points[7][0], points[7][1]);
}
void drawpoly(void) {
 initEdgeTable();
 drawPoly();
 ScanlineFill();}
int main(int argc, char** argv)
{int gdriver = DETECT, gm;
   initgraph(&gdriver, &gm, "C:\\TC\\BGI");
 drawpoly();
 getchar();
 closegraph();
}
```

Output:

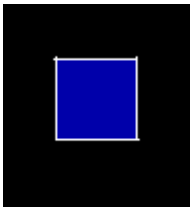


14. Flood Fill

fill.cpp:

```
#include <graphics.h>
#include <stdio.h>
#include<conio.h>
#include "dcgraphics.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 = DETECT, gm;
 initgraph(&gd, &gm,"C:\\TC\\BGI");
int x = 11;
int y = 11;
drawline(0,0,0,40);
drawline(40,40,0,40);
drawline(0,40,0,0);
drawline(40,0,40,40);
int newcolor = BLUE;
int oldcolor = 0;
 flood(320+x, 240-y, newcolor, oldcolor);
 getch();
return 0;
}
```

Output:



15. 2D Transformation

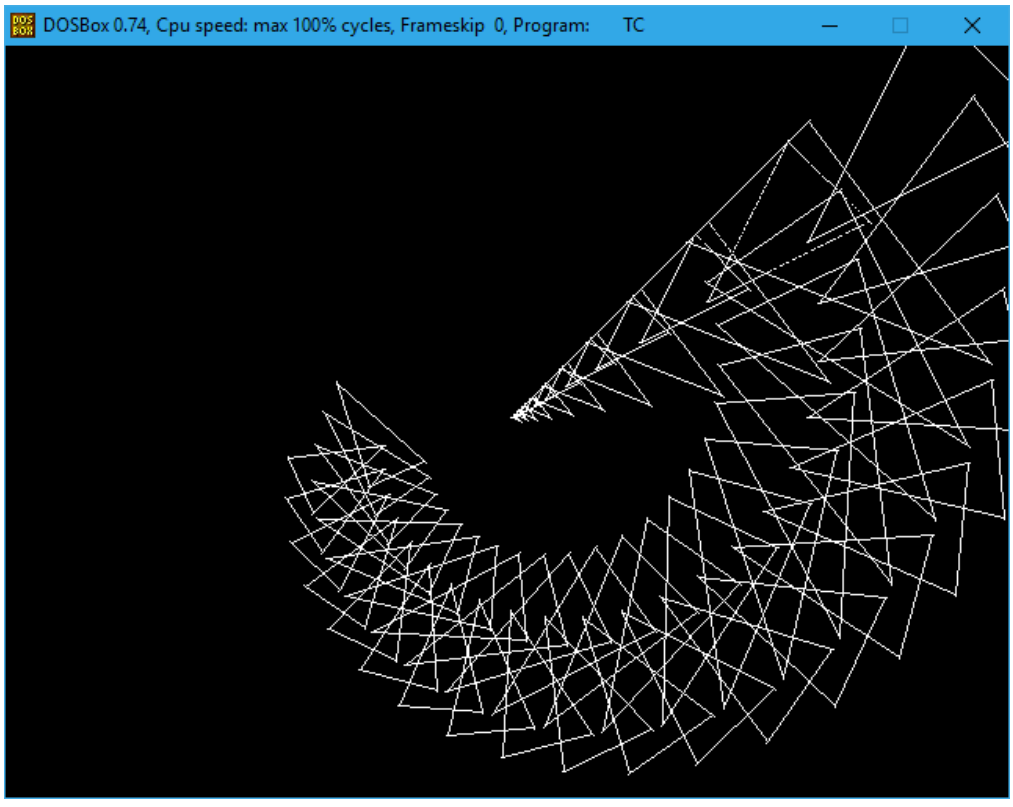
2d.cpp:

```
        //2D scaling and translation (rotation wasnt working) matrices use dont have extra col and row
implemented which should be there
 #include<graphics.h>
 #include<stdio.h>
 #include<conio.h>
 #include<stdlib.h>
 #include<DOS.h>
 #include<math.h>
 #include"dcgraphics.h"
 int round(float n){
    return floor((n+0.5));
}
void mm(float firstMatrix[][3], float secondMatrix[][3], float mult[][3])
{
        int i, j, k;

        // Initializing elements of matrix mult to 0.
        for(i = 0; i < 3; ++i)        {
                for(j = 0; j < 3; ++j)                  {
                        mult[i][j] = 0;
                }         }

        // Multiplying matrix firstMatrix and secondMatrix and storing in array mult.
        for(i = 0; i < 3; ++i)        {
                for(j = 0; j < 3; ++j)                  {
                        for(k=0; k<3; ++k)                              {
                                mult[i][j] += firstMatrix[i][k] * secondMatrix[k][j];
                        }                   }         }         }

void drawall(float firstMatrix[][3] , float secondMatrix[][3]  , int color = 15){
   for(int i  = 0 ; i < 3 ; i++){
      drawline(round(firstMatrix[i][0]) , round(firstMatrix[(i+1)%3][0]),round(firstMatrix[i]
[1]) ,round( firstMatrix[(i+1)%3][1]) ,color);
      drawline(round(secondMatrix[i][0]) , round(secondMatrix[(i+1)%3][0]), round(secondMatrix[i][1]) ,
round(secondMatrix[(i+1)%3][1]) , color);
   }
}
void copy(float firstMatrix[][3], float secondMatrix[][3]){
        int i, j, k;
        for(i = 0; i < 3; ++i)
                for(j = 0; j < 3; ++j)
                        firstMatrix[i][j] = secondMatrix[i][j];
}
void transform(float f[][3] , float s[][3] , float trans[][3],float temp[][3] ){
    mm(f,trans , temp);
   copy(f,temp);
   mm(s,trans , temp);
   copy(s,temp);
}
 int main(void)
{
   int gdriver = DETECT, gmode, errorcode;
   initgraph(&gdriver, &gmode, "C:\\TC\\BGI");
   errorcode = graphresult();
   if (errorcode != grOk)
   {
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
```

```
   printf("Press any key to halt:");
   getch();
   exit(1);
 }

 setcolor(getmaxcolor());
 float f[][3] = {
   {3,3,1},
   {5,5,1},
   {8,1,1}
 };
 float s[][3] = {
   {5,3,1},
   {7,7,1},
   {9,5,1}
 };
 float temp[][3]={
   {0,0,0},
   {0,0,0},
   {0,0,0}
 };
 float temp2[][3]={
   {0,0,0},
   {0,0,0},
   {0,0,0}
 };
 float trans[][3]={
   {0,0,0},
   {0,0,0},
   {0,0,0}
 };
 int setflag = 0;

 drawall(f , s);
 for(int i=0;i<10;i++){
 delay(50);
 drawall(f,s,0);
  trans[0][0] = 1.5;
  trans[1][1] = 1.5;
  trans[2][2] = 1;
  transform(f,s,trans,temp);
 drawall(f,s);

 }
for(int j=0;j<20;j++){
 delay(50);
 drawall(f,s,0);
 mm(temp , temp2 , trans);
trans[0][0] = 1;
trans[0][2] =-30 ;
  trans[1][1] = 1;
trans[1][2] = -1;
trans[2][2] =1;
transform(f,s,trans,temp);
mm(temp , temp2 , trans);
  trans[0][0] = 0.98;
  trans[1][1] = 0.98;
  trans[2][2] = 1;
  transform(f,s,trans,temp);
 drawall(f,s);
 }
  for(
```

```
    j=0;j<20;j++){
  delay(50);
  drawall(f,s,0);
   mm(temp , temp2 , trans);
 trans[0][0] = cos( j * 10 *3.14 / 180);
 trans[0][1] =-sin( j * 10 *3.14 / 180) ;
  trans[2][2] = 1;
 trans[1][0] = sin( j * 10 *3.14 / 180);
 trans[1][1] =cos( j * 10 *3.14 / 180);
 transform(f,s,trans,temp);
              mm(temp , temp2 , trans);
 trans[0][0] = 1;
 trans[0][2] =+30 ;
  trans[1][1] = 1;
 trans[1][2] = -1  ;
 trans[2][2] =1;
 transform(f,s,trans,temp);
 mm(temp , temp2 , trans);
  trans[0][0] = 0.95;
  trans[1][1] = 0.95;
  trans[2][2] = 1;
  transform(f,s,trans,temp);
  drawall(f,s);
  }
  drawall(f,s);
  getch();
  closegraph();
  return 0;
}
```

Output(showing path of figure i.e without erasing):



16. Projection of 3D Figure

view.cpp:

```
 #include<graphics.h>
 #include<stdio.h>
 #include<conio.h>
 #include<stdlib.h>
 #include<DOS.h>
 #include<math.h>
 #include"dcgraphics.h"
int round(float n){
   return floor((n+0.5));
}

void drawfv(int face1[][3] , int face2[][3], int xos , int yos , int color = 15){
        for( int i = 0 ;  i < 4 ; i++){
        drawline(xos+face1[i][0] , xos + face2[i][0] , yos + face1[i][1] , yos + face2[i][1],color );
        drawline(xos+face1[i][0] , xos + face1[((i+1)%4)][0] , yos + face1[i][1] , yos + face1[((i+1)%4)]
[1],color );
        drawline(xos+face2[i][0] , xos + face2[((i+1)%4)][0] , yos + face2[i][1] , yos + face2[((i+1)%4)]
[1],color );
        }
}

void drawsv(int face1[][3] , int face2[][3], int xos , int yos,int color = 15){
        for( int i = 0 ;  i < 4 ; i++){
        drawline(xos+face1[i][1] , xos + face2[i][1] , yos + face1[i][2] , yos + face2[i][2],color );
        drawline(xos+face1[i][1] , xos + face1[((i+1)%4)][1] , yos + face1[i][2] , yos + face1[((i+1)%4)]
[2],color );
        drawline(xos+face2[i][1] , xos + face2[((i+1)%4)][1] , yos + face2[i][2] , yos + face2[((i+1)%4)]
[2] ,color);
        }
}

void drawtv(int face1[][3] , int face2[][3], int xos , int yos,int color = 15){
        for( int i = 0 ;  i < 4 ; i++){
        drawline(xos+face1[i][2] , xos + face2[i][2] , yos + face1[i][0] , yos + face2[i][0],color );
        drawline(xos+face1[i][2] , xos + face1[((i+1)%4)][2] , yos + face1[i][0] , yos + face1[((i+1)%4)]
[0] ,color);
        drawline(xos+face2[i][2] , xos + face2[((i+1)%4)][2] , yos + face2[i][0] , yos + face2[((i+1)%4)]
[0] ,color);
        }
}


void drawiso(int face1temp[][3] , int face2temp[][3], int xos , int yos,int color=15){
        int face1[][3] = {{0,0,0},{0,0,0} ,{0,0,0} , {0,0,0}};
        int face2[][3] = {{0,0,0},{0,0,0} ,{0,0,0} , {0,0,0}};
        float iso[][3] = {{0.7071,0,0.7071},{0.4082,0.8166,-0.4082} ,{0,0,0} , {0,0,0}};
        for(int j=0;j<4;j++){
          for(int k = 0 ; k<3 ; k++){
                face1[j][k] = round(float(face1temp[j][0] * iso[k][0] + face1temp[j][1] * iso[k][1] + face1temp[j]
[2] * iso[k][2]));
                face2[j][k] = round(float(face2temp[j][0] * iso[k][0] + face2temp[j][1] * iso[k][1] + face2temp[j]
[2] * iso[k][2]));
           }
        }
        for( int i = 0 ;  i < 4 ; i++){
        drawline(xos+face1[i][0] , xos + face2[i][0] , yos + face1[i][1] , yos + face2[i][1] ,color);
```

```
        drawline(xos+face1[i][0] , xos + face1[((i+1)%4)][0] , yos + face1[i][1] , yos + face1[((i+1)%4)]
[1] ,color);
        drawline(xos+face2[i][0] , xos + face2[((i+1)%4)][0] , yos + face2[i][1] , yos + face2[((i+1)%4)]
[1] ,color);
        }
}


 int main(void)
{
  int gdriver = DETECT, gmode, errorcode;
  initgraph(&
  gdriver, &gmode, "C:\\TC\\BGI");
  errorcode = graphresult();
  /* an error occurred */
  if (errorcode != grOk)
  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Press any key to halt:");
    getch();
    exit(1);
  }

  setcolor(getmaxcolor());
  drawline(-107,-107,-240,240);
  drawline(106,106,-240,240);
  drawline(-320,320,0,0);

  int tlxos = -213  , tlyos = 120 ;
  int txos = tlxos + 213 , tyos = tlyos  , trxos = tlxos + 426 , tryos = tlyos;
  int bxos = txos , byos = tyos-240 ;
  int face1[][3] = { {0,0,0} , {0,80,0} , {80,80,0} , {80,0,0}};
  int face2[][3] = { {20,20,60} , {20,60,60} , {60,60,60} , {60,20,60}};
  drawsv(face1 , face2 , tlxos , tlyos);
  drawfv(face1 , face2 , txos , tyos);
  drawtv(face1 , face2 , trxos , tryos);
  drawiso(face1,face2,bxos,byos);
  getch();
  closegraph();
  return 0;
}
```

Output: