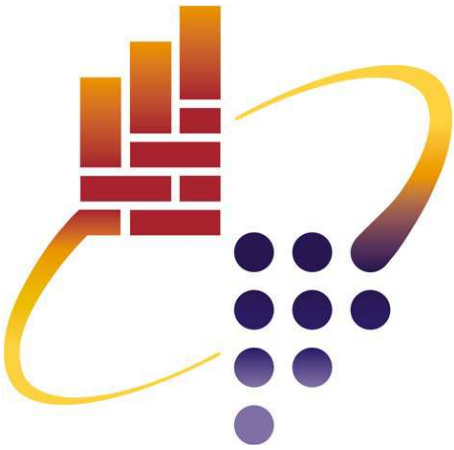
PRACTICAL IN COMPUTER GRAPHICS AND MULTIMEDIA



Submitted By

Vaibhav Jain

Student

Bachelor of Computer Applications

Swati Jain Institute Of Management Studies.

2001-2004.

Head Office

182, Jaora Compound,

Indore.

Swati Jain Academy (SJA)

Institute Campus

33, Sampat Farms,

Bicholi Mardana,

Indore

Swati Jain Institute of Management Studies (SJIMS)

Acknowledgments

No man is born complete and I am no exception. When the times were tensed and it seemed like I should kick the whole bunch of meaningless symbols and code into the recycle bin, pour some water on my keyboard and throw the book away for good, all that could sustain me was the support of teachers, friends and elders. I was lucky enough to be surrounded by such a people who were helpful and supportive. Without their help this Project File would have probably completed on released on my 75th birthday.

In am greatly thankful to Mr. Vishal Khasgiwal, our Teacher with out whose dedicated guidance and support this project would have being non existent.

I am also exceedingly thankful to the member faculty at Swati Jain Institute Of Management Studies as well as Swati Jain Madam who were there when their support and that wonderful sense of humor was badly and eagerly needed. Thanks, I can never forget you all.

And, finally a word of gratitude to my Parents and brother Ronak, who were always there with their support and encouragement, even though I'm a little crazy at times.

Vaibhav Jain

vaibhav@genesisconvent.com
57, Shiv Shahkti Nagar,
Kanadia Road,
Indore.
May 2004.

CERTIFICATE

This is to certify that **Vaibhav Jain** an enrollee of Bachelor of Computer Application and a student **Swati Jain Institute of Management Studies** has worked on the project **"Practical in Computer Graphics & Multimedia".** He has put sincere effort in the project and has performed tasks related to the project in the Computer Lab of **Swati Jain Institute of Management Studies**. This project may be considered as a partial fulfillment for the examinations conducted by **Devi Ahilya Vishva Vidyalaya, Indore**.

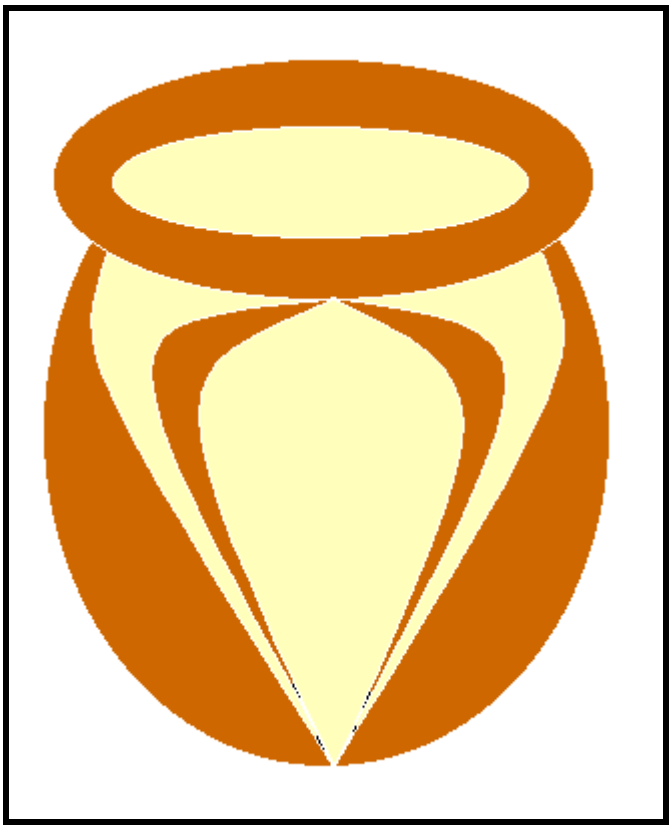
Date:

Mr. Vishal Khasgiwal

External

Index..

Index.



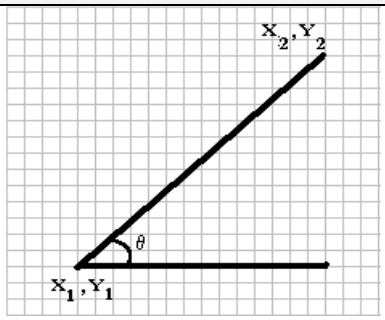
The Harry Potters Pot: Courtesy Ronak Jain

Computer Graphics

Cartesian Line Drawing Program

Objective

To plot a line starting and ending at given points using its Cartesian Plane Equation.



Theory

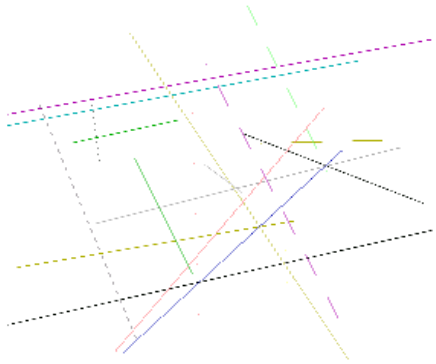
Let a line be starting from Point(x1,y1) & ending at Point(x2,y2) then its Cartesian-Equation is of form Y=MX+C
Where,

M=(y2-y1)/(x2-x1)
And, C= y1-M.x1

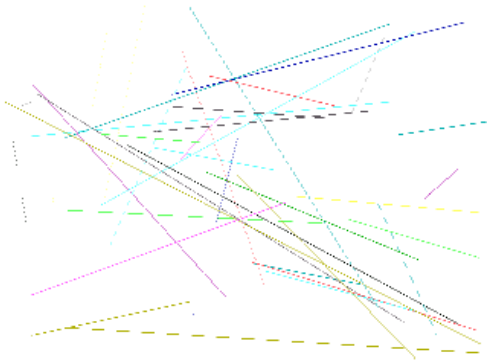
Code (linedraw.cpp)

```
//line plotting using its cartesian equation
//draws 20 random lines with random colors
#include <stdlib.h>
#include <graphics.h>
#include <conio.h>
#define abs(x) (x<0?-x:x)
#define SIGN(x) (x<0?-1:1)
#define NUM 20
void drawline(int x1,int y1,int x2,int y2)
{if(x2==x1) return;
 if(x2-x1==0)return;
 float slope=1.0*(y2-y1)/(x2-x1);
 int color=getcolor();
 int incx=SIGN(x2-x1);
 for(int x=x1;x!=x2;x+=incx)
 {float y=slope*(x-x1)+y1;
  putpixel(x,y,color);}
return;
}
void main()
{int gdriver=DETECT,gmode=0;
initgraph(&gdriver,&gmode,"d:\\tc\\bgi");
int x1,y1,x2,y2;
randomize();
cleardevice();
for(int i=0;i<NUM;i++)
{x1=random(640);
 x2=random(640);
 y1=random(480);
 y2=random(480);
 drawline(x1,y1,x2,y2);
 setcolor(random(16));
}return;
}
```

Output I



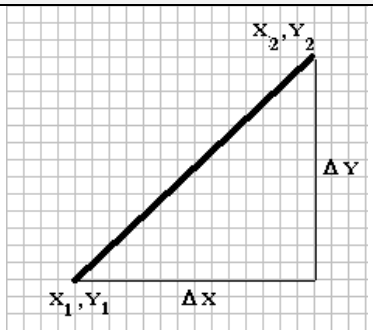
Output II



The D.D.A Line Drawing Method

Objective

To plot a line starting and ending at given points using the Digital Diffrential Analyzer Method.



Theory

Let a line be starting from Point(x1,y1) & ending at Point(x2,y2) . Then Δy per unit increase in Δx along the line path is given by.

$$D = (y2-y1)/(x2-x1) = \Delta y/\Delta x$$

Also,

$$Y` = y + D$$

Code (ddaline.cpp)

```
//graphics
//line plotting using DDA
#include <graphics.h>
#include <conio.h>
#define abs(x) (x<0?-x:x)
void drawline(int x1,int y1,int x2,int y2)
{int dx=x2-x1;
 int dy=y2-y1;
 int step=abs(dy);
 if(abs(dx)>abs(dy))
 step=abs(dx);
 float xincr=1.0*dx/step;
 float yincr=1.0*dy/step;
 for(float x=x1,y=y1;step--;)
 {putpixel(x,y,WHITE);
 x+=xincr;
 y+=yincr;
 }
return;
}
void main()
{char key='a';
int gdriver=DETECT,gmode=0;
initgraph(&gdriver,&gmode,"c:\\tc\\bgi");
int x=100,y=20;
while(key!=27)
{drawline(0,0,x,y);
switch(key)
{case 75:x--;break;
 case 77:x++;break;
 case 72:y--;break;
 case 80:y++;break;
 case 13:case ' ':key=27;continue;
}
key=getch();
cleardevice();
}
return;
}
```

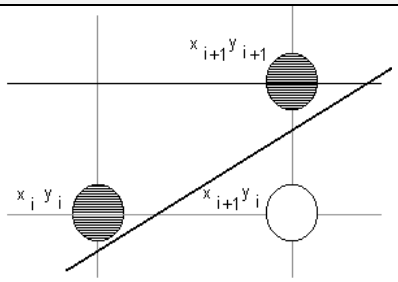
Output I

Output I

The Bresenham Line Drawing Method

Objective

To plot a line starting and ending at given points using the famous Bresenham`s line drawing Method.



Theory

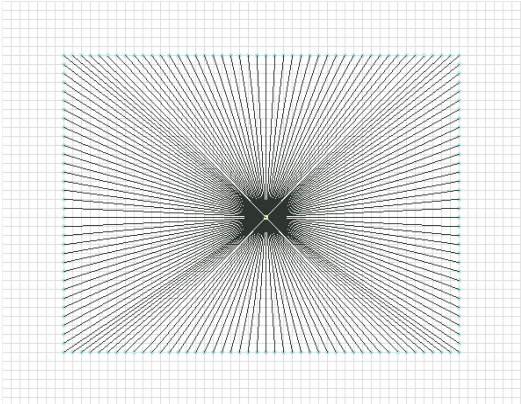
Let a line be starting from Point(x1,y1) & ending at Point(x2,y2) . Then if decision parameter Pi ≥0 then value of y-coordinate is incremented. Where,

$$P_{i+1} = P_i + 2.\Delta y \quad ;\text{if } P_i < 0$$
$$P_{i+1} = P_i + 2.\Delta y - 2.\Delta y \quad ;\text{if } P_i \geq 0$$

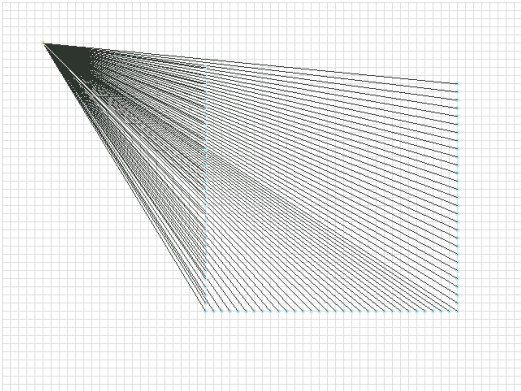
Code (linebre.cpp)

```
//Line drawing Algo using Bresenham`s Method
#include <graphics.h>
#include <conio.h>
#define SIGN(a) (a<0?-1:1)
void swap(int *a,int *b)
{int c=*a; *a=*b,*b=c;}
void drawgrid()
{int maxx=getmaxx(),maxy=getmaxy(), i;
 setcolor(DARKGRAY);
 for(i=0;i<maxx;i+=10) line(i,0,i,maxy);
 for(i=0;i<maxy;i+=10) line(0,i,maxx,i);}
void drawline(int x1,int y1,int x2,int y2)
{ int xx=0,yy=0,mx=1,my=1;  int *x=&xx,*y=&yy;
int dx=x2-x1,dy=y2-y1,p;  x2=dx;y2=dy;
if(dx<0)mx*=-1,dx*=-1,x2*=-1;
if(dy<0)my*=-1,dy*=-1,y2*=-1;
if(dx<dy){int temp=dx;dx=dy,dy=temp,x2=y2,x=&yy,
y=&xx;} for(p=0;x2>=0;(*x)++,x2--)
  { putpixel(x1+mx*xx,y1+my*yy,WHITE);
    if(p>0){p=p+2*dy-2*dx;
            (*y)++; }else  p=p+2*dy; }return;}
void main()
{ int gd=DETECT,gm;
 char keycode='a'; int x1=50,y1=50,x2=70,y2=20;
  initgraph(&gd,&gm,"d:\\tc\\bgi");
  cleardevice(); drawgrid();
 while(keycode!=27)
{ setfillstyle(SOLID_FILL,BLUE);
 fillellipse(x1,y1,2,2); setfillstyle(SOLID_FILL,RED);
 fillellipse(x2,y2,2,2);drawline(x1,y1,x2,y2);
 keycode=getch();
 switch(keycode)
{case 75:x2-=10;break; case 77:x2+=10;break;
 case 72:y2-=10;break; case 80:y2+=10;break;
 case 115:x1-=10;break; case 116:x1+=10;break;
 case -115:y1-=10;break; case -111:y1+=10;break;
 case 13: case ' ':cleardevice(); drawgrid(); break;
}}}
```

Output I



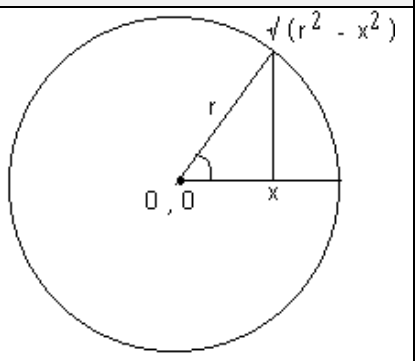
Output II



Cartesian Circle Drawing Program

Objective

To plot a circle at a given center and having required radius using the the Cartesian Equation of a circle.



Theory

Let a circle having radius R and center as Point(0,0). Then the Y co-ordinates for a given value of X is given by:

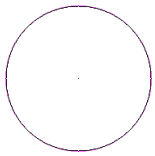
$$Y=\pm\sqrt{(R^2-X^2)};$$

Where, $0 \le X \le R$

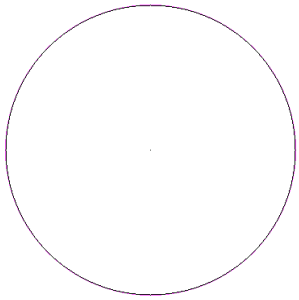
Code (crtcirc.cpp)

```
#include <conio.h>
#include <math.h>
#include <graphics.h>
#define UP 72
#define DOWN 80
#define LEFT 75
#define RIGHT 77
#define CTRLUP -115
#define CTRLDOWN -111
#define CTRLLEFT 115
#define CTRLRIGHT 116
void drawcircle(int cx,int cy,int radius)
{if(radius<=0) return;
 for(int x=0;x<radius/2;x++)
 {int y=sqrt((long)radius*radius-x*x);
  putpixel(cx+x,cy+y,WHITE);
  putpixel(cx+x,cy-y,WHITE);
  putpixel(cx-x,cy+y,WHITE);
  putpixel(cx-x,cy-y,WHITE);
  putpixel(cx+y,cy+x,WHITE);
  putpixel(cx+y,cy-x,WHITE);
  putpixel(cx-y,cy+x,WHITE);
  putpixel(cx-y,cy-x,WHITE); }}
void main()
{int gd=DETECT,gm; char key;
 initgraph(&gd,&gm,"c:\\tc\\bgi");
 int cx=getmaxx()/2,cy=getmaxy()/2,r=50;
 do{ cleardevice(); setcolor(GREEN); circle(cx,cy,r);
drawcircle(cx,cy,r); putpixel(cx,cy,WHITE);
 key=getch();
 switch(key)
 {case UP:  case RIGHT:r-=10;break;
  case LEFT:  case DOWN:r+=10;break;
  case CTRLDOWN:cy+=10;break;
  case CTRLUP:cy-=10;break;
  case CTRLLEFT:cx-=10;break;
  case CTRLRIGHT:cx+=10;break;
 } }while(key!=27);closegraph();}
```

Output I



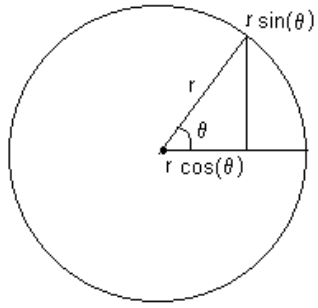
Output II



Polar Circle Drawing

Objective

To plot a circle at a given center and having required radius using the Polar Equation of a circle.



Theory

Let a circle having radius R and center as Point(0,0). Then the X & Y co-ordinates can be evaluated from parametric equation.

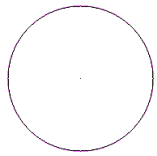
$$X = R.\ Cos\ (\theta)$$
$$Y = R.\ Sin\ (\theta)$$

Where, $0 \le \theta \le 2\pi$

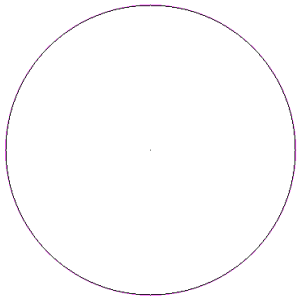
Code (pcircle.cpp)

```
//graphics
//circle drawing using polar equation
#include <math.h>
#include <conio.h>
#include <stdlib.h>
#include <graphics.h>
void drawcircle(int xx,int yy,unsigned radius)
{ float theta=0,inc=1.0/radius;
  for(theta=0;theta<=M_PI_4;theta+=inc)
  {int x=radius*cos(theta);
   int y=radius*sin(theta);
   putpixel(x+xx,y+yy,WHITE);
   putpixel(-x+xx,y+yy,WHITE);
   putpixel(-x+xx,-y+yy,WHITE);
   putpixel(x+xx,-y+yy,WHITE);

   putpixel(y+xx,x+yy,WHITE);
   putpixel(-y+xx,x+yy,WHITE);
   putpixel(-y+xx,-x+yy,WHITE);
   putpixel(y+xx,-x+yy,WHITE);
}


}
void main()
{int gd=DETECT,gm;
initgraph(&gd,&gm,"c:\\tc\\bgi");
randomize();
for(int i=0;i<40;i++)
drawcircle(random(560)+40,random(400)+40,random(4
0)+1);
getch();
}
```

Output I



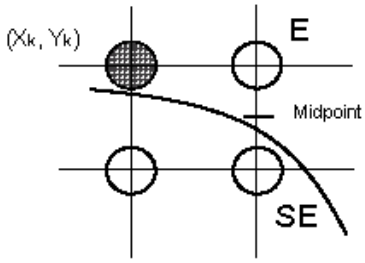
Output II



Mid-Point Circle Drawing Method

Objective

To plot a circle at a given center and having required radius using the Mid Point Circle drawing algorithm.



Theory

Let there be a circle with radius R and center as (0,0). Then if decision parameter Pi ≥0 then value of y-coordinate is decremented.

Where, $P_{i+1} = P_i + 2.X_{i+1} + 1$;if $P_i < 0$

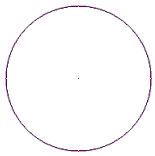
$P_{i+1} = P_i + 2.X_{i+1} + 1 - 2.Y_{i+1}$;if $P_i \geq 0$

Such that, $0 \leq X_i \leq Y_i$

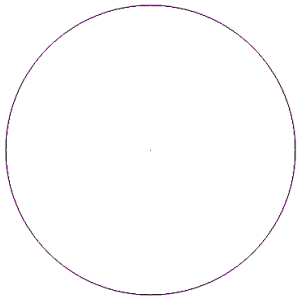
Code (circmid.cpp)

```
#include <conio.h>
#include <graphics.h>
#define UP 72
#define DOWN 80
#define LEFT 75
#define RIGHT 77
#define CTRLUP -115
#define CTRLDOWN -111
#define CTRLLEFT 115
#define CTRLRIGHT 116
void drawcircle(int cx,int cy,int radius)
{int p=1-radius;
 int x2=0,y2=2*radius;
 for(int x=0,y=radius;x<=y;x++,x2+=2)
 {putpixel(cx+x,cy+y,WHITE);
putpixel(cx+x,cy-y,WHITE); putpixel(cx-x,cy+y,WHITE);
putpixel(cx-x,cy-y,WHITE); putpixel(cx+y,cy+x,WHITE);
putpixel(cx+y,cy-x,WHITE); putpixel(cx-y,cy+x,WHITE);
  putpixel(cx-y,cy-x,WHITE);
   if(p<0)  p=p+x2+1;  else
  {p=p+x2+1-y2;   y--;  y2-=2;} }}
void main()
{int gd=DETECT,gm; char key;
 initgraph(&gd,&gm,"c:\\tc\\bgi");
 int cx=getmaxx()/2,cy=getmaxy()/2,r=50;
 do{ cleardevice();setcolor(GREEN); circle(cx,cy,r);
 drawcircle(cx,cy,r); putpixel(cx,cy,WHITE);
 key=getch();
 switch(key)
 {case UP: case RIGHT:r-=10;break;
  case LEFT: case DOWN:r+=10;break;
  case CTRLDOWN:cy+=10;break;
  case CTRLUP:cy-=10;break;
  case CTRLLEFT:cx-=10;break;
  case CTRLRIGHT:cx+=10;break;
 } }while(key!=27);closegraph();}
```

Output I



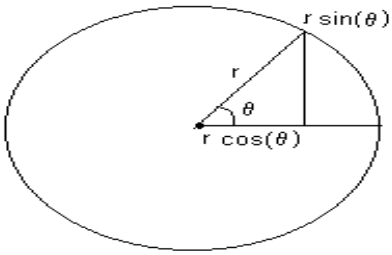
Output II



Cartesian Ellipse Drawing Program

Objective

To plot an ellipse at a given center and having required horizontal and vertical radius using the its Cartesian Equation.



Theory

Let there be an ellipse with axis as (0,0) and with horizontal radius as **A** and vertical radius as **B**. Then for a value of X , Y is given by

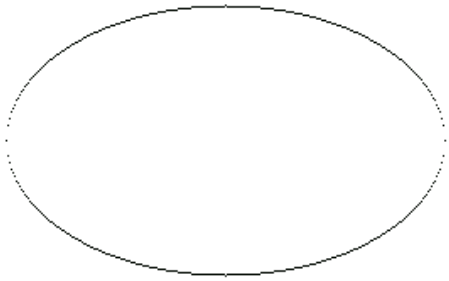
$$Y = \pm B/A \times \sqrt{(4xA^2 - X^2)}$$

Where, $0 \le X \le \mathbf{A}$

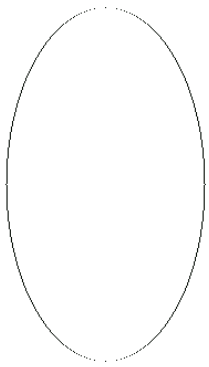
Code (ellipse.cpp)

```
#include <graphics.h>
#include <conio.h>
#include <math.h>
#define LEFT 75
#define RIGHT 77
#define UP 72
#define DOWN 80
void drawellipse(int cx,int cy,int b,int a)
{if(a<=0||b<=0) return;
 int a2=a*a; int b2=b*b;
 if(a>b) for(int x=0;x<=a;x++)
 {int y=(int)sqrt(1.0*(a2-x*x)/a2*b2);
putpixel(cx+x,cy+y,WHITE);putpixel(cx+x,cy-y,WHITE);
putpixel(cx-x,cy+y,WHITE);putpixel(cx-x,cy-y,WHITE);
} else
 for(int y=0;y<=b;y++)
 {int x=(int)sqrt(1.0*(b2-y*y)/b2*a2);
  putpixel(cx+x,cy+y,WHITE);
  putpixel(cx+x,cy-y,WHITE);
  putpixel(cx-x,cy+y,WHITE);
  putpixel(cx-x,cy-y,WHITE);
 }}
void main()
{int gd=DETECT,gm;
initgraph(&gd,&gm,"c:\\tc\\bgi");
int a=150,b=50;
char key;
do
{cleardevice();
 drawellipse(getmaxx()/2,getmaxy()/2,a,b);
 key=getch();
 if(key==UP) a+=10;
 if(key==DOWN) a-=10;
 if(key==RIGHT) b+=10;
 if(key==LEFT) b-=10;
}while(key!=27);}
```

Output I



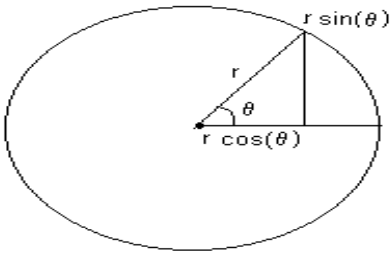
Output II



Polar Ellipse Drawing Program

Objective

To plot an ellipse at a given center and having required horizontal and vertical radius using the its Polar Equation.



Theory

Substituting X=R.Cos(θ) and X=R.Sin(θ) in the Cartesian equation of ellipse we get,

$$R=AxB/\sqrt{[B^2Cos^2(\theta)-A^2Sin^2(\theta)]}$$

So, $X=RxCos(\theta)$

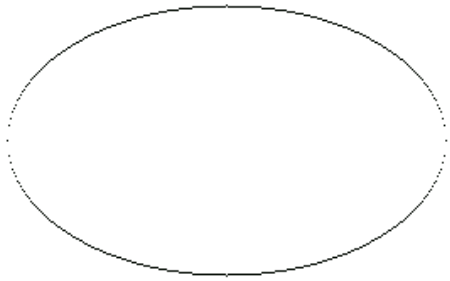
$Y=RxSin(\theta)$

Such that, $0 \le \theta \le 2\pi$

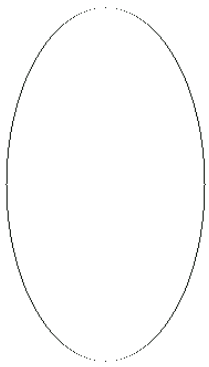
Code (pellipse.cpp)

```
//graphics
//Drawing and Ellipse with give value of center and A,B
//using its cartesian equation
//(c) Vaibhav Jain
//date: 14/4/04;
#include <graphics.h>
#include <conio.h>
#include <math.h>
#define LEFT 75
#define RIGHT 77
#define UP 72
#define DOWN 80
void drawellipse(int cx,int cy,int b,int a)
{ for(float theta=0;theta<=M_PI_2;theta+=0.01)
  {float r=a*b/sqrt((float)b*b*cos(theta)*cos(theta)+
                    a*a*sin(theta)*sin(theta));
  int x=r*cos(theta);
  int y=r*sin(theta);
  putpixel(cx+x,cy+y,WHITE);
  putpixel(cx+x,cy-y,WHITE);
  putpixel(cx-x,cy+y,WHITE);
  putpixel(cx-x,cy-y,WHITE);
  }
}
void main()
{int gd=DETECT,gm;
initgraph(&gd,&gm,"c:\\tc\\bgi");
int a=150,b=50;
char key;
do
{cleardevice();
 drawellipse(getmaxx()/2,getmaxy()/2,a,b);
 key=getch();
 if(key==UP) a+=10;
 if(key==DOWN) a-=10;
 if(key==RIGHT) b+=10;
 if(key==LEFT) b-=10;
}while(key!=27);
}
```

Output I



Output II



Rotation of A Square

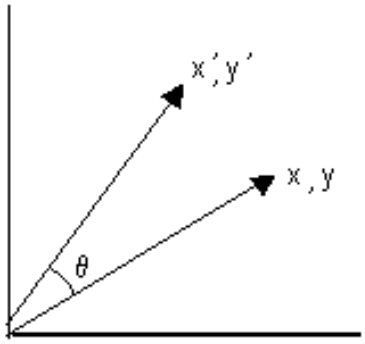
Objective

To simulate a rotating square by applying formulae of planer rotation on its vertexes.

Theory

If any Point(X,Y) is to be roatated by an angle θ , with respect to Origin then following transformation formulae can be applied,

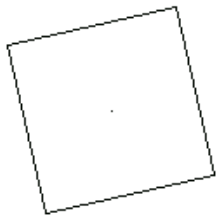
$$X`=X.Cos(\theta) – Y.Sin(\theta)$$
$$Y`=X.Sin(\theta) + Y.Cos(\theta)$$



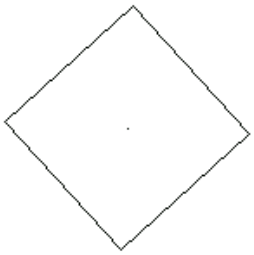
Code (squrot.cpp)

```
#include <math.h>
#include <dos.h>
#include <conio.h>
#include <graphics.h>
class Square{struct Point{float x,y;} points[4]; int degr;
public:Square(int left,int top,int right,int bottom)
{points[0].x=left;points[0].y=top;
 points[1].x=right;points[1].y=top;
 points[2].x=right;points[2].y=bottom;
 points[3].x=left;points[3].y=bottom; degr=0;}
Square & draw()
{for(int i=1;i<4;i++)
 line(points[i-1].x+getmaxx()/2,points[i-].y+getmaxy()/2,
points[i].x+getmaxx()/2,points[i].y+getmaxy()/2);
 line(points[i-1].x+getmaxx()/2,points[i-
1].y+getmaxy()/2,points[0].x+getmaxx()/2,points[0].y+g
etmaxy()/2);
putpixel(getmaxx()/2,getmaxy()/2,WHITE);
return *this;}
Square & rotate(int deg)
{double rad=-M_PI*deg/180;
for(int i=0;i<4;i++){Point p=points[i];
 points[i].x= p.x*cos(rad)-p.y*sin(rad);
 points[i].y= p.y*cos(rad)+p.x*sin(rad);
}degr+=deg;return *this;}
};
void main()
{int gd=DETECT,gm; char key;int del=10;
initgraph(&gd,&gm,"c:\\tc\\bgi");
Square sq(-50,-50,  +50,+50);
Square sqs[]={sq,sq.rotate(6),sq.rotate(6),sq.rotate(6)};
do{setcolor(WHITE);for(int i=0;i<1;i++)
sqs[i].rotate(-1).draw(); delay(del); setcolor(BLACK);
for(i=0;i<1;i++) sqs[i].draw();
if(kbhit())switch(key=getch())
            {case 80:del+=2;break;
             case 72:del-= (del>0?2:0);break;
             default:break; }}while(key!=27);}
```

Output I



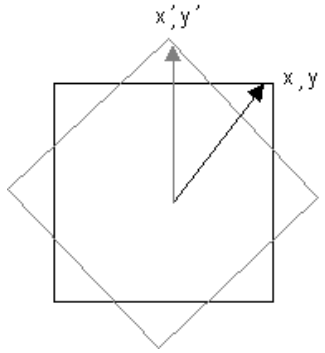
Output II



Rotation of A Polygon

Objective

To create an interactive program to simulate rotation of a polygon where number of sides the polygon has can be selected at the run time.



Theory

The Rotational Transformation discussed in the previous program can be equally applied to a polygon with **N** number of sides. The transformation formulae remain the same.

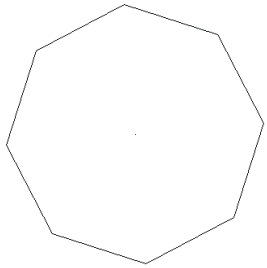
$$Xi`=Xi.Cos(\theta) – Yi.Sin(\theta)$$
$$Yi`=Xi.Sin(\theta) + Yi.Cos(\theta)$$

Where, I = 1,2,3….N

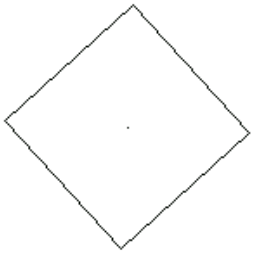
Code (polyrot.cpp)

```
#include <dos.h>
#include <math.h>
#include <conio.h>
#include <graphics.h>
#define UP 72
#define DOWN 80
#define LEFT 75
#define RIGHT 77
#define CTRLUP -115
#define CTRLDOWN -111
#define CTRLLEFT 115
#define CTRLRIGHT 116
void mydrawpoly(int sides,int r,int x,int y,int phase=0)
{if(sides<=0) return;
 float theta=2*M_PI/sides;
 float t=M_PI/180*phase;
 int x1=x+r*cos(t),x2;
 int y1=y+r*sin(t),y2;
 while(sides--)
 {t+=theta; x2=x+r*cos(t); y2=y+r*sin(t);
line(x1,y1,x2,y2);
  x1=x2,y1=y2; }}
void main()
{int gd=DETECT,gm; int phase=0; int sides=3;
 int radius=60; char key=0; int x=getmaxx()/2;
 int y=getmaxy()/2; initgraph(&gd,&gm,"d:\\tc\\bgi");
 do {phase++; putpixel(x,y,WHITE);
mydrawpoly(sides,radius,x,y,phase);
delay(10);cleardevice();
 if(kbhit()) switch(key=getch())
 {case UP:sides++;break;
  case DOWN:sides--;break;
  case LEFT:radius-=10;break;
  case RIGHT:radius+=10;break;
  case CTRLUP:y-=10;break;
  case CTRLDOWN:y+=10;break;
  case CTRLLEFT:x-=10;break;
  case CTRLRIGHT:x+=10;break;
 } }while(key!=27);closegraph();}
```

Output I



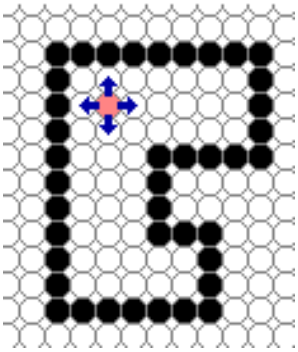
Output II



4-Point Recursive Boundary Fill

Objective

To implement the recursive boundary fill algorithm in C++. Fills 4-connected points to a given and continues recursively.



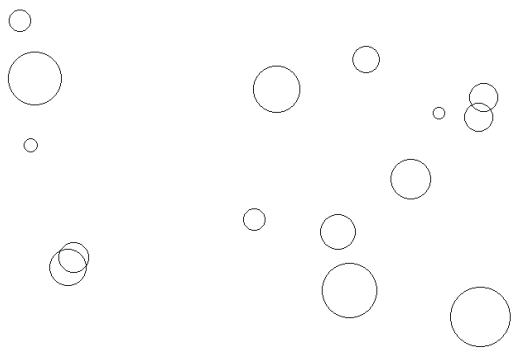
Theory

4-Point Boundry fill algorithm is an Area filling algorithm that uses recursion to fill any area enclosed by a specified color boundry. The algorithm ustilizes a procecudure that check the color of seed pixel . If it matches the specified boundry color that the procedure exits otherwise it fills the seed pixel with the fill color and recursively calls itself with Upper, Lower , Right & Left Pixel specified as seed pixels.

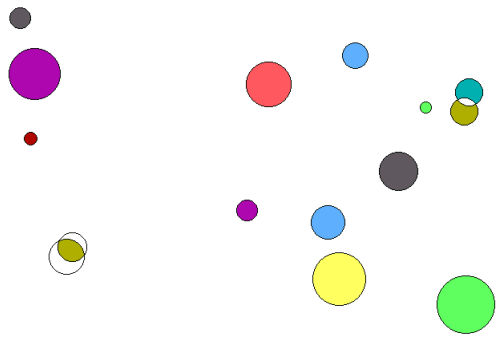
Code (bfill4.cpp)

```
//4-connected boundry fill algorithm
//draws 30 random circles of random radius and fills
with
// random color using 4 point connected boundry filling
#include <graphics.h>
#include <conio.h>
#include <stdlib.h>
#include <dos.h>
#define NUM 50
struct Circle{int cx,cy,radius;}circles[NUM];
void generateCircles()
{for(int i=0;i<NUM;i++)
{circles[i].cx=random(590)+20;
circles[i].cy=random(430)+20;
circles[i].radius=random(40);
 }}
void drawCircles()
{for(int i=0;i<NUM;i++)
 circle(circles[i].cx,circles[i].cy,circles[i].radius);}
void BoundryFill(int x,int y,int boundry,int newColor)
{if(x<0 || x>640) return; if(y<0 || y>480) return;
 int curcol=getpixel(x,y);
 if(curcol==boundry
 ||curcol==newColor) return;
 putpixel(x,y,newColor);
  BoundryFill(x,y+1,boundry,newColor);
  BoundryFill(x,y-1,boundry,newColor);
  BoundryFill(x+1,y,boundry,newColor);
  BoundryFill(x-1,y,boundry,newColor);
}
void main()
{int gd=DETECT,gm;
 initgraph(&gd,&gm,"d:\\tc\\bgi");
 generateCircles(); drawCircles(); getch();
 for(int i=0;i<NUM;i++)
BoundryFill(circles[i].cx,circles[i].cy,WHITE,random(15)
+1);getch();}
```

Output I



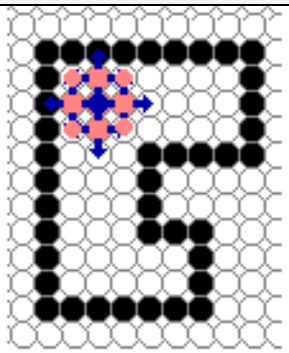
Output II



8-Point Recursive Boundary Fill

Objective

To implement the recursive boundary fill algorithm in C++. Fills 8-connected points to a given and continues recursively.



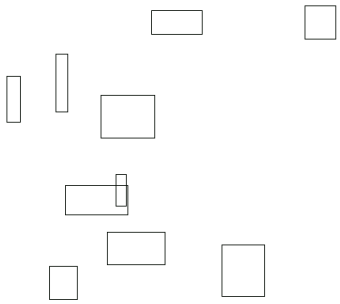
Theory

Like 4-Point Boundry fill this algorithm too is an Area filling algorithm that uses recursion to fill any area enclosed by a specified color boundry. The work in the same way as 4 Point algorithm. However, instead of 4 it recursively fills 8 points around it. These pixels are the Upper, Lower , Right , Left, Upper-Left, Upper-Right, Lower-Left, & the Lower – Right relative to the current seed pixel.

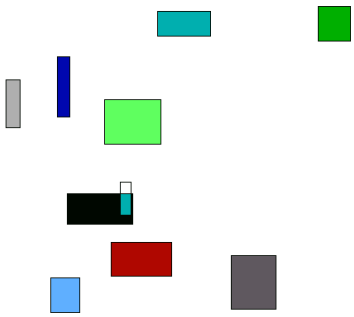
Code (bfill8.cpp)

```
//8-connected boundry fill algorithm
#include <graphics.h>
#include <conio.h>
#include <dos.h>
#include <stdlib.h>
#define NUM 10
struct Rectangle{int x,y,height,width;}rects[NUM];
void generateRects()
{randomize();
for(int i=0;i<NUM;i++)
{rects[i].x=random(570);rects[i].y=random(410);
rects[i].height=random(70)+10;
rects[i].width=random(70)+10;}}

void drawRects()
{for(int i=0;i<NUM;i++)
rectangle(rects[i].x,rects[i].y,
rects[i].x+rects[i].width,rects[i].y+rects[i].height);}

void BoundryFill(int x,int y,int boundry,int newColor)
{if(x<0 || x>640) return; if(y<0 || y>480) return;
 int curcol=getpixel(x,y);
 if(curcol==boundry||curcol==newColor) return;
 putpixel(x,y,newColor);
  BoundryFill(x,y+1,boundry,newColor);
  BoundryFill(x,y-1,boundry,newColor);
  BoundryFill(x+1,y,boundry,newColor);
  BoundryFill(x-1,y,boundry,newColor);
  BoundryFill(x+1,y+1,boundry,newColor);
  BoundryFill(x+1,y-1,boundry,newColor);
  BoundryFill(x-1,y+1,boundry,newColor);
  BoundryFill(x-1,y-1,boundry,newColor);}
void main()
{int gd=DETECT,gm;initgraph(&gd,&gm,"d:\\tc\\bgi");
 generateRects();drawRects();
 getch(); for(int i=0;i<NUM;i++)
BoundryFill(rects[i].x+rects[i].width/2,rects[i].y+rects[i].h
eight/2,WHITE,random(15)+1);getch();}
```

Output I



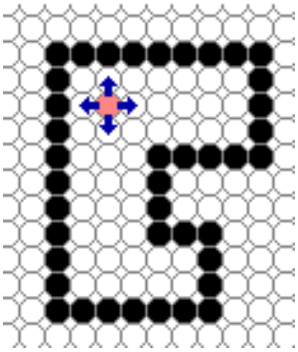
Output II



4-Point Recursive Flood Fill

Objective

To implement the recursive flood fill algorithm . Fills 8-connected points to a given point and continues recursively with the background color.



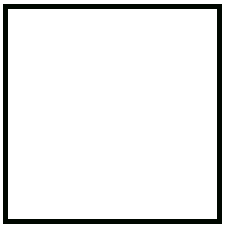
Theory

This too is an Area Fiilling algorithms utilizing recursion and like 4-Point Boundry fill , it too recursively fill 4 pixels around it. However Flood fill algorithms fill a flooded area instead of a bounded area. A flooded area is an area composed of pixel with same color value. Flood fill is usually seen at work in the Paint Brush Program that uses it to fill area with in figures.

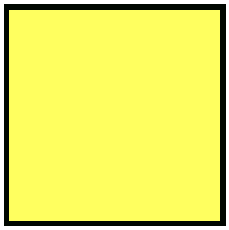
Code (fldfill4.cpp)

```
//graphics
//4-point flood fill algorithm
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <dos.h>
void myfloodfill(int x,int y,int backcolor,int color)
{if(kbhit()) exit(0);
if(getpixel(x,y)!=backcolor) return;
delay(10);
putpixel(x,y,color);
myfloodfill(x,y-1,BLACK,color);
myfloodfill(x,y+1,BLACK,color);
myfloodfill(x-1,y,BLACK,color);
myfloodfill(x+1,y,BLACK,color);
}

void main()
{int gd=DETECT,gm;
 initgraph(&gd,&gm,"c:\\tc\\bgi");
 int points[10]={getmaxx()/2+30,getmaxy()/2+30,
                 getmaxx()/2+30,getmaxy()/2-30,
                 getmaxx()/2-30,getmaxy()/2-30,
                 getmaxx()/2-30,getmaxy()/2+30
                 };

 points[8]=points[0];
 points[9]=points[1];
 randomize();
 drawpoly(5,points);
 getch();

myfloodfill(getmaxx()/2,getmaxy()/2,BLACK,random(15)
);
 printf("done");
 getch();
}
```

Output I



Output II



Boundary Fill With Auxiliary Stack

Objective

To implement the boundary fill algorithm that uses an auxiliary stack to minimize usage of memory and to improve filling speed.

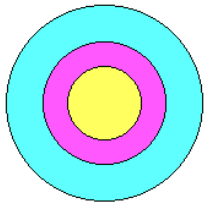
Theory

Area filling algorihms discussed above have a serious disadvatage of making use of recursion for their working. Usually recursion is slow and requires a lot of stack space to work. Thus above algorithms can only be used to fill small areas. However instead of using recursion we can simulate it by using an auxillary stack to store pixel data. Resulting algorithm is quit fast and requires considerably less space to work.

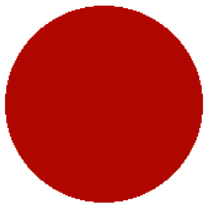
Code (bfillstk.cpp)

```
//graphics
//boundry fill using auxillary stack optimization
//(c) Vaibhav Jain
#include <dos.h>
#include <graphics.h>
#include <conio.h>
#include <.\vaibhav\stack.h>
#define true 1
#define false 0

struct Point        {int x,y;
                     public:  Point(int px,int py):x(px),y(py){}
                              Point(){}
                     };
Stack<Point>s;
void init()
{int gd=DETECT,gm;
 initgraph(&gd,&gm,"c:\\tc\\bgi");
}

void boundryfill(int seedx,int seedy,int boundry,int color)
{Point p(seedx,seedy);
 s.push(p);
 do
 {p=s.pop();
  if(getpixel(p.x,p.y)==boundry) continue;
  while(getpixel(p.x,p.y)!=boundry)p.x--;
  p.x++;
 int flagup=false,flagdown=false;
 int cup,cdwn;
  while(getpixel(p.x,p.y)!=boundry)
  {cup=getpixel(p.x,p.y-1);
   cdwn=getpixel(p.x,p.y+1);
   if(cup==boundry||cup==color) flagup=false;
   else if(!flagup)
   {flagup=true;
    s.push(Point(p.x,p.y-1));
    }
//code continues
```

Output I



Output II



Boundary Fill With Auxiliary Stack

Code (bfillstk.cpp)

```
if(cdwn==boundry||cdwn==color) flagdown=false;
 else if(!flagdown)  {flagdown=true; s.push(Point(p.x,p.y+1)); }
  putpixel(p.x,p.y,color);  p.x++; }}while(!s.isempty()); s.reset();}
void boundryfill(Point seed,int boundry,int color)
{boundryfill(seed.x,seed.y,boundry,color);}
void main()
{init(); setfillstyle(SOLID_FILL,RED);
 fillellipse(getmaxx()/2,getmaxy()/2,80,80);
 setfillstyle(SOLID_FILL,GREEN); fillellipse(getmaxx()/2,getmaxy()/2,50,50);
  setfillstyle(SOLID_FILL,BLUE); fillellipse(getmaxx()/2,getmaxy()/2,30,30);
 getch(); boundryfill(getmaxx()/2,getmaxy()/2,BLACK,LIGHTCYAN);
 getch();
}
```

Code (stack.h)

```
//Stack Library by Vaibhav Jain
//© 2002 All   Rights Reserved
#ifndef STACK.CPP_DEFINED_1123_234
#define STACK.CPP_DEFINED_1123_234
#include <stdlib.h>
#include <conio.h>
#include <stdio.h>
template<class T> class Stack
{int top;
 T * data;
 int size;
public:
Stack(int size=10)
                    {data=new T[size];
                    top=0;this->size=size;}

          Stack & push(T i)
          {if(top>=size)
                    {fprintf(stderr,"Stack Overflow");exit(1);}
           data[top]=i;
           top++;
           return *this;}

          T pop(){if(!top) {fprintf(stderr,"Stack Underflow");exit(1);}
           top--;
           return data[top];}

          ~Stack(void) {delete [] data;}

          int isempty(){return top;}

          int isfull() {return top==size;}
};/**/
#endif
//************************code concludes************************
```

Flood Fill With Auxiliary Stack

Objective

To implement the flood fill algorithm that uses an auxiliary stack to minimize usage of memory and to improve filling speed.

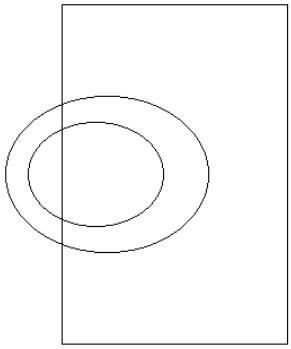
Theory

Area filling algorihms discussed above have a serious disadvatage of making use of recursion for their working. Usually recursion is slow and requires a lot of stack space to work. Thus above algorithms can only be used to fill small areas. However instead of using recursion we can simulate it by using an auxillary stack to store pixel data. Resulting algorithm is quit fast and requires considerably less space to work.

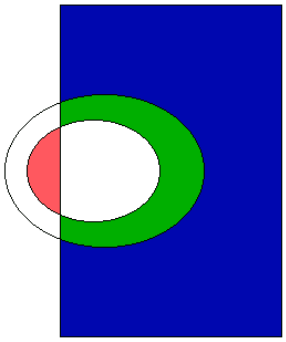
Code (ffillstk.cpp)

```
//graphics
//Flood fill using auxillary stack optimization
//(c) Vaibhav Jain
#include <dos.h>
#include <graphics.h>
#include <dos.h>
#include <conio.h>
#include <.\vaibhav\stack.h>
#define true 1
#define false 0
struct Point        {int x,y;
                    public:  Point(int px,int py):x(px),y(py){}
                             Point(){}
                    };
Stack<Point>s;void init()
{int gd=DETECT,gm;
 initgraph(&gd,&gm,"c:\\tc\\bgi");
}
void floodfill(int seedx,int seedy,int background,int
color)
{Point p(seedx,seedy);
 s.push(p);
 do{int flagtop=true,flagbottom=true;int clrup,clrdwn;
  p=s.pop();
  if(getpixel(p.x,p.y)!=background) continue;
  while(getpixel(p.x,p.y)==background)p.x--;p.x++;
while(getpixel(p.x,p.y)==background)
  {clrup=getpixel(p.x,p.y-1);
   clrdwn=getpixel(p.x,p.y+1);
   if(clrup==background && flagtop==true)
   {flagtop=false;   s.push(Point(p.x,p.y-1)); }else
    if(clrup!=background)
flagtop=true;
  if(clrdwn==background && flagbottom==true)
   {flagbottom=false; s.push(Point(p.x,p.y+1));
    }else    if(clrdwn!=background)
flagbottom=true; putpixel(p.x,p.y,color); p.x++;
} }while(!s.isempty());
 s.reset();}
//code continues
```

Output I



Output II



Flood Fill With Auxiliary Stack

Code (ffillstk.cpp)

```
inline void floodfill(Point seed,int boundry,int color)
{floodfill(seed.x,seed.y,boundry,color);}

void main()
{init();
 rectangle(getmaxx()/2-50,getmaxy()/2-150,
            getmaxx()/2+150,getmaxy()/2+150);
 circle(getmaxx()/2-20,getmaxy()/2,60);
 circle(getmaxx()/2-10,getmaxy()/2,90);
 floodfill(getmaxx()/2+81,getmaxy()/2,BLACK,BLUE);
 floodfill(getmaxx()/2+71,getmaxy()/2,BLACK,GREEN);
 floodfill(getmaxx()/2-51,getmaxy()/2,BLACK,LIGHTRED);
 getch();
}
```

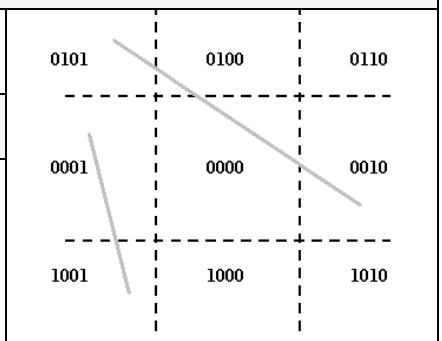
Code (stack.h)

```
//Stack Library by Vaibhav Jain
//© 2002 All   Rights Reserved
#ifndef STACK.CPP_DEFINED_1123_234
#define STACK.CPP_DEFINED_1123_234
#include <stdlib.h>
#include <conio.h>
#include <stdio.h>
template<class T> class Stack
{int top;
 T * data;
 int size;
public:
Stack(int size=10)
                    {data=new T[size];
                    top=0;this->size=size;}

          Stack & push(T i)
          {if(top>=size)
                    {fprintf(stderr,"Stack Overflow");exit(1);}
           data[top]=i;
           top++;
           return *this;}

          T pop(){if(!top) {fprintf(stderr,"Stack Underflow");exit(1);}
           top--;
           return data[top];}

          ~Stack(void) {delete [] data;}

          int isempty(){return top;}

          int isfull() {return top==size;}
};/**/
#endif
//************************code concludes************************
```

Cohen Sutherland Line Clipping Algorithm..

Objective

To implement the Cohen-Sutherland Line Clipping Algorithm in C++. The program draws 30 random lines and clip against given rectangle.



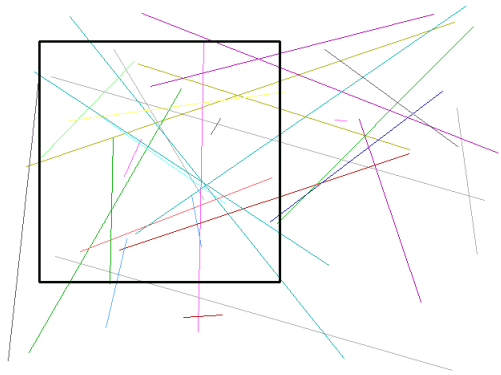
Theory

Cohen- Suterland line clipping algorithm works by assigning Bit-Codes to the two end points of a line. Than this bit codes are Logically And`ed with each other. If the result of And operation is non-zero than tha line is rejected from rendering pipeline. Otherwise the line is fucther processed to determine the clipping boundries.

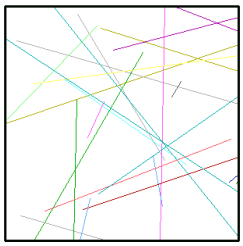
Code (suthrcli.cpp)

```
//conhen sutherland line clipping algorithm
#include <graphics.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#define LEFT 1
#define RIGHT 2
#define TOP 4
#define BOTTOM 8
#define NUM 30
#define BETWEEN(x,a,b) ( ((x)>=(a) && (x)<=(b)) ||
((x)>=(b) && (x)<=(a)) )

struct point{int x,y;};
struct line{int x1,y1,x2,y2,color;};
typedef struct line rect;
typedef int Chcode;
struct line lines[NUM];

void cliplineatrect(struct line &l,rect r,Chcode *a)
{if(*a&LEFT)
        { int y=l.y1+(r.x1-l.x1)*1.0*(l.y2-l.y1)/(l.x2-l.x1);
          if(BETWEEN(y,r.y1,r.y2))
           *a=0,l.x1=r.x1,l.y1=y;
        }
 if(*a&RIGHT)
        { int y= l.y1+(r.x2-l.x1)*1.0*(l.y2-l.y1)/(l.x2-l.x1);
          if(BETWEEN(y,r.y1,r.y2))
           *a=0,l.x1=r.x2,l.y1=y;
        }
 if(*a&TOP)
        { int x= l.x1+(r.y1-l.y1)*1.0*(l.x2-l.x1)/(l.y2-l.y1);
          if(BETWEEN(x,r.x1,r.x2))
           *a=0,l.x1=x,l.y1=r.y1;
        }
 if(*a&BOTTOM)
        { int x= l.x1+(r.y2-l.y1)*1.0*(l.x2-l.x1)/(l.y2-l.y1);
          if(BETWEEN(x,r.x1,r.x2))
//code continues
```

Output I



Output II



Cohen Sutherland Line Clipping Algorithm.

Code (suthrcli.cpp)

```
*a=0,l.x1=x,l.y1=r.y2;}}

void init()
{randomize();
for(int i=0;i<NUM;i++)
 { lines[i].x1=random(640); lines[i].x2=random(640);
   lines[i].y1=random(480);lines[i].y2=random(480);
   lines[i].color=random(15); }}
inline void drawline(struct line l,int /*index*/)
            {setcolor(l.color); line(l.x1,l.y1,l.x2,l.y2);}
void drawlines()
            {for(int i=0;i<NUM;i++) drawline(lines[i],i);}

Chcode getCohenCode(int x,int y,rect r)
{Chcode c=0;
if(x<r.x1) c|=LEFT;
if(x>r.x2) c|=RIGHT;
if(y<r.y1) c|=TOP;
if(y>r.y2) c|=BOTTOM;
return c;}

void swappoints(struct line &l)
{int temp=l.x1;l.x1=l.x2,l.x2=temp;
 temp=l.y1;l.y1=l.y2,l.y2=temp;}

void drawclipped(rect r)
{ for(int i=0;i<NUM;i++)
 {int a=getCohenCode(lines[i].x1,lines[i].y1,r);//code for x1,y1
  int b=getCohenCode(lines[i].x2,lines[i].y2,r);//code for x2,y2
  if((a&b)!=0) continue; //line is rejected
  cliplineatrect(lines[i],r,&a);
  swappoints(lines[i]);
  cliplineatrect(lines[i],r,&b);
  if(!(a|b))/**/
  drawline(lines[i],i);}}

void drawrectangle(rect r)          {rectangle(r.x1,r.y1,r.x2,r.y2);}

void main()
{int gd=DETECT ,gm;
initgraph(&gd,&gm,"d:\\tc\\bgi");
rect r={50,50,350,350};
init();drawlines();
setcolor(WHITE);setlinestyle(SOLID_LINE,0,3);
drawrectangle(r);getch();
cleardevice();setlinestyle(SOLID_LINE,0,1);
drawclipped(r);setcolor(WHITE);
setlinestyle(SOLID_LINE,0,3);
drawrectangle(r);getch();
return;
}
//************************code concludes************************
```

Polygon-Edge Clipping Algorithm …

Objective

To implement the Cohen-Sutherland Line Clipping Algorithm to clip edges of a given polygon.

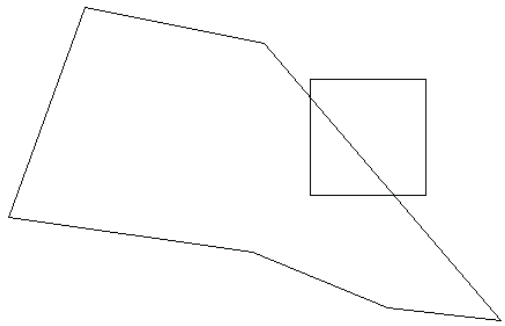
Theory

Although a polygon is a collection of lines and Cohen- Suterland line clipping algorithm can be employed to clip to lined against a rectabngle. But, the polygon loosed its structural defination and cannot be further processed as a polygon. This problem is illustrated in this program which clips a polygon by clipping its edges using Cohen-Sutherland line clipping algorithm.

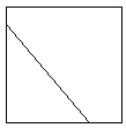
Code (plylnclp.cpp)

```
//Polygon Clipping using cohen-sutherland
//line clipping algorithm
#include <stdlib.h>
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
#define LEFT 1
#define RIGHT 2
#define TOP 4
#define BOTTOM 8
#define BETWEEN(x,a,b) ( ((x)>=(a) && (x)<=(b)) ||
((x)>=(b) && (x)<=(a)) )
#define MAXSIDES 100
#define SIDES 6
typedef int Chcode;
typedef struct {int x1,y1,x2,y2;}Line,Rect;
struct Point{int x,y;};
struct Polygon{Point points[MAXSIDES+1];int count;};
Polygon p;

int isinside(Point p,Rect r)
{if(p.x<r.x1) return 0;
 if(p.x>r.x2) return 0;
 if(p.y<r.y1) return 0;
 if(p.y>r.y2) return 0;
return 1;
}

void addpolypoint(Polygon & p,Point pt)
{ int k=p.count-1;
  if(p.count==0||p.points[k].x!=pt.x||p.points[k].y!=pt.y)
  {p.points[p.count].x=pt.x;
   p.points[p.count].y=pt.y;
   p.count++;
  }}
void drawpolygon(Polygon &p)
{p.points[p.count]=p.points[0];
 drawpoly(p.count+1,(int *) p.points);
//code continues
```

Output I



Output II



Polygon-Edge Clipping Algorithm..

Code (plylnclp.cpp)

```
for(int i=0;i<p.count;i++)
 {char buff[4];
 }
 setcolor(WHITE);
}

void generatepolygon(Polygon &p,int sides)
{randomize();
 if(sides>MAXSIDES) sides=MAXSIDES;
  for(int i=0;i<sides;i++)
  {p.points[i].x=random(640);
   p.points[i].y=random(480);
  }
  p.count=sides;
}

Chcode getCohenCode(Point p,Rect r)
{Chcode c=0;
if(p.x<r.x1) c|=LEFT;
if(p.x>r.x2) c|=RIGHT;
if(p.y<r.y1) c|=TOP;
if(p.y>r.y2) c|=BOTTOM;
return c;
}

void cliplineatrect(Line &l,Rect r,Chcode *a)
{if(*a&LEFT)
            { int y=l.y1+(r.x1-l.x1)*1.0*(l.y2-l.y1)/(l.x2-l.x1);
              if(BETWEEN(y,r.y1,r.y2))
               *a=0,l.x1=r.x1,l.y1=y;
            }
 if(*a&RIGHT)
            { int y= l.y1+(r.x2-l.x1)*1.0*(l.y2-l.y1)/(l.x2-l.x1);
              if(BETWEEN(y,r.y1,r.y2))
               *a=0,l.x1=r.x2,l.y1=y;
            }

 if(*a&TOP)
            { int x= l.x1+(r.y1-l.y1)*1.0*(l.x2-l.x1)/(l.y2-l.y1);
              if(BETWEEN(x,r.x1,r.x2))
               *a=0,l.x1=x,l.y1=r.y1;
            }
if(*a&BOTTOM)
            { int x= l.x1+(r.y2-l.y1)*1.0*(l.x2-l.x1)/(l.y2-l.y1);
              if(BETWEEN(x,r.x1,r.x2))
               *a=0,l.x1=x,l.y1=r.y2;
            }}

void swappoints(Line &l)
{int temp=l.x1;l.x1=l.x2,l.x2=temp;
 temp=l.y1;l.y1=l.y2,l.y2=temp;
}

//code continues
```

Polygon-Edge Clipping Algorthm.

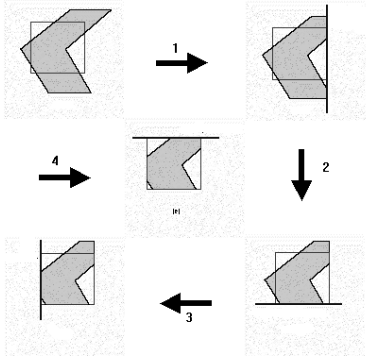
Code (plylnclp.cpp)

```
Polygon clippolygon(Polygon & p,Rect r)
{
#define NEXTP(x) (((x)+1)%p.count)
Polygon np={0};
 for(int i=0;i<p.count;i++)
 {int a=getCohenCode(p.points[i],r);
  int b=getCohenCode(p.points[NEXTP(i)],r);
  if(a&b) continue;
  Line l={p.points[i].x,p.points[i].y,p.points[NEXTP(i)].x,
          p.points[NEXTP(i)].y};
  cliplineatrect(l,r,&a);
  putpixel(l.x1,l.y1,RED);
  swappoints(l);
  cliplineatrect(l,r,&b);
  putpixel(l.x1,l.y1,RED);
  swappoints(l);
  if(!(a|b))
  {Point pt={l.x1,l.y1};
   addpolypoint(np,pt);
   pt.x=l.x2;pt.y=l.y2;
   addpolypoint(np,pt);
  }
 }
if(np.points[np.count-1].x==np.points[0].x&&
   np.points[np.count-1].y==np.points[0].y) np.count--;
return np;
}
void makepolygon(Polygon &p)
{Point p2[]=   {{getmaxx()/2-30-40,getmaxy()/2-30+30},
                {getmaxx()/2-30-40,getmaxy()/2+30+30},
                {getmaxx()/2+30-40,getmaxy()/2+30+30},
                {getmaxx()/2+30-40,getmaxy()/2-30+30},
                };
 p.points[0]=p2[0]; p.points[1]=p2[1]; p.points[2]=p2[2]; p.points[3]=p2[3];
 p.count=4;
}
void main()
{int gd=DETECT,gm;
 initgraph(&gd,&gm,"d:\\tc\\bgi");
 Rect r={getmaxx()/2-50,getmaxy()/2-50,getmaxx()/2+50,getmaxy()/2+50};
 settextjustify(CENTER_TEXT,CENTER_TEXT);
 generatepolygon(p,SIDES);
 drawpolygon(p); rectangle(r.x1,r.y1,r.x2,r.y2);
 getch(); cleardevice(); rectangle(r.x1,r.y1,r.x2,r.y2);
 Polygon p2=clippolygon(p,r);
 drawpolygon(p2);
 getch();
}

//**********************code concludes**********************
```

Sutherland-Hodgeman Polygon Clipping….

Objective

To implement the Sutherland-Hodgeman Polygon clipping Algorithm in C++ that takes a polygon and clips against to given rectangle.



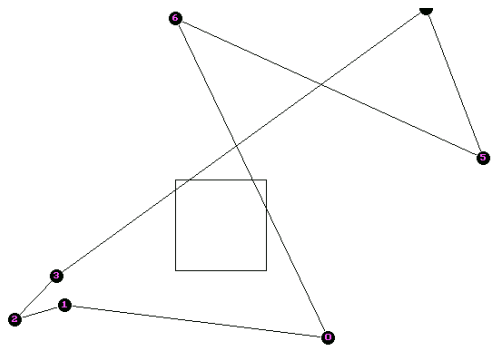
Theory

Suther-Hodgeman Algorithm clips a polygon against a given rectangle and the result is a polygon instead of set of lines. It works by successively clipping the polygon against all four edges of the clip rectangle. I.e First Polygon is clipped against Left Edge then Top then Right and than Bottom.

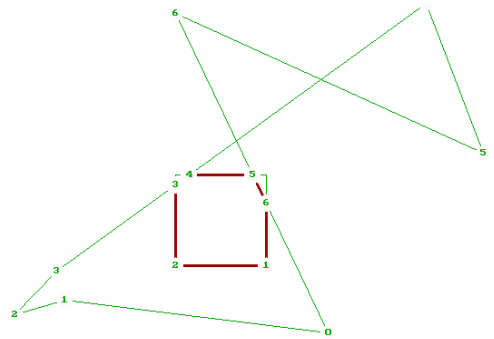
Code (suthhodge.cpp)

```
//graphics
//sutherland-hodgement polygon clipping algorithm
#include <stdlib.h>
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
#define BETWEEN(x,a,b) ( ((x)>=(a) && (x)<=(b)) ||
((x)>=(b) && (x)<=(a)) )
#define MAXSIDES 30
#define SIDES 7
typedef int Chcode;
typedef struct {int x1,y1,x2,y2;}Line,Rect;
struct Point{int x,y;
                Point(int xx,int yy):x(xx),y(yy){}
                Point():x(0),y(0){}
               };
struct Polygon{Point points[MAXSIDES+2];int count;};
struct EdgeList{Line lines[MAXSIDES+1];int count;};

enum Edge{Left,Right,Top,Bottom};
Polygon p;
void drawEdgeList(EdgeList &e)
{for(int i=0;i<e.count;i++)
 line(e.lines[i].x1,e.lines[i].y1,e.lines[i].x2,e.lines[i].y2);
}

Polygon & buildPolygonFromEdgeList(const EdgeList
&els,Polygon &np)
{np.count=0;
if(els.count==0) return np;
 Point *arr=(Point*)els.lines;
 np.points[0]=arr[0];
 for(int i=1;i<2*els.count;i++)
 if(arr[i].x!=np.points[np.count].x||
    arr[i].y!=np.points[np.count].y)
          np.count++, np.points[np.count]=arr[i];
np.count++;
return np;}
//code continues
```

Output I



Output II



Sutherland-Hodgeman Polygon Clipping…

Code (suthhodge.cpp)

```
void createEdgeList(Polygon &p, EdgeList &edl)
{edl.count=p.count;
#define NEXTP(x) (((x)+1)%p.count)
  for(int i=0;i<p.count;i++)
  {edl.lines[i].x1=p.points[i].x;
   edl.lines[i].y1=p.points[i].y;
   edl.lines[i].x2=p.points[NEXTP(i)].x;
   edl.lines[i].y2=p.points[NEXTP(i)].y;
  }
}

void addpolypoint(Polygon & p,Point pt)
{ int k=p.count-1;
  if(p.count==0||p.points[k].x!=pt.x||p.points[k].y!=pt.y)
  {p.points[p.count].x=pt.x;
   p.points[p.count].y=pt.y;
   p.count++;
  }
}

void drawpolygon(Polygon &p)
{p.points[p.count]=p.points[0];
 drawpoly(p.count+1,(int *) p.points);
 for(int i=0;i<p.count;i++)
 {char buff[4];
  sprintf(buff,"%d",i);
  setcolor(BLACK);
  fillellipse(p.points[i].x,p.points[i].y,textwidth(buff),textheight(buff));
  setcolor(GREEN);
  outtextxy(p.points[i].x,p.points[i].y,buff);
 }
 setcolor(WHITE);
}
void generatepolygon(Polygon &p,int sides)
{randomize();
 if(sides>MAXSIDES) sides=MAXSIDES;
  for(int i=0;i<sides;i++)
  {p.points[i].x=random(640);
   p.points[i].y=random(480);
  }
  p.count=sides;
}

isInsideEdge(Point &pt,Rect & r,Edge e)
{switch(e)
{case Left: return pt.x>r.x1;
 case Right: return pt.x<r.x2;
 case Top: return pt.y<r.y2;
 case Bottom: return pt.y>r.y1;
 default:return 0;
}
}
//code continues
```

Sutherland-Hodgeman Polygon Clipping..

Code (suthhodge.cpp)

```
int clipLineAtEdge(Line &l,Rect r,Edge e)
{int h;
int a=isInsideEdge(Point(l.x1,l.y1),r,e); //is x1,y1 inside the edge
int b=isInsideEdge(Point(l.x2,l.y2),r,e); //is x2,y2 inside the edge

if(e==Left) h=r.x1;
else if(e==Right) h=r.x2;
else if(e==Top)  h=r.y2;
else if(e==Bottom) h=r.y1;
if(e==Left||e==Right)
     {
          if(a^b)   //is the line crossing the edge
      {int y=l.y1+(h-l.x1)*1.0*(l.y2-l.y1)/(l.x2-l.x1); //get y intercept
           putpixel(h,y,RED);
           if(a==1) l.x2=h,l.y2=y;
           else /*b==1*/ l.x1=h,l.y1=y;
      }
           else if((a&b)==0) return 0;//is the line outside inside
           //the clipping edge no need to clip this line ignore;
           //if a&b==1 then both points are inside the edge
          return 1;
     }
else /*if(e==Top||e==Bottom)*/
     {
      if(a^b)        //is the line crossing the edge
      {int x= l.x1+(h-l.y1)*1.0*(l.x2-l.x1)/(l.y2-l.y1);
           if(a==1) l.x2=x,l.y2=h;
           else /*b==0*/ l.x1=x,l.y1=h;
           putpixel(x,h,RED);
      }
           else if((a&b)==0) return 0;//is the line outside inside
           //the clipping edge no need to clip this line ignore;
           //if a&b==1 then both points are inside the edge
          return 1;
     }
}
void correctEdges(EdgeList &els) //corrects broken edges in edge list
{Line *arr=els.lines;
#define PREEDGE(x) ((x+els.count-1)%els.count)
#define NXTEDGE(x) ((x+1)%els.count)
 for(int i=0;i<els.count;i++)
 if(arr[PREEDGE(i)].x2!=arr[i].x1 ||
    arr[PREEDGE(i)].y2!=arr[i].y1)
          { for(int j=els.count;j>i;j--)
             arr[j]=arr[j-1];
             els.count++;
             arr[i].x1=arr[PREEDGE(i)].x2;
             arr[i].y1=arr[PREEDGE(i)].y2;
             arr[i].x2=arr[NXTEDGE(i)].x1;
             arr[i].y2=arr[NXTEDGE(i)].y1;
          }
}

//code continues
```

Sutherland-Hodgeman Polygon Clipping.

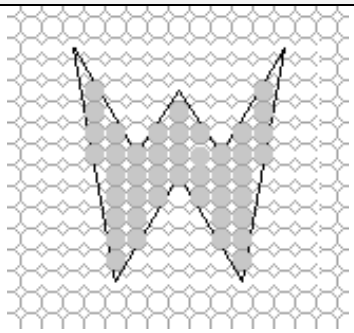
Code (suthhodge.cpp)

```
Polygon &clipPolygon(Polygon & p,Rect r,Polygon &np)
{EdgeList els;
createEdgeList(p,els);
for(Edge e=Left;e<=Bottom;((int)e)++)
 {int count=0;
  for(int i=0;i<els.count;i++)
  if(clipLineAtEdge(els.lines[i],r,e)) //line cannot be ignored
            els.lines[count]=els.lines[i],count++;
 els.count=count; correctEdges(els);
 cleardevice(); setcolor(GREEN);
 drawpolygon(p); setcolor(GREEN);
 rectangle(r.x1,r.y1,r.x2,r.y2);
 setcolor(WHITE); drawEdgeList(els);
 getch();
 }buildPolygonFromEdgeList(els,np);
return np;
}

void makepolygon(Polygon &p)
{Point p2[]=   {Point(getmaxx()/2-30-40,getmaxy()/2-30+30),
                     Point(getmaxx()/2-30-40,getmaxy()/2+30+30),
                     Point(getmaxx()/2+30-40,getmaxy()/2+30+30),
                     Point(getmaxx()/2+30-40,getmaxy()/2-30+30),
                     };
 p.points[0]=p2[0]; p.points[1]=p2[1]; p.points[2]=p2[2]; p.points[3]=p2[3];
 p.count=4;
}

void main()
{int gd=DETECT,gm;
 initgraph(&gd,&gm,"c:\\tc\\bgi");
 Rect r={getmaxx()/2-50,getmaxy()/2-50,getmaxx()/2+50,getmaxy()/2+50};
 settextjustify(CENTER_TEXT,CENTER_TEXT);
 generatepolygon(p,SIDES);
 //makepolygon(p);
 drawpolygon(p);
 rectangle(r.x1,r.y1,r.x2,r.y2);
 getch();

 cleardevice();
 rectangle(r.x1,r.y1,r.x2,r.y2);
 setcolor(GREEN);
 drawpolygon(p);
            Polygon p2;
            clipPolygon(p,r,p2);
            setlinestyle(SOLID_LINE,0,3);
            setcolor(RED);
            drawpolygon(p2);
 getch();
}
//***************************code concludes********************
```

Scan-Line Filling – I..

Objective

To implement the scan line fill algorithm without vertex correction in C++. Takes an arbitrary polygon and fills it with desired color.



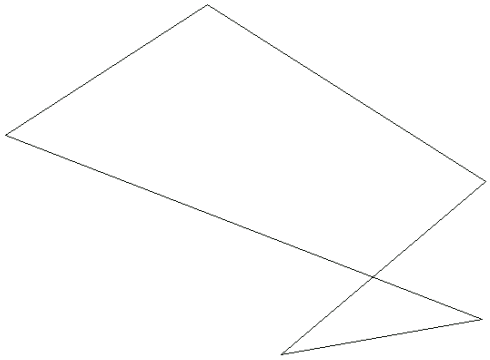
Theory

Instead of relying on recursive scanning techniques we can use scan line techniques to fill polygons. A scan line that starts as well as end outside the polygon, is a line parallel to the X-axis and which intersects altleast one of the polygon edge. Then we can fill the polygon by iterating along this line and start the filling when be hit an edge and stop when we hit another.

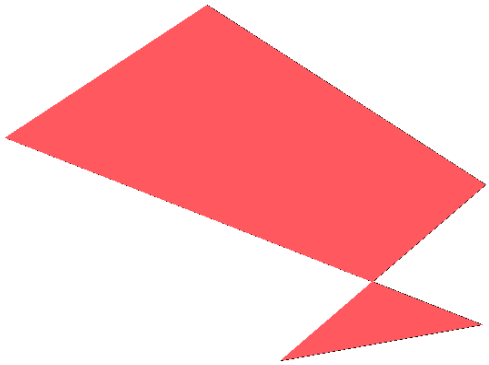
Code (scanfill.cpp)

```
//scanfill algorithm implementation without vertext
correction
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#define SIDES 10
#define MAXSIDES 20
#define BETWEEN(x,a,b) ( ((x)>=(a) && (x)<=(b)) ||
((x)>=(b) && (x)<=(a)) )
struct Point{int x,y;};
struct Polygon{ Point points[MAXSIDES+1];int count;};
typedef struct {int x1,y1,x2,y2;}Line,Rect;
void bubblesort(int c,int * arr)
{c--; for(int i=0;i<c;i++)
  for(int j=0;j<(c-i);j++)
  if(arr[j]>arr[j+1])
 {int temp=arr[j];  arr[j]=arr[j+1];  arr[j+1]=temp;}}
int removeduplicates(int c,int *arr)
{for(int i=0,j=0;i<c;i++)
      if(arr[j]!=arr[i])      arr[++j]=arr[i];
return j+1;}
void drawPolygon(Polygon &p)
{p.points[p.count]=p.points[0];
 drawpoly(p.count+1,(int*) p.points);}
void generatePolygon(Polygon &p,int sides)
{randomize();
 if(sides>MAXSIDES) sides=MAXSIDES;
 if(sides<3) sides=3; for(int i=0;i<sides;i++)
{p.points[i].x=random(640); p.points[i].y=random(480);}
 p.count=sides;}
Rect getBoundingRect(const Polygon &p)
{Rect r={0};
 if(p.count>0){r.x1=p.points[0].x;r.y1=p.points[0].y;
                r.x2=r.x1;r.y2=r.y1;}
 for(int i=0;I<p.count;i++)
{if(p.points[i].x<r.x1) r.x1=p.points[i].x;
  if(p.points[i].y<r.y1) r.y1=p.points[i].y;
//code continues
```

Output I



Output II



Scan-Line Filling – I.

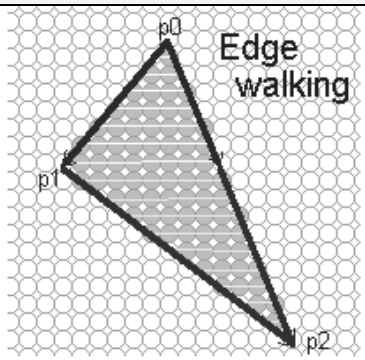
Code (scanfill.cpp)

```
  if(p.points[i].x>r.x2) r.x2=p.points[i].x;
  if(p.points[i].y>r.y2) r.y2=p.points[i].y;
 }
return r;
}
void scanFill(Polygon &p,int color=RED)
{Rect r=getBoundingRect(p);
#define NEXTP(x) ((x+1)%p.count)
 Point (&pts)[]=p.points;
 int *arr=new int[p.count];
 for(int y=r.y1;y<=r.y2;y++)
 {int index=0;
  for(int i=0;i<p.count;i++)
      if(BETWEEN(y,pts[i].y,pts[NEXTP(i)].y))
      {int x= pts[i].x + (y-pts[i].y)
                                *1.0*(pts[NEXTP(i)].x-pts[i].x)
                                /(pts[NEXTP(i)].y-pts[i].y);
       arr[index++]=x;
      }
      else
      continue;
  bubblesort(index,arr);
  index=removeduplicates(index,arr);
  setcolor(color);
  for(i=0;i<index-1;i++)
   if(i%2==0)
   line(arr[i],y,arr[i+1],y);
 }
 delete []arr;
}
void init()
{int gd=DETECT,gm;
 initgraph(&gd,&gm,"c:\\tc\\bgi");
}
void main()
{init();
 Polygon p;
 generatePolygon(p,SIDES);
 drawPolygon(p);
 Rect r=getBoundingRect(p);
 //rectangle(r.x1,r.y1,r.x2,r.y2);
 getch();
 scanFill(p,~LIGHTRED);
 getch();
}

//**********************code concludes**********************
```

Scan-Line Filling –II..

Objective

To implement the scan line fill algorithm with vertex correction in C++. Takes an arbitrary polygon and fills it with desired color.



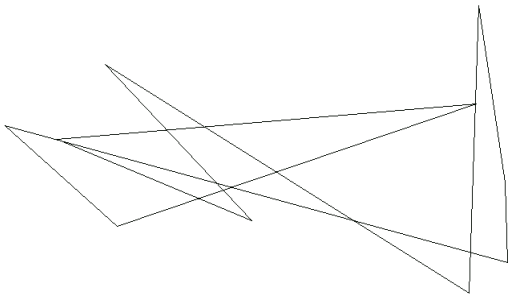
Theory

The previous method explained suffers from the ill condition of vertex intersection, in which case the above method fails. To ensure proper filling we must ensure that no vertex intersect the scan line. This requirment can be satisfied by performing vertex correction on the vertices of the polygon. This procedure detects concave edges and decreses the values of y by unity of enclosing vertex.

Code (scanfil2.cpp)

```
//graphics
//Scanfilling with vertex intersection solved
//by shorting of segments
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#define SIDES 10
#define MAXSIDES 20
#define BETWEEN(x,a,b) ( ((x)>=(a) && (x)<=(b)) ||
((x)>=(b) && (x)<=(a)) )
#define SIGN(x) ((x)<0?-1:1)
struct Point{int x,y;};
struct Polygon{ Point points[MAXSIDES+1];int count;};
typedef struct {int x1,y1,x2,y2;}Line,Rect;
struct EdgeList{Line lines[MAXSIDES+1];int count;};

void bubblesort(int c,int * arr)
{c--; for(int i=0;i<c;i++)  for(int j=0;j<(c-i);j++)
  if(arr[j]>arr[j+1]) {int temp=arr[j];   arr[j]=arr[j+1];
arr[j+1]=temp;}}

void drawPolygon(Polygon &p)
{p.points[p.count]=p.points[0];
drawpoly(p.count+1,(int*) p.points);}

void generatePolygon(Polygon &p,int sides)
{randomize();
 if(sides>MAXSIDES) sides=MAXSIDES;
 if(sides<3) sides=3;
 for(int i=0;i<sides;i++)
{p.points[i].x=random(640); p.points[i].y=random(480);}
 p.count=sides;}
void buildEdgeList(Polygon &p, EdgeList &edl)
{edl.count=p.count;
 Line (&lns)[]=edl.lines;
 Point (&pts)[]=p.points;
#define NEXTP(x) ((x+1)%p.count)
//code continues
```

Output I



Output II

Scan-Line Filling – II.

Code (scanfil2.cpp)

```
  for(int i=0;i<p.count;i++)
  { lns[i].x1=pts[i].x;lns[i].y1=pts[i].y;
    lns[i].x2=pts[NEXTP(i)].x;lns[i].y2=pts[NEXTP(i)].y;
    int a=SIGN(pts[i].y-pts[NEXTP(i)].y);
    int b=SIGN(pts[NEXTP(i)].y-pts[NEXTP(NEXTP(i))].y);
    if(a==b) lns[i].y2+=a*1;
  }
}
void drawEdgeList(EdgeList &e)
{for(int i=0;i<e.count;i++) line(e.lines[i].x1,e.lines[i].y1,e.lines[i].x2,e.lines[i].y2);}

Rect getBoundingRect(const Polygon &p)
{Rect r={0};
 if(p.count>0){r.x1=p.points[0].x;r.y1=p.points[0].y;       r.x2=r.x1;r.y2=r.y1;}
 for(int i=0;i<p.count;i++)
 {if(p.points[i].x<r.x1) r.x1=p.points[i].x;  if(p.points[i].y<r.y1) r.y1=p.points[i].y;
  if(p.points[i].x>r.x2) r.x2=p.points[i].x;  if(p.points[i].y>r.y2) r.y2=p.points[i].y;
 }return r;}

void scanFill(Polygon &p,int color=RED)
{Rect r=getBoundingRect(p);
#define NEXTP(x) ((x+1)%p.count)
 EdgeList e; buildEdgeList(p,e);
 Line (&lns)[]=e.lines;
 int *arr=new int[e.count];
 for(int y=r.y1;y<=r.y2;y++)
 {int index=0;
  for(int i=0;i<e.count;i++)
      if(BETWEEN(y,lns[i].y1,lns[i].y2))
      {int x= lns[i].x1 + (y-lns[i].y1)  *1.0*(lns[i].x2-lns[i].x1) /(lns[i].y2-lns[i].y1);
       arr[index++]=x;
      } else
      continue;
  bubblesort(index,arr);  setcolor(color);
  for(i=0;i<index-1;i++)   if(i%2==0)
   line(arr[i],y,arr[i+1],y);
 } delete []arr;}

void init() {int gd=DETECT,gm; initgraph(&gd,&gm,"c:\\tc\\bgi");}

void main()
{init();
 Polygon p; EdgeList e;
 generatePolygon(p,SIDES);
 drawPolygon(p);
 Rect r=getBoundingRect(p);
getch(); cleardevice();
scanFill(p,LIGHTGRAY);
 getch();
}
//**********************code concludes**********************
```

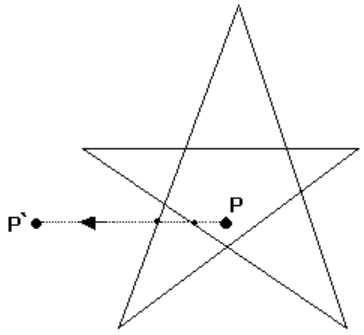
Odd Parity Inside/Outside Test..

Objective

To test whether a given point lies a polygon using the Odd Parity Inside/Outside Test.

Theory

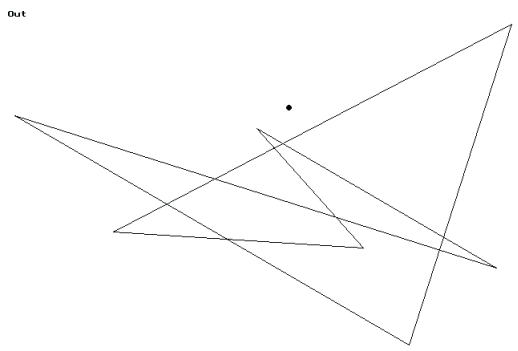
Odd/Even Parity test determines wheather a given point lies inside the boundry of a polygon or not. This test uses the concept of scan line that does not intersect any of the polygon vertices and which starts at the required point and ends at a point outside the polygon. We than determine number of polygon edges the scan line interests. If this number is Odd then the point is inside the polygon else the point is outside the polygon



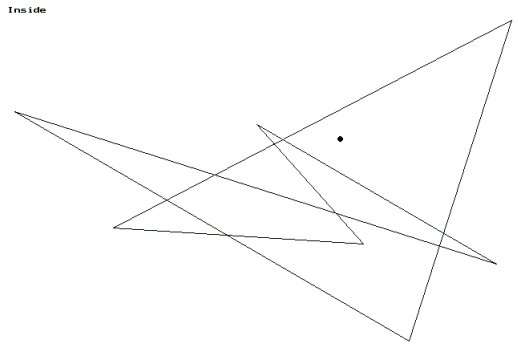
Code (tstinout.cpp)

```
//graphics
//Inside Outside Test Using Odd Parity Rule
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#define SIDES 4
#define MAXSIDES 100
#define BETWEEN(x,a,b) ( ((x)>=(a) && (x)<=(b)) ||
((x)>=(b) && (x)<=(a)) )
#define SIGN(x) ((x)<0?-1:1)
#define LEFT 75
#define RIGHT 77
#define UP    72
#define DOWN  80
#define CTRLLEFT 115
#define CTRLRIGHT 116
#define CTRLUP    -115
#define CTRLDOWN  -111
struct Point{int x,y;};
struct Polygon{ Point points[MAXSIDES+1];int count;};
typedef struct {int x1,y1,x2,y2;}Line,Rect;
struct EdgeList{Line lines[MAXSIDES+1];int count;};
void drawPolygon(Polygon &p)
{p.points[p.count]=p.points[0];
 drawpoly(p.count+1,(int*) p.points);
}
void generatePolygon(Polygon &p,int sides)
{randomize();
 if(sides>MAXSIDES) sides=MAXSIDES;
 if(sides<3) sides=3;
 for(int i=0;i<sides;i++)
{p.points[i].x=random(640); p.points[i].y=random(480);}
 p.count=sides;}
void buildEdgeList(Polygon &p, EdgeList &edl)
{edl.count=p.count;
 Line (&lns)[]=edl.lines;
Point (&pts)[]=p.points;
//code continues
```

Output I



Output II



Odd Parity Inside/Outside Test.

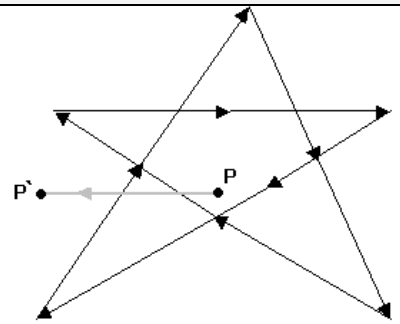
Code (tstinout.cpp)

```
#define NEXTP(x) ((x+1)%p.count)
  for(int i=0;i<p.count;i++)
  { lns[i].x1=pts[i].x;lns[i].y1=pts[i].y;
    lns[i].x2=pts[NEXTP(i)].x;lns[i].y2=pts[NEXTP(i)].y;
    int a=SIGN(pts[i].y-pts[NEXTP(i)].y);
    int b=SIGN(pts[NEXTP(i)].y-pts[NEXTP(NEXTP(i))].y);
    if(a==b) lns[i].y2+=a*1;
  }}

Rect getBoundingRect(const Polygon &p)
{Rect r={0};
 if(p.count>0){r.x1=p.points[0].x;r.y1=p.points[0].y;   r.x2=r.x1;r.y2=r.y1;}
 for(int i=0;i<p.count;i++)
 {if(p.points[i].x<r.x1) r.x1=p.points[i].x;  if(p.points[i].y<r.y1) r.y1=p.points[i].y;
  if(p.points[i].x>r.x2) r.x2=p.points[i].x;  if(p.points[i].y>r.y2) r.y2=p.points[i].y;
 }return r;}
int isInsidePolygon(Point pt,Polygon &p)
{Rect r=getBoundingRect(p);
 EdgeList e;
 buildEdgeList(p,e);
 Line (&lns)[]=e.lines;
 r.x1--; //set the scanpoint be outside the polygon
 int y=pt.y,count=0;
 for(int i=0;i<e.count;i++)
if(BETWEEN(y,lns[i].y1,lns[i].y2))
 {int x= lns[i].x1 + (y-lns[i].y1) *1.0*(lns[i].x2-lns[i].x1) /(lns[i].y2-lns[i].y1);
          if(BETWEEN(x,r.x1,pt.x)) count++; } return count%2;}
void init(){int gd=DETECT,gm; initgraph(&gd,&gm,"c:\\tc\\bgi");}
Polygon & addPoint(Polygon &p,int x,int y)
{Point pt={x,y}; p.points[p.count]=pt; p.count++; return p;}
void main()
{init();
 Polygon p={0}; char key; Point pt={60,70};
 generatePolygon(p,7);
 setfillstyle(EMPTY_FILL,BLACK);
 do{ drawPolygon(p); putpixel(pt.x,pt.y,YELLOW); bar(0,0,50,20);
 if(isInsidePolygon(pt,p))
 outtextxy(0,10,"Inside");
 else outtextxy(0,10,"Out");
 key=getch();  putpixel(pt.x,pt.y,BLACK);
 if(key==UP) pt.y--;
 if(key==DOWN) pt.y++;
 if(key==LEFT) pt.x--;
 if(key==RIGHT) pt.x++;
 if(key==CTRLUP) pt.y-=10;
 if(key==CTRLDOWN) pt.y+=10;
 if(key==CTRLLEFT) pt.x-=10;
 if(key==CTRLRIGHT) pt.x+=10;
}while(key!=27);
 getch();
//************************code concludes*******************
```

Side Winding Inside/Outside Test..

Objective

To determine whether a given point lies a polygon using the Non Zero Side Winding Method.



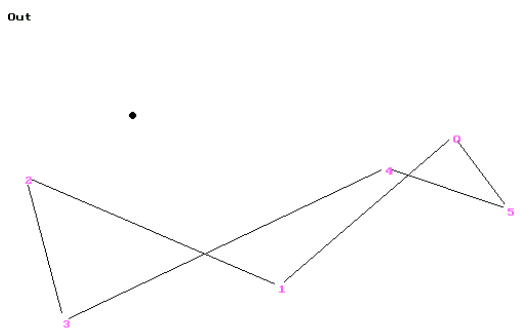
Theory

This test is also similer to the previous test as it too uses a scan line to perform the test at polygon edges. However this test differs in the way that it counts number of times the polygon edges wind around the scan line. If this wind count is odd than the point is inside the polygon else it is not. The wind number of each intersecting edge is determined from its deirection. If it is going from left to right the wind number is +1 otherwise it is –1.

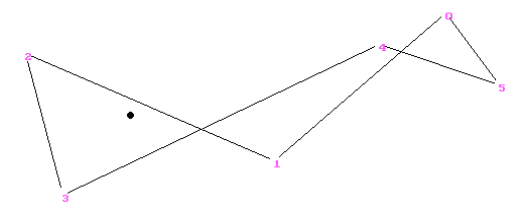
Code (tsinout2.cpp)

```
//graphics
//Inside Outside Test Using Non Zero Side Winding
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#define SIDES 6
#define MAXSIDES 100
#define BETWEEN(x,a,b) ( ((x)>=(a) && (x)<=(b)) ||
((x)>=(b) && (x)<=(a)) )
#define SIGN(x) ((x)<0?-1:1)
#define LEFT 75
#define RIGHT 77
#define UP    72
#define DOWN  80
#define CTRLLEFT 115
#define CTRLRIGHT 116
#define CTRLUP    -115
#define CTRLDOWN  -111
struct Point{int x,y;};
struct Polygon{ Point points[MAXSIDES+1];int count;};
typedef struct {int x1,y1,x2,y2;}Line,Rect;
struct EdgeList{Line lines[MAXSIDES+1];int count;};

void drawPolygon(Polygon &p)
{p.points[p.count]=p.points[0];
 drawpoly(p.count+1,(int *) p.points);
 for(int i=0;i<p.count;i++)
 {char buff[4]; sprintf(buff,"%d",i);setcolor(BLACK);
fillellipse(p.points[i].x,p.points[i].y,textwidth(buff)-
4,textheight(buff));
setcolor(GREEN);
outtextxy(p.points[i].x,p.points[i].y,buff);}
setcolor(WHITE);}

void generatePolygon(Polygon &p,int sides)
{randomize();
//code continues
```

Output I



Output II



Side Winding Inside/Outside Test.

Code (tsinout2.cpp)

```
if(sides>MAXSIDES) sides=MAXSIDES;
 if(sides<3) sides=3;
 for(int i=0;i<sides;i++){p.points[i].x=random(640); p.points[i].y=random(480);}
 p.count=sides;}

void buildEdgeList(Polygon &p, EdgeList &edl)
{edl.count=p.count; Line (&lns)[]=edl.lines; Point (&pts)[]=p.points;
#define NEXTP(x) ((x+1)%p.count)
  for(int i=0;i<p.count;i++)
  { lns[i].x1=pts[i].x;lns[i].y1=pts[i].y;
    lns[i].x2=pts[NEXTP(i)].x;lns[i].y2=pts[NEXTP(i)].y;
    int a=SIGN(pts[i].y-pts[NEXTP(i)].y);
    int b=SIGN(pts[NEXTP(i)].y-pts[NEXTP(NEXTP(i))].y);
    if(a==b) lns[i].y2+=a*1;
  }}

Rect getBoundingRect(const Polygon &p)
{Rect r={0};
 if(p.count>0){r.x1=p.points[0].x;r.y1=p.points[0].y; r.x2=r.x1;r.y2=r.y1;}
 for(int i=0;i<p.count;i++)
 {if(p.points[i].x<r.x1) r.x1=p.points[i].x;  if(p.points[i].y<r.y1) r.y1=p.points[i].y;
  if(p.points[i].x>r.x2) r.x2=p.points[i].x;  if(p.points[i].y>r.y2) r.y2=p.points[i].y;
 }return r;}

int isInsidePolygon(Point pt,Polygon &p)
{Rect r=getBoundingRect(p); EdgeList e;
 buildEdgeList(p,e); Line (&lns)[]=e.lines;
 r.x1--; //set the scanpoint be outside the polygon
 int wind=0; int y=pt.y;
 for(int i=0;i<e.count;i++)
      if(BETWEEN(y,lns[i].y1,lns[i].y2)) //does scanline insersects edge
      {int x= lns[i].x1 + (y-lns[i].y1)  *1.0*(lns[i].x2-lns[i].x1) /(lns[i].y2-lns[i].y1);
            if(BETWEEN(x,r.x1,pt.x)) if(lns[i].y2-lns[i].y1==0)   wind+=SIGN(lns[i].x2-lns[i].x1);
                        else wind+=SIGN(lns[i].y2-lns[i].y1);
 } return wind;}

void init(){int gd=DETECT,gm; initgraph(&gd,&gm,"c:\\tc\\bgi");}
Polygon & addPoint(Polygon &p,int x,int y)
{Point pt={x,y}; p.points[p.count]=pt; p.count++; return p;}
void main()
{ init(); Polygon p={0}; Point pt={60,70};
 generatePolygon(p,SIDES); char key; setfillstyle(EMPTY_FILL,BLACK);
 do{ drawPolygon(p); putpixel(pt.x,pt.y,YELLOW); bar(0,0,50,20);
 if(isInsidePolygon(pt,p))
 outtextxy(0,10,"Inside");
 else outtextxy(0,10,"Out"); key=getch();
 putpixel(pt.x,pt.y,BLACK);
 if(key==UP) pt.y--; if(key==DOWN) pt.y++;
 if(key==LEFT) pt.x--; if(key==RIGHT) pt.x++;
 if(key==CTRLUP) pt.y-=10; if(key==CTRLDOWN) pt.y+=10;
 if(key==CTRLLEFT) pt.x-=10; if(key==CTRLRIGHT) pt.x+=10;
}while(key!=27);}
//******************code concludes*********************
```

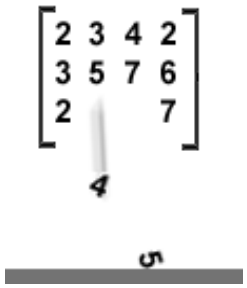
Planer Transformations...

Objective

To implement a Transformation class in C++ that should provide member functions to provide various types of Transformations.

Theory

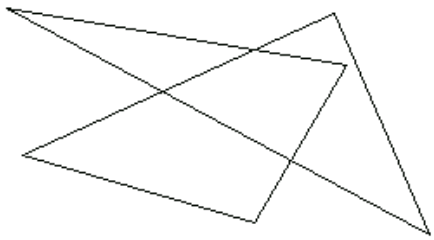
Transformaion Matrices provide an easy way to apply various types of transformations to a point. A Tanformation matrix is a matrix whose elements indicate the type and amount of transformation to be peformed. When the homgenous matrix of a point is multiplied with this matrix the resultant marix contains the transformed point in it.



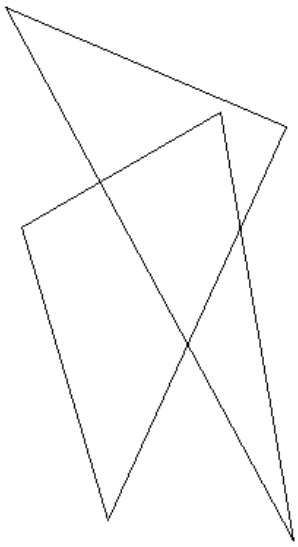
Code (transform.cpp)

```
#include <stdlib.h>
#include <graphics.h>
#include <dos.h>
#include <conio.h>
#include <math.h>
#pragma warn -inl
struct Point
{float x,y;
Point(float xx,float yy):x(xx),y(yy){}
Point():x(0),y(0){}
};
struct Line
{int x1,x2,y1,y2;};
class Transformation;
class Polygon
{ Point pts[30];
int count,fill;
public:
        Polygon()
        {count=fill=0;}
Polygon & addVertex(Point &pt)
        {pts[count++]=pt;return *this;}
Polygon & removeVertex(int index)
        {if(index<count && index>=0)
                {for(int i=index+1;i<count;i++)
                  pts[i-1]=pts[i];
                  count--;
                }
        return *this;
        }
Polygon & insertVertex(int index,Point pt)
        {if(index>=count || index<0) return *this;
         for(int i=count;i>index;i--)
                  pts[i+1]=pts[i];
         count++;
         pts[i]=pt;
         return *this;
        }
//code continues
```

Output I



Output II



Planer Transformations..

Code (transform.cpp)

```
Polygon & applyTransformation(Transformation & t)
{for(int i=0;i<count;i++) pts[i]=t.TransformPt(pts[i]);return *this;}

Polygon Transform(Transformation & t)
        {Polygon p2(*this);p2.applyTransformation(t); return p2;}

void setFill(int i) {fill=i;}
void Draw()
        {if(count==0) return;
         int arr[60];
         for(int i=0;i<count;i++)
                {arr[2*i]=pts[i].x,arr[2*i+1]=pts[i].y;}
         if(!fill)
           {drawpoly(count,arr);
            line(pts[0].x,pts[0].y,pts[count-1].x,pts[count-1].y);
           }  else   fillpoly(count,arr);
         }};

class Transformation
{ float matrix[3][3];
  static void matmult(float (*arr)[3],const float (*arr2)[3])
        { float tarr[3]={0};
          for(int i=0;i<3;i++)
          {for(int j=0;j<3;j++)
             {tarr[j]= arr[i][0]*arr2[0][j];
              tarr[j]+=arr[i][1]*arr2[1][j];
              tarr[j]+=arr[i][2]*arr2[2][j];
             }
           arr[i][0]=tarr[0];
           arr[i][1]=tarr[1];
           arr[i][2]=tarr[2];
          }
        }
 public:
Transformation()
        {matrix[0][0]=1;matrix[0][1]=0;matrix[0][2]=0;matrix[1][0]=0;matrix[1][1]=1;matrix[1][2]=0;
         matrix[2][0]=0;matrix[2][1]=0;matrix[2][2]=1;
        }

void Reset()
        {matrix[0][0]=1;matrix[0][1]=0;matrix[0][2]=0; matrix[1][0]=0;matrix[1][1]=1;matrix[1][2]=0;
         matrix[2][0]=0;matrix[2][1]=0;matrix[2][2]=1;}

Point TransformPt(const Point & pt)
        {float arr[3]; for(int i=0;i<3;i++)
             {arr[i]=pt.x*matrix[0][i]; arr[i]+=pt.y*matrix[1][i];
              arr[i]+=1*matrix[2][i];     }
         arr[0]/=arr[2]; arr[1]/=arr[2];return Point(arr[0],arr[1]);
        }
void Translate(int dx,int dy) // sets transformation to translations
        { Reset();  matrix[2][0]=dx;  matrix[2][1]=dy;}

//code continues
```

Planer Transformations.

Code (transform.cpp)

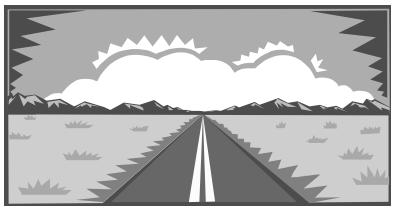
```
void Scale(float sx,float sy)// sets transformation to scaling
        {  Reset(); matrix[0][0]=sx; matrix[1][1]=sy;}

void Rotate(float rad)// sets transformation to rotation
        { Reset(); matrix[0][0]=cos(rad);matrix[0][1]=sin(rad);
         matrix[1][0]=-matrix[0][1];matrix[1][1]=matrix[0][0];
        }
void addTranslation(int dx,int dy) //add given translation to currunt
        { Reset();  matrix[2][0]+=dx;  matrix[2][1]+=dy;}

void addScaling(float sx,float sy)//add given scale factor to currunt
        {  matrix[0][0]*=sx;   matrix[1][1]*=sy;}
void addRotation(float rad)//adds given rotation angle to currunt
        {float tarr[3][3]={0}; tarr[0][0]=cos(rad);tarr[0][1]=sin(rad);
         tarr[1][0]=-tarr[0][1];tarr[1][1]=tarr[0][0];
         tarr[2][2]=1; matmult(matrix,tarr);
        }
void addReflection(int x,int y){if(x) matrix[1][1]*=-1; if(y) matrix[0][0]*=-1;}
void Reflect(int x,int y){Reset(); if(x) matrix[1][1]=-1; if(y) matrix[0][0]=-1;}
Transformation & Compose(Transformation & t){matmult(matrix,t.matrix); return *this;}
};

void init()
{int gd=DETECT,gm;
initgraph(&gd,&gm,"c:\\tc\\bgi");
}

void main()
{ init();
  Transformation t1,t2,t3,t4,T;
  Polygon py;
  t1.Rotate(M_PI_2);
  t2.Scale(1.5,1.5);
  t3.Reflect(1,0);//reflect with respect to x-axis
  t4.Translate(getmaxx()/2,getmaxy()/1.1);

  T.Compose(t1).Compose(t2).Compose(t3).Compose(t4);

  randomize();
  for(int i=0;i<6;i++)
          py.addVertex(Point(random(320),random(240)));

  py.Draw();
   getch();
  cleardevice();

   py.applyTransformation(T);
   py.Draw();
   getch();
}
//************************code concludes************************
```

The 3-D Plane.

Objective

To render a plane in 3 dimension using its two dimensional projection formulae. Also view horizon is simulated to provide dept cueing.

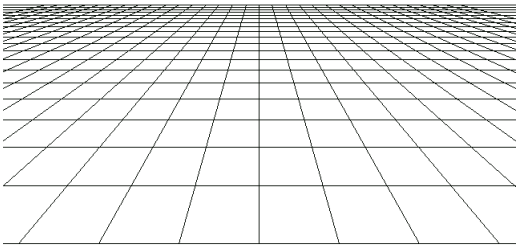
Theory



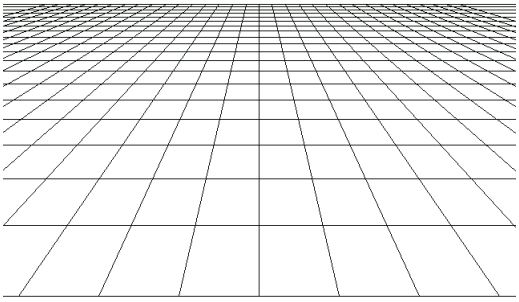
Code (plane.cpp)

```
//graphics
//A 3D View of a Plane diminishing in the Horizan
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#define UP 72
#define LEFT 75
#define RIGHT 77
#define DOWN 80
void main()
{int gd=DETECT,gm;
 char key;
 int viewheight=100,viewdistance=20;
 initgraph(&gd,&gm,"d:\\tc\\bgi");
 do{
 cleardevice();
 if(key==UP) viewheight+=10;
 if(key==DOWN) viewheight-=10;
 if(key==LEFT) viewdistance+=1;
 if(key==RIGHT) viewdistance-=1;
 int y;
 for(int i=0;i<100;i+=5)
 {y=1.0*viewheight*i/(i+viewdistance);
  y=getmaxy()/1.2-y;
  line(0,y,getmaxx(),y);
 }
 for(i=-1600;i<1600;i+=100)
 {int x1=getmaxx()/2+i;
  int
x2=getmaxx()/2+i*viewdistance/(100+viewdistance);
  line(x1,getmaxy()/1.2,x2,y);
 }

 key=getch();
 }while(key!=27);
}
```

Output I



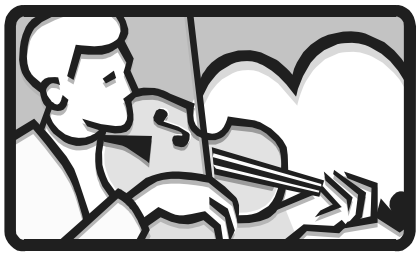
Output II



Computer Art –I.

Objective

Using equations of a circle combined with animations fantastic works of art can be produced. Here is an example.

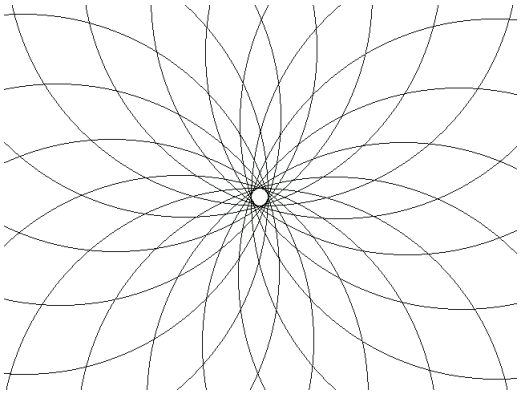


Theory

Code (design1.cpp)

```
//Computer Graphics
#include <dos.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>
void main()
{int gd=DETECT,gm;
initgraph(&gd,&gm,"c:\\tc\\bgi");
int cx=getmaxx()/2,cy=getmaxy()/2;
int r=320;
float phase=0.0;
setfillstyle(SOLID_FILL,BLACK);
while(!kbhit())
{

delay(10);
//cleardevice();
bar(cx-2*r,cy-2*r,cx+2*r,cy+2*r);
phase+=0.01;
for(float t=phase;t<(2*M_PI+phase);t+=M_PI/10)
{ int cxx=cx+(r-10)*cos(t);
  int cyy=cy+(r-10)*sin(t);
  circle(cxx,cyy,r);
}

}
getch();
}
```

Output



The Mouse & The Computer..

Objective

Using Microsoft Mouse Driver Interface create a text based mouse driven free hand drawing program.

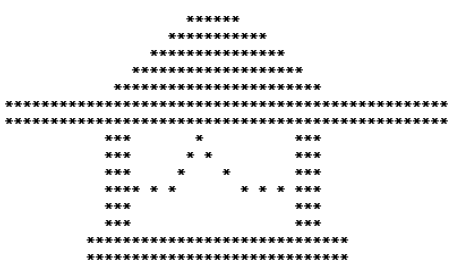
Theory

Microsoft provides a low level interface to mouse via is Mouse Driver. This driver can be accessed via the int 33h interrupt vector. The driver provies various facilities to ineract with the mouse like getting current mouse position, detecting mouse button presses , setting mouse pointer shape etc. Details of the interface can be found in Ray Ducan's MS-DOS Encyclopedia.

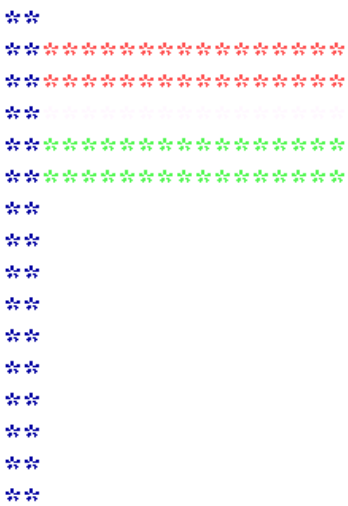
Code (mouse1.cpp)

```
//A text Based Paint Brush Program
#include <stdlib.h>
#include <dos.h>
#include <stdio.h>
#include <conio.h>
#define LEFT_BUTTON 1
#define RIGHT_BUTTON 2
char far *screen=(char*)MK_FP(0xb800,0);
typedef struct {int x,y;}Point;
typedef  unsigned Buttons;
void setMousePtr(unsigned andmask,unsigned
xormask)
{REGS regi;
regi.x.ax=0xa;
regi.x.bx=0;
regi.x.cx=andmask;
regi.x.dx=xormask;
int86(0x33,&regi,&regi);
}
int InitMouse()
{REGS regi;
regi.x.ax=0;
int86(0x33,&regi,&regi);
return !!regi.x.ax;
}
Point * getMousePos(Point *p)
{REGS regi,rego;
regi.x.ax=0x3;
int86(0x33,&regi,&regi);
p->x=regi.x.cx;
p->y=regi.x.dx;
return p;
}
Buttons * getButtonStatus(Buttons *b)
{REGS regi;
regi.x.ax=0x5;
int86(0x33,&regi,&regi);
*b=regi.x.ax;
//code continues
```

Output I



Output II



The Mouse & Computer Art –II.

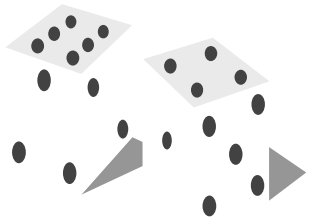
Code (mouse1.cpp)

```
return b;
}
void ShowMouse()
{REGS regi;
regi.x.ax=0x1;
int86(0x33,&regi,&regi);
}
void HideMouse()
{REGS regi;
regi.x.ax=0x2;
int86(0x33,&regi,&regi);
}
int main()
{int color=1;
clrscr();
InitMouse();
//setMousePtr(0xf0ff,color<<8);
*(screen)='*';*(screen+1)=color;
ShowMouse();
Point p;
Buttons b;
while(!kbhit())
 { delay(10);
   getMousePos(&p);
   getButtonStatus(&b);
   if(b&LEFT_BUTTON && b&RIGHT_BUTTON)
    {clrscr();
    *(screen)='*';*(screen+1)=color;
    continue;
    }

   if(b&LEFT_BUTTON)
   {HideMouse();
    p.y<<=2,p.x>>=2,*(screen+p.x+p.y*5)=(char)'*';
   *(screen+p.x+p.y*5+1)=color;
   ShowMouse();
   }
   if(b&RIGHT_BUTTON) {color=(color+1)%16;*(screen+1)=color;}

 }
HideMouse();
return 0;
}
//********************code concludes**********************
```

'Snake'- The Game….

Objective

To create an interactive game called 'Snake' with ability to manage score and different level of difficulties. Als



Theory

Code (snake.c)

```
#include <stdlib.h>
#include <dos.h>
#include <stdio.h>
#include <conio.h>
typedef struct {int x,y;} point;
void generate(void),swappoints(void),render(void),
newfrog(void),msgbox(char mesag[]);
void init(),newlevel();
void fill(unsigned x1,unsigned y1,unsigned x2,
unsigned y2,char ch);
void wait(void),makebox(void);
int collided(void),pause(void);
unsigned int  score=0,lives=3;
int dx,dy,size,sir,level=0;
int delays=3000;
point snake[20],frog,newbone(void);
main(void)
{char nkey='a',temp[2];
int isrender=0;
int options=0;
newl:
newlevel();
while(nkey!=27)
{ isrender++;
  if (options==1)  {newlevel();options=0;}
  if (isrender>=delays)
  {temp[0]=snake[size-1].x;temp[1]=snake[size-1].y;
  gotoxy(temp[0],temp[1]);putch(' ');
  swappoints();
  snake[0].x+=dx;snake[0].y+=dy;
  render();
  isrender=0;
  }
if (options==2)  {size++;options=0;}
 if ((snake[0].x==frog.x)&&(snake[0].y==frog.y))
{score+=level*10;
 putch(7);
//code continues
```

Output

```
+-----------------The Snake----------------+
|                                          |
|                                          |
|                                          |
|                                          |
|                                       *  |
|            ========>                     |
|                                          |
|                                          |
|                                          |
|                                          |
|                                          |
|                                          |
|                                          |
|                                          |
|                                          |
|                                          |
|                                          |
|                                          |
+-----Lives: 3  Level: 1  Score:50   --- -+
```

‘Snake’- The Game...

Code (snake.c)

```
            newfrog();
        options=2;
        if (size>=20) options=1;else snake[size]=newbone();
        render();
        }

  if(kbhit())
  {nkey=getch();
   switch(nkey)
    {case 75:if (dx==0 ){dy=0;dx=-1;sir='<';isrender=delays;render();}break;
     case 77:if (dx==0){dy=0;dx=+1;sir='>';isrender=delays;render();}break;
     case 72:if (dy==0 ){dy=-1;dx=0;sir='^';isrender=delays;render();}break;
     case 80:if (dy==0){dy=+1;dx=0;sir='v';isrender=delays;render();}break;
     case 27: if (pause()){nkey=3;isrender=delays;
                        clrscr();
              makebox();
                        }break;
     case 32:isrender=delays;
    }
  }

            if (collided())
            {lives--;if (lives<=0)
                        {msgbox("  Game Over  ");wait();exit(1);}
                        else
                        {msgbox(" \n\r Oppss.. You Crashed\n\r");wait();init();}
            }
}
printf("Exiting.....");
return 0;
}
void generate()
{ int i;
for(i=0;i<size;i++)
{ snake[i].x=size-i+1;
  snake[i].y=2;
}
}
void swappoints()
{ int i;
 for(i=size-1;i>0;i--)
 {snake[i].x= snake[i-1].x;
 snake[i].y= snake[i-1].y;
 }}
collided(void)
{int i;
if (((snake[0].x)>=79)||((snake[0].x)<=1))  return 1;
if (((snake[0].y)>=25)||((snake[0].y)<=1))  return 1;
for(i=1;i<size;i++)
if ((snake[0].x+dx==snake[i].x)&&(snake[0].y+dy==snake[i].y)) return 1;
return 0;

//code continues
```

'Snake'-The Game..

Code (snake.c)

```
}
void newfrog(void)
{
frog.x=random(75)+2;
frog.y=random(21)+2;
}
point newbone(void)
{point temp;
temp.x=snake[size-1].x+dx;
temp.y=snake[size-1].y+dy;
return temp;
}
void render()
{ static int ja=1,i;
  for(i=0;i<size;i++)
 {gotoxy(snake[i].x,snake[i].y);
  putch((i==0)?sir:219);
}
gotoxy(frog.x,frog.y);
if (ja) putch('O'); else putch('*');
ja=!ja;
gotoxy(40,25);printf("Lives:%2u  Level:%2u  Score:%-5u",lives,level,score);
}
int box(unsigned x1,unsigned y1,unsigned x2,unsigned y2)
{int i;
gotoxy(x1,y1);for(i=x1;i<=x2;i++)putch(205);
gotoxy(x1,y2);for(i=x1;i<=x2;i++)putch(205);
for(i=y1;i<y2;i++){gotoxy(x1,i);putch(186);}
for(i=y1;i<y2;i++){gotoxy(x2,i);putch(186);}
gotoxy(x1,y1);putch(201);gotoxy(x1,y2);putch(200);
gotoxy(x2,y1);putch(187);gotoxy(x2,y2);putch(188);
return 1;
}
void fill(unsigned x1,unsigned y1,unsigned x2,unsigned y2,char ch)
{int i,j;
for(i=x1;i<=x2;i++)
{gotoxy(y1,i);
for(j=y1;j<=y2;j++)
putch(ch);
}
}void msgbox(char mesag[])
{int nlines=0,size=0,i=0,j=0,temp[4];
int maxlen=0;
struct text_info ti;gettextinfo(&ti);
temp[0]=ti.curx;temp[1]=ti.cury;size=strlen(mesag);
temp[2]=ti.screenwidth/2;temp[3]=ti.screenheight/2;
for(i=0;i<=size;i++,j++)
if (mesag[i]=='\n'){nlines++;if(j>maxlen) maxlen=j;j=0;}
if(j>maxlen) maxlen=j;
box(temp[2]-maxlen/2-2,temp[3]-nlines/2-1,temp[2]+maxlen/2+2,temp[3]+nlines/2+1);
window(temp[2]-maxlen/2,temp[3]-nlines/2,temp[2]+maxlen/2,temp[3]+nlines/2);

//code continues
```

‘Snake’-The Game.

Code (snake.cpp)

```
temp[0]=ti.curx;temp[1]=ti.cury;size=strlen(mesag);
temp[2]=ti.screenwidth/2;temp[3]=ti.screenheight/2;
for(i=0;i<=size;i++,j++)
if (mesag[i]=='\n'){nlines++;if(j>maxlen) maxlen=j;j=0;}
if(j>maxlen) maxlen=j;
box(temp[2]-maxlen/2-2,temp[3]-nlines/2-1,temp[2]+maxlen/2+2,temp[3]+nlines/2+1);
window(temp[2]-maxlen/2,temp[3]-nlines/2,temp[2]+maxlen/2,temp[3]+nlines/2);
cprintf("%s",mesag);window(1, 1, 80, 25);
gotoxy(temp[0],temp[1]);}
void init()
{srand(dx*dx*10);
clrscr();
dx=1;dy=0;
size=5;sir='>';
generate();newfrog();
makebox();render();}
void newlevel()
{ char buffer[50];
  srand(clock());
  size=5;  init();
  score+=level*100;
  level++;  delays-=500;
  if(level>=5) sprintf(buffer,"\n\r         Great!!!!\n\r  You Completed the Show"); else
  if(level>1) sprintf(buffer,"       Congrats!!!!\n\r   Starting Level %d  ",level);
  else sprintf(buffer,"  Starting Level %d  ",level);
  msgbox(buffer);
  wait();  clrscr();
  if (level>=5) exit(0);
  makebox();}
int pause(void)
{ char nkey='s';
 msgbox("Paused...\n\rPress Escape To Exit\n\rAny Other Key To Continue");
                 while(!kbhit());
                 nkey=getch();
            if (nkey==27) return 0;
                 else return 1;}
void wait(void )
{ int i='a';while(!((i==13)||(i==27)))
if (kbhit()) i=getch();}
void makebox(void)
{box(1,1,79,25);
gotoxy(35,1);printf("The Snake");}
point midpoint(void)
{  struct text_info ti;point temp;
   gettextinfo(&ti);
   temp.x=ti.screenwidth/2;
   temp.y=ti.screenheight/2;
   return temp;
}
//************************code concludes*********************
```

‘Cars’- The Game…

Objective

To Create an interactive car racing game.

Theory

Code (carrs.c)

```
//project
#include "vjbox.h"
#include <assert.h>
#define GAMEOVER "\n\r        Game Over!!!!       \n\r"
#define RENDER 1000
int score=0,level=1,counter=RENDER;
void drawcarxy(),fill(),render(),regen();
point car={10,10};
point cars[4];
void main()
{void generate();
int nkey='a';
car.y=midpoint().y*1.5;car.x=midpoint().x-4;
clrscr();generate();
while(nkey!=27)
{ if (counter>=RENDER) {render();counter=0;
  if(collided()) {msgbox(GAMEOVER);wait();exit(1);}
  }
  counter++;
  if (kbhit())
  {nkey=getch();
   switch (nkey)
   {case LF_ARROW:if (car.x==midpoint().x-4) break;
                  textcolor(0);drawcarxy(car.x,car.y);
                  textcolor(15);car.x=midpoint().x-4;
                  counter=RENDER;break;
    case RT_ARROW:if (car.x==midpoint().x+4) break;
            textcolor(0);drawcarxy(car.x,car.y);
                  textcolor(15);car.x=midpoint().x+4;
                  counter=RENDER;break;
    case SPACE_BAR :counter=RENDER;break;    case
ESCAPE_KEY:counter=RENDER;break;
} }}}
void generate()
{int i;for(i=0;i<3;i++)
{regen(&cars[i]);
cars[i].y=i*10;}}
//code continues
```

Output

```
*       #          *
* #######          *
*      #           *
*     #            *
* #######          *

*       #          *
* #######          *
*      #           *
*     #            *
* #######          *

*       #          *
* #######          *
*      #           *
*     #            *
* #######          *
*              #   *
*          ########*
*              #   *
*              #   *
*          ####### *    Score 40
```

‘Cars’- The Game..

Code (carrs.c)

```
void regen(point * car)
{car->y=-9;
 if (rand()%2) car->x=midpoint().x-4;
 else car->x=midpoint().x+4;
}
void drawcarxy(int x,int y)
{fill(x-3,y,x+3,y,'#');
 fill(x,y-1,x,y+3,'#');
 fill(x-3,y+3,x+3,y+3,'#');
}
void render()
{void footpath(void);int i;
for(i=0;i<3;i++)
{textcolor(0);
drawcarxy(cars[i].x,cars[i].y);
cars[i].y++;
if (cars[i].y>midpoint().y*2+2) regen(&cars[i]);
textcolor(15);
drawcarxy(cars[i].x,cars[i].y);
}
 drawcarxy(car.x,car.y);
 footpath();
gotoxy(60,24);printf("Score:%u",score);
}
void footpath(void)
{static int count=0;
 point mid;int i;
 mid=midpoint();
 for(i=mid.y*2;i>1;i--,count++)
 { if (count%3==0)
   {  gotoxy(mid.x-9 ,i);putch(' ');
      gotoxy(mid.x+9,i);putch(' ');
   }
     else
   {  gotoxy(mid.x-9 ,i);putch('*');
      gotoxy(mid.x+9,i);putch('*');
   }
  }
if (count>=9000) count=0;
}
int collided(void)
{ int i;point ca;
for(i=0;i<=3;i++)
{ ca=cars[i];
if ((car.x==ca.x)&&between(car.y-ca.y,-3,+3)) return 1;
if (car.y==ca.y){
                        score+=level*10;
                        }
}
return 0;}
int between( int exp,int r1,int r2)
{ if ((exp>=r1)&&(exp<=r2)) return 1; else return 0;
}
//code continues
```

‘Cars’-The Game.

Code (vjbox.h)

```
//vjbox.h Copyright 2002 Vaibhav Jain..All Rigths Reserved
#include <conio.h>
#include <stdio.h>
#define UP_ARROW 72
#define DN_ARROW 80
#define LF_ARROW 75
#define RT_ARROW 77
#define SPACE_BAR 32
#define ESCAPE_KEY 27
typedef struct {int x,y;} point;
int box(unsigned x1,unsigned y1,unsigned x2,unsigned y2)
{int i;gotoxy(x1,y1);for(i=x1;i<=x2;i++)putch(205);
gotoxy(x1,y2);for(i=x1;i<=x2;i++)putch(205);
for(i=y1;i<y2;i++){gotoxy(x1,i);putch(186);}
for(i=y1;i<y2;i++){gotoxy(x2,i);putch(186);}
gotoxy(x1,y1);putch(201);gotoxy(x1,y2);putch(200);
gotoxy(x2,y1);putch(187);gotoxy(x2,y2);putch(188);
return 1;}
void fill(int x1,int y1,int x2,int y2,char ch)
{int i,j;for(i=y1;(i<=y2)&&(i<=25);i++)
 if (i<1) continue;
 else for(j=x1;(j<=x2)&&(j<=79);j++)  { if (j<1) continue;     gotoxy(j,i);     putch(ch);  }
}

void msgbox(char mesag[])
{int nlines=0,size=0,i=0,j=0,temp[4];
int maxlen=0;
struct text_info ti;gettextinfo(&ti);
temp[0]=ti.curx;temp[1]=ti.cury;size=strlen(mesag);
temp[2]=ti.screenwidth/2;temp[3]=ti.screenheight/2;
for(i=0;i<=size;i++,j++)
if (mesag[i]=='\n'){nlines++;if(j>maxlen) maxlen=j;j=0;}
if(j>maxlen) maxlen=j;
fill(temp[2]-maxlen/2-2,temp[3]-nlines/2-1,temp[2]+maxlen/2+2,temp[3]+nlines/2+1,' ');
box(temp[2]-maxlen/2-2,temp[3]-nlines/2-1,temp[2]+maxlen/2+2,temp[3]+nlines/2+1);
window(temp[2]-maxlen/2,temp[3]-nlines/2,temp[2]+maxlen/2,temp[3]+nlines/2);
cprintf("%s",mesag);window(1, 1, 80, 25);
gotoxy(temp[0],temp[1]);
}

point midpoint(void)
{  struct text_info ti;point temp;
   gettextinfo(&ti);  temp.x=ti.screenwidth/2; temp.y=ti.screenheight/2;
   return temp;}
void wait(void )
{ int i='a';
while(!((i==13)||(i==27)))
if (kbhit()) i=getch();
}
void beep()
{ sound(1000);
delay(100);
nosound();}
//********************code concludes********************
```

Bouncing Ball..

Objective

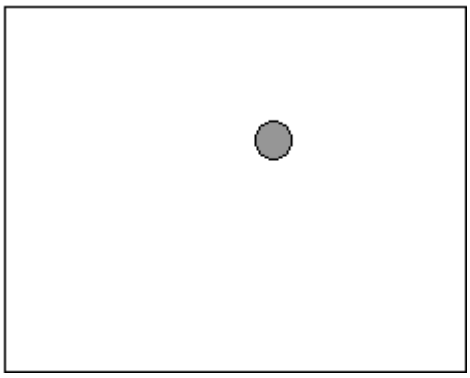
To Make a bouncing ball using TurboC Graphics Library. & using dynamic memory allocation.

Theory

Code (balls2.c)

```
//project
#include <vjgraph.h>
#include <alloc.h>
#define DELAY 5
#define XRADIUS 10
#define YRADIUS 10
#define INCER 3
point ball={20,20}; int dx=INCER,dy=INCER;
void main(void)
{ void *buff,tuk();int imgsize; char nkey='a';  init();
  setfillstyle(SOLID_FILL,getmaxcolor());
  fillellipse(ball.x,ball.y,XRADIUS,YRADIUS);
  imgsize=imagesize(ball.x-XRADIUS,ball.y-YRADIUS,
  ball.x+XRADIUS,ball.y+YRADIUS);
buff=malloc(imgsize);
if (buff==NULL){closegraph();printf("Error Allocating
Memory");exit(1);}
getimage(ball.x-XRADIUS,ball.y-
YRADIUS,ball.x+XRADIUS,ball.y+YRADIUS,buff);
  while(nkey!=27) { putimage(ball.x-XRADIUS,ball.y-
YRADIUS,buff,XOR_PUT);
if (kbhit()){ nkey=getch();
if(nkey==13) {dx*=-1;dy*=-1;}
if(nkey==UP_ARROW) dy=-INCER;
if(nkey==DN_ARROW) dy=+INCER;
if(nkey==RT_ARROW) dx=+INCER;
if(nkey==LF_ARROW) dx=-INCER;    }
if((ball.x>=maxx-XRADIUS)||(ball.x<=XRADIUS))
{dx*=-1;tuk();}
if((ball.y>=maxy-YRADIUS)||(ball.y<=YRADIUS))
{dy*=-1;tuk();} ball.x+=dx;ball.y+=dy;
 putimage(ball.x-XRADIUS,ball.y-
YRADIUS,buff,XOR_PUT); delay(DELAY); }
free(buff);closegraph();
}
void tuk(void)
{ int i; for(i=100;i<=200;i++,sound(i));
nosound();}
//code continues
```

Output



Bouncing Ball.

Code (vjgraph.h)

```
//project
//vjgraph.h Copyright 2002 Vaibhav Jain..All Rigths Reserved
#include <graphics.h>
#define UP_ARROW 72
#define DN_ARROW 80
#define LF_ARROW 75
#define RT_ARROW 77
#define SPACE_BAR 32
#define ESCAPE_KEY 27
int gdriver = DETECT, gmode,maxx,maxy;
typedef struct {int x,y;} point;
typedef struct {int left,top,right,bottom;} rect;
void init()
{int errorcode;
initgraph(&gdriver, &gmode, "");
errorcode = graphresult();
if (errorcode != grOk)  /* an error occurred */
{
   printf("Graphics error: %s\n", grapherrormsg(errorcode));
   printf("Press any key to halt:");
   getch();
   exit(1);            /* return with error code */
}
maxx=getmaxx();maxy=getmaxy();
}
//*********************code concludes*********************
```

N-Bouncing & Colliding Balls..

Objective

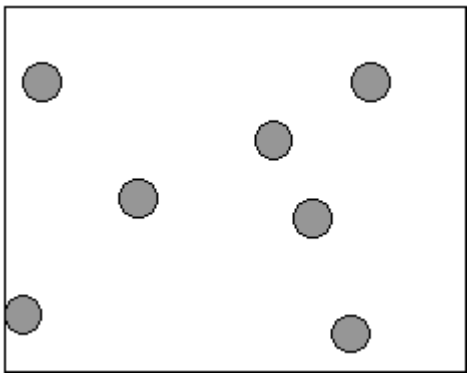
To stimulate N number of bouncing & colliding balls on screen using Graphics Library

Theory

Code (paras.c)

```
#define SIZE 20
#define DELAY 0
#include <graphics.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>
typedef struct
{ int x,y,dx,dy,radius;
} circles;
int sign(int x);
void main()
{void generate(circles cir[],int size);
void render(circles cir[],int size);
void collide(circles cir[],int size);
int gdriver = DETECT, gmode;
circles data[SIZE];
initgraph(&gdriver, &gmode, "");
cleardevice();generate(data,SIZE);
setcolor(0);
while (!kbhit()){delay(DELAY);
render(data,SIZE);collide(data,SIZE);}
closegraph();
} void collide(circles cir[],int size)
{int i,j;circles *temp;void swap(int*,int*), render(circles
cir[],int size);
unsigned long int a;
for (i=0;i<size;i++)
{temp=&cir[i];
for (j=0;j<size;j++)
{if (j==i) continue;
a=temp->x-cir[j].x;a*=a;            a+=(temp->y-
cir[j].y)*(temp->y-cir[j].y);if (a<=400)       {swap(&temp-
>dx,&cir[j].dx); swap(&temp->dy,&cir[j].dy);
 render(cir,size);} }
if(((temp->x)>=getmaxx())||((temp->x<=0)))
temp->dx=-1*temp->dx;

//code continues
```

Output



N-Bouncing & Colliding Balls.

Code (paras.c)

```
if(((temp->y)>=getmaxy())||((temp->y<=0)))
temp->dy=-1*temp->dy;
}

}
void render(circles cir[],int size)
{ int i;circles *temp;
  for(i=0;i<size;i++)
  {temp=&cir[i];
   setfillstyle(SOLID_FILL,0);
   fillellipse(temp->x,temp->y,temp->radius,temp->radius);
   temp->x+=temp->dx;temp->y+=temp->dy;
   setfillstyle(SOLID_FILL,getmaxcolor());
   fillellipse(temp->x,temp->y,temp->radius,temp->radius);
 }
}
void generate(circles cir[],int size)
{int j;srand(2);
for (j=0;j<size;j++)
{
 cir[j].x=random(getmaxx());
 cir[j].y=random(getmaxy());
 cir[j].radius=10;
 cir[j].dx=random(10);
 cir[j].dy=random(10);
}
} int sign(int x)
{int y;
if (x==0) return 1;
y=x/abs(x);
return y;
}
void swap(int * a,int *b)
{int temp;
temp=*a;
*a=*b;
*b=temp;
}

//********************code concludes********************
```

The Clock..

Objective

To stimulate a real world analog clock with hours, minutes and second hands, using Rotation Transformation.

Theory

Code (clock.c)

```
#include <stdlib.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>
#include <dos.h>
#define RAD M_PI/180
#define DELAY 1500
#define RSEC 100
#define RMIN 70
#define RHRS 50
char*
nums[]={"1","2","3","4","5","6","7","8","9","10","11","12"}
;
int midx,midy;
void synchronize(int *,int *,int*);
void drawclock(){
int i;float j;
for(i=20;i<=70;i++) circle(midx,midy,RSEC+i);
settextjustify(1, 1);
settextstyle(1,HORIZ_DIR,4);
for(i=1;i<=12;i++)
{j=(i*30-90)*RAD;
outtextxy(midx+RSEC*cos(j),midy+RSEC*sin(j),nums[i-
1]);
}
}
 void settime2( int hrs,int min, int sec)
{static float h,m,s;
setcolor(0);setlinestyle(1,1,0);
line(midx,midy,midx+RSEC*cos(s),midy+RSEC*sin(s));setl
inestyle(0,1,2);
line(midx,midy,midx+RMIN*cos(m),midy+RMIN*sin(m));s
etlinestyle(0,1,3);
line(midx,midy,midx+RHRS*cos(h),midy+RHRS*sin(h));
h= (float)(hrs+(float)min/60+(float)sec/3600)*30-90;
m= (float)(min+(float)sec/60)*6-90;
s= (float)sec*6-90;

//code continue
```

Output

The Clock.

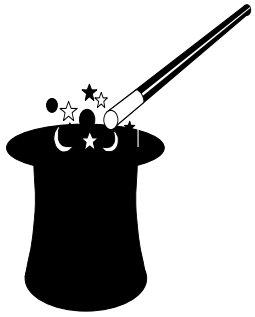
Code (clock.c)

```
h*=RAD;m*=RAD;s*=RAD;
setcolor(getmaxcolor());setlinestyle(1,1,0);
line(midx,midy,midx+RSEC*cos(s),midy+RSEC*sin(s));setlinestyle(0,1,2);
line(midx,midy,midx+RMIN*cos(m),midy+RMIN*sin(m));setlinestyle(0,1,3);
line(midx,midy,midx+RHRS*cos(h),midy+RHRS*sin(h));
}
void main()
{int gdriver = DETECT, gmode, errorcode,s=0,h=0,m=0;
initgraph(&gdriver, &gmode, "");
errorcode = graphresult();
if (errorcode != grOk)
{printf("Graphics error: %s\n", grapherrormsg(errorcode));
printf("Press any key to halt:");
getch();
exit(1); }
midx=getmaxx()/2;midy=getmaxy()/2;
drawclock();
synchronize(&h,&m,&s);
while(!kbhit())
{
s++;if(s>59) { s=0;m++;}
if(m>59) {m=0;h++;}
if(h>12) h=1;
settime2(h,m,s);
sound(50);
delay(10);
nosound();
delay(DELAY-20);}
getch();
closegraph();
restorecrtmode();
}
void synchronize(int * h,int *m,int *s)
{struct time t;
gettime(&t);
*s=t.ti_sec;
*h=t.ti_hour;
*m=t.ti_min;
}

//********************code concludes********************
```

Line Animations..

Objective

To animate lines like ‘Mystify Your Mind’ Screen Saver of windows.

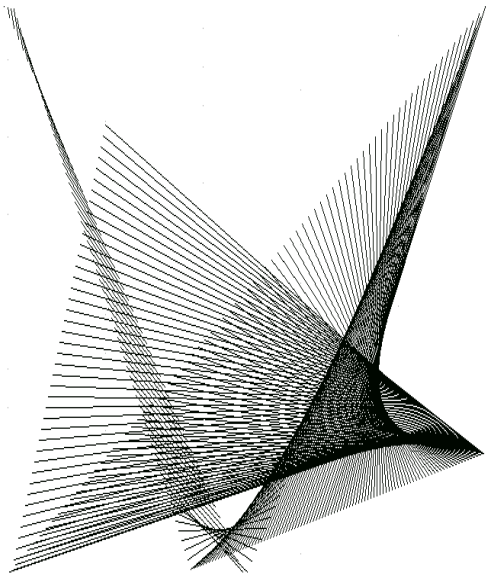


Theory

Code (nokia3.c)

```
#include <graphics.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>
#include <time.h>
#define RAND 10
#define DELAY 21
int main(void)
{ int gdriver = DETECT, gmode, errorcode;
  initgraph(&gdriver, &gmode, "");
  int maxx=getmaxx(),maxy=getmaxy(),r=3, extra;
  int
x1=0,y1=0,incerx1=1,incery1=1,xs1=0,ys1=0,col1=15,r
1=10,coco1=0;
  int
x2=0,y2=0,incerx2=1,incery2=1,xs2=0,ys2=0,col2=15,r
2=10;
   cleardevice();
   setlinestyle(0,2,1)  ;
   setfillstyle(1,15);
   settextstyle(0,0,1);
while(1)
   {  coco1=rand()%500;
      if(coco1%200==0)
      {cleardevice(); col1=rand()%10;
         if(col1==0) col1=15; }
      if(xs1==0) x1=x1+incerx1;
      if(ys1==0) y1=y1+incery1;
      if(xs1==1) x1=x1-incerx1;
      if(ys1==1) y1=y1-incery1;
      if(xs2==0) x2=x2+incerx2;
if(ys2==0) y2=y2+incery2;
      if(xs2==1) x2=x2-incerx2;
      if(ys2==1) y2=y2-incery2;
      setcolor(col1);
      line(x1, y1, x2, y2);

//code continue
```

Output



Line Animations.

Code (nokia3.c)

```
      delay(DELAY);
      if(x1<r1){incerx1=random(RAND);
       xs1=0;}
      if(x1>maxx-r1){incerx1=random(RAND);
       xs1=1;}
      if(y1<r1){ys1=0;
       incery1=random(RAND);}
      if(y1>maxy-r1){ys1=1;
       incery1=random(RAND);
        if(col1==0) col1=15;
      }
     if(x2<r2){incerx2=random(RAND);
       xs2=0;}
      if(x2>maxx-r2){incerx2=random(RAND);
       xs2=1;}
      if(y2<r2){ys2=0;
      incery2=random(RAND);}
      if(y2>maxy-r2){ys2=1;
      incery2=random(RAND);
       if(col2==0) col2=15;
     }

     if(kbhit())
     { if(getch()=='p')getch();
       else{ closegraph();
      restorecrtmode();exit(1);
      }
    }

 }
}

//********************code concludes********************
```

Graph Of A Function..

Objective

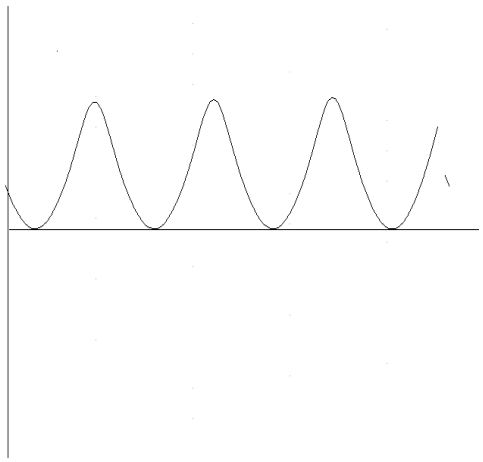
To draw the graph of a predefined function which defines y in terms of x. Example y=sin(x)*cos(x) etc.

Theory

Code (graph.cpp)

```
#include <complex.h>
#include <graphics.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#include <dos.h>

float cot(float x)
{float yy;
 yy=1/tan(x);
  return yy;
}
float sec(float x)
{float yy;
 yy=1/cos(x);
  return yy;
}
float cosec(float x)
{float yy;
 yy=1/sin(x);
 return yy;
}
int main(void)
{ int gdriver = DETECT, gmode, errorcode,dela=50;
  initgraph(&gdriver, &gmode, "");
  float x=-5,prey=getmaxy()/2;
  double y;
  float yy=0,si=20,magi=50;
while(1)
  {si=20;
  //cleardevice();
   line(20,getmaxy()/2,getmaxx(),getmaxy()/2);
   line(18,getmaxy(),18,0);
while(si<=600)
  {
    magi=magi+0.05;
    x=x+0.1;
//code continues
```

Output



Graph of a Function.

Code (graph.cpp)

```
while(si<=600)
  {
    magi=magi+0.05;
    x=x+0.1;
    y=tan(sin(x)*sin(x));
    y=y*magi;
    yy=(getmaxy()/2)-y;
    sound(abs(ceil(y*10)));
    line(si-5,prey,si,yy);
    prey=yy;
    si=si+5;
    if(kbhit())
    {
     switch(getch())
     {
     case 'p':
     nosound();
     getch();
     break;
     case 's':
     dela=dela+1;
     break;
      case 'a':
     if(dela<2)
     dela=2;
     dela=dela-1;
     break;
     case 'e':
     nosound();
     exit(1);
     break;
     case 'z':
      magi=magi+1;
      break;
      case 'x':
      magi=magi-1;
      break;
      case 'r':
      cleardevice();
      line(20,getmaxy()/2,getmaxx(),getmaxy()/2);
       line(18,getmaxy(),18,0);
       break;
     }
    }
    delay(dela);
    nosound();
   }
   }
}
//*********************code concludes*********************
```

Project Calculator

Objective

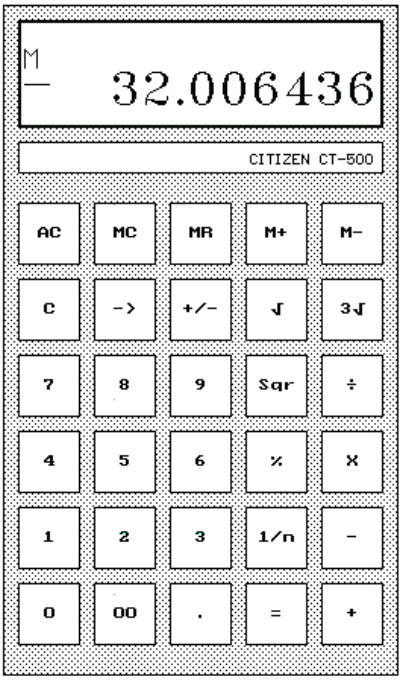
To stimulate a real world calculator model "**CITIZEN CT-500**" on the computer display and to stimulate all of its operations.

Theory

Code (calcu4.c)

```
//project
//THE REAL CALCULATOR
#include <calcu.h>
#include <ctype.h>
#define FILLCOL 4
char disptext[20]="0.",textpos=0;
registers r1={0,TRUE},r2={0,TRUE},mem={0,TRUE};
uint dec_pressed=FALSE;//is decimal pressed
uint reset_display=FALSE;
uint mem_err=FALSE;
uint err=FALSE;
uint sign=FALSE;
int preop=0;
int handlextended(int key);
double getdispnum(void);
void main()
{char keycode='a';int i=0,pretemp=0;//pretemp is for
percentage
                                        //calculations..
 double temp;
 init();initallobjects();drawcalc();
 //sign=TRUE;err=TRUE;
 //mem.cf=FALSE;
 displaynum(0);
 r1.value=getdispnum();
 //mem.value=12.002;
 //display(disptext);
 while(keycode!=ESCAPE_KEY)
 {
if(kbhit())
   { keycode=getch();
     for(i=0;i<32;i++) if (keycodes[i]==keycode) break;
     if (i>=32) continue;
     if (isdigit(keycode)||(keycode=='.'))
addnumtodisp(keycode);
switch(keycode)
     {case 0: i=handlextended(0);break;
//code continues
```

Output



Project Calculator

Code (calcu4.c)

```
    case  13:
    case '+':
    case '-':
    case '*':
    case '/':dops(keycode) ;break;
    case '%':pretemp=preop;
                if(preop) dops(13);
                if(pretemp==4) temp=getdispnum()/100;
                if(pretemp==5) temp=getdispnum()*100;
                displaynum(temp);
                break;
    case 8: delnumdisp();break;
    case 'c':strcpy(disptext,"0.");
                textpos=0;
                display(disptext);
                break;
    case's': temp=getdispnum();
                temp*=temp;
                displaynum(temp);break;
    case 'r':temp=getdispnum();
                temp=(double)1/temp;
                displaynum(temp);break;
    case '@':temp=getdispnum();
                temp=sqrt(temp);
                displaynum(temp);break;
    case '#':temp=getdispnum();
                temp=pow(temp,0.3333);
                displaynum(temp);break;
    case 'o':addnumtodisp('0');
                addnumtodisp('0');break;
    case 'm':displaynum(mem.value);
                reset_display=TRUE;break;
    case 63 :drawcalc();continue;
    case 'z':if (preop) dops(13);
                mem.value-=getdispnum();
                if (mem.value!=0) mem.cf=FALSE;
                else mem.cf=TRUE;
                display(disptext);
                reset_display=TRUE;break;
    case 32: if (preop) dops(13);
                mem.value+=getdispnum();
                if (mem.value!=0) mem.cf=FALSE;
                else mem.cf=TRUE;
                display(disptext);
                reset_display=TRUE;break;
    case 32: if (preop) dops(13);
                mem.value+=getdispnum();
                if (mem.value!=0) mem.cf=FALSE;
                else mem.cf=TRUE;
                display(disptext);
                reset_display=TRUE;break;
    }
//code continue
```

Project Calculator

Code (calcu4.c)

```
                    display(disptext);
                    reset_display=TRUE;break;
    }

    tuk();
    if (i==-1) continue;
    setfillstyle(SOLID_FILL,FILLCOL);
    floodfill(buttons[i].x+1,buttons[i].y+1,getmaxcolor());
    delay(100);
    setfillstyle(EMPTY_FILL,7);
    floodfill(buttons[i].x+1,buttons[i].y+1,getmaxcolor());
    }


 }
//closegraph();
}
int handlextended(int key)
{int i;
if (key==0) i=getch();
else i=key;
switch(i)
{case 67:sign=!sign;display(disptext);
            i=7;
            break;
 case 147: resetreg(&mem);
              display(disptext);
              i=1;
              break;
 case 'S':resetreg(&r1);
             resetreg(&r2);
             resetreg(&mem);
             dec_pressed=FALSE;//is decimal pressed
             reset_display=FALSE;
             mem_err=FALSE;
             err=FALSE;
             sign=FALSE;
             textpos=0;
             preop=0;
             strcpy(disptext,"0.");
             display(disptext);
             i=0;
             break;
  default:return -1;
}
return i;
}
 int addnumtodisp(char num)
{if (reset_display)
           {textpos=0;
dec_pressed=FALSE;

//code continues
```

Project Calculator

Code (calcu4.c)

```
            reset_display=FALSE;
            sign=FALSE;
            strcpy(disptext,"0.");
            //display(disptext);
            //return 1;
            }
if(num=='.') {dec_pressed=TRUE;return 1;}
if (textpos>=10) return 0;
if(dec_pressed) disptext[++textpos]=num;
else  {disptext[textpos]=num;
            disptext[++textpos]='.';
      }
disptext[textpos+1]='\0';
display(disptext);
return 1;
}

double getdispnum(void)
{ double factor,num=0;int len=0;
  while(disptext[++len]!='.');
  factor=pow10(len-1);
  for(len=0;disptext[len]!='\0';len++,factor/=10)
  if (disptext[len]=='.') {factor*=10;continue;}
  else num+=(disptext[len]-48)*factor;
  if(sign) num*=-1;
  return num;
}
int delnumdisp()
{//if (strlen(disptext)<2) return 0;
if (disptext[2]=='\0')
{strcpy(disptext,"0.");
 textpos=0;
 dec_pressed=FALSE;
 display(disptext);
 reset_display=TRUE;
 //display(disptext);
 //textpos=1;
 return 1;
}
if (!dec_pressed){disptext[textpos]='\0';
disptext[textpos-1]='.';}
if(dec_pressed){
            if (disptext[textpos]=='.')
            {dec_pressed=FALSE;
             disptext[textpos]='\0';
             disptext[textpos-1]='.';   }
else      disptext[textpos]='\0';
}textpos--;
display(disptext);
return 1;}
int dops(int keycode)
{ int op;
//code continue
```

Project Calculator

Code (calcu4.c)

```
int dops(int keycode)
{ int op;
  if((op=isoperator(keycode)) ==0) return 0;
  if (r1.cf)
  {r1.value=getdispnum();
                    r1.cf=0;
                    reset_display=TRUE;
                    preop=op;
                    return 1;}
  r2.value=getdispnum();r2.cf=0;
  switch(preop)
  {case 0:return 1;
   case 2:r1.value=r1.value+r2.value;
            resetreg(&r2);break;
   case 3:r1.value=r1.value-r2.value;
            resetreg(&r2);break;
   case 4:r1.value=r1.value*r2.value;
            resetreg(&r2);break;
   case 5:r1.value=r1.value/r2.value;
            resetreg(&r2);break;
  }
  displaynum(r1.value);
  reset_display=TRUE;
  if (op!=1) preop=op; else
  {preop=0;
  resetreg(&r1);
  }
  return 1;
}
```

Code (calcu.h)

```
#include <vjgraph.h>
#include <dos.h>
#include <string.h>
#include <math.h>
#define TRUE 1
#define FALSE 0
#define BUTTON 40
#define HGAP  10
#define VGAP 10
#define FONTSIZE 1
#define SCRFNTSIZE 4
#define STICKY  3000
typedef unsigned char uint;
char *caption="CITIZEN CT-500";
rect calculator,screen,logo;
int screenpoly[8];
extern uint sign,err,mem_err,reset_display;
extern char disptext[];
typedef struct {double value;uint cf;} registers;

//code continue
```

Project Calculator

Code (calcu.h)

```
extern registers mem;
const char *captions[30]={"AC","MC","MR","M+","M-","C","->","+/-","\xFB"
                          ,"3\xFB","7","8","9","Sqr","\xF6","4","5","6",
                          "%","X","1","2","3","1/n","-","0","00","."
                          ,"=","+"};
/*const int keycodes[32]={83,-109,109,-112,-114,99,8,32,64,35,55,56,57,
                          115,47,52,53,54,37,42,49,50,51,114,45,48,
                          111,46,13,43,27,0};*/
const int keycodes[32]={83,0,109,32,'z',99,8,0,64,35,55,56,57,
                          115,47,52,53,54,37,42,49,50,51,114,45,48,
                          111,46,13,43,27,0};

typedef struct{int x,y;char *caption;}button;
button buttons[31];
void drawcalc()
{int i;
 cleardevice();
 rectangle(calculator.left,calculator.top,calculator.right,calculator.bottom);
 rectangle(screen.left,screen.top,screen.right,screen.bottom);
 rectangle(logo.left,logo.top,logo.right,logo.bottom);
 settextjustify(CENTER_TEXT,CENTER_TEXT);
 settextstyle(0,0,FONTSIZE);
 for(i=0;i<30;i++)
 {rectangle(buttons[i].x,buttons[i].y,buttons[i].x+BUTTON,buttons[i].y+BUTTON);
 outtextxy(buttons[i].x+BUTTON/2,buttons[i].y+BUTTON/2,buttons[i].caption);
 }
 settextjustify(RIGHT_TEXT,CENTER_TEXT);
 settextstyle(2,0,0);
 outtextxy(logo.right-HGAP/2,logo.top+BUTTON/4,caption);
 setfillstyle(CLOSE_DOT_FILL,15);
 floodfill(calculator.left+1,calculator.top+1,getmaxcolor());

} void initallobjects()
{void initbuttons(),capsoff();
 int cheight,cwidth;
 capsoff();
 cheight=7*(BUTTON+VGAP)+2*BUTTON+VGAP;
 cwidth=5*(BUTTON+HGAP)+HGAP;
 calculator.left=(maxx-cwidth)/2;
 calculator.top=(maxy-cheight)/2;
 calculator.right=(maxx+cwidth)/2;
 calculator.bottom=(maxy+cheight)/2;
 screen.left=calculator.left+HGAP;
 screen.top=calculator.top+VGAP;
 screen.right=calculator.right-HGAP;
 screen.bottom=calculator.top+2*BUTTON;
 logo.left=screen.left;
 logo.right=screen.right;
 logo.top=screen.bottom+VGAP;
 logo.bottom=logo.top+BUTTON/2;
 settextjustify(CENTER_TEXT,CENTER_TEXT);

//code continue
```

Project Calculator

Code (calcu.h)

```
settextstyle(0,0,FONTSIZE);
 setcolor(getmaxcolor());
 screenpoly[0]=screen.left+1; screenpoly[1]=screen.top+1;
 screenpoly[2]=screen.right-1; screenpoly[3]=screen.top+1;
 screenpoly[4]=screen.right-1; screenpoly[5]=screen.bottom-1;
 screenpoly[6]=screen.left+1; screenpoly[7]=screen.bottom-1;
 initbuttons();
}
void capsoff()
{ unsigned char far *kb;
kb=(char *)0x417;
*kb=32;
}
void initbuttons()
{int i;
   buttons[0].caption=(char *)captions[0];
   buttons[0].y=screen.bottom+VGAP+BUTTON;
   buttons[0].x=calculator.left+HGAP;
   for(i=1;i<30;i++)
    {if (i%5==0)
    {buttons[i].y=buttons[i-1].y+BUTTON+VGAP;
     buttons[i].x=buttons[0].x;
    }
    else
    {buttons[i].x=buttons[i-1].x+BUTTON+HGAP;
     buttons[i].y=buttons[i-1].y;
    }
    buttons[i].caption=(char *)captions[i];
    }
buttons[30].x=-100;
buttons[30].y=-100;
}
void tuk(void)
{ int i; for(i=100;i<=200;i+=i,sound(i));
nosound();
}
void beep(void)
{ sound(2000);
 delay(200);
 nosound();
}
int isoperator(int num)
{switch(num)
{ case  13:return 1;
  case '+':return 2;
  case '-':return 3;
case '*':return 4;
  case '/':return 5;
  //case '%':return 6;
  default :return 0;
}}
//code continues
```

Project Calculator

Code (calcu.h)

```
int display(char * text)
{char texttodisp[20];
 strcpy(texttodisp,text);
 if (strlen(text)>11) strncpy(text,texttodisp,11);
 //texttodisp[10]='.';
 texttodisp[11]='\0';
 setfillstyle(SOLID_FILL,0);
 fillpoly(4,screenpoly);
 settextstyle(1,0,SCRFNTSIZE);
 settextjustify(RIGHT_TEXT,CENTER_TEXT);
 outtextxy(screen.right,screen.top+BUTTON,texttodisp);
 settextstyle(SMALL_FONT,0,10);
 if (sign) outtextxy(screen.left+textwidth("-")+4,
                                 screen.top+BUTTON*0.90,"-");
 settextstyle(SMALL_FONT,0,7);
 if (!mem.cf) outtextxy(screen.left+textwidth("M")+4,
                        screen.top+BUTTON-textheight("M"),"M");
 if (err) outtextxy(screen.left+textwidth("E")+4,screen.bottom
              -textheight("E"),"E");
 return 1;
}
int displaynum(double num)
{char buff[22];int i;
if (num<0) {sign=TRUE;num*=-1;}
if ((num>9999999999)||(num<-9999999999)) err=TRUE;
sprintf(buff,"%10.10f",num);
for(i=0;i<11;i++) disptext[i]=buff[i];
disptext[i]='\0';
i=strlen(disptext)-1;
while(disptext[i]=='0')
{disptext[i]='\0';i--;}
display(disptext);
reset_display=TRUE;
return 1;
}
void resetreg(registers *reg)
{ reg->value=0;
  reg->cf=TRUE;
}

//********************code concludes********************
//bye bye my sweet little file… good bye.
```