Abstract

Modern Technology brought the use of pictures in various fields. Also a simple picture speaks thousand words. So applications are required to create images and save them. An attempt has been made to do the same using simple algorithms.

The objective of the project is to create an application that can draw simple images. As C is the simple language for coding, C language has been choosen for the purpose. The Paint using ‘C’ application is created to easily draw images using several tool options. They include options like drawing circles, lines, rectangles, ellipses, polygon and solid fill areas in 2D. However, complex 3D images are not considered for the project.

The images created using this application can be saved. We can further retrieve the images and view them. However, the options for editing of images like cropping are not provided. We are able to save the images into raw file format. So the images can be viewed only in this application.

This application is compatible with Windows Operating System and C compiler.

ACKNOWLEDGEMENTS

The success accomplished in our project would not have been possible but for the timely help and guidance of many people. We wish to express our sincere and heartfelt gratitude to all those who have helped us in one-way or the other for the successful completion of this mini project.

We take this opportunity to express our sincere gratitude and indebtedness to the project guide for giving us timely suggestions and directing us towards perfection, for their guidance, advice and constant encouragement, without which it would have been difficult for us to complete this mini project work successfully.

We express our thanks to **K. Sreekanth**, Division Head, Department of Information Science and Technology, for kindly permitting us to carryout this work. We thank **Prof. K Rajasekhara Rao**, Dean, School of Computing, **Dr. L .S. S. Reddy,** Principal for the motivation and support provided.

Finally we thank one and all for extending their support in successful completion of our mini project work.

Santosh Kumr T (Y6IT297)
Sreekanth G (Y6IT301)
Suresh Kumar B (Y6IT315)

INDEX

1. Introduction

‘C’ paint is just similar to MS paint. It allows drawing pictures and saving them. The structure of ‘c’ paint is divided into menu bar, tool bar, color tool bar, status bar and draw pad to draw pictures. ’C’ graphics provides 16 number of colors, so that only 16 colors are available to use. A few no. of tool bars (rectangle tool bar, line tool, circle tool, etc..,) are provided by ‘C’ paint. Status bar displays help messages while using ‘C’ paint. Draw pad facilitates the user to draw the pictures inside draw pad window. The algorithms, coding and implementation of ‘C’ paint we discussed further. The execution of the programs may work a bit slowly.

The coding is done in such a way that it can be improved and implemented with more applications in an easy way.

2. Initializing Graphics

Initially ‘c’ language is in text mode, it should be changed to graphics mode. Hence we must first change over to graphics mode. In our program the function initgraph() is responsible for switching the mode from text to graphics .DETECT is a macro defined in 'graphics.h'. It requests initgraph() to automatically determine which graphics driver to load in order to switch to the highest resolution graphics mode. The initgraph() function takes three parameters, the graphics driver, the graphics mode and the path to the driver file.

Once the driver has been loaded, initgraph() sets up the numeric values of the graphics mode chosen in the variables gd and gm respectively. Here we are assuming that the driver files are in the directory 'c:\\turboc30\\bgi'. Hence the path passed to initgraph() is 'c:\\turboc30\\bgi'.

3. Structure of Paint using ‘C’

The basic structure of ‘c’ paint is shown below. The screen or the window is divided into 5 areas:

1. Menu bar
2. Tool bar
3. Color tool bar
4. Status bar
5. Draw pad

The following gives a overview of the structure.

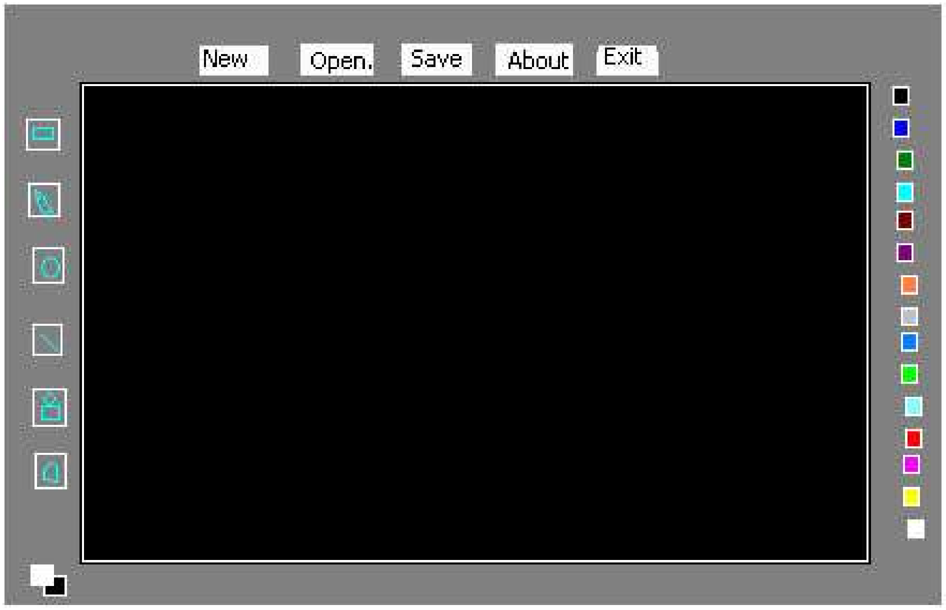
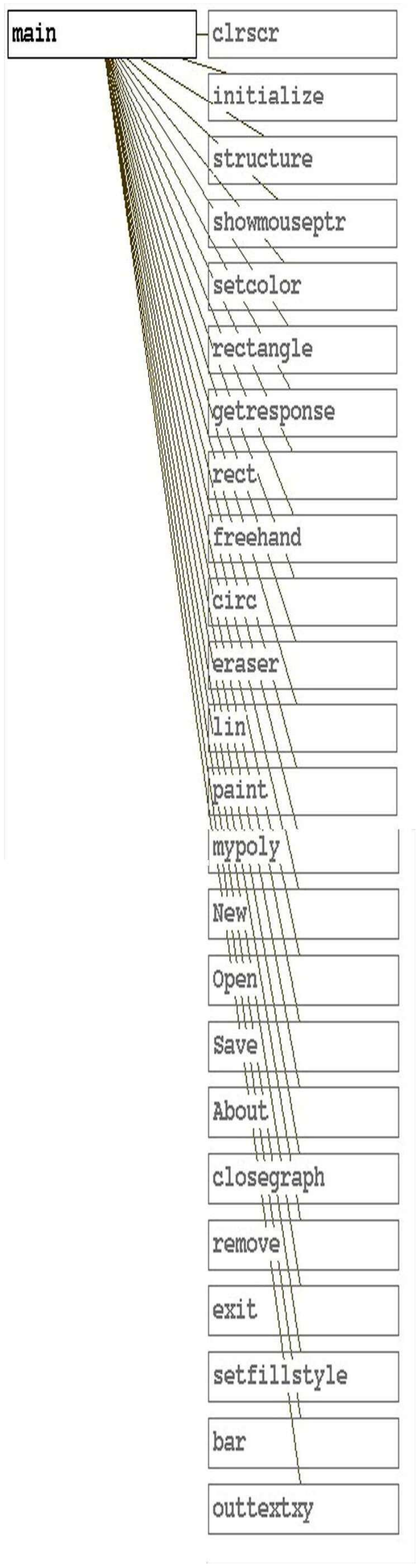


Fig: Basic Structure of the 'C' Paint application

Various pre-defined functions used to design the structure are:

1. Setbkcolor() is used to change back ground color.
2. using outtextxy() new, open, save, about, exit options are displayed.
3. Using bar() and setfillstyle() function draw pad is drawn.
4. Using rectangle() function 6 squares are drawn with same size at fixed x position and varying y position.
5. Using rectangle() function 15 squares boxes are drawn at right side of the screen at fixed x position and varying y position.
6. A special symbol is placed at the bottom to indicate current drawing color and current back ground color and there are drawn with bar() functions.
7. Status bar placed at the bottom is drawn using bar() function .
8. In the status bar using outtextxy() x and y positions are displayed.
9. All the menu bar options are placed inside a rectangle window using rectangle() function.

The flow chart the main function which gives the entire structure of the program is show here.


main
clrscr
initialize
structure
showmouseptr
setcolor
rectangle
getresponse
rect
freehand
circ
eraser
lin
paint
mypoly
New
Open
Save
About
closegraph
remove
exit
setfillstyle
bar
outtextxy

4. Algorthim

4.1. Menubar

Menu bar provides the users with open and save options for the pictures. It consists of five options.

1. New
2. Open
3. Save
4. Credits
5. Exit

Procedure to open a new file:

1. Move mouse pointer to new option.
2. Use left click of mouse to select it.
3. Using new(), user defined function a new file is opened.

4.1.1. New: clear the draw pad area using bar() function.

Procedure to open an existing file

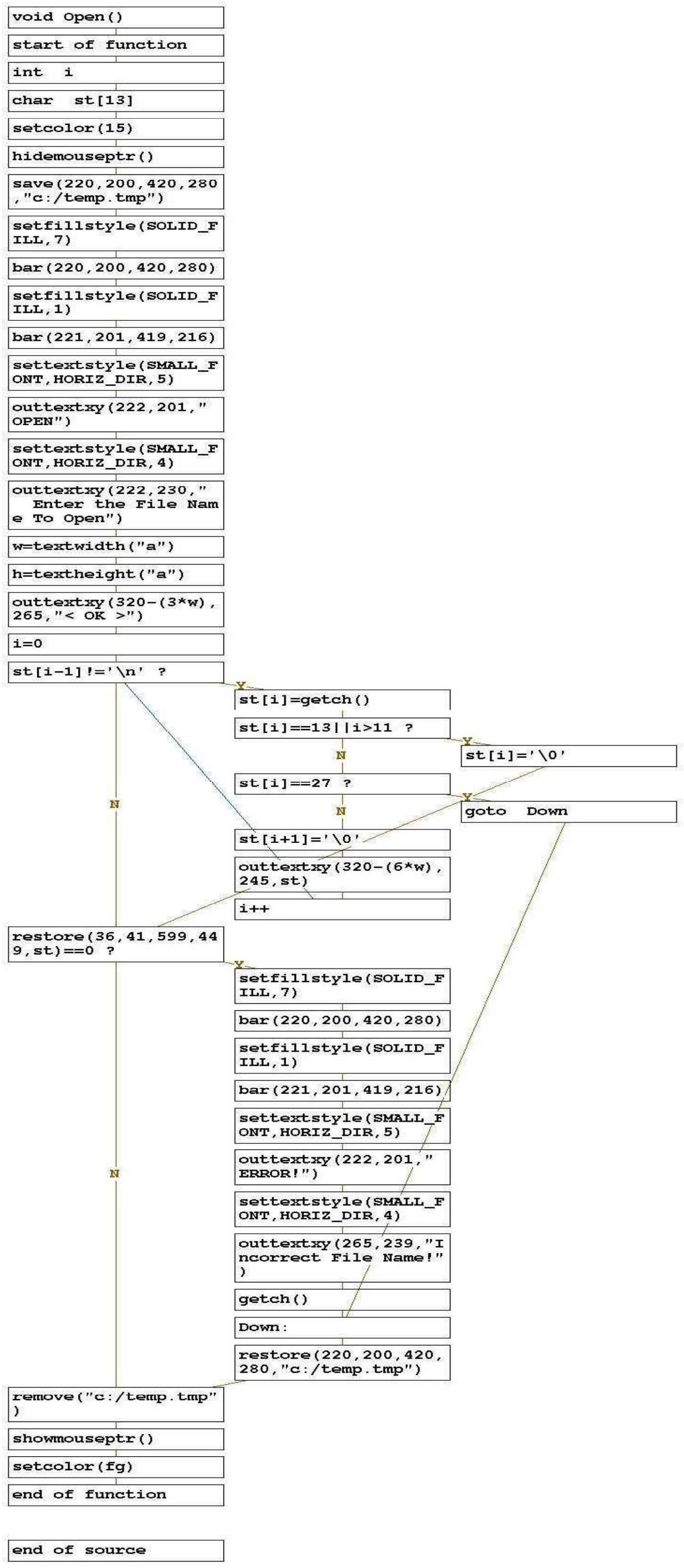
1. Move mouse pointer to open option.
2. Use left click of mouse to select it.
3. Using open(), user defined function existing file is opened.

4.1.2. **Open**():

1. Using fopen() function open file given by the user.
2. Copy the data into a temp file and save it temporarily.
3. Plot each pixel using putpixel() function by reading data from temp file.
4. Close the temp file
5. If file doesn't exists display an error message (incorrect file name).

Procedure to save the picture:

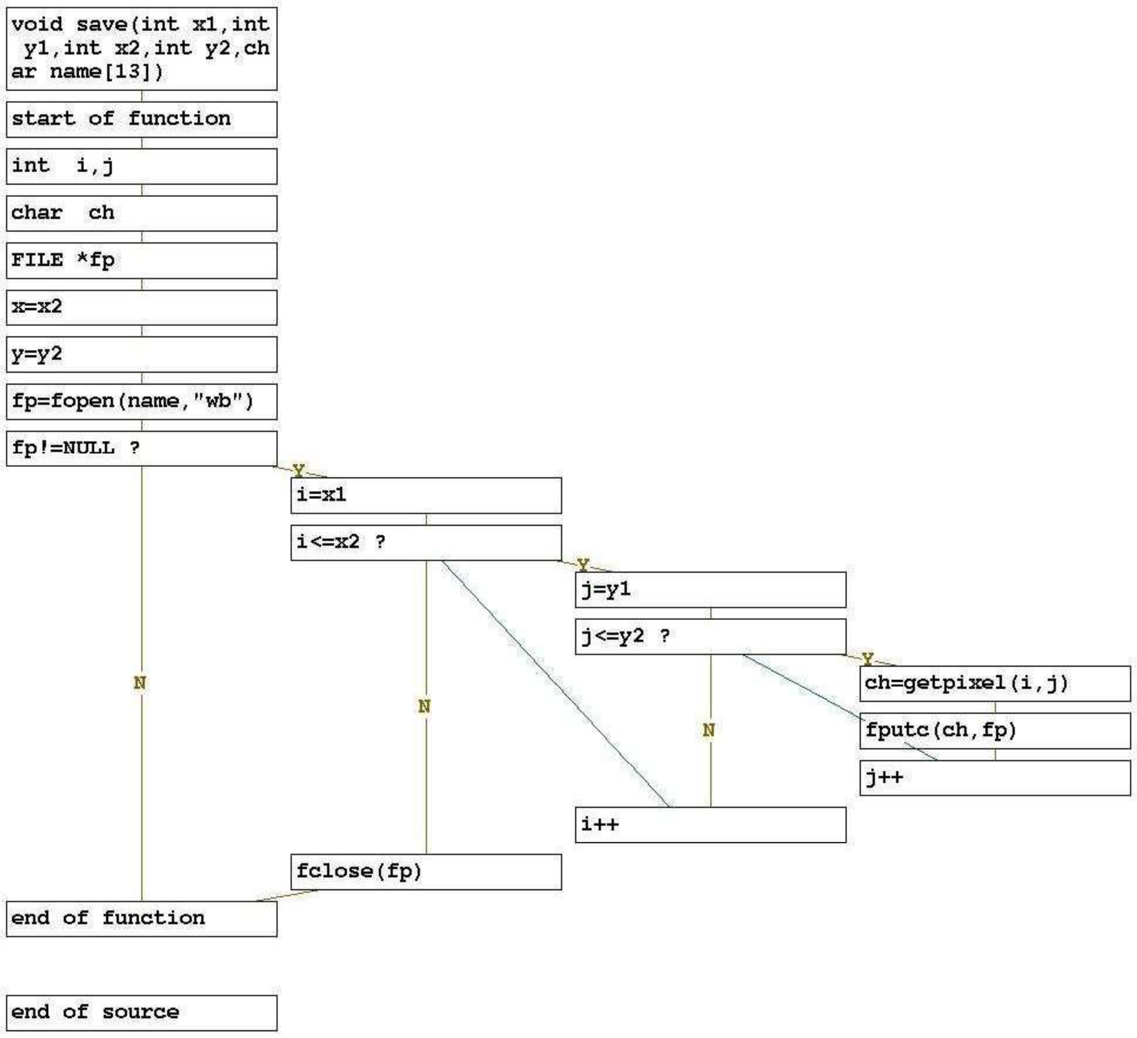
1. Move mouse pointer to save option.
2. Use left click of mouse to select it.
3. Using save(), user defined function the picture is saved with given file name.



```
void Open()
start of function
int  i
char  st[13]
setcolor(15)
hidemouseptr()
save(220,200,420,280
,"c:/temp.tmp")
setfillstyle(SOLID_F
ILL,7)
bar(220,200,420,280)
setfillstyle(SOLID_F
ILL,1)
bar(221,201,419,216)
settextstyle(SMALL_F
ONT,HORIZ_DIR,5)
outtextxy(222,201,"
OPEN")
settextstyle(SMALL_F
ONT,HORIZ_DIR,4)
outtextxy(222,230,"
  Enter the File Nam
e To Open")
w=textwidth("a")
h=textheight("a")
outtextxy(320-(3*w),
265,"< OK >")
i=0
st[i-1]!='\n' ?
    st[i]=getch()
    st[i]==13||i>11 ?
        st[i]='\0'
    st[i]==27 ?
        goto  Down
    st[i+1]='\0'
    outtextxy(320-(6*w),
    245,st)
    i++
restore(36,41,599,44
9,st)==0 ?
    setfillstyle(SOLID_F
    ILL,7)
    bar(220,200,420,280)
    setfillstyle(SOLID_F
    ILL,1)
    bar(221,201,419,216)
    settextstyle(SMALL_F
    ONT,HORIZ_DIR,5)
    outtextxy(222,201,"
    ERROR!")
    settextstyle(SMALL_F
    ONT,HORIZ_DIR,4)
    outtextxy(265,239,"I
    ncorrect File Name!"
    )
    getch()
    Down:
    restore(220,200,420,
    280,"c:/temp.tmp")
remove("c:/temp.tmp"
)
showmouseptr()
setcolor(fg)
end of function

end of source
```

4.1.3. Save():

1. Using fopen() function open a new file with given name in write mode.
2. Get each pixel using getpixel() function and write it into the file.
3. After writing the data close the file using fclose() function.



4.1.4. About():

Gives the information about the creators and developers of the program.

4.1.5. Exit():

Exits from ‘c’ paint.

4.2. Toolbar

Tool bar makes the users works or drawing pictures easier. The various tool bars provided are:

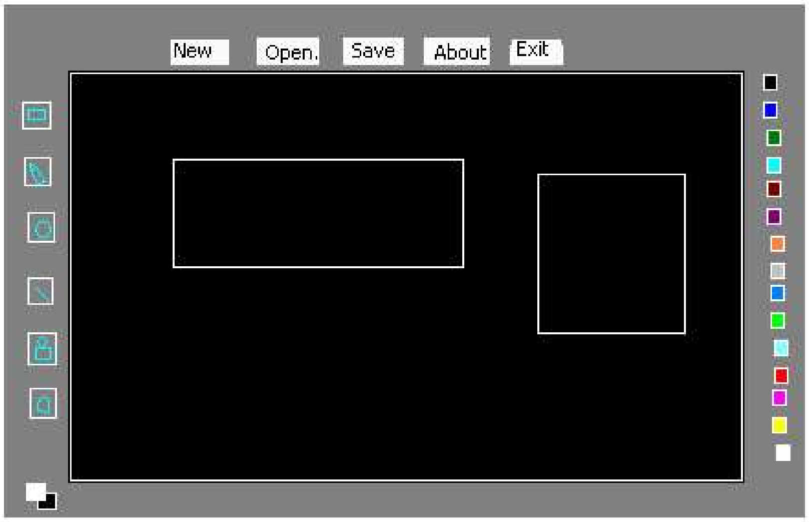
1. Rectangle tool
2. Pencil tool
3. Circle tool
4. Line tool
5. Fill area tool
6. Polygon tool

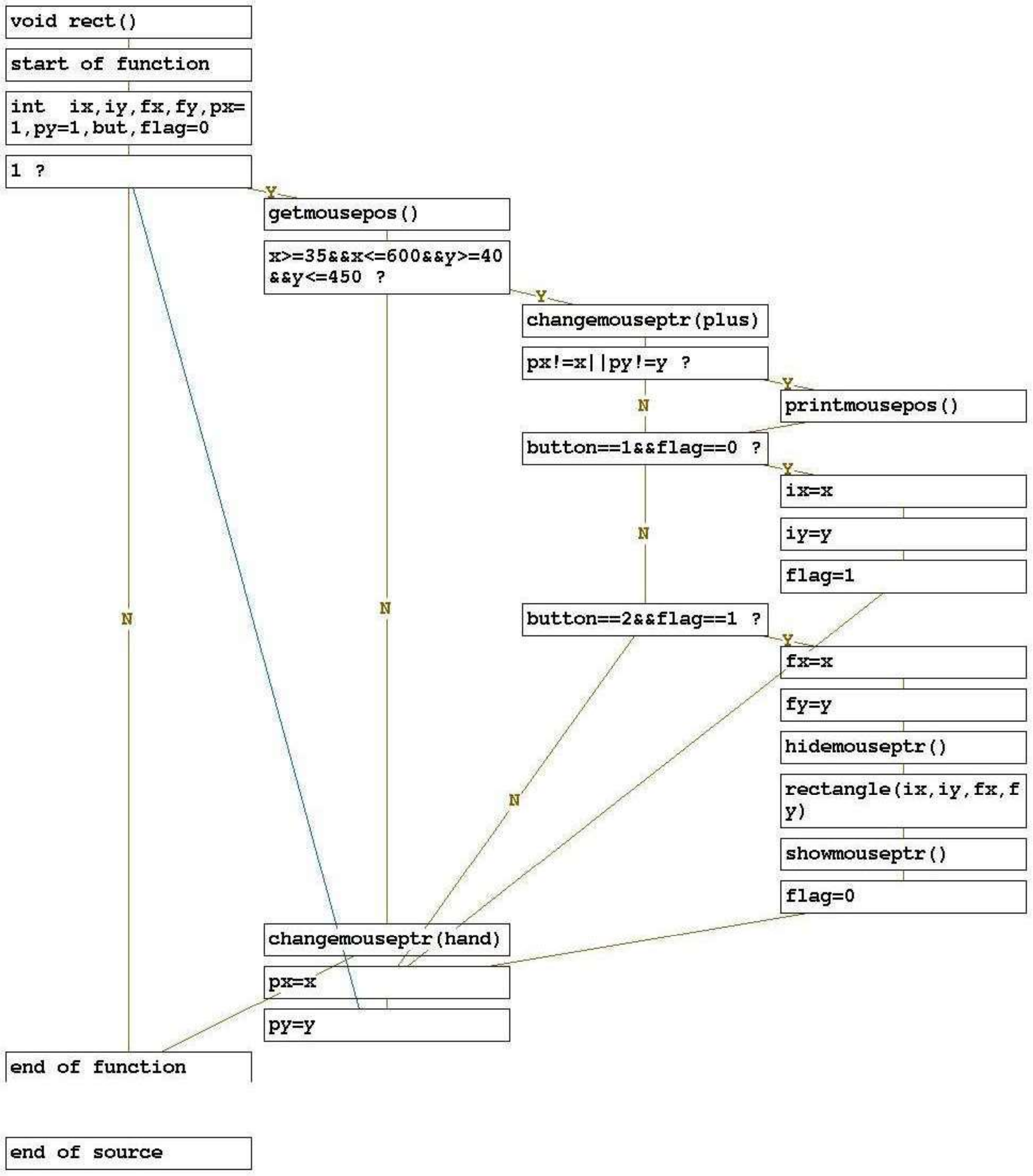
4.2.1. Procedure to draw rectangle:

1. Move mouse pointer to rectangle tool.
2. Use left click of mouse to select it.
3. Using rect(), user defined function a rectangle is drawn.

```
void rect()
{
      int ix,iy,fx,fy,px=1,py=1,but,flag=0,tx,ty,t=0;
        while(1)
        {
        getmousepos();
            if(x>=35&&x<=600&&y>=40&&y<=450)
            {
              if(button==1 && t==0)
              {
                tx=x;ty=y;
                t=1;
              }
            changemouseptr(plus);
                   if(px!=x||py!=y)
                   printmousepos();
                   if(button==1&&flag==0)
                   {
                   ix=x;iy=y;
                   flag=1;
                   }
                   else if(button==1 && flag==1 )
                   {
                   fx=x;fy=y;
                   hidemouseptr();
                   setcolor(fg);
                   rectangle(tx,ty,fx,fy);
                   setcolor(bg);
                   rectangle(tx,ty,fx,fy);
                   showmouseptr();
                   flag=0;
                   }
                   if(button!=1)
                   {
                     setcolor(fg);
                     rectangle(tx,ty,fx,fy);
```

```
                    t=0;
                  }
            }

            else
            {
              if(t==1)
              {
              setcolor(fg);
        rectangle(tx,ty,fx,fy);
              }
              t==0;
              changemouseptr(hand);
              break;
            }
                px=x;
                py=y;
        }
}
```



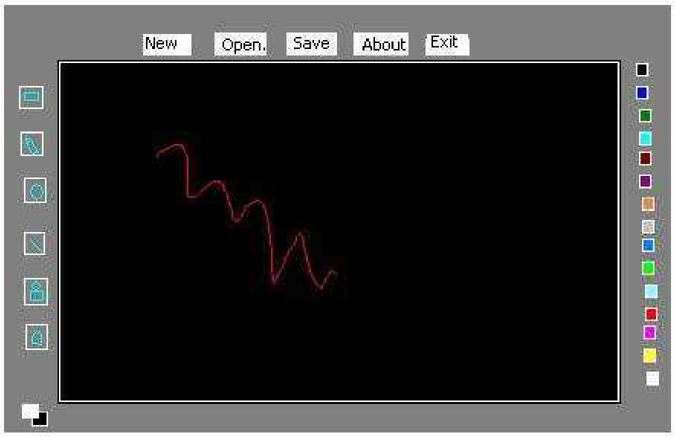


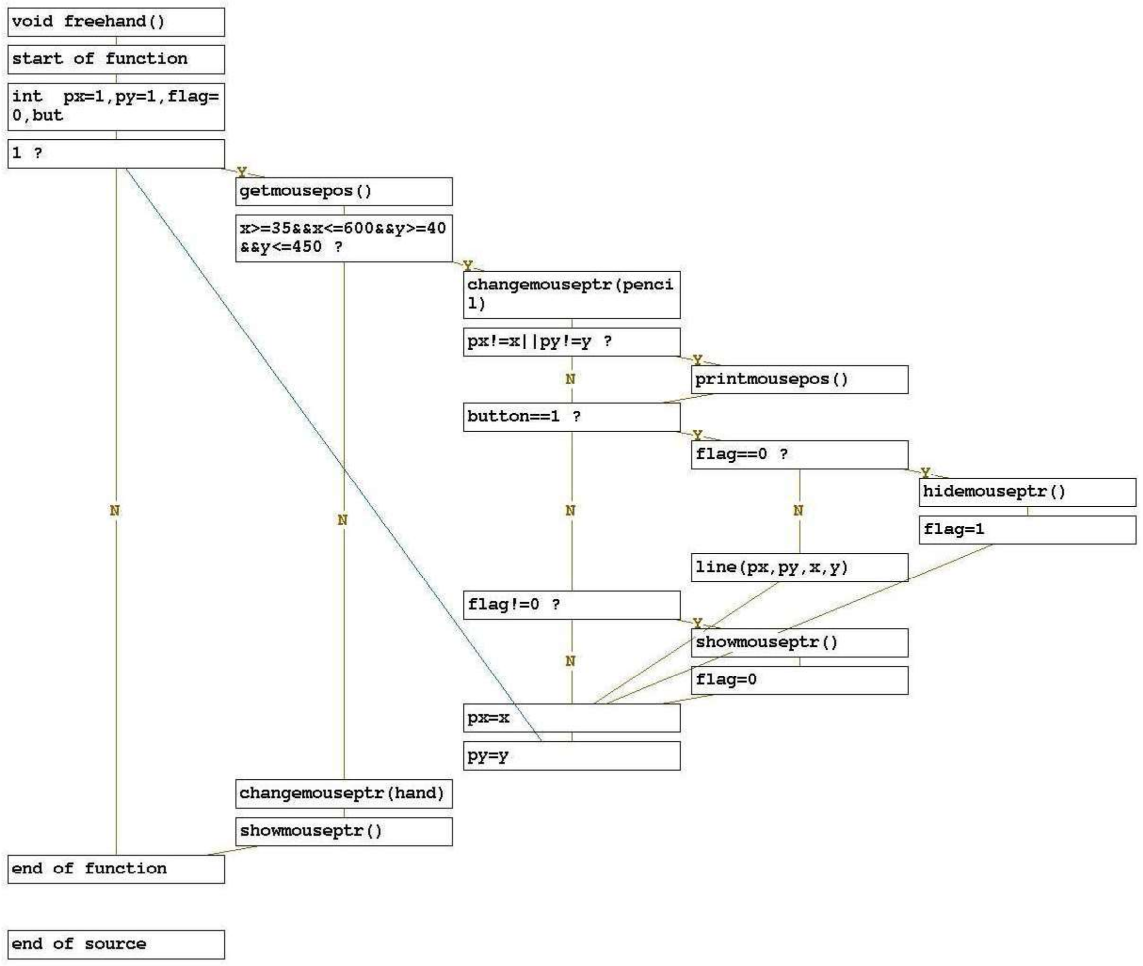
4.2.2. Pencil Tool

It is used for drawing images with free hand.

```
void freehand()
{
   int px=1,py=1,flag=0,but;
```

```
    while(1)
    {
    getmousepos();
    if(x>=35&&x<=600&&y>=40&&y<=450)
    {
    changemouseptr(pencil);
            if(px!=x||py!=y)
            printmousepos();

             if(button==1)
             {
                 if(flag==0)
                 {
                        hidemouseptr();
                        flag=1;
                 }
                 else
                        line(px,py,x,y);
              }
              else if(flag!=0)
              {

                 showmouseptr();
                 flag=0;

              }
              px=x;
              py=y;
           }
           else
           {
                 changemouseptr(hand);
                 showmouseptr();
                 break;
           }
    }
}
```



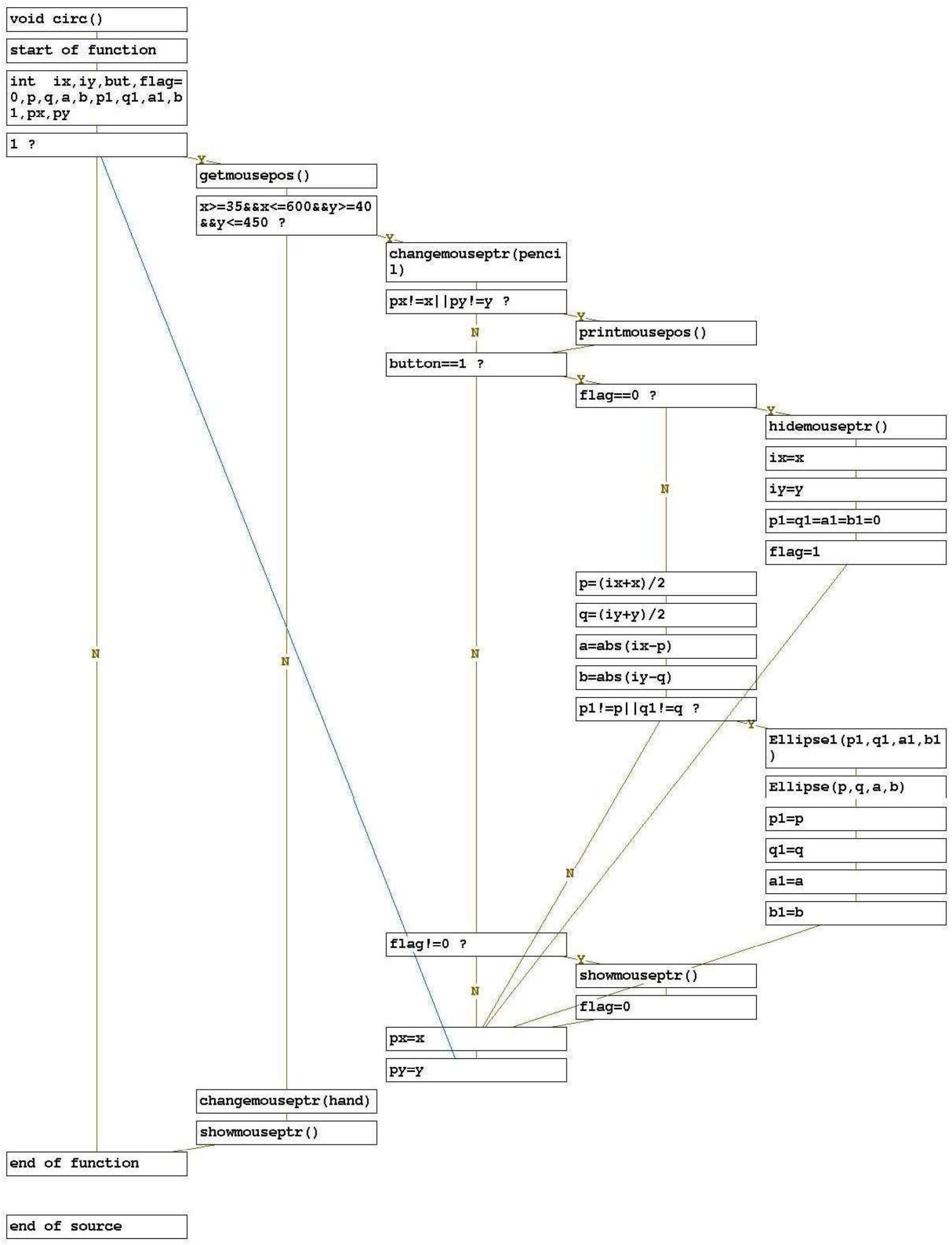


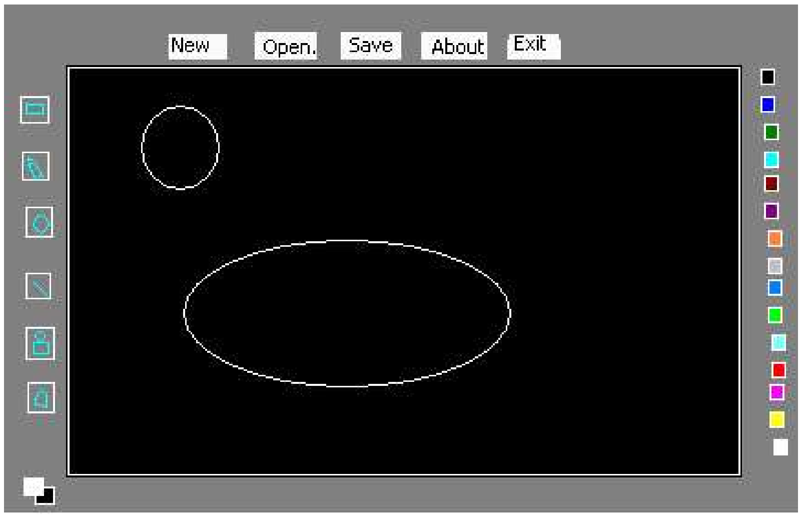
4.2.3. Circle

We can draw either circle or ellipse or circle using this tool.

The following code describes the procedure for a circle function.

```
void circ()
{
   int ix,iy,but,flag=0,p,q,a,b,p1,q1,a1,b1,px,py;

   while(1)
   {
         getmousepos();
         if(x>=35&&x<=600&&y>=40&&y<=450)
         {
                changemouseptr(pencil);
			if(px!=x||py!=y)
			{
				printmousepos();
			}
			if(button==1)
			{
				if(flag==0)
				{
					hidemouseptr();
					ix=x;iy=y;
					p1=q1=a1=b1=0;
					flag=1;
				}
				else
				{
					p=(ix+x)/2;
					q=(iy+y)/2;
					a=abs(ix-p);
					b=abs(iy-q);

					if(p1!=p||q1!=q)
					{
						Ellipse1(p1,q1,a1,b1);
						Ellipse(p,q,a,b);
						p1=p;q1=q;a1=a;b1=b;
					}
				}
			}
			else if(flag!=0)
			{
				showmouseptr();
				flag=0;
			}

			px=x;
			py=y;
		}
		else
		{
			changemouseptr(hand);
			showmouseptr();
				break;

		}

	}

}
```


void circ()
start of function
int ix,iy,but,flag=0,p,q,a,b,p1,q1,a1,b1,px,py
1 ?
Y
getmousepos()
x>=35&&x<=600&&y>=40&&y<=450 ?
Y
changemouseptr(pencil)
px!=x||py!=y ?
Y
printmousepos()
N
button==1 ?
Y
flag==0 ?
Y
hidemouseptr()
ix=x
iy=y
p1=q1=a1=b1=0
flag=1
N
p=(ix+x)/2
q=(iy+y)/2
a=abs(ix-p)
b=abs(iy-q)
p1!=p||q1!=q ?
Y
Ellipse1(p1,q1,a1,b1)
Ellipse(p,q,a,b)
p1=p
q1=q
a1=a
b1=b
N
N
N
N
flag!=0 ?
Y
showmouseptr()
flag=0
N
px=x
py=y
changemouseptr(hand)
showmouseptr()
end of function
end of source

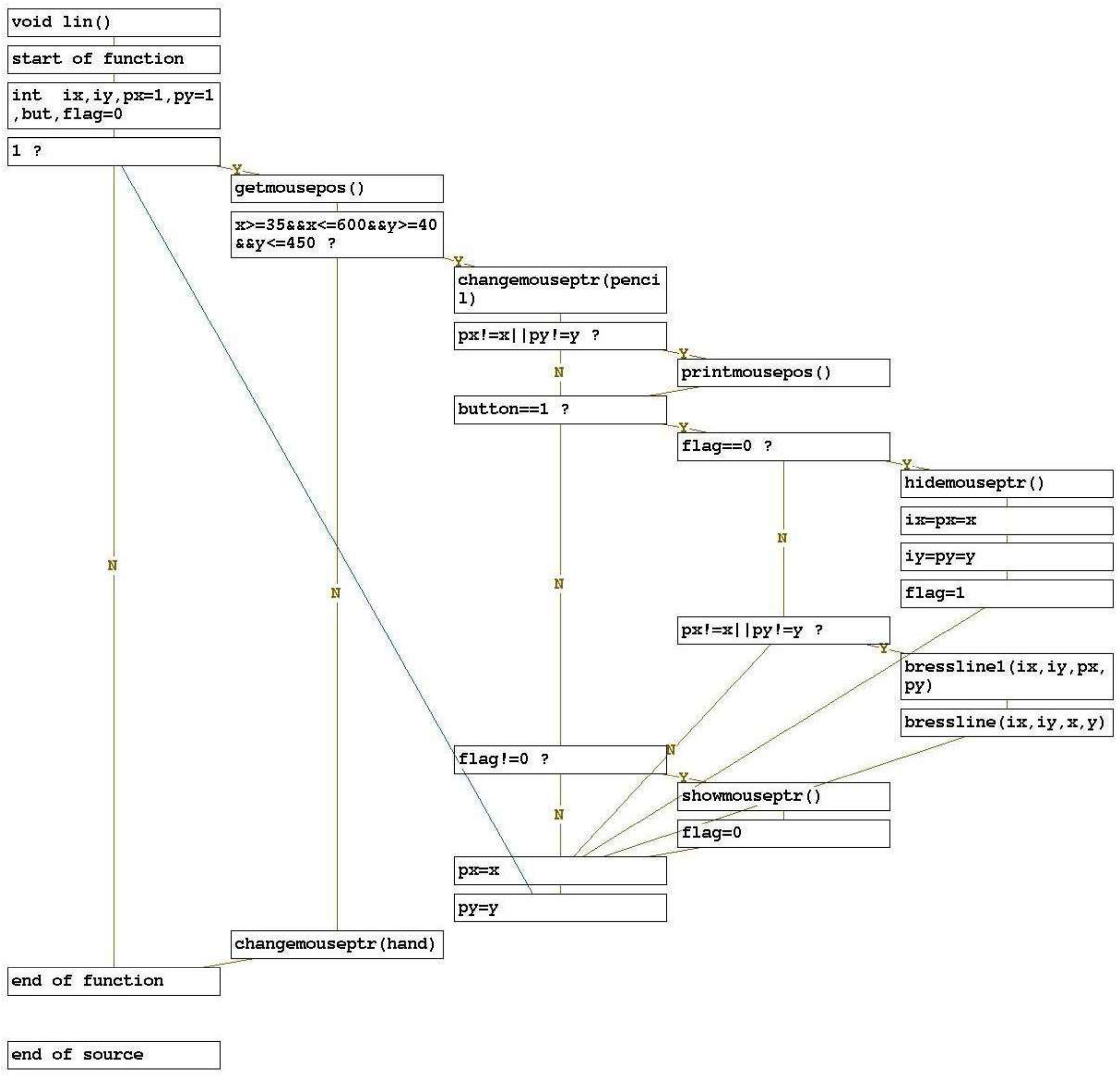


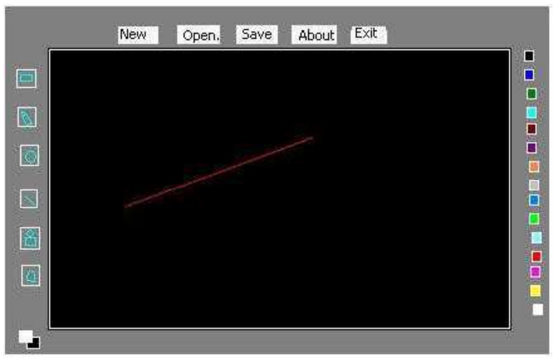
4.2.4. Line Tool

Using the bresenhams algorithm for drawing lines, we can draw a line.

```
void bressline ( int x1, int y1, int x2, int y2 )
{
   int incdec, t, i ;

   if ( x1 > x2 )
   {
         t = x1 ; x1 = x2 ; x2 = t ;
         t = y1 ; y1 = y2 ; y2 = t ;
   }

   dx = x2 - x1 ; dy = y2 - y1 ;

   if ( dx == 0 ) /* vertical line */
   {
         if ( y1 > y2 )
         {
               t = y1 ; y1 = y2 ; y2 = t ;
         }

         for ( i = y1,j=0 ; i <= y2 ; i++,j++ )
         {
               buffer[j]=getpixel (x1,i);
               putpixel ( x1, i,fg) ;
         }
         return ;
   }

   if ( dy == 0 )  /* horizontal line */
   {
         for ( i = x1,j=0 ; i < x2 ; i++ ,j++)
         {
               buffer[j]=getpixel (i,y1);
```

```
            putpixel ( i, y1, fg ) ;
      }
      return ;
}
 /* 0 < m < 1 */
if ( dy < dx && dy > 0 )
{
      e_noinc = 2 * dy ;
      e = 2 * dy - dx ;
      e_inc = 2 * ( dy - dx ) ;
      drawline ( x1, y1, x2, y2, PREDX, INCR ) ;
}
/* m = 1 */
if ( dy == dx && dy > 0 )
{
      e_noinc = 2 * dy ;
      e = 2 * dy - dx ;
      e_inc = 2 * ( dy - dx ) ;
      drawline ( x1, y1, x2, y2, PREDX, INCR ) ;
}
/* 1 < m < infinity */
if ( dy > dx && dy > 0 )
{
      e_noinc = 2 * dx ;
      e = 2 * dx - dy ;
      e_inc = 2 * ( dx - dy ) ;
      drawline ( x1, y1, x2, y2, PREDY, INCR ) ;
}
/* 0 > m > -1 */
if ( -dy < dx && dy < 0 )
{
      dy = -dy ;
      e_noinc = 2 * dy ;
      e = 2 * dy - dx ;
      e_inc = 2 * ( dy - dx ) ;
      drawline ( x1, y1, x2, y2, PREDX, DECR ) ;
}
/* m = -1 */
if ( dy == -dx && dy < 0 )
{
      dy = -dy ;
      e_noinc = ( 2 * dy ) ;
      e = 2 * dy - dx ;
      e_inc = 2 * ( dy - dx ) ;
      drawline ( x1, y1, x2, y2, PREDX, DECR ) ;
}
/* -1 > m > 0 */
if ( -dy > dx && dy < 0 )
{
      dx = -dx ;
      e_noinc = - ( 2*dx ) ; e = 2 * dx - dy ;
      e_inc = - 2 * ( dx - dy ) ;
      drawline ( x2, y2, x1, y1, PREDY, DECR ) ;
}
```


void lin()
start of function
int ix,iy,px=1,py=1 ,but,flag=0
1 ?
Y
getmousepos()
x>=35&&x<=600&&y>=40 &&y<=450 ?
Y
changemouseptr(pencil)
px!=x||py!=y ?
Y
printmousepos()
N
button==1 ?
Y
flag==0 ?
Y
hidemouseptr()
ix=px=x
iy=py=y
flag=1
N
px!=x||py!=y ?
Y
bressline1(ix,iy,px,py)
bressline(ix,iy,x,y)
N
flag!=0 ?
Y
showmouseptr()
flag=0
N
px=x
py=y
N
N
changemouseptr(hand)
end of function
end of source

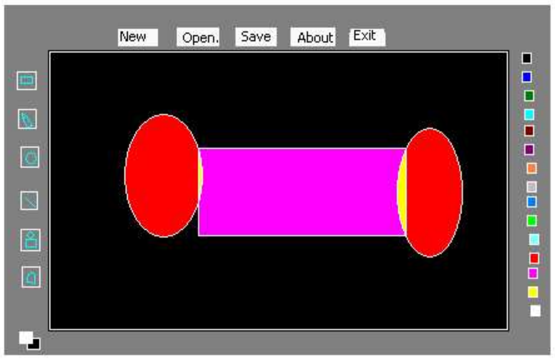

New
Open.
Save
About
Exit

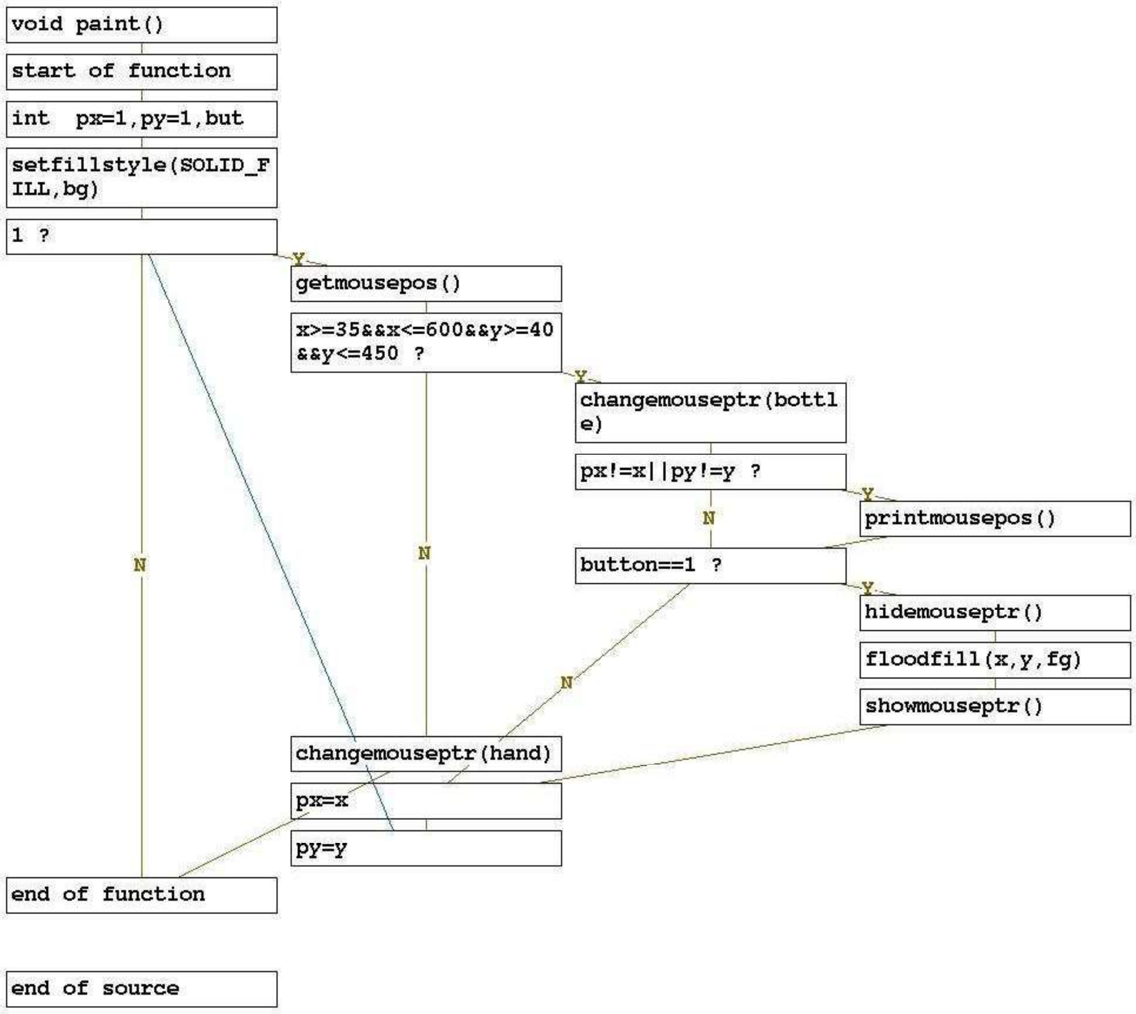
4.2.5. Fill Area tool

This is used to fill a closed area.

Using the flood fill algorithm, we fill the required area.

```
void paint()
{
   int px=1,py=1,but;
   setfillstyle(SOLID_FILL,bg);

     while(1)
     {
     getmousepos();

         if(x>=35&&x<=600&&y>=40&&y<=450)
         {
         changemouseptr(bottle);

               if(px!=x||py!=y)
                printmousepos();

                if(button==1)
                {
                hidemouseptr();
                floodfill(x,y,fg);
                showmouseptr();
                }
         }
         else
         {
          changemouseptr(hand);
          break;
         }
             px=x;
             py=y;
     }
}
```





4.2.6. Polygon tool:

By using bresenhams line algorithm, we draw the lines that are necessary to construct the polygon and use a simple for loop to join the lines.

4.3. Color tool bar

'C' graphics provides only 16 different colors shown in EGA directory. All these 16 colors are available in the directory " bgi ". The 16 colors and their corresponding integer values are given below:

BLACK (0), BLUE (1), GREEN(2), CYAN (3), RED (4), MAGENTA (5), BROWN(6), LIGHTGRAY (7), DARKGRAY(8), LIGHTBLUE (9), LIGHTGREEN (10), LIGHTCYAN (11), LIGHTRED (12), LIGHTMAGENTA (13), YELLOW (14), WHITE (15).

The colors with integer value >8 cannot be used as background color.

4.4. Status bar

The status bar is placed at the bottom of the display screen to show the help options and also to show where the pointer is currently present.

4.5. Drawpad

A draw pad is provided to draw the required image. We can select the color the background of the drawpad.

4.6. Mouse pointer

The various mouse functions can be accessed by setting up the AX register with different values (service number) and issuing interrupt number 51. The functions are listed below.

4.6.1. initmouseptr()

In this function AX is set to "1". When this function is called in main() it displays the mouse pointer. The position of the pointer can be changed by using the mouse.

4.6.2. hidemouseptr()

In this function AX is set to "2".When this function is called in main() it hides the mouse pointer. This function is useful while drawing figures, first the mouse pointer is kept hidden, then the figure is been drawn and again the mouse pointer is been called.

4.6.3. getmouseptr()

In this function AX is set to "3". This function returns the position of the mouse pointer. It contains three parameters,they are xpos,ypos,click. xpos and ypos returns the position of x co-ordinate and y co-ordinate respectively. Click is the integer variable which returns the values 1,2,3 corresponding to the button pressed on the mouse and 0 for buttons being not pressed. If any key is pressed kbhit returns nonzero integer; if not it returns zero.

4.6.4. setmouseptr()

In this function AX is set to "4". This function sets the mouse pointer to specific position . CX is been loaded by x co-ordinate of the mouse pointer and DX is been loaded with the y co-ordinate of the mouse pointer.

5. Scope for future work

1. The algorithms used in this project work bit a slow. We need to improve the algorithms.

2. More applications can be added to the current applications.
3. The program can be made user friendly by adding more help options.
4. The display of the output can be improved.
5. Currently the images are saved in raw file format. The program must be extended to save the file into .bmp or .jpeg format.
6. The algorithm if written in languages other than C can make us add more applications easily.
7. We may extend it to draw 3D images.

6. Conclusion

Finally we are able to produce an application that would allow us to draw images using several options provided and save them and retrieve them whenever necessary.