

R.V. COLLEGE OF ENGINEERING
COMPUTER SCIENCE AND
ENGINEERING DEPARTMENT
BANGALORE

COMPUTER GRAPHICS LAB PROJECT REPORT

DESIGN AND IMPLEMENTATION OF A 2D
GRAPHICS EDITING PACKAGE

DEVELOPED BY

SRIVAS N. CHENNU 1RV98CS086

VISHWAS N. 1RV98CS105

6TH SEMESTER CSE

RVCE

CERTIFICATE



This is to Certify that **Srivas N. Chennu (1RV98CS086)** of 6th semester CSE Department has successfully and satisfactorily completed the Graphics Editor package Mini-Project work in partial fulfillment of the Computer Graphics Lab as prescribed by the VTU for the academic year 2001-2002.

**SIGNATURE OF THE
TEACHER IN CHARGE**

**SIGNATURE OF THE
HEAD OF DEPARTMENT**

CONTENTS

INTRODUCTION.....	4
SOFTWARE REQUIREMENT SPECIFICATION.....	5
REQUIREMENT ANALYSIS	5
PROBLEM	5
GOAL	6
SCOPE, REFERENCES, OVERVIEW.....	6
CONSTRAINTS.....	6
DESIGN.....	7
EDITOR ALGORITHM	8
FEATURES	9
MODULE DESCRIPTION	11
FUNCTIONAL DESCRIPTION.....	12
IMPLEMENTATION.....	21
GLOBAL.H.....	21
FILEOP.C	22
FUNC.C.....	23
GEDIT.C.....	42
MOUSE.C.....	56
SHEAR.C.....	57
TESTING.....	58
CONCLUSION.....	59
BIBLIOGRAPHY	60

INTRODUCTION

A graphics - editing package is a collection of software functions, which allows the creation, manipulation, and storage of graphical images. A typical graphics editing package should have the basic graphic editing capabilities like freehand drawing, line drawing, providing the user with different kinds of brushes, functions to draw common shapes like rectangle, circle, ellipse, etc. color filling, and other such standard functionalities. It should also have the basic image transformation capabilities such as rotation, translation, scaling, shear, reflect, etc. Also included should be the cut, copy, paste and clear viewport operations.

Graphics editing software can be mainly classified on basis of design into the following categories.

An Object oriented editor is one in which in each graphic primitive drawn in the viewport is an object. Such objects can be selected individually and can be subjected to any of the transformations provided in the editor. The advantage of such editors is that the code can be easily written in using an Object Oriented Programming language like C++. Also advanced functionalities can be easily implemented because all that the editor has to do is to keep a stack of objects being drawn on the screen. The disadvantage is that the user can only select objects and not a part of the screen.

Another kind of editor is a pixel-based editor where in drawing primitives on the viewport is like painting on a canvas. Once an object is drawn it cannot be individually selected. Instead only a rectangular portion of the screen can be usually selected and subjected to various transformations or other operations. In other words the smallest object that can be selected and modified is a pixel. This editor is of the latter kind. The basic advantage is that of the simplicity in code of such an editor where in the smallest unit is a pixel. The disadvantage being that an individual object cannot be selected and subjected to transformations.

SOFTWARE REQUIREMENT SPECIFICATION

This is mainly the specification of the problem and how the solution is to be brought about.

Since requirements of a Graphics Editor are widely known, only the basic requirements and specifications are stated below without elaborating each of its functionalities.

REQUIREMENT ANALYSIS

This phase deals with the statement and understanding of the problems, the goals, and the constraints, which are listed below for the development of the Graphics Editor.

PROBLEM

The problem or aim is to develop software, which works on dos platform and is basically used to create, edit and save graphical data. The main mode of input or interaction with the user is through the mouse. Suitable buttons should be provided for all the functions implemented. The various functions to edit graphical images should be implemented along with basic image transformations and cut, copy and paste functions. The buttons should be laid out properly with appropriate icons on them, which reflect the button's functionality.

The software being designed should be one, which will help the user create and manipulate images and give him enough features and freedom to do the same. The software should also allow him to use Text and graphics. The software must run on DOS and should support a minimum of 16 colors.

GOAL

The goal is to implement the operations stated in the problem statement efficiently, consuming minimal resources.

SCOPE, REFERENCES, OVERVIEW

The scope of the project is to build a graphics editor working on DOS platform. Its functionalities should be similar to that of those in MS-Paint. Specifically, the following functionality should be implemented.

- ❑ Tools to plot basic output primitives
- ❑ File operations such as Save, Save As, New, Load etc.
- ❑ Other tools include spray, brush, eraser, fill, etc.
- ❑ 2-D transformations such as translation, rotation, shear and scaling.
- ❑ Features to Cut, Copy and Paste parts of the canvas i.e. the selected regions.
- ❑ The Graphics Editor may also have text editing features, extra file operations, 3-D rendering, etc.

CONSTRAINTS

As the software is being built to run on a DOS platform, which gives access to a maximum of only 640kB of conventional memory, I have to make efficient use of the memory. Also to make the software run even on low-end machines the code should be efficient and optimal with the minimum of redundancies. Needless to say, the editor should also be robust and fast.

DESIGN

The Graphics Package is designed using the library of graphics manipulation functions, which are incorporated as a part of the package, wherein a good user interface is first made to run the package.

The objects, which can be drawn using the editor, are stored as structures. A line is taken to be an array of x & y co-ordinates. Similarly, a rectangle is stored as a set of 4 points; a circle's structure consists of the center & radius, etc.

The high level design of the graphics package incorporates the concepts of procedural and modular programming. The entire graphics package can be decomposed into the following components, based on their function.

- **The Graphics Manipulation Subroutines** – This component constitutes the ‘backend’ of the software, and contains all the functions for the creation, manipulation and rendering of graphics primitives. For example, this subroutine library contains functions for drawing lines, circles, clipping previously drawn primitives etc.
- **The User Interface Management Subroutines** – These functions manage the user interaction processes of the graphics package. Essentially they contain the input device management and display device management logic. This layer of functions accept user input and in turn call the graphics manipulation subroutines which actually execute the user’s request by drawing primitives on the screen.
- **File Management Subroutines** – The file management subroutines are used primarily to work with disk files. These subroutines are responsible for managing File Input/Output. They accept requests from the User Interface library for reading and writing image files from and to the disk storage devices.

EDITOR ALGORITHM

Given below is the fundamental algorithm that forms the basis of the input handling and processing logic in the graphics editor.

Step 1: [Initialization of the window and variables]

Display startup screen.
Initialize the image data and the variables related to it.
Initialize the window variables.
Initialize the esc_flag to zero.
Esc-flag = 0.

Step 2: [Repeat Step 3 until esc-flag = 1]

Step 3: Do

 Begin
 Read mouse status/key from keyboard.
 If mouse button is clicked and pointer is in the icon window
 Select appropriate tool where button is clicked.
 Else if mouse button is clicked and pointer in color palette
 Select color as foreground/background color.
 Else if mouse button is clicked and pointer in pattern window
 Select desired pattern.
 Else if mouse button is clicked and pointer in canvas
 Perform appropriate function depending on tool.
 Else if key pressed
 Look for function to be performed on key press.
Else if (esc_flag)
 Esc_flag = 1;
 Else
 Ignore the input.
End

Step 4: End of the Algorithm.

FEATURES

The following are the features that are provided by the package.

Pen:

After this button is clicked, the user can use the mouse as a pen. The user left clicks on the screen and drags the mouse cursor along any path. The path will be plotted on the screen on the release of the pen.

Eraser:

After this button has been clicked, the user is allowed to use the mouse cursor as an eraser. This works just like the pen but plots thick points of white color.

Line:

Selecting this option the user is allowed to draw straight lines using the rubber band techniques of any color. The user left clicks at one end point of the line and drags the mouse. The line stretches between the initial point and the final point. The left button is released after the desired length is reached.

Circle:

This option allows the user to draw circles at any position using the rubber-band technique. The user left clicks on the center and drags the mouse increasing the radius. When the desired radius is reached the user releases the mouse left button.

Rectangle:

Rectangles are also drawn using the rubber band method. The rectangle stretches from the left top corner to the bottom right corner.

Ellipse:

Ellipses are drawn just like the rectangles.

Fill Color:

Using the option-enclosed regions can be filled with the selected colors. Windows displaying the color appear at the bottom of the screen. Any pattern and color may be selected.

Spiral:

After selecting this option the user has to click two points on the screen. The first point will be treated as the center and the distance between the points will be treated as the radius of the spiral.

Clip:

This option is used for clipping as well as blanking. The user is prompted to draw the clip window and on clicking the clipping is completed.

Load:

The user can load files by clicking on this icon. The user is prompted to enter the file name and the contents of the file are displayed on the screen.

Save:

The user can save the files using this option. He has only to type the file name.

Clear:

On clicking this button the screen is cleared.

Exit:

The Exit button is used to terminate the graphics editor and exit to the Operating System.

MODULE DESCRIPTION

Given below is basic description of the various modules in the graphics package and their functions.

GLOBAL.H

This file contains prototypes and declarations used by all the modules give below.

GEDIT.C

This file contains the main function of the entire program. The execution starts from this file. This file also contains functions to draw the front end i.e. the screen, the icons, the palettes etc.

FUNC.C

This file contains various functions required to achieve the purpose of the graphic editor operations like drawing a CIRCLE, RECTANGLE, and LINE and to perform basic transformations etc.

MOUSE.C

This file contains the various functions to control mouse operations like initializing the mouse, showing and hiding the mouse pointer, restricting the mouse pointer within a boundary etc.

SHEAR.C

This file contains the functions to produce shearing of the entire working area of the graphics editor in x or y directions.

FILEOP.C

This file contains the functions to save and open a file used by the graphic editor.

HELP.C

This file contains the function that displays the help on the user screen.

MISCELL.C

This file contains miscellaneous functions, which are used to check for CAPSLOCK, INSERT CHECK etc.

FUNCTIONAL DESCRIPTION

GEDIT.C

void clear_icon () :

This function draws the clearscreen icon.

void spray_icon () :

This function draws the spray icon.

void pencil () :

This function draws the scratch tool icon.

void open_icon () :

This function draws the file open icon.

void save_icon () :

This function draws the file save icon.

void fill_icon () :

This function draws the flod fill icon.

void translate_icon () :

This function draws the translate icon.

void eraser_icon () :

This function draws the eraser icon.

void text_icon () :

This function draws the icon for rotation and shear.

`void draw_icon () :`

This function calls the above functions to draw individual icons in a specified position on the screen.

`void draw_screen () :`

This function draws the front end of the graphics editor by calling the above functions and by making use of in-built functions.

`int check (int x,int y) :`

This function is used to check whether the mouse pointer lies within the drawing range of the editor screen.

`void show_cord () :`

This function displays the co-ordinates of the mouse pointer's position on the screen.

`void clears () :`

This function clears the drawing area of the graphic editor.

`void show_sel_col () :`

This function shows the selected background color and foreground color at the bottom of the screen.

`void regioncheck (int x,int y) :`

This function checks for the `tool_id` selected by specifying the region of selection for a particular function to perform and there by updating the `tool_id` by verifying for a mouse click in the defined regions of existing icons.

`void bkgndchk (int x,int y) :`

This function checks for the background color by checking the mouse click on the color palette and updates the background color selected.

`void main () :`

This function as the name indicates is the main function for the entire graphic editor and is the function where execution begins.

FUNC.C

void swap (int *a,int *b) :

This function as is obvious, swaps the values of a and b. Since the pointers to a and b variables are passed, we simulate call by reference thus swapping the values of a and b.

void putpoint (int x,int y,int color,int xor) :

This function is similar to putpixel but uses interrupt to achieve the same. This function prints a pixel of the specified color with the option of xoring the new color onto the old pixel color.

void flip () :

This function is used to flip the image on the whiteboard of the screen.

void rotate () :

This function is used to rotate the image on the whiteboard by 180 degree.

void getimg (int x1,int y1,int x2,int y2) :

This function is used to store the image specified by the co-ordinates x1, y1 and x2, y2 of rectangular area in a file.

void putimg (int x1,int y1,int x2,int y2) :

This function is used to put an image onto the whiteboard specified by the co-ordinates x1, y1 and x2, y2 of rectangular area from a file.

void storepixel (int x,int y,int col) :

This function stores details of a particular pixel such as its co-ordinates and its color.

void putback () :

This function is used to put back the stored pixels effectively back into their previous positions.

void putback1 () :

This function is used to put back the stored pixels effectively back into their previous positions only after checking whether the pixel's position is within the whiteboard.

void my_line (int x1,int y1,int x2,int y2) :

This function draws a line between the given co-ordinates as arguments.

void freehand () :

This function enables free hand drawing by selecting the scratch tool icon on the screen.

void eraser () :

This function acts like an eraser by plotting a square of white pixels on the area where the mouse is clicked on selecting the eraser icon.

void cirplotpoints (int xc,int yc,int x,int y) :

This function plot points for a circle based on the arguments passed.

void getcircle (int xc,int yc,int rad) :

This function gets the radius of the circle and the centre by calculating the mouse movements and then plots the circle by calling cirplotpoints function.

void my_circle () :

This function gets the initial centre of the circle and plots the circle by calling the getcircle function.

void ellplotpoints (float x,float y,int xc,int yc) :

This function plot points for an ellipse based on the arguments passed.

`void getell (int xc,int yc,float rx,float ry) :`

This function actually calculates the major and minor axis of the ellipse by noting the mouse movements and plots the ellipse.

`void my_ellipse () :`

This function actually calculates the origin of the mouse click when ellipse is selected and calls the function `getell` to actually plot the ellipse.

`void myline (int x1,int y1,int x2,int y2,int color,int code) :`

This function is similar to the built-in function `line ()`. This function gives an extra option to xor the original contents of the screen with the new color. This function employs Bresenham's Line Drawing Algorithm.

`void myrectangle (int x1,int y1,int x2,int y2,int color,int code) :`

This function draws a rectangle at the specified coordinates. This rectangle can be xorred on to the original screen contents depending on the value of code which can be in XOR or NOXOR mode. This function uses the `myline` function to draw a rectangle.

`void dottedrect (int x1,int y1,int x2,int y2,int color,int xor) :`

This function draws a dotted rectangle which, again, can be xorred onto the screen. This dotted rectangle is drawn making use of a variable named `toggle` to print and skip adjacent pixels.

`void drawrectangle (int firstmx,int firstmy,int firstmb) :`

This function draws a rectangle on the canvas using the rubber band technique. The new rectangle is XORred onto the screen and the old rectangle is erased after XORing it with the another rectangle at the same place. When the user lets go of the mouse button, the rectangle is redrawn in the color of his choice. This function invokes the `myrectangle` function.

void drawdottedrect (int firstmx,int firstmy,int firstmb) :

This function is invoked by the select function, which needs a dotted rectangle drawn to enable the user to see the chosen area on the canvas. This function draws a dotted rectangle on the canvas using the rubber band technique. The new dotted rectangle is XORred onto the screen and the old dotted rectangle is erased after XORing it with the another dotted rectangle at the same place. When the user lets go of the mouse button, the dotted rectangle is redrawn in the color of his choice. This function invokes the dottedrect function.

void copy (int choice) :

This function is either used to cut or copy the image specified by the select option by storing the selected area into a file and also clearing the area in case of cut operation i.e when choice=1 with white pixels.

void fillitup (int x,int y) :

This function compares the pixels positions foreground and background color and plots a pixel with the background color if the foreground and background color do not match.

void fill () :

This function compares the pixels positions foreground and background color and ccalls the function fillitup if the foreground and background color do not match.

void spray () :

This function plots pixels randomly over a circular area with the specified foreground color just like sprinkling colors over a circular region.

void screen_spiral (int sx,int sy,float rad) :

This function plots a spiral on the screen at the specified co-ordinates by the mouse pointer dependin on the radius.

void translate (int tx, int ty) :

This function translates the whole setup of the white board based on the translation factors specified by the arguments tx and ty in x and y direction respectively.

void runproc () :

This function selects a particular task to be performed based on the tool_id selected and calls their respective defined above functions to perform the specified task.

void display_box (int x1,int y1,int x2,int y2,int rd) :

This function is used to display a message box for operations such as save and opening of a file, to take entries for translation, rotation, shear values etc and also provides choices for performing functions such as rotation or shear etc.

MOUSE.C

`int initmouse () :`

This function is used to initialise the mouse at the start of the program.

`void showmouseptr () :`

This function is used to show the mouse pointer.

`void hidemouseptr () :`

This function is used to hide the mouse pointer.

`void restrictmouseptr (int x1,int y1,int x2,int y2)`

This function is used to restrict the mouse pointer to a specific rectangular area depending on the arguments x1, y1, x2, y2.

`void getmouseptr (int *button,int *x,int *y) :`

This function is to get the co-ordinates of the mouse pointers position at that instance and also the status of the button i.e to check for left or right mouse button.

SHEAR.C

void y_shear(int sh) :

This function is to perform shearing of the whiteboard in y-direction depending on the value of the argument passed i.e sh.

void x_shear(int sh) :

This function is to perform shearing of the whiteboard in x-direction depending on the value of the argument passed i.e sh.

FILEOP.C

void store_file (char * filed) :

This function is used to save the whiteboard contents into the file specified by the user, which is the argument filed.

void open_file (char * filed) :

This function is used to open an already saved file specified by the argument filed and to display it on the whiteboard for viewing or further editing if required.

IMPLEMENTATION

The complete source code listing for the graphics package is described below. The source code is organized according the file modules listed above.

GLOBAL.H

```
#include<graphics.h>
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<dos.h>
#include<bios.h>
#include<stdlib.h>
#define PI 3.141592654
#define XOR 0x80
#define NOXOR 0

union REGS in,out;
int fcolor=0;
int selx1=0,sely1,selx2,sely2,copyflag=0;
FILE *fp;
void far *buf;
int maxx,maxy,x_arr[2000],y_arr[2000],col_arr[2000];
int count=0,tool_id=1,old=0,bkcol=15;
char far *p1,*p2,*p3,*p4;
void display_box();
void show_cord();
int selected=0;
```

FILEOP.C

```
void store_file(char * filed)
{
    int i,j;
    int ch,prev,cnt;
    hidemouseptr();
    fp=fopen(filed,"w");
    for(i=60;i<639;i++)
    for(j=41;j<409;j++)
    {
        ch=getpixel(i,j);

        if(ch==13)
            ch=16;
        putc(ch,fp);
    }
    showmouseptr();
}

void open_file(char * filed)
{
    int i,j;
    int ch;
    fp=fopen(filed,"r");
    hidemouseptr();
    for(i=60;i<639;i++)
    for(j=41;j<409;j++)
    {
        ch=getc(fp);

        if(ch==16) ch=13;
        putpixel(i,j,ch);
    }
    showmouseptr();
    fclose(fp);
}
```

FUNC.C

```
void swap(int *a,int *b)
{
    int temp;
    temp=*a;
    *a=*b;
    *b=temp;
}

void putpoint(int x,int y,int color,int xor)
{
    union REGS i;
    i.h.ah=0x0c;
    i.h.al=( color | xor );
    i.h.bh=0;
    i.x.cx=x;
    i.x.dx=y;
    int86(0x10,&i,&i);
}

void flip()
{ int y,x,i,j;
  hidemouseptr();
  for(y=41;y<409;y++)
    for(x=60;x<=(639-59)/2+60;x++)
      {
        i=getpixel(x,y);
        j=getpixel(639-x+60,y);
        if(i!=j)
          { putpixel(x,y,j);
            putpixel(639-x+60,y,i);
          }
      }
  showmouseptr();
}

void rotate()
{ int i,j,k,t;
  hidemouseptr();
  for(i=1;i<370-2;i++)
  for(j=1;j<=i-2;j++)
  {
    t=getpixel(i+60,j+41);
    k=getpixel(j+60,i+41);
    if(t!=k)
```

RVCE

```
        { putpixel(i+60,j+41,k);
          putpixel(j+60,i+41,t);
        }
    }
    //clears the left boundary
    setfillstyle(SOLID_FILL,15);
    bar(429,40,639,409);
    showmouseptr();
}

void getimg(int x1,int y1,int x2,int y2)
{
    int i,j;
    int ch,prev,cnt;
    hidemouseptr();
    fp=fopen("text.pic","w");
    for(i=x1;i<=x2+1;i++)
    {
        for(j=y1;j<=y2;j++)
        {
            ch=getpixel(i,j);
            if(ch==13) ch=16;
            putc(ch,fp);
        }
    }
    showmouseptr();
}

void putimg(int x1,int y1,int x2,int y2)
{
    int i,j;
    int ch;
    fp=fopen("text.pic","r");
    hidemouseptr();
    for(i=x1;i<=x2+1;i++)
    {
        for(j=y1;j<=y2;j++)
        {
            ch=getc(fp);
            if(ch==16) ch=13;
            if (i<639 && j<409 && i>59 && j>40)
                putpixel(i,j,ch);
        }
    }
    showmouseptr();
    fclose(fp);
}

void storepixel(int x,int y,int col)
{
    count++;
    x_arr[count]=x;
    y_arr[count]=y;
}
```

RVCE

```
    col_arr[count]=col;
}

void putback()
{
    int i;
    for(i=0;i<count;i++)
        putpixel(x_arr[i],y_arr[i],col_arr[i]);
    count=0;
}

void putback1(int col)
{
    int i;
    for(i=0;i<count;i++)
        if(check(x_arr[i],y_arr[i]))
            putpixel(x_arr[i],y_arr[i],col);
}

void my_line1(int x1,int y1,int x2,int y2)
{
    int p,x,y,k,i,dx,dy;
    float m;
    dy=abs(y2-y1);
    dx=abs(x2-x1);
    if (dx==0 || (abs((float)dy/dx)>=1) )
        {if (y1>y2) {k=x1,x1=x2,x2=k;
            k=y1,y1=y2,y2=k;}
        x=x1;
        y=y1;
        storepixel(x,y,getpixel(x,y));
        p=2*dx-dy;
        while (y!=y2)
            {if (p<0) p+=2*dx;
             else {(x2>x1)?x++:x--;
                 p+=(2*dx-2*dy);}
              y++;
              storepixel(x,y,getpixel(x,y));
            }
        }
    else
        {if (x1>x2) {k=x1,x1=x2,x2=k;
            k=y1,y1=y2,y2=k;}
        x=x1;
        y=y1;
        storepixel(x,y,getpixel(x,y));
        p=2*dy-dx;
        while (x!=x2)
            {if (p<0) p=p+2*dy;
```

RVCE

```
        else {(y2>y1)?y++:y--;
              p=p+2*dy-2*dx;}
        x++;
        storepixel(x,y,getpixel(x,y));
    }
}

void my_line()
{
    int x,y,button1,i,prevx,prevy,tx,ty;
    getmouseptr(&button1,&x,&y);
    if(button1)
    {
        count=0;
        hidemouseptr();
        tx=prevx=x;
        ty=prevy=y;
        my_line1(prevx,prevy,x,y);
        if (button1==2)
            putback1(bkcol);
        else
            putback1(fcolor);
        while(button1 )
        {
            show_cord();
            setcolor(fcolor);
            getmouseptr(&button1,&x,&y);
            if(x!=tx||y!=ty)
            {
                putback();
                my_line1(prevx,prevy,x,y);
                if (button1==2)
                    putback1(bkcol);
                else
                    putback1(fcolor);
                tx=x;ty=y;
            }
        }
        showmouseptr();
    }
}

void freehand()
{
    int s,x,y,button1,i,j,prevx,prevy,m,l,k;
    getmouseptr(&button1,&x,&y);
    if(button1)
    {
        hidemouseptr();
        prevx=x;
        prevy=y;
    }
}
```

RVCE

```
        while(button1)
        {
            show_cord();
            if (button1==2)
                setcolor(bkcol);
            else
                setcolor(fcolor); i=0;
                if(check(prevx,prevy) && check(x,y))
                    line(prevx,prevy+i,x,y+i);
                prevx=x;
                prevy=y;
                getmouseptr(&button1,&x,&y);
        }
        showmouseptr();
    }
}

void eraser()
{
    int s,x,y,button1,i=0,j,prevx,prevy,m,l,k;
    setcolor(15);
    getmouseptr(&button1,&x,&y);
    if((button1 & 1)==1)
    {
        hidemouseptr();
        prevx=x;
        prevy=y;
        while((button1 & 1)==1)
        {
            show_cord();
            setcolor(15);
            setfillstyle(SOLID_FILL,WHITE);
            if(check(prevx,prevy) && check(x,y)&&check(prevx-10,prevy-
10)&&check(prevx+10,prevy+10))
                bar(prevx-10,prevy+i-10,x,y+i);
            prevx=x;
            prevy=y;
            getmouseptr(&button1,&x,&y);
        }
        showmouseptr();
    }
    setcolor(fcolor);
}

void circplotpoints(int xc,int yc,int x,int y)
{
    x_arr[count]=xc+x;y_arr[count]=yc+y;
```

RVCE

```
col_arr[count++]=getpixel(xc+x,yc+y);
x_arr[count]=xc+x;y_arr[count]=yc-y;
col_arr[count++]=getpixel(xc+x,yc-y);
x_arr[count]=xc-x;y_arr[count]=yc+y;
col_arr[count++]=getpixel(xc-x,yc+y);
x_arr[count]=xc-x;y_arr[count]=yc-y;
col_arr[count++]=getpixel(xc-x,yc-y);
x_arr[count]=xc+y;y_arr[count]=yc+x;
col_arr[count++]=getpixel(xc+y,yc+x);
x_arr[count]=xc+y;y_arr[count]=yc-x;
col_arr[count++]=getpixel(xc+y,yc-x);
x_arr[count]=xc-y;y_arr[count]=yc+x;
col_arr[count++]=getpixel(xc-y,yc+x);
x_arr[count]=xc-y;y_arr[count]=yc-x;
col_arr[count++]=getpixel(xc-y,yc-x);
}

void getcircle(int xc,int yc,int rad)
{
    int x,y,p;
    x=0;
    y=rad;
    p=3-2*rad;
    circplotpoints(xc,yc,x,y);
    while(x<y)
    {
        x++;
        circplotpoints(xc,yc,x,y);
        if(p<0)
            p+=4*x+6;
        else
        {
            p+=4*(x-y)+10;
            y--;
        }
    }
    if(x==y)
        circplotpoints(xc,yc,x,y);
}

void my_circle()
{
    int button1,xc,yc,i,j,x,y,rad,temp;
    showmouseptr();
    getmouseptr(&button1,&x,&y);
    count=0; /*reset count to 0 for this circle*/
    if(button1 ==1)
    {
        xc=i=x;
        yc=j=y;
        if(check(xc,yc))
```

RVCE

```
    {
    hidemouseptr();
    rad=0;
    getcircle(xc,yc,rad);
    putback1(fcolor);
    showmouseptr();
    }
    getmouseptr(&button1,&x,&y);
    while((button1)!=1)
    {
        show_cord();
        setcolor(fcolor);
    if(i!=x || j!=y)
    {
        i=x;j=y;
        rad=sqrt(abs(abs((xc-i)*(xc-i)) + abs((yc-j)*(yc-j))));
        {
            hidemouseptr();
            putback();
            getcircle(xc,yc,rad);
            putback1(fcolor);
            showmouseptr();
        }
    }
    getmouseptr(&button1,&x,&y);
    }
}
}
```

```
void ellplotpoints(float x,float y,int xc,int yc)
{
    if (x<1) x=0;
    if (y<1) y=0;

    x_arr[count]=xc+x;y_arr[count]=yc+y;
    col_arr[count++]=getpixel(xc+x,yc+y);
    x_arr[count]=xc+x;y_arr[count]=yc-y;
    col_arr[count++]=getpixel(xc+x,yc-y);
    x_arr[count]=xc-x;y_arr[count]=yc+y;
    col_arr[count++]=getpixel(xc-x,yc+y);
    x_arr[count]=xc-x;y_arr[count]=yc-y;
    col_arr[count++]=getpixel(xc-x,yc-y);
}
}
```

```
void getell(int xc,int yc,float rx,float ry)
{
    float x,y,p,endv,k;
    x=0;
```

RVCE

```
y=ry;
k=(ry*ry)/(rx*rx);
p=2*k-2*ry+1;
while (x<rx)
    {ellplotpoints(x,y,xc,yc);
    if (y!=0 && (k*x/y)>1) break;
    if (p<0) p+=4*k*x+6*k;
    else p+=4*k*x-4*y+6*k+4,y--;
    x++;
    }
endv=y;
y=0;
x=rx;
k=(rx*rx)/(ry*ry);
p=2*k-2*rx+1;
while (y<=endv)
    {ellplotpoints(x,y,xc,yc);
    if (p<0) p+=4*k*y+6*k;
    else p+=4*k*y-4*x+6*k+4,x--;
    y++;
    }
}

void my_ellipse()
{
    int button1,xc,yc,sx,sy,height,width,i,j,x,y;
    float rx,ry;

    setcolor(fcolor);
    showmouseptr();
    getmouseptr(&button1,&x,&y);
    count=0;
    if(button1 ==1)
    {

        sx=i=x;
        sy=j=y;
        if(check(sx,sy))
        {
            hidemouseptr();
            getell(sx,sy,1.0,1.0);
            putback1(fcolor);
            showmouseptr();
        }

        getmouseptr(&button1,&x,&y);

        while((button1)==1)
        {
            show_cord();
            setcolor(fcolor);
```

RVCE

```
if(i!=x || j!=y)
{
    i=x;j=y;
    height=abs(sy-j);
    width=abs(sx-i);
    if((float)(floor(width/2))< 1.0 )
    rx=1;
    else
    rx=(float)(floor(width/2)); //if(rx<1)rx=2;

    if((float)(floor(height/2)) < 1.0)
    ry=1;
    else
    ry=(float)(floor(height/2)); //if(ry<1)ry=2;

    xc=(int)(floor((sx+i)/2)); //if(xc<1)xc=2;
    yc=(int)(floor((sy+j)/2)); //if(yc<1)yc=2;
    {
        hidemouseptr();
        putback();
        getell(xc,yc,rx,ry);
        putback1(fcolor);
        showmouseptr();
    }
    getmouseptr(&button1,&x,&y);
}
showmouseptr();
}
setcolor(fcolor);
}

void myline(int x1,int y1,int x2,int y2,int fcolor,int code)
{
    int x,y,p,sign,inc;
    float slope,deltax,deltay;
    if (x1>x2)
    {
        swap(&x1,&x2);
        swap(&y1,&y2);
    }
    x=x1;
    y=y1;
    if (x1==x2)
    {
        for(p=y1;p!=y2;((y1-y2)<0)?p++:p--)
            putpoint(x1,p,fcolor,code);
        putpoint(x1,y2,fcolor,code);
        return;
    }
}
```

RVCE

```
    }
    else if (y1==y2)
    {
        for(p=x1;p!=x2;((x1-x2)<0)?p++:p--)
            putpoint(p,y1,fcolor,code);
        putpoint(x2,y1,fcolor,code);
        return;
    }
    deltax=abs(x2-x1);
    deltay=abs(y2-y1);
    slope=deltay/deltax;
    sign=y1-y2;
    p=2*deltay-deltax;
    if (sign<0)
        inc=1;
    else
        inc=-1;
    if (slope<1)
    {
        while(x!=x2)
        {
            putpoint(x,y,fcolor,code);
            if (p<0)
                p+=2*deltay;
            else
            {
                p+=2*(deltay-deltax);
                y+=inc;
            }
            x++;
        }
        putpoint(x,y,fcolor,code);
    }
    else
    {
        p=2*deltax-deltay;
        while(y!=y2)
        {
            putpoint(x,y,fcolor,code);
            if (p<0)
                p+=2*deltax;
            else
            {
                p+=2*(deltax-deltay);
                x++;
            }
            y+=inc;
        }
        putpoint(x,y,fcolor,code);
    }
}
```

RVCE

```
void myrectangle(int x1,int y1,int x2,int y2,int fcolor,int code)
{
    if (x1>x2)
    {
        swap(&x1,&x2);
        swap(&y1,&y2);
    }
    if (y1>y2)
        swap(&y1,&y2);
    if (x1<59)
        x1=59;
    if (y1<40)
        y1=40;
    if (x2>639)
        x2=639;
    if (y2>409)
        y2=409;
    myline(x1,y1,x2,y1,fcolor,code);
    myline(x1,y1,x1,y2,fcolor,code);
    myline(x2,y1,x2,y2,fcolor,code);
    myline(x1,y2,x2,y2,fcolor,code);
}

void drawrectangle()
{
    int firstmx,firstmy,firstmb;
    int oldmx,oldmy,mx,my,mb,ccolor;
    getmouseptr(&firstmb,&firstmx,&firstmy);
    if (fcolor==BLACK) ccolor=WHITE;
    else ccolor=fcolor;
    if (firstmx>59 && firstmx<639)
    if (firstmy>40 && firstmy<409)
    {oldmx=firstmx;
    oldmy=firstmy;
    restrictmouseptr(59,40,639,409);
    do
    {
        getmouseptr(&mb,&mx,&my);
        if (oldmx!=mx || oldmy!=my)
        {
            hidemouseptr();
            myrectangle(firstmx,firstmy,oldmx,oldmy,ccolor,XOR);
            myrectangle(firstmx,firstmy,mx,my,ccolor,XOR);
            showmouseptr();
            oldmx=mx;
            oldmy=my;
        }
    }
    while(mb==firstmb);
}
```

RVCE

```
hidemouseptr();
myrectangle(firstmx,firstmy,mx,my,fcolor,NOXOR);
showmouseptr();
restrictmouseptr(0,0,639,479);
}
}

void dottedrect(int x1,int y1,int x2,int y2,int color,int xor)
{
    int i;
    unsigned char toggle=0;
    if (x1>x2)
    {
        swap(&x1,&x2);
        swap(&y1,&y2);
    }
    if (y1>y2)
        swap(&y1,&y2);
    for (i=x1;i<=x2;i++)
    {
        toggle^=0xff;
        if (toggle)
        {
            putpoint(i,y1,color,xor);
            putpoint(i,y2,color,xor);
        }
    }
    toggle=0xff;
    for (i=y1;i<=y2;i++)
    {
        toggle^=0xff;
        if (toggle)
        {
            putpoint(x1,i,color,xor);
            putpoint(x2,i,color,xor);
        }
    }
}

void drawdottedrect()
{
    int firstmx,firstmy,firstmb;
    int oldmx,oldmy,mx,my,mb,color=WHITE;
    getmouseptr(&firstmb,&firstmx,&firstmy);
    if(firstmx>59 && firstmx<639)
    if(firstmy>40 && firstmy<409)
    {oldmx=firstmx;
    oldmy=firstmy;
    restrictmouseptr(59,40,639,409);
    if (selected!=0)
    {
```

RVCE

```
hidemouseptr();
dottedrect(selx1,sely1,selx2,sely2,color,XOR);
showmouseptr();
}
do
{
getmouseptr(&mb,&mx,&my);
if (oldmx!=mx || oldmy!=my)
{
hidemouseptr();
dottedrect(firstmx,firstmy,oldmx,oldmy,color,XOR);
dottedrect(firstmx,firstmy,mx,my,color,XOR);
showmouseptr();
oldmx=mx;
oldmy=my;
}
}
while(mb==firstmb);
if (firstmx>mx)
{
swap(&firstmx,&mx);
swap(&firstmy,&my);
}
if (firstmy>my)
swap(&firstmy,&my);
selx1=firstmx;
sely1=firstmy;
selx2=mx;
sely2=my;
restrictmouseptr(0,0,639,479);
selected=1;
}
}

void copy(int choice)
{
int mb,mx,my;
if (selected!=0)
{
hidemouseptr();
dottedrect(selx1,sely1,selx2,sely2,WHITE,XOR);
showmouseptr();
}
geting(selx1,sely1,selx2+2,sely2);
if(choice)
{
hidemouseptr();
setfillstyle(SOLID_FILL,WHITE);
bar(selx1,sely1,selx2,sely2);
showmouseptr();
}
}
```

RVCE

```
    selected=0;
}

void fillitup(int x,int y)
{
    int x1=x,x2=x+1,i;
    while(getpixel(x1,y)==bkcol && fcolor!=bkcol)
    {
        putpixel(x1,y,fcolor);
        x1--;
    }
    while(getpixel(x2,y)==bkcol &&fcolor !=bkcol)
    {
        putpixel(x2,y,fcolor);
        x2++;
    } /*SO FAR THE PRESENT LINE HAS BEEN FILLED*/
    /* NOW FILL ABOVE AND BELOW*/
    for(i=x1+2;i<=x2-2;i++)
    {
        if(getpixel(i,y-1)==bkcol && fcolor!=bkcol)
            fillitup(i,y-1);
        if(getpixel(i,y+1)==bkcol && fcolor !=bkcol)
            fillitup(i,y+1);
    }
}

void fill()
{
    int button1,x,y;
    getmouseptr(&button1,&x,&y);
    if(check(x,y))
    if(button1==1)
    {
        hidemouseptr();
        bkcol=getpixel(x,y);
        if(fcolor!=bkcol)
            fillitup(x,y);
        showmouseptr();
    }
}

void spray()
{
    int i,b,x,y,radius,p,t,density;
    // density=(density>10)?10:density;
    showmouseptr();
    radius=10;
    getmouseptr(&b,&x,&y);
    if(b==1)
    {
        hidemouseptr();
```

RVCE

```
while(b==1)
{
    delay(5);
    show_cord();
    setcolor(fcolor);

    // for(i=1;i<10;i++)
    p=rand()%radius;t=rand()%radius;
    if((radius*radius)>(p*p+t*t) )
        if(check(x+p,y+t))putpixel(x+p,y+t,fcolor);
    p=rand()%radius;t=rand()%radius;
    if((radius*radius)>(p*p+t*t) )
        if(check(x-p,y+t))putpixel(x-p,y+t,fcolor);
    p=rand()%radius;t=rand()%radius;
    if((radius*radius)>(p*p+t*t) )
        if(check(x+p,y-t))putpixel(x+p,y-t,fcolor);
    p=rand()%radius;t=rand()%radius;
    if((radius*radius)>(p*p+t*t) )
        if(check(x-p,y-t))putpixel(x-p,y-t,fcolor);

    getmouseptr(&b,&x,&y);
}
}
showmouseptr();
}

void getspiral(int sx,int sy,float rad)
{
    float s,t;
    int x,y;
    for (t=0;t<=6*PI;t+=0.01)
        {s=rad*(t/(6*PI));

x=x_arr[count]=(int)(sx+s*cos(t)),y=y_arr[count]=(int)(sy+s*sin(t)),col
_arr[count++]=getpixel(x,y);
        }
}

void my_spiral()
{
    int button,sx,sy,i,j,x,y;
    float rad;
    showmouseptr();
    getmouseptr(&button,&x,&y);
    count=0;
    if(button ==1)
    {
        sx=i=x;
        sy=j=y;
        if(check(sx,sy))
```

RVCE

```
    {
hidemouseptr();
getspiral(sx,sy,0.0);
putback();
showmouseptr();
    }

    getmouseptr(&button,&x,&y);

    while((button)==1)
    {   show_cord();
setcolor(fcolor);
if(i!=x || j!=y)
{

    i=x;j=y;
rad=sqrt(abs(abs((sx-i)*(sx-i)) + abs((sy-j)*(sy-j))));
    {
        hidemouseptr();
        putback();
        getspiral(sx,sy,rad);
        putback1(fcolor);
        showmouseptr();
    }
}
getmouseptr(&button,&x,&y);
}
showmouseptr();
}
}
```

```
void screen_spiral(int sx,int sy,float rad)
{
float s,t;
for (t=0;t<=6*PI;t+=0.01)
{s=rad*(t/(6*PI));
putpixel((int)(sx+s*cos(t)),(int)(sy+s*sin(t)),BLACK);
}
}
```

```
void translate(int tx, int ty)
{ int i,j,t;
hidemouseptr();
for(i=638;i>59;i--)
for(j=409;j>40;j--)
if(check(i+tx,j+ty))
{ t=getpixel(i,j);
putpixel(i+tx,j+ty,t);
putpixel(i,j,bkcol);
}
}
```

RVCE

```
    showmouseptr();
}

void runproc()
{
    void clears();
    int mb, mx, my;
    setfillstyle(SOLID_FILL, BLUE);
    setcolor(RED);
    bar(0, 460, 300, 479);
    setcolor(15);
    switch(tool_id)
    {
        case 1:  outtextxy(5, 463, "SCRATCH TOOL");
                freehand();
                break;

        case 2:  outtextxy(5, 463, "LINE");
                my_line();
                break;

        case 3:  outtextxy(5, 463, "ELLIPSE");
                my_ellipse();
                break;

        case 4:  outtextxy(5, 463, "RECTANGLE");
                drawrectangle();
                break;

        case 5:  outtextxy(5, 463, "CIRCLE");
                my_circle();
                break;

        case 6:  outtextxy(5, 463, "TRANSLATION");
                display_box(160, 165, 530, 215, 2);
                tool_id=1;
                break; //clip
                case 7:  outtextxy(5, 463, "SPRAY");
                        spray();
                        break;
                case 8:  outtextxy(5, 463, "FILL");
                        fill();
                        break;

        case 9:  outtextxy(5, 463, "ERASER");
                eraser();
                break;

        case 10: outtextxy(5, 463, "ROTATION AND SHEAR");
                display_box(160, 155, 450, 205, 3); tool_id=1; break;

        case 11: outtextxy(5, 463, "CLEAR SCREEN");
                clears();
                break; //clear screen

        case 12: outtextxy(5, 463, "SPIRAL");
                my_spiral();
                break;

        case 13: outtextxy(5, 463, "SELECT");
                drawdottedrect();
                break;
    }
}
```

RVCE

```
case 14:outtextxy(5,463,"COPY");
    if (old==13)
        copy(0);
        break;
case 15:outtextxy(5,463,"CUT");
    if (old==13)
        copy(1);
        break;
case 16:outtextxy(5,463,"PASTE");
    {
        //copy(0);
        //clears();
        getmouseptr(&mb,&mx,&my);
        putimg(mx,my,mx+selx2-selx1+1,my+sely2-sely1);
        //putimg(selx1,sely1,selx2,sely2);
        selected=0;
        //old=tool_id;
        //tool_id=1;
        //fflush(text.pic);
    }
    break;
case 17:outtextxy(5,463,"CLIP");
    {
        copy(0);
        clears();
        //getmouseptr(&mb,&mx,&my);
        //putimg(mx,my,mx+selx2-selx1+1,my+sely2-sely1);
        putimg(selx1,sely1,selx2,sely2);
        selected=0;
        //old=tool_id;
        //tool_id=1;
        //fflush(text.pic);
    }
    break;
}
}
```

```
void display_box(int x1,int y1,int x2,int y2,int rd)
{
    char filed[30];
    int check;
    int choice,ch,sx;
    int tx,ty;
    hidemouseptr();
    getimg(x1,y1,x2,y2);
    setfillstyle(SOLID_FILL,DARKGRAY);
    bar(x1,y1,x2,y2);
    setcolor(15);
    rectangle(x1,y1,x2,y2);
    rectangle(x1+2,y1+2,x2-2,y2-2);
}
```

RVCE

```
switch(rd)
{
  case 1:case 0:
    setcolor(15);
    outtextxy(230,180,"Enter Filename");
    gotoxy(33,13);
    check=getch();
    if(check==27)
    {putimg(x1,y1,x2,y2);
    break; }
    scanf("%s",filed);
    if(rd)
      open_file(filed);
    else
    {
    putimg(x1,y1,x2,y2);
    store_file(filed);
    }
    break;

  case 2:
    setcolor(15);
    outtextxy(170,180,"Enter translation factors(tx,ty):");
    gotoxy(55,12);
    scanf("%d %d",&tx,&ty);
    putimg(x1,y1,x2,y2);
    translate(tx,ty);
    break;
  case 3://160,165,450,315
    outtextxy(165,165,"Enter Choice:");
    outtextxy(168,178,"(1)Rotate ");
    outtextxy(168,188,"(2)Shear");
    gotoxy(35,11);
    scanf("%d",&choice);
    setfillstyle(SOLID_FILL,7);
    bar(x1+3,y1+3,x2-3,y2-3);
    switch(choice)
    {
      case 1:
        outtextxy(165,165,"Enter Choice:");
        outtextxy(168,178,"(1)Rotate by 90");
        outtextxy(168,188,"(2)Rotate by 180");
        gotoxy(35,11);
        scanf("%d",&ch);
        putimg(x1,y1,x2,y2);
        switch(ch)
        {
          case 1: rotate();flip();
            break;
          case 2: flip();
            break;
        }
      }
    }
}
```

RVCE

```
    }
    break;
case 2:  outtextxy(172,165,"Enter the shear values(sx)");
        gotoxy(49,11);
        scanf("%d",&sx);
        putimg(x1,y1,x2,y2);
        y_shear(sx);
        setcolor(0);
        rectangle(59,40,639,409);
        break;
    }
break;
}
```

GEDIT.C

```
# include "global.h"
# include "mouse.c"
# include "fileop.c"
# include "shear.c"
# include "func.c"

int i,j;

void clear_icon()
{
    setcolor(0);
    rectangle(16,330,42,351);
    setfillstyle(SOLID_FILL,WHITE);
    floodfill(29,341,BLACK);
}

void spray_icon()
{
    int bucket[12]={27,315,34,321,39,315,32,309,24,310,27,315};

    for(i=1;i<50;i++)

        putpixel(14+rand()%15,302+rand()%15,i%15);

    setfillstyle(SOLID_FILL,BROWN);
    fillpoly(6,bucket);
    setcolor(0);
}
```

RVCE

```
    drawpoly(6,bucket);
}

void pencil()
{
    int pencil[12]={24,358,24,373,29,379,34,373,34,358,24,358};
    drawpoly(6,pencil);
    setfillstyle(1,LIGHTRED);
    fillpoly(6,pencil);
    setcolor(0);
    drawpoly(6,pencil);
    line(27,358,27,373);
    line(31,358,31,373);
    line(24,373,34,373);
    setfillstyle(SOLID_FILL,0);
    floodfill(28,374,0);
}

void open_icon()
{
    //open file icon
    int open[10]={573,38,593,38,597,29,577,29,573,38};
    setfillstyle(SOLID_FILL,15);
    setcolor(0);
    bar(569,21,599,39);
    rectangle(569,21,599,39);
    setfillstyle(SOLID_FILL,YELLOW);
    bar(573,23,593,38);
    setcolor(0);
    rectangle(573,23,593,38);
    drawpoly(5,open);
    floodfill(594,30,0);
}

void save_icon()
{
    setfillstyle(SOLID_FILL,15);
    setcolor(0);
    bar(608,21,626,39);
    rectangle(608,21,626,39);
    setfillstyle(SOLID_FILL,0);
    bar(611,24,623,36);
    setfillstyle(SOLID_FILL,15);
    bar(615,24,619,28);
    bar(615,32,619,36);
    setcolor(0);
    rectangle(611,24,623,36);
    setfillstyle(SOLID_FILL,10);
    bar(616,33,617,35);
}
```

RVCE

```
void fill_icon()
{
    int fill[12]={25,286,33,294,40,286,32,278,25,286};
    setcolor(0);
    setfillstyle(SOLID_FILL,BROWN);
    fillpoly(5,fill);
    setcolor(15);
    for(i=0;i<3;i++)
    {
        line(27+i,284-i,22,284-i);
        line(20+i,284-i,20+i,294-i-i);
    }
    setcolor(0);
    line(30,291,37,284);
    line(31,292,38,285);
}

void translate_icon()
{
    setlinestyle(SOLID_LINE,1,1);
    rectangle(16,191,42,210);
    rectangle(26,202,42,210);
}

void eraser_icon()
{
    setcolor(0);
    setfillstyle(SOLID_FILL,3);
    bar3d(16,175,27,182,13,1);
}

void text_icon()
{
    setcolor(0);
    circle(29,256,11) ;
    circle(29,256,7);
    setfillstyle(SOLID_FILL,0);
    floodfill(29,256,0);
}

void draw_icon()
{
    int l,t,r,b,i;
    l=10,t=48,r=48,b=73;
    for(i=0;i<12;i++)
    { setcolor(0);
      rectangle(l,t+i*28,r,b+i*28);
      setfillstyle(SOLID_FILL,LIGHTGRAY);
      floodfill(l+10,t+i*28+10,BLACK);
    }
```

RVCE

```
    setcolor(15);
    line(l,t+i*28,r,t+i*28);
    line(l,t+i*28,l,b+i*28);
    setcolor(0);
    line(l,b+i*28,r,b+i*28);
    line(r,t+i*28,r,b+i*28);
}
for(i=0;i<5;i++)
{
    setcolor(0);
    rectangle(195+i*48,421,237+i*48,448);
    setfillstyle(SOLID_FILL,LIGHTGRAY);
    floodfill(200+i*47,425,BLACK);
    setcolor(15);
    line(195+i*48,421,237+i*48,421);
    line(195+i*48,421,195+i*48,448);
    setcolor(0);
    line(237+i*48,421,237+i*48,448);
    line(195+i*48,448,237+i*48,448);
}
setcolor(0);
    line(l,t+11*28,r,t+11*28);
    line(l,t+11*28,l,b+11*28);
    setcolor(15);
    line(l,b+11*28,r,b+11*28);
    line(r,t+11*28,r,b+11*28);
setcolor(0);
setlinestyle(1,1,2);
rectangle(198,424,233,445);
outtextxy(248,431,"COPY");
outtextxy(301,431,"CUT");
outtextxy(341,431,"PASTE");
outtextxy(394,431,"CLIP");
setlinestyle(0,1,0);
setcolor(0);
// spray
spray_icon();
//pencil
pencil();
    //line icon
setcolor(0);
line(15,52,43,69);
    //rectangle icon
rectangle(15,80,43,98);
    //circle icon
circle(29,117,9);
circle(29,117,10);
    //translate icon
translate_icon();
    //ellipse icon
ellipse(29,145,0,360,15,10);
```

RVCE

```
    //fill icon
    fill_icon();
    //text
    text_icon();
    //open file
    open_icon();
    //save icon
    save_icon();
    //eraser icon
    eraser_icon();
    //spiral icon
    screen_spiral(29,230,12); //x,y,rad
    clear_icon();
}

void draw_screen()
{
    int i;
    setcolor(0);
    setfillstyle(SOLID_FILL,BLUE);
    bar(0,0,639,20); //box1
    setcolor(0);
    rectangle(0,0,639,20);
    setfillstyle(SOLID_FILL,4);
    bar(0,21,639,39); //box2
    setcolor(0);
    rectangle(0,20,639,19);
    setfillstyle(SOLID_FILL,15);
    bar(59,40,639,409); //box3
    setcolor(0);
    rectangle(59,40,639,409);
    setfillstyle(SOLID_FILL,7);
    bar(0,40,59,409); //box4
    setcolor(0);
    rectangle(0,40,59,409);
    setfillstyle(SOLID_FILL,RED);
    bar(0,410,639,459); //box5
    setcolor(0);
    rectangle(0,410,639,459);
    setfillstyle(SOLID_FILL,15);
    bar(9,419,39,449); //box7
    setcolor(0);
    rectangle(9,419,39,449);
    setfillstyle(SOLID_FILL,0);
    bar(14,424,34,444); //box8
    setcolor(0);
    rectangle(14,424,34,444);
    setcolor(0);
    rectangle(14,424,34,444);
    setfillstyle(SOLID_FILL,BLUE);
    bar(0,460,639,479); //box9 ;
```

RVCE

```
    draw_icon();
    setfillstyle(SOLID_FILL,15);
    bar(52,417,182,451);
    setcolor(0);
    for(i=0;i<8;i++)
    { setfillstyle(SOLID_FILL,i);
      bar(54+16*i,419,68+16*i,433); //box10 palette
      setcolor(0);
      rectangle(54+16*i,419,68+16*i,433);
      setfillstyle(SOLID_FILL,i+8);
      bar(54+16*i,435,68+16*i,449);
      setcolor(0);
      rectangle(54+16*i,435,68+16*i,449);
    }
    setcolor(0);
    rectangle(52,417,182,451);
    setfillstyle(SOLID_FILL,15);
    bar(622,1,639,18); // exit
    setcolor(0);
    line(622,1,639,18); //exit
    line(639,1,622,18); //exit
    setcolor(15);
    outtextxy(5,463,"SCRATCH TOOL");
    setcolor(15);
    settextstyle(1,0,1); outtextxy(235,0,"GRAPHICS EDITOR");

}

int check(int x,int y)
{
    if((x>59 && x<640) && (y>40 && y<409))return 1;
    else
    return 0;
}

void show_cord()
{
    int x,y,button1;
    static int prevx=0,prevy=0;
    char str[4];
    getmouseptr(&button1,&x,&y);
    if(!(prevx==x&&prevy==y))
    { setcolor(10);
      settextstyle(0,HORIZ_DIR,1);
      setfillstyle(SOLID_FILL,RED);
      bar(585,430,637,437);
      if(check(x,y))
      {outtextxy(585,430,itoa(x-60,str,10));
        outtextxy(609,430,",");
        outtextxy(615,430,itoa(y-41,str,10));
        prevx=x,prevy=y;
      }
    }
}
```

RVCE

```
    }
  }
}

void clears()
{
  setfillstyle(SOLID_FILL,15);
  bar(60,41,638,408);
}

void show_sel_col()
{
  setcolor(0);
  rectangle(9,419,39,449); //box7
  setfillstyle(SOLID_FILL,bkcol);
  floodfill(10,420,0);
  setfillstyle(SOLID_FILL,fcolor);
  bar(14,424,34,444); //box8
  setcolor(0);
  rectangle(14,424,34,444);
  rectangle(9,419,39,449);
}

void regioncheck(int x,int y)
{
  int l,t,r,b;
  l=10,t=48,r=48,b=73;
  if((x>10 && x<48) && (y>356 && y<381)){old=tool_id;tool_id=1;}
  if((x>10 && x<48) && (y>48 && y<73)){old=tool_id;tool_id=2;}
  if((x>10 && x<48) && (y>132 && y<157)){old=tool_id;tool_id=3;}
  if((x>10 && x<48) && (y>76 && y<101)){old=tool_id;tool_id=4;}
  if((x>10 && x<48) && (y>104 && y<129)){old=tool_id;tool_id=5;}
  if((x>10 && x<48) && (y>188 && y<213)){old=tool_id;tool_id=6;}
  if((x>10 && x<48) && (y>300 && y<325)){old=tool_id;tool_id=7;}
  if((x>10 && x<48) && (y>272 && y<297)){old=tool_id;tool_id=8;}
  if((x>10 && x<48) && (y>160 && y<185)){old=tool_id;tool_id=9;}
  if((x>10 && x<48) && (y>244 && y<269)){old=tool_id;tool_id=10;}
  if((x>10 && x<48) && (y>328 && y<353)){old=tool_id;tool_id=11;}
  if((x>10 && x<48) && (y>216 && y<241)){old=tool_id;tool_id=12;}
  if((x>195 && x<237) && (y>421 && y<448)){old=tool_id;tool_id=13;}
  if((x>243 && x<285) && (y>421 && y<448)){old=tool_id;tool_id=14;}
  if((x>291 && x<333) && (y>421 && y<448)){old=tool_id;tool_id=15;}
  if((x>339 && x<381) && (y>421 && y<448)){old=tool_id;tool_id=16;}
  if((x>387 && x<429) && (y>421 && y<448)){old=tool_id;tool_id=17;}
  if((x>621 && x<639) && (y>1 && y<19)){ closegraph();exit(1);}
  if((x>569 && x<599) && (y>21 && y<40)){old=tool_id;
display_box(220,165,420,215,1),tool_id=1;}
  if((x>608 && x<626) && (y>21 && y<39)){old=tool_id;
display_box(220,165,420,215,0),tool_id=1;}
  /*pencil,line,ellipse,rectang,circle,clip,spray,fill,eraser,text*/
  //chk for color palette
  if((x>=54 && x<68) && (y>419 && y<433))fcolor=0;
```

RVCE

```
if((x>=70 && x<84)&& (y>419 && y<433) )fcolor=1;
if((x>=86 && x<100)&& (y>419 && y<433) )fcolor=2;
if((x>=102 && x<116)&& (y>419 && y<433) )fcolor=3;
if((x>=118 && x<132)&& (y>419 && y<433) )fcolor=4;
if((x>=134 && x<148)&& (y>419 && y<433) )fcolor=5;
if((x>=150 && x<164)&& (y>419 && y<433) )fcolor=6;
if((x>=166 && x<180)&& (y>419 && y<433) )fcolor=7;
if((x>=54 && x<68)&& (y>435&& y<449))fcolor=8;
if((x>=70 && x<84) && (y>435 && y<449))fcolor=9;
if((x>=86 && x<100) && (y>435 && y<449))fcolor=10;
if((x>=102 && x<116) && (y>435 && y<449))fcolor=11;
if((x>=118 && x<132) && (y>435 && y<449))fcolor=12;
if((x>=134 && x<148) && (y>435 && y<449))fcolor=13;
if((x>=150 && x<164) && (y>435 && y<449))fcolor=14;
if((x>=166 && x<180) && (y>435 && y<449))fcolor=15;

hidemouseptr();
switch(old)
{
  case 1 :setcolor(15);
    line(l,t+11*28,r,t+11*28);
    line(l,t+11*28,l,b+11*28);
    setcolor(0);
    line(l,b+11*28,r,b+11*28);
    line(r,t+11*28,r,b+11*28);
    break;
  case 2 :setcolor(15);
    line(l,t,r,t);
    line(l,t,l,b);
    setcolor(0);
    line(l,b,r,b);
    line(r,t,r,b);
    break;
  case 3 :setcolor(15);
    line(l,t+3*28,r,t+3*28);
    line(l,t+3*28,l,b+3*28);
    setcolor(0);
    line(l,b+3*28,r,b+3*28);
    line(r,t+3*28,r,b+3*28);
    break;
  case 4 :setcolor(15);
    line(l,t+28,r,t+28);
    line(l,t+28,l,b+28);
    setcolor(0);
    line(l,b+28,r,b+28);
    line(r,t+28,r,b+28);
    break;
  case 5 :setcolor(15);
    line(l,t+2*28,r,t+2*28);
    line(l,t+2*28,l,b+2*28);
    setcolor(0);
```

RVCE

```
    line(l,b+2*28,r,b+2*28);
    line(r,t+2*28,r,b+2*28);
    break;
case 6 :setcolor(15);
    line(l,t+5*28,r,t+5*28);
    line(l,t+5*28,l,b+5*28);
    setcolor(0);
    line(l,b+5*28,r,b+5*28);
    line(r,t+5*28,r,b+5*28);
    break;
case 7 :setcolor(15);
    line(l,t+9*28,r,t+9*28);
    line(l,t+9*28,l,b+9*28);
    setcolor(0);
    line(l,b+9*28,r,b+9*28);
    line(r,t+9*28,r,b+9*28);
    break;
case 8 :setcolor(15);
    line(l,t+8*28,r,t+8*28);
    line(l,t+8*28,l,b+8*28);
    setcolor(0);
    line(l,b+8*28,r,b+8*28);
    line(r,t+8*28,r,b+8*28);
    break;
case 9 :setcolor(15);
    line(l,t+4*28,r,t+4*28);
    line(l,t+4*28,l,b+4*28);
    setcolor(0);
    line(l,b+4*28,r,b+4*28);
    line(r,t+4*28,r,b+4*28);
    break;
case 10:setcolor(15);
    line(l,t+7*28,r,t+7*28);
    line(l,t+7*28,l,b+7*28);
    setcolor(0);
    line(l,b+7*28,r,b+7*28);
    line(r,t+7*28,r,b+7*28);
    break;
case 11:setcolor(15);
    line(l,t+10*28,r,t+10*28);
    line(l,t+10*28,l,b+10*28);
    setcolor(0);
    line(l,b+10*28,r,b+10*28);
    line(r,t+10*28,r,b+10*28);
    break;
case 12:setcolor(15);
    line(l,t+6*28,r,t+6*28);
    line(l,t+6*28,l,b+6*28);
    setcolor(0);
    line(l,b+6*28,r,b+6*28);
    line(r,t+6*28,r,b+6*28);
```

RVCE

```
    break;
case 13:setcolor(15);
    line(195,421,237,421);
    line(195,421,195,448);
    setcolor(0);
    line(237,421,237,448);
    line(195,448,237,448);
    break;
case 14:setcolor(15);
    line(195+1*48,421,237+1*48,421);
    line(195+1*48,421,195+1*48,448);
    setcolor(0);
    line(237+1*48,421,237+1*48,448);
    line(195+1*48,448,237+1*48,448);
    break;
case 15:setcolor(15);
    line(195+2*48,421,237+2*48,421);
    line(195+2*48,421,195+2*48,448);
    setcolor(0);
    line(237+2*48,421,237+2*48,448);
    line(195+2*48,448,237+2*48,448);
    break;
case 16:setcolor(15);
    line(195+3*48,421,237+3*48,421);
    line(195+3*48,421,195+3*48,448);
    setcolor(0);
    line(237+3*48,421,237+3*48,448);
    line(195+3*48,448,237+3*48,448);
    break;
case 17:setcolor(15);
    line(195+4*48,421,237+4*48,421);
    line(195+4*48,421,195+4*48,448);
    setcolor(0);
    line(237+4*48,421,237+4*48,448);
    line(195+4*48,448,237+4*48,448);
    break;
}
setcolor(15);
    line(l,t+7*28,r,t+7*28);
    line(l,t+7*28,l,b+7*28);
    setcolor(0);
    line(l,b+7*28,r,b+7*28);
    line(r,t+7*28,r,b+7*28);
setcolor(15);
    line(l,t+5*28,r,t+5*28);
    line(l,t+5*28,l,b+5*28);
    setcolor(0);
    line(l,b+5*28,r,b+5*28);
    line(r,t+5*28,r,b+5*28);
showmouseptr();
hidemouseptr();
```

RVCE

```
switch(tool_id)
{
case 1 :setcolor(0);
    line(l,t+11*28,r,t+11*28);
    line(l,t+11*28,l,b+11*28);
    setcolor(15);
    line(l,b+11*28,r,b+11*28);
    line(r,t+11*28,r,b+11*28);
    break;
case 2 :setcolor(0);
    line(l,t,r,t);
    line(l,t,l,b);
    setcolor(15);
    line(l,b,r,b);
    line(r,t,r,b);
    break;
case 3 :setcolor(0);
    line(l,t+3*28,r,t+3*28);
    line(l,t+3*28,l,b+3*28);
    setcolor(15);
    line(l,b+3*28,r,b+3*28);
    line(r,t+3*28,r,b+3*28);
    break;
case 4 :setcolor(0);
    line(l,t+28,r,t+28);
    line(l,t+28,l,b+28);
    setcolor(15);
    line(l,b+28,r,b+28);
    line(r,t+28,r,b+28);
    break;
case 5 :setcolor(0);
    line(l,t+2*28,r,t+2*28);
    line(l,t+2*28,l,b+2*28);
    setcolor(15);
    line(l,b+2*28,r,b+2*28);
    line(r,t+2*28,r,b+2*28);
    break;
case 6 :setcolor(0);
    line(l,t+5*28,r,t+5*28);
    line(l,t+5*28,l,b+5*28);
    setcolor(15);
    line(l,b+5*28,r,b+5*28);
    line(r,t+5*28,r,b+5*28);
    break;
case 7 :setcolor(0);
    line(l,t+9*28,r,t+9*28);
    line(l,t+9*28,l,b+9*28);
    setcolor(15);
    line(l,b+9*28,r,b+9*28);
    line(r,t+9*28,r,b+9*28);
    break;
}
```

```
case 8 :setcolor(0);
  line(l,t+8*28,r,t+8*28);
  line(l,t+8*28,l,b+8*28);
  setcolor(15);
  line(l,b+8*28,r,b+8*28);
  line(r,t+8*28,r,b+8*28);
  break;
case 9 :setcolor(0);
  line(l,t+4*28,r,t+4*28);
  line(l,t+4*28,l,b+4*28);
  setcolor(15);
  line(l,b+4*28,r,b+4*28);
  line(r,t+4*28,r,b+4*28);
  break;
case 10:setcolor(0);
  line(l,t+7*28,r,t+7*28);
  line(l,t+7*28,l,b+7*28);
  setcolor(15);
  line(l,b+7*28,r,b+7*28);
  line(r,t+7*28,r,b+7*28);
  break;
case 11:setcolor(0);
  line(l,t+10*28,r,t+10*28);
  line(l,t+10*28,l,b+10*28);
  setcolor(15);
  line(l,b+10*28,r,b+10*28);
  line(r,t+10*28,r,b+10*28);
  break;
case 12:setcolor(0);
  line(l,t+6*28,r,t+6*28);
  line(l,t+6*28,l,b+6*28);
  setcolor(15);
  line(l,b+6*28,r,b+6*28);
  line(r,t+6*28,r,b+6*28);
  break;
case 13:setcolor(0);
  line(195,421,237,421);
  line(195,421,195,448);
  setcolor(15);
  line(237,422,237,448);
  line(196,448,237,448);
  break;
case 14:setcolor(0);
  line(195+1*48,421,237+1*48,421);
  line(195+1*48,421,195+1*48,448);
  setcolor(15);
  line(237+1*48,422,237+1*48,448);
  line(196+1*48,448,237+1*48,448);
  break;
case 15:setcolor(0);
  line(195+2*48,421,237+2*48,421);
```

RVCE

```
        line(195+2*48,421,195+2*48,448);
        setcolor(15);
        line(237+2*48,422,237+2*48,448);
        line(196+2*48,448,237+2*48,448);
        break;
    case 16:setcolor(0);
        line(195+3*48,421,237+3*48,421);
        line(195+3*48,421,195+3*48,448);
        setcolor(15);
        line(237+3*48,422,237+3*48,448);
        line(196+3*48,448,237+3*48,448);
        break;
    case 17:setcolor(0);
        line(195+4*48,421,237+4*48,421);
        line(195+4*48,421,195+4*48,448);
        setcolor(15);
        line(237+4*48,422,237+4*48,448);
        line(196+4*48,448,237+4*48,448);
        break;
    }
    if (selected!=0 && old==13 && /*tool_id!=14 && tool_id!=15 &&*/
tool_id!=13)
    {
        hidemouseptr();
        dottedrect(selx1,sely1,selx2,sely2,WHITE,XOR);
        selected=0;
    }
    showmouseptr();
}

void bkgndchk(int x,int y)
{
    if((x>=54 && x<68) && (y>419 && y<433)){bkcol=0;show_sel_col(); }
    if((x>=70 && x<84)&& (y>419 && y<433) ){bkcol=1;show_sel_col(); }
    if((x>=86 && x<100)&& (y>419 && y<433) ){bkcol=2;show_sel_col(); }
    if((x>=102 && x<116)&& (y>419 && y<433) ){bkcol=3;show_sel_col(); }
    if((x>=118 && x<132)&& (y>419 && y<433) ){bkcol=4;show_sel_col(); }
    if((x>=134 && x<148)&& (y>419 && y<433) ){bkcol=5;show_sel_col(); }
    if((x>=150 && x<164)&& (y>419 && y<433) ){bkcol=6;show_sel_col(); }
    if((x>=166 && x<180)&& (y>419 && y<433) ){bkcol=7;show_sel_col(); }
    if((x>=54 && x<68) && (y>435 && y<449)){bkcol=8; show_sel_col(); }
    if((x>=70 && x<84) && (y>435 && y<449)){bkcol=9; show_sel_col(); }
    if((x>=86 && x<100)&& (y>435 && y<449)){bkcol=10;show_sel_col(); }
    if((x>=102 && x<116)&& (y>435 && y<449)){bkcol=11;show_sel_col(); }
    if((x>=118 && x<132)&& (y>435 && y<449)){bkcol=12;show_sel_col(); }
    if((x>=134 && x<148)&& (y>435 && y<449)){bkcol=13;show_sel_col(); }
    if((x>=150 && x<164)&& (y>435 && y<449)){bkcol=14;show_sel_col(); }
    if((x>=166 && x<180)&& (y>435 && y<449)){bkcol=15;show_sel_col(); }
}
```

RVCE

```
void main()
{
    int gd, gm, i, maxx, maxy, button1, x, y;
    gd=DETECT;
    initgraph(&gd, &gm, "e:\\lang\\tcpp\\bgi");
    draw_screen();
    maxx=getmaxx();
    maxy=getmaxy();
    if(initmouse()==0)
    {
        closegraph();
        printf("no mouse drivers initialised\n");
        getch();
        exit(1);
    }
    restrictmouseptr(1, 3, maxx-1, maxy-1);
    while(!kbhit())
    {
        show_cord();
        showmouseptr();
        getmouseptr(&button1, &x, &y);

        if(button1==1)
        {
            regioncheck(x, y);
            show_sel_col();
            runproc();
        }
        if(button1==2)
        {
            bkgndchk(x, y);
            show_sel_col();
            runproc();
        }
    }
    closegraph();
    getch();
}
```

MOUSE.C

```
int initmouse()
{
    in.x.ax=0;
    int86(0x33,&in,&out);
    return(out.x.ax);
}

void showmouseptr()
{
    in.x.ax=1;
    int86(0x33,&in,&out);
}

void hidemouseptr()
{
    in.x.ax=2;
    int86(0x33,&in,&out);
}

void restrictmouseptr(int x1,int y1,int x2,int y2)
{
    in.x.ax=7;
    in.x.cx=x1;
    in.x.dx=x2;
    int86(0x33,&in,&out);
    in.x.ax=8;
    in.x.cx=y1;
    in.x.dx=y2;
    int86(0x33,&in,&out);
}

void getmouseptr(int *button,int *x,int *y)
{
    in.x.ax=3;
    int86(0x33,&in,&out);
    *button=out.x.bx;
    *x=out.x.cx;
    *y=out.x.dx;
}
```

SHEAR.C

```
void y_shear( int sh )
{
    int i, j, y=0 ;
    for( j=60 ; j<639 ; j++ )
    {
        if( y < 290 )
            y = (j-60) * tan( sh * ( PI / 180 ) ) ;
        for( i=41 ; i<409 ; i++ )
        {
            if( i+y > 409)putpixel( j, i, 15 ) ;
            else
                if(check(j,i+y))
                    putpixel( j, i, getpixel( j, i+y ) ) ;
        }
    }
}

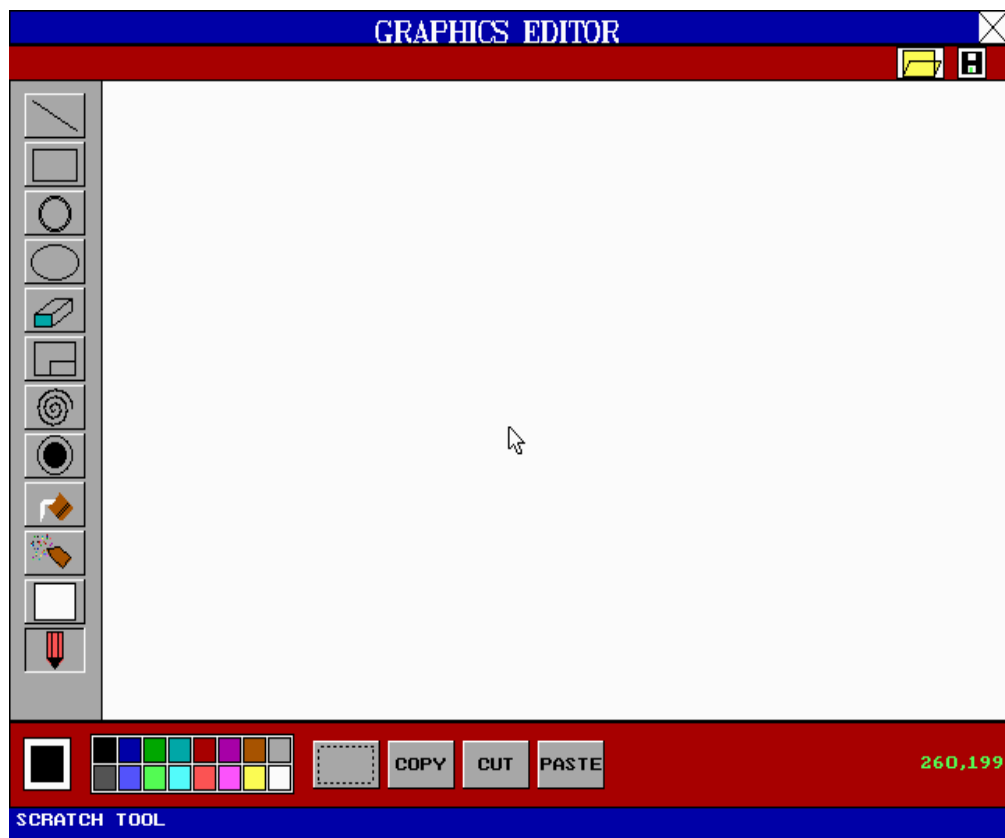
void x_shear( int sh )
{
    int i, j, x=0 ;
    for( i=409 ; i>40 ; i-- )
    {
        if( x <= 585 )
            x = ( 409 - i ) * tan( sh * ( PI / 180 ) ) ;
        for( j=638 ; j>59 ; j-- )
        {
            if( j-x < 61 ) putpixel( j, i,15 ) ;
            else
                if(check(j-x,i))
                    putpixel( j, i, getpixel( j-x, i ) ) ;
        }
    }
}
```

TESTING

The Graphics Package has been completely tested on the DOS Operating System Platform. The implementation of the package makes it dependent on the existence of a fully functional DOS OS layer.

During the process of testing, it was found that the package was able to provide good graphics processing capabilities when working with primitives of reasonable size and number. In general, the speed of the rendering of the images on the user screen was found to be dependent on the availability of essential system resources, like CPU Processing Time, Free Memory etc.

Below is the user screen as seen by the user of the graphics package.



CONCLUSION

The Graphics Editor has been designed, implemented and tested on the DOS Operating System platform. It can be used to draw images like lines, rectangles, squares, circles, ellipses, spiral, etc. It also includes saving and loading of images.

Advanced graphics manipulation operations like clipping, shearing etc have been made available to enable the user to perform a variety of objects on the graphics primitives.

The graphics creation and manipulation algorithms in the package have been implemented and tested for efficiency of operation, and were found to be satisfactory.

The graphics editor has a good, intuitive and user-friendly interface, which enables the user to get accustomed to the software very soon, and derive maximum performance from it.

BIBLIOGRAPHY

The following references provided essential domain concepts and fundamentals required for implementing the Graphics Package.

- Computer Graphics – Principles and Practice
 - James D. Foley
 - Andries van Dam
 - Steven K. Feiner
 - John F. Hughes

- Graphics Programming in C
 - Richard T. Stevens

THE END
