**2011**

Roll No:

Registration No :

M.SC. 3RD SEMESTER EXAMINATION

2/2/2011

MULTIMEDIA ASSIGNMENT



**ASSIGNMENT 1:**

**Problem:**

Write a C Program to draw some basic shapes such as Lines, Rectangle, Ellipse, Sector, and Polygon.

**Code:**

#include<graphics.h>

#include<stdlib.h>

int xMax,yMax;

void menu();

void choice(int);

void screenInitialize();

void drawCoordinates(char []);

void menu() //this function provides an interface to draw the basic shapes.

{

int c;

do

{

printf("\t\t\t\tDifferent Shapes to Draw\n");

printf("\t\t\t\t\t1.Line\n");

printf("\t\t\t\t\t2.Square\n");

printf("\t\t\t\t\t3.Rectangle\n");

printf("\t\t\t\t\t4.Ellipse\n");

printf("\t\t\t\t\t5.Circle\n");

printf("\t\t\t\t\t6.Sector\n");

printf("\t\t\t\t\t7.Polygon\n");

printf("\t\t\t\t\t8.Exit\n");

printf("\t\t\tEnter choice(1/2/3/4/5/6/7/8) :- ");

scanf("%d",&c);

choice(c);

}while(1);

}

void choice(int c)

{

int x1,y1,x2,y2,a,b,r,sang,eang;

int poly[12]={350,450, 350,410, 430,400, 350,350, 300,430, 350,450 };

//this is a closed polygon where n=5.Hence the last end point co-ordinate is

//same as the first end point co-ordinate.

switch(c)

{ case 1: //draws a line

printf("Line\n");

printf("Enter Co-ordinates :- \n");

printf("Enter 1st point :- \n");

printf("X1 = "); //the first end point co-ordinate values

scanf("%d",&x1);

printf("Y1 = ");

scanf("%d",&y1);

printf("Enter 2nd point :- \n");

printf("X2 = "); //the 2nd end point co-ordinate values

scanf("%d",&x2);

printf("Y2 = ");

scanf("%d",&y2);

screenInitialize(); //initializes the screen

drawCoordinates("Line");

line(xMax/2+x1,yMax/2-y1,xMax/2+x2,yMax/2-y2); //a line is drawn

break;

case 2: //draws a square.

printf("Square\n");

printf("Enter Co-ordinates :- \n");

printf("Enter point :- \n");

printf("X1 = "); //takes the co-ordinate values

scanf("%d",&x1);

printf("Y1 = ");

scanf("%d",&y1);

printf("Enter Length of sides :- ");//takes the length of the side

scanf("%d",&a);

screenInitialize();

drawCoordinates("Square");

rectangle(xMax/2+x1,yMax/2-y1,xMax/2+x1+a,yMax/2-y1+a);

break;

case 3: //draws a rectangle.

printf("Rectangle\n");

printf("Enter Co-ordinates :- \n");

printf("Enter point :- \n");

printf("X1 = ");

scanf("%d",&x1);

printf("Y1 = ");

scanf("%d",&y1);

printf("Enter Length :- ");

scanf("%d",&a);

printf("Enter Breath :- ");

scanf("%d",&b);

screenInitialize();

drawCoordinates("Rectangle");

rectangle(xMax/2+x1,yMax/2-y1,xMax/2+x1+a,yMax/2-y1+b);

break;

case 4: //draws an ellipse.

printf("Ellipse\n");

printf("Enter Co-ordinates of Center:- \n");

printf("Enter point :- \n");

printf("X1 = ");

scanf("%d",&x1);

printf("Y1 = ");

scanf("%d",&y1);

printf("Enter X-Radian :- ");

scanf("%d",&a);

printf("Enter Y-Radian :- ");

scanf("%d",&b);

screenInitialize();

drawCoordinates("Ellipse");

ellipse(xMax/2+x1,yMax/2-y1,0,360,a,b);

break;

case 5: //draws a circle

printf("Circle\n");

printf("Enter Co-ordinates of Center:- \n");

printf("Enter point :- \n");

printf("X1 = ");

scanf("%d",&x1);

printf("Y1 = ");

scanf("%d",&y1);

printf("Enter Radius :- ");

scanf("%d",&r);

screenInitialize();

drawCoordinates("Circle");

circle(xMax/2+x1,yMax/2-y1,r);

break;

case 6: //draws a sector

printf("Sector\n");

printf("Enter Co-ordinates of Center:- \n");

printf("Enter point :- \n");

printf("X1 = ");

scanf("%d",&x1);

printf("Y1 = ");

scanf("%d",&y1);

printf("Enter Start-Angle :- ");

scanf("%d",&sang);

printf("Enter End-Angle :- ");

scanf("%d",&eang);

printf("Enter X-Radian :- ");

scanf("%d",&a);

printf("Enter Y-Radian :- ");

scanf("%d",&b);

screenInitialize();

drawCoordinates("Sector");

sector(xMax/2+x1,yMax/2-y1,sang,eang,a,b);

break;

case 7: //draws a 5 sided polygon.

printf("Polygon\n");

screenInitialize();

drawCoordinates("Polygon");

drawpoly(6, poly);

break;

case 8: //to exit the main menu.

printf("\nExiting");

cleardevice();

closegraph();

exit(0);

default:

printf("\aWrong Choice-");

delay(200);

printf("\aEnter a valid choice\n");

break;

}

getch();

clrscr(); //clears text mode window.

cleardevice(); //clears the graphics screen.

}

void screenInitialize()

{

int gdriver = DETECT, gmode, errorcode; //initializes the graphics driver

initgraph(&gdriver, &gmode, "C:\\tc\\bgi");//initializes the graphics system

outport(0x0378,0x00);

errorcode = graphresult(); //returns an error code of last unsuccessful graphics

//operation

if (errorcode != grOk)

{

printf("Graphics error: %s\n",grapherrormsg(errorcode));

printf("Press any key to halt:");

getch();

exit(0);

}

xMax=getmaxx(); //get max screen X-coordinate.

yMax=getmaxy(); //get max screen Y-co-ordinate.

}

void drawCoordinates(char a[]) //it draws the co-ordinate graph on which the shape

{ //is drawn

char msg[80];

setcolor(20); //sets the colors

sprintf(msg,a); //prints the message sent as parameter a

outtextxy(xMax/2-textwidth(msg)/2,0,msg);

setcolor(255);

line(4,yMax/2,xMax-4,yMax/2);

line(xMax/2,textheight(msg),xMax/2,yMax-2\*textheight(msg));

outtextxy(9,yMax/2+5,"X");

gotoxy(3,16);

printf("'");

outtextxy(xMax-textwidth("X")-9,yMax/2+5,"X");//displays a string in viewport.

outtextxy(xMax/2,textheight(msg)+5,"Y");

outtextxy(xMax/2,yMax-3\*textheight("Y")-4,"Y");

gotoxy(42,29);

printf("'");

setcolor(50);

sprintf(msg,"Press any Key to Continue");

outtextxy(430,470,msg);

setcolor(255);

rectangle(4,textheight(msg),xMax-4,yMax-2\*textheight(msg)+1);

setcolor(random(255));

}

int main()

{

clrscr();

menu();

getch();

return 0;

}

**ASSIGNMENT 2:**

**Problem:**

Write a program with colors, pixels, bars, and cleardevice using random number generation. We have a function random (no), it returns a random number between 0 and no. The effect is by drawing random radius, random color circles with same center and random pixels. khbit() function (defined in conio.h) returns a nonzero value when a key is pressed in the keyboard. So, the loop will continue until a key is pressed.

**Code:**

#include<graphics.h>

#include<stdlib.h>

int xMax,yMax;

void screenInitialize();

void draw();

void screenInitialize()

{

int gdriver = DETECT, gmode, errorcode;

initgraph(&gdriver, &gmode, "C:\\tc\\bgi"); //Initialize the graphics driver.

outport(0x0378,0x00);

errorcode = graphresult(); //Check For Error

if (errorcode != grOk)

{

printf("Graphics error: %s\n",grapherrormsg(errorcode));

printf("Press any key to halt:");

getch();

exit(0);

}

/\* Get the max x and max y coordinated of the screen \*/

xMax=getmaxx();

yMax=getmaxy();

}

void draw()

{

setcolor(0);

setfillstyle(1,12);

/\*Here we draw two boxes using bar command inside which the circles will be drawn\*/

bar(xMax/2-300,yMax/2-200,xMax/2+300,yMax/2+200);

setfillstyle(1,8);

bar(xMax/2-250,yMax/2-150,xMax/2+250,yMax/2+150);

/\*This loop will continue untill the user presses any key \*/

while(!kbhit())

{

/\*Here we draw the pixels on the screen.\*/

putpixel(random(439)+101,random(279)+101,random(16));

delay(300);

setcolor(random(16));

delay(300);

/\*Draw the circle\*/

circle(320,240,random(170));

delay(300);

}

closegraph();

}

int main()

{

screenInitialize(); //Initialize the graphics system.

draw(); //Draw the random circles.

return 0;

}

**ASSIGNMENT 3:**

**Problem:**

Given a finite number of cities (choose cities from 1 to a finite number) along with the distance of travel (distance between two cities randomly selected) between each pair of them. The aim is to find the cheapest distance of visiting all cities and returning to the start point.

**Code:**

#include<stdio.h>

#include<conio.h>

#define ALL -1

#define MAXCITIES 10

enum BOOL{FALSE,TRUE};

long \*visited;//visited nodes set here

long \*min\_circuit;//min inner circuit for given node as start node at position indexed 0

long \*ham\_circuit;//optimal circuit with length stored at position indexed 0

long min\_circuit\_length;//min circuit lenth for given start node

int n;//city count

long matrix[MAXCITIES][MAXCITIES];//nondirectional nXn symmetric matrix

//to store path distances as sourceXdestination

long INFI;// INFINITY value to be defined by user

// function resets minimum circuit for a given start node

//with setting its id at index 0 and setting furthr node ids to -1

void reset\_min\_circuit(int s\_v\_id)

{

int i;

min\_circuit[0]=s\_v\_id;

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

{

min\_circuit[i]=-1;

}

}

// marks given node id with given flag

// if id==ALL it marks all nodes with given flag

void set\_visited(int v\_id,BOOL flag)

{

int i;

if(v\_id==ALL)

{

for(i=0;i<n;i++)

{

visited[i]=flag;

}

}

else

{

visited[v\_id]=flag;

}

}

// function sets hamiltonion circuit for a given path length

//with setting it at index 0 and setting furthr nodes from current min\_circuit

void SET\_HAM\_CKT(long pl)

{

ham\_circuit[0]=pl;

for(int i=0;i<n;i++)

{

ham\_circuit[i+1]=min\_circuit[i];

}

ham\_circuit[n+1]=min\_circuit[0];

}

//function sets a valid circuit by finiding min inner path for a given

//combination start vertex and next vertex to start vertex such that

// the 2nd vertex of circuits is always s\_n\_v and start and dest node is

//always s\_v for all possible values of s\_n\_v, and then returns the

// valid circuit length for this combination

long get\_valid\_circuit(int s\_v,int s\_n\_v)

{

int next\_v,min,v\_count=1;

long path\_length=0;

min\_circuit[0]=s\_v;

min\_circuit[1]=s\_n\_v;

set\_visited(s\_n\_v,TRUE);

path\_length+=matrix[s\_v][s\_n\_v];

for(int V=s\_n\_v;v\_count<n-1;v\_count++)

{

min=INFI;

for(int i=0;i<n;i++)

{

if( matrix[V][i]<INFI && !visited[i] && matrix[V][i]<=min )

{

min=matrix[V][next\_v=i];

}

}

set\_visited(next\_v,TRUE);

V=min\_circuit[v\_count+1]=next\_v;

path\_length+=min;

}

path\_length+=matrix[min\_circuit[n-1]][s\_v];

return(path\_length);

}

void main()

{

int pathcount,i,j,source,dest;

long dist=0;

long new\_circuit\_length=INFI;

clrscr();

printf("Make sure that infinity value < sum of all path distances\nSet Infinity at (signed long):");

scanf("%ld",&INFI);

printf("Enter no. of cities(MAX:%d):",MAXCITIES);

scanf("%d",&n);

printf("Enter path count:");

scanf("%d",&pathcount);

printf("Enter paths:< source\_id destination\_id distance >\n ids varying from 0 to %d\n",n-1);

//init all matrix distances to infinity

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

matrix[i][j]=INFI;

}

}

//populate the matrix

for(i=0;i<pathcount;i++)

{

printf("[path %d]:",i);

scanf("%d %d %ld",&source,&dest,&dist);

if(source!=dest)

{

matrix[source][dest]=matrix[dest][source]=dist;

}

}

visited=new long[n];

min\_circuit=new long[n];

ham\_circuit=new long[n+2];

min\_circuit\_length=INFI;

// algorithm

//for each vertex, S\_V as a staring node

for(int S\_V\_id=0;S\_V\_id<n;S\_V\_id++)

{

//for each and non start vertex as i

for(i=0;i<n;i++)

{

//set all to unvisited

set\_visited(ALL,FALSE);

// set staring vertex as visited

set\_visited(S\_V\_id,TRUE);

//reset/init minimum circuit

reset\_min\_circuit(S\_V\_id);

// obtain circuit for combination of S\_V and i

new\_circuit\_length=get\_valid\_circuit(S\_V\_id,i);

// if newer length is less than the previously

//calculated min then set it as min and set the

//current circuit in hamiltonion circuit

if(new\_circuit\_length<=min\_circuit\_length)

{

SET\_HAM\_CKT(min\_circuit\_length=new\_circuit\_length);

}

}

}

// if any circuit found

if(min\_circuit\_length<INFI)

{

printf("\n\nMinimum circuit length is: %ld\nCircuit is:\n",min\_circuit\_length);

for(i=1;i<n+2;i++)

{

printf("<%ld> ",ham\_circuit[i]);

}

}

else

{

printf("\n\nNo hamiltonian circuit !");

}

getch();

delete []visited;

delete []min\_circuit;

delete []ham\_circuit;

}

**ASSIGNMENT 4:**

**Problem:**

Read and display an image. A *region of interest* (ROI) is a portion of an image to filter or perform some other operation on. You can define an ROI by creating a *binary mask*, which is a binary image that is the same size as the image you want to process with pixels that define the ROI set to 1 and all other pixels set to 0.

Create a Binary Mask.

You can use the createMask method of the imroi base class to create a binary mask for any type of ROI object – impoint, imline, imrect, imelipse, impoly, or imfreehand. The createMask method returns a binary image the same size as the input image, containing 1s inside the ROI and 0s everywhere else.

For example, suppose you want to filter the grayscale image I, filtering only those pixels whose values are greater than 0.5. You can create the appropriate mask with this command: BW = (I > 0.5).

**Code:**

%----------- READ THE IMAGE AND DISPLAY IT ------------%

img = imread('coins.png');

h\_im = imshow(img);

%-------FIND REGION OF INTEREST(ROI): CREATE MASK & APPLY FILTER ------%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%---------- ROI OF POINT SHAPE -----------%

e1 = impoint(gca,100,100);

BW1 = createMask(e1,h\_im);

figure, imshow(BW1);

H = fspecial('unsharp');

J1 = roifilt2(H,img,BW1);

figure, imshow(J1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%---------- ROI OF LINE SHAPE -----------%

e2 = imline(gca,[100 150;170 150]);

BW2 = createMask(e2,h\_im);

figure, imshow(BW2);

H = fspecial('unsharp');

J2 = roifilt2(H,img,BW2);

figure, imshow(J2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%---------- ROI OF RECTANGLE SHAPE -----------%

e3 = imrect(gca,[10 10 100 100]);

BW3 = createMask(e3,h\_im);

figure, imshow(BW3);

H = fspecial('unsharp');

J3 = roifilt2(H,img,BW3);

figure, imshow(J3);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%---------- ROI OF POLYGON SHAPE -----------%

c = [222 272 300 270 221 194];

r = [21 21 75 121 121 75];

e4 = impoly(gca,[67 47;67 97;121 125;167 95;167 46;121 19]);

BW4 = createMask(e4,h\_im);

figure, imshow(BW4);

H = fspecial('unsharp');

J4 = roifilt2(H,img,BW4);

figure, imshow(J4);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%---------- ROI OF ELLIPSE SHAPE -----------%

e5 = imellipse(gca,[38 30 150 100]);

BW5 = createMask(e5,h\_im);

figure, imshow(BW5);

H = fspecial('unsharp');

J5 = roifilt2(H,img,BW5);

figure, imshow(J5);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%---------- ROI OF FREEHAND SHAPE -----------%

e6 = imfreehand(gca);

% ---- wait on interactive freehand selection window-----%

pos=wait(e6);

%------double click to resume opertion after selection---%

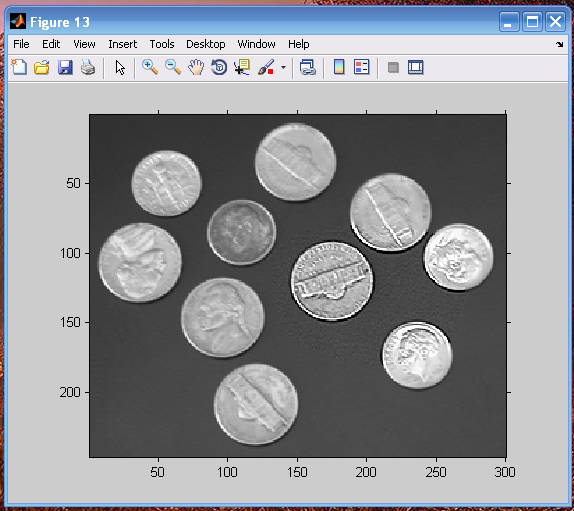
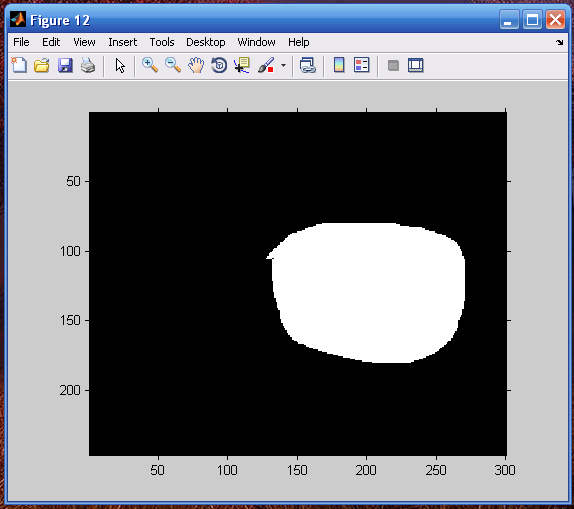
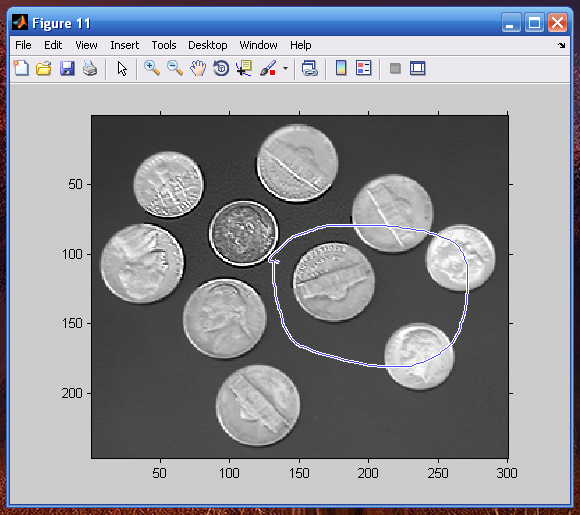
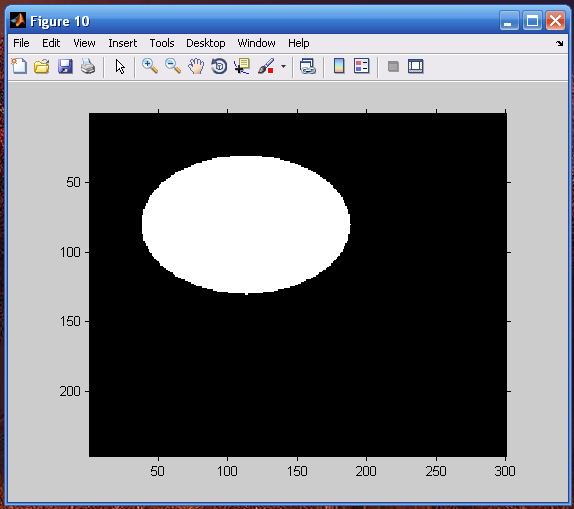
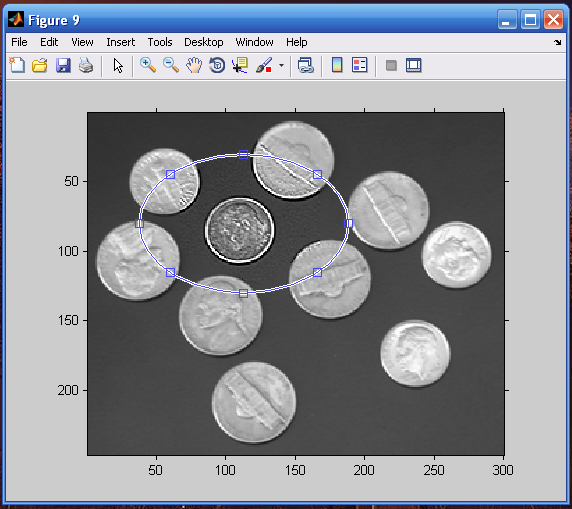
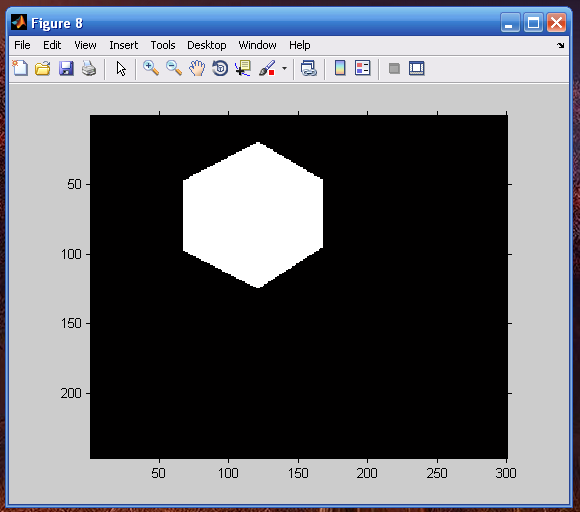
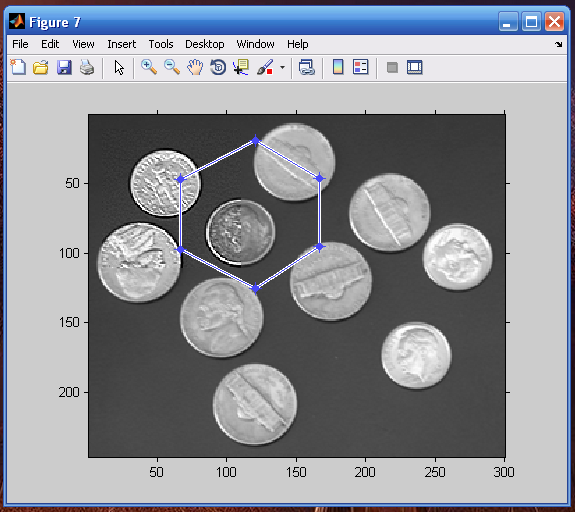
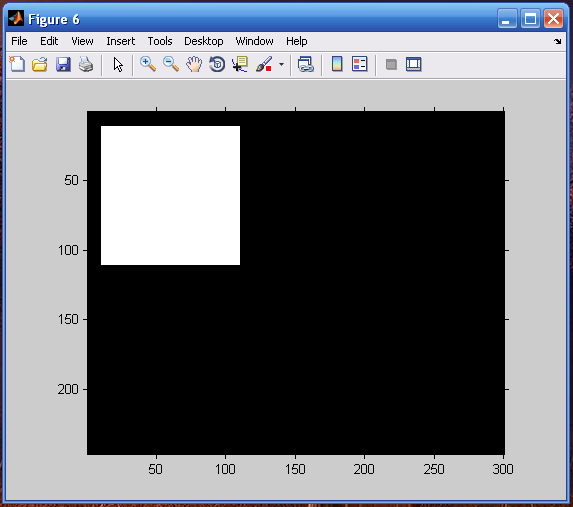
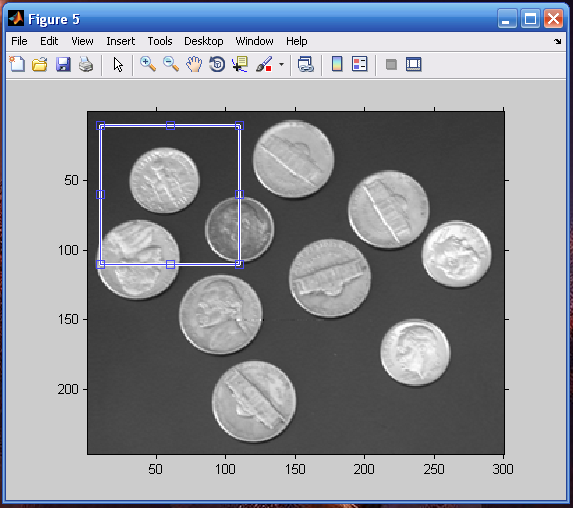
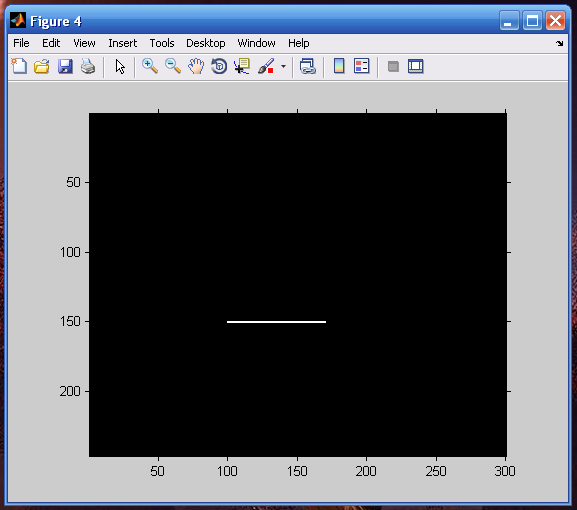
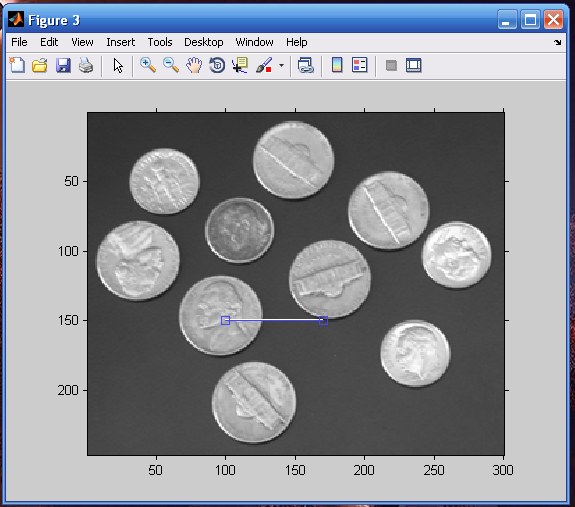
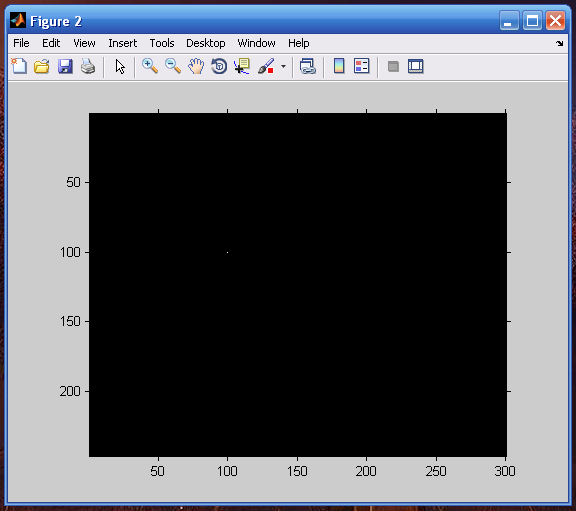
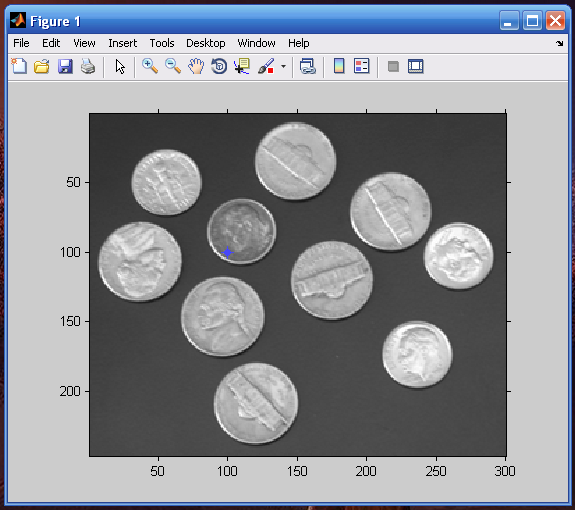
BW6 = createMask(e6,h\_im);

figure, imshow(BW6);

H = fspecial('unsharp');

J6 = roifilt2(H,img,BW6);

figure, imshow(J6);



**ASSIGNMENT 5:**

**Problem:**

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:

1. Reset mouse and get status
2. Hide mouse pointer
3. Show mouse pointer

**Code:**

#include<graphics.h>

#include<stdlib.h>

#include<stdio.h>

#include<dos.h>

union REGS in, out;

int xMax,yMax;

void detect\_mouse()

{

in.x.ax = 0;

int86 (51,&in,&out);

if (out.x.ax == 0)

{

printf("Mouse Fail To Initialize");

printf("Press any key to halt:");

getch();

exit(0);

}

}

/\*Restrict the movemrnt of the mouse within a area on screen.\*/

void restrict(int x1,int y1,int x2,int y2)

{

in.x.ax = 7;

in.x.cx = x1;

in.x.dx = x2;

int86 (51,&in,&out);

in.x.ax = 8;

in.x.cx = y1;

in.x.dx = y2;

int86 (51,&in,&out);

}

void draw()

{

char msg[80];

sprintf(msg,"Mouse Functions");

outtextxy(xMax/2-textwidth(msg)/2,0,msg);

rectangle(4,textheight(msg),xMax-4,yMax-3\*textheight(msg)+1);

restrict(4,textheight(msg),xMax-4,yMax-3\*textheight(msg)+1);

}

/\*Display The Mouse Pointer in the Middle of the Screen.\*/

void show\_mouse()

{

in.x.ax = 1;

in.x.cx=xMax/2;

in.x.dx=yMax/2;

int86 (51,&in,&out);

}

/\*Initialize the Graphics system and Look for any exceptions. \*/

void screen\_initialize()

{

int gdriver = DETECT, gmode, errorcode;

initgraph(&gdriver, &gmode, "C:\\tc\\bgi");

outport(0x0378,0x00);

errorcode = graphresult();

if (errorcode != grOk)

{

printf("Graphics error: %s\n",grapherrormsg(errorcode));

printf("Press any key to halt:");

getch();

exit(0);

}

/\* Get the MAX and MIN Coordinates of the screen .\*/

xMax=getmaxx();

yMax=getmaxy();

detect\_mouse(); /\*Check if the Mouse port is Functioning.\*/

draw(); /\*Draw Canvas and Setup Environment.\*/

show\_mouse(); /\*Display Mouse.\*/

}

/\*Hide The Mouse.\*/

void hide\_mouse()

{

in.x.ax = 2;

int86(51,&in,&out);

}

/\* Get The Status of the Mouse Position and Button Status\*/

void get\_mouse\_pos(int \*button,int \*x,int \*y)

{

in.x.ax = 3;

int86 (51,&in,&out);

\*button=out.x.bx;

\*x = out.x.cx;

\*y = out.x.dx;

}

/\* Reset the Mouse \*/

void mouse\_reset()

{

in.x.ax=0;

int86(51,&in,&out);

}

/\* Run time code for diaplaying the Mouse on screen and other details.\*/

void mouse\_on\_screen()

{

int x,y,button;

screen\_initialize(); /\*Initialize the graphics system.\*/

show\_mouse(); /\*Show the mouse pointer.\*/

/\* While the User doesn’t interrupt continue displaying the mouse on screen.\*/

while (!kbhit () )

{

get\_mouse\_pos(&button,&x,&y);

gotoxy(30,30);

printf("Co-ordinates --- X:- %d Y:- %d ",x,y);

in.x.ax = 3;

int86 (51,&in,&out);

if(out.x.bx == 1)

{

gotoxy(2,30);

printf("Left Click Occoured ");

/\*If the user Clicks LEFT Button HIDE the mouse pointer \*/

hide\_mouse();

}

else if(out.x.bx == 2)

{

gotoxy(2,30);

/\*If the user Clicks RIGHT Button UN-HIDE the mouse pointer \*/

printf("Right Click Occoured ");

show\_mouse();

}

}

clrscr();

cleardevice();

}

int main ()

{

char c;

do

{

mouse\_on\_screen(); /\*Set Up the environment for displaying the mouse.\*/

printf("\nPress E to Exit or R to Reset Mouse :");

fflush(stdin);

scanf("%c",&c);

/\*If the user presses ‘E’ then EXIT \*/

if(c == 'E' || c == 'e')

{

closegraph();

cleardevice();

exit(0);

}

/\*If the user presses ‘R’ then RESET mouse \*/

else if(c == 'R' || c == 'r') {

mouse\_reset();

continue;

}

}while(1);

return 0;

}