**Logo

Description automatically generated**

**COMPUTER GRAPHICS**

**ASSIGNMENT**

**M.C.A. – 4th SEMESTER (2021-22)**

**DEPARTEMENT OF COMPUTER SCIENCE**

**AND APPLICATIONS**

***Submitted To:*** ***Submitted By:***

Prof. Dr. R.K.Singla Tanya Singh M.C.A.II (4th Semester )

Roll. Number 42

INDEX

|  |  |  |
| --- | --- | --- |
| S No. | Programs | Page No |
| 1. | Mathematical Function -Sin Function | 4-5 |
| 2. | Circle Drawing | 6-7 |
| 3. | Mouse Programming | 8-10 |
| 4. | Clock Display | 11-13 |
| 5. | Towers of Hanoi | 14-17 |
| 6. | Sorting Algorithms | 18-21 |
| 7. | Paint System | 22-54 |
| 8. | Bresenham’s Line Drawing | 55-57 |
| 9. | Bresenham’s Circle Drawing | 58-59 |
| 10. | Bresenham’s Ellipse Drawing | 60-62 |
| 11. | Line Clipping in 2D | 63-66 |
| 12. | Transformation in 2D | 67-72 |
| 13. | View Algorithm | 73-76 |
| 14. | Dot Plot | 77-78 |
| 15. | Rubber Band or Elastic Lines | 79-82 |
| 16. | 15-Puzzle Design | 83-93 |
| 17. | Back Face Elimination Method | 94-98 |
| 18. | What is OpenGL and what is GLUT library? Name and explain the seven major groups of OpenGL API functions, with appropriate examples for each function. Also, explain the purpose of void glutDisplayFunc(void ('func)(void)); and glutMainLoop(); | 99-101 |
| 19. | Explain the working of Cohen-Sutherland Line clipping algorithm with the help of an example. How does this algorithm identify trivally-in and trivally-out lines? | 102-104 |
| 20. | What is the objective of interactive computer graphics? Develop an animated algorithm for the demonstration of Bubble Sort. | 105-111 |
| 21. | What are the various 2-dimensional transformations? Explain with the help of suitable sketches. Give their matrix representations. Summarize briefly the problem solved by the introduction of homogeneous coordinates.  Prove that use of 2-D rotation and scaling commute if S, = S, or 0 = nr for integral n. | 112-118 |
| 22. | Describe the Bresenham's Circle drawing algorithm. How does it differ from the mid-point circle- drawing algorithm? What efficiencies are achieved by this difference? You may use a diagram to aid your answer. Using the mid-point circle-drawing algorithm, draw a circle with centre as (2, 5) and radius as 6. | 119-124 |

**Program 1:[Mathematical Function] Write a program to draw the sin function. This function is given by sin(x)= , sin( ) x x π π with .]10.0,1 0.0) =1.0. 0.0−[∈x For this purpose, use the GL\_LINE\_STRIP primitive of OpenGL. The sampling that sin (step is 0.25 when x varies in the given interval. Recall that sinc(0.0)=1.0**

#include<iostream>

#include<Gl/glut.h>

#include<math.h>

const GLfloat factor = 0.2f;

void myDisplay(void)

{

GLfloat x;

glClear(GL\_COLOR\_BUFFER\_BIT);

{

glColor3f(1,1,0);

glBegin(GL\_LINES);

{

// x-axis

glVertex3f(-100.0f, 0.0f, 0.0f);

glVertex3f(100.0f, 0.0f, 0.0f);

// y-axis

glVertex3f(0.0f, -100.0f, 0.0f);

glVertex3f(0.0f, 100.0f, 0.0f);

glEnd();

// GL\_LINE\_STRIP: The adjacent vertices are considered lines. Thus, if you pass *n* vertices, you will get *n*-1 lines. If the user only specifies 1 vertex, the drawing command is ignored.

glBegin(GL\_LINE\_STRIP);

for (x = -2.0f / factor; x < 2.0f / factor; x += 0.030f)

{

glVertex2f((x \* factor) / 4, sin(3.14159 \* x) / (3.14159 \* x));

}

glEnd();

glFlush();

}

}

glFlush();

// glutSwapBuffers swaps the buffers of the *current window* if double buffered.

**//Usage**-Performs a buffer swap on the *layer in use* for the *current window*. //Specifically, glutSwapBuffers promotes the contents of the back buffer of the *layer in //use* of the *current window* to become the contents of the front buffer. The contents of //the back buffer then become undefined. The update typically takes place during the //vertical retrace of the monitor, rather than immediately after glutSwapBuffers is called.

glutSwapBuffers();

}

int main(int argc, char\* argv[])

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | GLUT\_SINGLE);

glutInitWindowPosition(100, 100);

glutInitWindowSize(640, 480);

glutCreateWindow("Sinc OpenGL first\_program");

glutDisplayFunc(&myDisplay);

glutMainLoop();

return 0;

}

Schematic

Description automatically generated with medium confidence

**Program 2: [Circle Drawing] Write a program to draw a circle centered at x y) ( c ,c with a given radius r, whose points are given by the following equations:**

**X= Cx \* rcos(theta)**

**Y=Cy\* rsin(theta)**

// Circle Drawing

#include <graphics.h>

#include <stdlib.h>

#include<stdio.h>

#define color 11

void SymmetricPlot(int xc,int yc,int x,int y)

{

putpixel(x+xc,y+yc,color);

putpixel(x+xc,-y+yc,color);

putpixel(-x+xc,-y+yc,color);

putpixel(-x+xc,y+yc,color);

putpixel(y+xc,x+yc,color);

putpixel(y+xc,-x+yc,color);

putpixel(-y+xc,-x+yc,color);

putpixel(-y+xc,x+yc,color);

}

void PolarCircle(int xc,int yc,int r)

{

int x,y,d;

x=0;

y=r;

d=3-2\*r;

SymmetricPlot(xc,yc,x,y);

while(x<=y)

{

if(d<=0)

d=d+4\*x+6;

else

{

d=d+4\*x-4\*y+10;

y=y-1;

}

x=x+1;

SymmetricPlot(xc,yc,x,y);

}

}

int main(void)

{

int gdriver = DETECT, gmode, errorcode;

int Cx,Cy,r;

initgraph(&gdriver, &gmode, "c:\\turboc3\\bgi");

errorcode = graphresult();

if (errorcode != grOk)

{

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

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

getch();

exit(1);

}

printf("Enter the values of Cx and Cy ,that is center points of circle : ");

scanf("%d%d",&Cx,&Cy);

printf("Enter the radius of circle : ");

scanf("%d",&r);

PolarCircle(Cx,Cy,r);

getch();

closegraph();

return 0;

}

Shape, circle

Description automatically generated

**Program 3: [Mouse Programming] The goal of this Exercise is to write a source code to control a mouse so as to accept graphics input. The mouse sensing interrupt on systems such as Macintosh and PC Windows operates at the highest priority level. That is, the mouse is "alive" and the cursor responds to mouse motion even when other processes are in operation. In fact, on such systems the quickest test for system failure is an unresponsive mouse. Your C/C++ Graphics program will usually assume that the necessary mouse driver has already been loaded into RAM. [A mouse driver is not bundled with Turbo-C++ Package]. This resident software is usually accessed through interrupt 33H. The mouse driver provides direct low-level control over the mouse hardware. Your C Graphics makes calls to the driver, not to the hardware. Develop a program to demonstrate the various functions of a mouse such as:**

**Detecting⎫ & initializing a mouse (Function # 0)**

**Turning on the mouse cursor (Function # 1)⎫**

**Turning off the mouse cursor (Function # 2)⎫**

**Determining mouse location⎫ & status of buttons (Function # 3)**

**Resetting the location of mouse cursor (Function # 4)⎫**

**Setting the minimum⎫ & maximum horizontal range (Function # 7)**

**Setting the minimum⎫ & maximum vertical range (Function # 8)**

#include<dos.h>

#include<conio.h>

#include<stdio.h>

#include<graphics.h>

union REGS in,out;//already defined registers

void status()

{

in.x.ax=0; // x is word register //0-mouse connected or not ie status

//bx-which button pressed left or right

int86(0x33,&in,&out);//interuppt86-port of mouse ,address of registers

}

void show()

{

in.x.ax=1;//1- pointer show

int86(0x33,&in,&out);

}

void main()

{

clrscr();

status();

show();

in.x.ax=3;// 3-mouse status bx

// kbhit() functionality is basically stand for the Keyboard Hit. This function is deals //with keyboard pressing

while(!kbhit())

{

int86(0x33,&in,&out);

//bx-0 no button pressed

//1- left pressed

//2-right pressed

switch(out.x.bx)

{

case 1:

gotoxy(33,12);

printf("LEFT BUTTON");

gotoxy(35,13);

printf("%d %d",out.x.cx,out.x.dx);

break;

case 2:

gotoxy(33,12);

printf("RIGHT BUTTON");

gotoxy(35,13);

//cx-x coordinate , dx-y coordinate

printf("%d %d",out.x.cx,out.x.dx);

break;

case 0://0-button not pressed

gotoxy(33,12);

printf("Number Pressed");

gotoxy(35,13);

printf("%d %d",out.x.cx,out.x.dx);

break;

}

delay(100);

clrscr();

}

}

Text

Description automatically generated

**Program 4: [Clock Display] Develop a program to display a clock face in the center of the screen. Draw the hands as arrows (for hours and minutes) and rotate the hands so that the smaller (hour) hand rotates through 30 degrees each time the larger (minute) hand makes one complete revolution (360 degrees). A second’s hand can be added that sweeps through 360 degrees for each 6 degrees of rotation of the minute hand.**

#include<conio.h>

#include<graphics.h>

#include<math.h>

#include<dos.h>

#define WBC 5

//^watchbackcolor

#define X 200

#define Y 200

void dial(int x, int y);

void sechand(int timeminute);

void minhand(int t)

{

int x1,y1;

setlinestyle(0,0,3);

x1= X+ (80 \* cos(t\*0.1047));

y1= Y+ (80 \* sin(t\*0.1047));

setcolor(BLACK);

line( X, Y, x1, y1);

setcolor(WBC+1);

line( X, Y, X+ 80 \* cos((t-1)\*0.1047),Y+ 80 \* sin((t-1)\*0.1047));

circle(X,Y,4);

}

void sechand(int t)

{

int x1,y1;

setlinestyle(0,0,3);

x1= X+(100 \* cos(t\*0.1047));

y1= Y+(100 \* sin(t\*0.1047));

setcolor(RED);

line(X, Y, x1, y1);

setcolor(WBC+1);

line(X, Y, X+ 100 \* cos((t-1)\*0.1047),Y+ 100 \* sin((t-1)\*0.1047));

circle(X,Y,4);

}

void dial(int x,int y)

{

int const size=200;

setfillstyle(1,WBC);

fillellipse(x,y,size,size);

setfillstyle(1,WBC+1);

fillellipse(x,y,size-20,size-20);

outtextxy(x,y-(size-40),"12");

outtextxy(x,y+(size-40),"6");

outtextxy(x+(size-40),y,"3");

outtextxy(x-(size-40),y,"9");

outtextxy(x+size/3,y-2\*size/3,"1");

outtextxy(x+2\*size/3,y-size/3,"2");

outtextxy(x+2\*size/3,y+size/3,"4");

outtextxy(x+size/3,y+2\*size/3,"5");

outtextxy(x-size/3,y+2\*size/3,"7");

outtextxy(x-2\*size/3,y+size/3,"8");

outtextxy(x-size/3,y-2\*size/3,"11");

outtextxy(x-2\*size/3,y-size/3,"10");

circle(x,y,4);

}

void main()

{

int gd=DETECT, gm,i,j, flag=1;

initgraph(&gd,&gm,"C:\\turboc3\\bgi");

dial(200,200);

do

{

minhand(i);

for(j=0;j<60;j++)

{

sechand(j);

delay(1000

);

if(kbhit()) {

flag =0;

break;

}

}

i++;

}while(flag);

closegraph();

}

Chart

Description automatically generated

**Program 5:** **[Towers of Hanoi] Solve graphically the Towers of Hanoi problem with variable number of disks.**

#include<stdio.h>

#include<graphics.h>

#include<dos.h>

#include<math.h>

#include<conio.h>

int tower[3][10];// the three towers' disks as stack

int top[3];//top of the three stacks

int from,to;//moving 'from' tower number 'to' tower number

int diskInAir;//number of disk moved (1 to n)

int l,b,u;

void push(int to, int diskno)

//putting disk on tower

{

tower[to-1][++top[to-1]]=diskno;

}

int pop(int from)

//take topmost disk from tower

{

return(tower[from-1][top[from-1]--]);

}

void drawStill()

{

int j,i,disk;

cleardevice();

for(j=1;j<=3;j++)

{

//draw tower

setfillstyle(CLOSE\_DOT\_FILL,WHITE);

bar(j\*l,u,j\*l+5,b);

//draw all disks on tower

for(i=0;i<=top[j-1];i++)

{

disk=tower[j-1][i];

setfillstyle(SOLID\_FILL,1+disk);

bar(j\*l-15-disk\*5,b-(i+1)\*10,j\*l+5+15+disk\*5,b-i\*10);

}

}

}

void animator()

//to show the movement of disk

{

int x,y,dif,sign;

diskInAir=pop(from);

x=from\*l;

y=b-(top[from-1]+1)\*10;

//taking disk upward from the tower

for(;y>u-20;y-=15)

{

drawStill();

setfillstyle(SOLID\_FILL,1+diskInAir);

bar(x-15-diskInAir\*5,y-10,x+5+15+diskInAir\*5,y);

delay(100);

}

y=u-20;

dif=to\*l-x;

sign=dif/abs(dif);

//moving disk towards a target tower

for(;abs(x-to\*l)>25;x+=sign\*15)

{

drawStill();

setfillstyle(SOLID\_FILL,1+diskInAir);

bar(x-15-diskInAir\*5,y-10,x+5+15+diskInAir\*5,y);

delay(100);

}

x=to\*l;

//placing disk on a target tower

for(;y<b-(top[to-1]+1)\*10;y+=15)

{

drawStill();

setfillstyle(SOLID\_FILL,1+diskInAir);

bar(x-15-diskInAir\*5,y-10,x+5+15+diskInAir\*5,y);

delay(100);

}

push(to,diskInAir);

drawStill();

}

void moveTopN(int n, int a, int b, int c)

//Move top n disk from tower 'a' to tower 'c'

//tower 'b' used for swapping

{

if(n>=1)

{

moveTopN(n-1,a,c,b);

drawStill();

delay(1000);

from=a;

to=c;

//animating the move

animator();

moveTopN(n-1,b,a,c);

}

}

void main()

{

int i,gd=DETECT,gm,n;

printf("Enter number of disks");

scanf("%d",&n);

initgraph(&gd,&gm,"C:\\TURBOC3\\BGI\\");

//setting all tower empty

for(i=0;i<3;i++)

{

top[i]=-1;

}

//putting all disks on tower 'a'

for(i=n;i>0;i--)

{

push(1,i);

}

l=getmaxx()/4;

b=getmaxy()-50;

u=getmaxy()/3+100;

//start solving

moveTopN(n,1,2,3);

delay(4000);

getch();

}

A screenshot of a computer

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated with medium confidence

**Program 6: Develop an animated program for demonstration of various sorting algorithms**

#include<stdio.h>

#include<conio.h>

#include<dos.h>

#include<stdlib.h>

//#include"mouseanimate.h"

#include<graphics.h>

void sort(int \*,int);

void move(int ,int,int \*);

#include<graphics.h>

#define startx 300

#define starty 50

void drawrect(int a[],int n)

{

int i = 0;

char \*str = "\0";

int s = 0;

setfillstyle(1,1);

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

{

s= starty+(i\*30);

itoa(a[i],str,10);

bar(startx,s,startx+20,s+20);

outtextxy(startx+5,s+5,str);

}

}

void move(int i,int j,int a[])

{

int y1,y2,flag,x;

char \*str1,\*str2;

flag=0;

x=startx;

y1=starty+i\*30;

y2=starty+j\*30;

while(flag!=3)

{

bar(x,y1,x+20,y1+20);

bar(x,y2,x+20,y2+20);

itoa(a[i],str1,10);

outtextxy(x+5,y1+5,str1);

itoa(a[j],str2,10);

outtextxy(x+5,y2+5,str2);

delay(10);

// bar(x,y1,x+20,y1+20);

// bar(x,y2,x+20,y2+20);

if(flag==0)

{

x++;

}

else if(flag==1)

{ y1++;

y2--;

}

else if(flag==2)

{

x--;

}

if(x>=startx+100)

{

flag=1;

}

if(y1>=starty+j\*30)

{

flag=2;

}

if(x<=startx)

{

flag=3;

}

}

bar(x,y1,x+20,y1+20);

bar(x,y2,x+20,y2+20);

itoa(a[i],str1,10);

itoa(a[j],str2,10);

outtextxy(x+5,y1+5,str1);

outtextxy(x+5,y2+5,str2);

}

int main()

{

int gd=DETECT,gm,n,a[10],i;

initgraph(&gd,&gm,"c:\\turboc3\\bgi");

printf("\n Enter the no of element");

scanf("%d",&n);

printf("\n Enter element please");

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

{

scanf("%d",&a[i]);

}

drawrect(a,n);

sort(a,n);

getch();

closegraph();

return(0);

}

void sort(int a[],int n)

{

int i,j,temp;

for(i=0;i<n-1;i++)

{

for(j=0;j<n-i-1;j++)

{

if(a[j]>a[j+1])

{

move(j,j+1,a);

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

}

}

Graphical user interface, text

Description automatically generated

**Program 7: [Paint System] Develop an application program that allows the user to act as an artist with the help of a computer. The user fashions the image by sketching -often using a mouse- and by selecting colors and patterns to create the desired effects. The application should provide many tools for the artist. For example, previously stored images can be called up from the mass storage and merged with new images; "palettes” of different colors can be accessed etc.**

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <dos.h>

#include <math.h>

#include <graphics.h>

#define PI 3.14159265

#define TRUE 1

union REGS in,out;

//mouse interrupt with functions 0, 1, 2, 3, 7

void initmouse() {

    in.x.ax=0;

    int86(0x33,&in,&out);

    }

void showmouse() {

    in.x.ax=1;

    int86(0x33,&in,&out);

    }

void hidemouse() {

    in.x.ax=2;

    int86(0x33,&in,&out);

    }

void getmousepos(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;

    }

void restrictmouse(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);

    }

//(52,30,628,418) is the drawing area with lightgray background

int posx, posy, control\_button, colx,coly, col\_button; //global variables

void selectcolour(int, int, int); //prototype

int colour, col2; //global variables in selectcolour()

//save image in picture.dat in /bin directory of Turbo-C++

void save\_image(){

    //setting impression of all button presses

    for(int k=80;k<212;k+=21) {

        setcolor(BLACK);

        rectangle(5,k,25,k+20); rectangle(26,k,46,k+20);

        setcolor(WHITE);

        line(5,k,5,k+20); line(5,k,25,k);

        line(26,k,26,k+20); line(26,k,46,k);

    }

    //giving impression of save button press

    setcolor(BLACK);

    line(5,80,5,100); line(5,80,25,80);

    setcolor(WHITE);

    line(25,100,25,80); line(25,100,5,100);

        hidemouse();

        FILE \*fp=fopen("picture.dat","wb");

        for(int x=52+1;x<=628-1;x++){

            for(int y=30+1;y<=418-1;y++)

            {

                int cr=getpixel(x,y);

                fwrite(&cr,2,1,fp);

            }

        }

        fclose(fp);

        showmouse();

} //end of save\_image()

//load image file image.dat from /bin

void load\_image() {

    //setting impression of all button press

    for(int k=80;k<212;k+=21) {

    setcolor(BLACK);

    rectangle(5,k,25,k+20); rectangle(26,k,46,k+20);

    setcolor(WHITE);

    line(5,k,5,k+20); line(5,k,25,k);

    line(26,k,26,k+20); line(26,k,46,k);

    }

    //giving impression of load button press

    setcolor(BLACK);

    line(26,80,26,100); line(26,80,46,80);

    setcolor(WHITE);

    line(46,100,46,80); line(46,100,26,100);

        FILE \*fp=fopen("picture.dat","rb");

        int cr;

        for(int x=52+1;x<=628-1;x++){

            for(int y=30+1;y<=418-1;y++)

            {

                fread(&cr,2,1,fp);

                putpixel(x,y,cr);

            }

        }

        fclose(fp);

}//end of load\_image()

//create pencil button for pencil();

void \*pb; //global

void createpb() {

    //filled lightgray rectangle for backgroud of buttons

    setcolor(LIGHTGRAY);

    rectangle(3,3,300,300);

    setfillstyle(SOLID\_FILL,LIGHTGRAY);

    floodfill(40,40,LIGHTGRAY);

    //pencil from end to before nose

    setcolor(RED);

    line(60,50,50,57); line(63,54,53,61);

    line(50,57,53,61); line(60,50,63,54);

    setfillstyle(SOLID\_FILL,RED);

    floodfill(60,52,RED);

    //nose of pencil

    setcolor(BLACK);

    line(50,57,47,62); line(47,62,53,61);

    line(60,50,50,57); line(63,54,53,61);

    line(50,57,53,61); line(60,50,63,54);

    int size=imagesize(45,48,65,65);

    pb= new int [size];

    getimage(45,48,65,65,pb);

    cleardevice();

} // end of createpb()

void \*eraser; //create eraser button for rub()

void createeraser() {

    setcolor(LIGHTGRAY);

    rectangle(10,10,200,200);

    setfillstyle(SOLID\_FILL,LIGHTGRAY);

    floodfill(50,50,LIGHTGRAY);

    setcolor(RED);

    rectangle(30,112,37,115);

    line(30,112,35,105); line(35,105,42,105);

    line(42,105,37,112); line(42,105,42,109);

    line(42,109,37,115);

    setfillstyle(SOLID\_FILL,RED);

    floodfill(33,110,RED); floodfill(40,110,RED);

    setcolor(WHITE);

    rectangle(30,112,37,115);

    setfillstyle(SOLID\_FILL,WHITE);

    floodfill(33,113,WHITE);

    int size = imagesize(28,102,45,118);

    eraser = new int [size];

    getimage(28,102,45,118,eraser);

    cleardevice();

} //end of createeraser()

//create fill button for getcolour()

void \*fill;

void createfill() {

    setcolor(LIGHTGRAY);

    rectangle(3,3,300,300);

    setfillstyle(SOLID\_FILL,LIGHTGRAY);

    floodfill(40,40,LIGHTGRAY);

    setcolor(BLACK);

    line(50,57,47,62); line(47,62,53,61);

    line(60,50,50,57); line(63,54,53,61);

    line(60,50,63,54);

    int size=imagesize(45,48,65,65);

    fill= new int [size];

    getimage(45,48,65,65,fill);

    cleardevice();

} //end of createfill()

//graphical user interface

void gui() {

    int i,x=getmaxx(),y=getmaxy();

    createpb();

    createfill();

    createeraser();

    putimage(6,102,pb,COPY\_PUT); //pencil button

    putimage(6,123,fill,COPY\_PUT); //colour filler button

    putimage(27,103,eraser,COPY\_PUT); //eraser button

    setcolor(BLUE);

    rectangle(0,0,x,15); //top blue

    setfillstyle(SOLID\_FILL,BLUE);

    floodfill(10,10,BLUE);

    //plain yellow square/sheet on top blue

    setcolor(YELLOW);

    rectangle(5,3,12,12);

    setfillstyle(SOLID\_FILL,YELLOW);

    floodfill(9,9,YELLOW);

    settextstyle(12,0,5);

    outtextxy(20,4,"Paint Simulator");

    setcolor(LIGHTGRAY);

    rectangle(0,15,x-1,30); //top

    rectangle(0,30,50,y); // left

    rectangle(50,y-60,x-1,y); // bottom

    rectangle(x-1,30,x-10,y-60); // right

    setfillstyle(SOLID\_FILL,LIGHTGRAY); // fill above borders

    floodfill(20,20,LIGHTGRAY);

    floodfill(40,60,LIGHTGRAY);

    floodfill(60,y-20,LIGHTGRAY);

    floodfill(x-5,y-100,LIGHTGRAY);

    setcolor(BLACK);

    line(0,31,50,31);

    line(50,y-61,0,y-61);

    line(x-1,31,x-10,31);

    line(x-1,y-61,x-10,y-61);

    line(0,y-15,x-1,y-15); // filler lines

    line(x-100,y-15,x-100,y);

    line(x-200,y-15,x-200,y);

    rectangle(10,y-50,37,y-23); // Outer box of selected colour

    setcolor(WHITE);

    line(37,y-23,37,y-50);

    line(37,y-23,10,y-23);

    //16 color boxes at the bottom

    setcolor(BLACK);

    int k,c=0;

    for(k=40;k<157;k+=15) {

        setcolor(c);

        rectangle(k,y-50,k+12,y-38);

        setfillstyle(SOLID\_FILL,c);

        floodfill(k+5,y-40,c);

        setcolor(BLACK);

        line(k,y-50,k,y-38);//drawing and filling of colour boxes(1-8)

        line(k,y-50,k+12,y-50);

        setcolor(WHITE);

        line(k+12,y-38,k+12,y-50); line(k+12,y-38,k,y-38);

        c++;

        }

    for(k=40;k<157;k+=15) {

        setcolor(c);

        rectangle(k,y-35,k+12,y-23);

        setfillstyle(SOLID\_FILL,c);

        floodfill(k+5,y-25,c);

        setcolor(BLACK);

        line(k,y-35,k,y-23); line(k,y-35,k+12,y-35);

        setcolor(WHITE); //drawing and filling of colour boxes(9-16)

        line(k+12,y-23,k+12,y-35); line(k+12,y-23,k,y-23);

        c++;

        }

    setcolor(WHITE);

    rectangle(52,32,x-12,y-62);

    setfillstyle(SOLID\_FILL,WHITE); //Drawing area (52,32) to (628,418)

    floodfill(200,200,WHITE);

    setcolor(BLACK);

    for(k=80;k<212;k+=21) {

        rectangle(5,k,25,k+20);

        rectangle(26,k,46,k+20);

        setcolor(WHITE);

        line(5,k,5,k+20);

        line(5,k,25,k); // Function Buttons

        line(26,k,26,k+20);

        line(26,k,46,k);

        setcolor(BLACK);

        }

    rectangle(8,k+8,43,k+68); // button option box

    setcolor(WHITE);

    line(43,k+68,8,k+68); line(43,k+68,43,k+8);

    setcolor(BLACK);

    rectangle(13,y-45,25,y-33); // selected foreground colour box

    setfillstyle(SOLID\_FILL,BLACK);

    floodfill(15,y-40,BLACK);

    setcolor(WHITE);

    line(26,y-40,32,y-40);

    line(20,y-28,20,y-32);

    line(32,y-40,32,y-28);

    line(32,y-28,20,y-28); //selected background colour box

    line(26,y-40,26,y-33);

    line(26,y-32,20,y-32);

    setfillstyle(SOLID\_FILL,WHITE);

    floodfill(30,y-37,WHITE);

    line(13,y-45,13,y-33); line(13,y-45,25,y-45);

    setcolor(BLACK);

    line(26,y-40,32,y-40); line(20,y-28,20,y-32);

    setcolor(LIGHTGRAY);

    rectangle(x-16,2,x-5,12); // close icon

    setfillstyle(SOLID\_FILL,LIGHTGRAY);

    floodfill(x-10,8,LIGHTGRAY);

    setcolor(BLACK);

    line(x-13,4,x-7,10); line(x-13,10,x-7,4);

    line(x-5,12,x-5,2); line(x-5,12,x-16,12);

    setcolor(WHITE);

    line(x-16,2,x-16,12); line(x-16,2,x-5,2);

    setcolor(BLACK);

    ellipse(15,153,0,360,8,4); //ellipse button

    rectangle(9,168,22,178); // rectangle button

    line(9,190,20,201); //line button

    // For polygon button

    line(33,127,30,136); line(30,136,39,136);

    line(33,127,38,127); line(38,127,36,132);

    line(36,132,40,132); line(40,132,39,136);

    // For Paint Brush button

    line(35,145,35,151); line(38,145,38,151);

    line(35,145,38,145); line(35,151,32,153);

    line(38,151,41,153); line(32,153,32,159);

    line(41,153,41,159); line(32,159,41,159);

    line(41,154,32,154); line(41,155,32,155);

    line(35,159,35,157); line(38,159,38,157);

    // For curved line button

    ellipse(32,173,180,0,4,2);

    ellipse(39,173,0,180,4,2);

    // For bucket button for fill

    line(32,200,40,200); line(32,200,32,193);

    line(40,200,40,193); line(32,193,40,193);

    line(36,193,38,189); line(38,193,40,189);

    line(38,189,40,190); line(37,193,39,189);

    line(32,195,40,195); line(32,196,40,196);

    // For spray can button

    line(7,223,7,215);   line(7,215,12,215);

    line(12,215,12,223); line(12,223,7,223);

    line(8,215,8,213);   line(11,215,11,213);

    line(11,213,8,213);  line(11,213,19,210);

    line(11,213,19,216); line(11,213,19,215);

    line(11,213,19,213); line(11,213,19,211);

    //For text button

    settextstyle(1,0,1);

    outtextxy(30,203,"A");

    //buttons for save and load

    setusercharsize(7,25,7,25);//more than size 1/4 and less than 1/3

    outtextxy(7, 83,"Save");

    outtextxy(28, 83,"Load");

    settextstyle(12,0,5);

    outtextxy(8,y-10,"DCSA - Panjab university");

} //end of gui()

//pencil: free hand drawing after left-clicking on pencil icon

void pencil() {

    //setting impression of all button press

    for(int k=80;k<212;k+=21) {

        setcolor(BLACK);

        rectangle(5,k,25,k+20); rectangle(26,k,46,k+20);

        setcolor(WHITE);

        line(5,k,5,k+20); line(5,k,25,k);

        line(26,k,26,k+20); line(26,k,46,k);

    }

    //giving impression of pencil button press

    setcolor(BLACK);

    line(5,101,5,121); line(5,101,25,101);

    setcolor(WHITE);

    line(25,121,25,101); line(25,121,5,121);

    int button,x,y,prevx,prevy; //local

    while(TRUE) {

        getmousepos(button,x,y);

        if((button & 1==1 && x>5 && x<46 && y>80 && y<225)||(button & 1==1 && x>625 && x<635 && y>2 && y<12)) {

            control\_button=1;

            posx=x; posy=y;

            break;

        }

        if(button && x>=40 && x<=157 && y>=430 && y<=457){

            if(button==1) col\_button=1;

            if(button==2) col\_button=2;

            colx=x; coly=y;

            selectcolour(col\_button,colx,coly);

        }

        setcolor(colour);

        if(button & 1 ==1 && x>52 && x<628 && y>32 && y<418){

            prevx=x; prevy=y;

            while((button & 1) ==1){

                restrictmouse(52,32,627,417);

                hidemouse();

                line(prevx,prevy,x,y);

                showmouse();

                prevx=x; prevy=y;

                getmousepos(button,x,y);

            }

        restrictmouse(0,0,getmaxx(),getmaxy());

        }

    }

} // end of pencil()

void getcolour(){

    for(int k=80;k<212;k+=21){

    setcolor(BLACK);

    rectangle(5,k,25,k+20); rectangle(26,k,46,k+20);

    setcolor(WHITE);

    line(5,k,5,k+20); line(5,k,25,k);

    line(26,k,26,k+20); line(26,k,46,k);

    }

    setcolor(BLACK);

    line(5,122,25,122); line(5,122,5,142);

    setcolor(WHITE);

    line(25,142,25,122); line(25,142,5,142);

    int button,x,y,prevx,prevy,gcolour,e,f,u; //local

    k=226;

    while(!kbhit()){

        getmousepos(button,x,y);

        if((button & 1==1 && x>5 && x<46 && y>80 && y<225)||(button & 1==1 && x>625 && x<635 && y>2 && y<12)){

            control\_button=1;

            posx=x;

            posy=y;

            break;

        }

        if(button & 1==1 && x>52 && x<627 && y>32 && y<417){

            while(button & 1==1){

                restrictmouse(52,32,627,417);

                hidemouse();

                gcolour=getpixel(x,y);

                showmouse();

                setcolor(gcolour);

                for(e=9;e<43;e++)

                    for(f=k+9;f<k+68;f++)

                        putpixel(e,f,gcolour);

                getmousepos(button,x,y);

            }

            for(e=9;e<43;e++)

                for(f=k+9;f<k+68;f++)

                    putpixel(e,f,LIGHTGRAY);

            colour=gcolour;

            restrictmouse(0,0,getmaxx(),getmaxy());

            colour=gcolour;

            u=getmaxy();

            rectangle(13,u-45,25,u-33); // selected foreground colour box

            setfillstyle(SOLID\_FILL,colour);

            floodfill(15,u-40,colour);

            setcolor(BLACK);

            line(25,u-33,25,u-45); line(25,u-33,13,u-33);

            setcolor(WHITE);

            line(13,u-45,25,u-45); line(13,u-45,13,u-33);

        } //end of if

    } //end of while

}//end of getcolour()

//int colour, col2; //global vars in selectcolour() already defined above

void selectcolour(int col\_button, int colx, int coly) {

    if(colx>40 && colx<52 && coly>430 && coly<442) {

        if(col\_button==2) col2=0;

        if(col\_button==1) colour=0;

    }

    if(colx>55 && colx<67 && coly>430 && coly<442) {

        if(col\_button==2) col2=1;

        if(col\_button==1) colour=1;

    }

    if(colx>70 && colx<82 && coly>430 && coly<442) {

        if(col\_button==2) col2=2;

        if(col\_button==1) colour=2;

    }

    if(colx>85 && colx<97 && coly>430 && coly<442) {

        if(col\_button==2) col2=3;

        if(col\_button==1) colour=3;

    }

    if(colx>100 && colx<112 && coly>430 && coly<442) {

        if(col\_button==2) col2=4;

        if(col\_button==1) colour=4;

    }

    if(colx>115 && colx<127 && coly>430 && coly<442) {

        if(col\_button==2) col2=5;

        if(col\_button==1) colour=5;

    }

    if(colx>130 && colx<142 && coly>430 && coly<442) {

        if(col\_button==2) col2=6;

        if(col\_button==1) colour=6;

    }

    if(colx>145 && colx<157 && coly>430 && coly<442) {

        if(col\_button==2) col2=7;

        if(col\_button==1) colour=7;

    }

    if(colx>40 && colx<52 && coly>445 && coly<457) {

        if(col\_button==2) col2=8;

        if(col\_button==1) colour=8;

    }

    if(colx>55 && colx<67 && coly>445 && coly<457) {

        if(col\_button==2) col2=9;

        if(col\_button==1) colour=9;

    }

    if(colx>70 && colx<82 && coly>445 && coly<457) {

        if(col\_button==2) col2=10;

        if(col\_button==1) colour=10;

    }

    if(colx>85 && colx<97 && coly>445 && coly<457) {

        if(col\_button==2) col2=11;

        if(col\_button==1) colour=11;

    }

    if(colx>100 && colx<112 && coly>445 && coly<457) {

        if(col\_button==2) col2=12;

        if(col\_button==1) colour=12;

    }

    if(colx>115 && colx<127 && coly>445 && coly<457) {

        if(col\_button==2) col2=13;

        if(col\_button==1) colour=13;

    }

    if(colx>130 && colx<142 && coly>445 && coly<457) {

        if(col\_button==2) col2=14;

        if(col\_button==1) colour=14;

    }

    if(colx>145 && colx<157 && coly>445 && coly<457) {

        if(col\_button==2) col2=15;

        if(col\_button==1) colour=15;

    }

    int y=getmaxy();

    if(col\_button==1) {

        setcolor(colour);

        rectangle(13,y-45,25,y-33); // selected foreground colour box

        setfillstyle(SOLID\_FILL,colour);

        floodfill(15,y-40,colour);

        setcolor(BLACK);

        line(25,y-33,25,y-45); line(25,y-33,13,y-33);

    } //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    //Right Click

    if(col\_button==2) {

        setcolor(col2);

        line(26,y-40,32,y-40);

        line(20,y-28,20,y-32);

        line(32,y-40,32,y-28);

        line(32,y-28,20,y-28); //selected background colour box

        line(26,y-40,26,y-32);

        line(26,y-32,20,y-32);

        setfillstyle(SOLID\_FILL,col2);

        floodfill(30,y-37,col2);

        setcolor(WHITE);

        line(32,y-40,32,y-28); line(32,y-28,20,y-28);

        setcolor(BLACK);

        line(26,y-40,32,y-40); line(20,y-28,20,y-32);

    }

} // end of selectcolor()

int check\_rub=0; //global

struct point {

    int xc;

    int yc;

    };

point p1,p2,p3,p4; //global

void four\_pt\_bez(point p1,point p2,point p3,point p4) {

    int x0,y0,x1,y1,x2,y2,x3,y3,vx,vy,x01,y01,x12,y12,x23,y23;

    int x012,y012,x123,y123,x0123,y0123,x,y;

    x0=p1.xc; y0=p1.yc;

    x3=p2.xc; y3=p2.yc;

    x1=p3.xc; y1=p3.yc;

    x2=p4.xc; y2=p4.yc;

    x01=x1-x0; y01=y1-y0;

    x12=x2-x1; y12=y2-y1;

    x23=x3-x2; y23=y3-y2;

    x012=x12-x01; y012=y12-y01;

    x123=x23-x12; y123=y23-y12;

    x0123=x123-x012; y0123=y123-y012;

    for(float t=0;t<1;t+=0.0001) {

        x=t\*t\*t\*x0123+3\*t\*t\*x012+3\*t\*x01+x0;

        y=t\*t\*t\*y0123+3\*t\*t\*y012+3\*t\*y01+y0;

    if(x>52 && x<628 && y>32 && y<417) line(x,y,x,y);

    }

} //end of four\_pt\_bez()

void thr\_pt\_bez(point p1,point p2,point p3) {

    int x0,y0,x1,y1,x2,y2,x3,y3,vx,vy,x01,y01,x12,y12,x23,y23;

    int x012,y012,x123,y123,x0123,y0123,x,y;

    x0=p1.xc; y0=p1.yc;

    x3=p2.xc; y3=p2.yc;

    x1=x2=p3.xc; y1=y2=p3.yc;

    x01=x1-x0; y01=y1-y0;

    x12=x2-x1; y12=y2-y1;

    x23=x3-x2; y23=y3-y2;

    x012=x12-x01; y012=y12-y01;

    x123=x23-x12; y123=y23-y12;

    x0123=x123-x012; y0123=y123-y012;

    for(float t=0;t<1;t+=0.0001) {

        x=t\*t\*t\*x0123+3\*t\*t\*x012+3\*t\*x01+x0;

        y=t\*t\*t\*y0123+3\*t\*t\*y012+3\*t\*y01+y0;

        if(x>52 && x<628 && y>32 && y<417) line(x,y,x,y);

    }

} //end tr\_pt\_bez()

void bez2();

void bez() {

    int button1 ,x1,y1,k=0;

    while(!k) {

        getmousepos(button1,x1,y1);

        if(button1==1) {

            setwritemode(XOR\_PUT);

            line(p1.xc,p1.yc,p2.xc,p2.yc);

            while(button1==1) {

                // restrictmouse(0,0,getmaxx(),getmaxy());

                setcolor(15-colour);

                p3.xc=x1;

                p3.yc=y1;

                hidemouse();

                thr\_pt\_bez(p1,p2,p3);

                thr\_pt\_bez(p1,p2,p3);

                showmouse();

                getmousepos(button1,x1,y1);

            }

            // restrictmouse(52,32,627,417);

            // setcolor(colour);

            hidemouse();

            thr\_pt\_bez(p1,p2,p3);

            k=1;

            showmouse();

            bez2();

        }

    }

} //end of bez()

void bez2() {

    int button1,x1,y1,k=0;

    while(!k) {

        showmouse();

        getmousepos(button1,x1,y1);

        if(button1==1) {

            // setcolor(0);

            setwritemode(XOR\_PUT);

            thr\_pt\_bez(p1,p2,p3);

            setwritemode(XOR\_PUT);

            while(button1==1) {

                // restrictmouse(0,0,getmaxx(),getmaxy());

                p4.xc=x1;

                p4.yc=y1;

                setcolor(15-colour);

                hidemouse();

                four\_pt\_bez(p1,p2,p3,p4);

                four\_pt\_bez(p1,p2,p3,p4);

                showmouse();

                getmousepos(button1,x1,y1);

            }

            // restrictmouse(0,0,getmaxx(),getmaxy());

            setwritemode(COPY\_PUT);

            setcolor(colour);

            hidemouse();

            four\_pt\_bez(p1,p2,p3,p4);

            k=1;

            showmouse();

        }

    }

} //end of bez2()

void curve(){

    int button,x,y,prevx,prevy;

    for(int k=80;k<212;k+=21){

        setcolor(BLACK);

        rectangle(5,k,25,k+20);

        rectangle(26,k,46,k+20);

        setcolor(WHITE);

        line(5,k,5,k+20); line(5,k,25,k);

        line(26,k,26,k+20); line(26,k,46,k);

    }

    setcolor(BLACK);

    line(26,164,26,184); line(26,164,46,184);

    setcolor(WHITE);

    line(46,184,46,164); line(46,184,26,184);

    while(TRUE){

        getmousepos(button,x,y);

        if((button & 1==1 && x>5 && x<46 && y>80 && y<225)||(button & 1==1 && x>625 && x<635 && y>2 && y<12)){

            control\_button=1;

            posx=x; posy=y;

            break;

        }

        if(button && x>=40 && x<=157 && y>=430 && y<=457){

            if(button==1) col\_button=1;

            if(button==2) col\_button=2;

            colx=x; coly=y;

            selectcolour(col\_button,colx,coly);

        }

        if(button & 1 == 1 && x>52 && x<628 && y>32 && y<418){

            prevx=x; prevy=y;

            setwritemode(XOR\_PUT);

            while((button & 1) == 1 && x>120){

                restrictmouse(52,32,627,417);

                setcolor(15-colour);

                hidemouse();

                line(prevx,prevy,x,y);

                line(prevx,prevy,x,y);

                showmouse();

                getmousepos(button,x,y);

            }

            restrictmouse(0,0,getmaxx(),getmaxy());

            hidemouse();

            // setcolor(colour);

            line(prevx,prevy,x,y);

            showmouse();

            p1.xc=prevx; p1.yc=prevy;

            p2.xc=x; p2.yc=y;

            bez();

        } //end of if

    } //end of while

} //end of curve()

struct coordinate {

    int x,y; //structure

    coordinate \*next;

    };

coordinate \*control,\*last,\*temp; //global pointer varables holding coordinate

void insert(int x,int y) { //data type

    coordinate \*new\_coord;

    new\_coord=new coordinate;

    new\_coord->x=x;

    new\_coord->y=y; //insert link function

    new\_coord->next=NULL;

    last->next=new\_coord;

    last=new\_coord;

}//end of insert()

void bucket(int x,int y,int backcolour,int colour){

    if(backcolour==colour)return;

    last=control=new coordinate; //last & control points to new blocks of memory

    control->x=x; // large enough to store a coordinate variable

    control->y=y;

    control->next=NULL; //Null pointer value

    while(control!=NULL){

        putpixel(x,y,colour);

        if(y-1>=32 && getpixel(x,y-1)==backcolour){

            putpixel(x,y-1,colour);

            insert(x,y-1);

        }

        if(x+1<628 && getpixel(x+1,y)==backcolour){

            putpixel(x+1,y,colour);

            insert(x+1,y);

        }

        if(y+1<418 && getpixel(x,y+1)==backcolour){

            putpixel(x,y+1,colour);

            insert(x,y+1);

        }

        if(x-1>=52 && getpixel(x-1,y)==backcolour){

            putpixel(x-1,y,colour);

            insert(x-1,y);

        }

        temp=control;

        control=temp->next; //change the control to next link

        delete temp;

        x=control->x;

        y=control->y;

    } //end of while

} //end of bucket()

void ffill(){

    int button,x,y,backcolour;

    for(int k=80;k<212;k+=21){

        setcolor(BLACK);

        rectangle(5,k,25,k+20); rectangle(26,k,46,k+20);

        setcolor(WHITE);

        line(5,k,5,k+20); line(5,k,25,k);

        line(26,k,26,k+20); line(26,k,46,k);

    }

    setcolor(BLACK);

    line(26,185,26,205); line(26,185,46,185);

    setcolor(WHITE);

    line(46,205,46,185); line(46,205,26,205);

    while(TRUE){

        getmousepos(button,x,y);

        if((button & 1==1 && x>5 && x<46 && y>80 && y<225)||(button & 1==1 && x>625 && x<635 && y>2 && y<12)){

            control\_button=1;

            posx=x; posy=y;

            break;

        }

        if(button && x>=40 && x<=157 && y>=430 && y<=457){

            if(button==1)col\_button=1;

            if(button==2)col\_button=2;

            colx=x;

            coly=y;

            selectcolour(col\_button,colx,coly);

        }

        if(button & 1 ==1 && x>52 && x<628 && y>32 && y<418){

            hidemouse();

            backcolour=getpixel(x,y);

            bucket(x,y,backcolour,colour);

        }

        showmouse();

    } //end of while

} //end of ffill()

void text(){

    int button,x,y,m,n,col[200],v,u,g=0,i,j;

    char ch,a[100];

    void \*bk;

    for(int k=80;k<212;k+=21){

        setcolor(BLACK);

        rectangle(5,k,25,k+20); rectangle(26,k,46,k+20);

        setcolor(WHITE);

        line(5,k,5,k+20); line(5,k,25,k);

        line(26,k,26,k+20); line(26,k,46,k);

    }

    setcolor(BLACK);

    line(26,206,26,226); line(26,206,46,206);

    setcolor(WHITE);

    line(46,226,46,206); line(46,226,26,226);

    while(TRUE){

        getmousepos(button,x,y);

        if((button & 1==1 && x>5 && x<46 && y>80 && y<225)||(button & 1==1 && x>625 && x<635 && y>2 && y<12)){

            control\_button=1;

            posx=x; posy=y;

            break;

        }

        if(button && x>=40 && x<=157 && y>=430 && y<=457){

            if(button==1) col\_button=1;

            if(button==2) col\_button=2;

            colx=x; coly=y;

            selectcolour(col\_button,colx,coly);

        }

        if(button & 1==1 && x>52 && x<627 && y>32 && y<417){

            u=m=x;

            n=y;

            setcolor(colour);

            hidemouse();

            settextstyle(0,0,1);

            for(int i=0;i<100;i++)

            a[i]=' ';

            a[0]='\_';

            g=0;

            v=imagesize(m,n,getmaxx()-10,n+9);

            bk=new int[v];

            getimage(m,n,getmaxx()-10,n+9,bk);

            outtextxy(m,n,a);

            while((ch=getch())!=13){

                if(ch==8) { //backspace

                    if(g>0){

                        putimage(m,n,bk,COPY\_PUT);

                        a[g]=' ';

                        a[g-1]='\_';

                        outtextxy(m,n,a);

                        g--;

                        u-=8;

                    }

                }

                else

                {

                    if(u<(getmaxx()-24)){

                        putimage(m,n,bk,COPY\_PUT);

                        a[g]=ch;

                        a[g+1]='\_';

                        g++;

                        u+=8;

                        outtextxy(m,n,a);

                    }

                    else

                    {

                    putimage(m,n,bk,COPY\_PUT);

                    a[g-1]=ch;

                    a[g]=' '; //if it reaches the end

                    outtextxy(m,n,a);

                    }

                }//else

            }// end of while

            a[g]=' ';

            putimage(m,n,bk,COPY\_PUT);

            outtextxy(m,n,a);

            showmouse();

        } //end of if

    } //end of while

}//end of text()

void poly(){

    X: int button,x,y,prevx,prevy,j=0,orix,oriy;

    for(int k=80;k<212;k+=21){

        setcolor(BLACK);

        rectangle(5,k,25,k+20); rectangle(26,k,46,k+20);

        setcolor(WHITE);

        line(5,k,5,k+20); line(5,k,25,k);

        line(26,k,26,k+20); line(26,k,46,k);

    }

    setcolor(BLACK);

    line(26,122,46,122); line(26,122,26,142);

    setcolor(WHITE);

    line(46,122,26,122); line(46,122,46,142);

    while(TRUE){

        getmousepos(button,x,y);

        if((button & 1==1 && x>5 && x<46 && y>80 && y<225)||(button & 1==1 && x>625 && x<635 && y>2 && y<12)){

            control\_button=1;

            posx=x; posy=y;

            break;

        }

        if(button && x>=40 && x<=157 && y>=430 && y<=457){

            if(button==1) col\_button=1;

            if(button==2) col\_button=2;

            colx=x; coly=y;

            selectcolour(col\_button,colx,coly);

        }

        if(button==1 && x<orix+2 && x>orix-2 && y<oriy+2 && y>oriy-2){

            hidemouse();

            setcolor(colour);

            line(orix,oriy,prevx,prevy);

            showmouse();

            goto X;

        }

        if(button==1 && x>=100 && x<=120 && y>=140 && y<=180)break;

        if(button & 1 ==1 && x>52 && x<627 && y>32 && y<417){

            hidemouse();

            if(j==0){ orix=prevx=x; oriy=prevy=y; }

            setwritemode(XOR\_PUT);

            while((button & 1) ==1){

                restrictmouse(52,32,627,417);

                if(j==0) {

                    getmousepos(button,x,y);

                    prevx=x; prevy=y;

                    j=1;

                }

                getmousepos(button,x,y);

                setcolor(15-colour);

                hidemouse();

                line(prevx,prevy,x,y); line(prevx,prevy,x,y);

                showmouse();

            }

            restrictmouse(0,0,640,480);

            setcolor(colour);

            setwritemode(COPY\_PUT);

            if(x>52&&x<627&&y>32&&y<417){

                hidemouse();

                line(prevx,prevy,x,y);

                showmouse();

            }

            prevx=x; prevy=y;

        }//end of if

    showmouse();

    } //end of while

} //end of poly()

void spray(){

    for(int k=80;k<212;k+=21){

        setcolor(BLACK);

        rectangle(5,k,25,k+20); rectangle(26,k,46,k+20);

        setcolor(WHITE);

        line(5,k,5,k+20); line(5,k,25,k);

        line(26,k,26,k+20); line(26,k,46,k);

    }

    setcolor(BLACK);

    line(5,206,25,206); line(5,206,5,226);

    setcolor(WHITE);

    line(25,226,25,206); line(25,226,5,226);

    int button,x,y,prevx,prevy;

    while(TRUE){

        getmousepos(button,x,y);

        if((button & 1==1 && x>5 && x<46 && y>80 && y<225)||(button & 1==1 && x>625 && x<635 && y>2 && y<12)){

            control\_button=1;

            posx=x; posy=y;

            break;

        }

        if(button && x>=40 && x<=157 && y>=430 && y<=457){

            if(button==1) col\_button=1;

            if(button==2) col\_button=2;

            colx=x; coly=y;

            selectcolour(col\_button,colx,coly);

        }

        if(button & 1==1 && x>52 && x<627 && y>32 && y<417) {

            hidemouse();

            int i,j;

            while((button & 1) ==1){

                restrictmouse(52,32,627,417);

                setcolor(colour);

                for(i=x,j=y;i<x+5,j<y+5;i+=random(20),j+=random(20)){

                    if(i<x-5)break;

                    if(j<y-5) break;

                    i-=random(20);

                    j-=random(20);

                    if(i>52&&i<627&&j>32&&j<417)putpixel(i,j,colour);

                    getmousepos(button,x,y);

                }

                break;

            } //end of while

        }//end of if

    restrictmouse(0,0,640,480);

    showmouse();

    } //end of while

} //end of spray()

void thickline(int prevx,int prevy,int x,int y){

    int radius =3;

    if(check\_rub==1) {

        setcolor(WHITE);

        setfillstyle(SOLID\_FILL,WHITE);

        }

        else

        {

        setcolor(colour);

        setfillstyle(SOLID\_FILL,colour);

        }

    int lefx,upy,rigx,lowy;

    if(prevx>x){ lefx=x; rigx=prevx;}

        else { lefx=prevx; rigx=x;}

    if(prevy>y){ upy=y; lowy=prevy;}

        else { upy=prevy; lowy=y;}

    int s1,s2,q1,q2;

    s1=prevx; s2=prevy;

    q1=x; q2=y;

    if(lefx!=rigx) for(int i=lefx;i<=rigx;i++) fillellipse(i,(((q2-s2)\*(i-s1))/(q1-s1))+s2,radius,radius);

    if(upy!=lowy) for(int j=upy;j<=lowy;j++)fillellipse((((j-s2)\*(q1-s1))/(q2-s2))+s1,j,radius,radius);

    showmouse();

} //end of thickline()

void brush(){

    check\_rub=0;

    int button,x,y,prevx,prevy;

    for(int k=80;k<212;k+=21) {

        setcolor(BLACK);

        rectangle(5,k,25,k+20); rectangle(26,k,46,k+20);

        setcolor(WHITE);

        line(5,k,5,k+20); line(5,k,25,k);

        line(26,k,26,k+20); line(26,k,46,k);

    }

    setcolor(BLACK);

    line(26,143,46,143); line(26,143,26,163);

    setcolor(WHITE);

    line(46,163,46,143); line(46,163,26,163);

    while(TRUE){

        getmousepos(button,x,y);

        if((button &1==1 && x>5 && x<46 && y>80 && y<225)||(button & 1==1 && x>625 && x<635 && y>2 && y<12)){

            control\_button=1;

            posx=x; posy=y;

            break;

        }

        if(button && x>=40 && x<=157 && y>=430 && y<=457){

            if(button==1) col\_button=1;

            if(button==2) col\_button=2;

            colx=x; coly=y;

            selectcolour(col\_button,colx,coly);

        }

        int q=3; //q=radius of thick line -refer thickline function

        if(button & 1 ==1 && x>52+q && x<628-q && y>32+q && y<418-q) {

            setcolor(colour);

            prevx=x; prevy=y;

            while((button==1)) {

                restrictmouse(52+q,32+q,627-q,417-q);

                getmousepos(button,x,y);

                setcolor(colour);

                hidemouse();

                thickline(prevx,prevy,x,y);

                showmouse();

                prevx=x; prevy=y;

            }

            restrictmouse(0,0,getmaxx(),getmaxy());

        } //end of if

    } // end of while

} //end of brush()

void rub(){

    check\_rub=1;

    int button,x,y,prevx,prevy;

    for(int k=80;k<212;k+=21){

        setcolor(BLACK);

        rectangle(5,k,25,k+20); rectangle(26,k,46,k+20);

        setcolor(WHITE);

        line(5,k,5,k+20); line(5,k,25,k);

        line(26,k,26,k+20); line(26,k,46,k);

    }

    setcolor(BLACK);

    line(26,101,46,101); line(26,101,26,121);

    setcolor(WHITE);

    line(46,121,46,101); line(46,121,26,121);

    while(TRUE){

        getmousepos(button,x,y);

        if((button &1==1 && x>5 && x<46 && y>80 && y<225)||(button & 1==1 && x>625 && x<635 && y>2 && y<12)) {

            control\_button=1;

            posx=x; posy=y;

            break;

        }

        if(button && x>=40 && x<=157 && y>=430 && y<=457){

            if(button==1) col\_button=1;

            if(button==2) col\_button=2;

            colx=x; coly=y;

            selectcolour(col\_button,colx,coly);

        }

        int q=3; //q=radius of thick line - refer thickline function

        if(button & 1 ==1 && x>52+q && x<628-q && y>32+q && y<418-q){

            setcolor(WHITE);

            prevx=x; prevy=y;

            while((button==1)) {

                restrictmouse(52+q,32+q,627-q,417-q);

                getmousepos(button,x,y);

                setcolor(WHITE);

                hidemouse();

                thickline(prevx,prevy,x,y);

                showmouse();

                prevx=x; prevy=y;

            }

            restrictmouse(0,0,getmaxx(),getmaxy());

        }//end of if

    } //end of while

} //end of rub()

void line() {

    for(int k=80;k<212;k+=21) {

        setcolor(BLACK);

        rectangle(5,k,25,k+20); rectangle(26,k,46,k+20);

        setcolor(WHITE);

        line(5,k,5,k+20); line(5,k,25,k);

        line(26,k,26,k+20); line(26,k,46,k);

    }

    setcolor(BLACK);

    line(5,185,5,205); line(5,185,25,185);

    setcolor(WHITE);

    line(25,205,25,185); line(25,205,5,205);

    int button,x,y,prevx,prevy;

    while(TRUE){

        getmousepos(button,x,y);

        if((button & 1==1 && x>5 && x<46 && y>80 && y<225)||(button & 1==1 && x>625 && x<635 && y>2 && y<12)) {

            control\_button=1;

            posx=x; posy=y;

            break;

        }

        if(button && x>=40 && x<=157 && y>=430 && y<=457){

            if(button==1) col\_button=1;

            if(button==2) col\_button=2;

            colx=x; coly=y;

            selectcolour(col\_button,colx,coly);

        }

        if(button & 1 ==1 && x>52 && x<628 && y>32 && y<418){

            getmousepos(button,x,y);

            setwritemode(XOR\_PUT);

            prevx=x; prevy=y;

            while((button & 1) == 1){

                restrictmouse(52,32,627,417);

                getmousepos(button,x,y);

                setcolor(15-colour);

                hidemouse();

                line(prevx,prevy,x,y);

                line(prevx,prevy,x,y);

                showmouse();

            }

            setwritemode(COPY\_PUT);

            setcolor(colour);

            hidemouse();

            line(prevx,prevy,x,y);

            showmouse();

            restrictmouse(0,0,getmaxx(),getmaxy());

        } //end of if

    } //end of while

} // end of line()

void myellipse(int cenx,int ceny,int xrad,int yrad) {

    float cx,cy,angle=0;

    while(angle<360) {

        float THETA=PI/180.0 \* angle;

        cx=cenx+xrad\*cos(THETA);

        cy=ceny-yrad\*sin(THETA);

        line(cx,cy,cx,cy);

        angle+=.5;

    }

}//end of myellipse()

void circle() {

    for(int k=80;k<212;k+=21) {

        setcolor(BLACK);

        rectangle(5,k,25,k+20); rectangle(26,k,46,k+20);

        setcolor(WHITE);

        line(5,k,5,k+20); line(5,k,25,k);

        line(26,k,26,k+20); line(26,k,46,k);

    }

    setcolor(BLACK);

    line(5,143,5,163); line(5,143,25,143);

    setcolor(WHITE);

    line(25,163,5,163); line(25,163,25,143);

    int button,x,y,prevx,prevy,i,j;

    while(TRUE) {

        getmousepos(button,x,y);

        if((button & 1==1 && x>5 && x<46 && y>80 && y<225)||(button & 1==1 && x>625 && x<635 && y>2 && y<12)){

            control\_button=1;

            posx=x; posy=y;

            break;

        }

        if(button && x>=40 && x<=157 && y>=430 && y<=457){

            if(button==1) col\_button=1;

            if(button==2) col\_button=2;

            colx=x; coly=y;

            selectcolour(col\_button,colx,coly);

        }

        if(button & 1 ==1 && x>52 && x<628 && y>32 && y<418){

            getmousepos(button,x,y);

            setwritemode(XOR\_PUT);

            prevx=x; prevy=y;

            while((button & 1) == 1){

                restrictmouse(52,32,627,417);

                getmousepos(button,x,y);

                setcolor(15-colour);

                hidemouse();

                i=abs(x-prevx); j=abs(y-prevy);

                myellipse((prevx+x)/2,(prevy+y)/2,i/2,j/2);

                myellipse((prevx+x)/2,(prevy+y)/2,i/2,j/2);

                showmouse();

            }

            setwritemode(COPY\_PUT);

            setcolor(colour);

            hidemouse();

            ellipse((prevx+x)/2,(prevy+y)/2,0,360,i/2,j/2);

            showmouse();

            restrictmouse(0,0,getmaxx(),getmaxy());

        } //end of if

    showmouse();

    } //end of while

} //end of circle()

void rectangle() {

    for(int k=80;k<212;k+=21) {

        setcolor(BLACK);

        rectangle(5,k,25,k+20); rectangle(26,k,46,k+20);

        setcolor(WHITE);

        line(5,k,5,k+20); line(5,k,25,k);

        line(26,k,26,k+20); line(26,k,46,k);

    }

    setcolor(BLACK);

    line(5,164,5,184); line(5,164,25,164);

    setcolor(WHITE);

    line(25,184,25,164); line(25,184,5,184);

    int button,x,y,prevx,prevy;

    while(TRUE) {

        getmousepos(button,x,y);

        if((button & 1 == 1 && x>5 && x<46 && y>80 && y<225)||(button & 1==1 && x>625 && x<635 && y>2 && y<12)){

            control\_button=1;

            posx=x; posy=y;

            break;

        }

        if(button && x>=40 && x<=157 && y>=430 && y<=457) {

            if(button==1) col\_button=1;

            if(button==2) col\_button=2;

            colx=x;

            coly=y;

            selectcolour(col\_button,colx,coly);

        }

        if(button & 1 == 1 && x>52 && x<628 && y>32 && y<418) {

            getmousepos(button,x,y);

            setwritemode(XOR\_PUT);

            prevx=x; prevy=y;

            while((button & 1) == 1) {

                restrictmouse(52,32,627,417);

                getmousepos(button,x,y);

                setcolor(15-colour);

                hidemouse();

                rectangle(prevx,prevy,x,y);

                rectangle(prevx,prevy,x,y);

                showmouse();

            }

            setwritemode(COPY\_PUT);

            setcolor(colour);

            hidemouse();

            rectangle(prevx,prevy,x,y);

            showmouse();

            restrictmouse(0,0,getmaxx(),getmaxy());

        }

    }

} //end of tectangle()

void main() {

    int gdriver = DETECT, gmode;

    initgraph(&gdriver, &gmode, "C:\\turboc3\\bgi");

    gui();

    //(52,30,628,418) is the drawing area with lightgray background

    initmouse();

    showmouse();

    setcolor(BLACK);

    colour=0;

    pencil(); //to start with by default

    while(TRUE) {

        // Action after left-clicking on first row of icons

        if(control\_button==1 && posx>5 && posx<25 && posy>100 && posy<120) pencil();

        if(control\_button==1 && posx>5 && posx<25 && posy>121 && posy<141) getcolour();

        if(control\_button==1 && posx>5 && posx<25 && posy>142 && posy<162) circle();

        if(control\_button==1 && posx>5 && posx<25 && posy>163 && posy<183) rectangle();

        if(control\_button==1 && posx>5 && posx<25 && posy>184 && posy<204) line();

        if(control\_button==1 && posx>5 && posx<25 && posy>205 && posy<225) spray();

        // Acion after left-clicking second row of icons

        if(control\_button==1 && posx>26 && posx<46 && posy>100 && posy<120) rub();

        if(control\_button==1 && posx>26 && posx<46 && posy>121 && posy<141) poly();

        if(control\_button==1 && posx>26 && posx<46 && posy>142 && posy<162) brush();

        if(control\_button==1 && posx>26 && posx<46 && posy>163 && posy<183) curve();

        if(control\_button==1 && posx>26 && posx<46 && posy>184 && posy<204) ffill();

        if(control\_button==1 && posx>26 && posx<46 && posy>205 && posy<225) text();

        // binary file save and load

        if(control\_button==1 && posx>5 && posx<25 && posy>80 && posy<100)

        {

            save\_image();

            outtextxy(4,335,"SAVED");

            pencil(); //focus on pencil button

        }

        if(control\_button==1 && posx>26 && posx<46 && posy>80 && posy<100)

        {

            load\_image();

            pencil(); //focus on pencil button

        }

        // Acion after left-clicking on exit icon

        if(control\_button==1 && posx>625 && posx<635 && posy>2 && posy<12) exit(0);

    }

}

Graphical user interface, application

Description automatically generated

**Program 8: [Bresenhaum's Line Drawing] Develop a program to illustrate VGA line drawing in 640x480, 16-color mode by drawing four rectangles full of lines.**

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<stdlib.h>

#include<dos.h>

//void draw(int x1,int y1,int x2,int y2, int c);

void draw(int x1,int y1,int x2,int y2,int c)

{

float dx,dy,p;

int i,x,y,xend;

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

p=2\*dy-dx;

if(x1>x2)

{

x=x2;y=y2;

xend=x1;

}

else

{

x=x1; y=y1;

xend=x2;

}

putpixel(x,y,c);

while(x<xend)

{

if(p<0)

{

x=x+1;

putpixel(x,y,c);

p=p+(2\*dy);

}

else

{

x=x+1; y=y+1;

putpixel(x,y,c);

p=p+(2\*dy)-(2\*dx);

}

}

}

void drawline(int x0, int y0, int x1, int y1,int c)

{

    while(y0<y1)

    { if(x0 == x1)

       {

    putpixel(x0,y0,c);

    y0=y0+1;

    }

    }

}

void main()

{

int i,j,a,b, gd=DETECT,gm;

initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");

for( i = 0; i<=50; i= i+3)

{

draw(50,50+i, 150, 50+i,7);

drawline(50, 50,50, 100,7);

drawline(150,50,150,100,7);

delay(50);

}

for( a =0; a<=50;a=a+3)

{

draw(50,210-a,150,210-a,5);

drawline(50,160,50,210,5);

drawline(150,160,150,210,5);

delay(50);

}

for( b =0;b<=100;b=b+3)

{

drawline(150-b,310,150-b,410,3);

draw(50,310,150,310,3);

draw(50,410,150,410,3);

delay(50);

}

for( j =0; j<=100; j=j+3)

{

drawline(350+j,50,350+j,410,10);

draw(350,50,450,50,10);

draw(350,410,450,410,10);

delay(50);

}

getch();

}

Graphical user interface

Description automatically generated with medium confidence

**Program 9:** **[Bresenhaum's Circle Drawing] Develop a program to illustrate VGA circle drawing in 640x480, 16-color mode by drawing concentric circles, starting from a blob.**

#include<stdio.h>

#include<graphics.h>

#include<conio.h>

#include<dos.h>

#include<math.h>

void main()

{

       int gd=DETECT,gm;

       int d,r,x,y,xc,yc;

       clrscr();

       initgraph(&gd,&gm,"C://TURBOC3//BGI");

       printf("enter the radius");

       scanf("%d",&r);

       printf("enter the centre of circle");

       scanf("%d",&xc);

       scanf("%d",&yc);

       for(int i=0;i<=10;i++)

       {

       d=3-2\*r;

       x=0;

       y=r;

       while(x<=y)

       {

  putpixel(xc+x,yc+y,5);

  putpixel(xc-y,yc-x,5);

  putpixel(xc+y,yc-x,5);

  putpixel(xc-y,yc+x,5);

  putpixel(xc+y,yc+x,5);

  putpixel(xc-x,yc-y,5);

  putpixel(xc+x,yc-y,5);

  putpixel(xc-x,yc+y,5);

  if(d<=0)

  {

     d=d+4\*x+6;

  }

  else

  {

      d=d+4\*x-4\*y+10;

      y=y-1;

  }

  x=x+1;

  delay(40);

}

r=r+10;

}

getch();

}

Background pattern

Description automatically generated with medium confidence

**Program 10: [Bresenhaum's Ellipse Drawing] Develop a program to illustrate VGA ellipse drawing in 640x480, 16-color mode by drawing concentric ellipses starting from blob.**

#include <stdio.h>

#include <conio.h>

#include <graphics.h>

#include <math.h>

#include <dos.h>

void main()

{

long int d1,d2;

int i,gd=DETECT ,gm,x,y;

long int rx,ry,rxsq,rysq,tworxsq,tworysq,dx,dy;

printf("Enter the x Radius of the ellipse");

scanf("%ld",&rx);

printf("Enter the y Radius of the ellipse");

scanf("%ld",&ry);

initgraph(&gd, &gm, "c:\\turboc3\\bgi");

for( i=0;i<5;i++)

{

rxsq=rx\*rx;

rysq=ry\*ry;

tworxsq=2\*rxsq;

tworysq=2\*rysq;

x=0;

y=ry;

d1=rysq - (rxsq \* ry) + (0.25 \* rxsq);

dx= tworysq \* x;

dy= tworxsq \* y;

do

{

putpixel(200+x,200+y,9);

putpixel(200-x,200-y,9);

putpixel(200+x,200-y,9);

putpixel(200-x,200+y,9);

if (d1 < 0)

{

x=x+1;

y=y;

dx=dx + tworysq;

d1=d1 + dx + rysq;

}

else

{

x=x+1;

y=y-1;

dx= dx + tworysq;

dy= dy - tworxsq;

d1= d1 + dx - dy + rysq;

}

delay(50);

}while (dx < dy);

d2 = rysq \* ( x + 0.5) \* ( x + 0.5 ) + rxsq \* (y - 1) \* (y-1) - rxsq \* rysq;

do

{

putpixel(200+x,200+y,11);

putpixel(200-x,200-y,11);

putpixel(200+x,200-y,11);

putpixel(200-x,200+y,11);

if (d2 >0)

{

x=x;

y=y-1;

dy = dy - tworxsq;

d2 = d2 - dy + rxsq;

}

else

{

x= x+1;

y=y-1;

dy=dy - tworxsq;

dx= dx + tworysq;

d2 = d2 + dx -dy + rxsq;

}

delay(10);

} while ( y> 0);

rx=rx+10;

ry=ry+10;

}

getch();

closegraph();

}

Shape, circle

Description automatically generated

**Program 11: [Line Clipping in 2D] Develop a program to use Cohen-Sutherland recursive algorithm to line clip an arbitrary set of lines against a drawing window. Lines are represented as (x1, y1) and (x2, y2) co-ordinates in a text file.**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

// Defining region codes

const int INSIDE = 0; // 0000

const int LEFT = 1, RIGHT = 2, BOTTOM = 4, TOP = 8;

int x\_max = 10, y\_max = 8, x\_min = 4, y\_min = 4;

// assigning region code

int computeCode(double x, double y)

{

int code = INSIDE;

if (x < x\_min)

code |= LEFT;

else if (x > x\_max)

code |= RIGHT;

if (y < y\_min)

code |= BOTTOM;

else if (y > y\_max)

code |= TOP;

return code;

}

void cohenSthAlgo(double x1, double y1,

double x2, double y2)

{

// region codes (P1 & P2)

int code1 = computeCode(x1, y1);

int code2 = computeCode(x2, y2);

int accept = 0;

while (1)

{

if ((code1 == 0) && (code2 == 0))

{

accept = 1;

break;

}

else if (code1 & code2)

break;

else

{

int code\_out;

double x, y;

if (code1 != 0)

code\_out = code1;

else

code\_out = code2;

// y = y1 + slope \* (x - x1),

// x = x1 + (1 / slope) \* (y - y1)

if (code\_out & TOP)

{

x = x1 + (x2 - x1) \* (y\_max - y1) / (y2 - y1);

y = y\_max;

}

else if (code\_out & BOTTOM)

{

x = x1 + (x2 - x1) \* (y\_min - y1) / (y2 - y1);

y = y\_min;

}

else if (code\_out & RIGHT)

{

y = y1 + (y2 - y1) \* (x\_max - x1) / (x2 - x1);

x = x\_max;

}

else if (code\_out & LEFT)

{

y = y1 + (y2 - y1) \* (x\_min - x1) / (x2 - x1);

x = x\_min;

}

if (code\_out == code1)

{

x1 = x;

y1 = y;

code1 = computeCode(x1, y1);

}

else

{

x2 = x;

y2 = y;

code2 = computeCode(x2, y2);

}

}

}

if (accept)

{

cout <<"Line accepted from " << (int)(x1+0.5) << ", "

<< (int)(y1+0.5) << " to "<< (int)(x2+0.5)

<< ", " << (int)(y2+0.5) << endl;

line(x1, y1, x2, y2);

}

else

cout << "Line rejected" << endl;

}

void main()

{

clrscr();

double px1, py1, px2, py2, left, top, right, bottom;

int gd = DETECT,gm;

initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

cout<<"Enter window\'s minimum coordinates (x, y): ";

cin>>x\_min>>y\_min;

cout<<"Enter window\'s maximum coordinates (x, y): ";

cin>>x\_max>>y\_max;

cout<<"Enter initial coordinates of line (x, y): ";

cin>>px1>>py1;

cout<<"Enter final coordinates of line (x, y): ";

cin>>px2>>py2;

rectangle(x\_min, y\_max, x\_max, y\_min);

line(px1, py1, px2, py2);

outtextxy(300, 455,"Press any key to clip the line");

getch();

setcolor(YELLOW);

cohenSthAlgo(px1, py1, px2, py2);

getch();

}

A picture containing text

Description automatically generated

**Program 12: [Transformations in 2D] The goal of this exercise is to build a minimal system for performing the primary transformations on reasonably complex and general line drawings. Develop a program to transform a graphical object (made up of line segments) with the ability to perform primary transformations [Translation, Scaling and Rotation], any arbitrary concatenation of these transformations, and a few simple utilities in a reasonably friendly environment. Select a user-controlled loop in the function Menu() as the control structure for the program. That is, the function Menu() should require the user to move in any order among the following transformations and utility functions by use of one-letter keyboard commands or highlighting the menu option and pressing a return key: (L)oad: Reads a new graphics object data file at any point of the session,**

**(T)ranslate: Translates the current object by a user-provided (Tx, Ty),**

**(S)cale: Scales the current object by a user-provided (Sx, Sy)**

**(R)otate: Rotates the current object by a user-provided angle, Theta,**

**(D)isplay: The object being transformed is displayed upon this command,**

**(C)lear: Erases the current drawing window, (Q)uit: Exits the program.**

#include<stdio.h>

#include<graphics.h>

#include<stdlib.h>

#include<math.h>

#include<conio.h>

int x1,y1,x2,y2,midx,midy;

void axis();

void translation()

{

int tx,ty,xn1,yn1,xn2,yn2;

printf("\n Enter the translation:\n");

scanf("%d%d",&tx,&ty);

cleardevice();

outtextxy(400,100,"TRANSLATION");

xn1=x1+tx;

yn1=y1+ty;

xn2=x2+tx;

yn2=y2+ty;

axis();

rectangle(xn1,yn1,xn2,yn2);

getch();

}

void scaling()

{

float xn1,yn1,xn2,yn2;

float sx,sy;

printf("Enter the scaling factor");

scanf("%f%f",&sx,&sy);

cleardevice();

outtextxy(300,200,"SCALING");

xn1=x1\*sx;

yn1=y1\*sy;

xn2=x2\*sx;

yn2=y2\*sy;

axis();

rectangle(xn1,yn1,xn2,yn2);

getch();

}

void rotation()

{

int ang;

float rx,xn1,yn1,xn2,yn2,x1n1,y1n1,x2n2,y2n2;

printf("\n Enter the angle for rotation:\n");

scanf("%d",&ang);

cleardevice();

outtextxy(500,200,"ROTATION");

rx=(ang\*3.14)/180;

xn1=x1\*cos(rx)-y1\*sin(rx);

yn1=y1\*cos(rx)+x1\*sin(rx);

xn2=x2\*cos(rx)-y2\*sin(rx);

yn2=y2\*cos(rx)+x2\*sin(rx);

x1n1=x2\*cos(rx)-y1\*sin(rx);

y1n1=y1\*cos(rx)+x2\*sin(rx);

x2n2=x1\*cos(rx)-y2\*sin(rx);

y2n2=y2\*cos(rx)+x1\*sin(rx);

axis();

line(xn1,yn1,x1n1,y1n1);

line(x1n1,y1n1,xn2,yn2);

line(xn2,yn2,x2n2,y2n2);

line(x2n2,y2n2,xn1,yn1);

getch();

}

void get()

{

printf("\n Enter the coordinates x1,y1,x2,y2");

scanf("%d%d%d%d",&x1,&y1,&x2,&y2);

outtextxy(200,100,"ORIGINAL OBJECT");

x1= getmaxx() / 2-x1;

y1= getmaxy() / 2-y1;

x2 = getmaxx() / 2+x2;

y2 = getmaxy() / 2+y2;

axis();

getch();

}

void axis()

{

midx=getmaxx() / 2;

midy=getmaxy() / 2;

line(0,midy,midx\*2,midy);

line(midx,0,midx,midy\*2);

rectangle(x1,y1,x2,y2);

}

void main()

{

int ch,gd=DETECT,gm;

initgraph(&gd, &gm, "c:\\turboc3\\bgi");

get();

do

{

cleardevice();

outtextxy(10,10,"1)TRANSLATION");

outtextxy(10,20,"2)SCALING");

outtextxy(10,30,"3)ROTATION");

outtextxy(10,60,"4)EXIT");

outtextxy(10,70,"ENTER UR CHOICE:");

scanf("%d",&ch);

switch(ch)

{

case 1:

translation();

break;

case 2:

scaling();

break;

case 3:

rotation();

break;

case 4:

exit(0);

}

}while(ch<4);

}

Chart, box and whisker chart

Description automatically generated

Shape

Description automatically generated with medium confidence

**Translation : 10 20**

Graphical user interface

Description automatically generated with low confidence

**Scaling: 8 10**

A picture containing text

Description automatically generated

**Rotation:40**

Graphical user interface, text

Description automatically generated with medium confidence

**Program 13: [View Algorithm] Write a program to plot 4 cycles of sine and cosine wave for mapping a window to a viewport. Include the axes and scales in your output.**

#include<graphics.h>

#include<math.h>

#include<conio.h>

#include<dos.h>

#include<stdio.h>

#define pie 3.1412

void transform (float minang, float maxang, int option, int l,

int b, int t, int r);

void main(){

int option, l, b, t, r;

float ang2, iang, ang1, maxang, maxy;

int gd = VGA, gm = VGAHI, ec = 0;

clrscr();

gotoxy(20, 4); printf("sin/cosine curve drawing");

gotoxy(20, 6); printf("Press 1->SINE CURVE");

gotoxy(20, 8); printf("Press 2->COSINE CURVE");

gotoxy(20, 10); printf("Enter the option:");

gotoxy(40, 10);scanf("%d", &option);

gotoxy(20, 12);printf ("enter the initial angle in degrees:");

gotoxy(56, 12);scanf("%f", &iang); ang1=(iang \* pie) / 180;

gotoxy(20, 14);printf("Enter the maximum value of angle in degrees:");

gotoxy(70, 14);scanf("%f", &maxang); ang2=(maxang \* pie)/180;

printf("\n\n\n Press any key to conitune"); getch();

initgraph(&gd, &gm, "C:\\turboc3\\bgi");

ec=graphresult();

if(ec!=0)printf("Graphics system error\n",grapherrormsg(ec));

//printf("Enter the coordinates for the viewport:\n");

//scanf("%d %d %d %d", &l, &t, &r, &b);

l=0; t=0; r=639; b=479;

setviewport(l, t, r, b, 0);

//printf("\n%d %d", getmaxx(), getmaxy());

cleardevice();

rectangle (l, t, r, b);

line(l, b - ((b - t)/2), r, b-((b - t)/2));

transform (ang1, ang2, option, l, t, r, b);

getch();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Transformation of world coordinates to screen coordinates\*\*/

/\*World coordinates: xmin, ymin, xmax, y max \*\*/

/\*screen coordinates: sxmin, symin, sxmax, symax \*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void transform (float minang, float maxang, int option,int l, int

t, int r, int b){

float temp, x, y, xmax, ymax, xmin, ymin, sxmax, symax, sxmin,

symin;

double sx, sy;

sxmin = l; sxmax = r;

symin = t; symax = b;

xmin = minang; xmax = maxang;

ymin = -1.0; ymax = 1.0;

while (minang <= maxang){

if (option == 1) temp = sin(minang);

else temp = cos(minang);

y = -(temp);

x = minang;

sx = (double)((sxmin)+(((sxmax-sxmin)/(xmax- xmin)) \*(x-xmin)));

sy = (double)((symin)+(((symax-symin)/(ymax- ymin)) \*(y-ymin)));

putpixel((long)sx, (long)sy,3);

minang += 0.05;

}

}

**Sine Curve**

Text

Description automatically generated

**A picture containing background pattern

Description automatically generated**

**Cosine Curve**

Text

Description automatically generated

**Chart

Description automatically generated with low confidence**

**Program 14: [Dot Plot] Write a program to draw a ‘dot plot’ of the function f (x) e cos(2 x) x π − = where x varies from 0 to 4 by choosing some suitable increment, say 0.005, between consecutive x-values. Find the appropriate scaling and translation factors so that the dots will lie properly in a screen window with width W pixels and height H pixels.**

#include <windows.h> // use proper includes for your system

#include <math.h>

//#include <gl/Gl.h>

#include <gl/glut.h>

const int screenWidth = 640; // width of screen window in pixels

const int screenHeight = 480; // height of screen window in pixels

GLdouble A, B, C, D; // values used for scaling and shifting

void myInit(void) {

glClearColor(1.0, 1.0, 1.0, 0.0); // background color is white

glColor3f(0.5f, 0.0f, 1.0f);

glPointSize(2.0); // a 'dot' is 2 by 2 pixels

glMatrixMode(GL\_PROJECTION); // set "camera shape"

glLoadIdentity();

gluOrtho2D(0.0, (GLdouble)screenWidth, 0.0, (GLdouble)screenHeight);

A = screenWidth / 4.0; // set values used for scaling and shifting

B = 0.0;

C = D = screenHeight / 2.0;

}

void myDisplay(void) {

glClear(GL\_COLOR\_BUFFER\_BIT); // clear the screen

glBegin(GL\_POINTS);

for (GLdouble x = 0; x < 4.0; x += 0.005) {

GLdouble func = exp(-x) \* cos(2 \* 3.14159265 \* x);

glVertex2d(A \* x + B, C \* func + D);

}

glEnd();

glFlush(); // send all output to display

}

int main(int argc, char\*\* argv) {

glutInit(&argc, argv); // initialize the toolkit

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB); // set display mode

glutInitWindowSize(screenWidth, screenHeight); // set window size

glutInitWindowPosition(100, 150); // set window position on screen

glutCreateWindow("Dot Plot of a Function"); // open the screen window

glutDisplayFunc(myDisplay); // register redraw function

myInit();

glutMainLoop(); // go into a perpetual loop

}

Chart, line chart

Description automatically generated

**Program 15: [Rubber Band or Elastic Lines] Lines are the basic graphical primitives of all construction programs and any technique for simplifying and making line drawing more intuitive is highly desirable. One of the most useful techniques for constructing lines is the rubber band algorithm. Rubber banding involves sending a message via input device to select the anchor point (first point) of the line, moving the cursor to the terminal point of the line and sending another message to signal the end point. During the motion to the final point position, the rubber band algorithm draws, erases, and redraws the line repeatedly between the anchor point and the present cursor position to provide highly effective feedback to the designer on what the final result will look like. Develop a program to demonstrate Rubber Band Technique for constructing graphical objects (Press Left-Right mouse buttons left of drawing window to Quit).**

#include <GL/glut.h>

float xb, yb, xe, ye;

int first = 1;

void init()

{

glClearColor(0.0, 1.0, 1.0, 1.0);

glColor3f(1.0f, 0.0f, 0.0f);

glEnable(GL\_COLOR\_LOGIC\_OP);

glLogicOp(GL\_COPY);

}

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glLogicOp(GL\_XOR);

/\* Something interesing here \*/

glFlush();

}

void reshape(int w, int h)

{

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D (0.0, 500.0, 0.0, 500.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

void rubber(int x, int y)

{

/\* Draw old line in XOR mode \*/

glLogicOp(GL\_XOR);

glBegin(GL\_LINES);

glVertex2f(xb, yb);

glVertex2f(xe, ye);

glEnd();

glFlush();

xe = x;

ye = 500-y;

/\* Draw new line in XOR mode \*/

glBegin(GL\_LINES);

glVertex2f(xb, yb);

glVertex2f(xe, ye);

glEnd();

glFlush();

}

void mouse(int btn, int state, int x, int y)

{

if(btn==GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)

{

xb = x;

yb = 500-y;

if (first) {

xe = xb;

ye = yb;

first = 0;

}

}

if(btn==GLUT\_LEFT\_BUTTON && state == GLUT\_UP)

{

/\* Draw old line in XOR mode \*/

glBegin(GL\_LINES);

glVertex2f(xb, yb);

glVertex2f(xe, ye);

glEnd();

glFlush();

glLogicOp(GL\_COPY);

xe = x;

ye = 500-y;

/\* Draw new line in XOR mode \*/

glBegin(GL\_LINES);

glVertex2f(xb, yb);

glVertex2f(xe, ye);

glEnd();

glFlush();

first = 1;

glLogicOp(GL\_XOR);

}

}

int main(int argc, char\*\* argv)

{

/\* Initialize mode and open a window in upper left corner of screen \*/

/\* Window title is name of program (arg[0]) \*/

glutInit(&argc,argv);

glutInitDisplayMode (GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500,500);

glutInitWindowPosition(0,0);

glutCreateWindow("Rubberband Lines");

glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutMouseFunc(mouse);

glutMotionFunc(rubber);

init();

glutMainLoop();

}

Chart, line chart

Description automatically generated

**Program 16: [GUI Application: 15-Puzzle Design] Design Specifications for Puzzle:**

**You decide that Turbo-C++ Puzzle should perform the following:**

**It should allow re-initialization at any point during play.ϖ**

**It should allow movement of one/ two/ three tiles up, down, left and rightϖ**

**It should signal successful solution through audio and Video feedback.**

int state[5][5];

int goalstate1[5][5];

int goalstate2[5][5];

int Rblank,Cblank;

int TileRow[16],TileCol[16];

void \*Tile[17];

int D = 60; /\* Geometric size of each tile \*/

char Ans;

#include<stdlib.h>

#include<conio.h>

#include<dos.h>

#include<math.h>

#include<string.h>

#include<graphics.h>

#include<stdio.h>

union REGS ii,oo;

int tx,ty,button;

void store\_tiles(void);

void start\_init(void);

int tile\_click(void);

int move\_tile\_click(int);

void main()

{ int index,i,flag=0,toggle;

 int gd=VGA, gm=VGAHI;

 initgraph(&gd,&gm,"c:\\tc\\bgi"); randomize();

 music();

 check\_graphics\_error();

 store\_tiles();

 rectangle(10,10,630,460);

 rectangle(20,20,610,440);

 line(320,20,320,440);

 settextstyle(TRIPLEX\_FONT,HORIZ\_DIR,2);

 outtextxy(380,50,"15-Puzzle Design");

 outtextxy(330,80,"1. Click Left+Right To Exit");

 outtextxy(330,110,"2. Click Left To Move Tile");

 start\_init();

 reset\_mouse();

 set\_horiz\_limit(40,300);

 set\_vert\_limit(40,300);

 Ans='N';

 do {

if(Ans!='N')start\_init();

Ans='N';flag=0;

set\_position(50,50);

show\_pointer();

do {

 get\_position(); if(button==3){closegraph();restorecrtmode();exit(1);}

 if(button!=1){/\*setcolor(WHITE);

 rectangle(330,140,370,180);

 line(335,145,365,175);delay(100);

 setcolor(BLACK);

 rectangle(330,140,370,180);

 line(335,145,365,175);delay(100);\*/

 continue;

 }

 setcolor(WHITE);

 index=tile\_click();

 flag=0;

 if(index!=0){music();flag=move\_tile\_click(index);}

 }while(flag==0);

if(flag!=0)

 { music();

 outtextxy(140,300,"Puzzle Solved...");

 hide\_pointer();

 }

outtextxy(140,320,"Continue [Y/N]?");

/\* display of graphics cursor-Blinking Line \*/

while(!kbhit())

{ setcolor(WHITE);

 line(265,325,271,325);

 line(265,326,271,326);

 delay(1000);

 setcolor(BLACK);

 line(265,326,271,326);

 line(265,325,271,325);

 delay(1000);music(); /\*kaled();\*/

}

setcolor(WHITE);

Ans=getch(); setfillstyle(SOLID\_FILL,0);

 bar(135,300,280,340);

 /\*bar(345,345,420,420);\*/

}while(toupper(Ans)!='N');

 closegraph();restorecrtmode();

}/\*end of main \*/

void start\_init()

{

 int TileV[17],i,j,k;

 /\* set up goal states \*/

 k=0;

 for(i=1;i<=4;i++)

 { for(j=1;j<=4;j++)

{ k=k+1;

 goalstate1[i][j]=k;

 goalstate2[i][j]=16-k;

}

 }

 randomize();

 /\* clear arrays and move tiles off the board \*/

 for(i=1;i<=16;i++) TileV[i]=0;

 for(i=1;i<=4;i++)

 { for(j=1;j<=4;j++)state[i][j]=0;}

 /\* now set each tile to a different random number \*/ for(i=1;i<=4;i++)

 { for(j=1;j<=4;j++)

{ if((i\*j)>1)

 {while(state[i][j]==0)

 { k=random(16);

if(k>0 && k<16)

 { if(TileV[k]==0)

 { TileV[k]=1;

TileRow[k]=i;

TileCol[k]=j;

putimage(D\*j,D\*i,Tile[k],COPY\_PUT);

state[i][j]=k;

 }

 }

 }

 }

}

 }

 /\* set row and column of blank tile \*/

 Rblank=Cblank=1;

 putimage(D\*Cblank,D\*Rblank,Tile[16],COPY\_PUT);

}/\* end of start\_init();\*/

int tile\_click()

{ unsigned c=0,r=0,i;

 char str4[10],str5[10];

 for(i=1;i<=4;i++)

 { if( tx>=(D\*i) && tx<(D\*(i+1)) )c=i;

 if( ty>=(D\*i) && ty<(D\*(i+1)) )r=i;

 }

/\* itoa(c,str4,10); setfillstyle(SOLID\_FILL,0); bar(345,345,420,420);itoa(r,str5,10);

 outtextxy(350,350,str4);outtextxy(400,350,str5);\*/

 if(c==0 || r==0)return(0);

/\* itoa(state[r][c],str4,10); outtextxy(375,375,str4);\*/

 return(state[r][c]);

}

int move\_tile\_click(int index1)

{ char str1[50],str2[50],str3[50],str[50],blank[2]=" ";

 int dx,dxm,dy,dym,temp,neighbour,neighbour1,neighbour2,n1,n2,pi,pj;

 /\* shift tiles within row \*/

 if(TileRow[index1] == Rblank)

 { dx=TileCol[index1]-Cblank;

dxm=abs(dx);

switch(dxm)

{

 case 1 : /\* move one tile horizontally \*/

 state[Rblank][Cblank]=index1;

 hide\_pointer();

 putimage(Cblank\*D,Rblank\*D,Tile[index1],COPY\_PUT);

 temp=Cblank;

 Cblank=TileCol[index1];

 TileCol[index1]=temp;

 putimage(Cblank\*D,Rblank\*D,Tile[16],COPY\_PUT);

 state[Rblank][Cblank]=0;

 show\_pointer();

 break;

 case 2 : /\* move two tiles horizontally \*/

 neighbour=state[Rblank][Cblank+sign(dx)]; state[Rblank][Cblank]=neighbour;

 hide\_pointer();

 putimage(Cblank\*D,Rblank\*D,Tile[neighbour],COPY\_PUT);

 state[Rblank][Cblank+sign(dx)]=index1;

 putimage((Cblank+sign(dx))\*D,Rblank\*D,Tile[index1],COPY\_PUT);

 temp=Cblank;

 Cblank=TileCol[index1];

 TileCol[index1]=TileCol[neighbour];

 TileCol[neighbour]=temp;

 putimage(Cblank\*D,Rblank\*D,Tile[16],COPY\_PUT);

 state[Rblank][Cblank]=0;

 show\_pointer();

 break;

 case 3 : /\* move three tiles horizontally \*/

 neighbour1=state[Rblank][Cblank+sign(dx)];

 neighbour2=state[Rblank][Cblank+2\*sign(dx)];

 state[Rblank][Cblank]=neighbour1;

 hide\_pointer();

 putimage(Cblank\*D,Rblank\*D,Tile[neighbour1],COPY\_PUT);

 state[Rblank][Cblank+sign(dx)]=neighbour2;

 putimage((Cblank+sign(dx))\*D,Rblank\*D,Tile[neighbour2],COPY\_PUT);

 state[Rblank][Cblank+2\*sign(dx)]=index1;

 putimage((Cblank+2\*sign(dx))\*D,Rblank\*D,Tile[index1],COPY\_PUT);

 temp=Cblank;

 Cblank=TileCol[index1];

 TileCol[index1]=TileCol[neighbour2];

 TileCol[neighbour2]=TileCol[neighbour1];

 TileCol[neighbour1]=temp;

 putimage(Cblank\*D,Rblank\*D,Tile[16],COPY\_PUT);

 state[Rblank][Cblank]=0;

 show\_pointer(); break;

} /\* end of switch \*/

 } /\* end of if \*/

 /\* shift tiles within column \*/

if(TileCol[index1]==Cblank)

 { dy=TileRow[index1]-Rblank;

dym=abs(dy);

switch(dym)

{

 case 1 : /\* move one tile vertically \*/

 state[Rblank][Cblank]=index1;

 hide\_pointer();

 putimage(Cblank\*D,Rblank\*D,Tile[index1],COPY\_PUT);

 temp=Rblank;

 Rblank=TileRow[index1];

 TileRow[index1]=temp;

 putimage(Cblank\*D,Rblank\*D,Tile[16],COPY\_PUT);

 state[Rblank][Cblank]=0;

 show\_pointer();

 break;

 case 2 : /\* move two tiles vertically \*/

 neighbour=state[Rblank+sign(dy)][Cblank];

 state[Rblank][Cblank]=neighbour;

 hide\_pointer();

 putimage(Cblank\*D,Rblank\*D,Tile[neighbour],COPY\_PUT);

 state[Rblank+sign(dy)][Cblank]=index1;

 putimage(Cblank\*D,(Rblank+sign(dy))\*D,Tile[index1],COPY\_PUT);

 temp=Rblank;

 Rblank=TileRow[index1];

 TileRow[index1]=TileRow[neighbour]; TileRow[neighbour]=temp;

 putimage(Cblank\*D,Rblank\*D,Tile[16],COPY\_PUT);

 state[Rblank][Cblank]=0;

 show\_pointer();

 break;

 case 3 : /\* move three tiles vertically \*/

 neighbour1=state[Rblank+sign(dy)][Cblank];

 neighbour2=state[Rblank+2\*sign(dy)][Cblank];

 state[Rblank][Cblank]=neighbour1;

 hide\_pointer();

 putimage(Cblank\*D,Rblank\*D,Tile[neighbour1],COPY\_PUT);

 state[Rblank+sign(dy)][Cblank]=neighbour2;

 putimage(Cblank\*D,(Rblank+sign(dy))\*D,Tile[neighbour2],COPY\_PUT);

 state[Rblank+2\*sign(dy)][Cblank]=index1;

 putimage(Cblank\*D,(Rblank+2\*sign(dy))\*D,Tile[index1],COPY\_PUT);

 temp=Rblank;

 Rblank=TileRow[index1];

 TileRow[index1]=TileRow[neighbour2];

 TileRow[neighbour2]=TileRow[neighbour1];

 TileRow[neighbour1]=temp;

 putimage(Cblank\*D,Rblank\*D,Tile[16],COPY\_PUT);

 state[Rblank][Cblank]=0;

 show\_pointer();

 break;

} /\* end of switch \*/

 } /\* end of if \*/

 /\* Test to see if puzzle is solved \*/

 strcpy(str1," ");

 strcpy(str2," ");

 strcpy(str3," "); for(pi=1;pi<=3;pi++)

 { for(pj=1;pj<=4;pj++)

 { strcat(str1,itoa(state[pi][pj],str,10));

 strcat(str1,blank);

 strcat(str2,itoa(goalstate1[pi][pj],str,10));

 strcat(str2,blank);

 strcat(str3,itoa(goalstate2[pi][pj],str,10));

 strcat(str3,blank);

 }

 }

 pi=4;

 for(pj=1;pj<=3;pj++)

 { strcat(str1,itoa(state[pi][pj],str,10));

strcat(str1,blank);

strcat(str2,itoa(goalstate1[pi][pj],str,10));

strcat(str2,blank);

strcat(str3,itoa(goalstate2[pi][pj],str,10));

strcat(str3,blank);

 }

 /\* if puzzle is solved then beep and give suitable message \*/

 n1=strcmp(str1,str2);

 n2=strcmp(str1,str3);

 if(n1==0 || n2==0)

{ music();

 return(1);

}

 return(0);

} /\* end of move\_tile\_click() \*/

int sign(int p){ if(p<0) return(-1);

 else if(p>0) return(1);

 else return(0);

}

music()

{ unsigned x1;

 for(x1=1;x1<=5;x1++)

 { sound(random(500));

delay(50);

 }

 nosound();

 return 0;

}

check\_graphics\_error()

{ unsigned error;

 error=graphresult();

 if(error!=grOk)

 { closegraph();

restorecrtmode();

printf("%s",grapherrormsg(error));

exit(1);

 }

 return 0;

}

void store\_tiles(void)

{ int p1,p2,i,j,zi,zj,k=0,len;

 unsigned size;

 char st[3]=" ",temp[3]=" ";

size=imagesize(0,0,D,D);

for(zi=0;zi<4;zi++)

 { i=zi+1; for(zj=0;zj<4;zj++)

{j=zj+1;

k=k+1;

 strcpy(st,temp);

 itoa(k,st,10);

 len=strlen(st);

 if(len==1){st[2]='\0';st[1]=st[0];st[0]=32;}

 if((i==4) && (j==4))st[0]='\0';

 p1= D\*j+(j-1); p2=D\*i+(i-1);

 setlinestyle(SOLID\_LINE,0,THICK\_WIDTH);

 rectangle(p1,p2,p1+(D-1),p2+(D-1));

 settextstyle(SANS\_SERIF\_FONT,HORIZ\_DIR,2);

 outtextxy(D\*j+22,D\*i+22,st);

 Tile[k]=malloc(size);

 getimage(p1,p2,p1+(D-1),p2+(D-1),Tile[k]);

 }

 }

/\*getch();\*/

cleardevice();

}

/\* mouse functions \*/

reset\_mouse()

{

[ii.x.ax](http://ii.x.ax/)=0;

 int86(0x33,&ii,&oo); if([oo.x.ax](http://oo.x.ax/)==0)

 {

 outtextxy(getmaxx()/2-26\*4,10," ERROR mouse driver not installed");

 outtextxy(getmaxx()/2-26\*4,22," Installing.......");

 system("c:\\MOUSEG\\[mouse.com](http://mouse.com/)");

 delay(1000);

 cleardevice();

 }

 return 0;

}

show\_pointer()

{

[ii.x.ax](http://ii.x.ax/)=1;

 int86(0x33,&ii,&oo);

 return 0;

}

hide\_pointer()

{

[ii.x.ax](http://ii.x.ax/)=2;

 int86(0x33,&ii,&oo);

 return 0;

}

set\_horiz\_limit(int minx,int maxx)

{

[ii.x.ax](http://ii.x.ax/)=7;

[ii.x.cx](http://ii.x.cx/)=minx;

 ii.x.dx=maxx;

 int86(0x33,&ii,&oo); return 0;

}

set\_vert\_limit(int miny,int maxy)

{

[ii.x.ax](http://ii.x.ax/)=8;

[ii.x.cx](http://ii.x.cx/)=miny;

 ii.x.dx=maxy;

 int86(0x33,&ii,&oo);

 return 0;

}

get\_position()

{

[ii.x.ax](http://ii.x.ax/)=3;

 int86(0x33,&ii,&oo);

 tx=[oo.x.cx](http://oo.x.cx/);

 ty=oo.x.dx;

 button=oo.x.bx;

 return 0;

}

set\_position(int xx,int yy)

{

[ii.x.ax](http://ii.x.ax/)=4;

[ii.x.cx](http://ii.x.cx/)=xx;

 ii.x.dx=yy;

 int86(0x33,&ii,&oo);

 return 0;

}set\_excl\_area(int left,int top,int right,int bottom)

{ [ii.x.ax](http://ii.x.ax/)=16;

[ii.x.cx](http://ii.x.cx/)=left;

 ii.x.dx=top;

[ii.x.si](http://ii.x.si/)=right;

 ii.x.di=bottom;

 int86(0x33,&ii,&oo);

 return 0;

}

enable\_driver()

{

[ii.x.ax](http://ii.x.ax/)=32;

 int86(0x33,&ii,&oo);

 return 0;

}

disable\_driver()

{

[ii.x.ax](http://ii.x.ax/)=31;

 int86(0x33,&ii,&oo);

 return 0;

}

**Move Tiles:**

**A picture containing application

Description automatically generated**

**Calendar

Description automatically generated**

**Program 17: [Back-Face Elimination method] Develop a program using 3D graphics to implement back-face hidden surface elimination method and various 3D transformations.**

#include <graphics.h>

#include <math.h>

#include <stdio.h>

#include <dos.h>

/\* define global variables \*/

int rho=40,d=250;

/\* define the 10 corners of the house \*/

int v[10][3]=

{

{5,7,-5}, {5,7,5}, {5,-7,5}, {5,-7,-5},

{-5,7,-5}, {-5,-7,-5}, {-5,-7,5}, {-5,7,5},

{0,7,8}, {0,-7,8}

};

int sv[10][2],rot=1;

int nps[7]={5,6,5,6,5,5,5};

static int s[7][6] = {

{0,1,2,3,0}, {0,4,7,8,1,0}, {4,5,6,7,4},

{3,2,9,6,5,3}, {2,1,8,9,2}, {6,9,8,7,6},

{0,3,5,4,0}

};

int n[7][3], e[12][3];

int originx=320,originy=150, page=0;

char c;

double theta=.7,phi=0.9;

float s1,s2,c1,c2;

main()

{

int gd=VGA,gm=VGAMED;

initgraph(&gd,&gm,"c:\\turboc3\\bgi");

for(;;)

{

if(kbhit()) keypressed();

rotation();

pageflip();

generatepoint();

normvector();

visibility();

drawedges();

}

}/\* end of main \*/

screenxy(int x, int y, int z, int \*scx, int \*scy)

{

float xe,ye,ze;

xe=-x\*s1 + y\*c1;

ye=-x\*c1\*c2 - y\*s1\*c2 + z\*s2;

ze=-x\*s2\*c1 - y\*s1\*s2 - z\*c2 + rho;

\*scx = (d\*xe)/ze;

\*scy = (d\*ye)/ze;

return 0;

}

rotation()

{

switch(rot)

{ case 1 : phi +=.1;break;

case 2 : theta +=.1;break;

case 3 : phi -=.1;break;

case 4 : theta -=.1;break;

case 5 : rho -=800;break;

case 6 : rho +=800;break;

case 7 : d -=800;break;

}

s1=sin(theta); s2=sin(phi);

c1=cos(theta); c2=cos(phi);

return 0;

}

keypressed()

{

c=getch();

if(c==27) { restorecrtmode();exit(0);}

if(c==0)

{

c=getch();

switch(c)

{ case 72 : rot=1;break;

case 77 : rot=2;break;

case 80 : rot=3;break;

case 75 : rot=4;break;

case 59 : rot=5;break;

case 60 : rot=6;break;

case 61 : rot=7;break;

}

}

return 0;

}

pageflip()

{ setvisualpage(page);

page=1-page;

setactivepage(page);

clearviewport();delay(200);

return 0;

}

generatepoint()

{

int i,x,y,z;

for(i=0;i<10;i++)

{ x=v[i][0];

y=v[i][1];

z=v[i][2];

screenxy(x,y,z,&sv[i][0],&sv[i][1]);

sv[i][0]=originx+sv[i][0];

sv[i][1]=originy-sv[i][1];

}

return 0;

}

normvector()

{

int i,j1,j2,k,u1,u2,u3,v1,v2,v3;

for(i=0;i<7;i++)

{ j1=s[i][1]; j2=s[i][2]; k=s[i][0];

u1=v[j1][0]-v[k][0];

u2=v[j1][1]-v[k][1];

u3=v[j1][2]-v[k][2];

v1=v[j2][0]-v[k][0];

v2=v[j2][1]-v[k][1];

v3=v[j2][2]-v[k][2];

n[i][0]=u2\*v3-v2\*u3;

n[i][1]=u3\*v1-v3\*u1;

n[i][2]=u1\*v2-v1\*u2;

}

return 0;

}

visibility()

{

int m,i,e2,e1,j,k,flag;

float xe,ye,ze,wx,wy,wz,dot;

xe=rho\*s2\*c1; ye=rho\*s2\*s1; ze=rho\*c2;

m=0;

for(i=0;i<7;i++)

{

e2=s[i][0];

wx=xe-v[e2][0];

wy=ye-v[e2][1];

wz=ze-v[e2][2];

dot=n[i][0]\*wx+n[i][1]\*wy+n[i][2]\*wz;

if(dot<=0)continue;

e1=s[i][0];

for(j=1;j<=nps[i];j++)

{

e2=s[i][j];

for(k=0;k<=m;k++)

{ flag=0;

if(e[k][0]==e2 && e[k][1]==e1)

{

e[k][2]=2;

flag=1;

}

if(flag==1)break;

}/\* end of k\*/

if(flag==0){

e[m][0]=e1;

e[m][1]=e2;

e[m][2]=1;

m=m+1;

}

e1=e2;

}/\* end of j \*/

}/\* end of i \*/

return 0;

}/\* end of function \*/

drawedges()

{

int i,j,k;

for(i=0;i<12;i++)

{

if(e[i][2]==0)continue;

j=e[i][0];k=e[i][1];

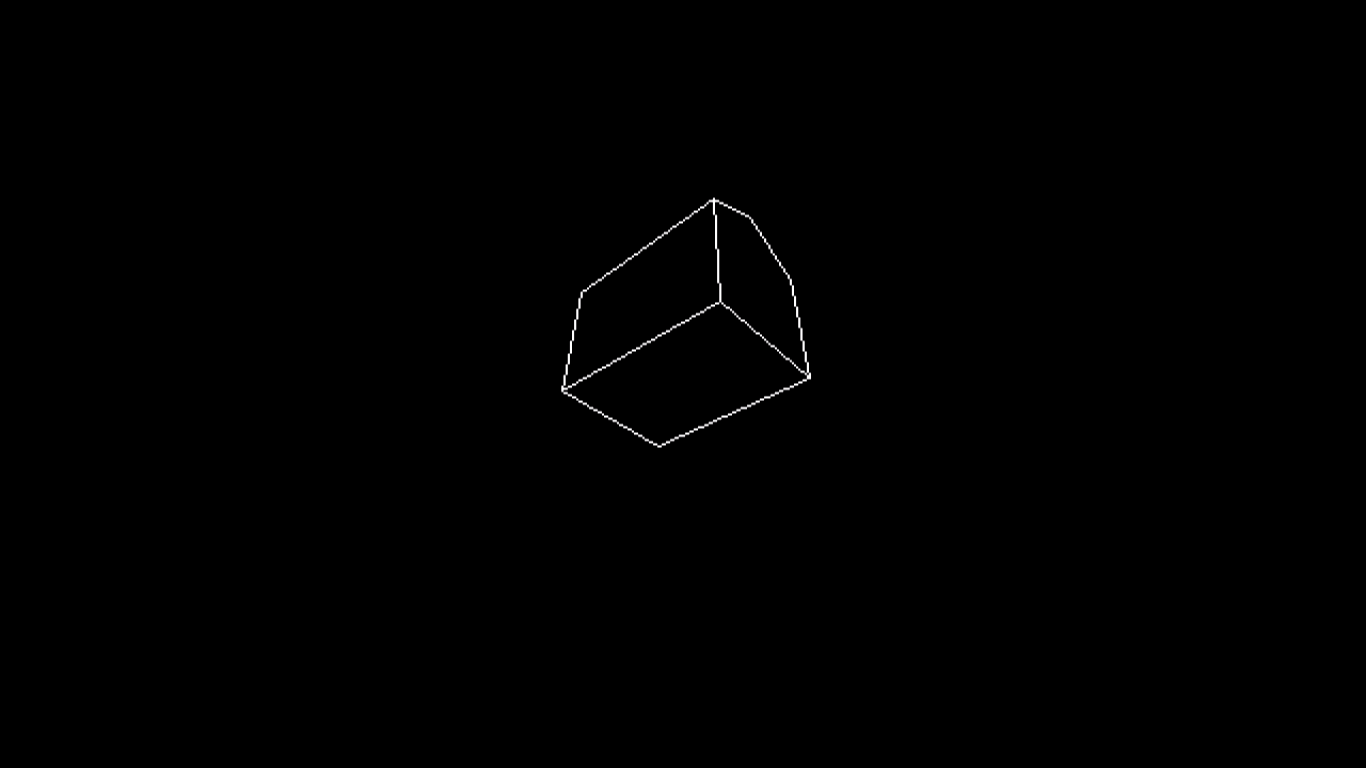
setcolor(WHITE);

line(sv[j][0],sv[j][1],sv[k][0],sv[k][1]);

}

return 0;

}

**PRESS ARROW KEYS TO SHOW DIFFERENT OUTPUT**

**Question 18:What is OpenGL and what is GLUT library? Name and explain the seven major groups of OpenGL API functions, with appropriate examples for each function. Also, explain the purpose of void glutDisplayFunc(void ('func)(void)); and glutMainLoop();**

**Ans :**OpenGl - • OpenGL (Open Graphics Library) is a software interface to the graphics hardware.

•OpenGL is specifically designed for efficient processing of three dimensional applications.

•Graphics functions in any package are defined as the set of specifications that are independent of any programming language. A language binding is then defined for a particular high level programming language.

•The OpenGL bindings are done with C, C++, Fortran or Ada.

GLUT library -

The OpenGL Utility Toolkit (GLUT) is a library of utilities for OpenGL programs, which primarily perform system-level I/O with the host operating system. Functions performed include window definition, window control, and monitoring of keyboard and mouse input. Routines for drawing a number of geometric primitives (both in solid and wireframe mode) are also provided, including cubes, spheres and the Utah teapot. GLUT also has some limited support for creating pop-up menus.

The two aims of GLUT are to allow the creation of rather portable code between operating systems (GLUT is cross-platform) and to make learning OpenGL easier. Getting started with OpenGL programming while using GLUT often takes only a few lines of code and does not require knowledge of operating system–specific windowing APIs

following seven major groups of OpenGL API functions, with examples for each function.

1. Primitive functions
2. Attribute functions
3. Viewing functions
4. Transformation functions
5. Input functions
6. Control functions
7. Query functions.

Primitive functions - 1. Define the low level objects or atomic entities that our system can display.

2.Depending on the APl, the primitives can include points, lines, segments, polygons, pixels, text and various types of curves and surfaces.

Example -glBegin(GL\_POINTS); glVertex2f(100.0, 200.0); glEnd(); Attribute functions –

1. To perform operations ranging from choosing the color with which we display a line segment.

Eg- glColor3f(1.0, 0.0, 0.0); glLineWidth(1.0);

1. To pick a pattern with which to fill the inside of a polygon.
2. To select a typeface(font) for the title on a graph. Viewing Functions-

1. The viewing functions allow us to specify various views, although APls differ in the degree of flexibility they provide in choosing a view.

glViewport (xvmin, yvmin, vpWidth, vpHeight); gluOrtho2D (xwmin, xwmax, Xwmin, ywmax); glulookAt(.);

Transformation functions –

1. Transformation function that allows carrying out transformations of objects, such as rotation, translation and scaling.

* gLTranslatef (tx, ty, tz);
* gLRotatef (theta, vx, vy, vz);
* glScalef (sx, sy, sz); Input Function-

1 Allows us to deal with the diverse forms of input that characterize modern graphics systems

void glutKeyboardFunc(void ("func) (unsigned char key, int x, int y); Control function –

1. Enable us to communicate with the window system, to initialize our programs, and to deal with any errors that take place during the execution of our programs.
2. glutinitWindowSize(500, 500); Query Function-
3. How many colors are supported or the size of the display 2.Camera parameters or values in the frame buffer 3.glGetintegerv(GL\_RED\_BITS)- function returns the value or values of a selected parameter. Number of bits per R

Purpose of void glutDisplayFunc(void ('func)(void))

glutDisplayFunc(void (\*func)(void)) is the first and most important event callback function . Whenever GLUT determines that the contents of the window need to be redisplayed, the callback function registered by glutDisplayFunc() is executed.

Therefore, you should put all the routines you need to redraw the scene in the display callback

Purpose of glutMainLoop()-

glutMainLoop() is a function that loops within itself, processing events and triggering your callback functions when necessary (glutDisplayFunc, etc). After a call to glutMainLoop(), you have no control over the execution of your program besides the callback functions you initialized before the call.

And another thing, the glutDisplayFunc() callback only gets triggered when your app's window receives a refresh event, unless you explicitly post a redisplay. If you want your display function to be called repeatedly (i.e. for a game), you need to set your glutIdleFunc to post a redisplay.

Something like this:

int main(){ /\* ... / glutIdleFunc(updateGame); / ... \*/ glutMainLoop();}void updateGame(){ glutPostRedisplay

**Question 19:Explain the working of Cohen-Sutherland Line clipping algorithm with the help of an example. How does this algorithm identify trivally-in and trivally-out lines?**

**Ans : Line Clipping:**

The concept of line clipping is the same as point clipping. In line clipping, we will cut the portion of the line which is outside of the window and keep only the portion that is inside the window.

#### Cohen-Sutherland Line Clippings:

* + This algorithm uses the clipping window as shown in the following figure.

A picture containing text, antenna

Description automatically generated

* + We will use 4-bits to divide the entire region. These 4 bits represent the Top, Bottom, Right, and Left of the region as shown in the following figure. Here, the

Diagram

Description automatically generated with low confidenceTOP and LEFT bit is set to 1 because it is the TOP-LEFT corner.

* + There are 3 possibilities for the line:
  + Line can be completely inside the window (This line should be accepted).
  + Line can be completely outside of the window (This line will be completely removed from the region).
  + Line can be partially inside the window (We will find intersection point and draw only that portion of line that is inside region).

Algorithm:

Step 1 − Assign a region code for each endpoints.

Step 2 − If both endpoints have a region code 0000 then accept this line. Step 3 − Else, perform the logical ANDoperation for both region codes.

Step 3.1 − If the result is not 0000, then reject the line. Step 3.2 − Else you need clipping.

Step 3.2.1 − Choose an endpoint of the line that is outside the window.

Step 3.2.2 − Find the intersection point at the window boundary (based on region code). Step 3.2.3 − Replace endpoint with the intersection point and update the region code.

Step 3.2.4 − Repeat step 2 until we find a clipped line either trivially accepted or trivially rejected.

Step 4 − Repeat step 1 for other lines. Conclusion:

In summary, the C-S algorithm is efficient when outcode testing can be done cheaply (for example, by doing bitwise operations in assembly language) and trivial acceptance or rejection is applicable to the majority of line segments .(For example, large windows - everything is inside , or small windows - everything is outside).

**Question 20: What is the objective of interactive computer graphics? Develop an animated algorithm for the demonstration of Bubble Sort.**

**Ans:**Regarding the interactivity of computer science algorithm animations, there were made following recommendations:

* The control of the animations should be flexible. Except the controls for starting and stopping the animations, it is suggested to add control buttons for stepping

forwards or backwards in the visualization. Mayer also emphasized the importance of the

possibility to stop the animation. The default time between the logically related

parts of the visualized processes may not be enough for every student: some students need more time to think over and comprehend the steps of the algorithms.

Even better solution is when the animations automatically stop after few logically

related steps. Students can think of the visualized processes and continue observing the next steps of the animations by pressing a control button. In this case,

students do not have to think about the moments, when it is worth to stop the animations, thus they can concentrate more on the visualized process

* The speed of the animations should be varying, or should the user be able to change the speed. Different parts of the animations require different speed, e.g. in sorting algorithms the most important parts are the comparisons and swaps so the animations should be displayed in a slower speed, or they should be stopped

during these steps. When we try to watch the control variables of cycles during the sorting algorithms, the first few changes of these variables are the most important

to understand the main ideas of the algorithms so later the animations can be displayed at a higher speed . It is also a good solution if

students can change the speed of the animations.

* Modifying or changing the input data in the animations helps students to better understanding of the behavior of the algorithms. Entering their own input

data encourages students to participate more actively. By modifying data in the

animations, students can experiment with the visualization and observe different behaviors of the algorithms The results of pedagogical experiments show that students

who may enter own input data or modify data in the animations get significantly better results in test Experimenting with the animations may be even more intriguing when not only the input data can be modified, but the values of the variables can be changed during the animations.

* Animations should adapt to students’ knowledge level, or different animations of the same algorithms are recommended to use. For novice students, it might be hard to understand the algorithms, if the animations are too detailed, they contain many windows, or there are too many options to be set up. For beginners, it is better to use simple animations with predefined data sets. However, for advanced students, it might be valuable if they can enter their own data, modify some options, or observe detailed views of the animations .
* Questioning students during the animations might be useful. Asking questions related to the steps of the animations encourages students to pay more attention . On the other hand, examinations during the animations might distract attention from the visualized processes. This is the reason why it is important to choose the right moment and the right form of the questions. Students may be asked by the animation software, they can get questions on papers, in voice by a lecturer, or the questions can be part of electronic textbooks containing the animation. To get the answers to the questions is not important in every case, sometimes it is enough if students start thinking about the possible solutions .
* Animations should be entertaining. Students learn easier if the animations entertain them from the beginning until the end. It is not practical to repeat consecutively the same, long steps inserted game elements into the animations, e.g. game activities like “spot the error”, “predict the output” and “sort in order”. All these activities improve students’ level of critical thinking.

Bubble sort: human brain can easily process visuals in spite of long codes to understand the algorithms. In this article, [Bubble sort](https://www.geeksforgeeks.org/bubble-sort/) **visualization** has been implemented using [graphics.h](https://www.geeksforgeeks.org/add-graphics-h-c-library-gcc-compiler-linux/) **library**. As we all know that bubble sort swaps the adjacent elements if they are unsorted and finally the larger one is shifted towards the end of the array in each pass.

// C++ program for visualization of bubble sort

#include "graphics.h" #include <bits/stdc++.h>

using namespace std;

// Initialize the size

// with the total numbers to sorted

// and the gap to be maintained in graph vector<int> numbers;

int size = 200; int gap = 4;

// Function for swapping the lines graphically void swap(int i, int j, int x, int y)

{

// Swapping the first line with the correct line

// by making it black again and then draw the pixel

// for white color.

setcolor(GREEN); line(i, size, i, size - x); setcolor(BLACK); line(i, size, i, size - x); setcolor(WHITE); line(i, size, i, size - y);

// Swapping the first line with the correct line

// by making it black again and then draw the pixel

// for white color. setcolor(GREEN);

line(j, size, j, size - y); setcolor(BLACK); line(j, size, j, size - y); setcolor(WHITE); line(j, size, j, size - x);

}

// Bubble sort function void bubbleSort()

{

int temp, i, j;

for (i = 1; i < size; i++) {

for (j = 0; j < size - i; j++) {

if (numbers[j] > numbers[j + 1]) { temp = numbers[j]; numbers[j] = numbers[j + 1]; numbers[j + 1] = temp;

// As we swapped the last two numbers

// just swap the lines with the values.

// This is function call

// for swapping the lines swap(gap \* j + 1,

gap \* (j + 1) + 1, numbers[j + 1], numbers[j]);

}

}

}

}

// Driver program int main()

{

// auto detection of screen size int gd = DETECT, gm;

int wid1;

// Graph initialization initgraph(&gd, &gm, NULL);

// setting up window size (gap\*size) \* (size) wid1 = initwindow(gap \* size + 1, size + 1); setcurrentwindow(wid1);

// Initializing the array

for (int i = 1; i <= size; i++) numbers.push\_back(i);

// Find a seed and shuffle the array

// to make it random.

// Here different type of array

// can be taken to results

// such as nearly sorted, already sorted,

// reverse sorted to visualize the result unsigned seed

= chrono::system\_clock::now()

.time\_since\_epoch()

.count();

shuffle(numbers.begin(),

numbers.end(), default\_random\_engine(seed));

// Initial plot of numbers in graph taking

// the vector position as x-axis and its

// corresponding value will be the height of line. for (int i = 1; i <= gap \* size; i += gap) {

line(i, size, i, (size - numbers[i / gap]));

}

// Delay the code delay(200);

// Call sort bubbleSort();

for (int i = 0; i < size; i++) {

cout << numbers[i] << " ";

}

cout << endl;

// Wait for sometime . delay(5000);

// Close the graph closegraph();

return 0;

}

**Question 21:What are the various 2-dimensional transformations? Explain with the help of suitable sketches. Give their matrix representations. Summarize briefly the problem solved by the introduction of homogeneous coordinates.**

**Prove that use of 2-D rotation and scaling commute if S, = S, or 0 = nr for integral n.**

**Ans: Various 2-dimensional transformations -**

1. [**Translation**](https://www.javatpoint.com/computer-graphics-translation)
2. [**Scaling**](https://www.javatpoint.com/computer-graphics-scaling)
3. [**Rotating**](https://www.javatpoint.com/computer-graphics-rotation)
4. [**Reflection**](https://www.javatpoint.com/computer-graphics-reflection)
5. [**Shearing**](https://www.javatpoint.com/computer-graphics-shearing)

Translation -It is the straight line movement of an object from one position to another is called Translation. Here the object is positioned from one coordinate location to another.

Translation of point:To translate a point from coordinate position (x, y) to another (x1 y1), we add algebraically the translation distances Tx and Ty to original coordinate.

x1=x+Tx y1=y+Ty

The translation pair (Tx,Ty) is called as shift vector.

**Matrix for Translation:**

Chart, box and whisker chart

Description automatically generated

Matrix for Translation:

Scaling: -It is used to alter or change the size of objects. The change is done using scaling factors. There are two scaling factors, i.e. Sx in x direction Sy in y-direction. I

f the

original position is x and y. Scaling factors are Sx and Sy then the value of coordinates

1

after scaling will be x and y1.

If the picture to be enlarged to twice its original size then Sx = Sy =2. If Sxand Sy are not equal then scaling will occur but it will elongate or distort the picture.

If scaling factors are less than one, then the size of the object will be reduced. If scaling factors are higher than one, then the size of the object

the scaling will be enlarged.Shape, square

Description automatically generated

Matrix for Scaling:

Rotation:It is a process of changing the angle of the object. Rotation can be clockwise or anticlockwise. For rotation, we have to specify the angle of rotation and rotation point. Rotation point is also called a pivot point. It is print about which object is rotated.

Types of Rotation:

1. Anticlockwise
2. Counterclockwise

A picture containing chart

Description automatically generated

Matrix for rotation is a clockwise direction.

Matrix for rotation is an anticlockwise direction.

Matrix for homogeneous co-ordinate rotation (clockwise) Matrix for homogeneous co-ordinate rotation (anticlockwise)

#### Reflection:It is a transformation which produces a mirror image of an object. The mirror image can be either about x-axis or y-axis. The object is rotated by180°.

Types of Reflection:

1. Reflection about the x-axisReflection about the y-axis
2. Reflection about an axis perpendicular to xy plane and passing through the origin
3. Reflection about line y=x

**Reflection about x-axis:** The object can be reflected about x-axis with the help of the following matrix

Shape

Description automatically generated

**Reflection about y-axis:** The object can be reflected about y-axis with the help of following transformation matrix

A picture containing text, antenna

Description automatically generated

#### 3. Reflection about an axis perpendicular to xy plane and passing through origin:

In the matrix of this transformation is given below

Diagram, shape, polygon

Description automatically generated

In this value of x and y both will be reversed. This is also called as half revolution about the origin.

1. **Reflection about line y=x:** The object may be reflected about line y = x with the help of following transformation matrix

Diagram

Description automatically generated

#### Shearing:It is transformation which changes the shape of object. The

**sliding of layers of object occur. The shear can be in one direction or in two directions.**

**Shearing in the X-direction:** In this horizontal shearing sliding of layers occur. The homogeneous matrix for shearing in the x-direction is shown below

Diagram

Description automatically generated:

**Shearing in the Y-direction:** Here shearing is done by sliding along vertical or y-axis.

#### Shearing in X-Y directions –

*Prove that use of 2-D rotation and scaling commute if S, = S, or 0 = nr for integral n.*

Scaling: -It is used to alter or change the size of objects. The change is done using scaling factors. There are two scaling factors, i.e. Sx in x direction Sy in y-direction. If the original position is x and y. Scaling factors are Sx and Sy then the value of coordinates after scaling will be x1 and y1.

If the picture to be enlarged to twice its original size then Sx = Sy =2. If Sx And Sy are not equal then scaling will occur but it will elongate or distort the picture.

If scaling factors are less than one, then the size of the object will be reduced. If scaling factors are higher than one, then the size of the object the scaling will be enlarged.

**Question 22.Describe the Bresenham's Circle drawing algorithm. How does it differ from the mid-point circle- drawing algorithm? What efficiencies are achieved by this difference? You may use a diagram to aid your answer. Using the mid-point circle-drawing algorithm, draw a circle with centre as (2, 5) and radius as 6.**

**Ans**:As our computer screen is made up of pixels organised in the form of matrix, it becomes difficult to display a continuous and smooth arc on the computer screen.

Thus, to draw a circle on the computer screen we make use of the properties of the frame buffer i.e. we always select the pixel which is nearest to the printed pixel.

This can be achieved by using two algorithms:

1. Bresenham’s circle drawing algorithm
2. Mid-Point circle drawing algorithm

**Bresenham’s circle drawing algorithm:**

This algorithm uses the key feature of circle that it is highly symmetric. So, for whole 360 degree of circle we will divide it in 8-parts each octant of 45 degree. In order to do that we will use Bresenham’s Circle Algorithm for calculation of the locations of the pixels in the first octant of 45 degrees. It assumes that the circle is centered on the origin. So for every pixel (x, y) it calculates, we draw a pixel in each of the 8 octants of the circle as shown below :

Chart, diagram, pie chart

Description automatically generated

In order to calculate the next pixel location from a previously known pixel location (x, y). In Bresenham’s algorithm at any point (x, y) we have two option either to choose the next pixel in the east i.e. (x+1, y) or in the south east i.e. (x+1, y-1).

Chart, pie chart

Description automatically generated

And this can be decided by using the decision parameter d as:

If d > 0, then (x+1, y-1) is to be chosen as the next pixel as it will be closer to the arc.

else (x+1, y) is to be chosen as next pixel.

Now to draw the circle for a given radius ‘r’ and centre (xc, yc) We will start from (0, r) and move in first quadrant till x=y (i.e. 45 degree). We should start from listed initial conditionNow for each pixel, we will do the following operations: Set initial values of (xc, yc) and (x, y)

Set decision parameter d to d = 3 – (2 \* r).

call drawCircle(int xc, int yc, int x, int y) function. Repeat steps 5 to 8 until x < = y

Increment value of x.

If d < 0, set d = d + (4\*x) + 6

Else, set d = d + 4 \* (x – y) + 10 and decrement y by 1. call drawCircle(int xc, int yc, int x, int y) function

where drawCircle() function is given as:

// function to draw all other 7 pixels

// present at symmetric position drawCircle(**int** xc, **int** yc, **int** x, **int** y)

{

putpixel(xc+x, yc+y, RED); putpixel(xc-x, yc+y, RED); putpixel(xc+x, yc-y, RED);

putpixel(xc-x, yc-y, RED); putpixel(xc+y, yc+x, RED); putpixel(xc-y, yc+x, RED); putpixel(xc+y, yc-x, RED); putpixel(xc-y, yc-x, RED);

}

Difference:

As **Bresenham’s** is the optimised form of midpoint circle drawing algorithm and it prevents from calculation of floating-point numbers. In mid-point circle drawing algorithm, decision parameter depends on previous decision parameter and corresponding pixels whereas in bresenham decision parameter only depends on previous decision parameter.

Due to these differences bresenham’s circle drawing algorithm consumes less time as well as memory. It is more efficient and accurate too and also produces much smoother circle.

Numerical:

S-1; Enter the center h=2, k=5 and radius r=6.

S-2: Find d= 1-r, so d=1-6= -5 and take x = 0, y = 6 (Initial Points as x=0 & y=radius)

S-3: If (d>=0) then x = x +1 and y = y -1; **d = d + 2\*(x-y) + 5;**

S-4: if (d <0) then x =x +1 and **d = d +2\*x + 3;**

## **Where to stop:** x or y = 6/sqrt (2) which is 6/1.414= approx. = 4. So, either x or y reaches 4, the algorithm will stop.

Midpoint Circle Algorithm (r=6, h=2, k=5) Step Number X Y d Pixel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 0 | 6 | -5 | (x +h ,y +k) = 2,11 |
| 2 | 1 | 6 | 0 | 3,11 |
| 3 | 2 | 5 | -1 | 4,10 |
| 4 | 3 | 5 | 8 | 5,10 |
| 5 | 4 | 4 | stop | – |

Rest of the points will be plotted using 8-Way Symmetry. There would be 4\*8=32 Points would be created in total, which on joining will look like circle