Swami Keshvanand Institute of Technology, Management & Gramothan,
Ramnagaria, Jagatpura, Jaipur-302017, INDIA
Approved by AICTE, Ministry of HRD, Government of India
Recognized by UGC under Section 2(f) of the UGC Act, 1956
Tel. : +91-0141- 5160400 Fax: +91-0141-2759555
E-mail: info@skit.ac.in Web: www.skit.ac.in

A

Lab Manual

on

Computer Graphics & Multimedia Lab (5CS4-21)

Programme: B.Tech.
Semester: V
Session: 2021-22.

(Name : Harpreet Singh Gill)
(Designation : Associate Professor)
(Branch : CSE)

Swami Keshvanand Institute of Technology, Management & Gramothan

Vision and Mission of Institute

Vision: "To promote higher learning in technology and industrial research to make our country a global player."

Mission: "To promote quality education, training and research in the field of engineering by establishing effective interface with industry and to encourage the faculty to undertake industry sponsored projects for the students."

Vision of CSE Department

Vision of CSE department is to:

V1: Produce quality computer engineers trained in latest tools and technologies.

V2: Be a leading department in the region and country by imparting in-depth knowledge to the students in emerging technologies in computer science & engineering.

Mission of CSE Department

Mission of CSE department is to:

Deliver resources in IT enable domain through:

M1: Effective Industry interaction and project-based learning.

M2: Motivating our students for employability, entrepreneurship, research and higher education.

M3: Providing excellent engineering skills in a state-of-the art infrastructure.

Program Educational Objectives of CSE department

The graduates of CSE program will be:

PEO1: Prepared to be employed in IT industries and be engaged in learning, understanding, and applying new ideas.

PEO2: Prepared to be responsible professionals in their domain of interest.

PEO3: Able to apply their technical knowledge as practicing professionals or engaged in higher education.

PEO4: Able to work efficiently as an individual and in a professional team environment.

Program Outcomes of CSE Department

PO 1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization for the solution of complex engineering problems.

PO 2: Problem analysis: Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO 3: Design/Development of Solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for public health and safety, and cultural, societal, and environmental considerations.

PO 4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO 5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO 6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO 7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO 8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO 9: Individual and teamwork: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO 10: Communication: Communicate effectively on complex engineering activities with the engineering community and with the society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO 11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO 12: Life-long learning: Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes of CSE Department

PSO1: Core Engineering Skills: Exhibit fundamental concepts of Data Structures, Databases, Operating Systems, Computer Network, Theory of Computation, Advanced Programming and Software Engineering.

PSO2: Standard Software Engineering practices: Demonstrate an ability to design, develop, test, debug, deploy, analyze, troubleshoot, maintain, manage and secure a software.

PSO3: Future Endeavors: Recognize the need to have knowledge of higher education institutions/ organizations/ companies related to computer science & engineering.

5CS4-21: Computer Graphics & Multimedia Lab

Credit: 1 **Max. Marks:50 (IA:30, ETE:20)**
0L+0T+2P **End Term Exam: 2 Hours**

SN	List of Experiments
1	Implementation of Line, Circle and ellipse attributes
2	To plot a point (pixel) on the screen
3	To draw a straight line using DDA Algorithm
4	Implementation of mid-point circle generating Algorithm
5	Implementation of ellipse generating Algorithm
6	Two Dimensional transformations - Translation, Rotation, Scaling, Reflection, Shear
7	Composite 2D Transformations
8	Cohen Sutherland 2D line clipping and Windowing
9	Sutherland – Hodgeman Polygon clipping Algorithm
10	Three dimensional transformations - Translation, Rotation, Scaling
11	Composite 3D transformations
12	Drawing three dimensional objects and Scenes
13	Generating Fractal images

S. No.	Experiment
1.a	WAP to print Hello World!!
1.b	WAP to find the max of 3 numbers
1.c	WAP to sort 5 numbers in ascending order
1.d	WAP to swap values of 2 variables without using 3rd variable
2.a	WAP to implement primitive graphic functions
2.b	WAP to draw National Flag
3.a	WAP to draw moving car
3.b	WAP to draw blinking star
4	WAP to implement DDA Line Algorithm
5.	WAP to implement Bresenham's Line drawing algorithm
6.	WAP to implement midpoint circle algorithm
7.	WAP to implement midpoint ellipse algorithm
8.	WAP to implement 2D transformation: a.) Translation b.) Scaling c.) Rotation
9.	WAP to implement 2D transformation: a.) Reflection b.) Shearing
10.	WAP to implement 3D transformation: a.) Translation b.) Scaling c.) Rotation
11.	WAP to implement 3D transformation: a.) Reflection b.) Shearing
12.	WAP to implement Cohen-Sutherland Line Clipping Algorithm
13.	WAP to implement Sutherland-Hodgeman Polygon clipping Algorithm
14.	WAP to generate fractal images

Introduction (Zero Lab)

Aim: Study of basic graphics function

1) arc

Declaration :- void arc(int x, int y, int stangle, int endangle, int radius);

arc function is used to draw an arc with center (x,y) and stangle specifies starting angle, endangle specifies the end angle and last parameter specifies the radius of the arc. Arc function can also be used to draw a circle but for that starting angle and end angle should be 0 and 360 respectively.

2) bar

Declaration :- void bar(int left, int top, int right, int bottom);

Bar function is used to draw a 2-dimensional, rectangular filled in bar . Coordinates of left top and right bottom corner are required to draw the bar. Left specifies the X-coordinate of top left corner, top specifies the Y-coordinate of top left corner, right specifies the Xcoordinate of right bottom corner, bottom specifies the Y-coordinate of right bottom corner

3) circle

Declaration :- void circle(int x, int y, int radius);

circle function is used to draw a circle with center (x,y) and third parameter specifies the radius of the circle. The code given below draws a circle.

4) closegraph

Declaration :- void closegraph();

closegraph function closes the graphics mode, deallocates all memory allocated by graphics system and restores the screen to the mode it was in before you called initgraph.

5) ellipse

Declarations :-

void ellipse(int x, int y, int stangle, int endangle, int xradius, int yradius);

Ellipse is used to draw an ellipse (x,y) are coordinates of center of the ellipse, stangle is the starting angle, end angle is the ending angle, and fifth and sixth parameters specifies the X and Y radius of the ellipse. To draw a complete ellipse strangles and end angle should be 0 and 360 respectively.

6) floodfill

Declaration :- void floodfill(int x, int y, int border);

floodfill function is used to fill an enclosed area. Current fill pattern and fill color is used to fill the area.(x, y) is any point on the screen if (x,y) lies inside the area then inside will be filled otherwise outside will be filled,border specifies the color of boundary of area.

7) getbkcolor

Declaration : int getbkcolor();

getbkcolor function returns the current background color

8) getcolor

Declaration : int getcolor();

getcolor function returns the current drawing color.

9) line

Declaration: - void line(int x1, int y1, int x2, int y2);

line function is used to draw a line from a point(x1,y1) to point(x2,y2) i.e. (x1,y1) and (x2,y2) are end points of the line. The code given below draws a line.

10) outtext

Declaration :- void outtext(char *string);

outtext function displays text at current position.

11) outtextxy:: outtextxy function display text or string at a specified point(x,y) on the screen.

Declaration :- void outtextxy(int x, int y, char *string);

x, y are coordinates of the point and third argument contains the address of string to be displayed.

12) rectangle

Declaration :- void rectangle(int left, int top, int right, int bottom);

rectangle function is used to draw a rectangle. Coordinates of left top and right bottom corner are required to draw the rectangle. left specifies the X-coordinate of top left corner, top specifies the Y-coordinate of top left corner, right specifies the X-coordinate of right bottom corner, bottom specifies the Y-coordinate of right bottom corner.

13) setbkcolor

Declaration :- void setbkcolor(int color);

setbkcolor function changes current background color e.g. setbkcolor(YELLLOW) changes

the current background color to YELLOW. Remember that default drawing color is WHITE and background color is

14) setcolor

Declaration :- void setcolor(int color);

In Turbo Graphics each color is assigned a number. Total 16 colors are available. Strictly speaking number of available colors depends on current graphics mode and driver. For Example :- BLACK is assigned 0, RED is assigned 4 etc. setcolor function is used to change the current drawing color.e.g. setcolor(RED) or setcolor(4) changes the current drawing color to RED. Remember that default drawing color is WHITE.

15) setfillstyle

setfillstyle function sets the current fill pattern and fill color.

Declaration :- void setfillstyle(int pattern, int color);

Different fill styles:

enum fill_styles

{

EMPTY_FILL,

SOLID_FILL,

LINE_FILL,

LTSLASH_FILL,

SLASH_FILL,

BKSLASH_FILL,

LTBKSLASH_FILL,

HATCH_FILL,

XHATCH_FILL,

INTERLEAVE_FILL,

WIDE_DOT_FILL,

CLOSE_DOT_FILL,

USER_FILL

};

EXPERIMENT NO.2

AIM: To implement basic Graphics Function

Code:

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
void main()
{
int gd=DETECT,gm;
initgraph(&gd,&gm,"C:/TC/BGI");
setcolor(RED);
rectangle(100,20,200,10);
setcolor(BLUE);
setfillstyle(SOLID_FILL,BLUE);
circle(40,40,20);
floodfill(40,40,BLUE);
setcolor(YELLOW);
line(250,260,380,390);
setcolor(CYAN);
setfillstyle(2,CYAN);
ellipse(120,150,0,360,50,70);
floodfill(120,150,CYAN);
getch();
closegraph();
}
```

2.b) Aim: WAP to draw a car using basic graphics function

```
 #include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void main()
{
int gd = DETECT, gm;
initgraph(&gd, &gm, "C://TC//BGI");
//cleardevice();
line( 150, 100, 242, 100);
ellipse(242, 105, 0, 90, 10, 5);
line(150, 100, 120, 150);
line(252, 105, 280, 150);
```

```
line(100, 150, 320, 150);
line(100, 150, 100, 200);
line(320, 150, 320, 200);
line(100, 200, 110, 200);
line( 320, 200, 310, 200);
arc(130, 200, 0, 180, 20);
arc( 290, 200, 0, 180, 20);
line( 270, 200, 150, 200);
circle(130, 200, 17);
circle(290, 200, 17);
getch();
}
```

EXPERIMENT NO.3

AIM: WAP to implement line drawing DDA Algorithm

Code:

```
#include <graphics.h>
#include <stdio.h>
#include <math.h>
#include<conio.h>
void main( )
{
   float x,y,x1,y1,x2,y2,dx,dy,pixel;
   int i,gd=DETECT,gm;
   printf("Enter the value of x1 : ");
   scanf("%f",&x1);
   printf("Enter the value of y1 : ");
   scanf("%f",&y1);
   printf("Enter the value of x2 : ");
   scanf("%f",&x2);
   printf("Enter the value of y1 : ");
   scanf("%f",&y2);
   initgraph(&gd,&gm,"C:/TC/BGI");
   dx=abs(x2-x1);
   dy=abs(y2-y1);

   if(dx>=dy)
   pixel=dx;
   else
   pixel=dy;
   dx=dx/pixel;
   dy=dy/pixel;

   x=x1;
   y=y1;

   i=1;
   while(i<=pixel)
   {
        putpixel(x,y,1);
        x=x+dx;
        y=y+dy;
        i=i+1;
  delay(10);
```

```
    }
    getch();
    closegraph();
}
```

EXPERIMENT NO.4

AIM : WAP to implement line drawing Bresenham's Algorithm.

Code:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
    int gd = DETECT, gm;
    int dx, dy, p, end;
    float x1, x2, y1, y2, x, y;
    initgraph(&gd, &gm, "c:/tc/bgi");
    printf("Enter Value of X1: ");
    scanf("%f", &x1);
    printf("Enter Value of Y1: ");
    scanf("%f", &y1);
    printf("Enter Value of X2: ");
    scanf("%f", &x2);
    printf("Enter Value of Y2: ");
    scanf("%f", &y2);
    dx = abs(x1 - x2);
    dy = abs(y1 - y2);
    p = 2 * dy - dx;
    if(x1 > x2)
    {
        x = x2;
```

```
        y = y2;
        end = x1;
    }
    else
    {
        x = x1;
        y = y1;
        end = x2;
    }
    putpixel(x, y, 10);
    while(x < end)
    {
        x = x + 1;
        if(p < 0)
        {
            p = p + 2 * dy;
        }
        else
        {
            y = y + 1;
            p = p + 2 * (dy - dx);
        }
        putpixel(x, y, 10);
    }
    getch();
    closegraph(); }
```

EXPERIMENT NO.5

AIM: WAP to implement midpoint circle algorithm

Code:

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
void main()
{
int gd=DETECT,gm;
int i,r,x=0,y,xc,yc;
float d;
clrscr();
initgraph(&gd,&gm,"c://tc//bgi");
printf("Enter Radius\n");
scanf("%d",&r);
printf("Enter Center of circle\n");

scanf("%d",&xc);
scanf("%d",&yc);
d=1.25-r;
y=r;
do
{
if(d<0.0)
{
```

```
x=x+1;
d=d+2*x+1;
}
else
{

x=x+1;
y=y-1;
d=d+2*x-2*y+10;
}
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);
}
while(x<y);
getch();
}
```

EXPERIMENT NO.6

AIM : WAp to implement midpoint Ellipse algorithm

```
#include <stdio.h>
 #include <conio.h>
 #include <graphics.h>
 #include <dos.h>

 int main() {
      /* request auto detection */
      int gdriver = DETECT, gmode, err;
      long midx, midy, xradius, yradius;
      long xrad2, yrad2, twoxrad2, twoyrad2;
      long x, y, dp, dpx, dpy;

      /* initialize graphic mode */
      initgraph(&gdriver, &gmode, "C:/TC/BGI");
      err = graphresult();

      if (err != grOk) {
            /* error occurred */
            printf("Graphics Error: %s\n",
                        grapherrormsg(err));
            return 0;
      }

      /* x axis radius and y axis radius of ellipse */
      xradius = 100, yradius = 50;

      /* finding the center postion to draw ellipse */
      midx = getmaxx() / 2;
      midy = getmaxy() / 2;

      xrad2 = xradius * xradius;
      yrad2 = yradius * yradius;

      twoxrad2 = 2 * xrad2;
      twoyrad2 = 2 * yrad2;
      x = dpx = 0;
      y = yradius;
      dpy = twoxrad2 * y;

      putpixel(midx + x, midy + y, WHITE);
      putpixel(midx - x, midy + y, WHITE);
      putpixel(midx + x, midy - y, WHITE);
      putpixel(midx - x, midy - y, WHITE);

      dp = (long) (0.5 + yrad2 - (xrad2 * yradius) + (0.25 * xrad2));
```

```
while (dpx < dpy) {
	x = x + 1;
	dpx = dpx + twoyrad2;
	if (dp < 0) {
		dp = dp + yrad2 + dpx;
	} else {
		y = y - 1;
		dpy = dpy - twoxrad2;
		dp = dp + yrad2 + dpx - dpy;
	}

	/* plotting points in y-axis(top/bottom) */
	putpixel(midx + x, midy + y, WHITE);
	putpixel(midx - x, midy + y, WHITE);
	putpixel(midx + x, midy - y, WHITE);
	putpixel(midx - x, midy - y, WHITE);
	delay(100);
}

delay(500);

dp = (long)(0.5 + yrad2 * (x + 0.5) * (x + 0.5) +
		xrad2 * (y - 1) * (y - 1) - xrad2 * yrad2);

while (y > 0) {
	y = y - 1;
	dpy = dpy - twoxrad2;

	if (dp > 0) {
		dp = dp + xrad2 - dpy;
	} else {
		x = x + 1;
		dpx = dpx + twoyrad2;
		dp = dp + xrad2 - dpy + dpx;
	}

	/* plotting points at x-axis(left/right) */
	putpixel(midx + x, midy + y, WHITE);
	putpixel(midx - x, midy + y, WHITE);
	putpixel(midx + x, midy - y, WHITE);
	putpixel(midx - x, midy - y, WHITE);
	delay(100);
}

getch();

/* deallocate memory allocated for graphic screen */
closegraph();
```

```
    return 0;
}
```

EXPERIMENT NO.7

AIM: WAP to implement 2D transformation

a) Translation b)scaling c)rotation

code:(translation)

```
#include<graphics.h>
#include<stdlib.h>
#include<stdio.h>
#include<math.h>
void main()
{
int graphdriver=DETECT,graphmode,errorcode;
int i;
int x2,y2,x1,y1,x,y;
printf("Enter the 2 line end points:");
printf("x1,y1,x2,y2");
scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
initgraph(&graphdriver,&graphmode,"c://tc//bgi");
line(x1,y1,x2,y2);
printf("Enter translation co-ordinates ");
printf("x,y");
scanf("%d%d",&x,&y);
x1=x1+x;
y1=y1+y;
x2=x2+x;
y2=y2+y;
printf("Line after translation");
line(x1,y1,x2,y2);
getch();
closegraph();
}
```

b) Scaling

```
#include<graphics.h>
#include<stdlib.h>
#include<stdio.h>
#include<math.h>
void main()
{
```

```
int graphdriver=DETECT,graphmode,errorcode;
int i;
int x2,y2,x1,y1,x,y;
printf("Enter the 2 line end points:");
printf("x1,y1,x2,y2");
scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
initgraph(&graphdriver,&graphmode,"c://tc//bgi");
line(x1,y1,x2,y2);
printf("Enter scaling co-ordinates ");
printf("x,y");
scanf("%d%d",&x,&y);
x1=(x1*x);
y1=(y1*y);
x2=(x2*x);
y2=(y2*y);
printf("Line after scaling");
line(x1,y1,x2,y2);
getch();
closegraph();
}
```

c) rotation

```
#include<graphics.h>
#include<stdlib.h>
#include<stdio.h>
#include<math.h>
void main()
{
int graphdriver=DETECT,graphmode,errorcode;
int i;
int x2,y2,x1,y1,x,y,xn,yn;
double r11,r12,r21,r22,th;
clrscr();
printf("Enter the 2 line end points:");
printf("x1,y1,x2,y2");
scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
initgraph(&graphdriver,&graphmode,"c://tc//bgi");
line(x1,y1,x2,y2);
printf("\n\n\n[ Enter the angle");
scanf("%lf",&th);
r11=cos((th*3.1428)/180);
r12=sin((th*3.1428)/180);
r21=(-sin((th*3.1428)/180));
```

```
r22=cos((th*3.1428)/180);
//printf("%lf  %lf  %lf  %lf",r11,r12,r21,r22);
xn=((x2*r11)-(y2*r12));
yn=((x2*r12)+(y2*r11));
line(x1,y1,xn,yn);
getch();
closegraph();
}
```

EXPERIMENT NO.8

AIM : WAP to implement 2D transformation reflection and shearing

```
#include<stdio.h>
#include<process.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void disp(int n,float c[][3])
{
float maxx,maxy;
int i;
maxx=getmaxx();
maxy=getmaxy();
maxx=maxx/2;
maxy=maxy/2;
i=0;
while(i<n-1)
{

line(maxx+c[i][0],maxy-c[i][1],maxx+c[i+1][0],maxy-c[i+1][1]);
i++;
}
i=n-1;
line(maxx+c[i][0],maxy-c[i][1],maxx+c[0][0],maxy-c[0][1]);
setcolor(GREEN);
line(0,maxy,maxx*2,maxy);
```

```
line(maxx,0,maxx,maxy*2);
setcolor(WHITE);
}
void mul(int n,float b[][3],float c[][3],float a[][3])
{
int i,j,k;
for(i=0;i<n;i++)
for(j=0;j<3;j++)
a[i][j]=0;
for(i=0;i<n;i++)
for(j=0;j<3;j++)
for(k=0;k<3;k++)
{
a[i][j] = a[i][j] + (c[i][k] * b[k][j]);
}
}
void reflection(int n,float c[][3])
{
float b[10][3],a[10][3];
int i=0,ch,j;
cleardevice();
printf("\n\t* * MENU * *");
printf("\n\t1) ABOUT X-AXIS");
printf("\n\t2) ABOUT Y-AXIS");
printf("\n\t3) ABOUT ORIGIN");
printf("\n\t4) ABOUT X=Y");
```

```
printf("\n\t5) ABOUT -X=Y");
printf("\n\t6) EXIT");
printf("\n\tENTER YOUR CHOICE :   ");
scanf("%d",&ch);
clrscr();
cleardevice();
disp(n,c);
for(i=0;i<3;i++)
for(j=0;j<3;j++)
{
b[i][j]=0;
if(i==j)
b[i][j]=1;
}
switch(ch)
{
case 1:
b[1][1]=-1;
break;
case 2:
b[0][0]=-1;
break;
case 3:
b[0][0]=-1;
b[1][1]=-1;
break;
```

```
case 4:
b[0][0]=0;
b[1][1]=0;
b[0][1]=1;
b[1][0]=1;
break;
case 5:
b[0][0]=0;
b[1][1]=0;
b[0][1]=-1;
b[1][0]=-1;
break;
case 6:
break;
default:
printf("\n\tINVALID CHOICE ! ");
break;
}
mul(n,b,c,a);
setcolor(RED);
disp(n,a);
}
void shearing(int n,float c[][3])
{
float b[10][3],sh,a[10][3];
int i=0,ch,j;
```

```
cleardevice();
printf("\n\t* * * MENU * * *");
printf("\n\t1) X SHEARING");
printf("\n\t2) Y SHEARING");
printf("\n\t3) EXIT ");
printf("\n\tENTER YOUR CHOICE :   ");
scanf("%d",&ch);
if(ch==3)
return;
printf("\n\tENTER THE VALUE for SHEARING:        ");
scanf("%f",&sh);
clrscr();
cleardevice();
for(i=0;i<3;i++)
for(j=0;j<3;j++)
b[i][j]=0;
for(i=0;i<3;i++)
b[i][i]=1;
switch(ch)
{
case 1:
b[1][0]=sh;
break;
case 2:
b[0][1]=sh;
break;
```

```
case 3:
break;
default:
printf("\n\tINVALID CHOICE ! ");
break;
}
mul(n,b,c,a);
setcolor(RED);
disp(n,a);
}
void main()
{
int i,j,k,cho,n,gd=DETECT,gm;
float c[10][3],tx,ty,sx,sy,ra;
initgraph(&gd,&gm,"c://tc//bgi");
printf("\nEnter the number of vertices :   ");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nEnter the co-ordinates of the %d vertex :",i+1);
scanf("%f%f",&c[i][0],&c[i][1]);
c[i][2]=1;
}
do
{
clrscr();
```

```
cleardevice();
printf("\n\t\t\t * * * MENU * * *");
printf("\n\t 1) REFLECTION ");
printf("\n\t 2) SHEARING");
printf("\n\t 3) EXIT");
printf("\n\t ENTER YOUR CHOICE:   ");
scanf("%d",&cho);
switch(cho)
{
case 1:
clrscr();
cleardevice();
setcolor(BLUE);
disp(n,c);
reflection(n,c);
getch();
break;
case 2:
clrscr();
cleardevice();
setcolor(BLUE);
disp(n,c);
shearing(n,c);
getch();
break;
case 3 :
```

```
exit(0);
break;
default:
printf("\n\tInvalid choice !!");
break; }}while(cho!=3);
getch();
closegraph();
}
```

EXPERIMENT NO.9

AIM : Write a C-program for performing the basic transformations such as translation, Scaling, Rotation for a given 3D object?

Program for translation:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<process.h>
#include<graphics.h>
int x1,x2,y1,y2,mx,my,depth;
void draw();
void trans();
void main()
{
int gd=DETECT,gm,c;
initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
printf("\n\t\t3D Translation\n\n");
printf("\nEnter 1st top value(x1,y1):");
scanf("%d%d",&x1,&y1);
printf("Enter right bottom value(x2,y2):");
scanf("%d%d",&x2,&y2);
depth=(x2-x1)/4;
mx=(x1+x2)/2;
my=(y1+y2)/2;
draw();
getch();
cleardevice();
```

```
trans();
getch();
}
void draw()
{
bar3d(x1,y1,x2,y2,depth,1);
}
void trans()
{
int a1,a2,b1,b2,dep,x,y;
printf("\n Enter the Translation Distances:");
scanf("%d%d",&x,&y);
a1=x1+x;
a2=x2+x;
b1=y1+y;
b2=y2+y;
dep=(a2-a1)/4;
bar3d(a1,b1,a2,b2,dep,1);
setcolor(5);
draw();
}
```

Program for scaling:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<process.h>
```

```
#include<graphics.h>
int x1,x2,y1,y2,mx,my,depth;
void draw();
void scale();
void main()
{
int gd=DETECT,gm,c;
initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
printf("\n\t\t3D Scaling\n\n");
printf("\nEnter 1st top value(x1,y1):");
scanf("%d%d",&x1,&y1);
printf("Enter right bottom value(x2,y2):");
scanf("%d%d",&x2,&y2);
depth=(x2-x1)/4;
mx=(x1+x2)/2;
my=(y1+y2)/2;
draw();
getch();
cleardevice();
scale();
getch();
}
void draw()
{
bar3d(x1,y1,x2,y2,depth,1);
}
```

```
void scale()
{
int x,y,a1,a2,b1,b2,dep;
printf("\n\n Enter scaling Factors:");
scanf("%d%d",&x,&y);
a1=mx+(x1-mx)*x;
a2=mx+(x2-mx)*x;
b1=my+(y1-my)*y;
b2=my+(y2-my)*y;
dep=(a2-a1)/4;
bar3d(a1,b1,a2,b2,dep,1);
setcolor(5);
draw();
}
```

Program for Rotation:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<graphics.h>
int x1,x2,y1,y2,mx,my,depth;
void draw();
void rotate();
void main()
{
int gd=DETECT,gm,c;
```

```
initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
printf("\n3D Transformation Rotating\n\n");
printf("\nEnter 1st top value(x1,y1):");
scanf("%d%d",&x1,&y1);
printf("Enter right bottom value(x2,y2):");
scanf("%d%d",&x2,&y2);
depth=(x2-x1)/4;
mx=(x1+x2)/2;
my=(y1+y2)/2;
draw(); getch();
cleardevice();
rotate();
getch();
}
void draw()
{
bar3d(x1,y1,x2,y2,depth,1);
}
void rotate()
{
float t;
int a1,b1,a2,b2,dep;
printf("Enter the angle to rotate=");
scanf("%f",&t);
t=t*(3.14/180);
a1=mx+(x1-mx)*cos(t)-(y1-my)*sin(t);
```

```
a2=mx+(x2-mx)*cos(t)-(y2-my)*sin(t);
b1=my+(x1-mx)*sin(t)-(y1-my)*cos(t);
b2=my+(x2-mx)*sin(t)-(y2-my)*cos(t);
if(a2>a1)
dep=(a2-a1)/4;
else
dep=(a1-a2)/4;
bar3d(a1,b1,a2,b2,dep,1); setcolor(5);
}
```

EXPERIMENT NO.10

Aim: WAP to Generate Fractal Image
Code:

```
#include<stdio.h>
#include<math.h>
#include<graphics.h>
  int a;

void drawfern(int x,int y,int l,int arg,int n)
{
int x1,y1,i;
int l1,xpt,ypt;

if(n>0&&!kbhit())
 {
 x1=(int)(x-l*sin(arg*3.14/180));
 y1=(int)(y-l*cos(arg*3.14/180));
 line(x,y,x1,y1);
 l1=(int)(l/5);
 for(i=1;i<6;i++)
  {
  xpt=(int)(x-i*l1*sin(arg*3.14/180));
  ypt=(int)(y-i*l1*cos(arg*3.14/180));
  drawfern(xpt,ypt,(int)(l/(i+1)),arg+a,n-1);
  drawfern(xpt,ypt,(int)(l/(i+1)),arg-a,n-1);
  }
 }
}



void main()
{
int gd=DETECT,gm,x,y,l;
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI\\");
x=getmaxx()/2;
y=getmaxy()/2;
l=150;
a=45;
setcolor(YELLOW);
drawfern(x,y,l,0,5);
getch();
}
```

EXPERIMENT NO.11

Aim: WAP to implement Cohen Suther land Clipping algorithm

Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<graphics.h>
#include<dos.h>

typedef struct coordinate
{
   int x,y;
   char code[4];
}PT;
 void drawwindow();
void drawline(PT p1,PT p2);
PT setcode(PT p);
int visibility(PT p1,PT p2);
PT resetendpt(PT p1,PT p2);
 void main()
{
   int gd=DETECT,v,gm;
   PT p1,p2,p3,p4,ptemp;
      printf("\nEnter x1 and y1\n");
   scanf("%d %d",&p1.x,&p1.y);
```

```
printf("\nEnter x2 and y2\n");
scanf("%d %d",&p2.x,&p2.y);
   initgraph(&gd,&gm,"c://tc//bgi");
drawwindow();
delay(500);
   drawline(p1,p2);
delay(500);
cleardevice();
   delay(500);
p1=setcode(p1);
p2=setcode(p2);
v=visibility(p1,p2);
delay(500);
   switch(v)
{
case 0: drawwindow();
        delay(500);
        drawline(p1,p2);
        break;
case 1:    drawwindow();
        delay(500);
        break;
case 2:    p3=resetendpt(p1,p2);
        p4=resetendpt(p2,p1);
        drawwindow();
        delay(500);
```

```
            drawline(p3,p4);
            break;
    }
        delay(5000);
    closegraph();
}

void drawwindow()
{
    line(150,100,450,100);
    line(450,100,450,350);
    line(450,350,150,350);
    line(150,350,150,100);
}
 void drawline(PT p1,PT p2)
{
    line(p1.x,p1.y,p2.x,p2.y);
}
PT setcode(PT p)    //for setting the 4 bit code
{
    PT ptemp;
        if(p.y<100)
        ptemp.code[0]='1';    //Top
    else
        ptemp.code[0]='0';
        if(p.y>350)
```

```
        ptemp.code[1]='1';    //Bottom
    else
        ptemp.code[1]='0';
            if(p.x>450)
        ptemp.code[2]='1';    //Right
    else
        ptemp.code[2]='0';
            if(p.x<150)
        ptemp.code[3]='1';    //Left
    else
        ptemp.code[3]='0';
        ptemp.x=p.x;
    ptemp.y=p.y;
        return(ptemp);
}
 int visibility(PT p1,PT p2)
{
    int i,flag=0;
        for(i=0;i<4;i++)
    {
        if((p1.code[i]!='0') || (p2.code[i]!='0'))
            flag=1;          }
    if(flag==0)
        return(0);
        for(i=0;i<4;i++)
    {
```

```
        if((p1.code[i]==p2.code[i]) && (p1.code[i]=='1'))
            flag='0';          }
    if(flag==0)
        return(1);


    return(2);
}
 PT resetendpt(PT p1,PT p2)
{
    PT temp;
    int x,y,i;
    float m,k;
        if(p1.code[3]=='1')
        x=150;
        if(p1.code[2]=='1')
        x=450;
        if((p1.code[3]=='1') || (p1.code[2]=='1'))
    {
        m=(float)(p2.y-p1.y)/(p2.x-p1.x);
        k=(p1.y+(m*(x-p1.x)));
        temp.y=k;
        temp.x=x;
              for(i=0;i<4;i++)
            temp.code[i]=p1.code[i];
              if(temp.y<=350 && temp.y>=100)
            return (temp);
```

```
}
   if(p1.code[0]=='1')
   y=100;
   if(p1.code[1]=='1')
   y=350;
if((p1.code[0]=='1') || (p1.code[1]=='1'))
{
   m=(float)(p2.y-p1.y)/(p2.x-p1.x);
   k=(float)p1.x+(float)(y-p1.y)/m;
   temp.x=k;
   temp.y=y;
         for(i=0;i<4;i++)
      temp.code[i]=p1.code[i];
   return(temp);
}       else
   return(p1);      }
```

EXPERIMENT NO.12

Aim: WAP to implement Sutherland-Hodgeman Polygon Clipping Algorithm

Code:

```
#include<stdio.h>
#include<conio.h>
struct poly
{
int coeff,expo;
};
struct poly p1[10],p2[10],p3[20];
int readpoly(struct poly[]);
int addpoly(struct poly[],struct poly[],int,int,struct poly[]);
void display(struct poly[],int terms);
void main()
{
clrscr();
int t1,t2,t3;
t1=readpoly(p1);
printf("\nfirst plynomial is:");
display(p1,t1);
t2=readpoly(p2);
printf("\nfsecond plynomial is:");
display(p2,t2);
t3=addpoly(p1,p2,t1,t2,p3);
printf("\nresultant polynomial:");
```

```
display(p3,t3);
printf("\n");
getch();
}
int readpoly(struct poly p[10])
{
int t1,i;
printf("\nenter no of terms in polynomial");
scanf("%d",&t1);
printf("enter coeffiecient in desecnding order");
for(i=0;i<t1;i++)
{

printf("enter coefiecient %d",i);
scanf("%d",&p[i].coeff);
printf("enter exponent %d",i);
scanf("%d",&p[i].expo);
}
return(i);
}
int addpoly(struct poly p1[10],struct poly p2[10],int t1,int t2,struct poly p3[20])
{
int i,j,k;
i=j=k=0;
while(i<t1 && j<t2)
{
```

```
if(p1[i].expo<p2[j].expo)
{
 p3[k].coeff=p2[j].coeff;
p3[k].expo=p2[j].expo;
//p3[k]=p2[j];
k++;j++;
}
else if(p1[i].expo>p2[j].expo)
{
p3[k].coeff=p1[i].coeff;
p3[k].expo=p1[i].expo;
//p3[k]=p1[i];
i++;k++;
}
else
{ p3[k].coeff=p1[i].coeff+p2[i].coeff;
p3[k].expo=p1[i].expo;
i++;j++;k++;


}
}
while(i<t1)
{
p3[k].coeff=p1[i].coeff;
p3[k].expo=p1[i].expo;
i++;k++;
```

```
}
while(j<t2)
{
p3[k].coeff=p2[j].coeff;
p3[k].expo=p2[j].expo;
j++;k++;
}
return k;
}
void display(struct poly p[10],int term)
{
int k;
for (k=0;k<term-1;k++)
printf("%d(x^%d)+",p[k].coeff,p[k].expo);
printf("%d(x^%d)",p[term-1].coeff,p[term-1].expo);
}
```

Beyond The Syllabus

EXPERIMENT NO.13

Aim: Circle moving from left to right and vice versa

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
#include<dos.h>
void main()
{
int gd=DETECT,gm,i;
initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
//for moving circle from left to right,the following loop works
for(i=50;i<=getmaxx();i++)
{
setcolor(3);
setfillstyle(SOLID_FILL,9);
circle(50+i,50,50);
floodfill(52+i,52,3);
delay(20);
cleardevice();
}
//for moving circle from right to left, the following loop works
for(i=getmaxy();i>=0;i--)
{
setcolor(3);
```

```
setfillstyle(SOLID_FILL,9);
circle(i,50,50);
floodfill(i+2,52,3);
delay(20);
cleardevice();
}
//for moving circle from top to bottom,the following loop works
for(i=50;i<=getmaxy();i++)
{
setcolor(3);
setfillstyle(SOLID_FILL,9);
circle(50,i,50);
floodfill(52,i+2,3);
delay(20);
cleardevice();
}
//for moving circle from bottom to top,the following loop works
for(i=getmaxy();i>=0;i--)
{
setcolor(3);
setfillstyle(SOLID_FILL,9);
circle(50,i,50);
floodfill(52,i+2,3);
delay(20);
cleardevice();
}
```