



***COMPUTER GRAPHICS AND MULTIMEDIA EXPERIMENTS***

**EN18CS301012**

**Aditya Manthankar**

**CS-A**

**EXPERIMENT 1**

**Aim: To study introduction of computer graphics.**

## Definition of Computer Graphics:

It is the use of computers to create and manipulate pictures on a display device. It comprises of software techniques to create, store, modify, represents pictures.

## Why computer graphics used?

A picture can be understood easily just with a single look. Interactive computer graphics work using the concept of two-way communication between computer users. The computer will receive signals from the input device, and the picture is modified accordingly. Picture will be changed quickly when we apply command.

# Application of Computer Graphics:

**1. Education and Training:** Computer-generated model of the physical, financial and economic system is often used as educational aids. Model of physical systems, physiological system, population trends or equipment can help trainees to understand the operation of the system.

**2. Use in Biology:** Molecular biologist can display a picture of molecules and gain insight into their structure with the help of computer graphics.

**3. Computer-Generated Maps:** Town planners and transportation engineers can use computer-generated maps which display data useful to them in their planning work.

**4. Architect:** Architect can explore an alternative solution to design problems at an interactive graphics terminal. In this way, they can test many more solutions that would not be possible without the computer.

**5. Presentation Graphics:** Example of presentation Graphics are bar charts, line graphs, pie charts and other displays showing relationships between multiple parameters. Presentation Graphics is commonly used to summarize

* Financial Reports
* Statistical Reports
* Mathematical Reports
* Scientific Reports
* Economic Data for research reports
* Managerial Reports
* Consumer Information Bulletins
* And other types of reports

**6. Computer Art:** Computer Graphics are also used in the field of commercial arts. It is used to generate television and advertising commercial.

**7. Entertainment:** Computer Graphics are now commonly used in making motion pictures, music videos and television shows.

**8. Visualization:** It is used for visualization of scientists, engineers, medical personnel, business analysts for the study of a large amount of information.

**9. Educational Software:** Computer Graphics is used in the development of educational software for making computer-aided instruction.

**10. Printing Technology:** Computer Graphics is used for printing technology and textile design.

### Example of Computer Graphics Packages:

1. LOGO
2. COREL DRAW
3. AUTO CAD
4. 3D STUDIO
5. CORE
6. GKS (Graphics Kernel System)
7. PHIGS
8. CAM (Computer Graphics Metafile)
9. CGI (Computer Graphics Interface)

**EXPERIMENT 2**

**Aim: Write a program to draw various shapes using graphics.h header file.**

#include<graphics.h>

#include<conio.h>

int main(){

    int gd=DETECT,gm;

    initgraph(&gd,&gm,"c:\\tc\\bgi");

    printf("\t\t\t\n\nLINE");

    line(50,40,190,40);

    printf("\t\t\n\n\n\nRECTANGLE");

    rectangle(125,115,215,165);

    printf("\t\t\t\n\n\n\n\n\n\nARC");

    arc(120,200,180,0,30);

    printf("\t\n\n\n\nCIRCLE");

    circle(120,270,30);

    printf("\t\n\n\n\nECLIPSE");

    ellipse(120,350,0,360,30,20);

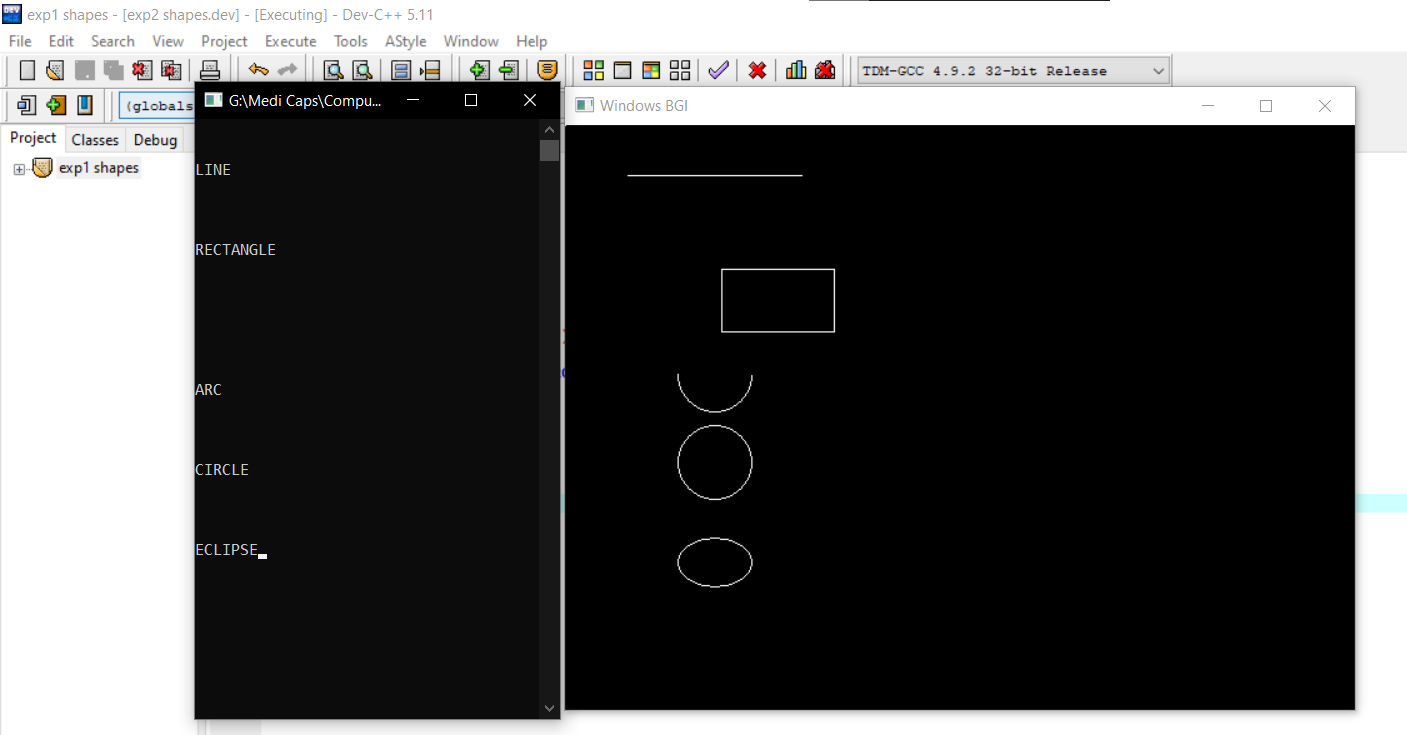
    getch();

closegraph();

return 0;

}

* **Output:**

****

**EXPERIMENT 3**

**Aim: Write a program to draw a moving car.**

#include<graphics.h>

#include<conio.h>

**int** main()

{

    int gd=DETECT,gm, i, maxx, cy;

    initgraph(&gd, &gm, "");

    setbkcolor(WHITE);

    setcolor(RED);

    maxx = getmaxx();

    cy = getmaxy()/2;

**for**(i=0;i<maxx-140;i++)

        {

        cleardevice();

        line(0+i,cy-20, 0+i, cy+15);

        line(0+i, cy-20, 25+i, cy-20);

        line(25+i, cy-20, 40+i, cy-70);

        line(40+i, cy-70, 100+i, cy-70);

        line(100+i, cy-70, 115+i, cy-20);

        line(115+i, cy-20, 140+i, cy-20);

        line(0+i, cy+15, 18+i, cy+15);

        circle(28+i, cy+15, 10);

        line(38+i, cy+15, 102+i, cy+15);

        circle(112+i, cy+15,10);

        line(122+i, cy+15 ,140+i,cy+15);

        line(140+i, cy+15, 140+i, cy-20);

        rectangle(50+i, cy-62, 90+i, cy-30);

        setfillstyle(1,BLUE);

        floodfill(5+i, cy-15, RED);

        setfillstyle(1, LIGHTBLUE);

        floodfill(52+i, cy-60, RED);

        delay(10);

         }

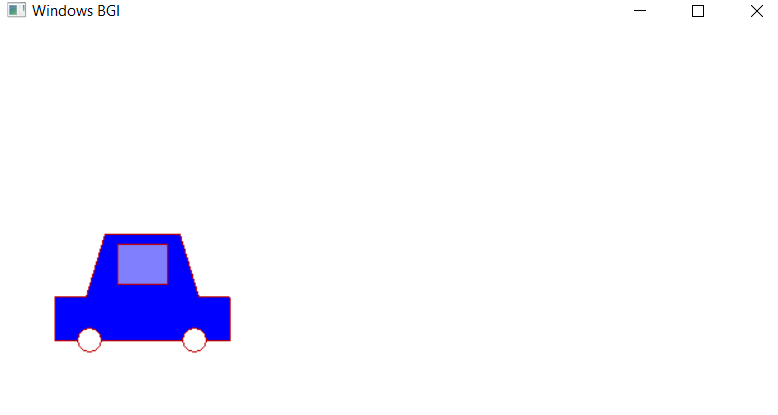
    getch();

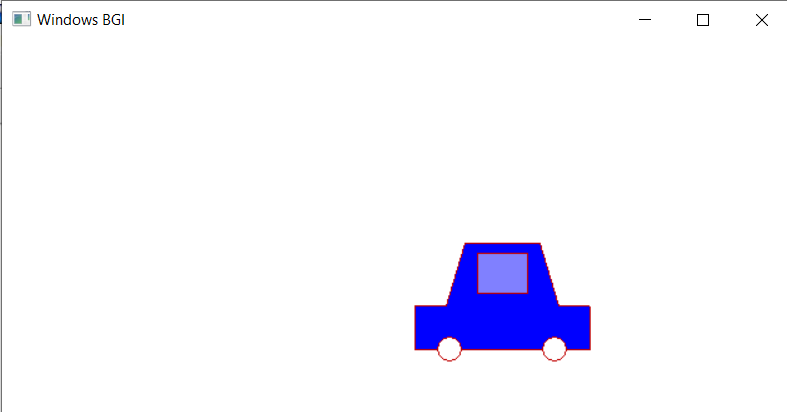
    closegraph();

**return** 0;

}

* **Output:**

****

****

**EXPERIMENT 4**

**Aim: Write a program to draw bouncing balls.**

#include <stdio.h>

#include <conio.h>

#include <graphics.h>

#include <dos.h>

/\* color and movement of ball \*/

void ball\_motion(int \*x, int \*y, int \*flag, int \*i, int val, int tmp) {

int maxy = getmaxy();

/\* set the drawing color \*/

setcolor(YELLOW);

/\* color of the ball \*/

setfillstyle(SOLID\_FILL, BLUE);

/\* draw ball at the given position \*/

pieslice(\*x, \*y, 0, 360, 10);

/\* subsequent position of ball in x - axis \*/

if (\*i % 5 == 0) {

if (tmp) {

\*x = \*x + 3;

} else {

\*x = \*x - 3;

}

\*i = 0;

}

/\* subsequent position of ball in y-axis \*/

if (\*flag) {

\*y = \*y - 10;

} else {

\*y = \*y + 10;

}

/\* reached y axis maximum \*/

if (\*y >= maxy) {

\*flag = 1;

} else if (\*y <= val) {

/\* reached y axix minimum \*/

\*flag = 0;

}

\*i = \*i + 1;

return;

}

int main() {

/\* request auto detection \*/

int gdriver = DETECT, gmode, err;

int x[4], y[4], stage[4], i, k = 0;

int flag[] = {0, 1, 0, 1};

int maxy, midy;

/\* initialize graphic mode \*/

initgraph(&gdriver, &gmode, "C:/TURBOC3/BGI");

err = graphresult();

if (err != grOk) {

/\* error occurred \*/

printf("Graphics Error: %s\n",

grapherrormsg(err));

return 0;

}

/\* used for x-axis movement (i - in ball\_motion() API) \*/

for (i = 0; i < 4; i++) {

stage[i] = 0;

}

/\* midy - mid of y axis \*/

midy = getmaxy() / 2;

/\* max of y axis \*/

maxy = getmaxy();

/\* initial position of 4 balls \*/

x[0] = 0, y[0] = 0;

x[1] = 150, y[1] = maxy;

x[2] = getmaxx() - 100, y[2] = midy;

x[3] = getmaxx(), y[3] = maxy;

/\* movement of all 4 balls \*/

while (!kbhit()) {

/\* clears graphic screeen \*/

cleardevice();

ball\_motion(&x[0], &y[0], &flag[0], &stage[0], 0, 1);

ball\_motion(&x[1], &y[1], &flag[1], &stage[1], midy, 1);

ball\_motion(&x[2], &y[2], &flag[2], &stage[2], midy, 0);

ball\_motion(&x[3], &y[3], &flag[3], &stage[3], 0, 0);

/\* sleep for 25 milliseconds \*/

delay(25);

}

getch();

/\* deallocate memory allocated for graphic screen \*/

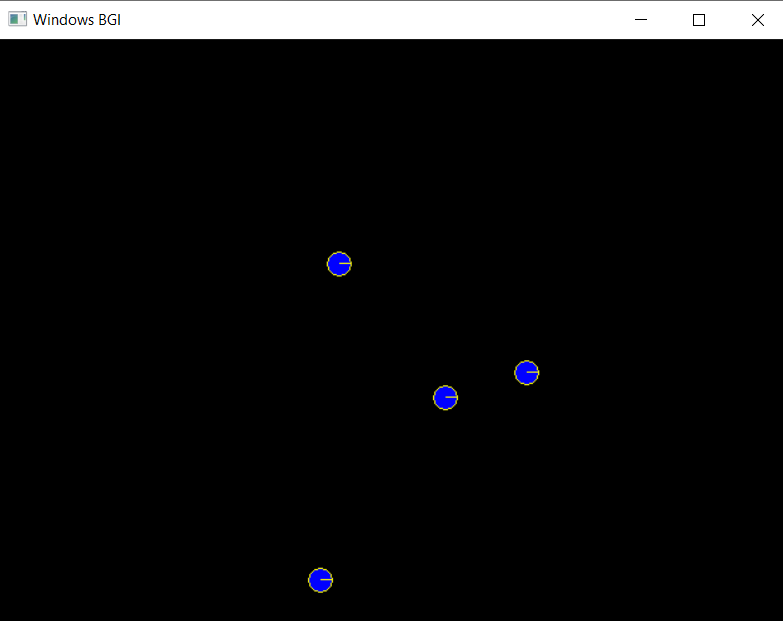
closegraph();

return 0;

}

* **Output:**

****

****

**EXPERIMENT 5**

**Aim: Write a program to draw human body moving in rain with umbrella.**

#include<graphics.h>

#include<stdlib.h>

#include<dos.h>

int main()

{

int gd=DETECT,gm;

int a,b ,rhx,rhy,j,i;

initgraph(&gd,&gm,"");

for(i=0;i<500;i+=5)

{

line(20,380,580,380); //platform

if(i%2==0)

{

line(25+i,380,35+i,340); //leftleg

line(45+i,380,35+i,340);//right leg

line(35+i,310,25+i,330);//left hand

delay(20);

}

else

{

line(35+i,380,35+i,340);

line(35+i,310,40+i,330);

delay(20);

}

line(35+i,340,35+i,310); //body

circle(35+i,300,10); //head

line(35+i,310,50+i,330); // hand

line(50+i,330,50+i,280); //umbrella stick

line(15+i,280,85+i,280); //umbrella right

arc(50+i,280,0,180,35); //umbrella body

arc(55+i,330,180,360,5);//umbrella handle

setfillstyle(INTERLEAVE\_FILL,BLUE);

floodfill(52+i,282,WHITE);

rhx=getmaxx();

rhy=getmaxy();

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

{

outtextxy(rand() % rhx + 1,rand() % rhy-50,"|");

setcolor(WHITE);

}

delay(150);

cleardevice();

}

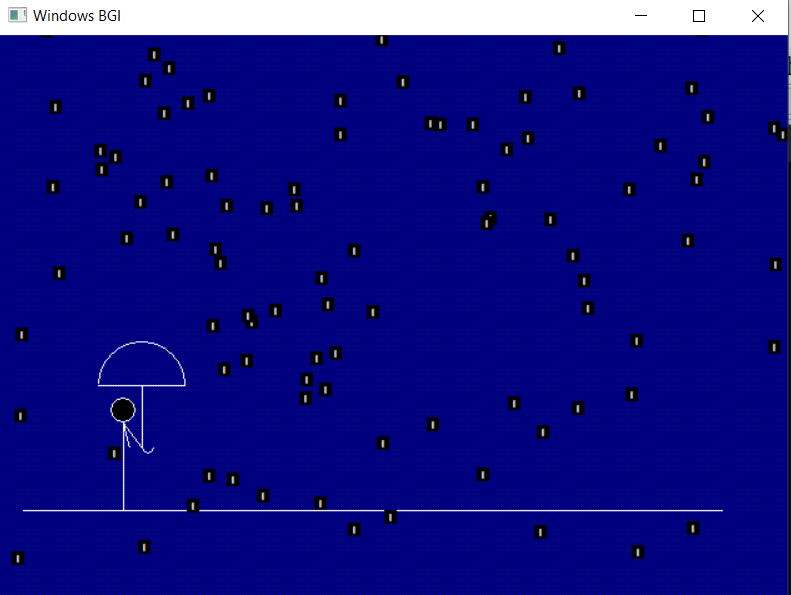
getch();

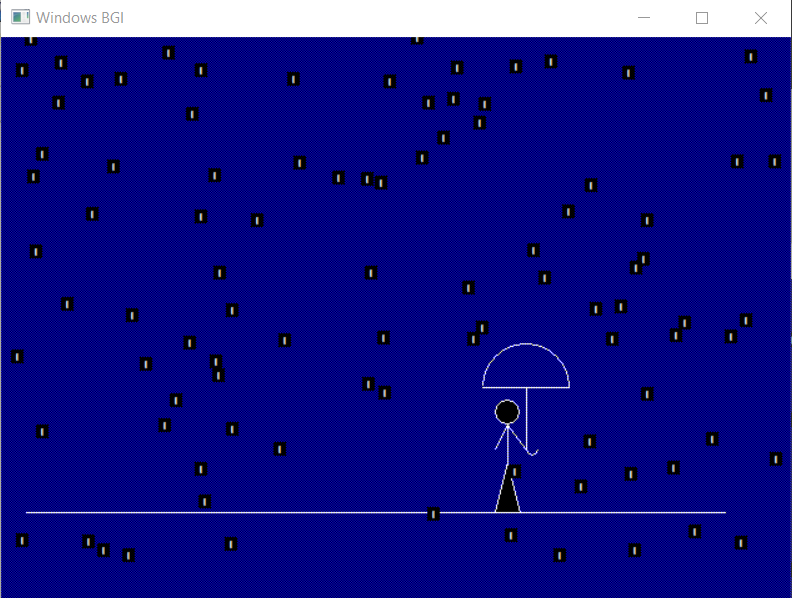
closegraph();

return 0;

}

* **Output:**

****

****

**EXPERIMENT 6**

**Aim: Write a program to draw a house.**

#include <graphics.h>

int main(){

int gdriver = DETECT, gmode;

initgraph(&gdriver, &gmode, "");

line(100, 100, 150, 50);

line(150, 50, 200, 100);

line(150, 50, 350, 50);

line(350, 50, 400, 100);

rectangle(100, 100, 200, 200);

rectangle(200, 100, 400, 200);

rectangle(130, 130, 170, 200);

rectangle(250, 120, 350, 180);

setfillstyle(5, 9);

floodfill(131, 131, WHITE);

floodfill(201, 101, WHITE);

setfillstyle(7, 10);

floodfill(101, 101, WHITE);

floodfill(150, 52, WHITE);

floodfill(163, 55, WHITE);

floodfill(251, 121, WHITE);

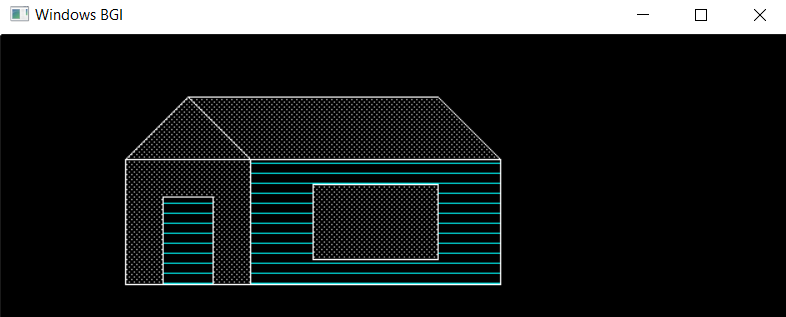
getch();

closegraph();

return 0;

}

* **Output:**

****

**EXPERIMENT 7**

## Aim: Write a program to draw a line using Bradenham’s algorithm.

#include<graphics.h>

#include<stdio.h>

#include<math.h>

int main(){

int gd,gm,x,y,end,p,x1,x2,y1,y2,dx,dy;

char ar1[100], ar2[100];

detectgraph(&gd,&gm);

initgraph(&gd,&gm,"C://TurboC3//BGI");

printf("Enter the value of x1 : ");

scanf("%d",&x1);

printf("Enter the value of y1 : ");

scanf("%d",&y1);

printf("Enter the value of x2 : ");

scanf("%d",&x2);

printf("Enter the value of y2 : ");

scanf("%d",&y2);

sprintf(ar1,"(%d,%d)",x1,y1);

sprintf(ar2,"(%d,%d)",x2,y2);

outtextxy(x1,y1,ar1);

outtextxy(x2,y2,ar2);

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,WHITE);

while(x<=end)

{

if(p<0)

{

x++;

p=p+2\*dy;

}

else{

x++;

y++;

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

}

putpixel(x,y,RED );

delay(20);

}

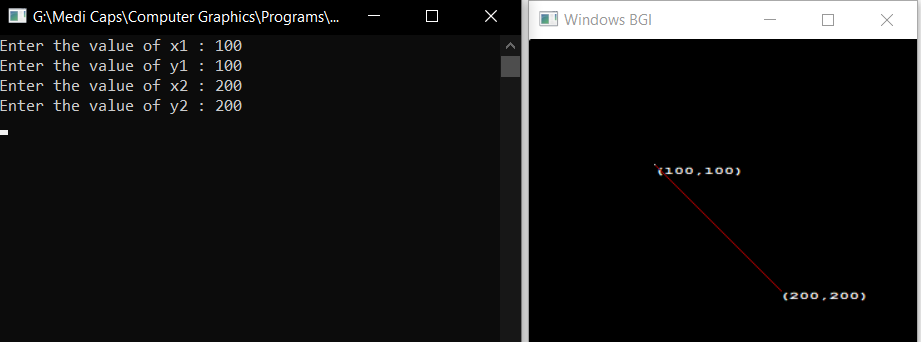
getch();

closegraph();

return 0;

}

* **Output:**

****

**EXPERIMENT 8**

**Aim: Write a program to draw a circle using midpoint algorithm.**

#include<stdio.h>

#include<graphics.h>

#include<conio.h>

void drawcircle(int x0 , int y0, int radius){

int x=radius, y=0, err=0;

while(x>=y){

putpixel(x0+x,y0+y,7);

putpixel(x0+y,y0+x,7);

putpixel(x0-y,y0+x,7);

putpixel(x0-x,y0+y,7);

putpixel(x0-x,y0-y,7);

putpixel(x0-y,y0-x,7);

putpixel(x0+y,y0-x,7);

putpixel(x0+x,y0-y,7);

if(err<=0){

y=y+1;

err=err+2\*y+1;}

if(err>0){

x=x-1;

err=err-2\*x+1;}}}

int main(){

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

char c1[10];

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

printf("enter the radius");

scanf("%d",&r);

sprintf(c1,"%dr",r);

printf("enter the coordinates");

scanf("%d %d",&x,&y);

circle(x,y,1);

line(x,y,x+r,y);

outtextxy(x+3,y+10,c1);

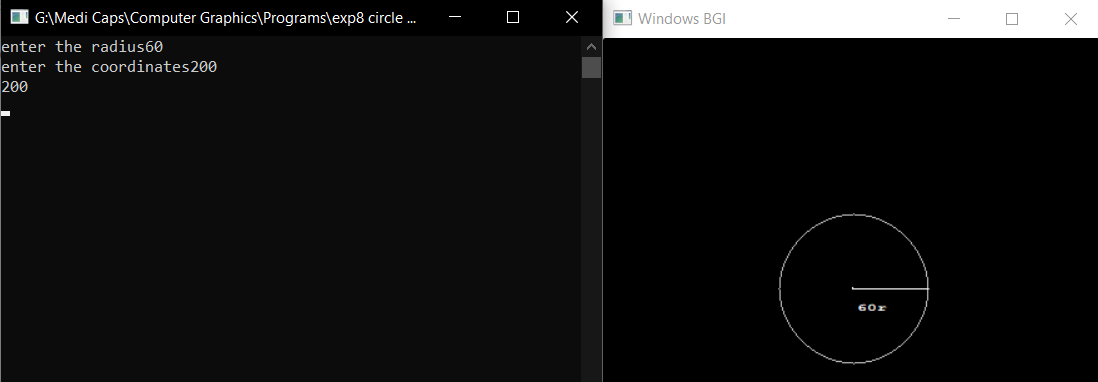
drawcircle(x,y,r);

getch();

closegraph();

return 0;}

* **Output:**



**EXPERIMENT 9**

**Aim: Write a program to draw a circle using Bresenham’s algorithm.**

#include<graphics.h>

#include<stdio.h>

#include<conio.h>

#include<math.h>

int main()

{

int gd=DETECT,gm;

int r,x,y,p,xc,yc;

initgraph(&gd,&gm,"");

cleardevice();

printf("--Bresenham's Circle Drawing---\n");

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

scanf("%d %d",&xc,&yc);

printf("Enter r value :\n");

scanf("%d",&r);

char c0[50];

sprintf(c0,"( %d , %d )",xc,yc);

outtextxy(xc,yc,c0);

x= 0;

y= r;

putpixel(xc+x,yc-y,8);

p=3-(2\*r);

for(x=0;x<=y;x++){

if(p<0){

y=y;

p=(p+(4\*x)+6);

}

else{

y=y-1;

p=p+(4\*(x-y)+10);

}

putpixel(xc+x,yc-y,8);

putpixel(xc-x,yc-y,8);

putpixel(xc+x,yc+y,8);

putpixel(xc-x,yc+y,8);

putpixel(xc+y,yc-x,8);

putpixel(xc-y,yc-x,8);

putpixel(xc+y,yc+x,8);

putpixel(xc-y,yc+x,8);

}

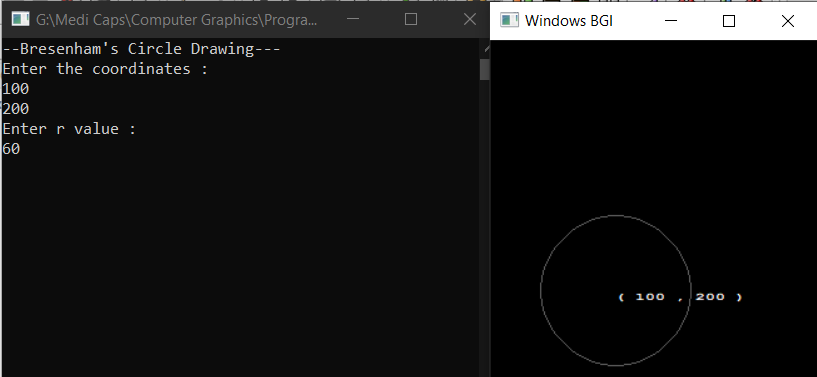
getch();

closegraph();

return 0;

}

* **Output:**



**EXPERIMENT 10**

**Aim: Write a program to implement flood fill algorithm.**

#include<graphics.h>

#include<dos.h>

void flodfill(int x,int y,int f,int o){

int c;

c=getpixel(x,y);

if(c==o){

setcolor(f);

putpixel (x,y,f);

flodfill(x+1,y,f,o);

flodfill(x,y+1,f,o);

flodfill(x+1,y+1,f,o);

flodfill(x-1,y-1,f,o);

flodfill(x-1,y,f,o);

flodfill(x,y-1,f,o);

flodfill(x-1,y+1,f,o);

flodfill(x+1,y-1,f,o);

delay(20);

}

}

int main(){

int gd=DETECT,gm;

initgraph(&gd,&gm,"");

rectangle(50,50,250,250);

flodfill(51,51,10,0);

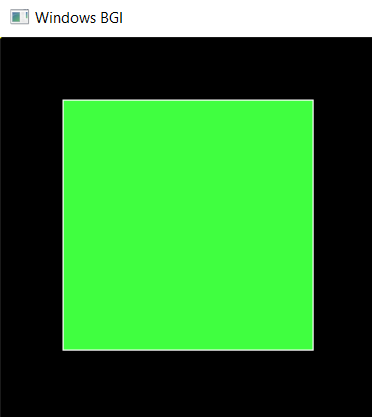
getch();

closegraph();

return 0;

}

* **Output:**

****

**EXPERIMENT 11**

**Aim: Write a program to implement boundary fill algorithm.**

#include<graphics.h>

#include<stdlib.h>

#include<conio.h>

#include<dos.h>

int main(){

void boundary\_fill(int x,int y,int f,int b);

int gdriver = DETECT, gmode;

initgraph(&gdriver, &gmode, "");

setcolor(getmaxcolor());

circle(100, 100, 25);

boundary\_fill(100,100,10,15);

getch();

closegraph();

return 0;

}

void boundary\_fill(int x,int y,int f,int b){

if(getpixel(x,y)!=b && getpixel(x,y)!=f){

putpixel(x,y,f);

delay(0);

boundary\_fill(x+1,y,f,b);

boundary\_fill(x-1,y,f,b);

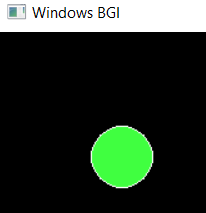
boundary\_fill(x,y+1,f,b);

boundary\_fill(x,y-1,f,b);

}

}

* **Output:**

****

**EXPERIMENT 12**

**Aim: Write a program to implement 3 basic transformation.**

1. **Rotation:**

#include<stdio.h>

#include<graphics.h>

#include<math.h>

int main(){

int gd=0,gm,x1,y1,x2,y2,x3,y3;

double s,c, angle;

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

setcolor(RED);

printf("Enter coordinates of triangle: ");

scanf("%d%d%d%d%d%d",&x1,&y1,&x2,&y2, &x3, &y3);

setbkcolor(WHITE);

cleardevice();

line(x1,y1,x2,y2);

line(x2,y2, x3,y3);

line(x3, y3, x1, y1);

getch();

setbkcolor(BLACK);

printf("Enter rotation angle: ");

scanf("%lf", &angle);

setbkcolor(WHITE);

c = cos(angle \*M\_PI/180);

s = sin(angle \*M\_PI/180);

x1 = floor(x1 \* c + y1 \* s);

y1 = floor(-x1 \* s + y1 \* c);

x2 = floor(x2 \* c + y2 \* s);

y2 = floor(-x2 \* s + y2 \* c);

x3 = floor(x3 \* c + y3 \* s);

y3 = floor(-x3 \* s + y3 \* c);

cleardevice();

line(x1, y1 ,x2, y2);

line(x2,y2, x3,y3);

line(x3, y3, x1, y1);

getch();

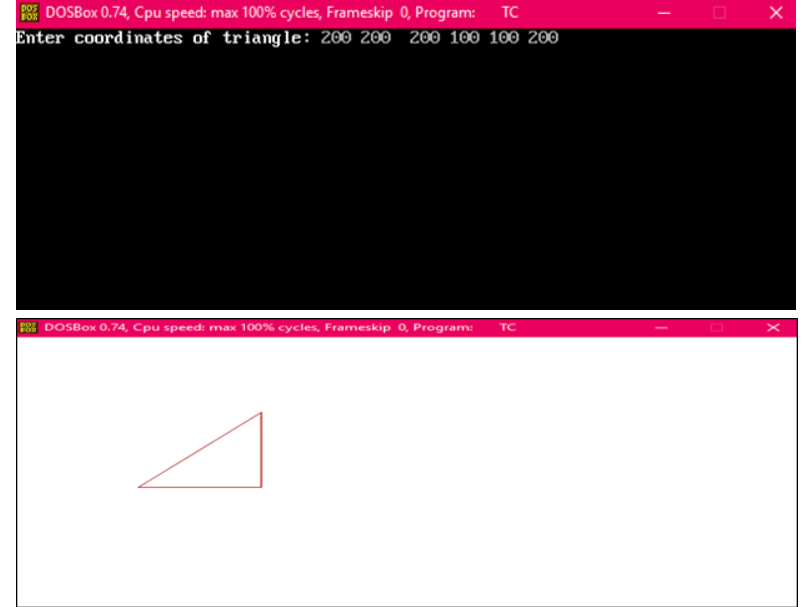
closegraph();

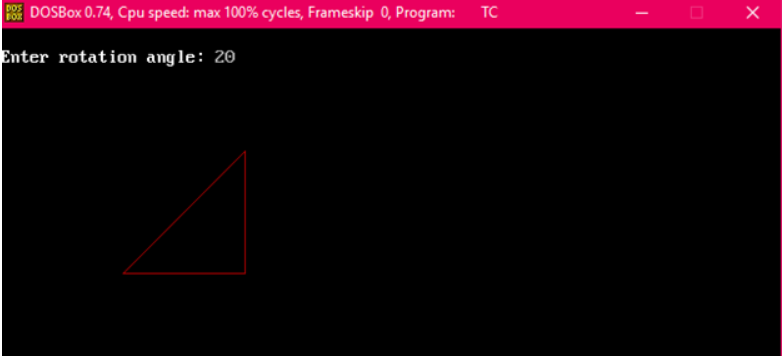
return 0;

}

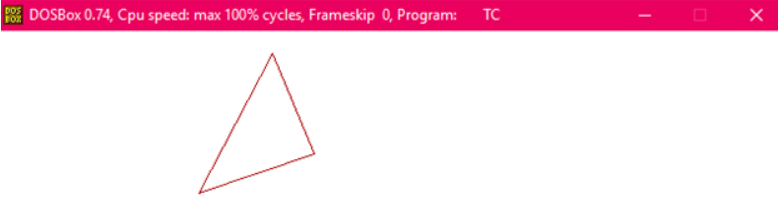
* **Output:**

**Before Rotation-**





**After Rotation-**

****

**2. Scaling:**

// C program to demonstrate scaling of abjects

#include<stdio.h>

#include<graphics.h>

void findNewCoordinate(int s[][2], int p[][1])

{

int temp[2][1] = { 0 };

for (int i = 0; i < 2; i++)

for (int j = 0; j < 1; j++)

for (int k = 0; k < 2; k++)

temp[i][j] += (s[i][k] \* p[k][j]);

p[0][0] = temp[0][0];

p[1][0] = temp[1][0];

}

void scale(int x[], int y[], int sx, int sy)

{

line(x[0], y[0], x[1], y[1]);

line(x[1], y[1], x[2], y[2]);

line(x[2], y[2], x[0], y[0]);

int s[2][2] = { sx, 0, 0, sy };

int p[2][1];

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

{

p[0][0] = x[i];

p[1][0] = y[i];

findNewCoordinate(s, p);

x[i] = p[0][0];

y[i] = p[1][0];

}

line(x[0], y[0], x[1], y[1]);

line(x[1], y[1], x[2], y[2]);

line(x[2], y[2], x[0], y[0]);

}

int main()

{

int x[] = { 100, 200, 300 };

int y[] = { 200, 100, 200 };

int sx = 2, sy = 2;

int gd, gm;

detectgraph(&gd, &gm);

initgraph(&gd, &gm," ");

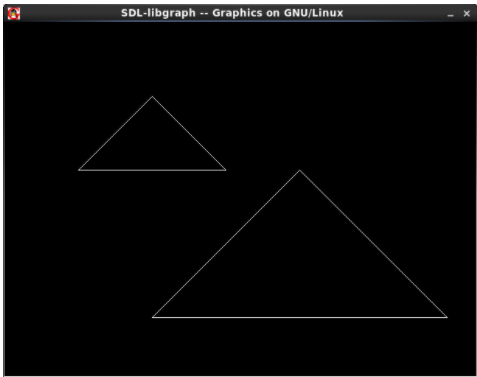
scale(x, y, sx,sy);

getch();

return 0;

}

* Output:



**3. Translation:**

#include<bits/stdc++.h>

#include<graphics.h>

using namespace std;

void translateRectangle ( int P[][2], int T[]){

int gd = DETECT, gm, errorcode;

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

setcolor (2);

rectangle (P[0][0], P[0][1], P[1][0], P[1][1]);

P[0][0] = P[0][0] + T[0];

P[0][1] = P[0][1] + T[1];

P[1][0] = P[1][0] + T[0];

P[1][1] = P[1][1] + T[1];

rectangle (P[0][0], P[0][1], P[1][0], P[1][1]);

}

int main(){

int P[2][2] = {5, 8, 12, 18};

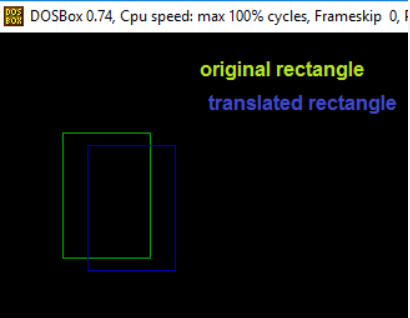
int T[] = {2, 1};

translateRectangle (P, T);

return 0;

}

* **Output:**

****

**EXPERIMENT 13**

**Aim: Write a program to implement Cohen Sutherland line clipping algorithm**.

#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:\\turboc3\\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);

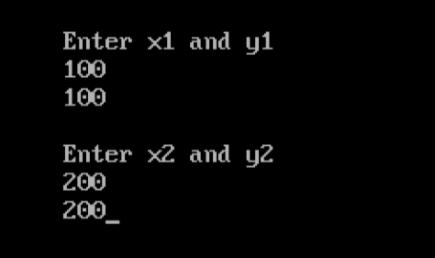
}

else

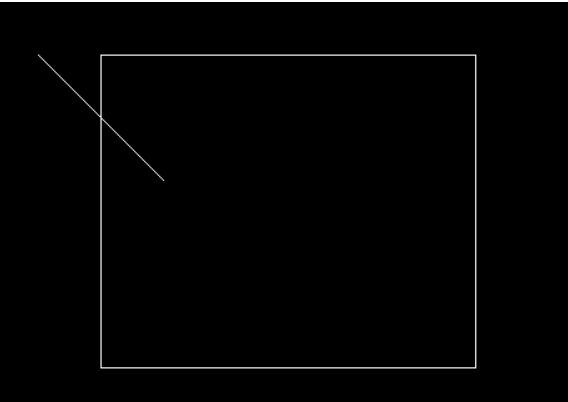
return(p1);

}

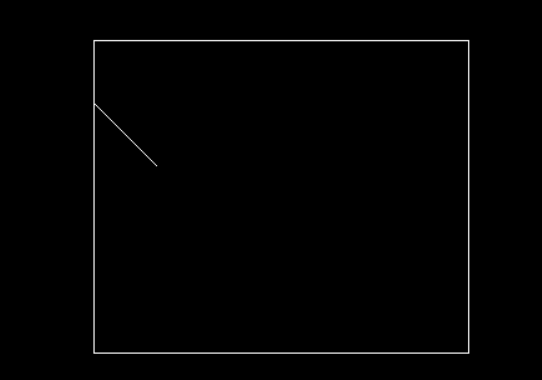
* **Output:**

****

**Before Clipping-**

****

**After Clipping-**

****

**Experiment 14**

**Aim: Write a program to implement Sutherland Hodgeman polygon clipping algorithm.**

#include<stdio.h>

#include<graphics.h>

#include<conio.h>

#include<stdlib.h>

int main()

{

int gd,gm,n,\*x,i,k=0;

int wx1=220,wy1=140,wx2=420,wy2=140,wx3=420,wy3=340,wx4=220,wy4=340;

int w[]={220,140,420,140,420,340,220,340,220,140};//array for drawing window

detectgraph(&gd,&gm);

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

printf("Window:-");

setcolor(RED);

drawpoly(5,w);

printf("Enter the no. of vertices of polygon: ");

scanf("%d",&n);

x = malloc(n\*2+1);

printf("Enter the coordinates of points:\n");

k=0;

for(i=0;i<n\*2;i+=2)

{

printf("(x%d,y%d): ",k,k);

scanf("%d,%d",&x[i],&x[i+1]);

k++;

}

x[n\*2]=x[0];

x[n\*2+1]=x[1];

setcolor(WHITE);

drawpoly(n+1,x);

printf("\nPress a button to clip a polygon..");

getch();

setcolor(RED);

drawpoly(5,w);

setfillstyle(SOLID\_FILL,BLACK);

floodfill(2,2,RED);

gotoxy(1,1);

printf("\nThis is the clipped polygon..");

getch();

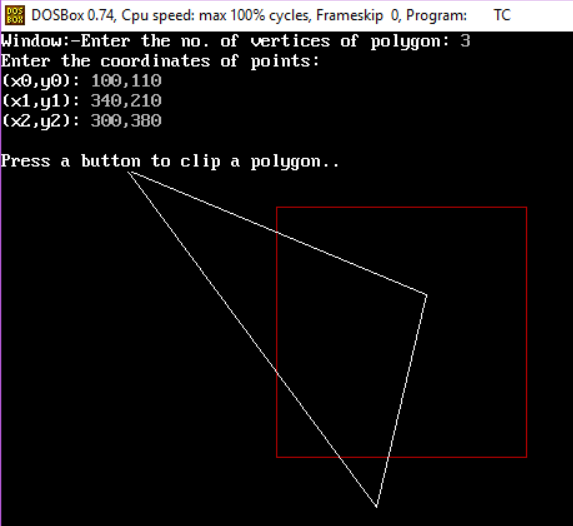
cleardevice();

closegraph();

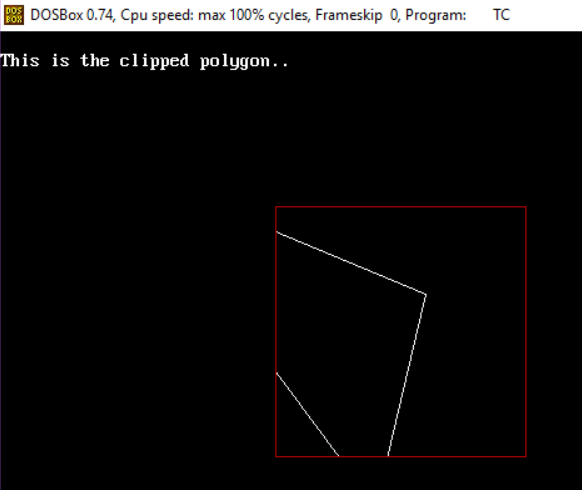
return 0;

}

* **Output:**

****

**Clipped Polygon-**

****