Experiment No. 1

Aim : Program to draw line using DDA algorithm

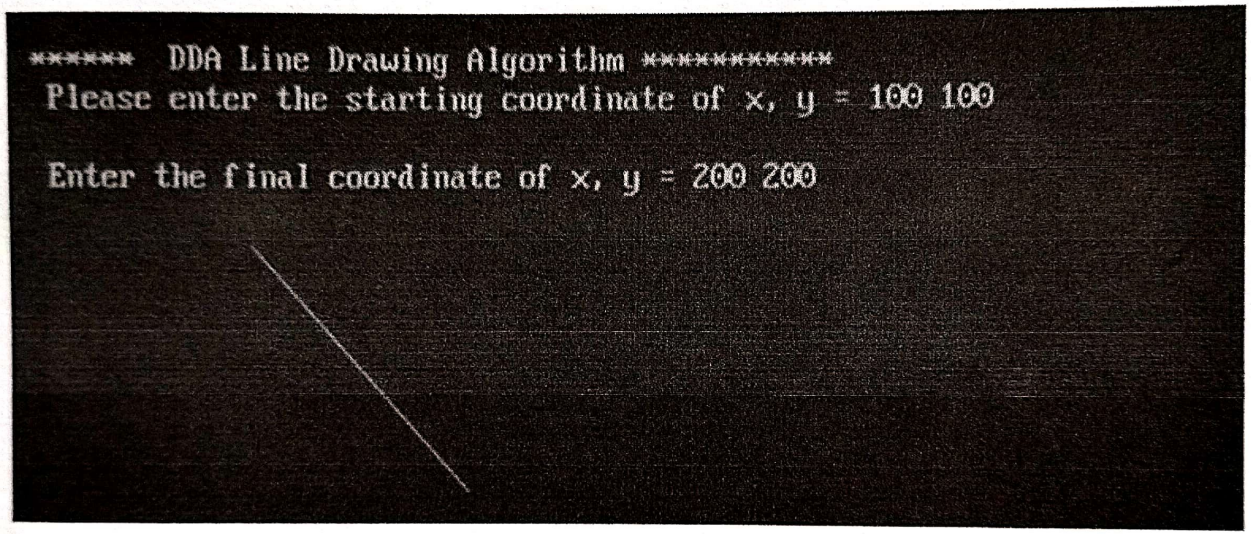
Program Code:

```
#include<graphics.h>
#include<math.h>
void main()
{
int x0,y0,x1,y1,i=0;
float delx,dely,len,x,y;
int gr=DETECT,gm;
initgraph(&gr,&gm,"C:\\TURBOC3\\BGI");
printf("\n****** DDA Line Drawing Algorithm **********");
printf("\n Please enter the starting coordinate of x, y = ");
 scanf("%d %d",&x0,&y0);
printf("\n Enter the final coordinate of x, y = ");
 scanf("%d %d",&x1,&y1);
dely=abs(y1-y0);
 delx=abs(x1-x0);
if(delx<dely)
 len = dely;
else  len=delx;
delx=(x1-x0)/len;
dely=(y1-y0)/len;
x=x0+0.5; y=y0+0.5;
do{
putpixel(x,y,3);
```

Scanned with OKEN Scanner

```
x=x+delx; y=y+dely;

i++;

delay(30);

} while(i<=len);

closegraph();

}
```

Output:



Scanned with OKEN Scanner

Experiment No. 2

Aim: Program to draw circle using Bresenham's algorithm

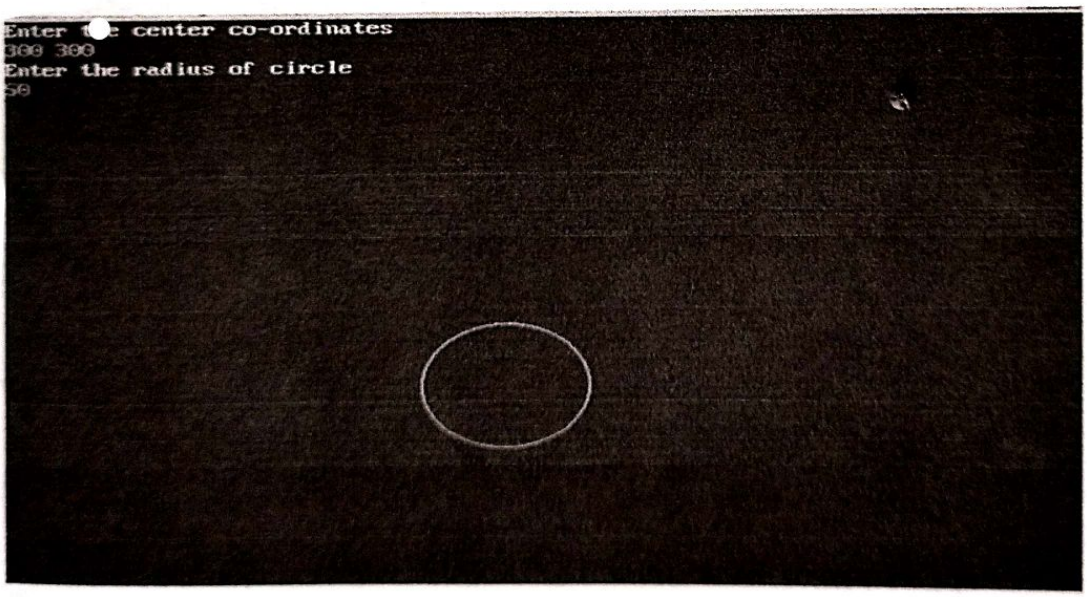
Program Code:

```
#include<iostream.h>
#include<graphics.h>
void drawCircle(int x, int y, int xc, int yc);
void main(){
        int gd = DETECT, gm;
        int r, xc, yc, pk, x, y;
        initgraph(&gd, &gm, "C:\\TC\\BGI");
        cout<<"Enter the center co-ordinates\n"; cin>>xc>>yc;
        cout<<"Enter the radius of circle\n"; cin>>r;
        pk = 3 - 2*r;
        x=0; y = r;
        drawCircle(x,y,xc,yc);
        while(x < y){
                if(pk <= 0){
                        pk = pk + (4*x) + 6;
                        drawCircle(++x,y,xc,yc);
                }
                else{
                        pk = pk + (4*(x-y)) + 10;
                        drawCircle(++x,--y,xc,yc);
                }
        }

        closegraph();
}
```

Scanned with OKEN Scanner

```
void drawCircle(int x, int y, int xc, int yc)

{

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

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

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

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

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

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

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

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

}
```

Output:



Scanned with OKEN Scanner

Experiment No. 3

Aim: Program for 2D transformation – Translation, Scaling, Rotation

Program Code:

```
#include <graphics.h>
#include <stdio.h>
#include<math.h>
void main(){
int gm,gd=DETECT,midx,midy,c;
float x1,x2,x3,y1,y2,y3,x11,x22,x33,y11,y22,y33,sfx,sfy,tpx,tpy,ang,t,a,b;
initgraph(&gd,&gm,"..\\bgi");
midx=getmaxx()/2; midy=getmaxy()/2;
line(midx,0,midx,getmaxy()); line(0,midy,getmaxx(),midy);
printf("2D Translation, Rotation and Scaling\n");
printf("Enter the points (x1,y1), (x2,y2) and (x3,y3) of triangle\n");
scanf("%f%f%f%f%f%f",&x1,&y1,&x2,&y2,&x3,&y3);
setcolor(WHITE);
line(midx+x1,midy-y1,midx+x2,midy-y2);
line(midx+x2,midy-y2,midx+x3,midy-y3);
line(midx+x3,midy-y3,midx+x1,midy-y1);
printf("\n 1.Transaction\n 2.Rotation\n 3.Scalling\n 4.exit\n");
a: printf("Enter your choice:");
scanf("%d",&c);
switch(c){
case 1:
printf("Enter the translation factor\n");
scanf("%f%f",&tpx,&tpy);
x11=x1+tpx; y11=y1+tpy;
x22=x2+tpx; y22=y2+tpy;
```

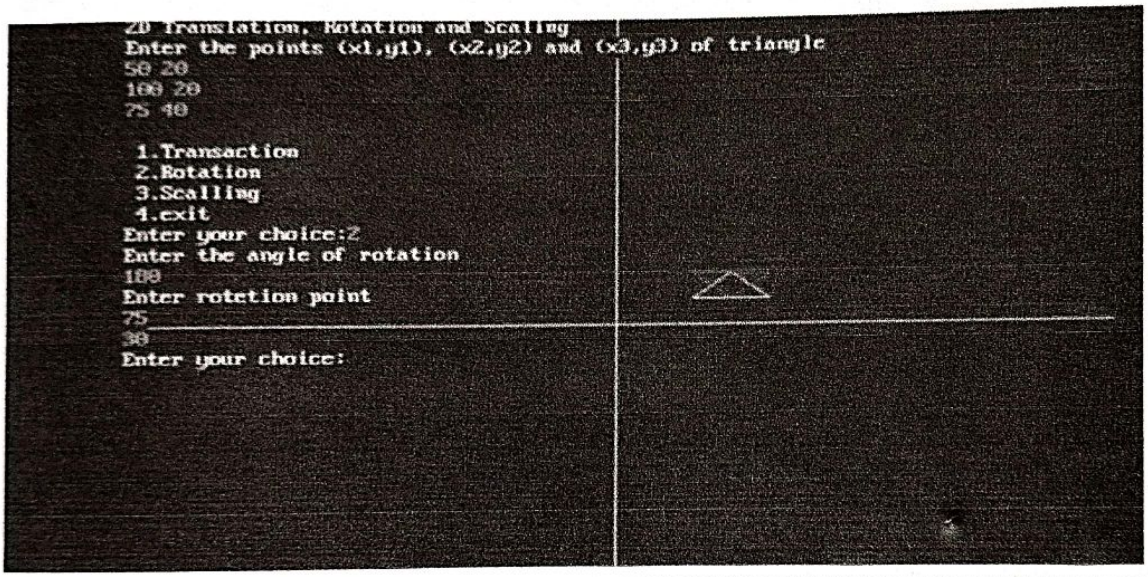
Scanned with OKEN Scanner

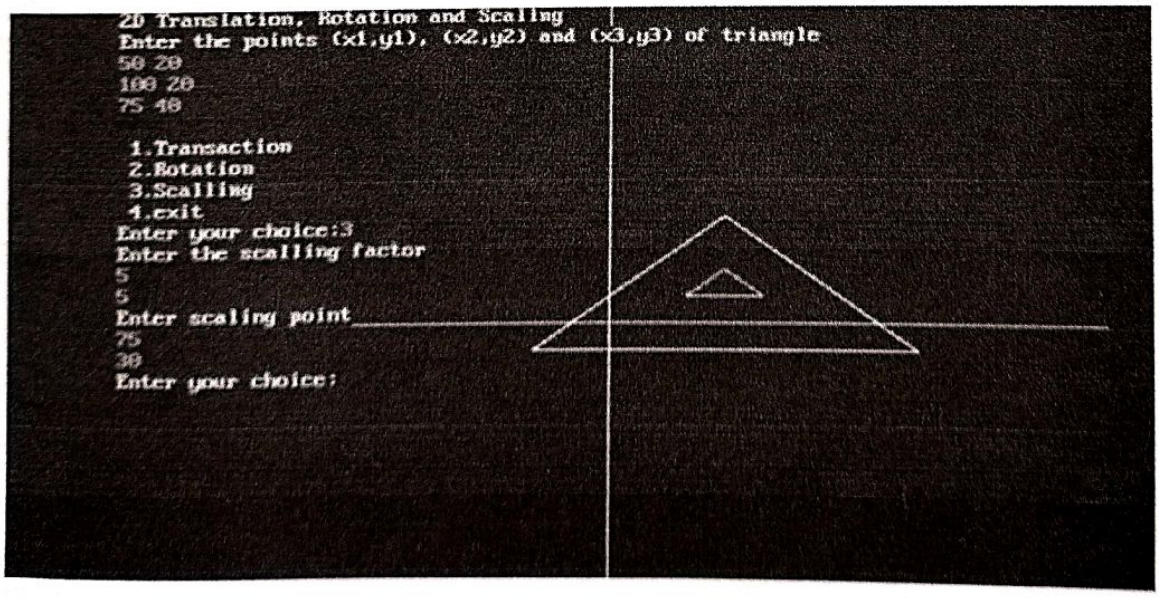
```
x33=x3+tpx; y33=y3+tpy;
setcolor(RED);
line(midx+x11,midy-y11,midx+x22,midy-y22);
line(midx+x22,midy-y22,midx+x33,midy-y33);
line(midx+x33,midy-y33,midx+x11,midy-y11);
break;
case 2:
printf("Enter the angle of rotation\n"); scanf("%f",&ang);
printf("Enter rotetion point\n"); scanf("%f%f",&a,&b);
t=3.14*ang/180;
x11=abs(x1*cos(t)-y1*sin(t)+a*(1-cos(t))+b*sin(t)); y11=abs(x1*sin(t)+y1*cos(t)+b*(1-cos(t))-a*sin(t));
x22=abs(x2*cos(t)-y2*sin(t)+a*(1-cos(t))+b*sin(t)); y22=abs(x2*sin(t)+y2*cos(t)+b*(1-cos(t))-a*sin(t));
x33=abs(x3*cos(t)-y3*sin(t)+a*(1-cos(t))+b*sin(t)); y33=abs(x3*sin(t)+y3*cos(t)+b*(1-cos(t))-a*sin(t));
setcolor(BLUE);
line(midx+x11,  iddy-y11,midx+x22,  iddy-y22);
line(midx+x22,  iddy-y22,midx+x33,  iddy-y33);
line(midx+x33,  iddy-y33,midx+x11,  iddy-y11);
break;
case 3:
printf("Enter the scalling factor\n"); scanf("%f%f",&sfx,&sfy);
printf("Enter scaling point\n"); scanf("%f%f",&a,&b);
x11=x1*sfx+a*(1-sfx); y11=y2*sfy+b*(1-sfy);
x22=x2*sfx+a*(1-sfx); y22=y2*sfy+b*(1-sfy);
x33=x3*sfx+a*(1-sfx); y33=y3*sfy+b*(1-sfy);
setcolor(YELLOW);
line(midx+x11,  iddy-y11,midx+x22,  iddy-y22);
line(midx+x22,  iddy-y22,midx+x33,  iddy-y33);
line(midx+x33,  iddy-y33,midx+x11,  iddy-y11);
```

Scanned with OKEN Scanner

```
break;

case 4: exit(0);

break;

default: printf("Enter the correct choice\n");

}

goto a;

}
```

Output:





Scanned with OKEN Scanner

Experiment No. 4

Aim: Program for 3D transformation- Translation, Scaling, Rotation

Program Code:

```
#include<stdio.h>

#include<graphics.h>

#include<math.h>

void trans();

void scale();

void rotate();

int maxx,maxy,midx,midy;

void main() {

        int ch;

        int gd=DETECT,gm;

        detectgraph(&gd,&gm);

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

        printf("\n 1.Translation \n2.Scaling\n 3.Rotation \n 4.exit");

        printf("enter your choice"); scanf("%d",&ch);

        do {

                switch(ch) {

                        case 1 :        trans();

                                        break;



                        case 2 :        scale();
```

Scanned with OKEN Scanner

```
                    break;

            case 3 :        rotate();

                    break;

            case 4 :    break;

        }

        printf("enter your choice");

        scanf("%d",&ch);

    } while(ch<4);

}

void trans() {

    int x,y,z,o,x1,x2,y1,y2;

    maxx=getmaxx(); maxy=getmaxy();

    midx=maxx/2;  midy=maxy/2;

    bar3d(midx+50,midy-100,midx+60,midy-90,10,1);

    printf("Enter translation factor");

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

    printf("After translation:");

    bar3d(midx+x+50,midy-(y+100),midx+x+60,midy-(y+90),10,1);

}

void scale() {
```

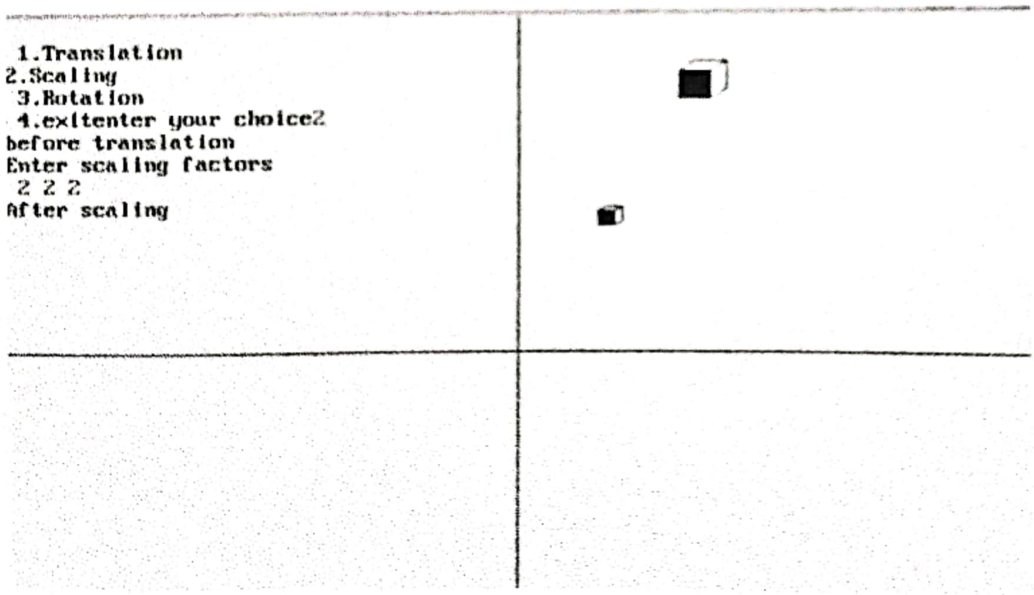
Scanned with OKEN Scanner

```
        int x,y,z,o,x1,x2,y1,y2;
        maxx=getmaxx();
        maxy=getmaxy();
        midx=maxx/2;
        midy=maxy/2;
        bar3d(midx+50,midy-100,midx+60,midy-90,5,1);
        printf("before translation\n");
        printf("Enter scaling factors\n");  scanf("%d %d %d", &x,&y,&z);
        printf("After scaling\n");
        bar3d(midx+(x*50),midy-(y*100),midx+(x*60),midy-(y*90),5*z,1);
}
void rotate()
{
        int x,y,z,o,x1,x2,y1,y2;
        maxx=getmaxx(); maxy=getmaxy();
        midx=maxx/2;  midy=maxy/2;

        bar3d(midx+50,midy-100,midx+60,midy-90,5,1);
        printf("Enter rotating angle");
        scanf("%d",&o);
        x1=50*cos(o*3.14/180)-100*sin(o*3.14/180);
        y1=50*sin(o*3.14/180)+100*cos(o*3.14/180);
```

Scanned with OKEN Scanner

```
x2=60*cos(o*3.14/180)-90*sin(o*3.14/180);

y2=60*sin(o*3.14/180)+90*cos(o*3.14/180);

printf("After rotation  about x axis");

bar3d(midx+50,midy-x1,midx+60,midy-x2,5,1);

printf("After rotation about yaxis");

bar3d(midx+x1,midy-100,midx+x2,midy-90,5,1);

}
```

Output:



Scanned with OKEN Scanner

Experiment No. 5

Aim: Write a program to fill polygon using scanline algorithm

Program Code:

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

#include<dos.h>

#include<graphics.h>

void main() {

int n,i,j,k,gd,gm,dy,dx;

int x,y,temp;

int a[20][2],xi[20];

float slope[20];

printf("\n\Enter no.of edges of polygon"); scanf("%d",&n);

printf("\n\Enter co-ordinates of polygon");

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

printf("\tX%dY%d:",i,i); scanf("%d%d",&a[i][0],&a[i][1]);

}

a[n][0]=a[0][0]; a[n][1]=a[0][1];

detectgraph(&gd,&gm);

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

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

Scanned with OKEN Scanner

```
line(a[i][0],a[i][1],a[i+1][0],a[i+1][1]);

}

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

dy=a[i+1][1]-a[i][1]; dx=a[i+1][0]-a[i][0];

if(dy==0) slope[i]=1.0;

if(dx==0) slope[i]=0.0;

if((dy!=0)&&(dx!=0)) slope[i]=(float)dx/dy;

}

for(y=0;y<480;y++){

k=0;

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

if((a[i][1]<=y)&&(a[i+1][1]>y)){||((a[i][1]>y)&&(a[i+1][1]<=y))xi[k]=(int)(a[i][0]+slope[i]*(y-a[i][1]));

k++;

}

}

for(j=0;j<k-1;j++)

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

if(xi[i]>xi[i+1]){

temp=xi[i];

xi[i]=xi[i+1];
```

Scanned with OKEN Scanner

```
xi[i+1]=temp;

}

}

setcolor(35);

for(i=0;i<k;i+=2){

line(xi[i],y,xi[i+1]+1,y);

}

}

}
```

Output:

Scanned with OKEN Scanner

Experiment No. – 6

Aim – Write a program to draw a line using Bresenham's Algorithm

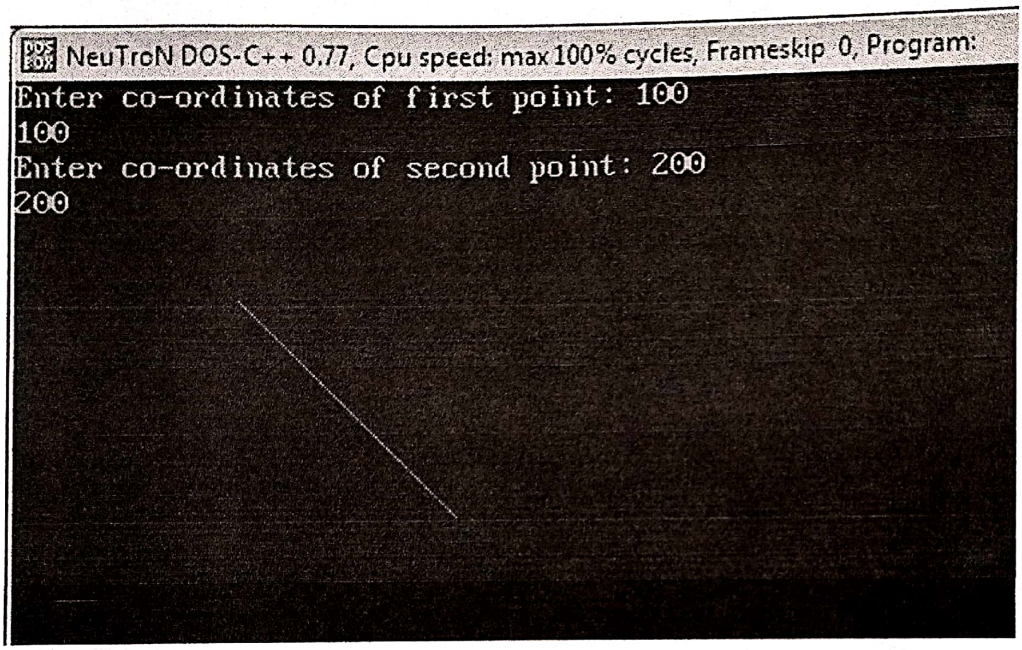
Program –

```
#include<stdio.h>
#include<graphics.h>
void drawline(int x0, int y0, int x1, int y1)
{
   int dx, dy, p, x, y;
   dx=x1-x0;
   dy=y1-y0;
   x=x0;
   y=y0;
   p=2*dy-dx;
   while(x<x1)
   {
      if(p>=0)
      {
         putpixel(x,y,7);
         y=y+1;
         p=p+2*dy-2*dx;
      }
      else
      {
         putpixel(x,y,7);
         p=p+2*dy;}
         x=x+1;
      }
}
int main()
{
   int gdriver=DETECT, gmode, error, x0, y0, x1, y1;
   initgraph(&gdriver, &gmode, "c:\\turboc3\\bgi");
   printf("Enter co-ordinates of first point: ");
   scanf("%d%d", &x0, &y0);
   printf("Enter co-ordinates of second point: ");
```

Scanned with OKEN Scanner

```
    scanf("%d%d", &x1, &y1);
    drawline(x0, y0, x1, y1);
    return 0;
}
```

Output –



Conclusion –

In this way successfully written and executed program for drawing line using Bresenham's algorithm.

Scanned with OKEN Scanner

Experiment No. 7

Aim: Write a program to draw following type of curve: Hilbert's Curve

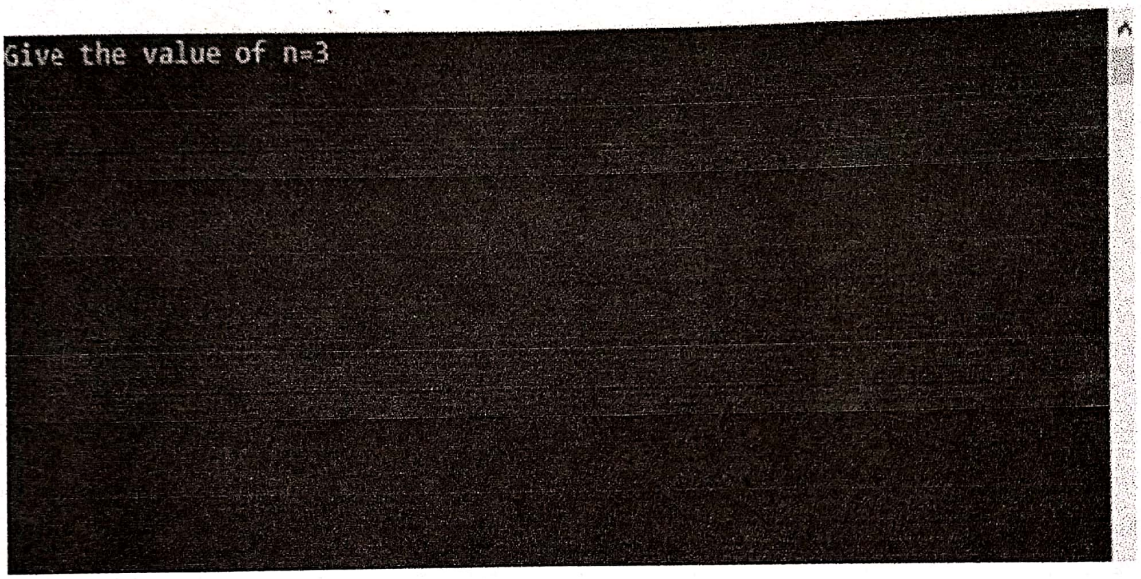
Program Code:

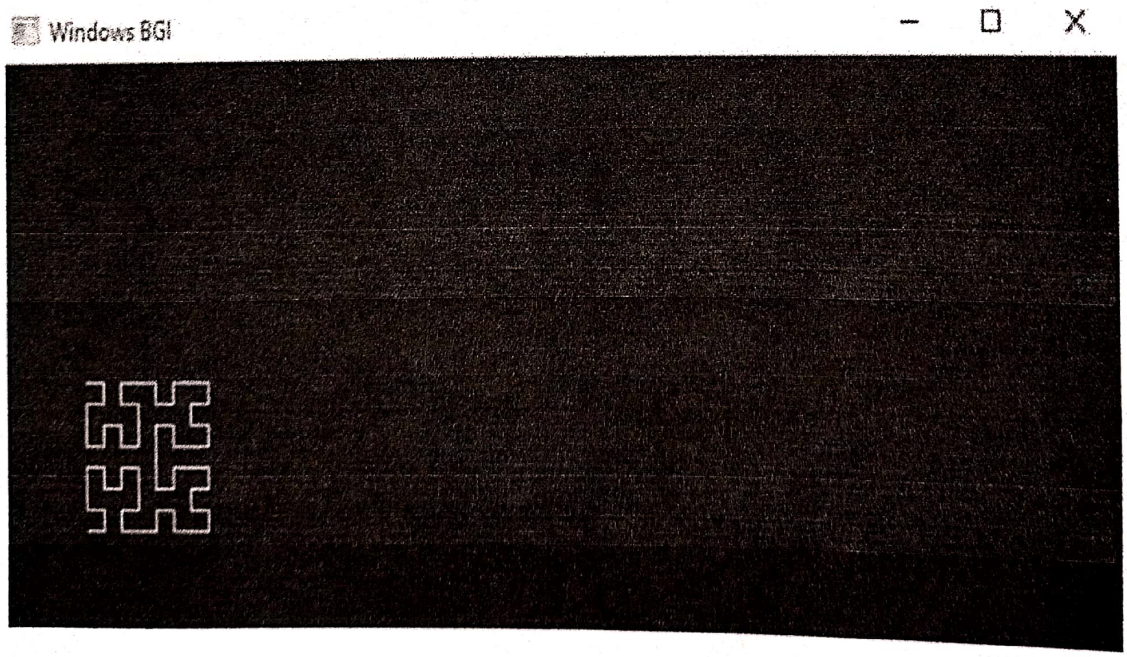
```
#include <iostream.h>
#include <stdlib.h>
#include <graphics.h>
#include <math.h>

void move(int j,int h,int &x,int &y) {
if(j==1)y-=h;
else if(j==2) x+=h;
else if(j==3) y+=h;
else if(j==4) x-=h;
lineto(x,y);
}

void hilbert(int r,int d,int l,int u,int i,int h,int &x,int &y){
if(i>0){
i--;
hilbert(d,r,u,l,i,h,x,y);
move(r,h,x,y);
delay(100);
hilbert(r,d,l,u,i,h,x,y);
move(d,h,x,y);
delay(100);
hilbert(r,d,l,u,i,h,x,y);
move(l,h,x,y);
delay(100);
hilbert(u,l,d,r,i,h,x,y);
}
}

int main()
{
int n,x1,y1;
int x0=50,y0=150,x,y,h=10,r=2,d=3,l=4,u=1;
cout<<"\nGive the value of n: ";
cin>>n;
x=x0;y=y0;
```

Scanned with OKEN Scanner

```
int gm,gd=DETECT;
initgraph(&gd,&gm,"C:\\Turboc3\\bgi");
moveto(x,y);
hilbert(r,d,l,u,n,h,x,y);
delay(100);
closegraph();

return 0;
}
```

Output:





Scanned with OKEN Scanner

Experiment No. - 8

Aim – Write a program of man walking in rain.

Program –

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<stdlib.h>
void main(){
int gr=DETECT,gm;
int i,x,y,j;
initgraph(&gr,&gm,"C:\\TURBOC3\\BGI");

// man
for(j=1;j<600;j=j+5)
{
line(0,400,  00,400);
circle(30+j,280,20); //head
line(30+j,300,30+j,350); //body
line(30+j,330,70+j,330); //hand
if(j%2==0){
line(30+j,350,25+j,400); //left leg
line(30+j,350,10+j,400); // right
}
else{
line(30+j,350,35+j,400); //transition
delay(20);
}
```

Scanned with OKEN Scanner

```
//umbrela
line(70+j,250,70+j,330);
pieslice(70+j,250,180,0,80);
// rain
for(i=0;i<300;i++)
{
x=random(800);
y=random(800);
outtextxy(x,y,"/");
}
delay(170);
cleardevice();
}
getch();
closegraph();
}
```

Output –

Conclusion- In this way successfully written and executed program for man walking in rain.

Scanned with OKEN Scanner

Expi no – 09

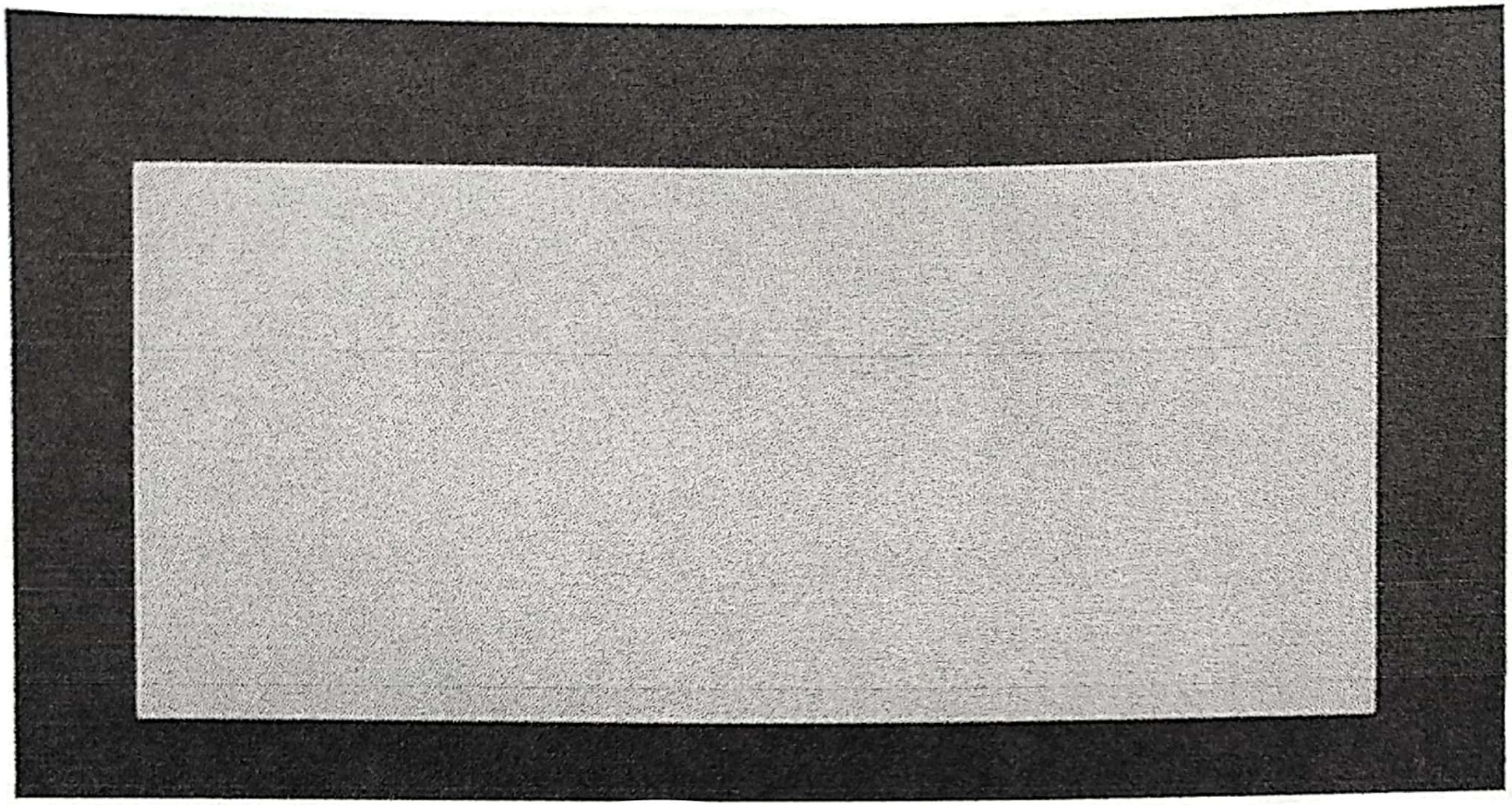
Aim – Write a project to fill colour in rectangle

Program –

```
#include<graphics.h>
#include<conio.h>
int main()
{
        int gd = DETECT, gm;
        initgraph(&gd, &gm, "C:\\turboc3\\bgi");

        // Implementing SOLID_FILL Pattern
        setfillstyle(SOLID_FILL,GREEN);

        // Rectangle Area
        rectangle(250,100,650,280);

        // Flood Fill Function
        floodfill(252,158,15);

        getch();
        closegraph();
        return 0;
}
```

Output –

Scanned with OKEN Scanner



Conclusion – We successfully learn project to fill colour in rectangle

Scanned with OKEN Scanner

Experiment – 10

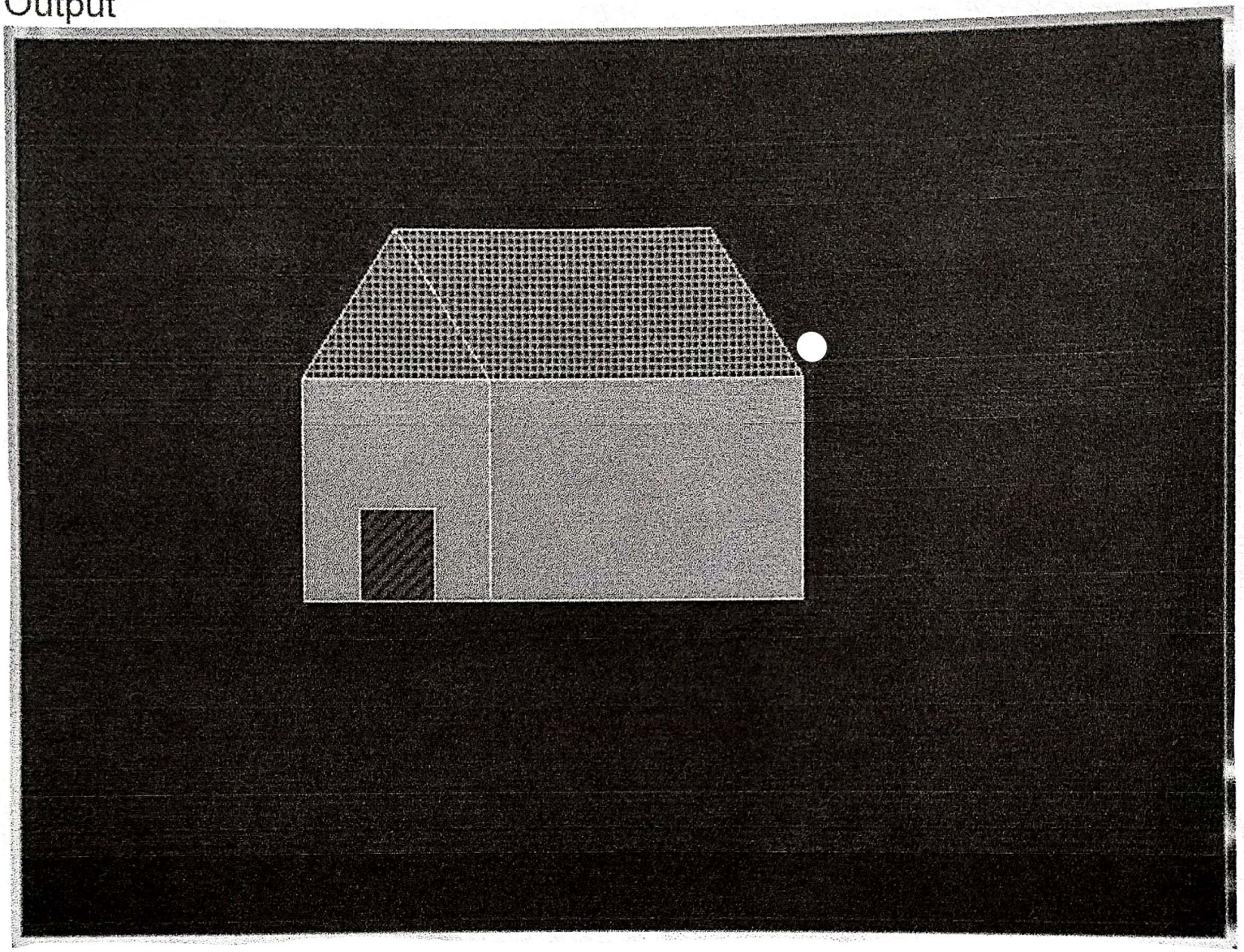
Aim – Write a program to draw a house

Program -

```
#include<graphics.h>

int main(){
 int gd = DETECT,gm;
    initgraph(&gd, &gm, "X:\\TC\\BGI");
    /* Draw Hut */
    setcolor(WHITE);
    rectangle(150,180,250,300);
    rectangle(250,180,420,300);
    rectangle(180,250,220,300);

    line(200,100,150,180);
    line(200,100,250,180);
    line(200,100,370,100);
    line(370,100,420,180);

    /* Fill colours */
    setfillstyle(SOLID_FILL, BROWN);
    floodfill(152, 182, WHITE);
    floodfill(252, 182, WHITE);
    setfillstyle(SLASH_FILL, BLUE);
    floodfill(182, 252, WHITE);
    setfillstyle(HATCH_FILL, GREEN);
    floodfill(200, 105, WHITE);
    floodfill(210, 105, WHITE);

    closegraph();
    return 0;
}
```

Scanned with OKEN Scanner

Output



Scanned with OKEN Scanner

Experiment No. 11

Aim: Write a graphics program for analog curve

Program Code:

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <time.h>
#include <graphics.h>
#include <dos.h>
void calcPoints(int radius, int midx, int midy, int x[12], int y[12]) {
    int x1, y1;
    /* 90, 270, 0, 180 degrees */
    x[0] = midx, y[0] = midy - radius;  x[6] = midx, y[6] = midy + radius;
    x[3] = midx + radius, y[3] = midy;  x[9] = midx - radius, y[9] = midy;

    /* 30, 150, 210, 330 degrees */
    x1 = (int) ((radius / 2) * sqrt(3));  y1 = (radius / 2);
    x[2] = midx + x1, y[2] = midy - y1;  x[4] = midx + x1, y[4] = midy + y1;
    x[8] = midx - x1, y[8] = midy + y1;  x[10] = midx - x1, y[10] = midy - y1;

    /* 60, 120, 210, 300 degrees */
    x1 = radius / 2;  y1 = (int) ((radius / 2)  * sqrt(3));
    x[1] = midx + x1, y[1] = midy - y1;  x[5] = midx + x1, y[5] = midy + y1;
    x[7] = midx - x1, y[7] = midy + y1;  x[11] = midx - x1, y[11] = midy - y1;
    return;
}

void minSecCalc(int radius, int midx, int midy, int secx[60], int secy[60]) {
    int i, j = 0, x, y;
    char str[32];

    /* 15 position(min/sec - 12 to 3) in first quadrant of clock  */
    secx[j] = midx, secy[j++] = midy - radius;

    for (i = 96; i < 180; i = i + 6) {
        secx[j] = midx - (radius * cos((i * 3.14) / 180));
        secy[j++] = midy - (radius * sin((i * 3.14) / 180));
    }
```

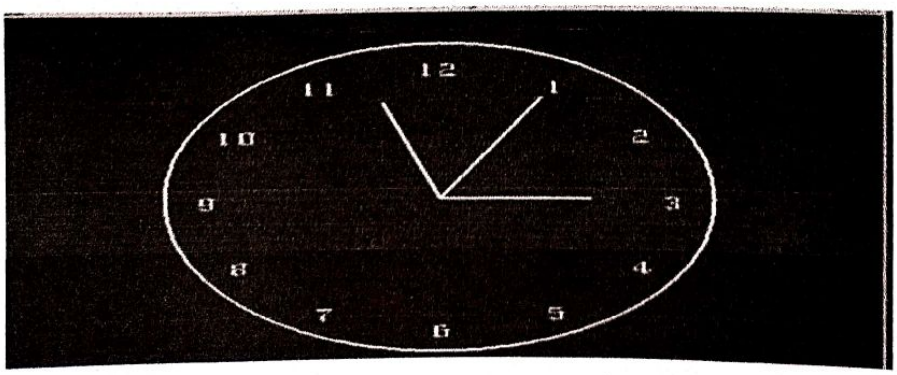
Scanned with OKEN Scanner

```
    secx[j] = midx + radius, secy[j++] = midy;
    for (i = 186; i < 270; i = i + 6) {
        secx[j] = midx - (radius * cos((i * 3.14) / 180));
        secy[j++] = midy - (radius * sin((i * 3.14) / 180));
    }
    secx[j] = midx, secy[j++] = midy + radius;
    for (i = 276; i < 360; i = i + 6) {
        secx[j] = midx - (radius * cos((i * 3.14) / 180));
        secy[j++] = midy - (radius * sin((i * 3.14) / 180));
    }
    secx[j] = midx - radius, secy[j++] = midy;
    for (i = 6; i < 90; i = i + 6) {
        secx[j] = midx - (radius * cos((i * 3.14) / 180));
        secy[j++] = midy - (radius * sin((i * 3.14) / 180));
    }
    return;
}

int main() {
    int gdriver = DETECT, gmode, err, tmp, i, j, midx, midy, radius, hr, min, sec,  x[12], y[12], minx[60],
miny[60],  hrx[12], hry[12], secx[60], secy[60], secx1, secy1;
    char str[256];
    time_t t1;
    struct tm *data;        initgraph(&gdriver, &gmode, "C:/TURBOC3/BGI");
    err = graphresult();

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

    /* mid position in x and y -axis */
    midx = getmaxx() / 2;  midy = getmaxy() / 2;
    radius = 200;

    calcPoints(radius - 30, midx, midy, x, y);
    calcPoints(radius - 90, midx, midy, hrx, hry);
    minSecCalc(radius - 50, midx, midy, minx, miny);
    minSecCalc(radius - 70, midx, midy, secx, secy);
```

Scanned with OKEN Scanner

```
    while (!kbhit()) {
        setlinestyle(SOLID_LINE, 1, 3);
        settextstyle(TRIPLEX_FONT, 0, 3);
        circle(midx, midy, radius);
        for (j = 0; j < 12; j++) {
            if (j == 0) {
                sprintf(str, "%d", 12);
            } else {
                sprintf(str, "%d", j);
            }
            settextjustify(CENTER_TEXT, CENTER_TEXT);
            moveto(x[j], y[j]);
            outtext(str);
        }
        t1 = time(NULL);
        data = localtime(&t1);
        sec = data->tm_sec % 60;
        line(midx, midy, secx[sec], secy[sec]);
        min = data->tm_min % 60;
        line(midx, midy, minx[min], miny[min]);
        hr = data->tm_hour % 12;
        line(midx, midy, hrx[hr], hry[hr]);
        delay(1000);
        cleardevice();
    }
    closegraph();
    return 0;
}
```

Output:



Scanned with OKEN Scanner

Experiment no – 12

Aim – Write a program for moving a cycle .

Program –

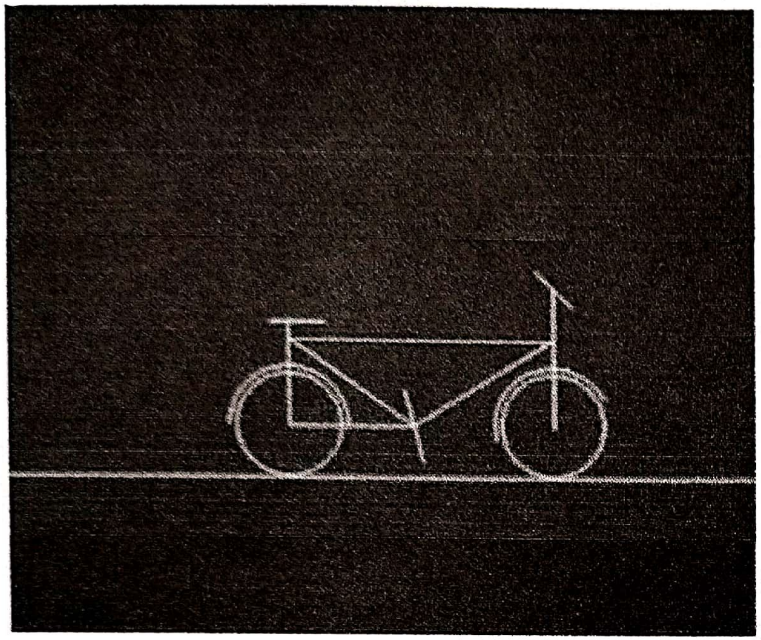
```
// C++ program to draw the moving
// cycle using computer graphics

#include <conio.h>
#include <dos.h>
#include <graphics.h>
#include <iostream.h>

// Driver code
int main()
{
        int gd = DETECT, gm, i, a;

        // Path of the program
        initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

        // Move the cycle
        for (i = 0; i < 600; i++) {
                // Upper body of cycle
                line(50 + i, 405, 100 + i, 405);
                line(75 + i, 375, 125 + i, 375);
                line(50 + i, 405, 75 + i, 375);
                line(100 + i, 405, 100 + i, 345);
                line(150 + i, 405, 100 + i, 345);
                line(75 + i, 345, 75 + i, 370);
                line(70 + i, 370, 80 + i, 370);
                line(80 + i, 345, 100 + i, 345);
```

Scanned with OKEN Scanner

```
            // Wheel
            circle(150 + i, 405, 30);
            circle(50 + i, 405, 30);

            // Road
            line(0, 436, getmaxx(), 436);

            // Stone
            rectangle(getmaxx() - i, 436,
                                650 - i, 431);

            // Stop the screen for 10 secs
            delay(10);

            // Clear the screen
            cleardevice();
        }

        getch();

        // Close the graph
        closegraph();
}
```

Scanned with OKEN Scanner

Output –



Conclusion – We successfully leran about the implementation of moving cycle using graphics.

Scanned with OKEN Scanner