



Vidya Vikas Education Trust's  
Universal College of Engineering, Kaman Road, Vasai-401208  
Accredited B+ Grade NAAC

# **LAB MANUAL**

## **COMPUTER GRAPHICS LAB**

**(CODE: CSL3030)**

Name: **VIVEK SHIVAKUMAR HOTTI**

Class: **SE Computer Engineering**

Semester: **III**

Roll No: **32**

Batch: **A2**

**BY**

**DEPARTMENT OF COMPUTER ENGINEERING**

**COMPUTER GRAPHICS**

**Academic Year: 2020-21**



**Vidya Vikas Education Trust's**  
**Universal College of Engineering, Kaman Road, Vasai-401208**  
**Accredited B+ Grade NAAC**

**Vision:**

To be recognized globally as a department provides quality technical education that eventually caters to helping and serving the community

**Mission:**

- To develop human resources with sound knowledge in theory and practice of computer science and engineering
- To motivate the students to solve real-world problems to help the society grow
- To provide a learning ambience to enhance innovations, team spirit and leadership qualities for students

**Lab Outcome:**

At the end of the course, the students should be able to

1. Implement various output and filled area primitive algorithms
2. Apply transformation, projection and clipping algorithms on graphical objects.
3. Perform curve and fractal generation methods.
4. Develop a Graphical application/Animation based on learned concept

**Description:**

Design and implementation of any case study/ applications /experiments / mini -project based on departmental level courses using modern tools.

**Term work:**

Term Work consists of Total 25 Marks:

Experiments: 10-marks,

Attendance Theory& Practical: 05-marks,

Assignments: 05-marks,

Mini Project: 5-marks

**Practical & Oral:**

Examination is to be conducted based on respective departmental level courses by pair of internal and external examiners appointed by the University of Mumbai.



## INDEX

Sr. No.	Name of the Experiment
□	<b>Experiment List</b>
1	To implement DDA Line Drawing algorithm.
2	To implement Bresenham's Line algorithm.
3	To implement midpoint Circle algorithm.
4	To implement Area Filling Algorithm: Boundary Fill.
5	To implement Area Filling Algorithm: Flood Fill.
6	To implement 2D Transformations: Translation, Scaling, Rotation, Reflection, Shear.
7	To implement Line Clipping Algorithm: Cohen Liang Barsky.
8	To implement Curve: Bezier for n control points
9	To perform Animation (such as Rising Sun, Moving Vehicle, Smileys, Screen saver etc.)
10	To implement Mini-project



## **EXPERIMENT NO. 01**

### ***To Implementation of DDA Line Drawing Algorithm***

#### **AIM:-**

To study and write a program to draw line using DDA algorithm.

#### **THEORY:-**

The process of “Turning ON” the pixels for a line segment is called vector generation. When we want to draw a line that means we have to make the pixels on that line to ON condition i.e. we have to change the intensity of the pixel present on that line. For this task, we have two different algorithms; one of this is DDA algorithm. The algorithm based on increment method. To draw a line, we must know the two endpoints there we are selecting the pixels which lie near the line segment. But it will not be easy to find all such pixels and we use line drawing for big project or in an animation picture may change frequently, in that case it is very efficient.

Digital differential analyzer is based on increment method. The slope intercept equation for line is

$$y = mx + b$$

$$m = (y_2 - y_1) / (x_2 - x_1) \quad m = Dy / Dx$$

Where  $Dy: y_2 - y_1$ ,  $Dx = x_2 - x_1$

For lines with slope  $m < 1$  i.e. lines with positive gentle slopes, we are moving in x-direction by uniform steps for calculating y-value by using

$$Dy / Dx = m$$

$$Dy = m.Dx$$

$$(y_{a+1}) - y_a = m [(x_{a+1}) - x_a]$$

But as we are moving in x-direction by 1 unit i.e. distance between two columns



$$(x_{a+1}) - x_a = 1$$

$$(y_{a+1}) - y_a = m(1)$$

$$y_{a+1} = m + y_a$$

$$y(\text{new}) = \text{slope} + y(\text{old})$$

$$Dx = (Dy/m)$$

$$(x_{a+1}) - x_a = [(y_{a+1}) - y_a] / m$$

$$[(y_{a+1}) - y_a] \text{ is } 1 \text{ unit i.e. distance between rows}$$

$$(x_{a+1}) - x_a = (1/m) \quad x_{a+1} = [(1/m) + x_a]$$

$$x(\text{new}) = [(1/\text{slope}) + x(\text{old})]$$

When slope is 1 i.e.  $(Dy/Dx) = m = 1$

$$Dy = Dx$$

$$y(\text{new}) = y(\text{old})$$

**Ceiling:** This is a function which returns smallest integer which is greater than or equal to its argument suppose value is 8.6, ans is 9.

**Floor:** This is a function which returns largest integer which is less than or equal to its argument. If value is 8.2 and if we passing this value to floor function it will return 8.

### Algorithm

Step 1: Start

Step 2: Read starting co-ordinates for line is ending co-ordinates for line.

Step 3:

Function dda (int x1, int y1, int x2, int y2)

Begin

$$Dx \leftarrow x_2 - x_1$$

$$Dy \leftarrow y_2 - y_1$$

$$\text{If } (\text{abs}(Dx) \geq \text{abs}(Dy))$$



**Vidya Vikas Education Trust's**  
**Universal College of Engineering, Kaman Road, Vasai-401208**  
**Accredited B+ Grade NAAC**

```
Then,
    Steps <- Dx
Else
    Steps <- Dy
X(inc) <- [(float) dx] / steps
Y(inc) <- [(float) dy] / steps
x <- x1
y <- y1
Function put pixel (x, y, WHITE);
    for (I = 1 to steps, I <- I+1)
        Begin
            x <- x+ x(inc)
            y <- y + y(inc)
            x1 <- x + 0.5
            y1 <- y + 0.5
            put pixel (x1, y1, WHITE)
        end
    end
end
Step 4: stop
```

**CODE:-**

```
#include<stdio.h>
#include<graphics.h>
int abs (int n)
{
    return ( (n>0) ? n : ( n * (-1)));
}
void DDA(int X0, int Y0, int X1, int Y1)
{
    int i;
    int dx = X1 - X0;
    int dy = Y1 - Y0;

    int steps = abs(dx) > abs(dy) ? abs(dx) :
    abs(dy);
    float Xinc = dx / (float) steps;
    float Yinc = dy / (float) steps;
    float X = X0;
    float Y = Y0;
    for (i = 0; i <= steps; i++)
    {
        putpixel (X,Y,GREEN);
        X += Xinc;
        Y += Yinc;
    }
}
```



**Vidya Vikas Education Trust's**  
**Universal College of Engineering, Kaman Road, Vasai-401208**  
**Accredited B+ Grade NAAC**

```
delay(100);
```

```
}
```

```
}
```

```
int main()
```

```
{
```

```
int gd = DETECT, gm;
```

```
int x0,x1,y0,y1;
```

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

```
printf("Enter x0, y0, x1, y1");
```

```
scanf("%d%d%d%d",&x0,&y0,&x1,&y1);
```

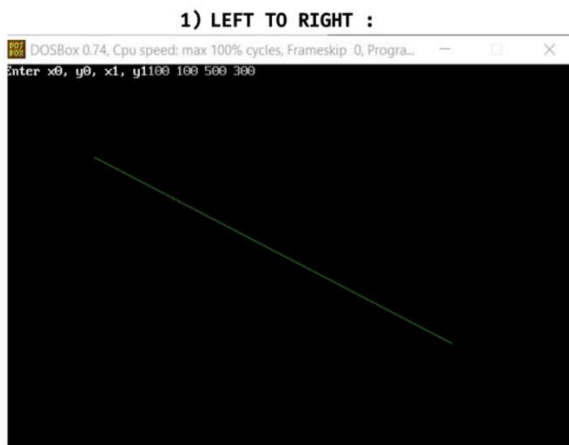
```
DDA(x0,y0,x1,y1);
```

```
getch();
```

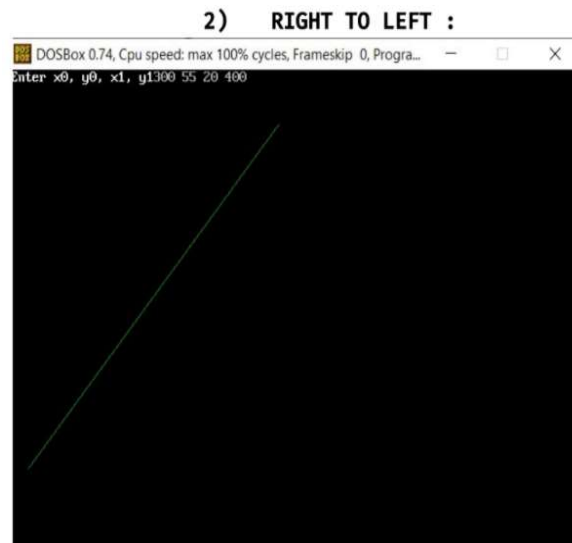
```
return 0;
```

```
}
```

**OUTPUT:-**



Inputs:  $(x_0, y_0) \rightarrow (100, 100)$   
 $(x_1, y_1) \rightarrow (500, 300)$



Inputs:  $(x_0, y_0) \rightarrow (300, 55)$   
 $(x_1, y_1) \rightarrow (20, 400)$

**CONCLUSION:- Hence we wrote a program to implement DDA Line Algorithm**



## **EXPERIMENT NO. 02**

### ***To Implementation of Bresenham's Line Drawing Algorithm***

#### **AIM:-**

To study and write a program to draw line using Bresenham's line Drawing algorithm

#### **THEORY:-**

There is one more generation algorithm for lines which is called Bresenham's line algorithm. For gentle slope case, consider P as height of pixel  $P > 0.5$

$$2P - 1 > 0$$

$$P = P + \text{slope} \text{ or } P = (P + \text{slope}) - 1$$

But slope is nothing but  $(dy) / (dx)$

$$2Pdx - dx > 0$$

$$\text{Let } G = 2Pdx - dx$$

Our test will become as  $G > 0$

$$\text{As } G = 2Pdx - dx$$

$$G + dx = 2Pdx$$

$$[(G + dx) / 2dx] = P$$

$$[(G + dx) / 2dx] = [(G + dx) / 2dx] + (dy / dx)$$

$$G = G + 2dy$$

$$\text{Similarly, } P = P + \text{slope} - 1$$

$$G = G + 2dy - 2dx$$

If we used  $G > 0$

$$G = 2dx.P - dx$$





$$P = (dx / dy)$$

$$G = [2dx (dy / dx)] - dx$$

$$G = 2dy - dx$$

### Algorithm

Step 1: START

Step 2: Accept the starting and ending co-ordinate of line.

Step 3:

Function bresen(x1, y1, x2, y2)

```
{  
    Begin  
    dx <- abs (x2- x1)  
    dy <- abs (y2-y1)  
    x <- x1  
    y <- y1  
    s1 <- sign (x2-x1)  
    s2 <- sign (y2-y1)  
    If (dy > dx)  
    Then  
    Begin  
        temp <- dx  
        dx <- dy  
        dy <- temp  
        exchange <- 1  
    end then  
    exchange <- 0  
    g <- (2* dy) - dx  
    i <- 1  
    while (i <=dx)  
    Begin  
        putpixel (x, y, WHITE)
```



**Vidya Vikas Education Trust's**  
**Universal College of Engineering, Kaman Road, Vasai-401208**  
**Accredited B+ Grade NAAC**

```
    delay (10)
    WHILE (g >=0)
    Begin
    If (exchange ==1)
        i <- i+1
    end
end
Step 4:
Function int sign (float arg)
Begin
If (arg <0)
Then
return - 1
else If (arg==0)
then
return 0
else
return 1
end
Step 5: Display the line
Step 6: Stop
```

**CODE:-**

**//Vivek Shivakumar Hotti: SE-COMPS-Roll-32; 20-10-2020 (Tuesday)**

```
#include<graphics.h>
#include<stdio.h>
#include<math.h>
int main()
{
    int
    gd,gm,x,y,end,p,x1,x2,y1,y2,dx
    ,dy;
    detectgraph(&gd,&gm);
    initgraph(&gd,&gm,"C://Turb
oC3//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);
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,YELLOW);
while(x<=end)
{
if(p<0)
{
x++;
p=p+2*dy;
}
else
{
x++;
y++;
p=p+2*(dy-dx);
}
putpixel(x,y,YELLOW);
delay(100);
}
getch();
closegraph();
return 0;
}
```



**OUTPUT:-**

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...

```
Enter the value of x1: 200
Enter the value of y1: 250
Enter the value of x2: 300
Enter the value of y2: 400
```

The screenshot shows a DOSBox window with a black background. At the top, the title bar reads 'DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...'. Below the title bar, there are four lines of white text: 'Enter the value of x1: 200', 'Enter the value of y1: 250', 'Enter the value of x2: 300', and 'Enter the value of y2: 400'. After these inputs, a yellow line is drawn on the black background, starting from the point (200, 250) and ending at the point (300, 400).

**CONCLUSION:-** Hence we wrote a program to Implement Bradenham's Line Drawing Algorithm.



### **EXPERIMENT NO. 03**

#### ***To Implementation of Midpoint Circle Algorithm***

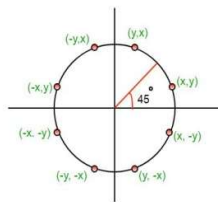
##### **AIM:-**

To study and write a program to draw line using Midpoint Circle Algorithm

##### **THEORY:-**

The **mid-point** circle drawing algorithm is an algorithm used to determine the points needed for rasterizing a circle.

We use the **mid-point** algorithm to calculate all the perimeter points of the circle in the **first octant** and then print them along with their mirror points in the other octants. This will work because a circle is symmetric about its centre.



The algorithm is very similar to the [Mid-Point Line Generation Algorithm](#). Here, only the boundary condition is different.

For any given pixel (x, y), the next pixel to be plotted is either (x, y+1) or (x-1, y+1). This can be decided by following the steps below.

1. Find the mid-point **p** of the two possible pixels i.e (x-0.5, y+1)
2. If **p** lies inside or on the circle perimeter, we plot the pixel (x, y+1), otherwise if it's outside we plot the pixel (x-1, y+1)

**Boundary Condition :** Whether the mid-point lies inside or outside the circle can be decided by using the formula:-

Given a circle centered at (0,0) and radius r and a point p(x,y)

$$F(p) = x^2 + y^2 - r^2$$

if  $F(p) < 0$ , the point is inside the circle

$F(p) = 0$ , the point is on the perimeter

$F(p) > 0$ , the point is outside the circle

##### **ALGORITHM:-**

**Step1:** Put  $x = 0$ ,  $y = r$  in equation 2  
We have  $p = 1 - r$



**Vidya Vikas Education Trust's**  
**Universal College of Engineering, Kaman Road, Vasai-401208**  
**Accredited B+ Grade NAAC**

**Step2:** Repeat steps while  $x \leq y$

Plot (x, y)

If ( $p < 0$ )

Then set  $p = p + 2x + 3$

Else

$p = p + 2(x-y)+5$

$y = y - 1$  (end if)

$x = x + 1$  (end loop)

**Step3:** End

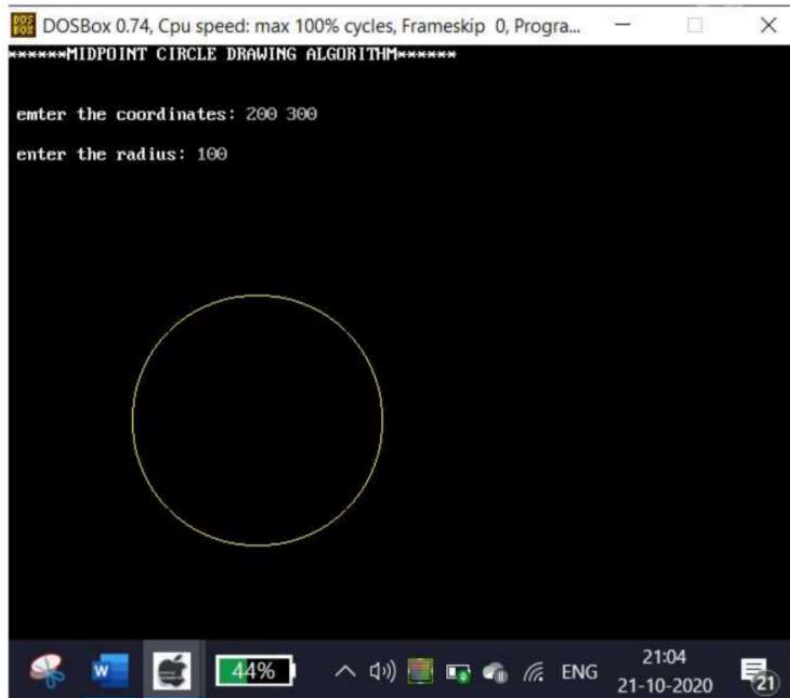
**CODE:-**

**//Vivek Shivakumar Hotti: SE-COMPS-Roll-32; 21-10-2020 (Wednesday)**

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
void main()
{
int x,y,x_mid,y_mid,radius,dp;
int g_mode,g_driver=DETECT;
clrscr();
initgraph(&g_driver,&g_mode,"C:\\\\TURBOC3\\\\BGI");
printf("**** MID POINT Circle drawing algorithm ***\n\n");
printf("\nenter the coordinates= ");
scanf("%d %d",&x_mid,&y_mid);
printf("\n now enter the radius =");
scanf("%d",&radius);
x=0;
y=radius;
dp=1-radius;
do
{
putpixel(x_mid+x,y_mid+y,YELLOW);
putpixel(x_mid+y,y_mid+x,YELLOW);
putpixel(x_mid-y,y_mid+x,YELLOW);
putpixel(x_mid-x,y_mid+y,YELLOW);
putpixel(x_mid-x,y_mid-y,YELLOW);
putpixel(x_mid-y,y_mid-x,YELLOW);
putpixel(x_mid+y,y_mid-x,YELLOW);
putpixel(x_mid+x,y_mid-y,YELLOW);
if(dp<0) {
dp+=(2*x)+1;
}
else{
y=y-1;
dp+=(2*x)-(2*y)+1;
}
x=x+1;
}while(y>x); getch();}
```



**OUTPUT:-**



**CONCLUSION:-** Hence we wrote a program to implement Midpoint Circle Algorithm.



## **EXPERIMENT NO. 04**

### ***To Implementation of Boundary Fill***

**AIM:-** To study polygon filling algorithm (Boundary fill algorithm).

#### **THEORY:-**

Boundary Fill Method:

In this method, edges of the polygon are drawn then starting with some point inside the polygon called as subpoint upto boundary pixel, the pixel are highlighted. This process is continued till boundary pixels are reached. To define the region within boundary 4 connected for 8 connected region is used.

In this, every pixel will reach may be combinational move in 4-direction or 8-directional respectively.

#### **Algorithm**

Step 1: Start

Step 2: Read number of edges of the polygon and Seed point coordinates.

Boundary fill(x, y, fcolor, bcolor)

{

If (get pixel (x,y) != b-color and get pixel(x,y) != fcolor )

Then

{

Putpixel (x, y, fcolor)

boundary\_fill (x, y+1, fcolor, bcolor)

boundary\_fill (x, y-1, fcolor, bcolor)

boundary\_fill (x+1, y, fcolor, bcolor)

boundary\_fill (x-1, y, fcolor, bcolor) }

}

Step 4: stop





**CODE:-**

**//Vivek Shivakumar Hotti: SE-COMPS-Roll-32; 22-10-2020 (Wednesday)**

```
#include<stdio.h>
#include<graphics.h>
void boundary_fill(int x,int y,int
boundary_color,int fill_color);
int main()
{
int gd,gm,x,y,x1,x2,y1,y2;
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"C://TurboC
3//BGI");
printf("Enter top-left point
of rectangle: ");
scanf("%d%d",&x1,&y1);
printf("Enter bottom-right
point of rectangle: ");
scanf("%d%d",&x2,&y2);
cleardevice();
setcolor(WHITE);
rectangle(x1,y1,x2,y2);
boundary_fill(x1+1,y1+1,15,6);
getch();
closegraph();
return 0;
}
void boundary_fill(int x,int y,int
boundary_color,int fill_color)
{
int current;
current=getpixel(x,y);
if(current!=boundary_color &&

current!=fill_color)
{
putpixel(x,y,fill_color);
delay(10);
boundary_fill(x+1,y,boundary_c
olor,fill_color);
boundary_fill(x,y+1,boundary_c
olor,fill_color);
boundary_fill(x-1,y,boundary_c
olor,fill_color);
boundary_fill(x,y-1,boundary_c
olor,fill_color);
}
}
```



Vidya Vikas Education Trust's  
Universal College of Engineering, Kaman Road, Vasai-401208  
Accredited B+ Grade NAAC

**OUTPUT:-**

```
Enter top-left point of rectangle: 10 20  
Enter bottom-right point of rectangle: 80 90
```



**CONCLUSION: - Hence we wrote a program to implement Boundary Fill Algorithm**



## **EXPERIMENT NO. 05**

### ***To Implementation of Flood Fill***

#### **AIM:-**

To study and implement polygon filling algorithm (Flood fill algorithm).

#### **THEORY:-**

Filling the polygon means highlighting all the pixels which lie inside the polygon with any color other than background color polygon are either to fill since they have linear boundaries.

The two basic approaches to fill the polygon they are:

1. Seed fill algorithm
2. Scan line algorithm.

#### **Algorithm**

Step 1: Start

Step 2: Read edges of the polygon and Seed point.

Floodfill (x, y, o-color, n-color)

{

If (getpixel (x,y)==o-color)

Then

{

Putpixel(x, y, o-color, n-color)

floodfill (x, y+1, o-color, n-color)

floodfill (x, y-1, o-color, n-color)

floodfill (x+1, y, o-color, n-color)

floodfill (x+1, y+1, o-color, n-color)

floodfill (x+1, y-1, o-color, n-color)

floodfill (x-1, y+1, o-color, n-color)



**Vidya Vikas Education Trust's**  
**Universal College of Engineering, Kaman Road, Vasai-401208**  
**Accredited B+ Grade NAAC**

```
        floodfill (x-1, y-1, o-color, n-color)
        floodfill (x-1, y, o-color, n-color)
    }
}
```

Step 4: stop

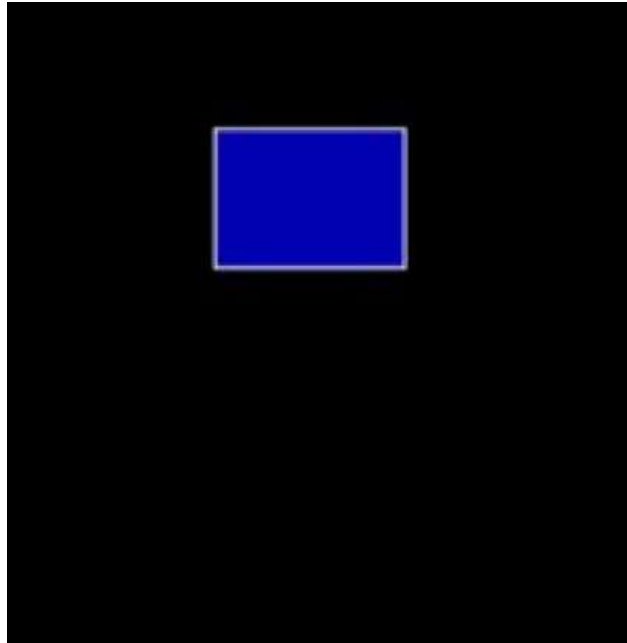
**CODE:-**

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void flood(int,int,int,int);
void main()
{
    intgd=DETECT,gm;
    initgraph(&gd,&gm,"C:/TURBOC3/bgi");
    rectangle(50,55,120,106);
    flood(51,56,1,0);
    getch();
}
void flood(int x,int y,int fillColor, int defaultColor)
{
    if(getpixel(x,y)==defaultColor)
    {
        delay(10);
        putpixel(x,y,fillColor);
        flood(x+1,y,fillColor,defaultColor);
        flood(x-1,y,fillColor,defaultColor);
        flood(x,y+1,fillColor,defaultColor);
        flood(x,y-1,fillColor,defaultColor);
    }
}
```



**Vidya Vikas Education Trust's  
Universal College of Engineering, Kaman Road, Vasai-401208  
Accredited B+ Grade NAAC**

**OUTPUT:-**



**CONCLUSION:- Hence we wrote a program to Implement Flood Fill Algorithm.**



## **EXPERIMENT NO. 06**

### ***To Implementation of 2D Transformations***

#### **AIM:-**

To study and write the program of 2D Transformation operation.

#### **THEORY:-**

One of the most common and important tasks in computer graphics is to transform the coordinates (position, orientation, and size) of either objects within the graphical scene or the camera that is viewing the scene. It is also frequently necessary to transform coordinates from one coordinate system to another, (e.g. world coordinates to viewpoint coordinates to screen coordinates.) All of these transformations can be efficiently handled using some simple matrix representations, which we will see can be particularly useful for combining multiple transformations into a single composite transform matrix.

#### **Translation in 2D**

point (X,Y) is to be translated by amount Dx and Dy to a new location (X',Y')

$$X' = Dx + X$$

$$Y' = Dy + Y$$

or  $P' = T + P$  where

$$P' = \begin{bmatrix} \bar{X}' \\ \bar{Y}' \\ - \end{bmatrix}$$

$$T = \begin{bmatrix} \bar{Dx} \\ \bar{Dy} \\ - \end{bmatrix}$$

$$P = \begin{bmatrix} \bar{X} \\ \bar{Y} \\ - \end{bmatrix}$$

---

#### **Scaling in 2D**



point (X,Y) is to be scaled by amount  $S_x$  and  $S_y$  to location (X',Y')

$$X' = S_x * X$$

$$Y' = S_y * Y$$

or  $P' = S * P$  where

$$P' = \begin{bmatrix} X' \\ Y' \end{bmatrix}$$

$$S = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$$

$$P = \begin{bmatrix} X \\ Y \end{bmatrix}$$

scaling is performed **about the origin (0,0)** not about the center of the line/polygon/whatever

Scale > 1 enlarge the object and move it away from the origin.

Scale = 1 leave the object alone

Scale < 1 shrink the object and move it towards the origin.

uniform scaling:  $S_x = S_y$

differential scaling  $S_x \neq S_y \rightarrow$  alters proportions

---

## Rotation in 2D

point (X,Y) is to be rotated about the origin by angle theta to location (X',Y')

$$X' = X * \cos(\theta) - Y * \sin(\theta)$$

$$Y' = X * \sin(\theta) + Y * \cos(\theta)$$

note that this does involve sin and cos which are much more costly than addition or multiplication

or  $P' = R * P$  where

$$P' = \begin{bmatrix} X' \\ Y' \end{bmatrix}$$



$$R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

$$P = \begin{bmatrix} X \\ Y \end{bmatrix}$$

### Algorithm

Step 1: Start

Step 2: Read number of vertices and coordinates of the vertices.

Step 3:

Select operation to be performed on the polygon from translation, scaling, rotation, reflection or shear.

Step 4. Show the output according to selection of the operation. If you want to perform other operation on the polygon select 1(YES) or 0(NO).

If choice is 1 Repeat Step 3

Else go to 4.

Step 4: Stop

### CODE:-

//Vivek Shivakumar Hotti: SE-COMPS-Roll-32; 23-10-2020 (Friday)

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<graphics.h>
int ch,x,y,az,i,w,ch1,ch2,xa,ya,ra,a
[10],b[10],da,db;
float x1,y1,az1,w1,dx,dy,theta,x1s,y
1s,sx,sy,a1[10],b1[10];
int main()
{
int gm ,gr;
detectgraph(&gm,&gr);
initgraph(&gm,&gr,"C:/TC/BGI");
printf("Enter the upper left corner
of the rectangle:\n");
scanf("%d%d",&x,&y);
printf("Enter the lower right corner
of the rectangle:\n");
scanf("%d%d",&az,&w);
rectangle(x,y,az,w);
da=az-x;
db=w-y;
a[0]=x;
b[0]=y;
a[1]=x+da;
b[1]=y;
a[2]=x+da;
b[2]=y+db;
a[3]=x;b[3]=y+db;

while(1)
{
printf("*****2D Transformations*****
***\n");
printf("1.Translation\n 2.Rotation\n
```





**Vidya Vikas Education Trust's**  
**Universal College of Engineering, Kaman Road, Vasai-401208**  
**Accredited B+ Grade NAAC**

```
3.Scaling\n 4.Reflection\n 5.Sheari
ng \n 6.Exit \n Enter your choice: \
n");
scanf("%d",&ch);
switch(ch)
{
case 1:
detectgraph(&gm,&gr);
initgraph(&gm,&gr,"C:/TC/BGI");
rectangle(x,y,az,w);
printf("*****Translation*****\n\
n");
printf("Enter the value of shift vec
tor:\n");
scanf("%f%f",&dx,&dy);
x1=x+dx;
y1=y+dy;
az1=az+dx;
w1=w+dy;
rectangle(x1,y1,az1,w1);
break;
case 2:
detectgraph(&gm,&gr);
initgraph(&gm,&gr,"C:/TC/BGI");
rectangle(x,y,az,w);
```

```
2 of 6
printf("*****Rotation*****\n\n")
;
printf("Enter the value of fixed poi
nt and angle of rotation:Enter the v
alue of fixed point and angle of rot
ation:\n");
scanf("%d%d%d",&xa,&ya,&ra);
theta=(float)(ra*(3.14/180));
for(i=0;i<4;i++)
{
a1[i]=(xa+((a[i]-xa)*cos(theta)-(-b[
i]-ya)*sin(theta)));
b1[i]=(ya+((a[i]-xa)*sin(theta)+(-b[
i]-ya)*cos(theta)));
}
for(i=0;i<4;i++)
{
if(i!=3)
line(a1[i],b1[i],a1[i+1],b1[i+1]);
else
line(a1[i],b1[i],a1[0],b1[0]);
}
break;
```

```
case 3:
detectgraph(&gm,&gr);
initgraph(&gm,&gr,"C:/TC/BGI");
rectangle(x,y,az,w);
printf("*****Scaling*****\n\n")
;
printf("Enter the value of scaling f
actor:\n");
scanf("%f%f",&sx,&sy);
x1=x*sx;
y1=y*sy;
az1=az*sx;
w1=w*sy;
rectangle(x1,y1,az1,w1);
break;
case 4:
detectgraph(&gm,&gr);
initgraph(&gm,&gr,"C:/TC/BGI");
rectangle(x,y,az,w);
printf("*****Reflection*****\n
");

printf("1.About x-axis\n2.About y-
axis\n3.About both axis\nEnter your

choice:\n");

scanf("%d",&ch1);
switch(ch1)
{
case 1:
printf("Enter the fixed point\n");
scanf("%d%d",&xa,&ya);
theta=(float)(90*(3.14/180));
for(i=0;i<4;i++)
{
a1[i]=(xa+((a[i]-xa)*cos(theta)-(-b[
i]-ya)*sin(theta)));
b1[i]=(ya+((a[i]-xa)*sin(theta)+(-b[
i]-ya)*cos(theta)));
}
for(i=0;i<4;i++)
{
if(i!=3)
line(a1[i],b1[i],a1[i+1],b1[i+1]);
else
line(a1[i],b1[i],a1[0],b1[0]);
}
break;
case 2:
```



**Vidya Vikas Education Trust's**  
**Universal College of Engineering, Kaman Road, Vasai-401208**  
**Accredited B+ Grade NAAC**

```
printf("Enter the fixed point\n");
scanf("%d%d",&xa,&ya);
theta=(float)(270*(3.14/180));
for(i=0;i<4;i++)
{
    a1[i]=(xa+((-a[i]-xa)*cos(theta)-(b
[i]-ya)*sin(theta)));
    b1[i]=(ya+((-a[i]-xa)*sin(theta)+(b
[i]-ya)*cos(theta)));
}
for(i=0;i<4;i++)
{
    if(i!=3)
        line(a1[i],b1[i],a1[i+1],b1[i+1]);
    else
        line(a1[i],b1[i],a1[0],b1[0]);
}
break;
case 3:
printf("Enter the fixed point\n");
scanf("%d%d",&xa,&ya);
theta=(float)(180*(3.14/180));
for(i=0;i<4;i++)
{
```

```
3 of 6
a1[i]=(xa+((-a[i]-xa)*cos(theta)-(-
b[i]-ya)*sin(theta)));
b1[i]=(ya+((-a[i]-xa)*sin(theta)+(-
b[i]-ya)*cos(theta)));
}
for(i=0;i<4;i++)
{
    if(i!=3)
        line(a1[i],b1[i],a1[i+1],b1[i+1]);
    else
        line(a1[i],b1[i],a1[0],b1[0]);
}
break;
}
break;
case 5:
detectgraph(&gm,&gr);
initgraph(&gm,&gr,"C:/TC/BGI");
rectangle(x,y,az,w);
printf("*****Shearing*****\n\n");

printf("1.x-direction shear\n2.y-
direction shear\nEnter your choice:\n");
```

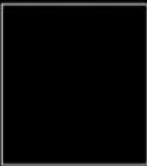
```
n");
scanf("%d",&ch2);
switch(ch2)
{
    case 1:
        printf("Enter the value of shear:\n"
);
        scanf("%f",&x1s);
        x1=x+(y*x1s);
        y1=y;
        az1=az+(w*x1s);
        w1=w;
        rectangle(x1,y1,az1,w1);
        break;
    case 2:
        printf("Enter the value of shear:\n"
);
        scanf("%f",&y1s);
        x1=x;
        y1=y+(x*y1s);
        az1=az;
        w1=w+(az*y1s);
        rectangle(x1,y1,az1,w1);
```



OUTPUT:-

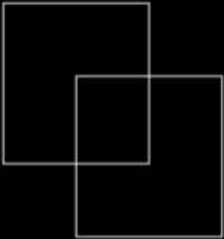
#### Drawing a rectangle

```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...
Enter the upper left corner of the rectangle:
100 90
Enter the lower right corner of the rectangle:
500 200
*****2D Transformations*****
1.Translation
2.Rotation
3.Scaling
4.Reflection
5.Shearing
6.Exit
Enter your choice:
```



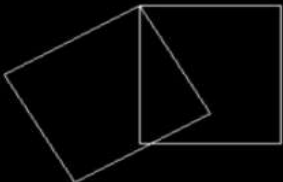
#### Translation

```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...
*****Translation*****
Enter the value of shift vector:
50 50
*****2D Transformations*****
1.Translation
2.Rotation
3.Scaling
4.Reflection
5.Shearing
6.Exit
Enter your choice:
```



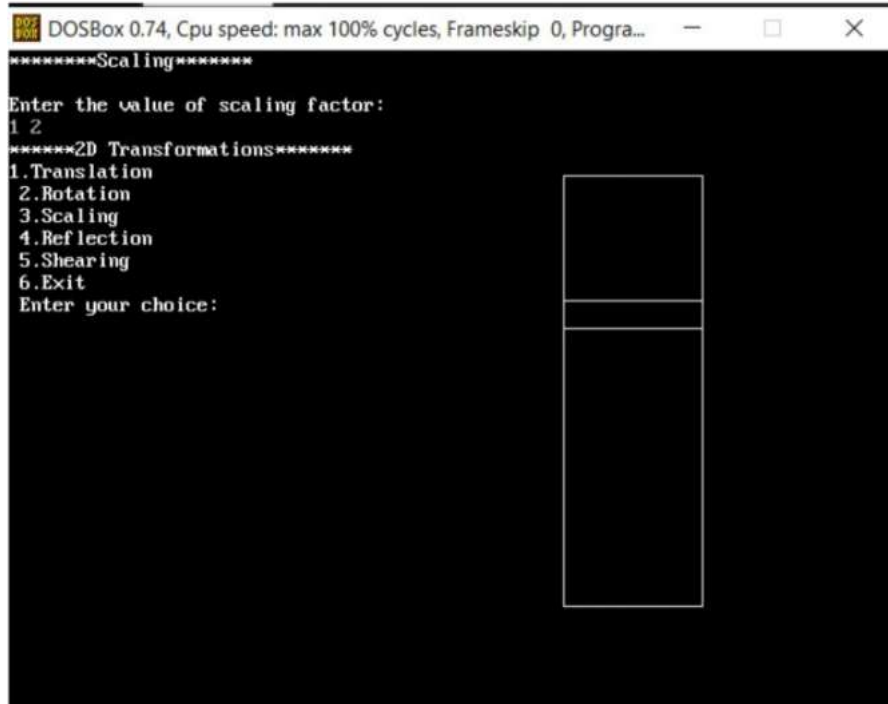
#### Rotation

```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...
*****Rotation*****
Enter the value of fixed point and angle of rotation:Enter the value of fixed po
int and angle of rotation:
100 90 60
*****2D Transformations*****
1.Translation
2.Rotation
3.Scaling
4.Reflection
5.Shearing
6.Exit
Enter your choice:
```

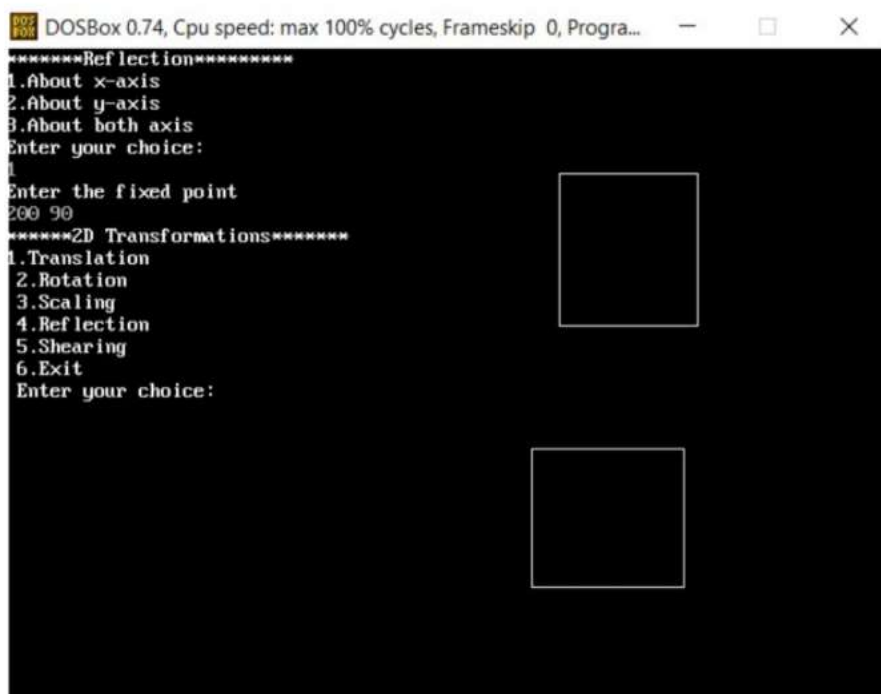




### Scaling



### Reflection





### Shearing

```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...
*****Shearing*****
1.x-direction shear
2.y-direction shear
Enter your choice:
1
Enter the value of shear:
0.5
*****2D Transformations*****
1.Translation
2.Rotation
3.Exit
4.Reflection
5.Scaling
6.Rotation
7.Translation
Enter your choice:
```

**CONCLUSION:-** Thus Program for 2D Transformation operation on the polygon has been studied and implemented.



## **EXPERIMENT NO. 07**

### ***To Implementation of Liang-Barsky line clipping Algorithm***

#### **AIM:-**

To study and write a program for clipping of line using Liang-Barsky algorithm.

#### **THEORY:-**

The **Liang-Barsky algorithm** is a line clipping algorithm. This algorithm is more efficient than Cohen-Sutherland line clipping algorithm and can be extended to 3-Dimensional clipping. This algorithm is considered to be the faster parametric line-clipping algorithm. The following concepts are used in this clipping:

1. The parametric equation of the line.
2. The inequalities describing the range of the clipping window which is used to determine the intersections between the line and the clip window.

The parametric equation of a line can be given by,

$$X = x_1 + t(x_2 - x_1)$$

$$Y = y_1 + t(y_2 - y_1)$$

Where,  $t$  is between 0 and 1.

Then, writing the point-clipping conditions in the parametric form:

$$x_{w_{min}} \leq x_1 + t(x_2 - x_1) \leq x_{w_{max}}$$

$$y_{w_{min}} \leq y_1 + t(y_2 - y_1) \leq y_{w_{max}}$$

The above 4 inequalities can be expressed as,

$$tp_k \leq q_k$$

Where  $k = 1, 2, 3, 4$  (correspond to the left, right, bottom, and top boundaries, respectively).

The  $p$  and  $q$  are defined as,

$$p_1 = -(x_2 - x_1), \quad q_1 = x_1 - x_{w_{min}} \text{ (Left Boundary)}$$

$$p_2 = (x_2 - x_1), \quad q_2 = x_{w_{max}} - x_1 \text{ (Right Boundary)}$$

$$p_3 = -(y_2 - y_1), \quad q_3 = y_1 - y_{w_{min}} \text{ (Bottom Boundary)}$$

$$p_4 = (y_2 - y_1), \quad q_4 = y_{w_{max}} - y_1 \text{ (Top Boundary)}$$

When the line is parallel to a view window boundary, the  $p$  value for that boundary is zero.

When  $p_k < 0$ , as  $t$  increase line goes from the outside to inside (entering).

When  $p_k > 0$ , line goes from inside to outside (exiting).

When  $p_k = 0$  and  $q_k < 0$  then line is trivially invisible because it is outside view window.

When  $p_k = 0$  and  $q_k > 0$  then the line is inside the corresponding window boundary.

Using the following conditions, the position of line can be determined:

CONDITION	POSITION OF LINE
	parallel to the clipping
$p_k = 0$	boundaries



**Vidya Vikas Education Trust's**  
**Universal College of Engineering, Kaman Road, Vasai-401208**  
**Accredited B+ Grade NAAC**

CONDITION	POSITION OF LINE
$p_k = 0$ and $q_k < 0$	completely outside the boundary
$p_k = 0$ and $q_k \geq 0$	inside the parallel clipping boundary
$p_k < 0$	line proceeds from outside to inside
$p_k > 0$	line proceeds from inside to outside

Parameters  $t_1$  and  $t_2$  can be calculated that define the part of line that lies within the clip rectangle.

When,

1.  $p_k < 0$ , maximum(0,  $q_k/p_k$ ) is taken.
2.  $p_k > 0$ , minimum(1,  $q_k/p_k$ ) is taken.

If  $t_1 > t_2$ , the line is completely outside the clip window and it can be rejected. Otherwise, the endpoints of the clipped line are calculated from the two values of parameter  $t$ .

#### Algorithm

1. Set  $t_{\min}=0$ ,  $t_{\max}=1$ .
2. Calculate the values of  $t$  ( $t(\text{left})$ ,  $t(\text{right})$ ,  $t(\text{top})$ ,  $t(\text{bottom})$ ),
  - (i) If  $t < t_{\min}$  ignore that and move to the next edge.
  - (ii) else separate the  $t$  values as entering or exiting values using the inner product.
  - (iii) If  $t$  is entering value, set  $t_{\min} = t$ ; if  $t$  is existing value, set  $t_{\max} = t$ .
3. If  $t_{\min} < t_{\max}$ , draw a line from  $(x_1 + t_{\min}(x_2-x_1), y_1 + t_{\min}(y_2-y_1))$  to  $(x_1 + t_{\max}(x_2-x_1), y_1 + t_{\max}(y_2-y_1))$
4. If the line crosses over the window,  $(x_1 + t_{\min}(x_2-x_1), y_1 + t_{\min}(y_2-y_1))$  and  $(x_1 + t_{\max}(x_2-x_1), y_1 + t_{\max}(y_2-y_1))$  are the intersection point of line and edge.

This algorithm is presented in the following code. Line intersection parameters are initialised to the values  $t_1 = 0$  and  $t_2 = 1$ .

#### CODE:-

```
//Vivek Shivakumar Hotti: SE-COMPS-Roll-32; 24-10-2020 (Saturday)
#include<conio.h>
#include<graphics.h>
int main()
{
int gd=DETECT,gm,x1,y1,x2,y2,xmax,
xmin,ymax,ymin,xx1,yy1,xx2,yy2,dx,d
y,i,p
[4],q[4];
float t1,t2,t[4];
initgraph(&gd,&gm,"C:/TC/BGI");
```



**Vidya Vikas Education Trust's**  
**Universal College of Engineering, Kaman Road, Vasai-401208**  
**Accredited B+ Grade NAAC**

```
printf("Enter the lower co-ordinates of window:");

scanf("%d%d",&xmin,&ymin);

printf("Enter the upper co-ordinates of window:");

scanf("%d%d",&xmax,&ymax);
setcolor(RED);
rectangle(xmin,ymin,xmax,ymax);
printf("Enter x1:");
scanf("%d",&x1);
printf("Enter y1:");
scanf("%d",&y1);
printf("Enter x2:");
scanf("%d",&x2);
printf("Enter y2:");
scanf("%d",&y2);
line(x1,y1,x2,y2);
dx=x2-x1;
dy=y2-y1;
p[0]=-dx;
p[1]=dx;
p[2]=-dy;
p[3]=dy;
```

```
q[0]=x1-xmin;
q[1]=xmax-x1;
q[2]=y1-ymin;
q[3]=ymax-y1;
for(i=0;i < 4;i++){
if(p[i]!=0){
t[i]=(float)q[i]/p[i];
}
if (t[0] > t[2]){
t1=t[0]; }
else{
t1=t[2]; }
if (t[1] < t[3]){
t2=t[1];
} else{
t2=t[3]; }
if (t1 < t2){
if(p[i]==0 && q[i] < 0)
printf("line completely outside the window.");
else
if(p[i]==0 && q[i] >= 0)
```

```
printf("line completely inside the window.");
xx1=x1+t1*dx;
xx2=x1+t2*dx;
yy1=y1+t1*dy;
yy2=y1+t2*dy;
printf("Line After Clipping As Shown Below:");
setcolor(WHITE);
line(xx1,yy1,xx2,yy2);
}

else{
printf("line lies out of the window");
}
getch();
}
```





Vidya Vikas Education Trust's  
Universal College of Engineering, Kaman Road, Vasai-401208  
Accredited B+ Grade NAAC

**OUTPUT:-**

```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...
Enter the lower co-ordinates of window:350 350
Enter the upper co-ordinates of window:400 400
Enter x1:250
Enter y1:350
Enter x2:400
Enter y2:410
Line After Clipping As Shown Below:
```

**CONCLUSION:-** Thus program for Liang Barsky Line Clipping Algorithm has been studied and implemented.



## **EXPERIMENT NO. 08**

### ***To Implementation of Bezier Curve Algorithm***

#### **AIM:-**

To study and implement Bezier curve.

#### **THEORY:-**

It is another approach for the construction of curve determined by a defining polygon.

Bezier curve has number of properties for curves and surface design.

Mid point approach: In this approach, Beizer curve can be constructed simply by taking midpoints. In this approach, midpoint of line connecting 4 control points(A,B,C,D) are determined by line segment and their midpoint ABC and BCD are determined.

#### **Algorithm**

Step 1. Let 4 control points say A(xA, yA), B(xB, yB), C(xC, yC), D(Dx, Dy).

Step 2. Divide the curve represented by points ABCD into 2 sections

$$x_{AB} = (x_A + x_B)/2 \quad x_{ABC} = (x_{AB} + x_{BC})/2$$

$$y_{AB} = (y_A + y_B)/2 \quad y_{ABC} = (y_{AB} + y_{BC})/2$$

$$x_{BC} = (x_B + x_C)/2 \quad x_{BCD} = (x_{BC} + x_{CD})/2$$

$$y_{BC} = (y_B + y_C)/2 \quad y_{BCD} = (y_{BC} + y_{CD})/2$$

$$x_{CD} = (x_C + x_D)/2 \quad x_{ABCD} = (x_{ABC} + x_{BCD})/2$$

$$y_{CD} = (y_C + y_D)/2 \quad y_{ABCD} = (y_{ABC} + y_{BCD})/2$$

Step 3. Repeat step 2 till we have section for each section.

Step 4. Repeat step 3 till we have section so short and then replace small section by straight line.



**CODE:-**

```
#include<graphics.h>
#include<stdio.h>
#include<math.h>
#include<conio.h>
int main()
{
int x[4], y[4],i,n;
double put_x, put_y, t;
int gd=DETECT, gm;
initgraph(&gd,&gm, "C:/TC/BGI");
printf("\n Bezier Curve Program");

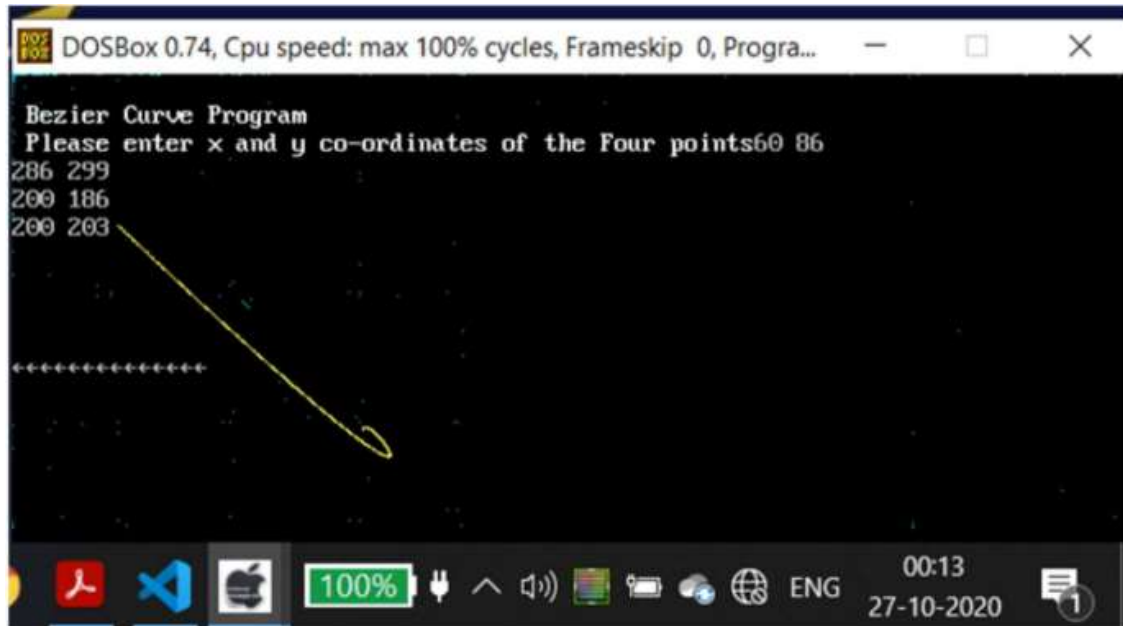
printf("\n Please enter x and y co-ordinates of the Four points");

for(i=0;i<n;i++)
{
scanf("%d %d", &x[i], &y[i]);
putpixel(x[i],y[i],3);
}
for(t=0.0;t<=1.0;t=t+0.001)
{
put_x=pow(1-
t,3)*x[0] + 3*t*pow(1-
t,2)*x[1]+3*t*t*(1-
t)*x[2]+pow(t,3)*x[3];
put_y=pow(1-
t,3)*y[0] + 3*t*pow(1-
t,2)*y[1]+3*t*t*(1-
t)*y[2]+pow(t,3)*y[3];
putpixel(put_x, put_y, YELLOW);
}

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



**OUTPUT:-**



**CONCLUSION:-** hence we have studied and implemented a program to generate a Bezier Curve.



## **EXPERIMENT NO. 9**

### ***To Implementation of Animation***

#### **AIM:-**

To study and implement Animation

#### **THEORY:-**

Animation is a method in which figures are manipulated to appear as moving images. In traditional animation, images are drawn or painted by hand on transparent celluloid sheets to be photographed and exhibited on film. Today, most animations are made with computer-generated imagery (CGI). Computer animation can be very detailed 3D animation, while 2D computer animation can be used for stylistic reasons, low bandwidth or faster real-time renderings. Other common animation methods apply a stop motion technique to two and three-dimensional objects like paper cutouts, puppets or clay figures.

#### **CODE:-**

```
#include<graphics.h>
#include<stdio.h>
#include<math.h>
#include<conio.h>
#include<dos.h>
void main()
{
    int gd=DETECT,gm,i,a;
    initgraph(&gd,&gm, "C:/TC/BGI");
    for(i=0;i<600;i++)
    {
        line(50+i,405,100+i,405);
        line(75+i,375,125+i,375);
        line(50+i,405,75+i,375);
        line(100+i,405,125+i,375);
        line(150+i,405,100+i,345);
        line(75+i,375,75+i,370);
        line(70+i,370,80+i,370);

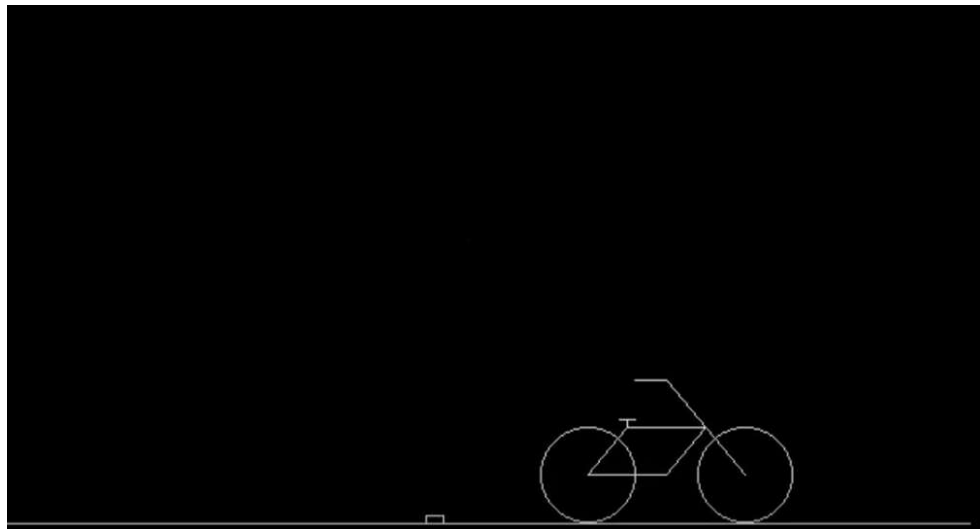
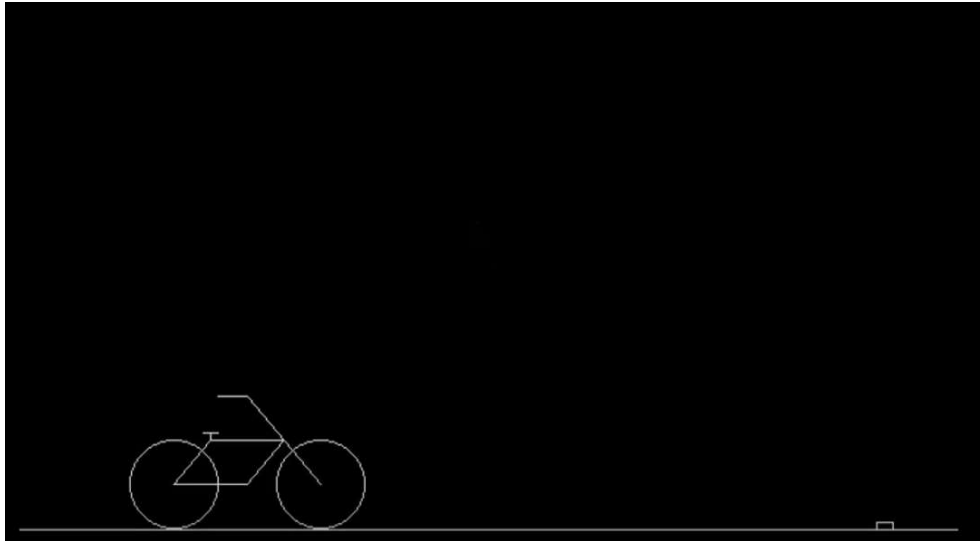
        line(80+i,345,100+i,345);
        //wheel
        circle(150+i,405,30);
        circle(50+i,405,30);
        //road
        line(0,436,getmaxx(),436);
        //stone
```



**Vidya Vikas Education Trust's**  
**Universal College of Engineering, Kaman Road, Vasai-401208**  
**Accredited B+ Grade NAAC**

```
rectangle(getmaxx()-i,436,650-  
i,431);  
delay(10); cleardevice();  
} getch(); closegraph(); }
```

**OUTPUT:-**



**CONCLUSION:- Thus we have implemented a program to animate a cycle.**



## **EXPERIMENT NO. 10**

# **To Implement MINIPROJECT**

### **TITLE of PROJECT:- SOLAR SYSTEM IN C AND BLENDER**

#### **THEORY:-**

Blender is a free and open-source 3D creation suite. With Blender, you can create 3D visualizations such as still images, 3D animations, VFX shots, and video editing. It is well suited to individuals and small studios who benefit from its unified pipeline and responsive development process. Being a cross-platform application, Blender runs on Linux, macOS, as well as Windows systems. It also has relatively small memory and drive requirements compared to other 3D creation suites. Its interface uses OpenGL to provide a consistent experience across all supported hardware and platforms.

#### **OBJECTS**

The geometry of a scene is constructed from one or more objects. These objects can range from lights to illuminate your scene, basic 2D and 3D shapes to fill it with models, armatures to animate those models, to cameras to take pictures or make video of it all.

Each Blender object type (mesh, light, curve, camera, etc.) is composed from two parts: an Object and Object Data (sometimes abbreviated to "ObData"):

Object: Holds information about the position, rotation and size of a particular element.

Object Data: Holds everything else. For example:

Meshes: Store geometry, material list, vertex groups, etc.

Cameras: Store focal length, depth of field, sensor size, etc.

Each object has a link to its associated object-data, and a single object-data may be shared by many objects

#### **Animation**

Animation is making an object move or change shape over time. Objects can be animated in many ways:

Moving as a whole object

Changing their position, orientation or size in time;

Deforming them

Animating their vertices or control points;

Inherited animation

Causing the object to move based on the movement of another object (e.g. its parent, hook, armature, etc.).

Animation is typically achieved with the use of KEYFRAMES.

#### **Using Texture Nodes**

To use texture nodes with the current texture, open the Texture Node editor.

A new texture can be created by either clicking the *New* button in the Texture Node editor, or the *New* button in the Texture panel. Once a texture is selected, it can be toggled to a function as a regular texture or a node texture by clicking the *Use Nodes* option in the Texture Node editor.

The default node setup will appear: a red-and-white checkerboard node connected to an *Output* named "Default". For texture nodes, multiple Outputs can exist in the node setup. Compare to other types of node contexts, which are limited to one active Output node.



**Vidya Vikas Education Trust's**  
**Universal College of Engineering, Kaman Road, Vasai-401208**  
**Accredited B+ Grade NAAC**

**CODE:-**

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

void planetMotion(int xrad, int yrad, int midx, int midy, int x[60],
int y[60]) {
    int i, j = 0;

    for (i = 360; i > 0; i = i - 6) {
        x[j] = midx - (xrad * cos((i * 3.14) / 180));
        y[j++] = midy - (yrad * sin((i * 3.14) / 180));
    }
    return;
}

int main() {

    int gdriver = DETECT, gmode, err;
    int i = 0, midx, midy;
    int xrad[9], yrad[9], x[9][60], y[9][60];
    int pos[9], planet[9], tmp;

    initgraph(&gdriver, &gmode, "C:/TC/BGI");
    err = graphresult();
    if (err != grOk) {
        /* error occurred */
        printf("Graphics Error: %s",
            grapherrormsg(err));
        return 0;
    }

    midx = getmaxx() / 2;
    midy = getmaxy() / 2;

    planet[0] = 7;
    for (i = 1; i < 9; i++) {
        planet[i] = planet[i - 1] + 1;
    }

    for (i = 0; i < 9; i++) {
        pos[i] = i * 6;
    }

    xrad[0] = 60, yrad[0] = 30;
    for (i = 1; i < 9; i++) {
        xrad[i] = xrad[i - 1] + 30;
        yrad[i] = yrad[i - 1] + 15;
    }

    for (i = 0; i < 9; i++) {
```





**Vidya Vikas Education Trust's**  
**Universal College of Engineering, Kaman Road, Vasai-401208**  
**Accredited B+ Grade NAAC**

```
planetMotion(xrad[i], yrad[i], midx, midy, x[i], y[i]);
}
while (!kbhit()) {

    setcolor(WHITE);
    for (i = 0; i < 9; i++) {
        ellipse(midx, midy, 0, 360, xrad[i], yrad[i]);
    }

    setcolor(YELLOW);
    setfillstyle(SOLID_FILL, YELLOW);
    circle(midx, midy, 20);
    floodfill(midx, midy, YELLOW);

    setcolor(CYAN);
    setfillstyle(SOLID_FILL, CYAN);
    pieslice(x[0][pos[0]], y[0][pos[0]], 0, 360, planet[0]);

    setcolor(GREEN);
    setfillstyle(SOLID_FILL, GREEN);
    pieslice(x[1][pos[1]], y[1][pos[1]], 0, 360, planet[1]);

    setcolor(BLUE);
    setfillstyle(SOLID_FILL, BLUE);
    pieslice(x[2][pos[2]], y[2][pos[2]], 0, 360, planet[2]);

    setcolor(RED);
    setfillstyle(SOLID_FILL, RED);
    pieslice(x[3][pos[3]], y[3][pos[3]], 0, 360, planet[3]);

    setcolor(BROWN);
    setfillstyle(SOLID_FILL, BROWN);
    pieslice(x[4][pos[4]], y[4][pos[4]], 0, 360, planet[4]);

    setcolor(LIGHTGRAY);
    setfillstyle(SOLID_FILL, LIGHTGRAY);
    pieslice(x[5][pos[5]], y[5][pos[5]], 0, 360, planet[5]);

    setcolor(BROWN);
    setfillstyle(SOLID_FILL, BROWN);
    pieslice(x[6][pos[6]], y[6][pos[6]], 0, 360, planet[6]);

    setcolor(LIGHTBLUE);
    setfillstyle(SOLID_FILL, LIGHTBLUE);
    pieslice(x[7][pos[7]], y[7][pos[7]], 0, 360, planet[7]);

    setcolor(LIGHTRED);
    setfillstyle(SOLID_FILL, LIGHTRED);
```



**Vidya Vikas Education Trust's**  
**Universal College of Engineering, Kaman Road, Vasai-401208**  
**Accredited B+ Grade NAAC**

```
pieslice(x[8][pos[8]], y[8][pos[8]], 0, 360, planet[8]);

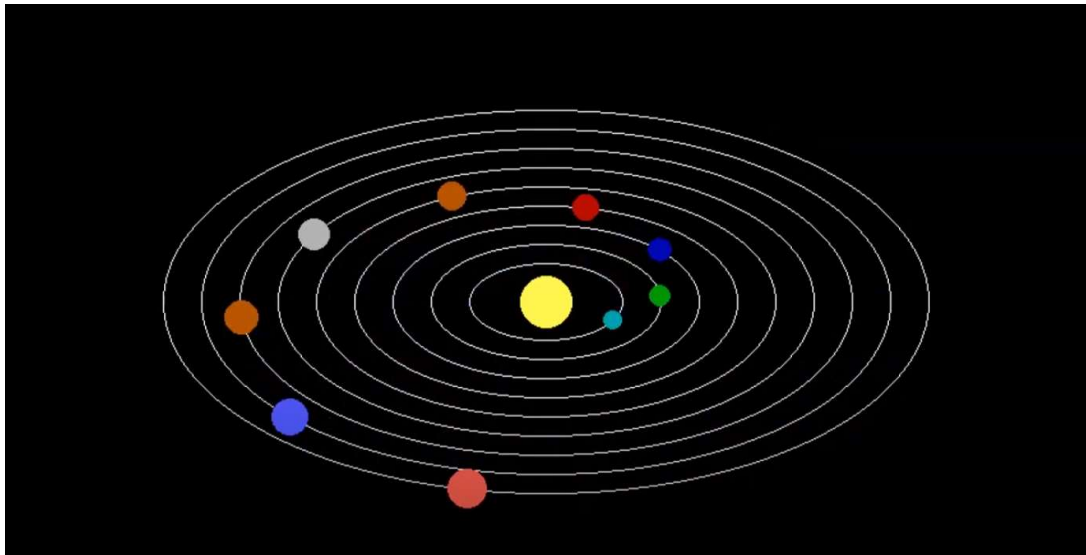
for (i = 0; i < 9; i++) {
    if (pos[i] <= 0) {
        pos[i] = 59;
    } else {
        pos[i] = pos[i] - 1;
    }
}

delay(100);

cleardevice();
}

closegraph();
return 0;
}
```

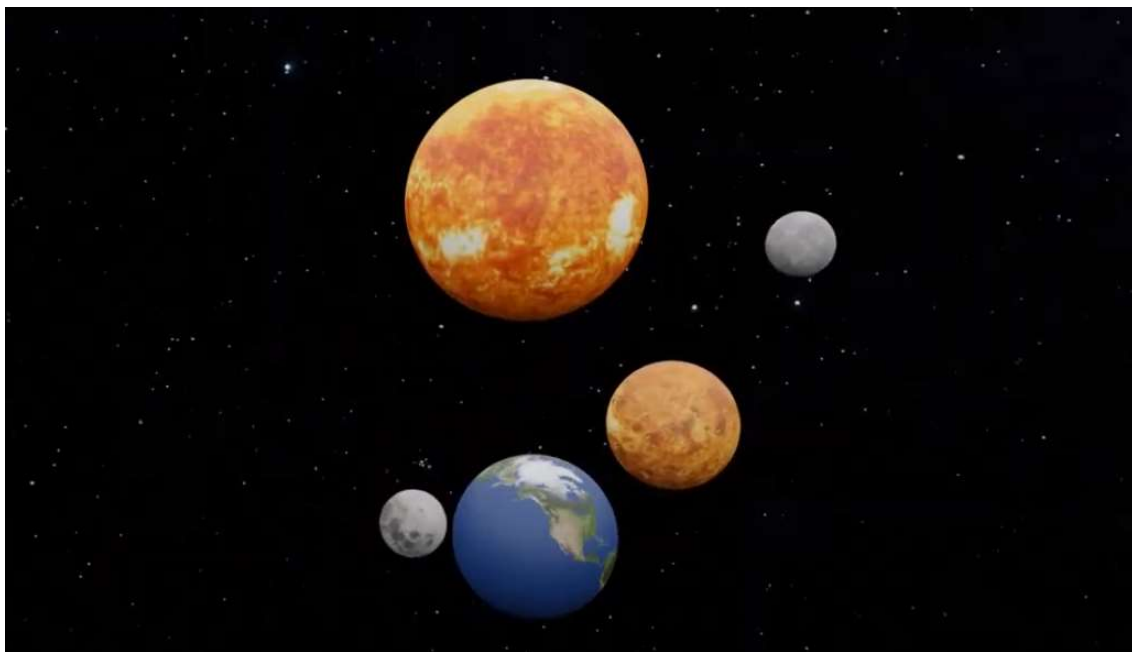
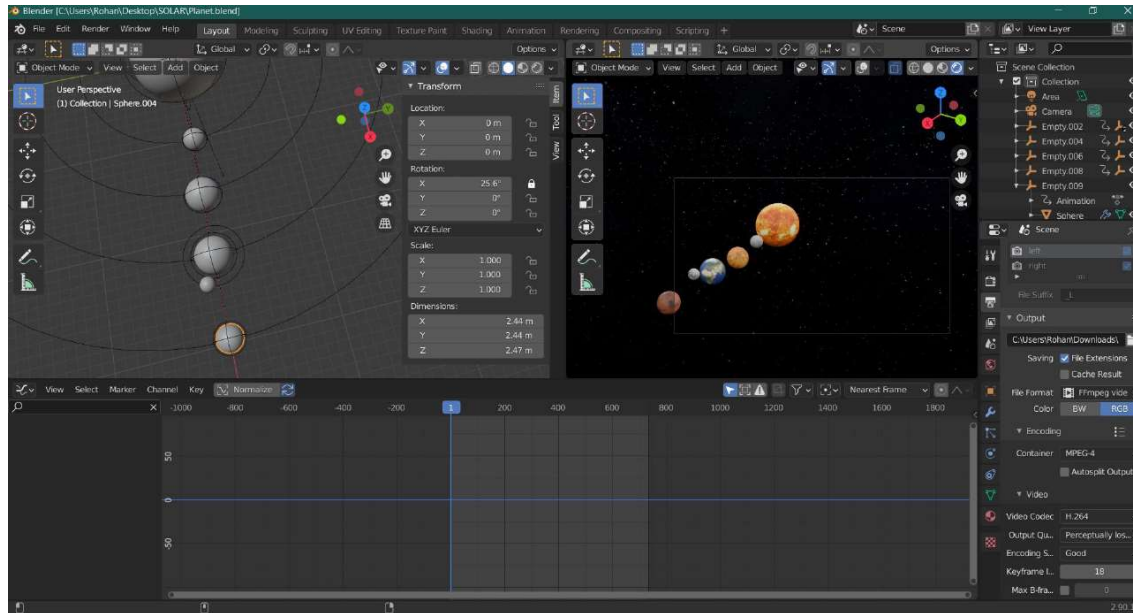
**OUTPUT:-**





**Vidya Vikas Education Trust's**  
**Universal College of Engineering, Kaman Road, Vasai-401208**  
**Accredited B+ Grade NAAC**

### **SOLAR SYSTEM USING BLENDER:**



**CONCLUSION:-** Hence me and my team (Rohan and Yash) constructed a solar system model in C and Blender.



**Vidya Vikas Education Trust's**  
**Universal College of Engineering, Kaman Road, Vasai-401208**  
**Accredited B+ Grade NAAC**

