JAIPUR ENGINEERING COLLEGE AND RESEARCH CENTRE

JAIPUR ENGINEERING COLLEGE AND RESEARCH CENTRE

JECRC Campus, Shri Ram Ki Nangal, Via-Vatika, Jaipur

LAB MANUAL

Lab Name : COMPUTER GRAPHICS & MULTIMEDIA LAB

Lab Code : 5CS4-21

Branch : Computer Science & Engineering

Year : 3rd Year

JAIPUR ENGINEERING COLLEGE AND RESEARCH CENTRE

Department of Computer Science & Engineering
Jaipur Engineering College and Research Centre, Jaipur
(Affiliated to RTU, Kota)

INDEX

Experiment 12	Write a C program to implement animations using transformations like rotation, scaling and translation. The simple animations are given below: • Moving Helicopter • Moving Fan
Experiment 13	Write a C program for creating two dimensional shapes of house, car, fish, man using lines, circles etc.

Content Beyond Syllabus	
Experiment 14	Implementation of Polygon fill algorithm
Experiment 15	Implementation of Bezier Curve

Vision of the Institute

To become a renowned centre of outcome based learning, and work towards academic, professional, cultural and social enrichment of the lives of individuals and communities.

Mission of the Institute

M1: Focus on evaluation of learning outcomes and motivate students to inculcate research aptitude by project based learning.

M2: Identify, based on informed perception of Indian, regional and global needs, areas of focus and provide platform to gain knowledge and solutions.

M3: Offer opportunities for interaction between academia and industry.

M4: Develop human potential to its fullest extent so that intellectually capable and imaginatively gifted leaders can emerge in a range of professions.

Vision of the Department

To become renowned Centre of excellence in computer science and engineering and make competent engineers & professionals with high ethical values prepared for lifelong learning.

Mission of the Department

M1: To impart outcome based education for emerging technologies in the field of computer science and engineering.

M2: To provide opportunities for interaction between academia and industry.

M3: To provide platform for lifelong learning by accepting the change in technologies.

M4: To develop aptitude of fulfilling social responsibilities.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

1. To provide students with the fundamentals of Engineering Sciences with more emphasis in **Computer Science &Engineering** by way of analysing and exploiting engineering challenges.
2. To train students with good scientific and engineering knowledge so as to comprehend, analyse, design, and create novel products and solutions for the real life problems.
3. To inculcate professional and ethical attitude, effective communication skills, teamwork skills, multidisciplinary approach, entrepreneurial thinking and an ability to relate engineering issues with social issues.
4. To provide students with an academic environment aware of excellence, leadership, written ethical codes and guidelines, and the self-motivated life-long learning needed for a successful professional career.
5. To prepare students to excel in Industry and Higher education by Educating Students along with High moral values and Knowledge

PROGRAM OUTCOMES (POs)

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
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.
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 the public health and safety, and the cultural, societal, and environmental considerations.
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.

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.

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.

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.

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

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

10.**Communication**: Communicate effectively on complex engineering activities with the engineering community and with 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.

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.

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 (PSOs)

PSO1. Ability to interpret and analyse network specific and cyber security issues, automation in real word environment.

PSO2. Ability to Design and Develop Mobile and Web-based applications under realistic constraints.

RTU Syllabus with List of Experiments

5CS4-21: COMPUTER GRAPHICS & MULTIMEDIA LAB

Class: VI Sem. B.Tech.	Evaluation
Branch: Computer Engg. Schedule per Week Practical Hrs.: 2	Examination Time =Three (3) Hours Maximum Marks = 50 [Sessional/Mid-term (30) & Endterm (20)]

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

Perform the following experiment using Virtual Lab http://www.vlab.co.in/	
14	Implementation of DDA
15	Implementation of Bresenham's Algorithm
16	Imlementation of Bresenham's circle algorithm
17	Implementation of 2D transformation

Course Outcomes

CO 1: Apply basic concepts and different types of graphics drawing algorithm.

CO 2: Analyze different transformations and animation.

Mapping of Experiments with Cos & BT Level

S. No.	Contents	COs	BT*
1	Write a C program to implementation of line, circle and ellipse attributes.	1	3
2	Write a C program to plot a point (pixel) on the screen with different colors by using graphics function.	1	3
3	Write a C program to implement DDA algorithm for drawing a line segment between two given end points A (x_1, y_1) and B(x_2, y_2).	1	3
4	Write a C program to implement Bresenham's line drawing algorithm for drawing a line segment between two given end points A (x_1, y_1) and B(x_2, y_2).	1	3
5	Write a C program to implement midpoint circle generation algorithm of given center (x, y) and radius r.	1	3
6	Write a C program to implement the Ellipse Generation Algorithm for drawing an ellipse of given center(x, y) and radius r_x and r_y .	1	3
7	Write a C program to implement the basic 2D transformations such as translation, scaling, rotation, shearing and reflection for a given 2D object.	2	3
8	Write a C program to implement flood fill algorithm.	1	3
9	Write a C program to implement the boundary fill algorithm for filling a rectangle with given color.	1	3
10	Write a C program to implement Cohen Sutherland line clipping and windowing algorithm.	1	3
11	Write a C program to implement the basic transformations such as translation, scaling, rotation for a given 3D object.	2	3

12	Write a C program to create animations using transformations like rotation, scaling and translation. The simple animations are given below: • Moving Helicopter • Moving Fan	2	6
13	Write a C program for creating two dimensional shapes of house, car, fish, man using lines, circles etc.	2	6
Perform the following experiments using Virtual Lab http://www.vlab.co.in/			
14	Implementation of DDA	1	3
15	Implementation of Bresenham's Algorithm	1	3
16	Implementation of Bresenham's circle algorithm	1	3
17	Implementation of 2D transformation	2	3
Content Beyond Syllabus			
18	Implementation of Bezier Curve	2	3
19	Implementation of Polygon fill algorithm	1	3

* BT - Bloom's Taxonomy

Mapping of Course Outcomes & POs/PSOs

	Engineering Knowledge	Problem analysis	Design/Development of Solution	Conduct Invest. of complex problems	Modern Tool Usage	The engineer and society	Environment and Sustainability	Ethics	Individual and Team Work	Communication	Project Management and Finance	Life-long Learning	network specific, cyber security and automation	Mobile and Web-based applications
	PO 1	**PO 2**	**PO 3**	**PO 4**	**PO 5**	**PO 6**	**PO 7**	**PO 8**	**PO 9**	**PO 10**	**PO 11**	**PO 12**	**PSO 1**	**PSO 2**
CO-1	3	3	3	3	2	2	1	1	1	1	2	3	2	2
CO-2	3	3	3	2	2	1	1	1	1	1	1	3	2	2

INTRODUCTION ABOUT LABORATORY & APPLICATIONS

This laboratory is combination of algorithms, numerical methods, representations of the shape and appearance of real-world objects, and methods for their display and manipulation. These algorithms must take into account the conflicting requirements for real-time interactions and usability by artists and animators, and are built upon a solid mathematical background.

Experiments performed in this course will provide students a better understanding about

- Apply basic concepts and different types of graphics drawing algorithm
- Analyse different transformations and animation

Students will design, implement, and use interactive graphical applications.

LAB REQUIREMENTS

- Software requirements: Turbo C++ or Turbo C compiler, MAYA
- Operating System: Windows XP (SP2)
- Hardware requirements: Windows and Linux: Intel 64/32 or AMD Athlon 64/32, or AMD Opteron processor 2 GB RAM , 2 GB hard disk space

INSTRUCTIONS SHEET

We need your full support and cooperation for smooth functioning of the lab.

DO's

- Please switch off the Mobile/Cell phone before entering Lab.
- Enter the Lab with complete source code and data.
- Check whether all peripheral are available at your desktop before proceeding for program.
- Intimate the lab In charge whenever you are incompatible in using the system or in case software get corrupted/ infected by virus.
- Arrange all the peripheral and seats before leaving the lab.
- Properly shutdown the system before leaving the lab.
- Keep the bag outside in the racks.
- Enter the lab on time and leave at proper time.
- Maintain the decorum of the lab.
- Utilize lab hours in the corresponding experiment.
- Get your Cd / Pen drive checked by lab In charge before using it in the lab.

DON'Ts

- No one is allowed to bring storage devices like Pan Drive /Floppy etc. in the lab.
- Don't mishandle the system.
- Don't leave the system on standing for long
- Don't bring any external material in the lab.
- Don't make noise in the lab.
- Don't bring the mobile in the lab. If extremely necessary then keep ringers off.
- Don't enter in the lab without permission of lab Incharge.
- Don't litter in the lab.
- Don't delete or make any modification in system files.
- Don't carry any lab equipment outside the lab.

1.1.1.1.1.1 BEFORE ENTERING IN THE LAB

1. All the students are supposed to prepare the theory regarding the next experiment.
2. Students are supposed to bring the practical file and the lab copy.
3. Previous practical should be written in the practical file.
4. Any student not following these instructions will be denied entry in the lab.

2 WHILE WORKING IN THE LAB

1. Adhere to experimental schedule as instructed by the lab in-charge.
2. Get the previously executed experiment signed by the instructor.
3. Get the output of the current experiment checked by the instructor in the lab copy.
4. Take responsibility of valuable accessories.
5. If anyone is caught carrying any equipment of the lab outside without permission, they will face strict disciplinary action.

Graphics Function

AIM:

To learn basic graphic's function

Description

3 BASIC GRAPHICS FUNCTION

1) INITGRAPH

- Initializes the graphics system.

3.1.1.1 Declaration

- Void far initgraph(int far *graphdriver)

3.1.1.2 Remarks

- To start the graphic system, you must first call initgraph.
- Initgraph initializes the graphic system by loading a graphics driver from disk (or validating a registered driver) then putting the system into graphics mode.
- Initgraph also resets all graphics settings (color, palette, current position, viewport, etc) to their defaults then resets graph.

2) GETPIXEL, PUTPIXEL

- Getpixel gets the color of a specified pixel.
- Putpixel places a pixel at a specified point.

3.1.1.3 Declaration

- Unsigned far getpixel(int x, int y)
- Void far putpixel(int x, int y, int color)

3.1.1.4 Remarks

- Getpixel gets the color of the pixel located at (x,y);
- Putpixel plots a point in the color defined at (x, y).

3.1.1.5 Return value

- Getpixel returns the color of the given pixel.
- Putpixel does not return.

3) CLOSE GRAPH

- Shuts down the graphic system.

3.1.1.6 Declaration

- Void far closegraph(void);

3.1.1.7 Remarks

- Close graph deallocates all memory allocated by the graphic system.
- It then restores the screen to the mode it was in before you called initgraph.

3.1.1.8 Return value

- None.

4) ARC, CIRCLE, PIESLICE

- arc draws a circular arc.
- Circle draws a circle
- Pieslice draws and fills a circular pieslice

3.1.1.9 Declaration

- Void far arc(int x, int y, int stangle, int endangle, int radius);
- Void far circle(int x, int y, int radius);
- Void far pieslice(int x, int y, int stangle, int endangle, int radius);

3.1.1.10 Remarks

- Arc draws a circular arc in the current drawing color
- Circle draws a circle in the current drawing color
- Pieslice draws a pieslice in the current drawing color, then fills it using the current fill pattern and fill color.

5) ELLIPSE, FILLELIPSE, SECTOR

- Ellipse draws an elliptical arc.
- Fillellipse draws and fills ellipse.
- Sector draws and fills an elliptical pie slice.

3.1.1.11 Declaration

- Void far ellipse(int x, int y, int stangle, int endangle, int xradius, int yradius)
- Void far fillellipse(int x, int y, int xradius, int yradius)
- Void farsectoe(int x, int y, int stangle, int endangle, int xradius, int yradius)

3.1.1.12 Remarks

- Ellipse draws an elliptical arc in the current drawing color.
- Fillellipse draws an elliptical arc in the current drawing color and than fills it with fill color and fill pattern.
- Sector draws an elliptical pie slice in the current drawing color and than fills it using the pattern and color defined by setfillstyle or setfillpattern.

6) FLOODFILL

- Flood-fills a bounded region.

3.1.1.13 Declaration

- Void far floodfill(int x, int y, int border)

3.1.1.14 Remarks

- Floodfills an enclosed area on bitmap device.
- The area bounded by the color border is flooded with the current fill pattern and fill color.
- (x,y) is a "seed point"
  - If the seed is within an enclosed area, the inside will be filled.
  - If the seed is outside the enclosed area, the exterior will be filled.
- Use fillpoly instead of floodfill wherever possible so you can maintain code compatibility with future versions.
- Floodfill doesnot work with the IBM-8514 driver.

3.1.1.15 Return value

- If an error occurs while flooding a region, graph result returns '1'.

7) GETCOLOR, SETCOLOR

- Getcolor returns the current drawing color.
- Setcolor returns the current drawing color.

3.1.1.16 Declaration

- Int far getcolor(void);
- Void far setcolor(int color)

3.1.1.17 Remarks

- Getcolor returns the current drawing color.
- Setcolor sets the current drawing color to color, which can range from 0 to getmaxcolor.
- To set a drawing color with setcolor , you can pass either the color number or the equivalent color name.

8) LINE,LINEREL,LINETO

- Line draws a line between two specified pints.
- Onerel draws a line relative distance from current position(CP).

- Linrto draws a line from the current position (CP) to(x,y).

3.1.1.18 Declaration

- Void far lineto(int x, int y)

3.1.1.19 Remarks

- Line draws a line from (x1, y1) to (x2, y2) using the current color, line style and thickness. It does not update the current position (CP).
- Linerel draws a line from the CP to a point that is relative distance (dx, dy) from the CP, then advances the CP by (dx, dy).
- Lineto draws a line from the CP to (x, y), then moves the CP to (x,y).

3.1.1.20 Return value

- None

9) RECTANGLE

- Draws a rectangle in graphics mode.

3.1.1.21 Declaration

- Void far rectangle(int left, int top, int right, int bottom)

3.1.1.22 Remarks

- It draws a rectangle in the current line style, thickness and drawing color.
- (left, top) is the upper left corner of the rectangle, and (right, bottom) is its lower right corner.

3.1.1.23 Return value

- None.

EXPERIMENT NO. 1

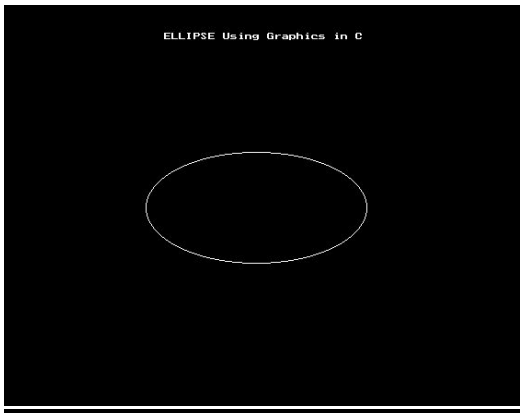
AIM:

To draw line, ellipse, heart, smiley face and circle using functio

```
/*C graphics program to draw a line.*/
#include <graphics.h>
#include <conio.h>
main()
{
  int gd = DETECT, gm;
  //init graphics
  initgraph(&gd, &gm, "C:/TURBOC3/BGI");
  /*
  if you are using turboc2 use below line to init graphics:
  initgraph(&gd, &gm, "C:/TC/BGI");
// C++ Implementation for drawing line
#include <graphics.h>
// driver code
int main()
{
   // gm is Graphics mode which is a computer display
   // mode that generates image using pixels.
   // DETECT is a macro defined in "graphics.h" header file
   int gd = DETECT, gm;

   // initgraph initializes the graphics system
   // by loading a graphics driver from disk
```

```
/* Ellipse*/
#include<stdio.h>
#include<graphics.h>
#include<conio.h>

int main(){
   int gd = DETECT,gm;
   int x ,y;
   initgraph(&gd, &gm, "X:\\TC\\BGI");
   /* Initialize center of ellipse with center of screen */
   x = getmaxx()/2;
   y = getmaxy()/2;

   outtextxy(x-100, 50, "ELLIPSE Using Graphics in C");
   /* Draw ellipse on screen */
   ellipse(x, y, 0, 360, 120, 60);

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

Output



/*Heart*/

```
#include<constream.h>
#include<graphics.h>
#include<dos.h>
void main()
{
int gd=DETECT,gm;
initgraph(&gd,&gm,"c:\\tc\\bgi");
cleardevice();

for(int l=1;l<=50;l++)
{
int b=1;
while(!kbhit())
{
for(int i=1;i<=20;i++)

settextstyle(3,0,5);
outtextxy(270,230,"Ankit");
arc(280,250,400,2000,50);
```

```
arc(355,250,700,500,50);
line(320,350,235,270);
line(320,350,400,270);
b++;

delay(8);
cleardevice();

if(b==70)
{
break;
}}}}
```

Output


```
/* Smiley Face*/
#include<graphics.h>
#include<stdio.h>
#include<conio.h>
void main()
{
```

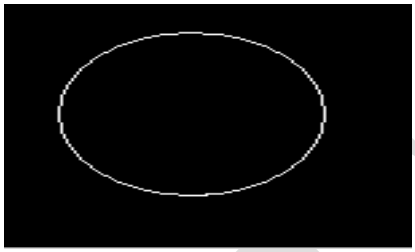
```
  int gd = DETECT,gm;
  initgraph(&gd, &gm, "C:\\TC\\BGI");
//for head
  circle(200,200,30);
//for left eye
  circle(190,190,5);
  arc(190,190,50,130,10);
//for right eye
  circle(210,190,5);
  arc(210,190,50,130,10);
//for smiley lips
  arc(200,210,180,360,10);
  line(187,210,193,210);
  line(207,210,213,210);
//for nose
  line(198,195,195,200);
  line(202,195,205,200);
  line(195,200,200,205);
  line(205,200,200,205);
  getch();
  closegraph();
}
```

OUTPUT:

```
// C Implementation for drawing circle
#include <graphics.h>

//driver code
int main()
{
    // gm is Graphics mode which is
    // a computer display mode that
    // generates image using pixels.
    // DETECT is a macro defined in
    // "graphics.h" header file
    int gd = DETECT, gm;

    // initgraph initializes the
    // graphics system by loading a
    // graphics driver from disk
    initgraph(&gd, &gm, "");

    // circle fuction
```

```
    circle(250, 200, 50);

    getch();

    // closegraph function closes the
    // graphics mode and deallocates
    // all memory allocated by
    // graphics system .
    closegraph();

    return 0;
}
```

Output



Viva Question:

1. Draws a rectangle in graphics mode.
2. What are the line attributes.
3. What are the ellipse attributes.
4. What are the circle attributes.
5. Why initgraph is used?
6. What is closegraph function?
7. What is Pixel?

8. What is putpixel function?
9. How setpixel is used?
10. What is picture element?

EXPERIMENT NO. 2

AIM:

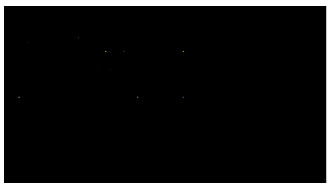
To draw a Pixel

Source Code:

```
/*  String */

// C Implementation for putpixel()
#include <graphics.h>
#include <stdio.h>

// driver code
int main()
{
    // gm is Graphics mode which is
    // a computer display mode that
    // generates image using pixels.
    // DETECT is a macro defined in
    // "graphics.h" header file
    int gd = DETECT, gm, color;

    // initgraph initializes the
    // graphics system by loading a
    // graphics driver from disk
    initgraph(&gd, &gm, "");

    // putpixel function
    putpixel(85, 35, GREEN);
```

```
    putpixel(30, 40, RED);
    putpixel(115, 50, YELLOW);
    putpixel(135, 50, CYAN);
    putpixel(45, 60, BLUE);
    putpixel(20, 100, WHITE);
    putpixel(200, 100, LIGHTBLUE);
    putpixel(150, 100, LIGHTGREEN);
    putpixel(200, 50, YELLOW);
    putpixel(120, 70, RED);

    getch();

    // closegraph function closes the
    // graphics mode and deallocates
    // all memory allocated by
    // graphics system .
    closegraph();

    return 0;
}
```

Output



EXPERIMENT NO. 3

AIM:

To Draw a Line Using DDA Algorithm

Agorithm:

1. Start.
2. declare gdriver=DETECT,gmode.
3. Initialise the graphic mode with the path location in TC folder.
4. Input the two line end-points and store the left end-points in (x1,y1).
5. Load (x1,y1) into the frame buffer;that is,plot the first point.put x=x1,y=y1.
6. Calculate dx=x2-x1 and dy=y2-y1.
7. If abs(dx) > abs(dy), do s=abs(dx).
8. Otherwise s= abs(dy).
9. Then xi=dx/s and yi=dy/s.
10. Start from k=0 and continuing till k<s,the points will be
    i. x=x+xi.
    ii. y=y+yi.
11. Place pixels using putpixel at points (x,y) in specified colour.
12. Close Graph.
13. Stop.
14. Declare variables x,y,x1,y1,x2,y2,k,dx,dy,s,xi,yi and also

Source Code:

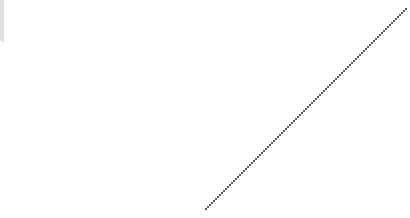
```
#include<stdio.h>
```

```
#include<conio.h>
#include<graphics.h>
void main()
{
  int x,y,x1,x2,y1,y2,k,dx,dy,s,xi,yi;
  int gdriver=DETECT,gmode;
  initgraph(&gdriver,&gmode,"C:\\tc\\bgi:");
 printf("enter first point");
 scanf("%d%d",&x1,&y1);
printf("enter second point");
scanf("%d%d",&x2,&y2); x=x1;
y=y1;
putpixel(x,y,7);
dx=x2-x1;
dy=y2-y1;
if(abs(dx)>abs(dy))
                s=abs(dx);
else
                s=abs(dy);
          xi=dx/s;
```

```
        yi=dy/s;
        x=x1;
        y=y1;
        putpixel(x,y,7);
        for(k=0;k<s;k++)
{
            x=x+xi;
            y=y+yi;
            putpixel(x,y,7);
}
getch();
closegraph();
 }
```

Output:

```
enter first point100
200
enter second point200
100
```



Result: Hence, Line Using DDA Algorithm is implemented.

Viva questions:

1. Differentiate DDA and Bresenham's Line drawing algorithm
2. What is DDA Algorithm?
3. What does DDA stands for?
4. What is refresh buffer?
5. What is resolution?
6. Why DDA is complex than bresenham's?
7. What is line equation?
8. How to calculate slope?
9. Why DDA is known as incremental scan conversion algorithm?
10. What are the operation performed in DDA?

EXPERIMENT NO. 4

AIM:

To Draw a Line Using Bresenham's Algorithm

Algorithm

Step1: Start Algorithm

Step2: Declare variable x1,x2,y1,y2,d,i1,i2,dx,dy

Step3: Enter value of x1,y1,x2,y2
Where x1,y1are coordinates of starting point
And x2,y2 are coordinates of Ending point

Step4: Calculate dx = x2-x1
Calculate dy = y2-y1
Calculate i1=2*dy
Calculate i2=2*(dy-dx)
Calculate d=i1-dx

Step5: Consider (x, y) as starting point and xendas maximum possible value of x.
If dx < 0
Then x = x2
y = y2
xend=x1
If dx > 0
Then x = x1
y = y1
xend=x2

Step6: Generate point at (x,y)coordinates.

Step7: Check if whole line is generated.
If x > = xend
Stop.

Step8: Calculate co-ordinates of the next pixel
If d < 0
Then d = d + i1
If d ≥ 0

Then d = d + i2

Increment y = y + 1

Step9: Increment x = x + 1

Step10: Draw a point of latest (x, y) coordinates

Step11: Go to step 7

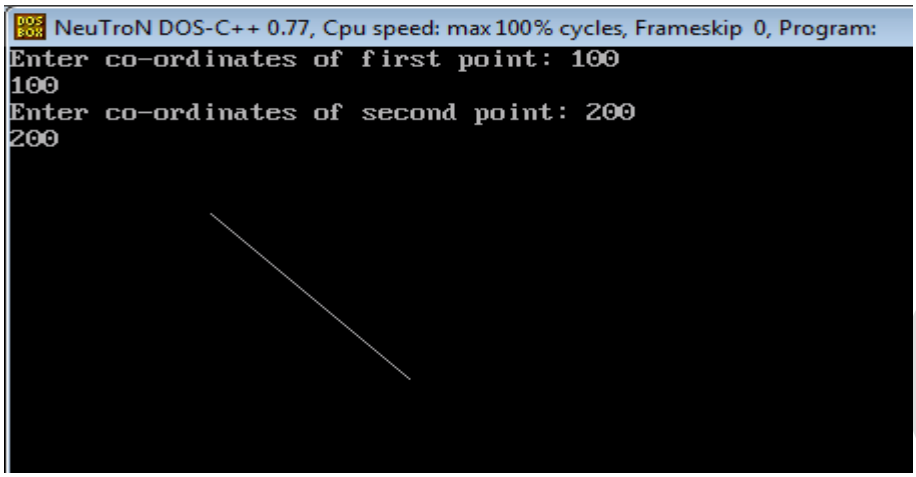
Step12: End of Algorithm

Source code

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

Output:



Result: Hence, Circle Using Bresenham's Algorithm is implemented.

Viva Questions:

1. Differentiate DDA and Bresenham's Line drawing algorithm
2. Explain the Bresenham's Line drawing algorithm?
3. What are the operation performed in Bresenham's line algorithm?
4. Why is accuracy of bresenham's is better than DDA?
5. Why bresenham's is efficient and faster line drawing algorithm?
6. What is decision parameter?
7. How to calculate decision parameter?
8. What are the conditions for decision parameter?
9. List out the advantages of bresenham's.
10. List out the disadvantages of bresenham's algorithm.

EXPERIMENT NO. 5

AIM:

Write a C program to implement midpoint circle generation algorithm of given center (x, y) and radius r.

Algorithm

Step1: Put x =0, y =r in equation 2

We have p=1-r

Step2: Repeat steps while x ≤ 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

Source Code

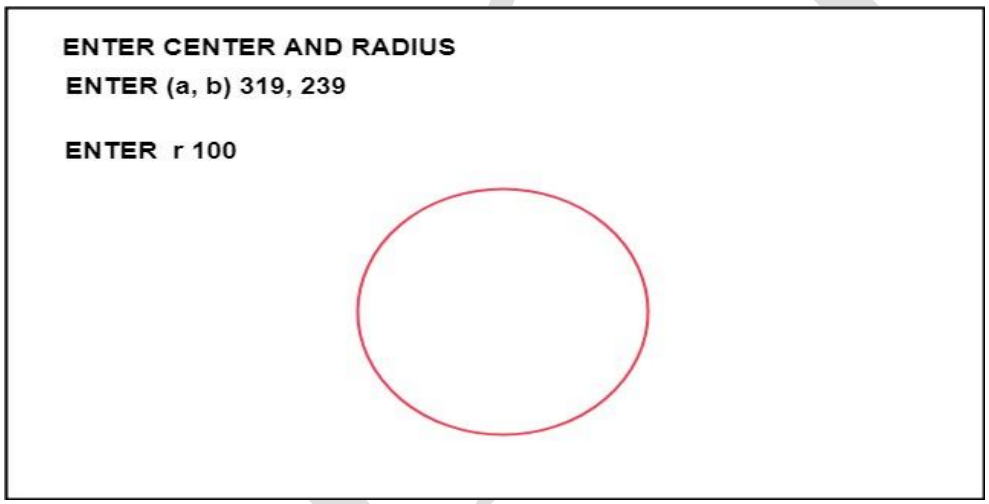
```
#include <graphics.h>
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#include <conio.h>
```

```
#include <iostream.h>

class bresen
{
   float x, y,a, b, r, p;
   public:
   void get ();
   void cal ();
};
   void main ()
   {
   bresen b;
   b.get ();
   b.cal ();
   getch ();
  }
   Void bresen :: get ()
  {
   cout<<"ENTER CENTER AND RADIUS";
    cout<< "ENTER (a, b)";
   cin>>a>>b;
   cout<<"ENTER r";
   cin>>r;
}
void bresen ::cal ()
{
   /* request auto detection */
   int gdriver = DETECT,gmode, errorcode;
```

```
int midx, midy, i;
/* initialize graphics and local variables */
initgraph (&gdriver, &gmode, " ");
/* read result of initialization */
errorcode = graphresult ();
if (errorcode ! = grOK)    /*an error occurred */
{
   printf("Graphics error: %s \n", grapherrormsg (errorcode);
   printf ("Press any key to halt:");
   getch ();
   exit (1); /* terminate with an error code */
}
x=0;
y=r;
putpixel (a, b+r, RED);
putpixel (a, b-r, RED);
putpixel (a-r, b, RED);
putpixel (a+r, b, RED);
p=5/4)-r;
while (x<=y)
{
   If (p<0)
   p+= (4*x)+6;
   else
   {
      p+=(2*(x-y))+5;
      y--;
   }
```

```
        x++;
        putpixel (a+x, b+y, RED);
        putpixel (a-x, b+y, RED);
        putpixel (a+x, b-y, RED);
        putpixel (a+x, b-y, RED);
        putpixel (a+x, b+y, RED);
        putpixel (a+x, b-y, RED);
        putpixel (a-x, b+y, RED);
        putpixel (a-x, b-y, RED);
    }
}
```

Output:



EXPERIMENT NO. 6

AIM:

3.2 To Draw An Ellipse Using Mid-Point Ellipse Drawing Algorithm.

Algorithm:

1. Start.
2. Initialize the graphic system using initgraph function.
3. Get the input of radius of major and minor arc from the user.
4. Store the values of major and minor arc in an another variable.
5. Square the values of major and minor arc.
6. Calculate decision parameter P = (square of minor axis – (square of major axis*minor axis)
7. + (0.25* square of major axis).
8. Put the pixels symmetrically at = (0, length of minor axis).
9. while (2*(square of minor axis*x)<=2*(square of major axis*y)), repeat steps 9 to step 17.
10. increment x axis by 1.
11. If P < 0
12. new P = (P+(square of minor axis* square of major axis)+ square of major axis)
13. Else
14. new P = (P+(square of minor axis*x axis)-(2*square of major axis*y axis)+ square of minor axis).
15. Decrement y by 1.
16. End of step 10 if else structure.
17. Plot symmetric points of ellipse in each quadrant.

18. End of step 8 loop.
19. This will give us ellipse only across minor axis now to draw an ellipse across major axis we proceed further.
20. Get last point of ellipse in 1st quadrant.
21. Initialize e = square of (x axis+.5)
22. Initialize f = square of (y axis-1).
23. Decision parameter P1 = ((square of minor axis*e)+(square of major axis*f)-(square of minor axis* square of major axis).
24. While y axis != 0 repeat steps 24 to step 32.
25. If P1>0
26. New P1 = (P1+ square of major axis-(2* square of major axis*x axis
27. Else
28. New P1 = (P1+(2*square of minor axis*(x axis+1))-(2* square of major axis*(y axis-1))+square of major axis).
29. Increment x axis by 1.
30. End of step 25 if else structure
31. Decrement y axis by 1.
32. Plot symmetric point in all quadrants
33. End of step 23 while loop.
34. Close the graphic system.
35. Stop.

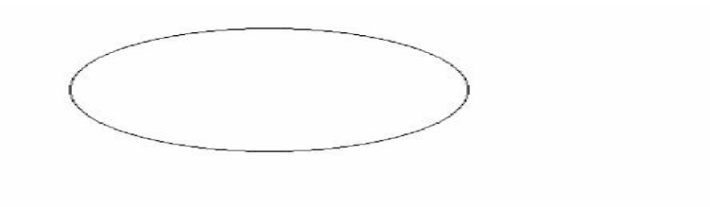
Source Code:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
 void ellips(int x,int y);
void completellipse(int r,int g,int u,int v)
{
float s,k,e,f,x; double p1,p2; s=r;k=g; e=(pow((s+.5),2));
f=(pow((k-1),2));
p2=((u*e)+(v*f)-(u*v)); ellips(s,k);
while(k>=0)
{
if(p2>0)
p2=(p2+v-(2*v*s));
else
{
}
k--;
p2=(p2+(2*u*(s+1))-(2*v*(k-1))+v); s++;
ellips(s,k);
}
}
void main()
{
int gdriver=DETECT,gmode; int a,b,x,y;
long u,v,p1; initgraph(&gdriver,&gmode,"C:\\tc\\bgi::"); printf("\n enter the length of major axis:");
scanf("\t%d",&a);
```

```
printf("\n enter the length of minor axis:"); scanf("\t%d",&b);
x=0;
y=b; u=pow(b,2);
v=pow(a,2);
p1=(u-(v*b)+(.25*v)); ellips(x,y); while(2*(u*x)<=2*(v*y))
{
x++;
if(p1<0)
p1=(p1+(2*u*v)+v);
else
{
}
p1=(p1+(2*u*x)-(2*v*y)+u); y--;
ellips(x,y);
}
completellipse(x,y,u,v);
getch();
closegraph();
}
void ellips(int x,int y)
{
putpixel(x+200,y+200,8);
putpixel(-x+200,y+200,8);
 putpixel(x+200,-y+200,8);
 putpixel(-x+200,-y+200,8);
}
```

Output:

```
enter the length of major axis:100
enter the length of minor axis:50
```



Viva Questions:

1. Explain Ellipse Function
2. Explain Mid-Point Ellipse Drawing Algorithm
3. What are the advantage of this algorithm?
4. What are the disadvantages of this algorithm?
5. Write the properties of mid-point ellipse algorithm.

EXPERIMENT NO. 7

Write a C program to implement the basic 2D transformations such as translation, scaling, rotation, shearing and reflection for a given 2D object.

AIM:

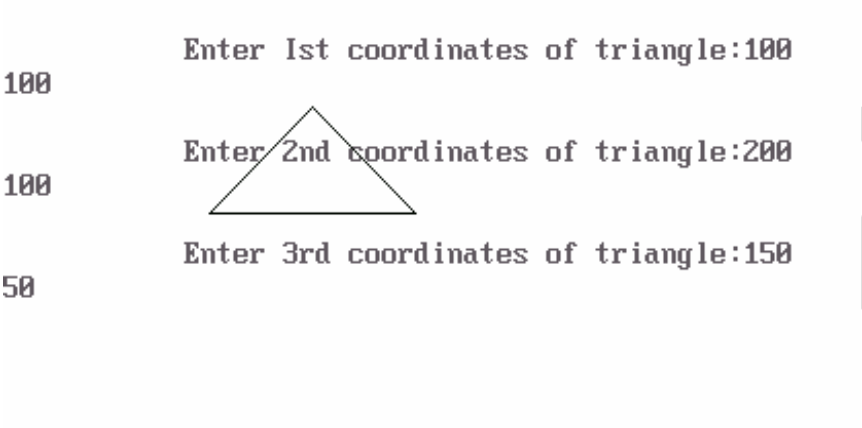
To rotate an object about origin

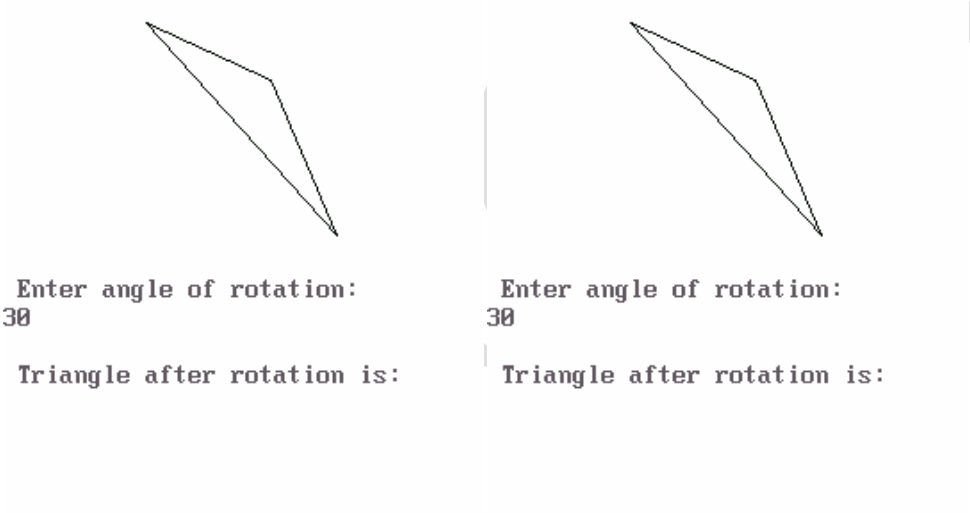
Source code:

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<process.h>
#include<math.h> void main()
{
clrscr();
int graphdriver=DETECT,graphmode; initgraph(&graphdriver,&graphmode,"...\\bgi");
int x,y,x1,a[3][3];
double b[3][3],c[3][3];
cout<<"\n Enter Ist coordinates of triangle:";
cin>>a[0][0]>>a[1][0];
cout<<"\n Enter 2nd coordinates of triangle:";
 cin>>a[0][1]>>a[1][1];
cout<<"\n Enter 3rd coordinates of triangle:";
cin>>a[0][2]>>a[1][2];
line(a[0][0],a[1][0],a[0][1],a[1][1]);
line(a[0][1],a[1][1],a[0][2],a[1][2]);
line(a[0][0],a[1][0],a[0][2],a[1][2]);
```

```
getch(); cleardevice();
cout<<"\n Enter angle of rotation:\n"; cin>>x;
b[0][0]=b[1][1]=cos((x*3.14)/180);
b[0][1]=-sin((x*3.14)/180);
b[1][0]=sin((x*3.14)/180); b[2][2]=1; b[2][0]=b[2][1]=b[0][2]=b[1][2]= 0;
for(int i=0;i<3;i++)
{
for(int j=0;j<3;j++)
{ c[i][j]=0;
for (int k=0; k<3;k++)
{
c[i][j]+=a[i][k]*b[k][j];
} x1=(c[i][j]+0.5);
a[i][j]=x1;
}
}
cout<<"\n Triangle after rotation is:\n" ;
line(a[0][0],a[1][0],a[0][1],a[1][1]);
line(a[0][1],a[1][1],a[0][2],a[1][2]);
line(a[0][0],a[1][0],a[0][2],a[1][2]);
getch(); closegraph();
}
```

Output:





Result: Hence, an object about origin, is rotated.

AIM:

To scale an object with scaling factors along X and Y directions

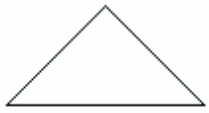
Source Program:

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
 void main()
{
int gd=DETECT,gm; initgraph(&gd, &gm,"");
cleardevice();
int x1,y1,x2,y2,x3,y3,x4,y4;
 float sx,sy;
cout<<"Enter the first coordinates of triangle\n";
cin>>x1>>y1;
cout<<"Enter the second coordinates of triangle\n";
cin>>x2>>y2;
cout<<"Enter the third coordinates of triangle\n";
 cin>>x3>>y3;
int poly[8]={x1,y1,x2,y2,x3,y3,x1,y1};
cleardevice();
drawpoly(4,poly); getch();
cout<<"Enter the scaling factors\n";
 cin>>sx>>sy;
x4=sx*x1-x1; y4=sy*y1-y1;
x1=sx*x1-x4; y1=sy*y1-y4; x2=sx*x2-x4;
y2=sy*y2-y4; x3=sx*x3-x4; y3=sy*y3-y4;
poly[0]=x1; poly[1]=y1; poly[2]=x2; poly[3]=y2;
 poly[4]=x3; poly[5]=y3; poly[6]=x1; poly[7]=y1;
```

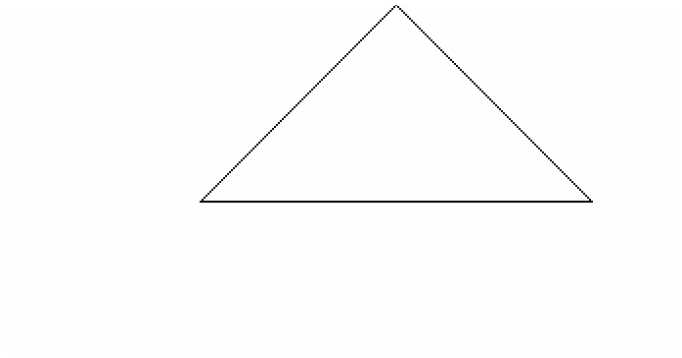
```
getch();
cleardevice();
drawpoly(4,poly);
getch();
closegraph();
}
```

Output:

```
Enter the first coordinates of triangle
100
100
Enter the second coordinates of triangle
200
100
Enter the third coordinates of triangle
150
50
```



```
Enter the scaling factors
2
2
```



Result:

Hence, scaling factors along with X and Y directions are implemented.

AIM:

To translate an object with translation parameters in X and Y directions

Source Program:

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<process.h>
#include<math.h>
void main()
{
clrscr();
int graphdriver=DETECT,graphmode; initgraph(&graphdriver,&graphmode,"...\\bgi");
int x,y,x1,y1,x2,y2,x3,y3;
cout<<"\n Enter Ist coordinates of triangle:"; cin>>x1>>y1;
cout<<"\n Enter 2nd coordinates of triangle:"; cin>>x2>>y2;
cout<<"\n Enter 3rd coordinates of triangle:"; cin>>x3>>y3;
cleardevice(); line(x1,y1,x2,y2);
line(x2,y2,x3,y3);
line(x1,y1,x3,y3); getch(); cleardevice();
cout<<"\n Enter translatio factors :\n"; cin>>x>>y;
x1-=x;
y1-=y;
x2-=x;
y2-=y;
x3-=x;
y3-=y;
```

```
cleardevice(); line(x1,y1,x2,y2);
line(x2,y2,x3,y3);
line(x1,y1,x3,y3);
 getch();
 closegraph();
}
```

Output:

```
            Enter 1st coordinates of triangle:100
100
            Enter 2nd coordinates of triangle:200
100
            Enter 3rd coordinates of triangle:150
50
```

Result: Hence translation of an object is implemented

AIM:

To Rotate a Point About a Point

Source Code:

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
#include<dos.h> void main()
{
clrscr();
int gm,gd=DETECT; initgraph(&gd,&gm,"");
int h,k,x1,y1,x2,y2,x3,y3; float t;
cout<<" OUTPUT"<<endl;
cout<<"Enter the coordinates of point"<<endl; cin>>x2>>y2;
putpixel(x2,y2,2);
cout<<"Enter the coordinates of point around which rotation is done"<<endl; cin>>h>>k;
putpixel(h,k,2);
cout<<"Enter the angle for rotation"<<endl; cin>>t;
cleardevice(); x1=(h*cos(t))-(k*sin(t));
y1=(h*sin(t))+(k*cos(t)); x3=x1+x2-h;
y3=y1+y2-k;
cout<<"Point after rotation is:";
putpixel(x3,y3,2);
getch();
closegraph();
}
```

Output:

```
Enter the coordinates of point
100
100
Enter the coordinates of point around which rotation is done
50
50
Enter the angle for rotation
30
```

```
Point after rotation is:
```

Result: Hence Rotation of a Point about a Point is implemented

AIM:

To Rotate a Point about Origin

Source Code:

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
#include<dos.h> void main()
{
clrscr();
int gm,gd=DETECT;
initgraph(&gd,&gm,"");
int h,k,x1,y1,x2,y2,x3,y3;
float t;
cout<<" OUTPUT"<<endl;
cout<<"Enter the coordinates of point"<<endl; cin>>x2>>y2;
putpixel(x2,y2,2);
cout<<"Enter the angle for rotation"<<endl; cin>>t;
cleardevice();
x1=int(x2*cos(t*3.14/180))-(y2*sin(t*3.14/180));
y1=int(x2*sin(t*3.14/180))+(y2*cos(t*3.14/180));
 cout<<"Point after rotation is:";
putpixel(x1,y1,2);
getch();
closegraph();
}
```

Output:

```
Enter the coordinates of point
100
100
Enter the angle for rotation
30
```

```
Point after rotation is:
```

Result: Hence Rotation of a Point About Origin is implemented

AIM:

To Reflect a Triangle

Source Code

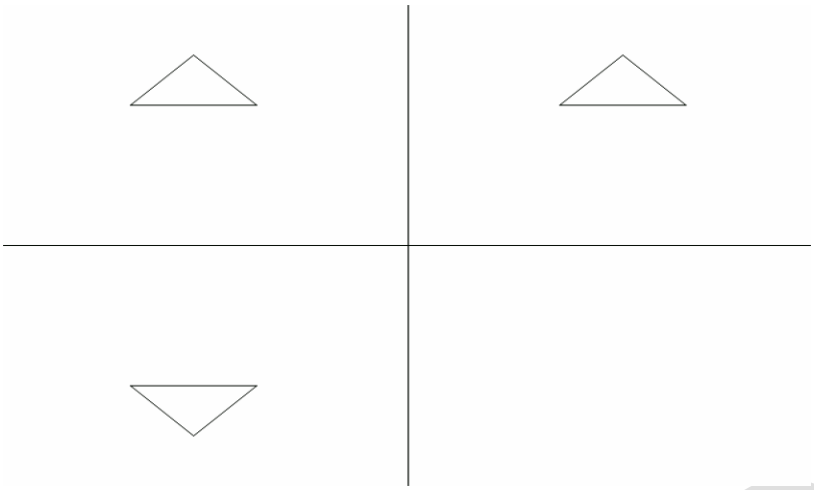
```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<process.h>
#include<math.h> void main()
{
clrscr();
int graphdriver=DETECT,graphmode; initgraph(&graphdriver,&graphmode,"...\\bgi");
int x,y,x1,a[3][3];
double b[3][3],c[3][3];
cout<<"\n Enter Ist coordinates of triangle:"; cin>>a[0][0]>>a[1][0];
cout<<"\n Enter 2nd coordinates of triangle:"; cin>>a[0][1]>>a[1][1];
cout<<"\n Enter 3rd coordinates of triangle:"; cin>>a[0][2]>>a[1][2];
cout<<"\n Enter 1. for reflection in x-axis:\n"; cout<<"\n Enter 2. for reflection in y-axis:\n"; cout<<"\n Enter
3. for reflection in both the axis:\n"; cin>>x;
cleardevice(); line(320,0,320,479); line(0,240,639,240);
line(a[0][0],a[1][0],a[0][1],a[1][1]);
line(a[0][1],a[1][1],a[0][2],a[1][2]);
line(a[0][0],a[1][0],a[0][2],a[1][2]);
switch(x)
{
case 1:b[0][0]=640-a[0][0];
b[0][1]=640-a[0][1];
b[0][2]=640-a[0][2];
```

```
b[1][0]=a[1][0];
b[1][1]=a[1][1];
b[1][2]=a[1][2]; line(320,0,320,479); line(0,240,639,240);
line(b[0][0],b[1][0],b[0][1],b[1][1]);
line(b[0][1],b[1][1],b[0][2],b[1][2]);
line(b[0][0],b[1][0],b[0][2],b[1][2]);
getch(); break;
case 2:b[1][0]=480-a[1][0];
b[1][1]=480-a[1][1];
b[1][2]=480-a[1][2];
b[0][0]=a[0][0];
b[0][1]=a[0][1];
b[0][2]=a[0][2]; line(320,0,320,479); line(0,240,639,240);
line(b[0][0],b[1][0],b[0][1],b[1][1]);
line(b[0][1],b[1][1],b[0][2],b[1][2]);
line(b[0][0],b[1][0],b[0][2],b[1][2]);
getch(); break;
case 3: b[0][0]=640-a[0][0];
b[0][1]=640-a[0][1];
b[0][2]=640-a[0][2];
b[1][0]=a[1][0];
b[1][1]=a[1][1];
b[1][2]=a[1][2]; line(320,0,320,479); line(0,240,639,240);
line(b[0][0],b[1][0],b[0][1],b[1][1]);
line(b[0][1],b[1][1],b[0][2],b[1][2]);
line(b[0][0],b[1][0],b[0][2],b[1][2]); b[1][0]=480-a[1][0];
b[1][1]=480-a[1][1];
b[1][2]=480-a[1][2];
```

```
b[0][0]=a[0][0];
b[0][1]=a[0][1];
b[0][2]=a[0][2]; line(320,0,320,479); line(0,240,639,240);
line(b[0][0],b[1][0],b[0][1],b[1][1]);
line(b[0][1],b[1][1],b[0][2],b[1][2]);
line(b[0][0],b[1][0],b[0][2],b[1][2]);
getch(); break;
}
getch(); closegraph();
}
```

Output:

```
          Enter 1st coordinates of triangle:100
100
          Enter 2nd coordinates of triangle:200
100
          Enter 3rd coordinates of triangle:150
50
 Enter 1. for reflection in x-axis:
 Enter 2. for reflection in y-axis:
 Enter 3. for reflection in both the axis:
3
```



Result: Hence Reflection a Triangle is implemented

Viva Questions:

1. What is scaling
2. What are the different transformation.
3. How to do 2D matrix representation.
4. Write the matrix representation of 2D translation.
5. Write the matrix representation of 2D scaling.
6. Write the matrix representation of 2D rotation.
7. Write the matrix representation of 2D shearing.
8. Write the matrix representation of 2D reflection.
9. What is homogenous coordinate?
10. What is shift vector?

EXPERIMENT NO. 8

Aim:

Write a C program to implement flood fill algorithm

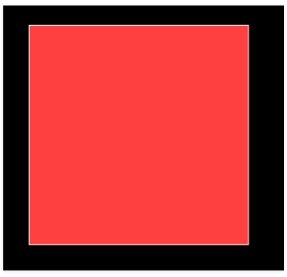
Source Code

```
// program to fill polygon using floodfill
// algorithm
#include <graphics.h>
#include <stdio.h>

// flood fill algorithm
void flood(int x, int y, int new_col, int old_col)
{
        // check current pixel is old_color or not
        if (getpixel(x, y) == old_col) {

                // put new pixel with new color
                putpixel(x, y, new_col);

                // recursive call for bottom pixel fill
                flood(x + 1, y, new_col, old_col);

                // recursive call for top pixel fill
                flood(x - 1, y, new_col, old_col);

                // recursive call for right pixel fill
                flood(x, y + 1, new_col, old_col);
```

```
                // recursive call for left pixel fill
                flood(x, y - 1, new_col, old_col);
        }
}

int main()
{
        int gd, gm = DETECT;

        // initialize graph
        initgraph(&gd, &gm, "");

        // rectangle coordinate
        int top, left, bottom, right;

        top = left = 50;
        bottom = right = 300;

        // rectangle for print rectangle
        rectangle(left, top, right, bottom);

        // filling start coordinate
        int x = 51;
        int y = 51;

        // new color to fill
        int newcolor = 12;
```

```
        // new color which you want to fill
        int oldcolor = 0;

        // call for fill rectangle
        flood(x, y, newcolor, oldcolor);
        getch();

        return 0;
}
```

Output:



Viva Questions:

1. What is filled algorithm?
2. What are the approaches for filling algorithm?
3. What is flood fill algorithm?
4. What are the steps to flood fill algorithm?
5. What is seed pixel?
6. What is 4 connected approach?
7. What is 8 connected approach?

8. What are the between 4 connected and 8 connected approach?
9. What are merits of flood fill algorithm?
10. What are the demerits of flood fill algorithm?

EXPERIMENT NO. 9

Aim:

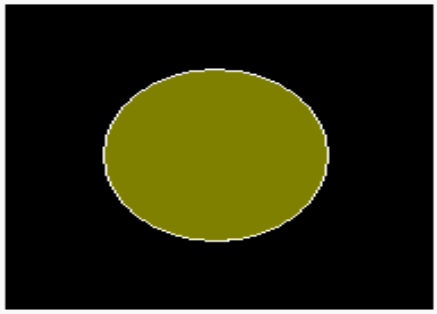
Write a C program to implement boundary fill algorithm

Source Code

```
// C Implementation for Boundary Filling Algorithm
#include <graphics.h>

// Function for 4 connected Pixels
void boundaryFill4(int x, int y, int fill_color,int boundary_color)
{
        if(getpixel(x, y) != boundary_color &&
        getpixel(x, y) != fill_color)
        {
                putpixel(x, y, fill_color);
                boundaryFill4(x + 1, y, fill_color, boundary_color);
                boundaryFill4(x, y + 1, fill_color, boundary_color);
                boundaryFill4(x - 1, y, fill_color, boundary_color);
                boundaryFill4(x, y - 1, fill_color, boundary_color);
        }
}
//driver code
int main()
{
        // gm is Graphics mode which is
        // a computer display mode that
        // generates image using pixels.
        // DETECT is a macro defined in
```

```
        // "graphics.h" header file
        int gd = DETECT, gm;

        // initgraph initializes the
        // graphics system by loading a
        // graphics driver from disk
        initgraph(&gd, &gm, "");

        int x = 250, y = 200, radius = 50;

        // circle function
        circle(x, y, radius);

        // Function calling
        boundaryFill4(x, y, 6, 15);

        delay(10000);

        getch();

        // closegraph function closes the
        // graphics mode and deallocates
        // all memory allocated by
        // graphics system .
        closegraph();

        return 0;
}
```

Output:



Viva Questions:

1. What is filled algorithm?
2. What are the approaches for filling algorithm?
3. What is Boundary fill algorithm?
4. What are the steps to Boundary fill algorithm?
5. What is seed pixel?
6. What is 4 connected approach?
7. What is 8 connected approach?
8. What are the between 4 connected and 8 connected approach?
9. What are merits of Boundary fill algorithm?
10. What are the demerits of Boundary fill algorithm?

EXPERIMENT NO. 10

Aim: To implement Cohen Sutherland 2D line clipping algorithm

Algorithm:

1. Start.
2. Initialize the graphic system using initgraph function.
3. Get the input of window co ordinates from the user and draw a window.
4. Get the input of line co ordinates from user and draw the line.
5. Calculate the region code of each end point of line using relation given in steps 6 to step
6. Let (x,y) be the co ordinates of end point of line and (xmin,ymin), (xmax,ymax) be co ordinates of world window
7. If y –ymax = +ve
8. MSB region code = 1.
9. Else MSB region code = 0.
10. If ymin – y = +ve
11. Region code = 1.
12. Else Region code = 0.
13. If x – xmax = +ve
14. Region code = 1.
15. Else Region code = 0.
16. If xmin – x = +ve
17. LSB Region code = 1.

18. Else LSB Region code = 0.

19. Calculate region code of both end points.

20. Logically and both region code.

21. If Logically anded result is = 0

22. Line is not a clipping candidate.

23. Else.

24. Line is a clipping candidate.

25. Calculate slope of line using formula slope=(y2-y1)/(x2-x1).

26. If line is to be horizontally clipped.

27. New y = ymin or ymax.

28. New x = x1 + ((new y - y1)/slope).

29. If line is to be vertically clipped.
30. New x = xmin or xmax.
31. New y = y1+slope*(new x –x1).
32. Clip the lines from these intersection points.
33. Display the new line.
34. Close the graphic system.
35. Stop.

.

Source code:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void storepoints(int,int,int,int,int,int,int[]); void main()
{
int gdriver=DETECT,gmode;
int x1,x2,y1,y2,xmax,ymax,xmin,ymin,a[10],b[10],xi1,xi2,yi1,yi2,flag=0; float m;
int i; clrscr();
printf("output"); printf("\n");
printf("enter the value of x1,y1,x2,y2: >"); scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
printf("enter the value of xmax,ymax,xmin,ymin:");
scanf("%d%d%d%d",&xmax,&ymax,&xmin,&ymin);
storepoints(x2,y2,ymin,ymax,xmax,xmin,b); for(i=1;i<=4;i++)
{
if(a[i]*b[i]==0)
flag=1;
else
}
```

```
flag=0;
if(flag==1)
{
m=(y2-y1)/(x2-x1); xi1=x1;
yi1=y1;
}
if(a[1]==1)
{
}
else
{
yi1=ymax; xi1=x1+((1/m)*(yi1-y1));
if(a[2]==1)
{
yi1=ymin; xi1=x1+((1/m)*(yi1-y1));
}
}
if(a[3]==1)
{
xi1=xmax; yi1=y1+(m*(xi1-x1));
}
if(a[4]==1)
{
}
else
xi1=xmin; yi1=y1+(m*(xi1-x1));
if(b[1]==1)
{
}
else
```

```
yi2=ymax; xi2=x2+((1/m)*(yi2-y2));
if(b[2]==1)
{
}
else
yi2=ymin; xi2=x2+((1/m)*(yi2-y2));
if(b[3]==1)
{
clrscr();
}
else
xi2=xmax; yi2=y2+((1/m)*(xi2-x2));
if(b[4]==1)
{
xi2=xmin; yi2=y2+(m*(xi2-x2));
}
initgraph(&gdriver,&gmode,"c://tc//bgi:");
 rectangle(xmin,ymin,xmax,ymax);
 line(x1,y1,x2,y2);
delay(5000);
closegraph();
clrscr();
initgraph(&gdriver,&gmode,"c://tc//bgi:");
 line(xi1,yi1,xi2,yi2);
 rectangle(xmin,ymin,xmax,ymax); if(flag==0)
{
printf("\n no clipping is required");
}
getch(); closegraph();
}
```

```
void storepoints(int x1,int y1,int ymax,int xmax,int xmin,int ymin,int c[10])
{
if((y1-ymax)>0)
c[1]=1;
else
c[1]=0;
if((ymin-y1)>0)
c[2]=1;
else
c[2]=0;
if((x1-xmax)>0)
c[3]=1;
else
c[3]=0;
if((xmin-x1)>0)
c[4]=1;
else
}
c[4]=0;
}
```

Output:

```
enter the value of
x1,y1,x2,y2:_>10
10

100

100
```

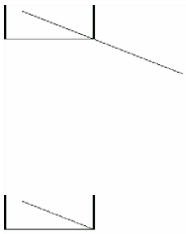
enter the value of

xmax,ymax,xmin,ymin50

50

0

0



Result: Hence, Cohen Sutherland 2D line clipping algorithm is implemented

Viva Questions:

1. What is Line Clipping
2. Explain Cohen Suthertland Line Clipping in detail.
3. What is line clipping example?
4. What are the steps of line clipping?
5. Which is the best line clipping algorithm?
6. What are the conditions for a line to be clipping candidate?
7. Why clipping is used in graphics?
8. How many methods of text clipping are there?
9. What are the three types of clipping?
10. Why do we need clipping?

EXPERIMENT NO. 11

AIM:

To implement 3D transformation Translation, Rotation, Scaling

Source Code

```
#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

void trans();

//void axis();

void scale();

void rotate();

int maxx,maxy,midx,midy;

/*void axis()

{

//        getch();
```

```
 //        cleardevice();

          line(midx,0,midx,maxy);

          line(0,midy,maxx,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();

                        getch();

                //      closegraph();

                        break;

        case 2 :        scale();

                        getch();

                //      closegraph();

                        break;
        case 3 :        rotate();

                        getch();

                //      closegraph();

                        break;
        case 4 :    break;

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

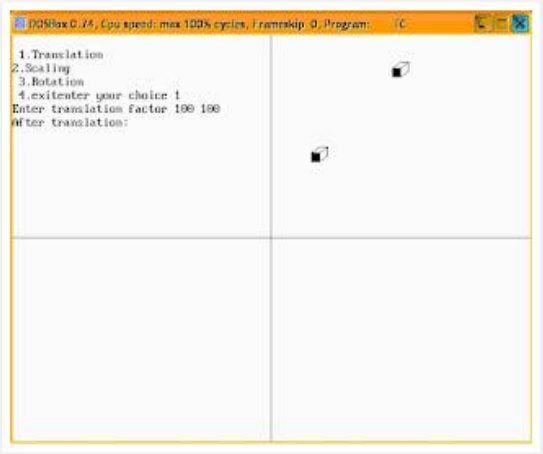
```
        } while(ch<4);
}
void trans()
{

        int x,y,z,o,x1,x2,y1,y2;
        maxx=getmaxx();
        maxy=getmaxy();
        midx=maxx/2;
        midy=maxy/2;
        //axis();
        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()

{

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

        maxx=getmaxx();

        maxy=getmaxy();

        midx=maxx/2;
```

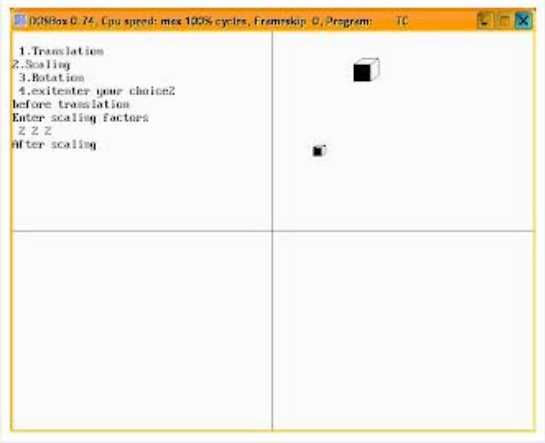
```
        midy=maxy/2;
        //axis();
        bar3d(midx+50,midy-100,midx+60,midy-90,5,1);
        printf("before translation\n");
        printf("Enter scaling factors\n");
        scanf("%d %d %d", &x,&y,&z);
        printf("After scaling\n");
        bar3d(midx+(x*50),midy-(y*100),midx+(x*60),midy-(y*90),5*z,1);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

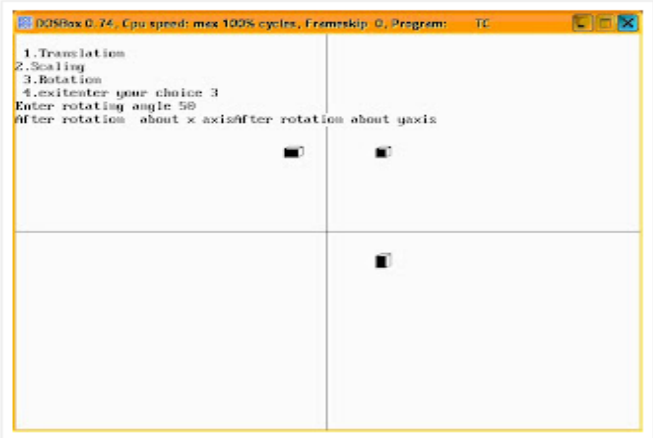
```
midy=maxy/2;
//axis();
bar3d(midx+50,midy-100,midx+60,midy-90,5,1);
printf("Enter rotating angle");
scanf("%d",&o);
x1=50*cos(o*3.14/180)-100*sin(o*3.14/180);
y1=50*sin(o*3.14/180)+100*cos(o*3.14/180);
x2=60*cos(o*3.14/180)-90*sin(o*3.14/180);
y2=60*sin(o*3.14/180)+90*cos(o*3.14/180);
//      axis();
//      printf("After rotation about z axis");
//      bar3d(midx+x1,midy-y1,midx+x2,midy-y2,5,1);
//axis();
printf("After rotation  about x axis");
bar3d(midx+50,midy-x1,midx+60,midy-x2,5,1);
```

```
        //axis();

        printf("After rotation about yaxis");

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

}
```

OUTPUT:







Viva Questions:

1. What are the three transforms in 3D Computer Graphics?
2. How many types of transformations are there in 3D?
3. What are the 3D transformation discuss with example?
4. What are the steps involved in 3D transformation in Computer Graphics?
5. What is Translation .
6. What is Rotation .
7. What is shearing in 3d transformation.
8. What is Reflection in 3d transformation.

EXPERIMENT NO. 12

Aim: To draw a moving helicopter

Source Code

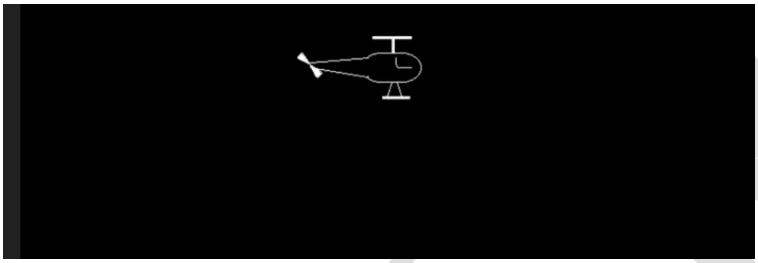
```
#include<graphics.h>
#include<stdio.h>
#include<conio.h>
#include<dos.h>


void bird();
void handup();
void handdown();
void heli();
void para();
int a,b,c,d,e,k;

void main()
{

int gdriver = DETECT,gmode;
initgraph(&gdriver,&gmode,"C:\\Turboc3\\BGI");

//setcolor(GREEN);
b=0;
c=0;
d=0;

for(a=0;a<80;a++)
{
heli();
```

```
delay(100);
cleardevice();
k=k+3;
}
heli();
for(a=0;a<90;a++)
{
para();
heli();
if(a>20){k=k+6;}
delay(100);
cleardevice();
if(a<50){
b=b+2;
}
}

getch();

}
void heli(){
ellipse(100+k,200,270,90,20,15);
line(100+k,185,80+k,185);
line(100+k,215,80+k,215);
ellipse(80+k,210,160,270,15,5);
ellipse(80+k,190,90,170,15,5);
line(65+k,190,5+k,195);
line(65+k,210,5+k,200);
//circle(5+k,197,15);
if(a%2==0){
```

```
pieslice(5+k,197,30,60,15);
pieslice(5+k,197,210,240,15);
}
else{
pieslice(5+k,197,120,150,15);
pieslice(5+k,197,300,330,15);
}
bar(90+k,185,92+k,170);
bar(70+k,168,110+k,170);
line(94+k,190,94+k,196);
line(96+k,200,110+k,200);
ellipse(102+k,195,160,220,8,8);
line(90+k,215,85+k,230);
line(95+k,215,100+k,230);
bar(80+k,230,108+k,232);
}

void para()
{
arc(340,260+b,0,180,20);
arc(340,263+b,0,180,20);
line(320,263+b,330,280+b);
line(360,263+b,350,280+b);
line(330,280+b,331,290+b);
line(328,280+b,330,290+b);
line(350,280+b,348,290+b);
line(349,280+b,347,290+b);
ellipse(340,280+b,0,360,4,6);
rectangle(336,285+b,344,302+b);
bar(330,288+b,336,289+b);
```

```
bar(338,302+b,339,310+b);
bar(342,302+b,343,310+b);
line(336,302+b,339,312+b);
ellipse(344,312+b,0,360,2,1);
ellipse(338,312+b,0,360,2,1);

}
```

Output:



Result: Hence a moving helicopter is drawn

Aim: To draw a moving fan

Source Code

```
#include<graphics.h>   //for graphics
#include<stdio.h>      //standard input/output
#include<string.h>    //string function
#include<conio.h>   //console input output
#include<math.h>   //math calculation
#define PI 3.14   //value constant pi=3.14
char speed[4]; //array declaration for speed
int theta=0,change=1,temp=0,ch;
float x,y,r=100;
void main()
{
int gd=DETECT,gm;
initgraph(&gd,&gm,"c:\\tc\\bgi"); //path where ur BGI file is stored
strcpy(speed,"on 1");
outtextxy(200,20,"Working Fan");
outtextxy(150,50,"Use number 0 1 2 3 to change fan speed");
outtextxy(475,375,"Fan Speed");
fan:
do
{
cleardevice();
```

```
outtextxy(200,20,"Working Fan");
outtextxy(150,50,"Use number 0 1 2 3 to change fan speed");
outtextxy(475,375,"Fan Speed");
if(kbhit())
{
ch=getch();
if(ch=='0')
{
rectangle(495,395,535,410);
floodfill(515,405,15);
temp=1;
}
if(ch=='1')
{
strcpy(speed,"on 1");
change=1;
}
else if(ch=='2')
{
strcpy(speed,"on 2");
change=3;
}
else if(ch=='3')
```

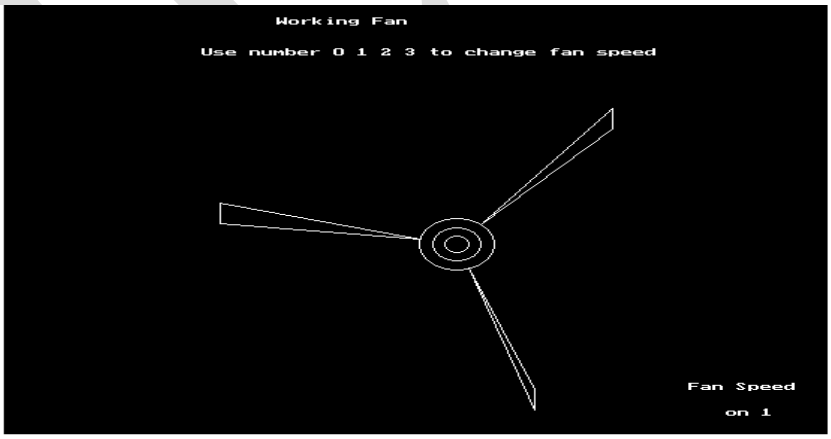
```
{
strcpy(speed,"on 3");
change=18;
}
}
outtextxy(500,400,speed);
circle(320,240,(r/12));
circle(320,240,(r/6));
circle(320,240,(r/4));
x=r*(cos((PI * theta)/180));
y=r*sin((PI * theta)/180);
line(320+(x/4),240+(y/4),320+(1.6*x),230+(1.6*y)); /*draw the line for fan*/
line(320+(x/4),240+(y/4),320+(1.6*x),250+(1.6*y));
line(320+(1.6*x),230+(1.6*y),320+(1.6*x),250+(1.6*y));
x=r*cos((PI *(theta+120))/180);
y=r*sin((PI *(theta+120))/180);
line(320+(x/4),240+(y/4),320+(1.6*x),230+(1.6*y));
line(320+(x/4),240+(y/4),320+(1.6*x),250+(1.6*y));
line(320+(1.6*x),230+(1.6*y),320+(1.6*x),250+(1.6*y));
x=r*cos((PI * (theta+240))/180);
y=r*sin((PI * (theta+240))/180);
line(320+(x/4),240+(y/4),320+(1.6*x),230+(1.6*y));
line(320+(x/4),240+(y/4),320+(1.6*x),250+(1.6*y));
```

```
line(320+(1.6*x),230+(1.6*y),320+(1.6*x),250+(1.6*y));
if(temp!=1) delay(36/change);
else
{
ch=getche();
if(ch=='\r') exit(0);
if(ch=='0') temp=1;
if(ch=='1')   //for speed
{
strcpy(speed,"on 1");
temp=0;
change=1;
}
else if(ch=='2')
{
strcpy(speed,"on 2");
temp=0;
change=3;
}
else if(ch=='3')
{
strcpy(speed,"on 3");
temp=0;
```

```
change=18;
}
else
{
strcpy(speed,"off!");
temp=1;
}
}
if(theta==360) theta=0;
theta++;
}
while(temp==0);
goto fan;
```

3.3 }

Output:



Result: Hence a moving fan is drawn

Viva Question:

1. What is graphics function
2. What is rectangle function
3. Draws a rectangle in graphics mode.
4. What are the line attributes.
5. What are the ellipse function parameters.
6. What are the circle function attributes/ parameters.
7. Why initgraph is necessary?
8. What is closegraph function and what is its need?

EXPERIMENT NO. 13

Aim: To draw a moving car

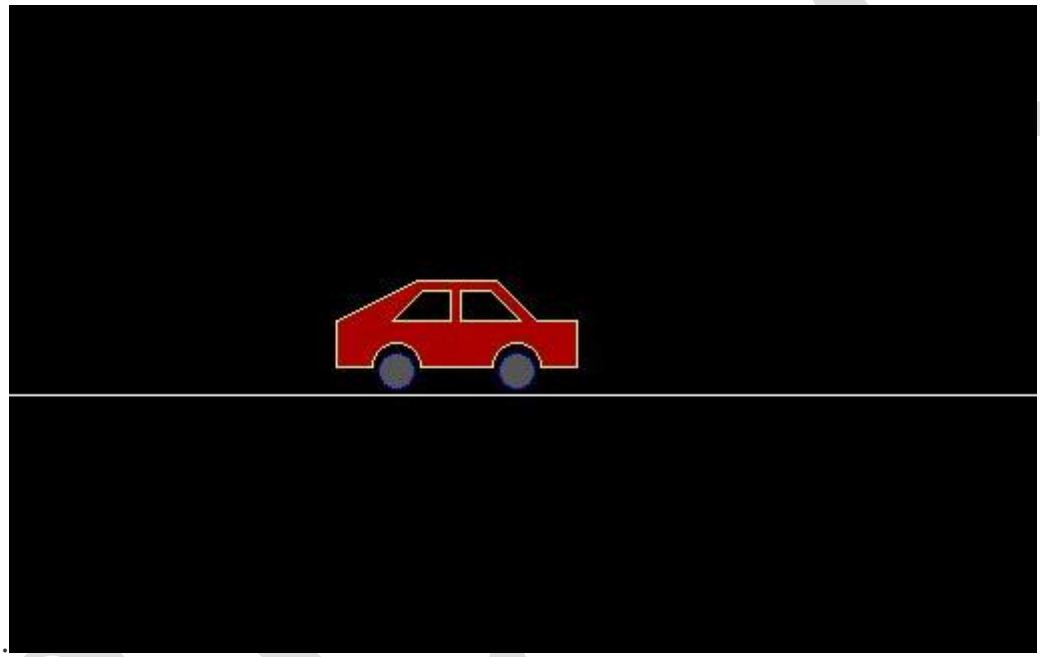
Source Code

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

int main() {
   int gd = DETECT, gm;
   int i, maxx, midy;

   /* initialize graphic mode */
   initgraph(&gd, &gm, "X:\\TC\\BGI");
   /* maximum pixel in horizontal axis */
   maxx = getmaxx();
   /* mid pixel in vertical axis */
   midy = getmaxy()/2;

   for (i=0; i < maxx-150; i=i+5) {
      /* clears screen */
      cleardevice();

      /* draw a white road */
      setcolor(WHITE);
      line(0, midy + 37, maxx, midy + 37);

      /* Draw Car */
```

```
setcolor(YELLOW);
setfillstyle(SOLID_FILL, RED);

line(i, midy + 23, i, midy);
line(i, midy, 40 + i, midy - 20);
line(40 + i, midy - 20, 80 + i, midy - 20);
line(80 + i, midy - 20, 100 + i, midy);
line(100 + i, midy, 120 + i, midy);
line(120 + i, midy, 120 + i, midy + 23);
line(0 + i, midy + 23, 18 + i, midy + 23);
arc(30 + i, midy + 23, 0, 180, 12);
line(42 + i, midy + 23, 78 + i, midy + 23);
arc(90 + i, midy + 23, 0, 180, 12);
line(102 + i, midy + 23, 120 + i, midy + 23);
line(28 + i, midy, 43 + i, midy - 15);
line(43 + i, midy - 15, 57 + i, midy - 15);
line(57 + i, midy - 15, 57 + i, midy);
line(57 + i, midy, 28 + i, midy);
line(62 + i, midy - 15, 77 + i, midy - 15);
line(77 + i, midy - 15, 92 + i, midy);
line(92 + i, midy, 62 + i, midy);
line(62 + i, midy, 62 + i, midy - 15);
floodfill(5 + i, midy + 22, YELLOW);
setcolor(BLUE);
setfillstyle(SOLID_FILL, DARKGRAY);
/*  Draw Wheels */
circle(30 + i, midy + 25, 9);
circle(90 + i, midy + 25, 9);
floodfill(30 + i, midy + 25, BLUE);
floodfill(90 + i, midy + 25, BLUE);
```

```
        /* Add delay of 0.1 milli seconds */
        delay(100);
    }

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

Output



Result: Hence a moving car is drawn

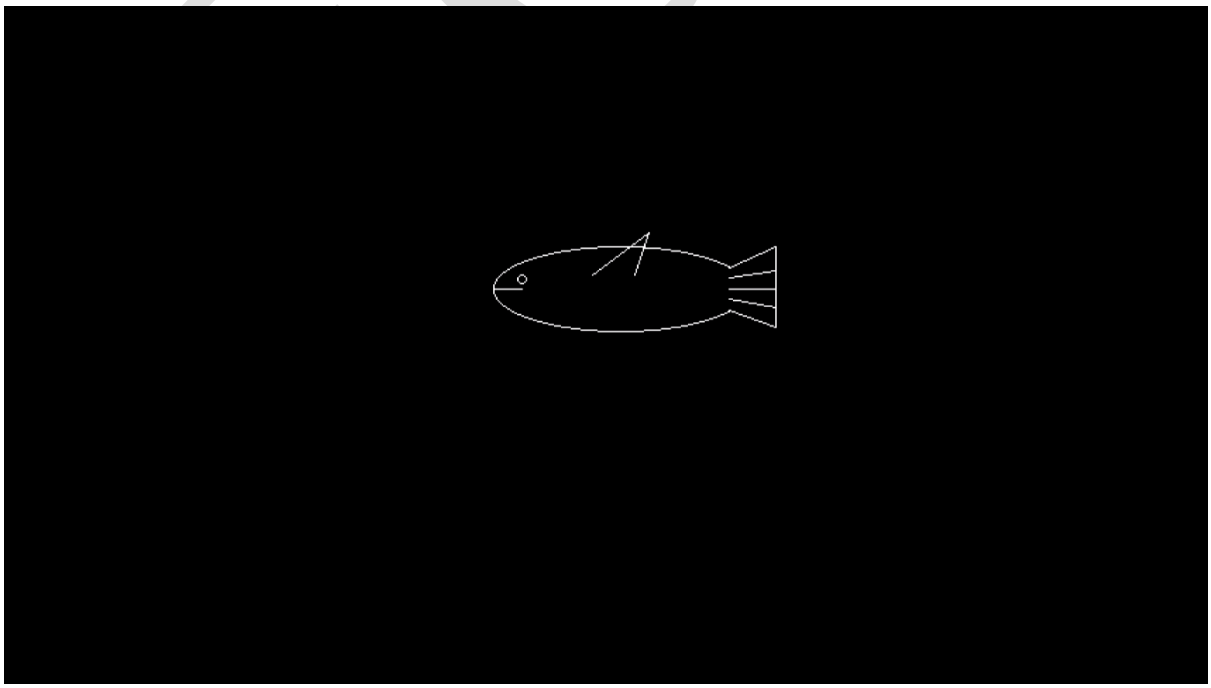
Aim: To draw a moving fish

Source Code

```
#include<time.h>
#include<conio.h>
#include<stdlib.h>
#include<dos.h>
#include<graphics.h>

void main()
{
 int gd=DETECT,gm;
 int x=10,y=200,x1=675,y1=380;
 int stangle=35,endangle=140,radius=90;

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

 while(!kbhit())
 {
  cleardevice();
  setbkcolor(BLACK);

  if(x<640)
  {
   x+=5;
   y+=1;
   arc(x,y,stangle,endangle+35,radius);
   arc(x,y-110,190,323,radius+2);
   circle(x+40,y-60,5);
   line(x-90,y-90,x-90,y-8);
  }
```

```
else
{
 x1-=5;
 y1-=1;
 arc(x1,y1,stangle-30,endangle+4,radius);
 arc(x1,y1-110,217,350,radius+2);
 circle(x1-40,y1-60,5);
 line(x1+90,y1-90,x1+90,y1-10);
}
setcolor(YELLOW);
delay(90);
}
closegraph();
}
```

OUTPUT



Result: Hence a moving fish is drawn

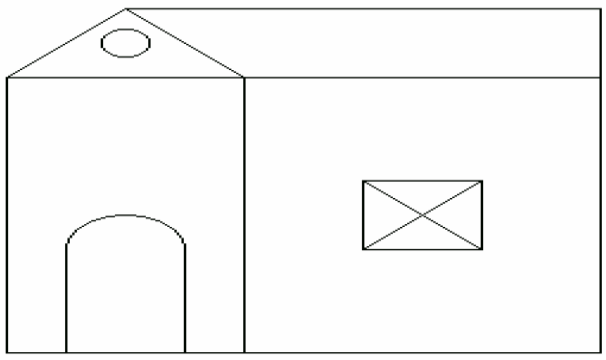
AIM:

To Draw a Hut Using Simple Graphic Functions

Source Code

```
#include<conio.h>
#include<iostream.h>
#include<graphics.h>
#include<math.h>
#include<dos.h>
#include<process.h> void main()
{
int graphdriver=DETECT,graphmode; initgraph(&graphdriver,&graphmode,"...\\bgi");
line(100,100,150,50);
line(150,50,200,100); line(100,100,200,100); line(150,50,350,50); line(200,100,350,100);
line(350,50,350,100);
circle(150,75,10); rectangle(100,100,200,300);
 rectangle(200,100,350,300); rectangle(250,175,300,225);
line(250,175,300,225); line(300,175,250,225); line(125,300,125,225); line(175,300,175,225);
arc(150,225,0,180,25);
getch(); closegraph();
}
```

Output:



Result: Hence a Hut Using Simple Graphic Functions is drawn

Viva Question:

1. What is graphics function
2. What is viewing transformation?
3. What is an animation?
4. What is Fractals?
5. What are Morphing and tweening?
6. What are blobby objects?
7. Distinguish between window port and viewport?
8. What is interactive computer Graphics?
9. What does it mean by RGB?

EXPERIMENT NO. 14

AIM:

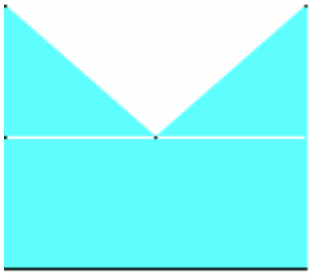
To implement Polygon fill algorithm

Source Code

```
#include<conio.h>
#include<iostream.h>
#include<graphics.h>
#include<math.h>
#include<dos.h>
#include<process.h> void main()
{
int graphdriver=DETECT,graphmode; initgraph(&graphdriver,&graphmode,"...\\bgi");
 int p=1,x;
int a[12]={100,100,150,150,200,100,200,200,100,200,100,100};
drawpoly(6,a);
for(int i=100;i<200;i++)
{
 p=1;
for(int j=100;j<=200;j++)
{
x=getpixel(j,i);
for(int d=0;d<11;d++)
{
if(j==a[d]&&i==a[d+1] ) break;
else
{
if(x>0&&d==10) p++;
if(p%2==0)
```

```
putpixel(j,i,4);
}
}
}
}
getch();
closegraph();
}
```

Output:



Result: Hence the Polygon is filled

Viva Question:

1. Which clipping algorithm is used for polygon clipping?
2. How do you clip a polygon in computer graphics?
3. Which clipping algorithm is used for polygon clipping Mcq?

4. Which vertex of the polygon is clipping first in polygon clipping?
5. The primary use of clipping in computer graphics is to remove?
6. Which clipping algorithm is used for polygon clipping?
7. How many methods for text clipping there?
8. Which vertex of the polygon is clipped first in polygon clipping?
9. We can resize the bitmap image.
10. The Cohen-Sutherland algorithm divides the region into number of spaces.

EXPERIMENT NO. 15

AIM:

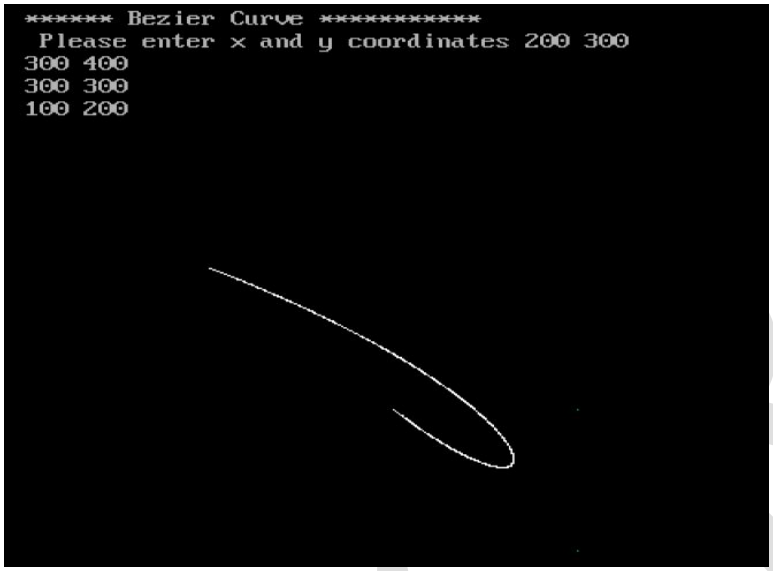
To Draw a Bezier Curve

Source Code

```
#include<graphics.h>
#include<math.h>
#include<conio.h>
#include<stdio.h>
void main()
{
int x[4],y[4],i;
double put_x,put_y,t;
int gr=DETECT,gm;
initgraph(&gr,&gm,"C:\\TURBOC3\\BGI");
printf("\n****** Bezier Curve **********");
printf("\n Please enter x and y coordinates ");
for(i=0;i<4;i++)
{
scanf("%d%d",&x[i],&y[i]);
putpixel(x[i],y[i],3);              // Control Points
}

for(t=0.0;t<=1.0;t=t+0.001)          // t always lies between 0 and 1
{
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]; //
Formula to draw curve
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, WHITE);          // putting pixel
```

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

OUTPUT:



Viva Question:

1. What is Bezier Curve?
2. What are the types of Bezier curve?
3. List out the properties of Bezier curve?
4. What is cubic Bezier curve?
5. What is quadratic Bezier curve?
6. How closed Bezier curve can be generated?
7. What is contral points?
8. What is parametric curve?
9. What is Curve?
10. What are the types of curves?