Swami Keshvanand Institute of Technology Management &
Gramothan, Jaipur

Lab Manual

(5CS4-21) Computer Graphics &Multimedia Lab

Semester: V

Branch: Computer Science & Engineering

Session 2021-22

Lab Instructors:

Manish Bhardwaj (Assistant Professor)

Harpreet Singh Gill (Associate Professor)

Rashmi Dadhich (Assistant Professor)

Department of Computer Science & Engineering

Swami Keshvanand Institute of Technology Management & Gramothan,

Ramnagaria, Jaipur-302017

TABLE OF CONTENTS

LAB RULES

Responsibilities of Users

Users are expected to follow some fairly obvious rules of conduct:

Always:

- Enter the lab on time and leave at proper time.
- Wait for the previous class to leave before the next class enters.
- Keep the bag outside in the respective racks.
- Utilize lab hours in the corresponding.
- Turn off the machine before leaving the lab unless a member of lab staff has specifically told you not to do so.
- Leave the labs at least as nice as you found them.
- If you notice a problem with a piece of equipment (e.g. a computer doesn't respond) or the room in general (e.g. cooling, heating, lighting) please report it to lab staff immediately. Do not attempt to fix the problem yourself.

Never:

- Don't abuse the equipment.
- Do not adjust the heat or air conditioners. If you feel the temperature is not properly set, inform lab staff; we will attempt to maintain a balance that is healthy for people and machines.
- Do not attempt to reboot a computer. Report problems to lab staff.
- Do not remove or modify any software or file without permission.

- Do not remove printers and machines from the network without being explicitly told to do so by lab staff.
- Don't monopolize equipment. If you're going to be away from your machine for more than 10 or 15 minutes, log out before leaving. This is both for the security of your account, and to ensure that others are able to use the lab resources while you are not.
- Don’t use internet, internet chat of any kind in your regular lab schedule.
- Do not download or upload of MP3, JPG or MPEG files.
- No games are allowed in the lab sessions.
- No hardware including USB drives can be connected or disconnected in the labs without prior permission of the lab in-charge.
- No food or drink is allowed in the lab or near any of the equipment. Aside from the fact that it leaves a mess and attracts pests, spilling anything on a keyboard or other piece of computer equipment could cause permanent, irreparable, and costly damage. (and in fact *has*) If you need to eat or drink, take a break and do so in the canteen.
- Don’t bring any external material in the lab, except your lab record, copy and books.
- Don’t bring the mobile phones in the lab. If necessary then keep them in silence mode.
- Please be considerate of those around you, especially in terms of noise level. While labs are a natural place for conversations of all types, kindly keep the volume turned down.

If you are having problems or questions, please go to either the faculty, lab in-charge or the lab supporting staff. They will help you. We need your full support and cooperation for smooth functioning of the lab.

INSTRUCTIONS

Before entering in the lab

All the students are supposed to prepare the theory regarding the next experiment.

Students are supposed to bring the practical file and the lab copy.

Previous programs should be written in the practical file.

All the students must follow the instructions, failing which he/she may not be allowed in the lab.

While working in the lab

Adhere to experimental schedule as instructed by the lab in-charge.

Get the previously executed program signed by the instructor.

Get the output of the current program checked by the instructor in the lab copy.

Each student should work on his/her assigned computer at each turn of the lab.

Take responsibility of valuable accessories.

Concentrate on the assigned practical and do not play games

If anyone caught red handed carrying any equipment of the lab, then he/she will have to face serious consequences.

SYLLABUS

RAJASTHAN TECHNICAL UNIVERSITY, KOTA

Syllabus

111 Year-V Semester: B.Tech. Computer Science and Engineering

5CS4-21: Computer Graphics & Multimedia Lab

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

	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

Introduction to the subject

Graphics is one of the most important elements of information. Computer graphics is the study of computer hardware that deals with storage, generation and display of all kinds of images in computer and the computing technologies to logically develop and manipulate those images. The term computer graphics has been used in a broad sense to describe "almost everything on computers that may be images, text or sound etc". Typically, the term computer graphics refers to several different things like:

- The representation and manipulation of image data by a computer.
- The various technologies used to create and manipulate images.
- The images so produced.

Interactive computer graphics become an effective tool for presentation of information in diverse fields such as **Science, engineering, medicine, business, industry, art, entertainment, advertising, education and training.** There is virtually no field in which graphical displays cannot be used to some advantage.

"One picture is worth of thousand words" can be modified in this computer era into **"One picture is worth of many kilobytes of data"**. So interfaces empowered with graphics enhance the communication between the computer and its users,

Multimedia refers to a mixture of interactive media or data type, predominantly text, graphics, audio & video simultaneously delivered by computer. Computer Graphics has revolutionized almost every computer-based application in science and technology.

List of Experiments beyond the syllabus

1. To draw a pixel on the screen
2. To draw different objects in different colors and dividing screen into four equal parts
3. To draw a hut on screen
4. To draw a cycle
5. To draw a moving cycle from left to right and appearing again from left end
6. To draw a ball bouncing in up and down direction
7. Drawing Smiley and moving its eyes, lips etc.
8. Creating animations
9. Using Keyboard keys for input

Basic Structure of A Computer Graphics Program

```
#include<graphics.h>
#include<conio.h>
void main()
{
   intgd=DETECT,gm;
   initgraph (&gd,&gm,"c:\\tc\\bgi");
   ……..Code to draw objects….
   getch();
}
```

Experiments

Experiment No.: 1

Name of Experiment: **Pre Defined Patterns (Line, Circle, Ellipse)**

Aim: To produce a single pixel and pre specified patterns on screen. (Same to all groups)

Algorithm: (Step wise):

1. Draw a rectangle using predefined functions from (0, 0) to (getmaxx(), getmaxy())
2. Draw two lines which partitioned the above rectangle into 4 quadrants.
3. Find the centre of each quadrant, And glow them with different colors.
4. Draw the following predefined patterns using colors as follows
5. Draw a circle in 1st quadrant
6. Draw a square in 2nd quadrant
7. Draw a ellipse in 3rd quadrant
8. Draw a triangle in 4th quadrant

Program:

```
#include <graphics.h>
#include <stdio.h>
#include<conio.h>
 void main()
{
   int gd = DETECT, gm, color;
   initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
   putpixel(85, 35, GREEN);
   putpixel(30, 40, RED);
   putpixel(115, 50, YELLOW);
line(135, 50,200,400);
circle(400,500,50);
rectangle(200,600,450,650);
ellipse(130,120,0,360,90,50);
getch();
closegraph();
}
```

Results for different data sets: (At least three)

Questions:

1. What is the function for drawing a line?
2. How to find the centre of 3rd quadrant?
3. How to get the resolution of your screen?
4. What is the use of initgraph() function.
5. What is the use of closegraph() function?
6. Why restorecrtmode() function is used?
7. How to change the color of your background screen?
8. Name the arguments that are to be passed to ellipse function?
9. How to get the coordinates of the bottom-right pixel on your screen?
10. Name the function used to put a pixel on your screen.

Experiment No.: 2

Name of Experiment**: To Plot a point (pixel) on the screen.**

Aim: **Draw A Pixel**

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

void main()
{
   int gd = DETECT, gm, color;
   initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
   putpixel(85, 35, GREEN);
   putpixel(30, 40, RED);
   putpixel(115, 50, YELLOW);
line(135, 50,200,400);
circle(400,500,50);
rectangle(200,600,450,650);
ellipse(130,120,0,360,90,50);
getch();
closegraph();
}
```

Experiment No.: 3

Name of Experiment**: To draw a straight line using DDA Algorithm.**

Aim: **To implement Digital Differential Analyzer (DDA) Algorithm.**

SHG1 & 5:Draw a line having (X1<X2) using DDA Algorithm having slope < than 1.

SHG2 & 6:Draw a line having (X1<X2) using DDA Algorithm having slope > than 1.

SHG3: Draw a line having (X1>X2) using DDA Algorithm having slope < than 1.

SHG4: Draw a line having (X1>X2) using DDA Algorithm having slope > than 1.

Algorithm: (Step wise):

1. Accept two end points as (x1, y1) and (x2, y2)
2. Find dx = x2 – x1 and dy = y2 – y1 and m = abs (dy/dx).
3. The deference with the greater magnitude determines the value of the parameter size.
   If abs (dx) > abs (dy) then size = abs(dx).
   Otherwise the size = abs (dy).
4. Start from the pixel (x1, y1) and determine increment or decrement which is needed to generate the next pixel at each step.
5. loop the following process for size times
   a. if (x1<x2) (line is to be drawn from left to right)
      if (m <1) then Xinc = 1 and Yinc = m
      else Xinc = 1/m and Yinc = 1
   b. else (Line is to be drawn from right to left)
      if (m >1) then Xinc = -1 and Yinc = -m
      else Xinc = -1/m and Yinc = -1

Test Data: (x1, y1) = (2, 7) and (x2, y2) = (15, 10)

Results for different data sets:

dx = 13 and dy = 3

size = 13

m = abs (3/13) = 0.2

so Xinc = 1 and Yinc = m = 0.2

Step	1	2	3	4	5	6	7	8	9	10	11	12	13
X	3	4	5	6	7	8	9	10	11	12	13	14	15
Y	7.2	7.4	7.6	7.8	8	8.2	8.4	8.6	8.8	9	9.2	9.4	9.6
(x,y)	(3,7)	(4,7)	(5,8)	(6,8)	(7,8)	(8,8)	(9,8)	(10,9)	(11,9)	(12,9)	(13,9)	(14,9)	(15,10)

rogram:

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
void main()
{
   intgd = DETECT ,gm, i;
   float x, y,dx,dy,steps;
   int x0, x1, y0, y1;
   initgraph(&gd, &gm, "C:\\TC\\BGI");
   setbkcolor(WHITE);
   x0 = 100 , y0 = 200, x1 = 500, y1 = 300;
   dx = (float)(x1 - x0);
   dy = (float)(y1 - y0);
   if(dx>=dy)
        {
     steps = dx;
   }
   else
        {
     steps = dy;
   }
   dx = dx/steps;
   dy = dy/steps;
   x = x0;
```

```
    y = y0;
    i = 1;
    while(i<= steps)
    {
       putpixel(x, y, RED);
       x += dx;
       y += dy;
       i=i+1;
    }
    getch();
    closegraph();
}
```

iva Questions: (minimum 10)

1. What does DDA stands for?
2. How do you find the slope of a line?
3. What is the difference between floor() and ceil() functions?
4. In which header file abs () function is found?
5. How we can round off any value?
6. What is the equation of line having end points (x1,y1) and (x2,y2)?

Experiment No.: 3.1

Name of Experiment**: To Draw a straight line using Bresenham's Line Drawing Algorithm**

Aim: **SHG 1, 3, 5:** Line of slope $|m| < 1$

SHG 2, 4, 6: Line of slope $|m| >= 1$

Algorithm: (Step wise):

1. Insert two I/P line end points and store the left end point in (X_o, Y_o).
2. Plot the first point. (Load the point into the frame buffer)
3. Calculate constants Δx, Δy, $2\Delta x$, $2\Delta y$ and obtain the starting value for the decision parameter as

   $P_o = 2\Delta y - \Delta x$
4. At each X_k along the line, starting at K=0 perform the following test:
   a. if $P_k<0$ the next point to plot is $(X_K + 1, Y_k)$ and

      $P_{k+1} = P_k + 2\Delta y$

   b. Otherwise the next point to plot is $(X_K + 1, Y_k+1)$ and

      $P_{k+1} = P_k + 2\Delta y - 2\Delta x$
5. Repeat the step 4 for Δx times.

Test Data: (x1, y1) = (20, 10) and (x2, y2) = (30, 18)

Results for different data sets:

$\Delta x = 10$ and $\Delta y = 8$

$P_o = 2\Delta y - \Delta x = 6$

Results for different data sets:

K	0	1	2	3	4	5	6	7	8	9
Pk	6	2	-2	14	10	6	2	-2	14	10
X,Y	21,11	22,12	23,12	24,13	25,14	26,15	27,16	28,16	29,17	30,18

Program:

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

```
    scanf("%d%d", &x0, &y0);
    printf("Enter co-ordinates of second point: ");
    scanf("%d%d", &x1, &y1);
    drawline(x0, y0, x1, y1);
    return 0;
}
```

Viva Questions: (minimum 10)

1. What is decision parameter in this algorithm?
2. What is the value of initial decision parameter?
3. What is the value of decision parameter at Kth step?
4. What is the value of decision parameter at (K+1)th if Pk is >0; step?
5. What is the value of decision parameter at (K+1)th if Pk is >0; step?
6. What is the value of decision parameter at (K+1)th if Pk is =0; step
7. What is the equation of line having end points (x1,y1) and (x2,y2)?
8. What is the equation of line in slope-intercept form?

Experiment No.: 04

Name of Experiment: Implementation of Mid Point Circle Drawing Algorithm

Aim:

1. Draw an arc between two points
2. Draw a circle

Algorithm:

1. Input radius r and circle centre (Xc, Yc) and obtain the first point on the circumference of the circle centered at (0,0).

   X = 0 and y = r.

2. Calculate initial decision parameter

   Po = 5/4 – r (1 – r for integer calculation)

3. At each X k, starting at K=0, perform the following test:

   a. if $P_k<0$ the next point to plot is $(X_K + 1, Y_k)$ and

   $P_{k+1} = P_k + 2 X_{K+1} + 1$

   b. Otherwise the next point to plot is $(X_K + 1, Y_k - 1)$ and

   $P_{k+1} = P_k + 2 X_{K+1} - 2 Y_{K+1} + 1$

4. Determine symmetry points in the other seven octants.
5. Move each calculated position (X, Y) onto the circular path centered at (Xc, Yc) and plot the coordinate values.

   X = Xc + X, Y = Yc + Y

6. Repeat steps 3 to 5 until X>=Y.

Test Data: Yc = 0, Yc = 0 and radius = 10

X0 = 0 and Y0 = 10

P0 = 1 – 10 = -9

Results for different data sets:

K	0	1	2	3	4	5	6
Pk	-9	-6	-1	6	-3	8	5
X,Y	1, 10	2, 10	3, 10	4, 9	5, 9	6, 8	7, 7

Program

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

scanf("%d",&xc);
scanf("%d",&yc);
d=1.25-r;
y=r;
do
{
if(d<0.0)
{
x=x+1;
d=d+2*x+1;
}
else
{
x=x+1;
y=y-1;
d=d+2*x-2*y+1;
```

```
}
putpixel(xc+x,yc+y,5);
putpixel(xc-y,yc-x,5);
putpixel(xc+y,yc-x,5);
putpixel(xc-y,yc+x,5);
putpixel(xc+y,yc+x,5);
putpixel(xc-x,yc-y,5);
putpixel(xc+x,yc-y,5);
putpixel(xc-x,yc+y,5);
}
while(x<y);
getch();
}
```

Viva Questions:

1. What is equation of circle?
2. How you transfer the circle to its original canter?
3. What is 8 way symmetry property?
4. What is the slope of circle at the point on the line Y=X.
5. What is the slope of circle at the point on the line Y=0.
6. What is the value of initial decision parameter?
7. What is the value of decision parameter at (K+1)th if Pk is >0; step?
8. What is the value of decision parameter at (K+1)th if Pk is <0; step?
9. What is the value of decision parameter at (K+1)th if Pk is =0; step?
10. What is the decision parameter in this algorithm?

Experiment No.: 05

Name of Experiment: Implementation of Mid Point Ellipse Drawing Algorithm

Aim:

1. Draw an arc between two points
2. Draw an ellipse

Algorithm:

1. Input radius r_x & r_y and ellipse center (Xc, Yc) and obtain the first point on the circumference of the ellipse centered at (0, 0).

   $X_0 = 0$ and $y_0 = r_x$.

2. Calculate initial decision parameter in the region 1

   $P1o = r_y^2 - r_x^2 r_y + (1/4) r_x^2$

3. At each X_k, starting at K=0, perform the following test:

   a. if $P1_k < 0$ the next point along the ellipse centered at (0. 0) is (X_{K+1}, Y_k) and

   $$P1_{k+1} = P1_k + 2 r_y^2 X_{K+1} + r_y^2$$

   b. Otherwise the next point along the ellipse is $(X_K + 1, Y_k - 1)$ and

   $$P1_{k+1} = P1_k + 2 r_y^2 X_{K+1} - 2 r_x^2 Y_{K+1} + r_y^2$$

   With

   $$2 r_y^2 X_{K+1} = 2 r_y^2 X_K + 2 r_y^2 \text{ and } 2 r_x^2 Y_{K+1} = 2 r_x^2 Y_K - 2 r_x^2$$

   And continue until $2 r_y^2 X >= 2 r_x^2 Y$.

4. Calculate initial decision parameter in the region 2 using the last point (Xo, Yo) calculated in region 1 as

   $$P2o = r_y^2 (X_0 + 1/2)^2 - r_x^2 (Y_0 - 1)^2 + r_x^2 r_y^2$$

5. At each Y_k in region 2, starting at K=0, perform the following test:

   a. if $P2_k < 0$ the next point along the ellipse centered at (0. 0) is $(X_K, Y_k - 1)$ and

   $$P2_{k+1} = P2_k - 2 r_x^2 Y_{K+1} + r_x^2$$

   b. Otherwise the next point along the ellipse is $(X_K + 1, Y_k - 1)$ and

   $$P2_{k+1} = P2_k + 2 r_y^2 X_{K+1} - 2 r_x^2 Y_{K+1} + r_x^2$$

   Using the same incremental calculations for x and y as in region 1.

6. Determine symmetry points in the other three octants.

5. Move each calculated position (X, Y) onto the elliptical path centered at (Xc, Yc) and plot the coordinate values.

   $X = Xc + X, \quad Y = Yc + Y$

6. Repeat steps for region 2 until y <=0

Test Data: $r_x = 8$ & $r_y = 6$

$2r_y^2 x = 0$

$2r_x^2 y = 2r_x^2 r_y$

Results for different data sets:

For region 1: P1o = -332

K	0	1	2	3	4	5	6
P1k	-332	-224	-44	208	-108	288	244
(X_{K+1}, Y_{k+1})	(1, 6)	(2, 6)	(3, 6)	(4, 5)	(5, 5)	(6, 46)	(7, 3)
$2 r_y^2 X_{K+1}$	72	144	216	288	360	432	504
$2 r_x^2 Y_{K+1}$	768	768	768	640	640	512	384

Since $2 r_y^2 X >= 2 r_x^2 Y$ ao we move out of the region 1.

For region 1: P2o = -151

K	0	1	2
P2k	-151	233	745
(X_{K+1}, Y_{k+1})	(8, 2)	(8, 1)	(8, 0)
$2 r_y^2 X_{K+1}$	576	576	-
$2 r_x^2 Y_{K+1}$	256	128	-

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

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

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

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

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

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

     xrad2 = xradius * xradius;
```

```
yrad2 = yradius * yradius;

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

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

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

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

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

```
        delay(100);
    }

    delay(500);

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

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

        if (dp > 0) {
            dp = dp + xrad2 - dpy;
        } else {
            x = x + 1;
            dpx = dpx + twoyrad2;
            dp = dp + xrad2 - dpy + dpx;
        }
        /* plotting points at x-axis(left/right) */
        putpixel(midx + x, midy + y, WHITE);
        putpixel(midx - x, midy + y, WHITE);
        putpixel(midx + x, midy - y, WHITE);
        putpixel(midx - x, midy - y, WHITE);
        delay(100);
    }
    getch();
    /* deallocate memory allocated for graphic screen */
    closegraph();
    return 0;
}
```

Viva Questions:

1. What is equation of ellipse?
2. How you transfer the ellipse to its original canter?
3. How we can draw the same points in other 3 Quadrants?
4. What is the condition to move out of the region 1?
5. What is the value of initial decision parameter for region 1?
6. What is the value of initial decision parameter for region 2?
7. What is the value of decision parameter for R1 at (K+1)th if Pk is >0; step?
8. What is the value of decision parameter for R2at (K+1)th if Pk is <0; step?
9. What is the value of decision parameter for R2at (K+1)th if Pk is =0; step?
10. What is Aspect Ratio?

Experiment-6

Aim: Two dimensional transformations- Translation, Rotation, Scaling

Translation : It is the straight line movement of an object from one position to another is called Translation. Here the object is positioned from one coordinate location to another.

Translation of point: To translate a point from coordinate position (x, y) to another (x1 y1), we add algebraically the translation distances Tx and Ty to original coordinate.

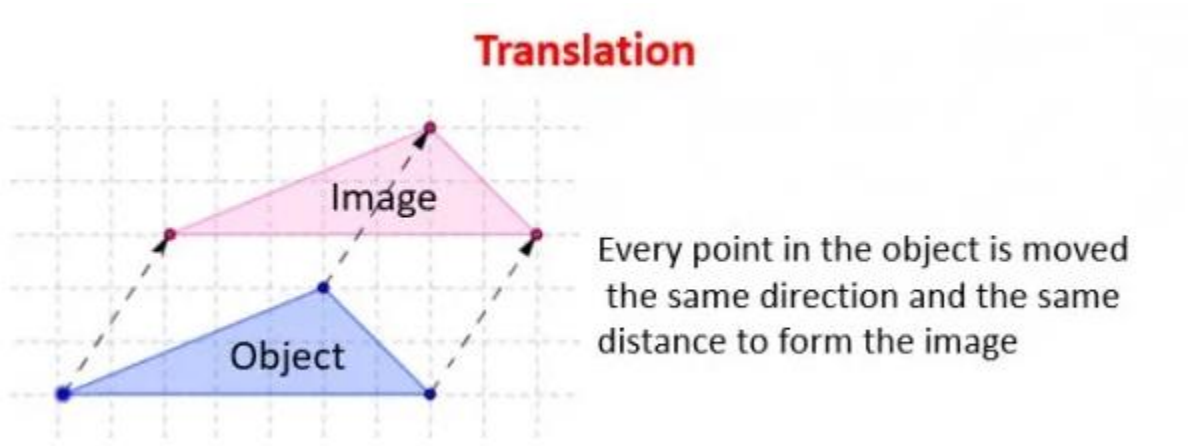
$x1=x+Tx$

$y1=y+Ty$

The translation pair (Tx,Ty) is called as translation vectors.

Translation is a movement of objects without deformation. Every position or point is translated by the same amount. When the straight line is translated, then it will be drawn using endpoints.

Example :



Top of Form

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
int gd=DETECT , gm ;
int x3,y3,x1,y1,x2,y2,tx,ty;
clrscr();
initgraph(&gd,&gm , "c:\\turboc3\\bgi");
printf("\n enter first coordinate : ");
scanf("%d%d",&x1,&y1);
printf("\n enter second coordinate: ");
scanf("%d%d",&x2,&y2);
printf("\n enter third coordinate: ");
scanf("%d%d",&x3,&y3);
line(x1,y1,x2,y2);
line(x2,y2,x3,y3);
line(x3,y3,x1,y1);
printf("\n enter translation vectors : ");
scanf("%d%d",&tx,&ty);
setcolor(BLUE);
line(x1+tx,y1+ty,x2+tx,y2+ty);
line(x2+tx,y2+ty,x3+tx,y3+ty);
line(x3+tx,y3+ty,x1+tx,y1+ty);
getch();
closegraph();
}
```

Rotation : It is a process of changing the angle of the object. Rotation can be clockwise or anticlockwise. For rotation, we have to specify the angle of rotation and rotation point. Rotation point is also called a pivot point. It is print about which object is rotated.

Types of Rotation

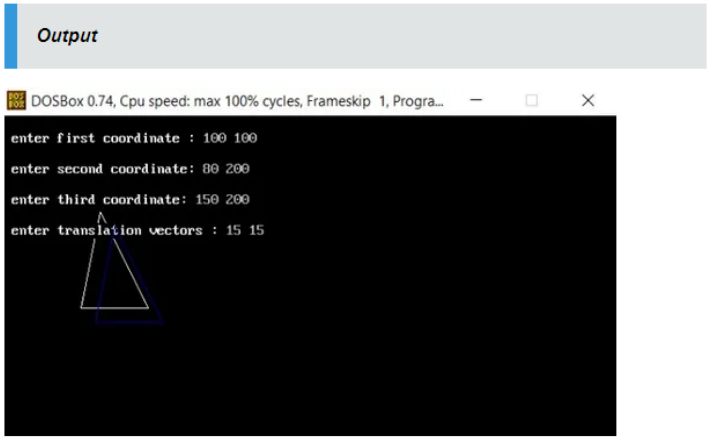
Anticlockwise - The positive value of the rotation angle rotates an object in a anti-clockwise direction.

Counterclockwise - The negative value of the pivot point (rotation angle) rotates an object in a clockwise direction.

Program

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void main()
{
   int gd=0,gm,x1,y1,x2,y2,x3,y3;
   double s,c, angle;
   initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
   setcolor(RED);
   printf("Enter coordinates of triangle: ");
   scanf("%d%d%d%d%d%d",&x1,&y1,&x2,&y2, &x3, &y3);
   setbkcolor(WHITE);
   cleardevice();
   line(x1,y1,x2,y2);
   line(x2,y2, x3,y3);
   line(x3, y3, x1, y1);
   getch();
   setbkcolor(BLACK);  //setting background color
   printf("Enter rotation angle: ");
   scanf("%lf", &angle);
   setbkcolor(WHITE);  //setting background color
  //M_PI : Pi, the ratio of a circle's circumference to its diameter.
   c = cos(angle *M_PI/180);
   s = sin(angle *M_PI/180);
   x1 = floor(x1 * c + y1 * s);
   y1 = floor(-x1 * s + y1 * c);
   x2 = floor(x2 * c + y2 * s);
   y2 = floor(-x2 * s + y2 * c);
   x3 = floor(x3 * c + y3 * s);
   y3 = floor(-x3 * s + y3 * c);
   cleardevice();
   line(x1, y1 ,x2, y2);
   line(x2,y2, x3,y3);
   line(x3, y3, x1, y1);
   getch();
   closegraph();
}
```

Output



Rotation : It is a process of changing the angle of the object. Rotation can be clockwise or anticlockwise. For rotation, we have to specify the angle of rotation and rotation point. Rotation point is also called a pivot point. It is print about which object is rotated.

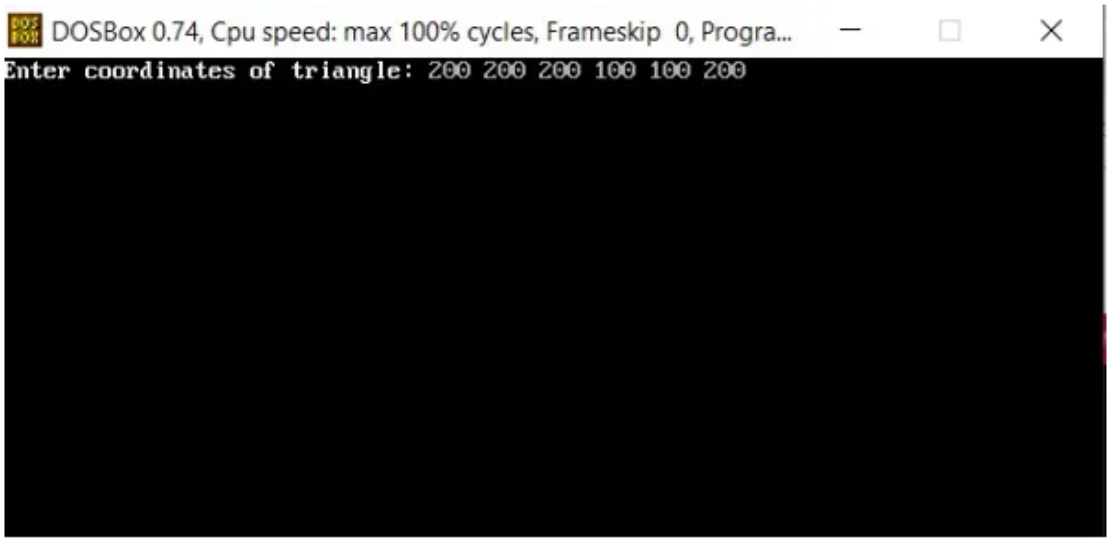
Types of Rotation

Anticlockwise - The positive value of the rotation angle rotates an object in a anti-clockwise direction.

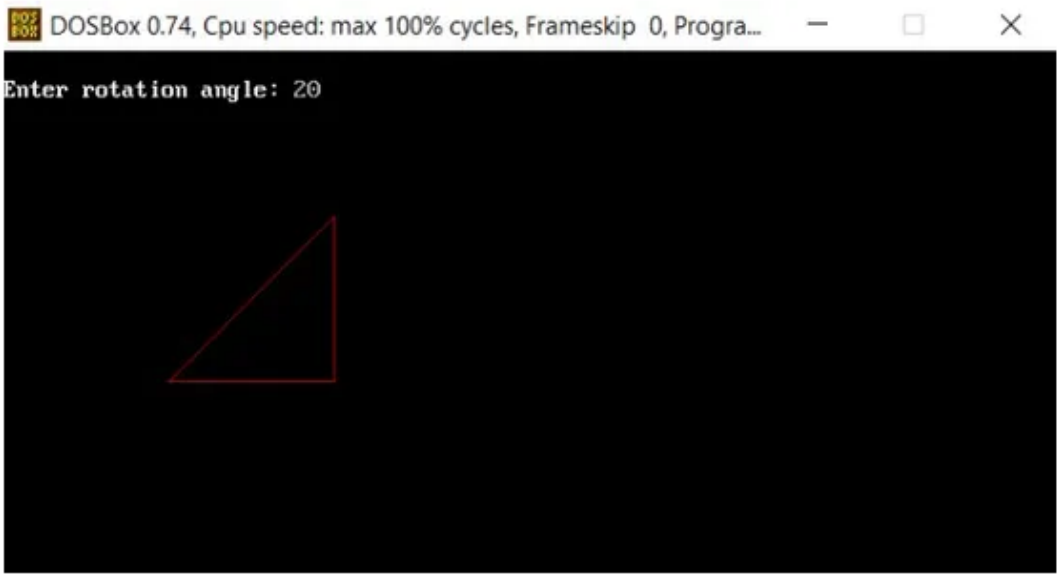
Counterclockwise - The negative value of the pivot point (rotation angle) rotates an object in a clockwise direction.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void main()
{
   int gd=0,gm,x1,y1,x2,y2,x3,y3;
   double s,c, angle;
   initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
   setcolor(RED);
   printf("Enter coordinates of triangle: ");
```

```
    scanf("%d%d%d%d%d%d",&x1,&y1,&x2,&y2, &x3, &y3);
    setbkcolor(WHITE);
    cleardevice();
    line(x1,y1,x2,y2);
    line(x2,y2, x3,y3);
    line(x3, y3, x1, y1);
    getch();
    setbkcolor(BLACK);  //setting background color
    printf("Enter rotation angle: ");
    scanf("%lf", &angle);
    setbkcolor(WHITE);  //setting background color
   //M_PI : Pi, the ratio of a circle's circumference to its diameter.
    c = cos(angle *M_PI/180);
    s = sin(angle *M_PI/180);
    x1 = floor(x1 * c + y1 * s);
    y1 = floor(-x1 * s + y1 * c);
    x2 = floor(x2 * c + y2 * s);
    y2 = floor(-x2 * s + y2 * c);
    x3 = floor(x3 * c + y3 * s);
    y3 = floor(-x3 * s + y3 * c);
    cleardevice();
    line(x1, y1 ,x2, y2);
    line(x2,y2, x3,y3);
    line(x3, y3, x1, y1);
    getch();
    closegraph();
}
```

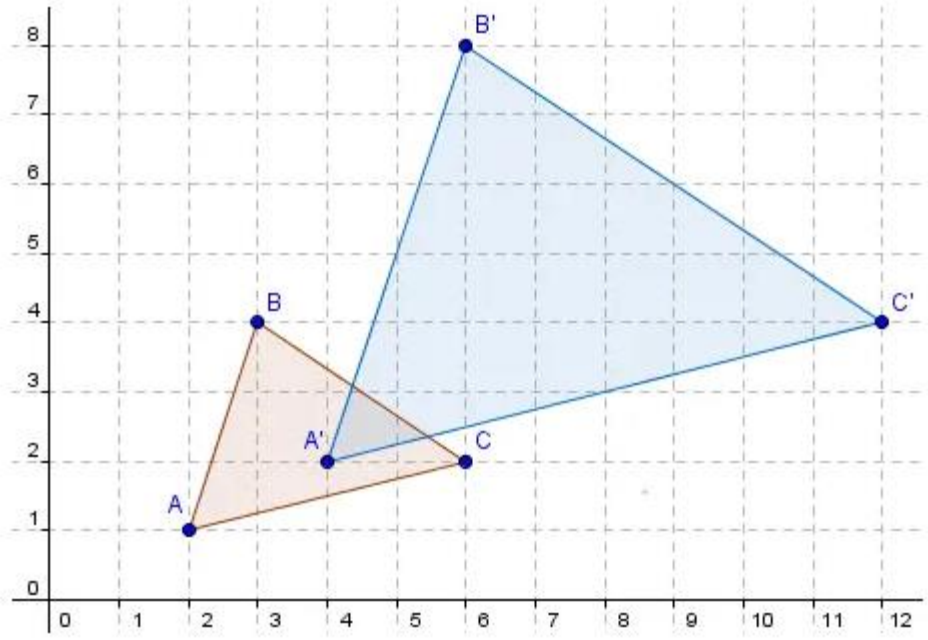

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...
Enter coordinates of triangle: 200 200 200 100 100 200


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


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...
Enter rotation angle: 20

Scaling of triangle

Scaling : It is used to change the size of objects. The change is done using scaling factors. There are two scaling factors, i.e. Sx in x direction Sy in y-direction. If the original position is x and y. Scaling factors are Sx and Sy



Program

Matrix Multiplication to find new Coordinates.

s[][] is scaling matrix.

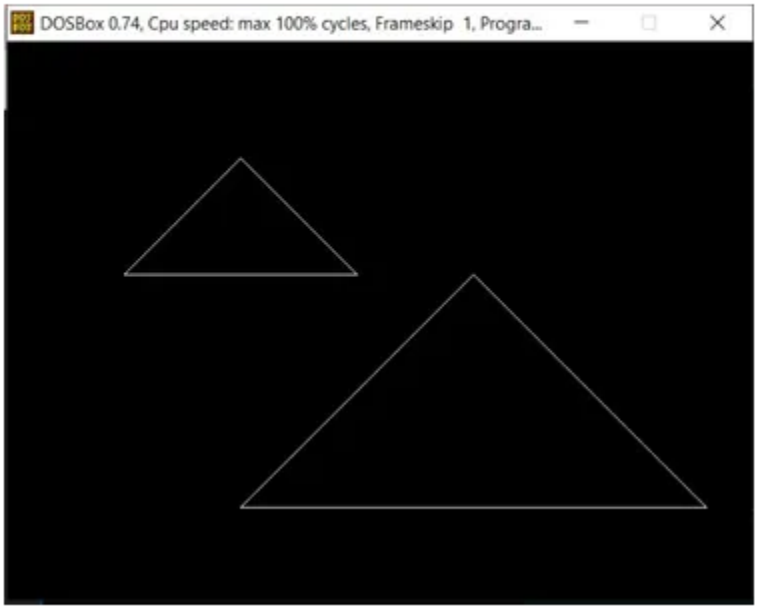
p[][] is to store points that needs to be scaled.

p[0][0] is x coordinate of point.

p[1][0] is y coordinate of given point.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void fC(int s[][2], int p[][1])
{
    int temp[2][1] = { 0 };
    for (int i = 0; i < 2; i++)
        for (int j = 0; j < 1; j++)
            for (int k = 0; k < 2; k++)
                temp[i][j] += (s[i][k] * p[k][j]);
   p[0][0] = temp[0][0];
   p[1][0] = temp[1][0];
}
void scale(int x[], int y[], int sx, int sy)
{
   // Triangle before Scaling
   line(x[0], y[0], x[1], y[1]);
   line(x[1], y[1], x[2], y[2]);
   line(x[2], y[2], x[0], y[0]);
   // Initializing the Scaling Matrix.
   int s[2][2] = { sx, 0, 0, sy };
   int p[2][1];
   for (int i = 0; i < 3; i++)
   {
      p[0][0] = x[i];
      p[1][0] = y[i];
      fC(s, p);
      x[i] = p[0][0];
      y[i] = p[1][0];
   }
   line(x[0], y[0], x[1], y[1]);
   line(x[1], y[1], x[2], y[2]);
   line(x[2], y[2], x[0], y[0]);
}
void main()
{
   int x[] = { 100, 200, 300 };
   int y[] = { 200, 100, 200 };
   int sx = 2, sy = 2;
   int gd, gm;
   detectgraph(&gd, &gm);
   initgraph(&gd, &gm," ");
   scale(x, y, sx,sy);
   getch();
}
```

Output



Program:

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

// Matrix Multiplication to find new Coordinates.
// s[][] is scaling matrix. p[][] is to store
// points that needs to be scaled.
// p[0][0] is x coordinate of point.
// p[1][0] is y coordinate of given point.
void findNewCoordinate(int s[][2], int p[][1])
{
        int temp[2][1] = { 0 };

        for (int i = 0; i < 2; i++)
                for (int j = 0; j < 1; j++)
```

```
                    for (int k = 0; k < 2; k++)
                        temp[i][j] += (s[i][k] * p[k][j]);

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

// Scaling the Polygon
void scale(int x[], int y[], int sx, int sy)
{
        // Triangle before Scaling
        line(x[0], y[0], x[1], y[1]);
        line(x[1], y[1], x[2], y[2]);
        line(x[2], y[2], x[0], y[0]);

        // Initializing the Scaling Matrix.
        int s[2][2] = { sx, 0, 0, sy };
        int p[2][1];

        // Scaling the triangle
        for (int i = 0; i < 3; i++)
        {
                p[0][0] = x[i];
                p[1][0] = y[i];

                findNewCoordinate(s, p);

                x[i] = p[0][0];
                y[i] = p[1][0];
        }
```

```
        // Triangle after Scaling
        line(x[0], y[0], x[1], y[1]);
        line(x[1], y[1], x[2], y[2]);
        line(x[2], y[2], x[0], y[0]);
}

// Driven Program
int main()
{
        int x[] = { 100, 200, 300 };
        int y[] = { 200, 100, 200 };
        int sx = 2, sy = 2;

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

        scale(x, y, sx,sy);
        getch();

        return 0;
}
```

Experiment-7

Aim:Composite 2D Transformations implementation of two dimensional composite transformations

A transformation is any operation on a point in space (x, y) that maps the point's coordinates into a new set of coordinates (x1, y1).The Two Dimensional Composite transformation represent a sequence of transformations as a single matrix which has the order of operations as Translation, Rotation, Scaling, Shearing, Reflection

Code:

```
#include <graphics.h> /* include the necessary header files*/
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
int xa,xb,xc,ya,yb,yc,y1a,y1b,y1c,x1a,x1b,x1c,x2a,x2b,x2c,y2a,y2b,y2c;
int x3a,x3b,x3c,y3a,y3b,y3c,x4a,x4b,x4c,y4a,y4b,y4c,x5a,x5b,x5c,y5a,y5b,y5c;
int tx,shx,t,ch,shy;
float ang,theta,sx,sy;
int main(void)
{
int gdriver = DETECT, gmode, errorcode;
initgraph(&gdriver, &gmode,"C:\\TC\\BGI"); /* request for auto detection*/
printf("\n\t\t\t 2D Composite Transformations");
printf("\n\n Enter all coordinates values :");
scanf("%d %d %d %d %d %d",&xa,&ya,&xb,&yb,&xc,&yc);
printf("\n\n The original Image"); /* get the coordinates for the original image*/
line(xa,ya,xb,yb); /* draw the original image*/
line(xb,yb,xc,yc);
line(xc,yc,xa,ya);
printf("\n\n Enter the value tranlsation factor :"); /* get the translation factor*/
```

```
scanf("%d",&tx);
printf("\n\n After Translation ");
x1a=xa+tx;
x1b=xb+tx;
x1c=xc+tx;
y1a=ya;
y1b=yb;
y1c=yc;
line(x1a,y1a,x1b,y1b); /* image after translation*/
line(x1b,y1b,x1c,y1c);
line(x1c,y1c,x1a,y1a);
delay(1);
printf("\n\n Next Operation is Rotation");
printf("\n\n Enter the rotation angle :"); /* get the angle of rotation*/
scanf("%f",&ang);
theta=((ang*3.14)/180); /* convert the angle*/
x2a=x1a*cos(theta)-y1a*sin(theta);
y2a=x1a*sin(theta)+y1a*cos(theta);
x2b=x1b*cos(theta)-y1b*sin(theta);
y2b=x1b*sin(theta)+y1b*cos(theta);
x2c=x1c*cos(theta)-y1c*sin(theta);
y2c=x1c*sin(theta)+y1c*cos(theta);
printf("\n\n After Rotation "); /* the rotated object*/
line(x2a,y2a,x2b,y2b);
line(x2b,y2b,x2c,y2c);
line(x2c,y2c,x2a,y2a);
delay(1);
printf("\n\n Next Operation is Scaling"); /* get the scale factor*/
printf("\n\n Enter the Scale factor :");
scanf("%f %f",&sx,&sy);
x3a=x2a+sx; /* modify the objects coordinates based on the scale factor*/
```

```
y3a=y2a+sy;
x3b=x2b+sx;
y3b=y2b+sy;
x3c=x2c+sx;
y3c=y2c+sy;
printf("\n\n After Scaling ");
line(x3a,y3a,x3b,y3b);
line(x3b,y3b,x3c,y3c);
line(x3c,y3c,x3a,y3a);
delay(1);
printf("\n\n Next Operation is Shearing");
printf("\n\n Enter 1 for x-axis \n 2 for y-axis: "); /* get the choice of shearing in the x or y axis*/
scanf("%d",&ch);
if(ch==1) /* get the shear value*/
{
printf("\n\n Enter the x-SHEAR (^.^) Value: ");
scanf("%d",&shx);
}
else
{
printf("\n\n Enter the y-SHEAR (^.^) Value: ");
scanf("%d",&shy);
}
if(ch==1)
{
x3a=x3a+shx*y3a;
y4a=y3a;
x3b=x3a+shx*y3a;
y4b=y3b;
x3c=x3a+shx*y3a;
y4c=y3c;
```

```
}
else
{
x4a=x3a;
y3a=y3a+shy*x3a;
x4b=x3b;
y3b=y3b+shy*x3b;
x4c=x3c;
y3c=y3c+shy*x3c;
}
printf("\n\n After Shearing "); /* draw the final object after shearing*/
line(x3a,y3a,x3b,y3b);
line(x3b,y3b,x3c,y3c);
line(x3c,y3c,x3a,y3a);
delay(1);
printf("\n\n Next Operation is Reflection");
t=abs(y3a-y3c); /* calculate the value for reflection*/
x5a=x3a;
x5b=x3b;
x5c=x3c;
y5a=y3a+10+(2*t);
y5b=y3b+10;
y5c=y3c+10;
printf("\n\n After Reflection "); /* the final object after all the transformations*/
line(x5a,y5a,x5b,y5b);
line(x5b,y5b,x5c,y5c);
line(x5c,y5c,x5a,y5a);
getch();
closegraph();
return 0;
}
```

Experiment-8

Aim: To Implement Cohen Sutherland 2D line clipping and windowing.

This is one of the oldest and most popular line clipping algorithm. To speed up the process this algorithm performs initial tests that reduce number of intersections that must be calculated. It does so by using a 4 bit code called as region code or outcodes. These codes identify location of the end point of line. Each bit position indicates a direction, starting from the rightmost position of each bit indicates left, right, bottom, top respectively. Once we establish region codes for both the endpoints of a line we determine whether the endpoint is visible, partially visible or invisible with the help of ANDing of the region codes. There arises 3 cases which are explained in the algorithm below in step 4.

Algorithm: A.

Cohen – Sutherland Algorihtm.

1. Every end point of the line is assigned a 4 digit binary code. Called REGION CODE that identifies the location of the point w.r.t the boundaries of a clipping window.
2. First we test a given line to determine weather it lies completely inside the window.
3. If it does not, we try to determine weather it lies completely outside the window.
4. Finally if cannot identify a line as completely inside or outside the window.
   We must perform intersection calculation with one or more clipping boundaries.
5. Intersection point with a clipping boundary can be calculated using the slope intercept form of the line Y-Y1 = m (X- X1)

Algorithm B

1. Read 2 end points of line as p1(x1,y1) and p2(x2,y2)

2. Read 2 corner points of the clipping window (left-top and right-bottom) as (wx1,wy1) and (wx2,wy2)

3. Assign the region codes for 2 endpoints p1 and p2 using following steps:-

initialize code with 0000

Set bit 1 if x<wx1

Set bit 2 if x>wx2

Set bit 3 if y<wy2

Set bit 4 if y>wy1

4. Check for visibility of line

a. If region codes for both endpoints are zero then line is completely visible. Draw the line go to step 9.
b. If region codes for endpoints are not zero and logical ANDing of them is also nonzero then line is invisible. Discard the line and move to step 9.
c. If it does not satisfy 4.a and 4.b then line is partially visible.

5. Determine the intersecting edge of clipping window as follows:-

a. If region codes for both endpoints are nonzero find intersection points p1' and p2' with boundary edges.
b. If region codes for any one end point is non zero then find intersection point p1' or p2'.

6. Divide the line segments considering intersection points.

7. Reject line segment if any end point of line appears outside of any boundary.

8. Draw the clipped line segment.

9. Stop.

Program

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

typedef struct coordinate
{
int x,y;
char code[4];
}PT;

void drawwindow();
void drawline(PT p1,PT p2);
PT setcode(PT p);
int visibility(PT p1,PT p2);
PT resetendpt(PT p1,PT p2);

void main()
{
int gd=DETECT,v,gm;
PT p1,p2,p3,p4,ptemp;
printf("\nEnter x1 and y1\n");
scanf("%d %d",&p1.x,&p1.y);
printf("\nEnter x2 and y2\n");
scanf("%d %d",&p2.x,&p2.y);
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
drawwindow();
delay(500);
```

```
drawline(p1,p2);
delay(500);
cleardevice();
delay(500);
p1=setcode(p1);
p2=setcode(p2);
v=visibility(p1,p2);
delay(500);
switch(v)
{
case 0: drawwindow();
delay(500);
drawline(p1,p2);
break;
case 1: drawwindow();
delay(500);
break;
case 2: p3=resetendpt(p1,p2);
p4=resetendpt(p2,p1);
drawwindow();
delay(500);
drawline(p3,p4);
break;
}
delay(5000);
closegraph();
}

void drawwindow()
{
line(150,100,450,100);
```

```
line(450,100,450,350);
line(450,350,150,350);
line(150,350,150,100);
}

void drawline(PT p1,PT p2)
{
line(p1.x,p1.y,p2.x,p2.y);
}

PT setcode(PT p) //for setting the 4 bit code
{
PT ptemp;
if(p.y<100)
ptemp.code[0]='1'; //Top
else
ptemp.code[0]='0';
if(p.y>350)
ptemp.code[1]='1'; //Bottom
else
ptemp.code[1]='0';
if(p.x>450)
ptemp.code[2]='1'; //Right
else
ptemp.code[2]='0';
if(p.x<150)
ptemp.code[3]='1'; //Left
else
ptemp.code[3]='0';
ptemp.x=p.x;
ptemp.y=p.y;
```

```
return(ptemp);
}

int visibility(PT p1,PT p2)
{
int i,flag=0;
for(i=0;i<4;i++)
{
if((p1.code[i]!='0') || (p2.code[i]!='0'))
flag=1;
}
if(flag==0)
return(0);
for(i=0;i<4;i++)
{
if((p1.code[i]==p2.code[i]) && (p1.code[i]=='1'))
flag='0';
}
if(flag==0)
return(1);
return(2);
}

PT resetendpt(PT p1,PT p2)
{
PT temp;
int x,y,i;
float m,k;
if(p1.code[3]=='1')
x=150;
if(p1.code[2]=='1')
```

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

Experiment No.: 09

Name of Experiment: Polygon clipping by Sutherland Hodgeman algorithm

Aim: To Implement Polygon clipping by Sutherland Hodgeman algorithm

Algorithm A

Sutherland Hodgeman Algorithm

1. The polygon is originally defined by a list of vertices p1,p2,p3,….pn which implies the list of edges p1p2,p2p3,p3p4,….pn-1pn.
2. These edges are first clipped against the left edge of window to yield the intermediate polygon.
3. Then this intermediate polygon is clipped against the top edge. This yields second intermediate polygon.
4. The process is repeated until the polygon is clipped against all the window edges.
5. The output of the algorithm is a list of polygon vertices , all of which are on the visible side of a clipping window.

Algorithm: B

1. Read the coordinates of the vertices in X[i] and Y[i] array. i varies from 1 to n.Where n is the number of vertices.
2. Check each vertex against each boundary of the window for which side it lies. Itcan be done as below:

Lower left corner of the Window (Xwmin, Ywmin)

Upper right corner of the Window (Xwmax, Ywmax).

a) If X[i] < Xwmin then the vertex lies on the left side of the window. And X= Xwmin, go to Step3.

b) If X[i] > Xwmax then the vertex lies on the right side of the window. And X= Xwmax go to step3.

c) If Y[i] > Ywmin then the vertex lies on the bottom of the window. And Y= Ywmin go to step3.

d) If Y[i] > Ywmax then the vertex lies on the top of the window. And Y=Ywmax go to step3.

3. Calculate the intersection with the respective boundary using m and m1

(slope)as:

X[0]= X[n]

X[n+1]= X[1]

Y[0]= Y[n]

Y[n+1]= Y[1]

M =(Y[i] – Y[I-1])/ (X[i]- X[i-1])

M1= (Y[i] – Y[I+1])/ (X[i]- X[i +1])

a) Intersection with the respective boundaries (left and right) are Y1[i]= Y[i] + m(X- X[i])

Y2[i]= Y[i] + m1(X- X[i])

b) Intersection with the bottom and top boundaries are: X1[i]= X[i] + (Y- Y[i])/m

X2[I]= X[I] + (Y- Y[i])/m1

4. Draw the lines using these intersections.

Program

```
#include<stdio.h>
#include<conio.h>
struct poly
{
int coeff,expo;
};
struct poly p1[10],p2[10],p3[20];
int readpoly(struct poly[]);
int addpoly(struct poly[],struct poly[],int,int,struct poly[]);
void display(struct poly[],int terms);
void main()
{
clrscr();
int t1,t2,t3;
t1=readpoly(p1);
printf("\nfirst plynomial is:");
display(p1,t1);
t2=readpoly(p2);
printf("\nfsecond plynomial is:");
display(p2,t2);
t3=addpoly(p1,p2,t1,t2,p3);
printf("\nresultant polynomial:");
display(p3,t3);
printf("\n");
getch();
}
int readpoly(struct poly p[10])
{
int t1,i;
printf("\nenter no of terms in polynomial");
```

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

```
else
{ p3[k].coeff=p1[i].coeff+p2[i].coeff;
p3[k].expo=p1[i].expo;
i++;j++;k++;

}
}
while(i<t1)
{
p3[k].coeff=p1[i].coeff;
p3[k].expo=p1[i].expo;
i++;k++;
}
while(j<t2)
{
p3[k].coeff=p2[j].coeff;
p3[k].expo=p2[j].expo;
j++;k++;
}
return k;
}
void display(struct poly p[10],int term)
{
int k;
for (k=0;k<term-1;k++)
printf("%d(x^%d)+",p[k].coeff,p[k].expo);
printf("%d(x^%d)",p[term-1].coeff,p[term-1].expo);
}
```

Experiment No.: 10

Name of Experiment: Three dimensional transformations – Translation, Rotation, Scaling

Aim:

Translation

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
int maxx,maxy,midx,midy;
void axis()
{
getch();
cleardevice();
line(midx,0,midx,maxy);
line(0,midy,maxx,midy);
}
void main()
{
int x,y,z,o,x1,x2,y1,y2;
int gd=DETECT,gm;
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"c:\\tc\\bgi");
//setfillstyle(0,getmaxcolor());
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);
//axis();
printf("After translation:");
bar3d(midx+x+50,midy-(y+100),midx+x+60,midy-(y+90),10,1);
getch();
closegraph();
}
```

3D Scaling:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
int maxx,maxy,midx,midy;
void axis()
{
getch();
cleardevice();
line(midx,0,midx,maxy);
line(0,midy,maxx,midy);
}
void main()
{
int x,y,z,o,x1,x2,y1,y2;
int gd=DETECT,gm;
detectgraph(&gd,&gm);
```

```
initgraph(&gd,&gm,"c:\\tc\\bgi");
//setfillstyle(0,getmaxcolor());
maxx=getmaxx();
maxy=getmaxy();
midx=maxx/2;
midy=maxy/2;

axis();

bar3d(midx+50,midy-100,midx+60,midy-90,5,1);
printf("Enter scaling factors");
scanf("%d%d%d", &x,&y,&z);
//axis();
printf("After scaling");
bar3d(midx+(x*50),midy-(y*100),midx+(x*60),midy-(y*90),5*z,1);
//axis();
getch();
closegraph();
}
```

3D Rotation:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
int maxx,maxy,midx,midy;
void axis()
{
getch();
cleardevice();
line(midx,0,midx,maxy);
line(0,midy,maxx,midy);
}
void main()
{
int x,y,z,o,x1,x2,y1,y2;
int gd=DETECT,gm;
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"c:\\tc\\bgi");
//setfillstyle(0,getmaxcolor());
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);
getch();
closegraph();
}
```

Experiment No.: 11

Name of Experiment: Composite 3D transformations

AIM:

Implementation of 3D Transformation

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
int maxx,maxy,midx,midy;

void axis()
{
 getch();
 cleardevice();
 line(midx,0,midx,maxy);
 line(0,midy,maxx,midy);
}
void main()
{
 int gd,gm,x,y,z,ang,x1,x2,y1,y2;
 detectgraph(&gd,&gm);
 initgraph(&gd,&gm,"C:/TC/BGI");
 setfillstyle(3,25);
 maxx=getmaxx();
 maxy=getmaxy();
 midx=maxx/2;
 midy=maxy/2;
 outtextxy(100,100,"ORIGINAL OBJECT");
 line(midx,0,midx,maxy);
 line(0,midy,maxx,midy);
```

```
bar3d(midx+100,midy-20,midx+60,midy-90,20,5);
axis();
outtextxy(100,20,"TRANSLATION");
printf("\n\n Enter the Translation vector: ");
scanf("%d%d",&x,&y);
bar3d(midx+100,midy-20,midx+60,midy-90,20,5);
bar3d(midx+(x+100),midy-(y+20),midx+(x+60),midy-(y+90),20,5);
axis();
outtextxy(100,20,"SCALING");
printf("\n Enter the Scaling Factor: ");
scanf("%d%d%d",&x,&y,&z);
bar3d(midx+100,midy-20,midx+60,midy-90,20,5);
bar3d(midx+(x*100),midy-(y*20),midx+(x*60),midy-(y*90),20*z,5);
axis();
outtextxy(100,20,"ROTATION");
printf("\n Enter the Rotation angle: ");
scanf("%d",&ang);
x1=100*cos(ang*3.14/180)-20*sin(ang*3.14/180);
y1=100*sin(ang*3.14/180)+20*sin(ang*3.14/180);
x2=60*cos(ang*3.14/180)-90*sin(ang*3.14/180);
y2=60*sin(ang*3.14/180)+90*sin(ang*3.14/180);
axis();
printf("\n After rotating about z-axis\n");
bar3d(midx+100,midy-20,midx+60,midy-90,20,5);
bar3d(midx+x1,midy-y1,midx+x2,midy-y2,20,5);
axis();
printf("\n After rotating about x-axis\n");
bar3d(midx+100,midy-20,midx+60,midy-90,20,5);
bar3d(midx+100,midy-x1,midx+60,midy-x2,20,5);
axis();
printf("\n After rotating about y-axis\n");
```

```
  bar3d(midx+100,midy-20,midx+60,midy-90,20,5);
  bar3d(midx+x1,midy-20,midx+x2,midy-90,20,5);
  axis();
  closegraph();
}
```

Experiment No.: 12

Name of Experiment: Drawing three dimensional objects and scenes

White Rectangle on a Black Background (3-Dimension co-ordinates)

```
#include "stdafx.h"
#include "gl/glut.h"
#include <gl/gl.h>
void Display(void)
{
glClearColor (0.0, 0.0, 0.0, 0.0);
glClear (GL_COLOR_BUFFER_BIT);
glColor3f (1.0, 1.0, 1.0);
glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
glBegin(GL_POLYGON);
glVertex3f (0.25, 0.25, 0.0);
glVertex3f (0.75, 0.25, 0.0);
glVertex3f (0.75, 0.75, 0.0);
glVertex3f (0.25, 0.75, 0.0);
glEnd();
glFlush();
}
int main (int argc, char **argv)
{
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize(640,480);
glutCreateWindow("Intro");
glClearColor(0.0,0.0,0.0,0.0);
glutDisplayFunc(Display);
glutMainLoop();
return 0;
}
```

Example to draw smooth shaded Trigangle with shades

```
#include "stdafx.h"
#include "gl/glut.h"
#include <gl/gl.h>
void init(void)
{
glClearColor (0.0, 0.0, 0.0, 0.0);
glShadeModel (GL_SMOOTH);
gluOrtho2D (0.0, 640.0, 0.0, 480.0);
glMatrixMode (GL_PROJECTION);
glLoadIdentity ();
}
void display(void)
{
glClear (GL_COLOR_BUFFER_BIT);
glBegin (GL_TRIANGLES);
glColor3f (1.0, 0.0, 0.0);
glVertex2f (50.0, 50.0);
glColor3f (0.0, 1.0, 0.0);
glVertex2f (250.0, 50.0);
glColor3f (0.0, 0.0, 1.0);
glVertex2f (50.0, 250.0);
glEnd();
glFlush ();
}
int main(int argc, char** argv)
{
glutInit(&argc, argv);
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize (500, 500);
```

```
        glutInitWindowPosition (100, 100);
        glutCreateWindow ("Shade");
        init ();
        glutDisplayFunc(display);
        glutMainLoop();
        return 0;
}
```

Experiment No.: 13

Name of Experiment: Generating Fractal images

```
#include<stdio.h>
#include<math.h>
#include<graphics.h>
  int a;
void drawfern(int x,int y,int l,int arg,int n)
{
int x1,y1,i;
int l1,xpt,ypt;
if(n>0&&!kbhit())
 {
 x1=(int)(x-l*sin(arg*3.14/180));
 y1=(int)(y-l*cos(arg*3.14/180));
 line(x,y,x1,y1);
 l1=(int)(l/5);
 for(i=1;i<6;i++)
  {
  xpt=(int)(x-i*l1*sin(arg*3.14/180));
  ypt=(int)(y-i*l1*cos(arg*3.14/180));
  drawfern(xpt,ypt,(int)(l/(i+1)),arg+a,n-1);
  drawfern(xpt,ypt,(int)(l/(i+1)),arg-a,n-1);
  }
 }
}
void main()
{
int gd=DETECT,gm,x,y,l;
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI\\");
x=getmaxx()/2;
y=getmaxy()/2;
```

```
l=150;
a=45;
setcolor(YELLOW);
drawfern(x,y,l,0,5);
getch();
}
```

Viva Questions

Q1. Can you give some basic features of computer graphics?

Ans. The salient feature of computer graphics is the creation and manipulation of graphics (artificial images) by computer.

Q2. Can you tell which major components (hardware and software) are needed for computer graphics?

Ans. Besides the basic computer, some special devices and software may be required especially for computer graphics. For hardware, a special high-resolution, color monitor is often demanded and some input tools, e.g. mouse and joy-sticker, and hard-copy devices, e.g. high-resolution color printer, may be required. For software, some special purpose utilities (device-dependent and device-independent) are needed for handling processing in computer graphics.

Q3. Can you list at least three important applications of computer graphics?

Ans. There are many interesting applications of computer graphics. Three common applications are graphic user interface (GUI), computer-aided design (CAD), and computer games.

Q4. Define Computer Graphics.

Ans. Computer graphics remains one of the most existing and rapidly growing computer fields. Computer graphics may be defined as a pictorial representation or graphical representation of objects in a computer.

Q 5. What is meant by scan code?

Ans. When a key is pressed on the keyboard, the keyboard controller places a code carry to the key pressed into a part of the memory called as the keyboard buffer. This code is called as the scan code.

Q 6. What does refreshing of the screen mean?

Ans. Some method is needed for maintaining the picture on the screen. Refreshing of screen is done by keeping the phosphorus glowing to redraw the picture repeatedly. i.e. by quickly directing the electronic beam back to the same points.

Q 7. Define Random Scan/Raster Scan displays.

Ans. Random scan is a method in which the display is made by the electronic beam, which is directed, only to the points or part of the screen where the picture is to be drawn.

The Raster scan system is a scanning technique in which the electrons sweep from top to bottom and from left to right. The intensity is turned on or off to light and unlight the pixel.

Q 8. Explain the merits and demerits of Penetration techniques.

Ans. The merits and demerits of the Penetration techniques are as follows:

1. It is an inexpensive technique.
2. It has only four colors.
3. The quality of the picture is not good when it is compared to other techniques.
4. It can display color scans in monitors.

Q 9. Explain the merits and demerits of DVST.

Ans. The merits and demerits of direct view storage tubes (DVST) are as follows:

1. It has a flat screen.
2. Refreshing of screen is not required.
3. Selective or part erasing of screen is not possible.
4. It has poor contrast.
5. Performance is inferior to the refresh CRT.

Q 10. What do you mean by emissive and non-emissive displays?

Ans. EMISSIVE:

The emissive display converts electrical energy into light energy. The plasma panels, thin film electro-luminescent displays are the examples.

NON-EMISSIVE:

They are optical effects to convert the sunlight or light from any other source to graphic form. Liquid crystal display is an example.

Q 11. Explain the merits and demerits of Plasma panel display.

Ans. ADVANTAGES:

1. Refreshing is not required.
2. Produce a very steady image free of Flicker.
3. Less bulky than a CRT.

DISADVANTAGES:

1. Poor resolution of up to 60 d.p.i.
2. It requires complex addressing and wiring.
3. It is costlier than CRT.

Q 12. What is persistence?

Ans. The time it takes the emitted light from the screen to decay one tenth of its original intensity is called as persistence.

Q 13. What is resolution?

Ans. The maximum number of points that can be displayed without an overlap on a CRT is called as resolution.

Q 14. What is Aspect Ratio?

Ans. The ratio of vertical points to the horizontal points necessary to produce length of lines in both directions of the screen is called Aspect Ratio. Usually the aspect ratio is ¾.

Q 15. What is meant by Addressability?

Ans. Addressability is the number of individual dots per inch (d.p.i.) that can be created. If the address of the current dot is (x, y) then the next dot will be (x + y), (x + y + 1) etc.

Q 16. What is a dot size?

Ans. Dot size may be defined as the diameter of a single dot on the devices output. Dot size is also called as the Spot size.

Q 17. What is interdot distance?

Ans. Interdot distance is the reciprocal of addressability. If the addressability is large, the interdot distance will be less. The interdot distance should be less to get smooth shapes.

Q 18. What is the difference between impact and non-impact printers?

Ans. Impact printers press formed character faces against an inked ribbon on to the paper. A line printer and dot-matrix printer are examples.

Non-impact printer and plotters use Laser techniques, inkjet sprays, Xerographic process, electrostatic methods and electro thermal methods to get images onto the papers. Examples are: Inkjet/Laser printers.

Q 19. What is the features of Inkjet printers?

Ans. Features of inkjet printers:

1. They can print 2 to 4 pages/minutes.
2. Resolution is about 360d.p.i. Therefore better print quality is achieved.
3. The operating cost is very low. The only part that requires replacement is ink cartridge.
4. Four colors cyan, yellow, magenta, black are available.

Q 20. What are the advantages of laser printers?

Ans. Advantages of laser printer:

1. High speed, precision and economy.
2. Cheap to maintain.
3. Quality printers.
4. Lasts for longer time.
5. Toner power is very cheap.

Q 21. What is the advantages of electrostatic plotters?

Ans. Advantages of electrostatic plotters:

1. They are faster than pen plotters and very high quality printers.
2. Recent electrostatic plotters include a scan-conversion capability.
3. Color electrostatic plotters are available. They make multiple passes over the paper to plot color pictures.

Q 22. Explain the differences between a general graphics system designed for a programmer and one designed for a specific application, such as architectural design?

Ans. Basically, packages designed for graphics programming contain functions for setting primitives, attributes, and parameters for various graphics operations such as viewing and transformations. Packages designed for applications allow a user to create scenes in terms of the particular application, rather than in terms of graphics functions.

Q 23. Consider three different raster systems with resolutions of 640 x 480, 1280 x 1024 and 2560 x 2048.

1. **What size is frame buffer (in bytes) for each of these systems to store 12 bits per pixel?**
2. **How much storage (in bytes) is required for each system if 24 bits per pixel are to be stored?**

Ans. 1. Because eight bits constitute a byte, frame-buffer sizes of the systems are as follows:
640 x 480 x 12 bits π 8 = 450KB
1280 x 1024 x 12 bits π 8 = 1920KB
2560 x 2048 x 12 bits π 8 = 7680KB
2. Similarly, each of the above results is just doubled for 24 (12x2) bits of storage per pixel.

Q 24. Consider two raster systems with the resolutions of 640 x 480 and 1280 x 1024.

1. **How many pixels could be accessed per second in each of these systems by a display controller that refreshes the screen at a rate of 60 frames per second?**
2. **What is the access time per pixel in each system?**

Ans. 1. Since 60 frames are refreshed per second and each frame consists of 640 x 480 pixels, the access rate of such a system is:
(640 x 480) * 60 = 1.8432 x 107 pixels/second.
Likewise, for the 1280 x 1024 system, the access rate is:
(1280 x 1024) * 60 = 7.86432 x 107 pixels/second.

2. According to the definition of access rate, we know that the access time per pixel should be 1/(access rate). Therefore, the access time is around 54 nanoseconds/pixel for the 640 x 480 system, and the access time is around 12.7 nanoseconds/pixel for the 1280 x 1024 system.

Q 25. Consider a raster system with the resolution of 1024 x 768 pixels and the color palette calls for 65,536 colors. What is the minimum amount of video RAM that the computer must have to support the above-mentioned resolution and number of colors?

Ans. Recall that the color of each pixel on a display is represented with some number of bits. Hence, a display capable of showing up to 256 colors is using 8 bits per pixels (i.e., "8-bit color").

Notice, first that the color palette calls for 65,536 colors. This number is but 216, which implies that 16 bits are being used to represent the color of each pixel on the display. The display's resolution is 1024 by 768 pixels, which implies that there is a total of 786,432 (1024 × 768) pixels on the display. Hence, the total number of bits required to display any of 65,536 colors on each of the screen's 786,432 pixels is 12,582,912 (786,432 × 16). Dividing this value by 8 yields an answer of 1,572,864 bytes. Dividing that value by 1,024 yields an answer of 1,536 KB. Dividing that value by 1,024 yields an answer of 1.5 MB.

Resources

Text Books:

1. Computer Graphics C Version by Hearn & Baker, Pearson Publication
2. Schaum's Outline Series, TMH Publication.
3 J. Foley, A.Van Dam, S. Feiner, J.Hughes, : Computer graphics- principles and practice pearson
4. Amarendra Sinha, Arun D Udai: Computer Graphics, McGraw Hill

Reference Books:

1. Interaction to computer graphics- David J.Eck, Hobart and William smith
2. Schaum outline computer graphics by Xiang , McGraw Hill
3. Animated realism by Judith Krigor, Routhedge ISBN: 9781136130052

Reference Websites:

1. www.sourcecodesworld.com