Page No.
Date

# Experiment 1

**Aim**: Study of fundamental graphics functions.

arc() – Draws a circular arc.

Declaration :–

void far arc(int x, int y, int st_angle, int end_angle, radius);

where (x, y) = centerpoint of arc.
stangle = starting angle in degrees.
endangle = ending angle in degrees.
radius = radius of arc.

ellipse() – Draws an elliptical arc in the current drawing colour.

Declaration :

void far ellipse (int x, int y, int stangle, int endangle, int xrad, int yrad);

fillellipse() :– Draws an ellipse, then fill the ellipse with the current fill colour and pattern.

Declaration :

void far ellipse (int x, int y, int xrad, int yrad);

getmaxx() – getmaxx returns the max. x-value (screen relative) for the current graphics driver & mode.

Page No. _____
Date

Declaration — int far getmaxx (void);

getmaxy() — returns the max. y-value (screen relative) for the current graphics driver & mode.
Declaration — int far getmaxy (void);

grapherrormsg() — Returns a pointer to the error message string associated with error code, the value returned by graphresult.
Declaration: char * far grapherrormsg (int errorcode);

initgraph() — It initialises the graphics system by loading a graphics driver from disk the putting system into graphics mode.
Declaration —
void far initgraph (int far * graphdriver, int far * graphmode char far* pathdriver);

Putpixel() — It plots a point in the colour defined by color at (x, y).
Putpixel() does not return.
Declaration —
void far putpixel (int x, int y, int color);

Page No. _____
Date

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

Declaration - void far rectangle(int left, int top, int right, int bottom);

setbkcolor() - Sets the background to the color specified by color.

Declaration - void far setbkcolor(int color);

closegraph() - It deallocates all memory allocated by the graphic system. It then restores the screen to the mode it was in before you called init graph.

Declaration -
void far closegraph(void);

line() - Draws a line from $(x_1, y_1)$ to $(x_2, y_2)$ using the current color, line style & thickness. It does not update the current position (CP).

Declaration -
void far line(int $x_1$, int $y_1$, int $x_2$, int $y_2$);

Page No.
Date

## SOURCE CODE

```
#include <conio.h>
#include <stdio.h>
#include <graphics.h>
{ int i;
  int gd = DETECT, gm, err;
  initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
  err = graphresult();
  if(err != grOK)
  {
    printf("%s", grapherrormsg(err));
  }
  printf("%d", getmaxx());
  printf("\n");
  printf("%d", getmaxy());
  line(319, 0, 319, 479);
  line(0, 239, 639, 239);
  arc(159, 119, 0, 360, 60);
  ellipse(159, 358, 0, 360, 60, 70);
  fillellipse(478, 398, 10, 10);
  arc(478, 358, 0, 360, 60);
  arc(478, 358, 0, 360, 60);
  arc(139, 109, 0, 360, 10);
  arc(179, 109, 0, 360, 10);
  arc(458, 109, 0, 360, 10);
  arc(498, 109, 0, 360, 10);
```

```
arc(458, 129, 0, 360, 2);
arc(458, 139, 0, 360, 2);
arc(458, 129, 0, 360, 2);
arc(498, 139, 0, 360, 2);
arc(139, 348, 0, 360, 10);
arc(179, 348, 0, 360, 10);
arc(458, 348, 0, 360, 10);
arc(498, 348, 0, 360, 10);
line(159, 119, 159, 139);
line(498, 119, 478, 139);
line(159, 358, 159, 378);
line(478, 358, 478, 378);
arc(159, 139, 200, 340, 20);
arc(478, 169, 20, 160, 20);
line(139, 388, 179, 388);
for(i=0; i<50; i++){
    putpixel(319+i, 239+i, BLUE);
}

setbkcolor(RED));
rectangle(269, 189, 319, 239);
    closegraph();
    getch();
}
```

16/8/17

Page No.
Date

# Experiment 02

**Aim :** Implementation of DDA line drawing algorithm - dda algo.

**Software Used :** Turbo C

**Source Code**

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include<graphics.h>
#include round(a) int (a+0.5)

void dda(int x1, int y1, int x2, int y2)
{
  int dx, dy, k;
  float x, y, xi, yi, s;
  dx = x2 - x1;
  dy = y2 - y1;
  x = x1;
  y = y1;

  if (abs(dx) > abs(dy))
  {
    s = abs(dx);
  }
```

```
else {
        s = abs(dy);
        }

    xi = dx/s;
    yi = dy/s;
    printf("%d", dx);
    printf("%d", dy);
    printf("%f", s);
    putpixel(x, y, 1);
    for(k=0; k<s; k++) {
    {
        x = x+xi;
        y = y+yi;
        putpixel(x, y, 1);
        printf("%f %f", x, y);
        printf("\n");
    }}
void main(){
    int gd = DETECT, gm;
    int x1, y1, x2, y2;
    initgraph(&gd, &gm, "C:\\TC\\BGI");
    printf("Enter the values of first coordinates : ");
    scanf("%d %d", &x1, &y1);
    printf("Enter the values of second coordinates : ");
    scanf("%d %d", &x2, &y2);
    dda(x1, y1, x2, y2);
    getch();
}
```

Date:
Page No.:

# Experiment 2 (B)

**Aim**: Implementation of line drawing algorithm. Bresenham Algorithm.

**Software Used**: Turbo C7.

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>
void BL(int x1, int y1, int x2, int y2)
{
  int dx = abs(x1-x2), dy = abs(y1-y2);
  int pk = 2*dy-dx;
  int twody = 2*dy, twodydx = 2*(dy-dx);
  int x, y, xend;
  if (x1>x2)
  {
     x = x2;
     y = y2;
     xend = x1;
     }
  else
     {
     y = y1;
     x = x1;
     xend = x2;
  }
```

```
putpixel (x, y, 1);
while (x < xend)
    {
        x++;
        if (pk < 0)
        {
            pk = pk + twody;
        }
    else
        {
            y++;
            pk = pk + twodydx;
        }
    putpixel (x, y, 1);
    printf ("%d" + %d", x, y);
    printf ("\n"); }
}
void main(){
    int gd = DETECT, gm;
    int x1, y1, x2, y2;
    initgraph (&gd, &gm, "C:\\TURBOC3\\BGI");
    printf ("Enter the first coordinates : ");
    scanf ("%d %d", &x1, &y1);
    printf ("Enter the second coordinates : ");
    Scanf ("%d %d" &x2, &y2);
    BL(x1, y1, x2, y2);
    getch();
    closegraph(); }
```

CLASSTIME | Page No.
Date / /

# Experiment 3 (a)

**Aim**: Implementation of circle drawing algorithms:
(a). Bresenham Algorithm.

**Software Used**: TURBO C.

**Source Code**: Bresenham's Circle Algorithm

```
#include <conio.h>
#include <stdio.h>
#include <graphics.h>
#include <math.h>
void plot(int xc, int yc, int x, int y)
{
  putpixel (xc+x, yc+y, WHITE);
  putpixel (xc-x, yc+y, WHITE);
  putpixel (xc+x, yc-y, WHITE);
  putpixel (xc-x, yc-y, WHITE);
  putpixel (xc+y, yc+x, WHITE);
  putpixel (xc-y, yc+x, WHITE);
  putpixel (xc+y, yc-x, WHITE);
  putpixel (xc-y, yc-x, WHITE);
}
void circle(int xc, int yc, int r)
{
 int x, y, d;
 x=0;
```

```
d = 3 - 2*r;
y = r;
plot(xc, yc, x, y);
while(x < y)
{
    x++;
    if(d < 0)
    {
        d = d + 4*x + 6;
    }
    else {
        y--;
        d = d + 4*(x-y) + 10;
    }
    plot(xc, yc, x, y);
  }
}
void main()
{
int gd = DETECT, gm;
int xc, yc, r, x, y;
initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
printf("Enter the center coordinates of circle:");
scanf("%d%d", &xc, &yc);
printf("Enter the radius");
scanf("%d", &r);
cir(xc, yc, r);   getch();
closegraph();
}
```

CLASSTIME | Page No.
Date / /

# Experiment 3 (b)

**Aim:** Implementation of circle drawing algorithm (b). Mid point circle algorithm.

## Source Code

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>
void plot (int xc, int yc, int x, int y)
{
   putpixel (xc+x, xc+y, WHITE);
   putpixel (xc-x, yc+y, WHITE);
   putpixel (xc+x, yc-y, WHITE);
   putpixel (xc-x, yc-y, WHITE);
   putpixel (xc+y, yc+x, WHITE);
   putpixel (xc-y, yc+x, WHITE);
   putpixel (xc+y, yc-x, WHITE);
   putpixel (xc-y, yc-x, WHITE);
}
void circ( int xc, int yc, int r)
{
   int x, p, y;
   x = 0;
   p = 1-r;
   y = r;
```

Date / /

```
plot(xc, yc, x, y);
while(x<y)
{
    x++;
    if(p<0)
    {
        p = p+2*x+1;
    }
    else
    {   y--;
        p+=2*(x-y)+1;
    }
    plot(xc, yc, x, y);
}}
void main()
{ int gd = DETECT, gm;
  int xc, yc, r, x, y;
  initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
  printf("Enter the center coordinates of circle:");
  scanf("%d%d", &xc, &yc);
  printf("Enter the radius");
  scanf("%d", &r);
  cir(xc, yc, r);
  getch();
  closegraph();
}
```

Date:
Page No.:

# Experiment 4

**Aim:** Program on 2-d transformation :
(i) Translation (ii) Scaling (iii) Rotation

**Software Used:** Turbo C.

**Source Code**

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>
void main()
{
  int gd = DETECT, gm;
  initgraph(&gd, &gm, "C:/TC/BGI");
  int c;
  printf("Enter your choice \n");
  printf("1. Translation  2. Rotation  3. Scaling");
  scanf("%d", &c);
  switch(c)
  {
    case 1:
    {
      int x1, y1, x2, y2, tx, ty;
      printf("Enter initial set of points:");
      scanf("%d %d %d %d", &x1, &y1, &x2, &y2);
      line(x1, y1, x2, y2);
```

```
printf("Enter tx & ty : \n");
scanf("%d %d", &tx, &ty);
line(x1+tx, y1+ty, x2+tx, y2+ty);
getch();
break; }
case 2 : {
    int x1, x2, y1, y2, xn, yn;
    float angle;
    printf("Enter initial set of points:\n");
    scanf("%d %d %d %d", &x1, &y1, &x2, &y2);
    line(x1, y1, x2, y2);
    printf("Enter the angle of rotation \n");
    scanf("%f", &angle);
    angle = (angle * 3.14)/180;
    xn = cos(angle) * x2 - sin(angle) * y2;
    yn = sin(angle) * x2 + cos(angle) * y2;
    line(x1, y1, x2, y2);
    getch(); }
case 3 : {
    int x1, int y1, int x2, int y2; int sx, int sy;
    printf("Enter initial rectangle:");
    scanf("%d %d %d %d", &x1, &y1, &x2, &y2);
    rectangle(x1, y1, x2, y2);
    printf("Enter the scaling factor :");
    scanf("%d %d", &sx, &sy);
    rectangle(x1 * sx, y1 * sy, x2 * sx, y2 * sy); }
break; }
}
```

Telco Dt.: Pg.:

# Experiment 05

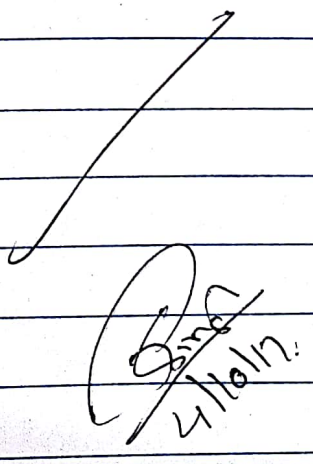
**Aim**: To perform 3d transformation – Scaling & translation.

**Software Used**: Turbo C7

**Source Code**

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
void 3dTrans()
{
  int midx, midy, tx, ty;
  midx = getmaxx()/2;
  midy = getmaxy()/2;
  bar3d(midx+50+tx, midy-100-ty, midx+70+tx,
        midy-80-ty, 20, 1);
  }
void scale()
{ int midx, midy, sx, sy;
  midx = getmaxx()/2;
  midy = getmaxy()/2;
  bar3d(midx+50, midy-100, midx+70, midy-80, 20, 1);
  printf("Enter the scaling factor:");
  scanf("%d %d", &sx, &sy);
  bar3d(midx+(50*sx), midy-(100*sy), midx+(70*sx)
        , midy-(80*sy), 20, 1);
  }
```

```
void main()
{
  int gd = DETECT, gm, errr, x, midx, midy;
  initgraph(&gd, &gm, "C:/TC/BGI");
  clrscr();
  midx = getmaxx()/2;
  midy = getmaxy()/2;
  line(0, midy, getmaxx(), midy);
  line(midx, 0, midx, getmaxy());
  printf("What do you want to do?");
  printf("\n Enter 1 for translation and 2 for scaling: ");
  scanf("%d", &x);
  switch(x)
  {
    case 1:
        {
          tran();
          break;
        }
    case 2:
        {
          scale();
          break;
        }
  }
  getch();
  closegraph();
}
```



Telco Dt.: Pg.:

# Experiment 6

- **Aim**: To implement Cohen Sutherland Line Clipping Algorithm
- **S/w Used**: Turbo C7
- **Source Code**

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
void main()
{
  int gd = DETECT, gm;
  float i, xmax, ymax, xmin, ymin, x1, y1, x2, y2, m;
  float start[4], end[4], code[4];
  clrscr();
  initgraph(&gd, &gm, "C://TURBOC3//BGI");
  printf("\n Enter the bottom left coordinate of viewport: ");
  scanf("%f %f", &xmin, &ymin);
  printf("\nEnter the top right coordinates of viewport: ");
  scanf("%f %f", &xmax, &ymax);
  printf(" Enter the coorcdinate for starting pt. of line: ");
  scanf("%f %f", &x1, &y1);
  printf("\n Enter the coordinate for ending pt. of line: ");
  scanf("%f %f", &x2, &y2);
```

```
for (i = 0; i < 4; i++)
{
    start [i] = 0;
    end [i] = 0;
}
m = (y2 - y1) / (x2 - x1)
if (x1 < xmin)  start[0] = 1;
if (x1 > xmax)  start[1] = 1;
if (y1 > ymax)  start[2] = 1;
if (y1 < ymin)  start[3] = 1;
if (x2 < xmin)  end[0] = 1;
if (x2 > xmax)  end[1] = 1;
if (y2 > ymax)  end[2] = 1;
if (y2 < ymin)  end[3] = 1;

for (i = 0; i < 4; i++)
        code[i] = start[i] && end[i]
if ((code[0] == 0) && (code[1] == 0) && (code[2] == 0)
    && (code[3] == 0))
{
    if ((start[0] == 0) && (start[1] == 0) && (start[2] == 0
        && (start[3] == 0))
    {
        cleardevice();
        printf("\n The line is totally visible & not
                a clipping candidate!");
```

```
rectangle (xmin, ymin, xmax, ymax);
line (x1, y1, x2, y2);
getch();
}
else{
    cleardevice();
    printf("\n The line is partially visible.");
    rectangle (xmin, ymin, xmax, ymax);
    line (x1, y1, x2, y2);
    getch();
if ((start[2] == 0) && (start[1] == 1))
 {
   x1 = x1 + (ymin - y1)/m;
   y1 = ymin;
 }
if ((end[2] == 0) && (end[3] == 1))
{ x2 = x2 + (ymin - y2)/m;
  y2 = xmin;
}

if ((start[2] == 1) && (start[3] == 0))
 {
   x1 = x1 + (ymax - y1)/m;
   y1 = ymax;
 }
```

```
if ((end[2] == 1) && (end[3] == 0))
{
  x2 = x2 + (ymax - y2)/m;
  y2 = ymax;
}
if ((start[1] == 0) && (start[0] == 1))
  {
    y1 = y1 + m * (xmin - x1);
    x1 = xmin;
  }
if ((end[1] == 0) && (end[0] == 1))
    {
      y2 = y2 + m * (xmin - x2);
      x2 = xmin;
    }
if ((start[1] == 1) && (start[0] == 0))
{
    y1 = y1 + m * (xmax - x1);
    x4 = xmax;
  }

 if ((end[1] == 1) && (end[0] == 0))
{
    y2 = y2 + m * (xmax - x2);
    x2 = xmax;
}
  closer();
  clearDevice();
```

Telco Dt.: Pg.:

```
printf("\n After clipping ...");
rectangle(xmin, ymin, xmax, ymax);
line(x1, y1, x2, y2);
getch();
}
} else {
    cleardevice();
    printf("\n The line is invisible !!");
    rectangle(xmin, ymin, xmax, ymax);
  }
  getch();
  closegraph();
}
```

# Experiment 87

**Aim**: To implement & show Bezier Curve.

**Software Used**: Turbo C-7.

**Source Code**

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>
void main()
{
  int gm, gd = DETECT;
  int i;
  float u;
  float px[4], py[4], cx, cy;
  initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
  for(i=0; i<=3; i++)
  {
    printf("Enter the coordinates of px");
    scanf("%f", &px[i]);
    printf("Enter the coordinate of py");
    scanf("%f", &py[i]);
  }
for(u=0.0; i<=1.0; i=u+0.005)
{
```

Telco Dt.: Pg.:

```
cx = (px[0] * pow(1-u, 3)) + (3*px[1] * u * pow(1-u, 2))
     + (3 * px[2] * pow(u, 2) * (1-u)) + (px[3] *
       pow(u, 3));
cy = (py[0] * pow(1-u, 3) + (3 * py[1] * u * pow(1-u, 2))
     + (3 * py[2] * pow(u, 2) * (1-u)) + (py[3] * pow(u, 3));
putpixel(cx, cy, 15);
}
getch();
closegraph();
}
```

Telco Dt.: Pg.:

# Experiment - 9

- **Aim**: Using blender, create a chair.
- **Software Used**: Blender
- **Steps** ~

1. Open Blender Software.
2. Select file > New to generate a new scene with requisite starter object.
3. These objects are located in 3D view (x, y, z).
4. First of all, there is a cube at the center.
5. Press the scroll button of the mouse to change the view of the cube.
6. Press S to scale the cube & move the mouse.
7. Press S & Z together to scale the cube in z-axis.
8. Go to Object mode > Select edit mode.
9. Press T to move to top view.
10. Press 'edit' mode.
11. Press Ctrl + R for loop out or select loop out from tool shelf.
12. Press ctrl + R four times on four corners of one face of the cuboid (scaled less) formed in step 6.
13. Move the camera.
14. Go from 'vertex mode' → 'face mode'.
15. Select 4 faces.

Telco Dt.: Pg.:

16. Press shift to select & hold the selection.
17. Press E to extrude & move the mouse.
18. All 4 legs of the table will come down together from selections designed in step 16.
19. We can also use tab to switch from object mode to edit mode.
20. Now, take one edge of the cuboid on the top surface & select two loop cuts (short) & one long cut on one side only, use shift.
21. Press E to extrude & move the mouse.
22. Press A to deselect.

## Result

We created a chair using blender & successfully learned its working.

Telco Dt.: Pg.:

# Experiment 10

- **Aim**: Create a fan having 4 turbines using Blender.

- **Software Used**: Blender.

- **Steps Involved** ~

1. Open Blender.
2. Select File > New to generate a new scene with requisite starter object.
3. You will get a cube as an object. Select it & delete it.
4. Select a sphere from tool shelf & press it to get it in view mode.
5. Press the scroll button of the mouse to ~~select~~ set the view of the plane.
6. Select Shape Rectangle 4 times from tool shelf & place it in vertical mode & horizontal mode to make it look like a turbine.
7. Select the rectangle & press S to scale the shape & move the mouse down to resize it in equal sizes.
8. Click on a rectangle, move it to the right side by 90° & do so for others too.

Telco Dt.:
Pg.:

9. Go to the timeline frame, move it by 10 fps.
10. Go to tool shelf, select for an animation in any view like line – loc, etc.
11. Do this for other 4 rectangles in the same way.
12. Make Sure that you are increasing the timeline by 10 – 20 fps again & again.
13. You will see that a yellow line is set at the timeline when translation animation takes place.
14. Set the maximum fps limit to 100 fps.
15. Select the play button from timeline frame menu.
16. Set the initial point as 0 fps on timeline frame, click on play button.
17. You will now see that fan starts rotating
18. You can also use Shortcut key R by selecting a turbine blade i.e. Rectangle using 'R' to rotate.

## Result

We have successfully created a rotating fan using Blender.