Computer Graphics Lab

(BCA – 351)

Department of Computer Applications

Maharaja Surajmal Institute

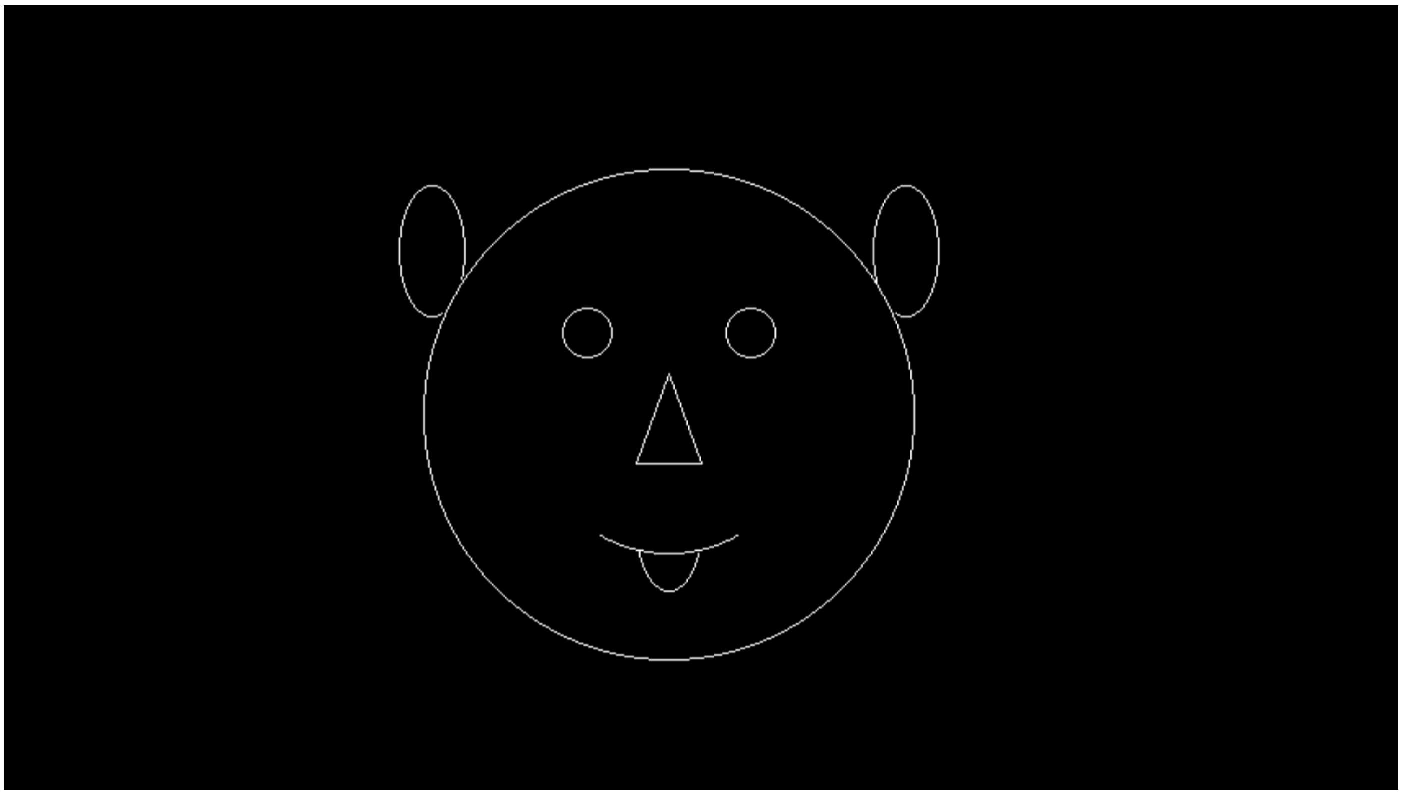
C – 4, Janakpuri

New Delhi -110058

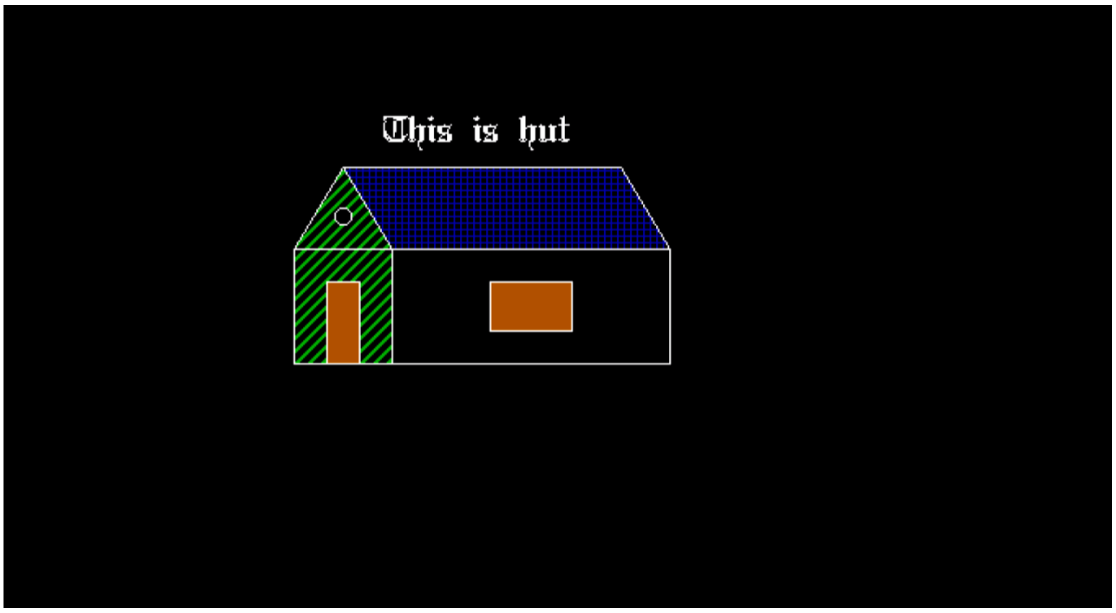
Submitted to:	Submitted by:
Dr. Pooja Singh	**Ankit Jee**
(Assistant Professor)	BCA - 5(M) Sec-A
	Enrol. No: **01214902018**

1. Drawing objects- Hut, Face

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

void main()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\Turboc3\\BGI");

    circle(300, 250, 150);

    circle(250, 200, 15);                //left eye
    ellipse(155, 150, 335, 290, 20, 40); //left ear

    circle(350, 200, 15);                //right eye
    ellipse(445, 150, 250, 210, 20, 40); //right ear

    line(300, 225, 280, 280); //for nose
    line(300, 225, 320, 280);
    line(280, 280, 320, 280);

    arc(300, 250, 240, 300, 85);         //for lips
    ellipse(300, 318, 205, 335, 20, 40); //for tongue
    getch();
    closegraph();
}
```

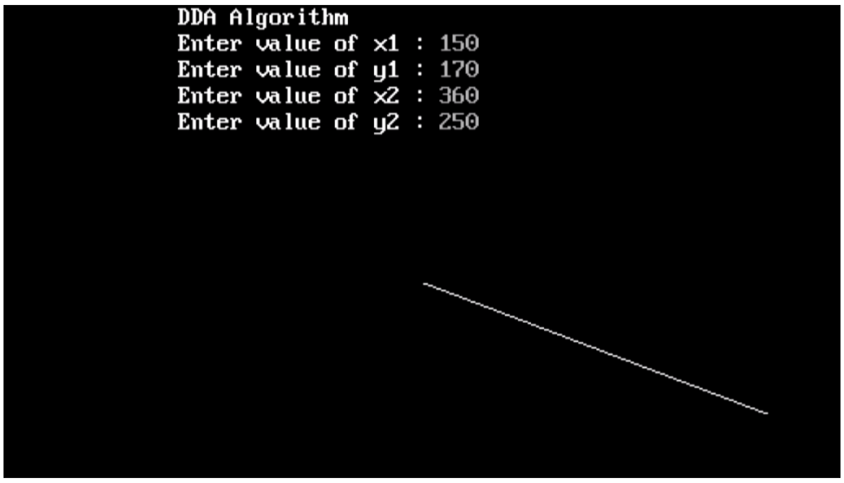


```c
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <dos.h>

void main()
{
    int gd = DETECT, gm, xend, d, x, y, x1, y1, x2, y2, dx, dy, dT, dS;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

    settextstyle(GOTHIC_FONT, 0, 3);
    outtextxy(225, 110, "This is hut");

    line(200, 150, 170, 200);
    line(200, 150, 230, 200);
    rectangle(170, 200, 400, 270);
    line(200, 150, 370, 150);
    line(370, 150, 400, 200);
    line(230, 200, 230, 270);
    circle(200, 180, 5);
    rectangle(290, 220, 340, 250);
    rectangle(190, 220, 210, 270);

    setfillstyle(SOLID_FILL, BROWN);
    floodfill(191, 221, WHITE);
    floodfill(291, 221, WHITE);

    setfillstyle(SLASH_FILL, GREEN);
    floodfill(171, 201, WHITE);
    floodfill(201, 155, WHITE);

    setfillstyle(HATCH_FILL, BLUE);
    floodfill(210, 155, WHITE);

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



2. DDA Line Drawing Algorithm

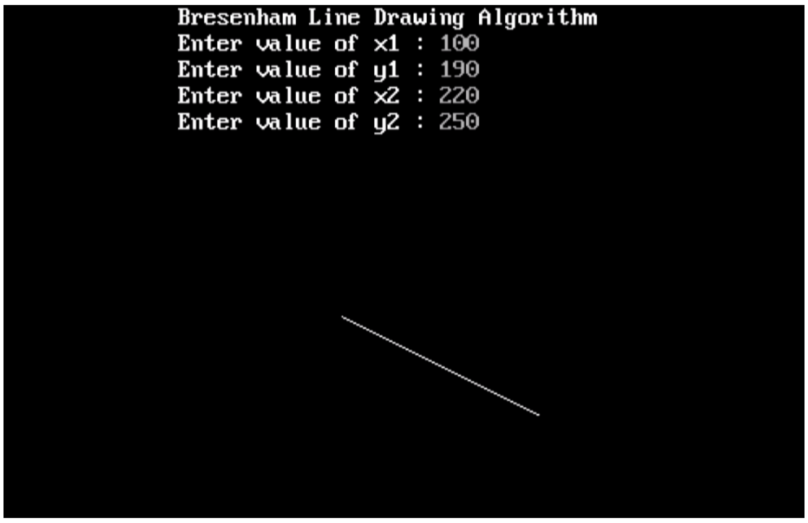
```c
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>

void main()
{
    int gd = DETECT, gm, i, x1, y1, x2, y2;
    float dx, dy, step, x, y;
    initgraph(&gd, &gm, "C:\\Turboc3\\BGI");
    printf("DDA Algorithm\n");
    printf("Enter value of x1 : ");
    scanf("%d", &x1);
    printf("Enter value of y1 : ");
    scanf("%d", &y1);
    printf("Enter value of x2 : ");
    scanf("%d", &x2);
    printf("Enter value of y2 : ");
    scanf("%d", &y2);

    dx = (float)(x2 - x1);
    dy = (float)(y2 - y1);

    step = (abs(dx) > abs(dy)) ? abs(dx) : abs(dy);

    dx = dx / step;
    dy = dy / step;

    x = x1;
    y = y1;

    for (i = 0; i <= step; ++i)
    {
        putpixel(x, y, WHITE);
        x += dx;
        y += dy;
        delay(50);
    }

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



```
DDA Algorithm
Enter value of x1 : 150
Enter value of y1 : 170
Enter value of x2 : 360
Enter value of y2 : 250
```

3. Bresenham's Line Drawing Algorithm

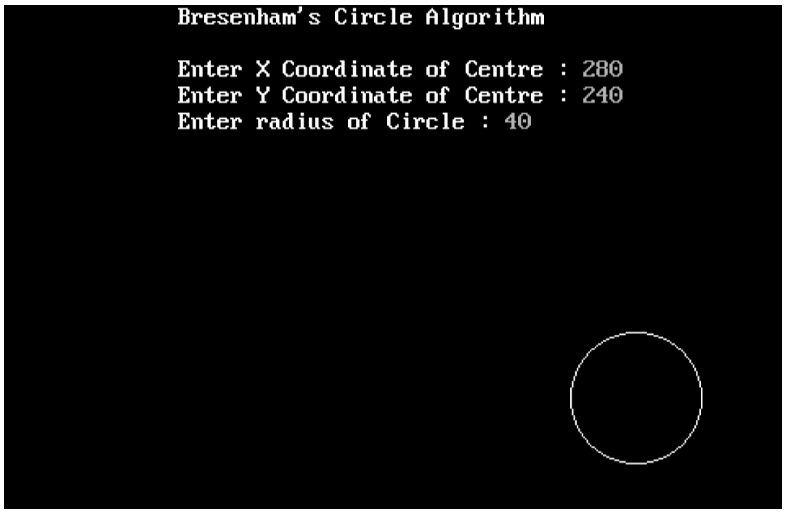
```c
#include<stdio.h>
#include<conio.h>
#include<graphics.h>

void main(){
    int gd=DETECT,gm,xend,d,x,y,x1,y1,x2,y2,dx,dy,dT,dS;
    initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");

    printf("Bresenham Line Drawing Algorithm\n");
    printf("Enter value of x1 : "); scanf("%d",&x1);
    printf("Enter value of y1 : "); scanf("%d",&y1);
    printf("Enter value of x2 : "); scanf("%d",&x2);
    printf("Enter value of y2 : "); scanf("%d",&y2);

    dx=x2-x1;
    dy=y2-y1;

    dT=2*(dy-dx);
    dS=2*dy;
    d=2*dy-dx;

    if(dx>0){
        x=x1;
        y=y1;
        xend=x2;
    }
    else{
        x=x2;
        y=y2;
        xend=x1;
    }

    putpixel(x,y,WHITE);

    while(x<xend){
        x++;
        if(d<0) d+=dS;
        else {
            d+=dT;
            y++;
        }
        putpixel(x,y,WHITE);
        delay(100);
    }

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



```
Bresenham Line Drawing Algorithm
Enter value of x1 : 100
Enter value of y1 : 190
Enter value of x2 : 220
Enter value of y2 : 250
```

4. Bresenham's Circle Mid-Point Algorithm

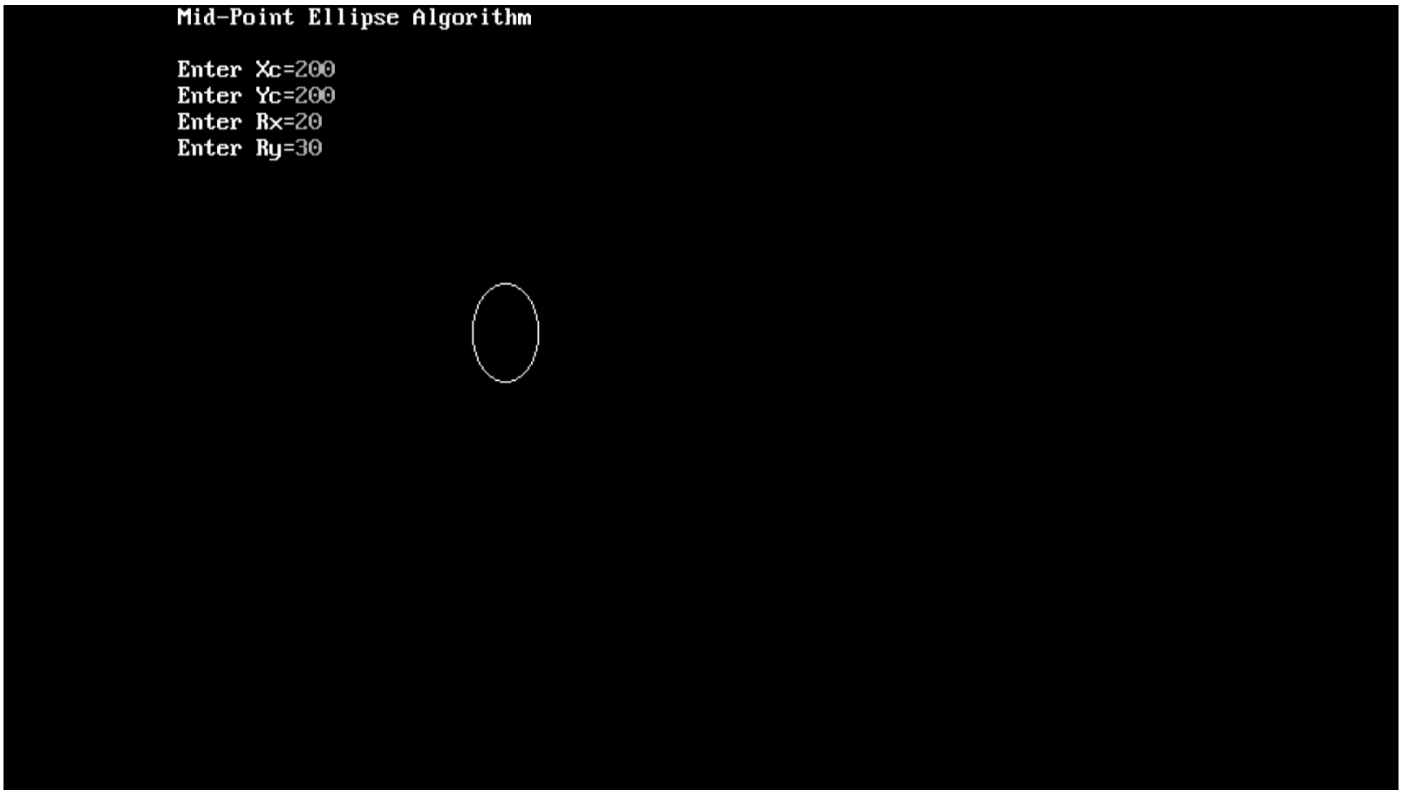
```c
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
int main()
{

    int gd = DETECT, gm, x, y, a, b, r, d;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

    printf("Bresenham's Circle Algorithm\n\n");
    printf("Enter X Coordinate of Centre : ");
    scanf("%d", &a);
    printf("Enter Y Coordinate of Centre : ");
    scanf("%d", &b);
    printf("Enter radius of Circle : ");
    scanf("%d", &r);

    x = 0;
    y = r;
    d = 3 - (2 * r);

    while (x <= y)
    {
        putpixel(a + x, b - y, WHITE);
        putpixel(a - x, b + y, WHITE);
        putpixel(a + x, b + y, WHITE);
        putpixel(a - x, b - y, WHITE);
        putpixel(a + y, b - x, WHITE);
        putpixel(a - y, b + x, WHITE);
        putpixel(a + y, b + x, WHITE);
        putpixel(a - y, b - x, WHITE);

        if (d < 0)
        {
            d = d + (4 * x) + 6;
        }
        else
        {
            d = d + (4 * (x - y)) + 10;
            y--;
        }
        x++;
    }
    getch();
    closegraph();
}
```



```
Bresenham's Circle Algorithm

Enter X Coordinate of Centre : 280
Enter Y Coordinate of Centre : 240
Enter radius of Circle : 40
```

5. Ellipse

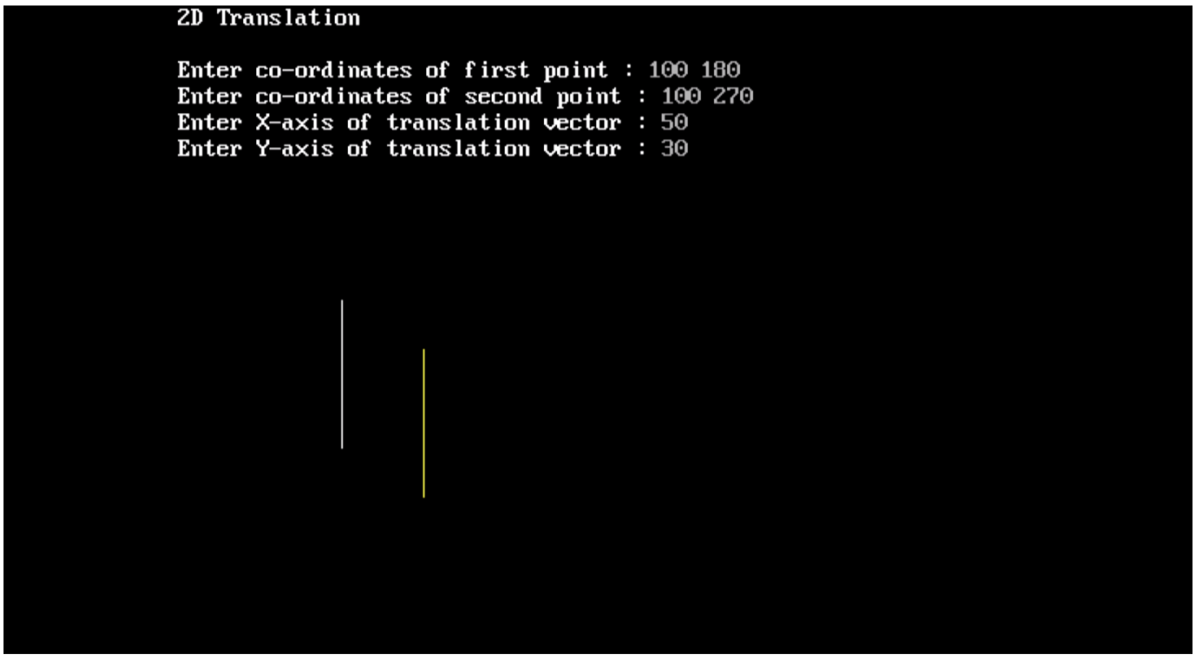
```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
void draw_ellipse(int xc, int yc, int rx, int ry)
{
    int x, y, p;

    x = 0;
    y = ry;
    p = (ry * ry) - (rx * rx * ry) + ((rx * rx) / 4);
    while ((2 * x * ry * ry) < (2 * y * rx * rx))
    {
        putpixel(xc + x, yc - y, WHITE);
        putpixel(xc - x, yc + y, WHITE);
        putpixel(xc + x, yc + y, WHITE);
        putpixel(xc - x, yc - y, WHITE);
        if (p < 0)
        {
            x = x + 1;
            p = p + (2 * ry * ry * x) + (ry * ry);
        }
        else
        {
            x = x + 1;
            y = y - 1;
            p = p + (2 * ry * ry * x + ry * ry) - (2 * rx * rx * y);
        }
    }
    p = ((float)x + 0.5) * ((float)x + 0.5) * ry * ry + (y - 1) * (y - 1) * rx * rx - rx *
 rx * ry * ry;
    while (y >= 0)
    {
        putpixel(xc + x, yc - y, WHITE);
        putpixel(xc - x, yc + y, WHITE);
        putpixel(xc + x, yc + y, WHITE);
        putpixel(xc - x, yc - y, WHITE);
        if (p > 0)
        {
            y = y - 1;
            p = p - (2 * rx * rx * y) + (rx * rx);
        }
        else
        {
            y = y - 1;
            x = x + 1;
            p = p + (2 * ry * ry * x) - (2 * rx * rx * y) - (rx * rx);
        }
    }
}
int main()
{
    int gd = DETECT, gm, xc, yc, rx, ry;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

    printf("Mid-Point Ellipse Algorithm\n\n");
    printf("Enter Xc=");
    scanf("%d", &xc);
    printf("Enter Yc=");
    scanf("%d", &yc);
    printf("Enter Rx=");
    scanf("%d", &rx);
    printf("Enter Ry=");
    scanf("%d", &ry);

    draw_ellipse(xc, yc, rx, ry);
    getch();
    closegraph();
}
```



```
Mid-Point Ellipse Algorithm

Enter Xc=200
Enter Yc=200
Enter Rx=20
Enter Ry=30
```

6. Translation in 2D

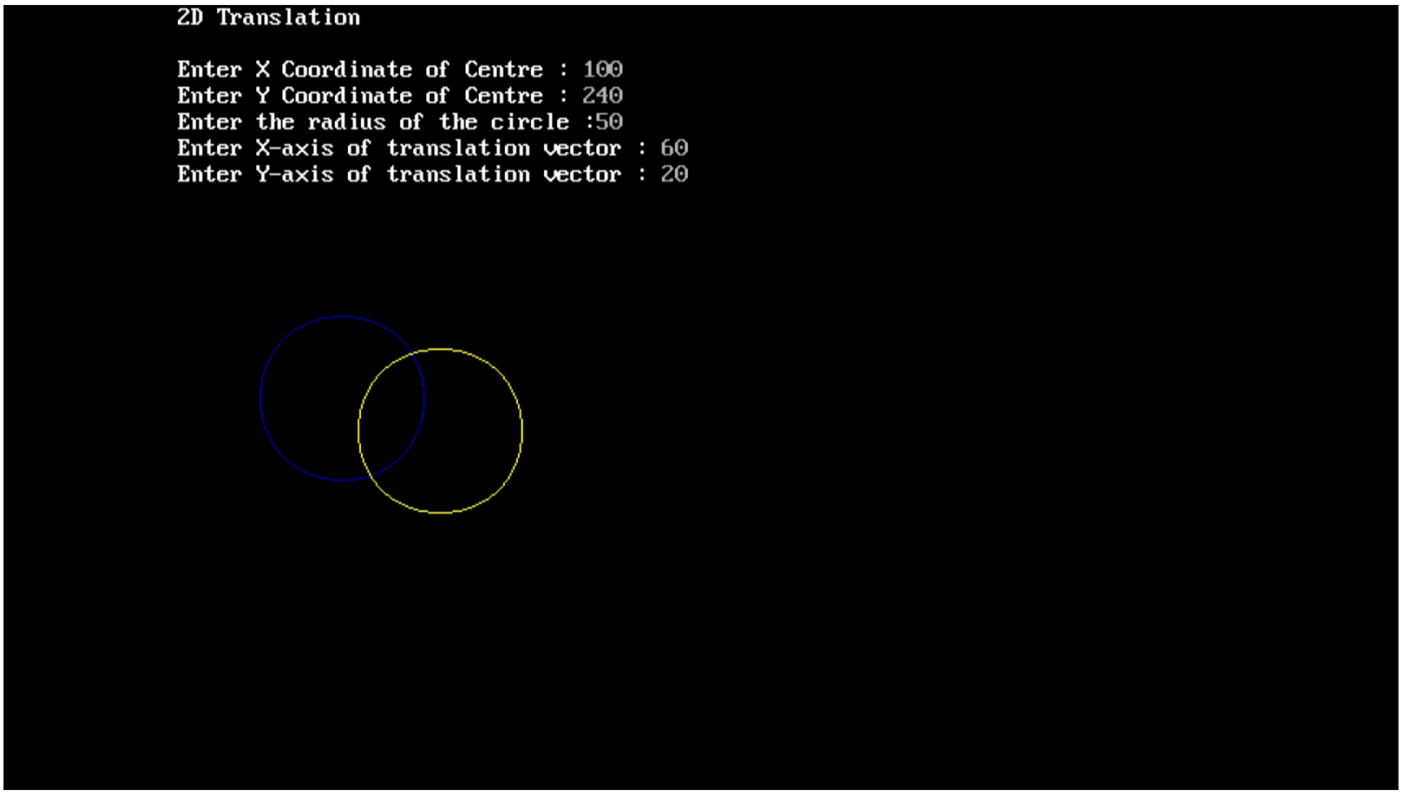
```c
#include <graphics.h>
#include <stdio.h>
#include <conio.h>

int main()
{
    int gd = DETECT, gm, X1, Y1, X2, Y2, Tx, Ty;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

    printf("2D Translation\n\n");
    printf("Enter co-ordinates of first point : ");
    scanf("%d %d", &X1, &Y1);
    printf("Enter co-ordinates of second point : ");
    scanf("%d %d", &X2, &Y2);
    printf("Enter X-axis of translation vector : ");
    scanf("%d", &Tx);
    printf("Enter Y-axis of translation vector : ");
    scanf("%d", &Ty);

    line(X1, Y1, X2, Y2);

    X1 += Tx;
    X2 += Tx;
    Y1 += Ty;
    Y2 += Ty;

    setcolor(YELLOW);
    line(X1, Y1, X2, Y2);

    getch();

    closegraph();
}
```



```
2D Translation

Enter co-ordinates of first point : 100 180
Enter co-ordinates of second point : 100 270
Enter X-axis of translation vector : 50
Enter Y-axis of translation vector : 30
```

7. Translation in 2D (Circle)

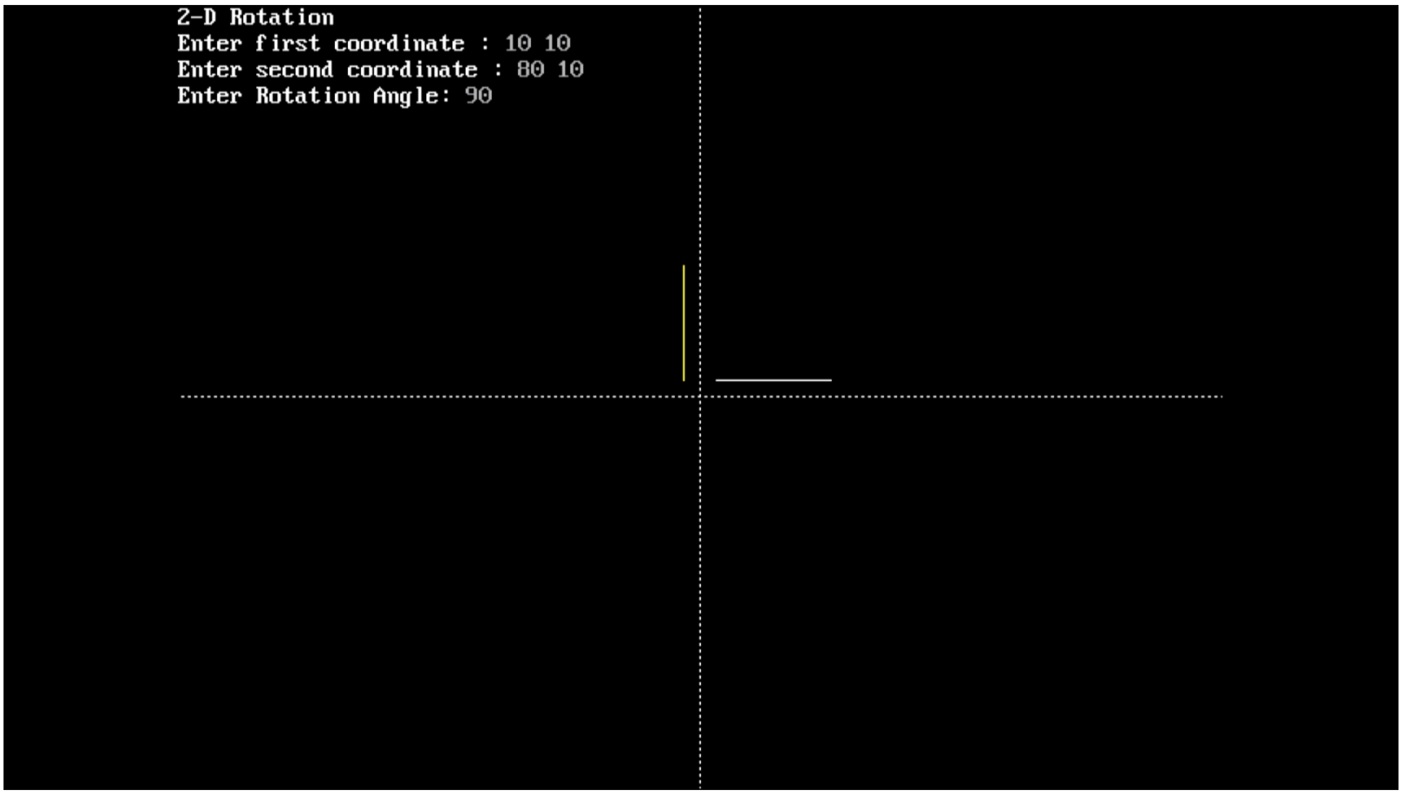
```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>

void symmetry(int x, int y, int xc, int yc, int col)
{
    putpixel(xc + x, yc - y, col);
    putpixel(xc + y, yc - x, col);
    putpixel(xc + y, yc + x, col);
    putpixel(xc + x, yc + y, col);
    putpixel(xc - x, yc + y, col);
    putpixel(xc - y, yc + x, col);
    putpixel(xc - y, yc - x, col);
    putpixel(xc - x, yc - y, col);
}

void draw_circle(int xc, int yc, int rad, int col)
{
    int x = 0;
    int y = rad;
    int p = 1 - rad;

    for (x = 0; y >= x; x++)
    {
        symmetry(x, y, xc, yc, col);

        if (p < 0)
            p += 2 * x + 1;
        else
        {
            p += 2 * (x - y) + 1;
            y--;
        }
    }
}

int main()
{
    int gd = DETECT, gm, xc, yc, R, Tx, Ty;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

    printf("2D Translation\n\n");
    printf("Enter X Coordinate of Centre : ");
    scanf("%d", &xc);
    printf("Enter Y Coordinate of Centre : ");
    scanf("%d", &yc);
    printf("Enter the radius of the circle :");
    scanf("%d", &R);
    printf("Enter X-axis of translation vector : ");
    scanf("%d", &Tx);
    printf("Enter Y-axis of translation vector : ");
    scanf("%d", &Ty);
    draw_circle(xc, yc, R, 1);

    xc += Tx;
    yc += Ty;

    draw_circle(xc, yc, R, 14);

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



```
2D Translation

Enter X Coordinate of Centre : 100
Enter Y Coordinate of Centre : 240
Enter the radius of the circle :50
Enter X-axis of translation vector : 60
Enter Y-axis of translation vector : 20
```

8. Rotation in 2D

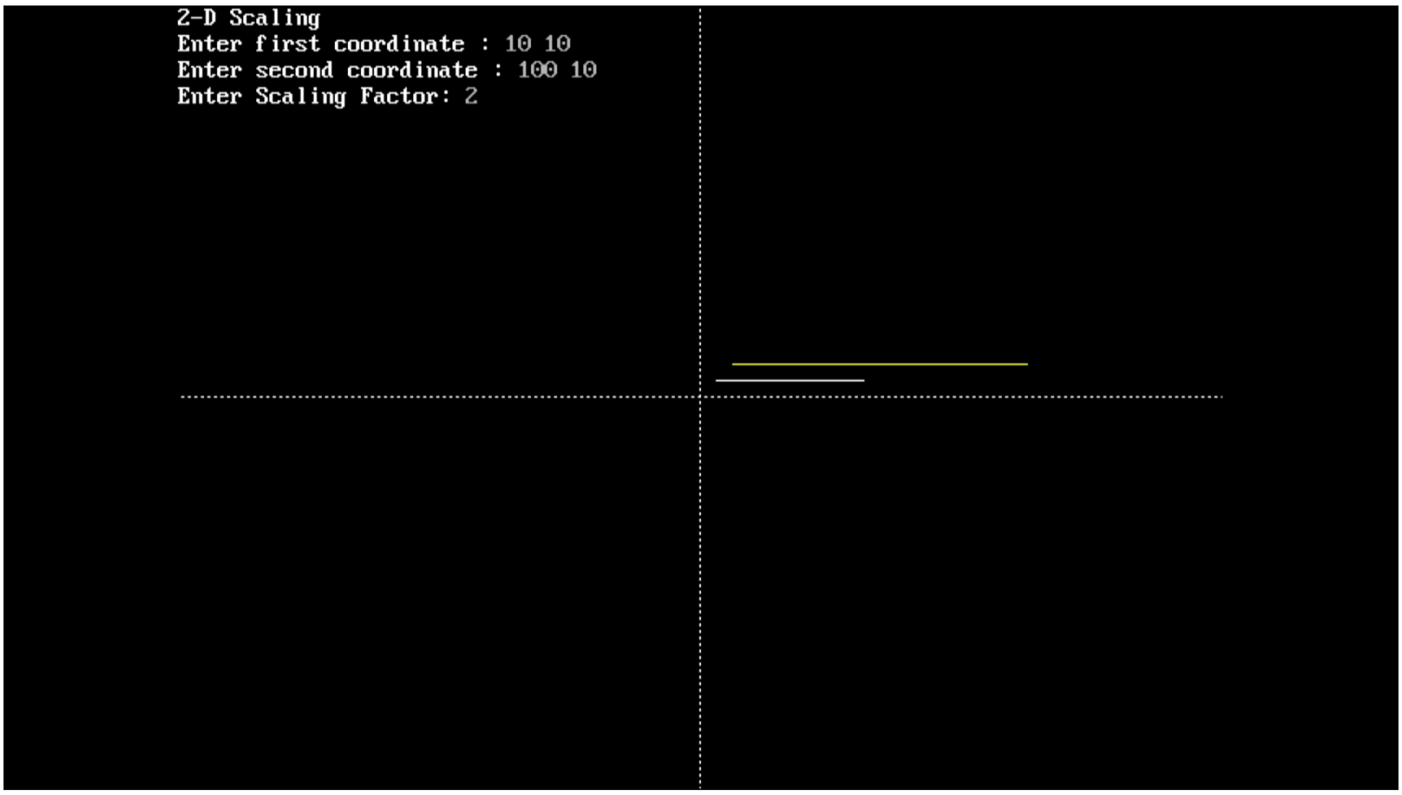
```c
#include <conio.h>
#include <math.h>
#include <graphics.h>

#define SIN(x) sin(x * 3.14159 / 180)
#define COS(x) cos(x * 3.14159 / 180)
#define ox (getmaxx() / 2)
#define oy (getmaxy() / 2)

void main()
{
    int gd = DETECT, gm, s, x1, y1, x2, y2, x, y;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

    printf("2-D Rotation\n");
    printf("Enter first coordinate : ");
    scanf("%d %d", &x1, &y1);
    printf("Enter second coordinate : ");
    scanf("%d %d", &x2, &y2);

    setlinestyle(DOTTED_LINE, 0, 1);
    line(0, oy, ox * 2, oy);
    line(ox, 0, ox, oy * 2);

    setlinestyle(SOLID_LINE, 0, 1);
    line(ox + x1, oy - y1, ox + x2, oy - y2);

    printf("Enter Rotation Angle: ");
    scanf("%d", &s);
    x = x1;
    y = y1;
    x1 = floor(x * COS(s) - y * SIN(s));
    y1 = floor(y * COS(s) + x * SIN(s));

    x = x2;
    y = y2;

    x2 = floor(x * COS(s) - y * SIN(s));
    y2 = floor(y * COS(s) + x * SIN(s));

    setcolor(YELLOW);
    line(ox + x1, oy - y1, ox + x2, oy - y2);

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



```
2-D Rotation
Enter first coordinate : 10 10
Enter second coordinate : 80 10
Enter Rotation Angle: 90
```

9. Scaling Line in 2D

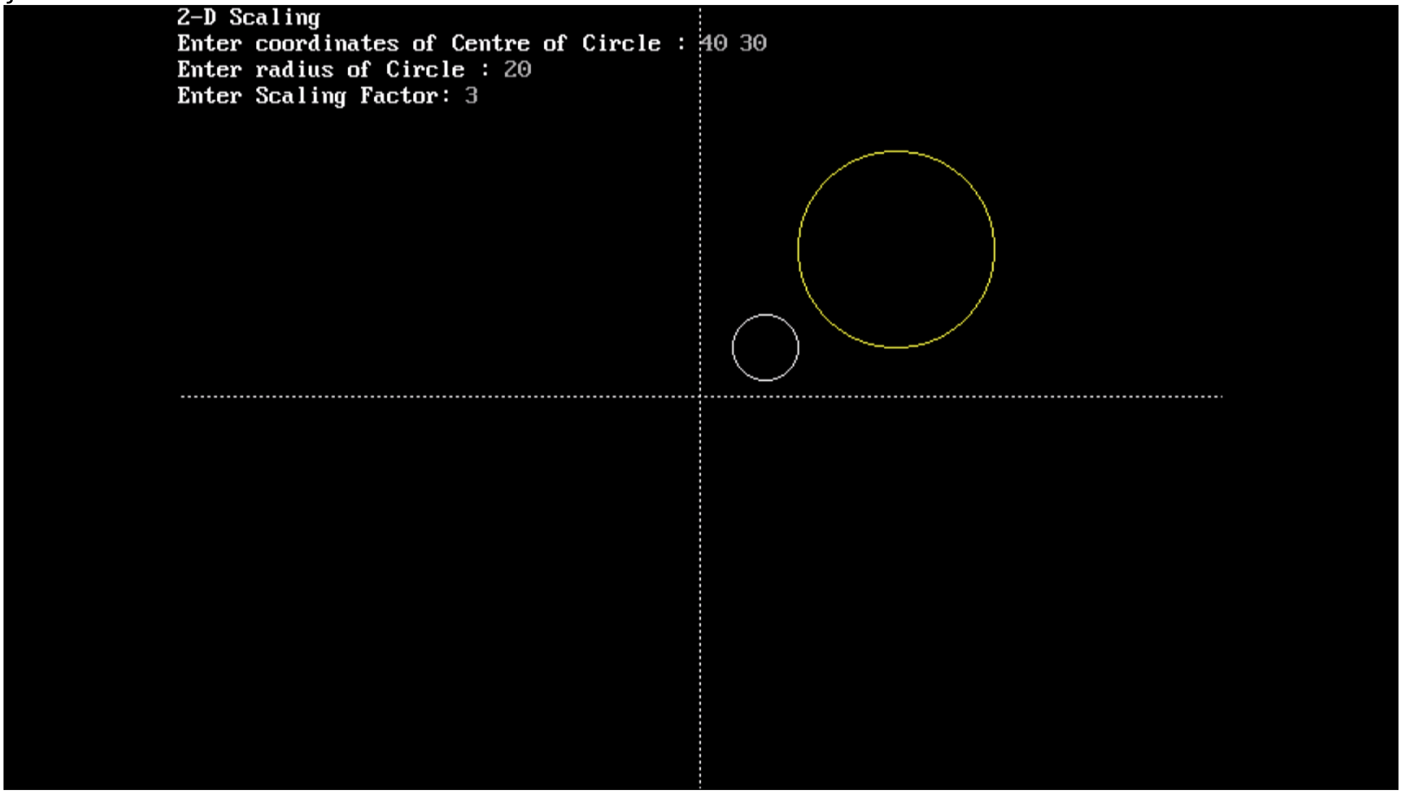
```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>

#define SIN(x) sin(x * 3.14159 / 180)
#define COS(x) cos(x * 3.14159 / 180)

#define ox (getmaxx() / 2)
#define oy (getmaxy() / 2)

void main()
{
    int gd = DETECT, gm, s, x1, y1, x2, y2;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

    printf("2-D Scaling\n");
    printf("Enter first coordinate : ");
    scanf("%d %d", &x1, &y1);
    printf("Enter second coordinate : ");
    scanf("%d %d", &x2, &y2);

    setlinestyle(DOTTED_LINE, 0, 1);
    line(0, oy, ox * 2, oy);
    line(ox, 0, ox, oy * 2);

    setlinestyle(SOLID_LINE, 0, 1);
    line(ox + x1, oy - y1, ox + x2, oy - y2);

    printf("Enter Scaling Factor: ");
    scanf("%d", &s);

    x1 *= s;
    y1 *= s;
    x2 *= s;
    y2 *= s;

    setcolor(YELLOW);
    line(ox + x1, oy - y1, ox + x2, oy - y2);

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



```
2-D Scaling
Enter first coordinate : 10 10
Enter second coordinate : 100 10
Enter Scaling Factor: 2
```

10. Scaling Circle in 2D

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

#define ox (getmaxx() / 2)
#define oy (getmaxy() / 2)

void main()
{
    int gd = DETECT, gm, s, x1, y1, r;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

    printf("2-D Scaling\n");
    printf("Enter coordinates of Centre of Circle : ");
    scanf("%d %d", &x1, &y1);
    printf("Enter radius of Circle : ");
    scanf("%d", &r);

    setlinestyle(DOTTED_LINE, 0, 1);
    line(0, oy, ox * 2, oy);
    line(ox, 0, ox, oy * 2);

    setlinestyle(SOLID_LINE, 0, 1);
    circle(ox + x1, oy - y1, r);

    printf("Enter Scaling Factor: ");
    scanf("%d", &s);

    x1 *= s;
    y1 *= s;
    r *= s;

    setcolor(YELLOW);
    circle(ox + x1, oy - y1, r);

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

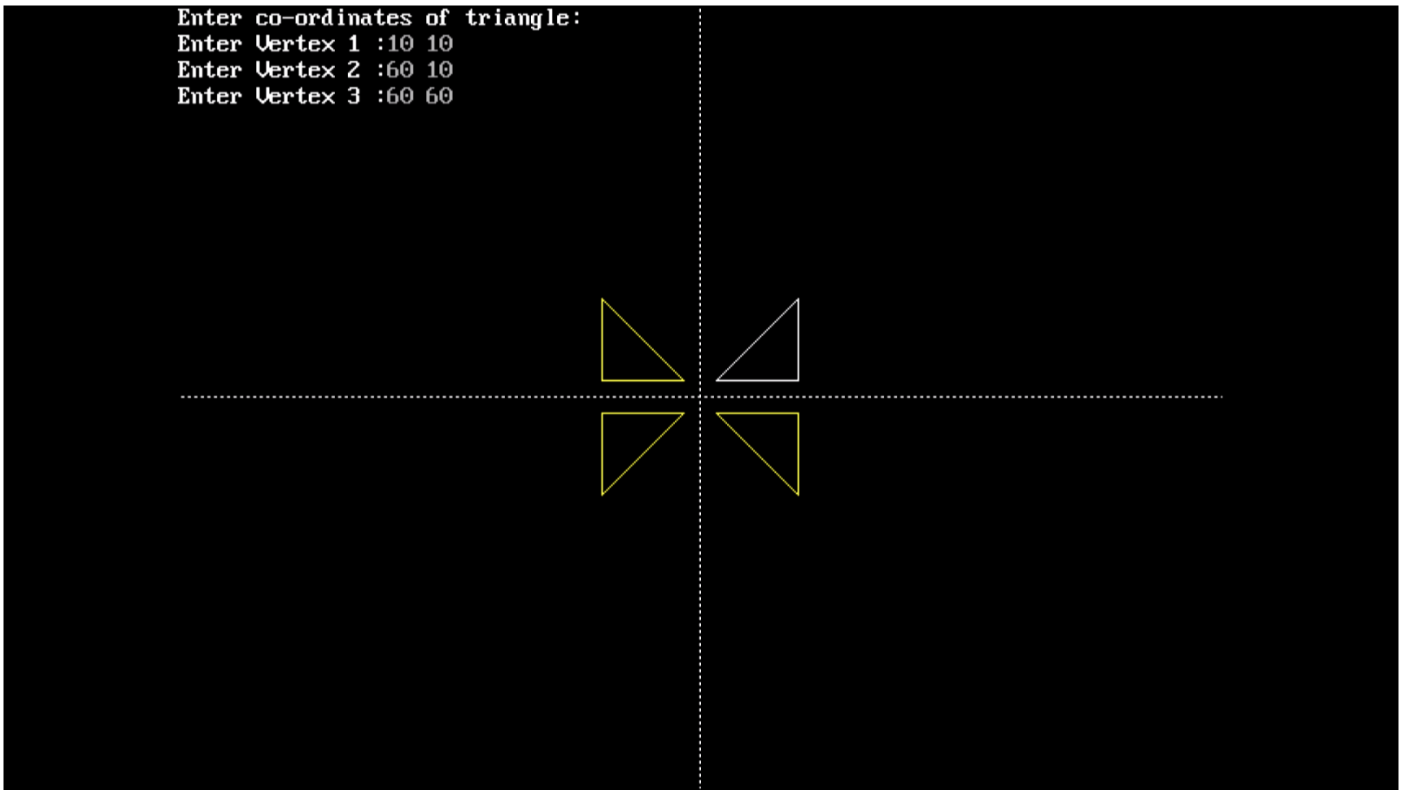


```
2-D Scaling
Enter coordinates of Centre of Circle : 40 30
Enter radius of Circle : 20
Enter Scaling Factor: 3
```

11. Reflection

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

#define ox (getmaxx() / 2)
#define oy (getmaxy() / 2)

void dispFigure(int v[3][3])
{
    int i = 0;
    while (i < 2)
    {
        line(ox + v[0][i], oy - v[1][i], ox + v[0][i + 1], oy - v[1][i + 1]);
        i++;
    }
    line(ox + v[0][i], oy - v[1][i], ox + v[0][0], oy - v[1][0]);
}

void ref_x(int v[3][3])
{
    int i, ref[3][3];
    for (i = 0; i <= 2; ++i)
    {
        ref[0][i] = v[0][i];
        ref[1][i] = -v[1][i];
    }
    dispFigure(ref);
}

void ref_y(int v[3][3])
{
    int i, ref[3][3];
    for (i = 0; i <= 2; ++i)
    {
        ref[0][i] = -v[0][i];
        ref[1][i] = v[1][i];
    }
    dispFigure(ref);
}

void refFigure(int v[3][3])
{
    int i, ref[3][3];
    for (i = 0; i <= 2; ++i)
    {
        ref[0][i] = -v[0][i];
        ref[1][i] = -v[1][i];
    }
    dispFigure(ref);
}

void main()
```

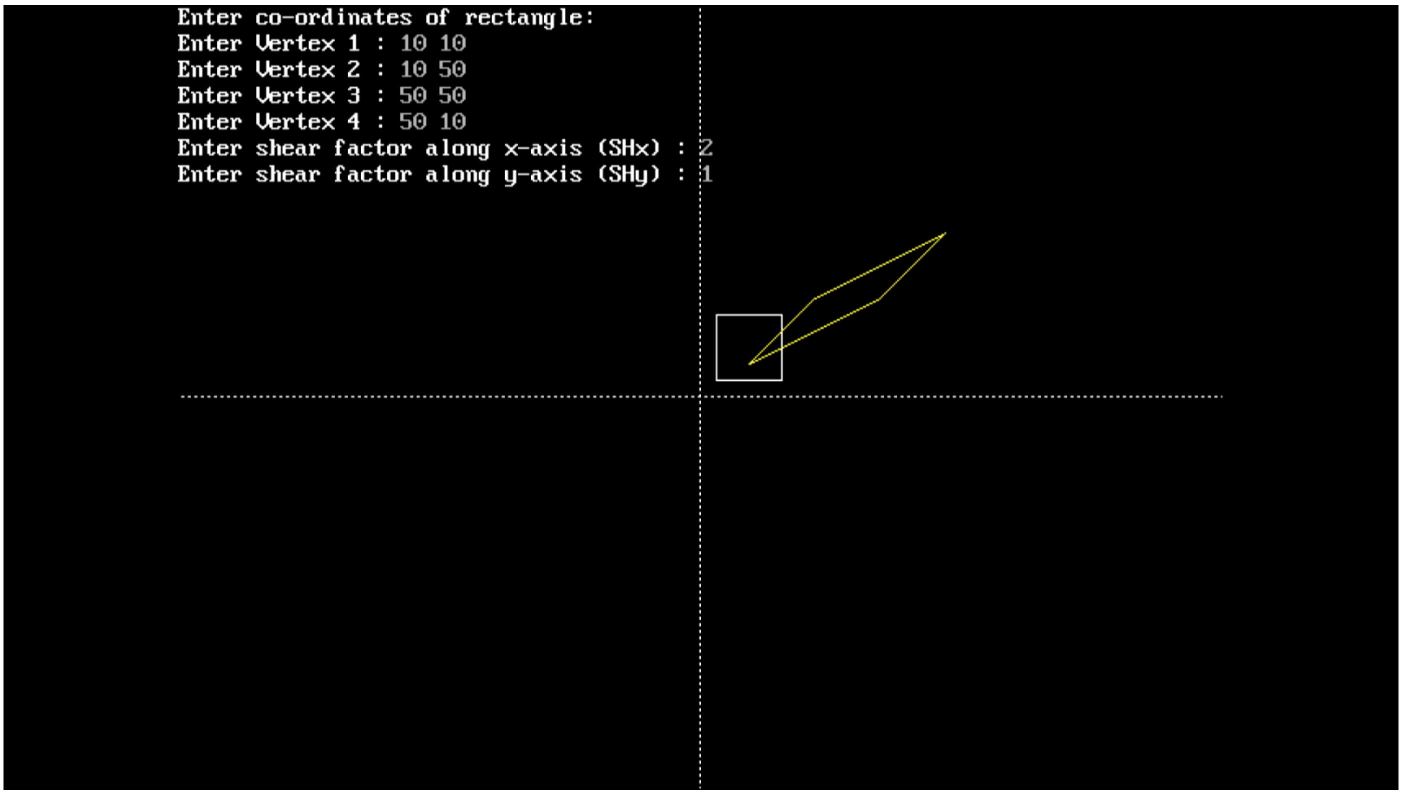
```
{
    int gd = DETECT, gm, i = 0, v[3][3];
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

    printf("Enter co-ordinates of triangle:\n");
    while (i < 3)
    {
        printf("Enter Vertex %d :", i + 1);
        scanf("%d %d", &v[0][i], &v[1][i]);
        v[2][i] = 1;
        i++;
    }

    setlinestyle(DOTTED_LINE, 0, 1);
    line(0, oy, ox * 2, oy);
    line(ox, 0, ox, oy * 2);
    setlinestyle(SOLID_LINE, 0, 1);

    dispFigure(v);
    setcolor(YELLOW);
    ref_x(v);
    ref_y(v);
    refFigure(v);

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



```
Enter co-ordinates of triangle:
Enter Vertex 1 :10 10
Enter Vertex 2 :60 10
Enter Vertex 3 :60 60
```

12. Shearing in 2D

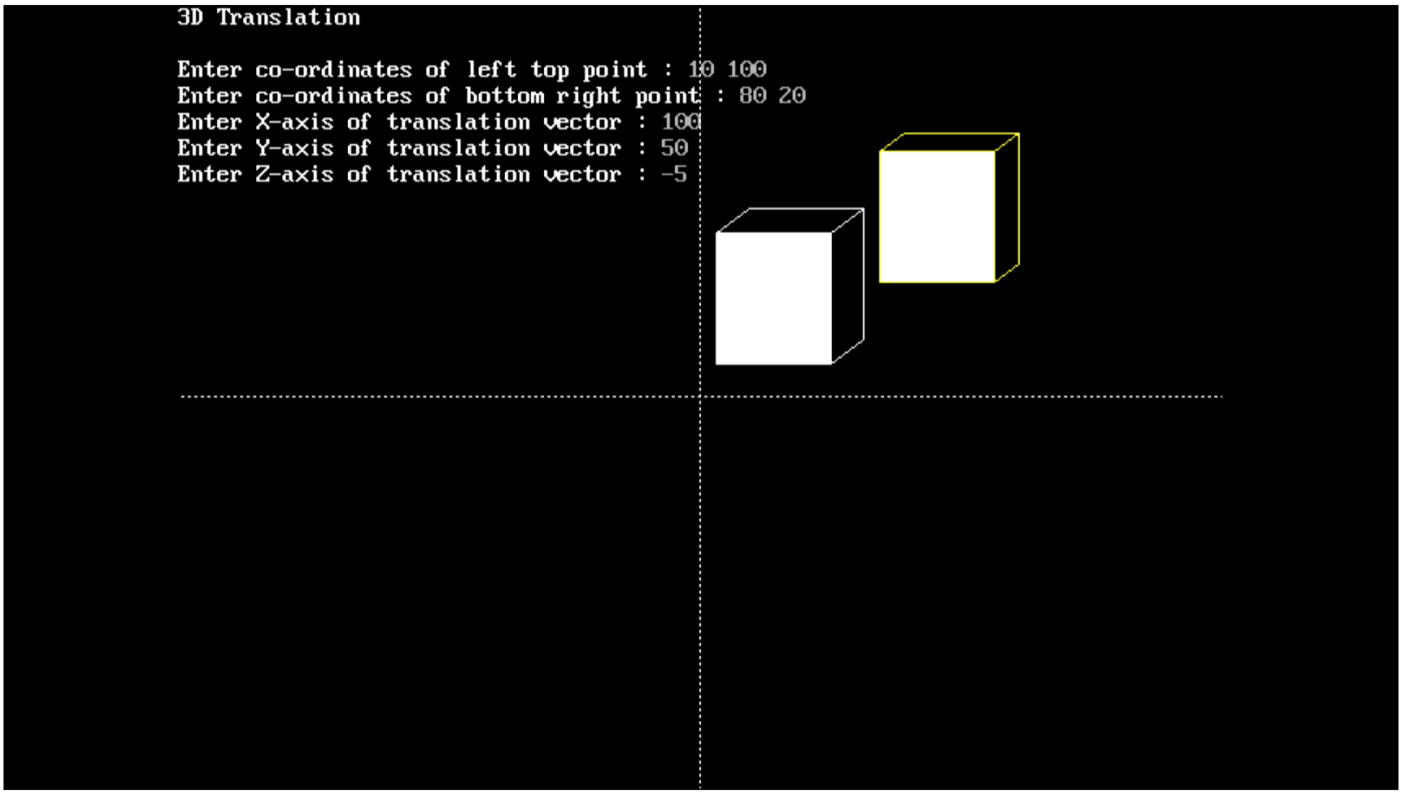
```c
#include <stdio.h>
#include <conio.h>
#include <graphics.h>

#define ox (getmaxx() / 2)
#define oy (getmaxy() / 2)

void dispFigure(int v[3][4])
{
    int i = 0;
    while (i < 3)
    {
        line(ox + v[0][i], oy - v[1][i], ox + v[0][i + 1], oy - v[1][i + 1]);
        i++;
    }
    line(ox + v[0][i], oy - v[1][i], ox + v[0][0], oy - v[1][0]);
}

int main()
{
    int gd = DETECT, gm, i, v[3][4], shx, shy, xnew, ynew;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

    printf("Enter co-ordinates of rectangle:\n");
    for (i = 0; i < 4; ++i)
    {
        printf("Enter Vertex %d : ", i + 1);
        scanf("%d %d", &v[0][i], &v[1][i]);
        v[2][i] = 1;
    }

    printf("Enter shear factor along x-axis (SHx) : ");
    scanf("%d", &shx);
    printf("Enter shear factor along y-axis (SHy) : ");
    scanf("%d", &shy);

    setlinestyle(DOTTED_LINE, 0, 1);
    line(0, oy, ox * 2, oy);
    line(ox, 0, ox, oy * 2);
    setlinestyle(SOLID_LINE, 0, 1);

    dispFigure(v);

    for (i = 0; i < 4; ++i)
    {
        xnew = v[0][i] + shx * v[1][i];
        ynew = v[1][i] + shy * v[0][i];

        v[0][i] = xnew;
        v[1][i] = ynew;
    }
    setcolor(YELLOW);
    dispFigure(v);
    getch();
}
```



```
Enter co-ordinates of rectangle:
Enter Vertex 1 : 10 10
Enter Vertex 2 : 10 50
Enter Vertex 3 : 50 50
Enter Vertex 4 : 50 10
Enter shear factor along x-axis (SHx) : 2
Enter shear factor along y-axis (SHy) : 1
```

13. 3D Translation

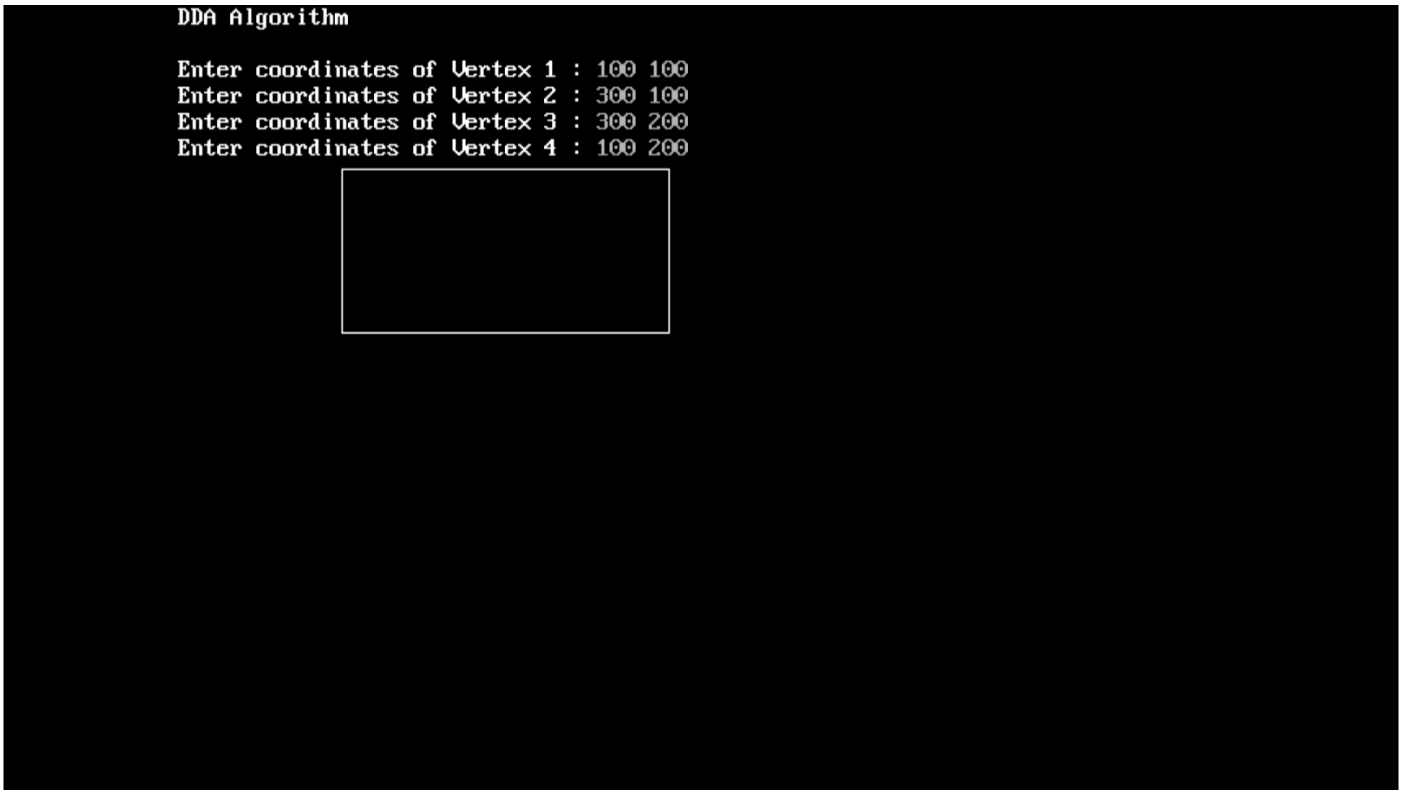
```c
#include <graphics.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>

#define ox (getmaxx() / 2)
#define oy (getmaxy() / 2)

void trans()
{
    int x, y, z, o, x1, x2, y1, y2;
    bar3d(ox + 50, oy - 100, ox + 100, oy - 50, 20, 1);
    delay(1000);
}

void main()
{
    int gd = DETECT, gm, X1, Y1, X2, Y2, Tx, Ty, Tz;

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

    printf("3D Translation\n\n");
    printf("Enter co-ordinates of left top point : ");
    scanf("%d %d", &X1, &Y1);
    printf("Enter co-ordinates of bottom right point : ");
    scanf("%d %d", &X2, &Y2);

    printf("Enter X-axis of translation vector : ");
    scanf("%d", &Tx);
    printf("Enter Y-axis of translation vector : ");
    scanf("%d", &Ty);
    printf("Enter Z-axis of translation vector : ");
    scanf("%d", &Tz);

    setlinestyle(DOTTED_LINE, 0, 1);
    line(0, oy, ox * 2, oy);
    line(ox, 0, ox, oy * 2);
    setlinestyle(SOLID_LINE, 0, 1);
    bar3d(ox + X1, oy - Y1, ox + X2, oy - Y2, 20, 1);

    setcolor(YELLOW);
    bar3d(ox + X1 + Tx, oy - Y1 - Ty, ox + X2 + Tx, oy - Y2 - Ty, 20 + Tz, 1);
    getch();
    closegraph();
}
```



```
3D Translation

Enter co-ordinates of left top point : 10 100
Enter co-ordinates of bottom right point : 80 20
Enter X-axis of translation vector : 100
Enter Y-axis of translation vector : 50
Enter Z-axis of translation vector : -5
```

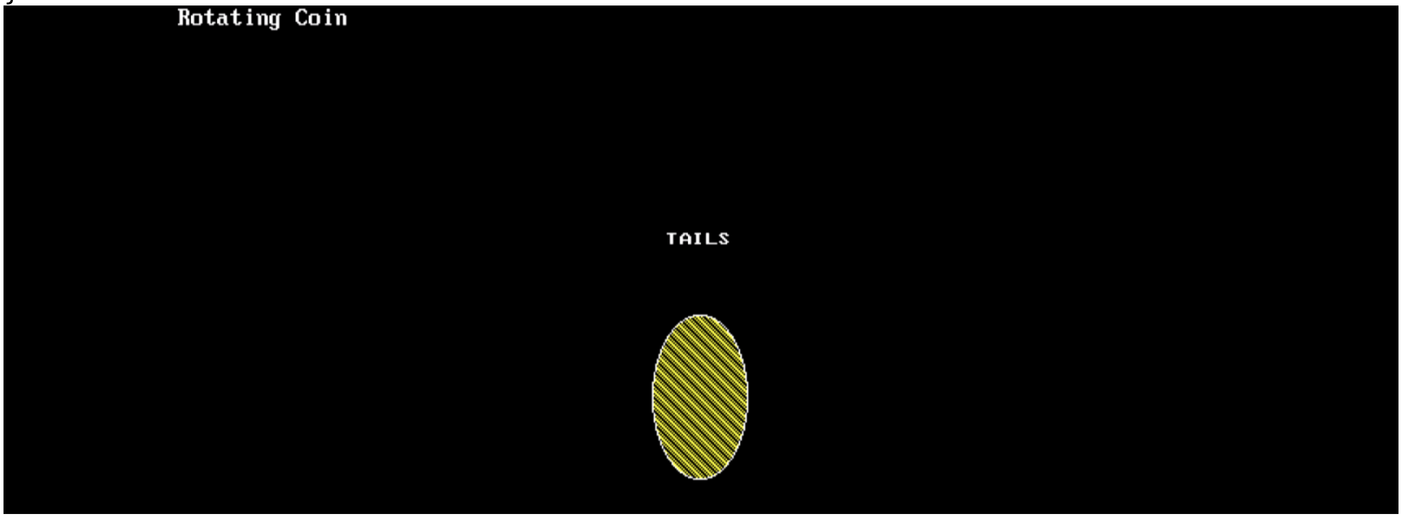
14. Draw a Rectangle using DDA Method

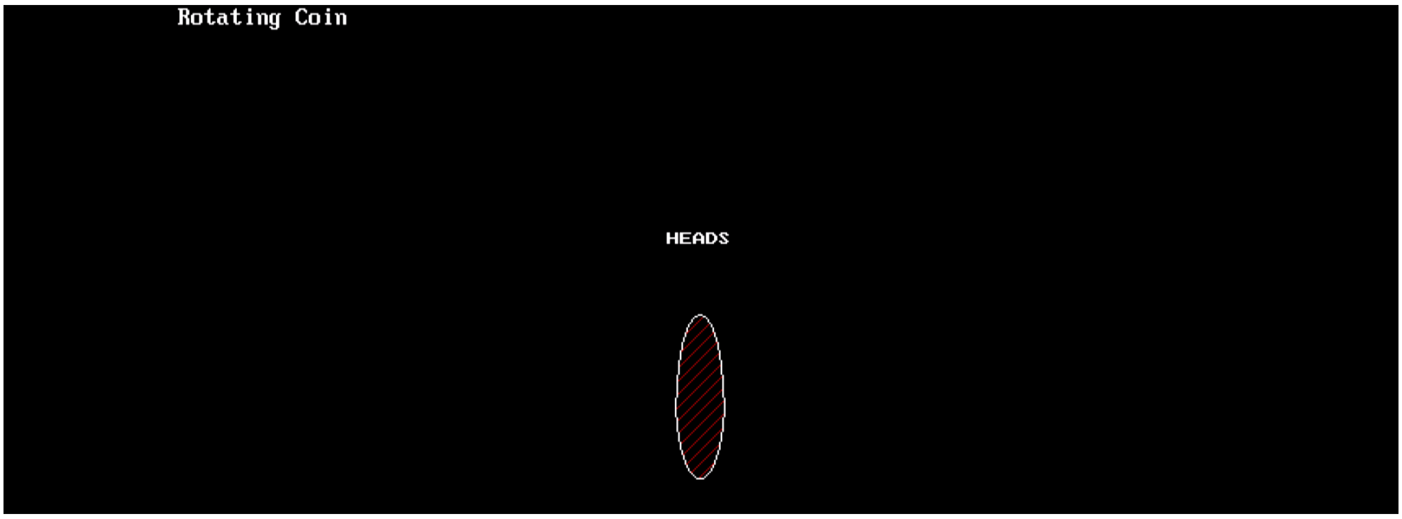
```c
#include <graphics.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
void draw_line(int x1, int y1, int x2, int y2)
{
    int dx, dy, step, x, y, i;
    dx = (float)(x2 - x1);
    dy = (float)(y2 - y1);

    step = (abs(dx) > abs(dy)) ? abs(dx) : abs(dy);

    dx = dx / step;
    dy = dy / step;

    x = x1;
    y = y1;

    for (i = 0; i <= step; ++i)
    {
        putpixel(x, y, WHITE);
        x += dx;
        y += dy;
        delay(50);
    }
}
int main()
{
    int x1, y1, x2, y2, x3, y3, x4, y4;
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

    printf("DDA Algorithm\n\n");
    printf("Enter coordinates of Vertex 1 : ");
    scanf("%d %d", &x1, &y1);
    printf("Enter coordinates of Vertex 2 : ");
    scanf("%d %d", &x2, &y2);
    printf("Enter coordinates of Vertex 3 : ");
    scanf("%d %d", &x3, &y3);
    printf("Enter coordinates of Vertex 4 : ");
    scanf("%d %d", &x4, &y4);

    draw_line(x1, y1, x2, y2);
    draw_line(x2, y2, x3, y3);
    draw_line(x3, y3, x4, y4);
    draw_line(x4, y4, x1, y1);

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



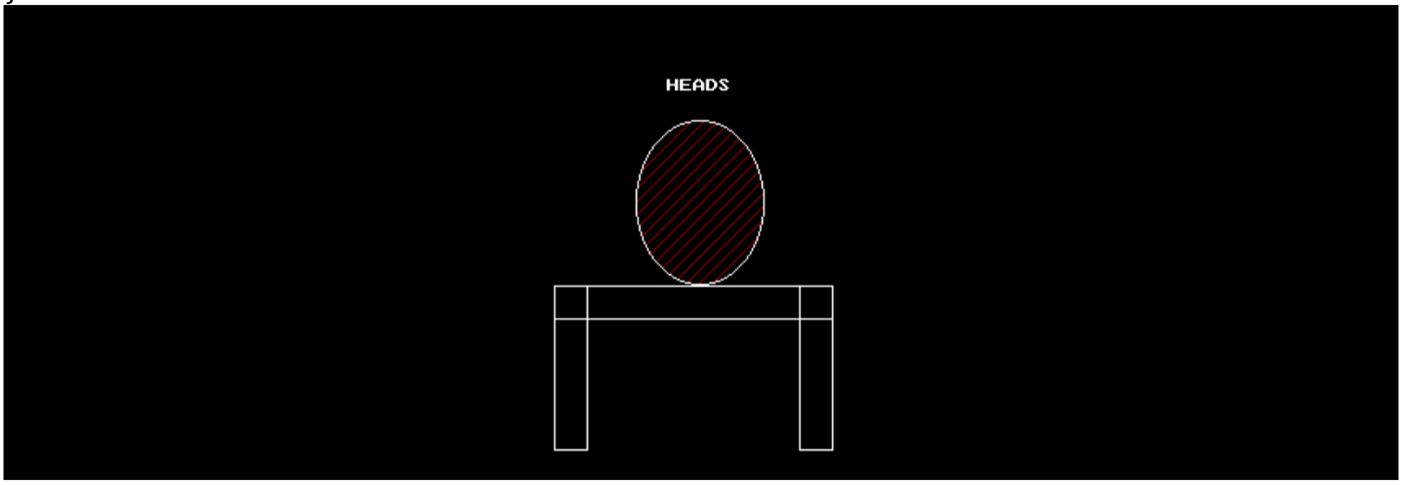
```
DDA Algorithm

Enter coordinates of Vertex 1 : 100 100
Enter coordinates of Vertex 2 : 300 100
Enter coordinates of Vertex 3 : 300 200
Enter coordinates of Vertex 4 : 100 200
```

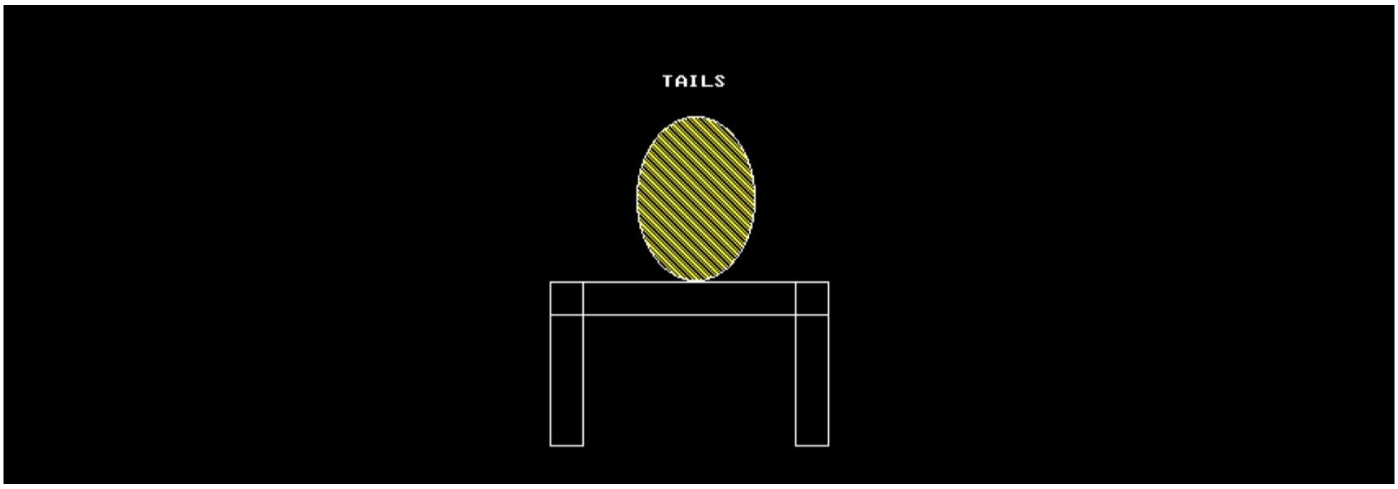
15. Program to rotate a coin

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

#define ox (getmaxx() / 2)
#define oy (getmaxy() / 2)

void main()
{

    int gd = DETECT, gm, xr, yr = 50, side = 0;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

    printf("Rotating Coin");
    while (!kbhit())
    {
        if (side)
        {
            setcolor(BLACK);
            outtextxy(ox - 20, oy - 100, "TAILS");
            setcolor(WHITE);
            outtextxy(ox - 20, oy - 100, "HEADS");
            for (xr = 0; xr <= 50; ++xr)
            {
                setfillstyle(LTSLASH_FILL, RED);
                fillellipse(ox, oy, xr, yr);
                delay(30);
                setcolor(BLACK);
                setfillstyle(SOLID_FILL, BLACK);
                fillellipse(ox, oy, xr, yr);
                setcolor(WHITE);
            }
            for (xr = 50; xr >= 0; --xr)
            {
                setfillstyle(LTSLASH_FILL, RED);
                fillellipse(ox, oy, xr, yr);
                delay(30);
                setcolor(BLACK);
                setfillstyle(SOLID_FILL, BLACK);
                fillellipse(ox, oy, xr, yr);
                setcolor(WHITE);
            }
            side = !side;
        }
        else
        {
            setcolor(BLACK);
            outtextxy(ox - 20, oy - 100, "HEADS");
            setcolor(WHITE);
            outtextxy(ox - 20, oy - 100, "TAILS");
            for (xr = 0; xr <= 50; ++xr)
            {
                setfillstyle(LTBKSLASH_FILL, YELLOW);
                fillellipse(ox, oy, xr, yr);
                delay(30);
                setcolor(BLACK);
                setfillstyle(SOLID_FILL, BLACK);
                fillellipse(ox, oy, xr, yr);
                setcolor(WHITE);
            }
            for (xr = 50; xr >= 0; --xr)
            {
                setfillstyle(LTBKSLASH_FILL, YELLOW);
                fillellipse(ox, oy, xr, yr);
                delay(30);
                setcolor(BLACK);
                setfillstyle(SOLID_FILL, BLACK);
                fillellipse(ox, oy, xr, yr);
                setcolor(WHITE);
            }
            side = !side;
        }
    }
}
```




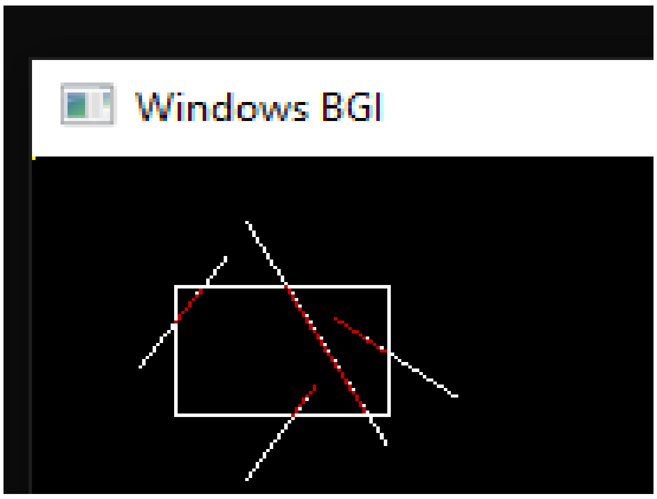
16. Program to rotate a coin on table

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

#define ox (getmaxx() / 2)
#define oy (getmaxy() / 2)

void main()
{

    int gd = DETECT, gm, xr, yr = 50, side = 0;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

    printf("Rotating Coin");

    rectangle(230, 290, 400, 310);
    rectangle(230, 290, 250, 390);
    rectangle(380, 290, 400, 390);

    while (!kbhit())
    {
        if (side)
        {
            setcolor(BLACK);
            outtextxy(ox - 20, oy - 75, "TAILS");
            setcolor(WHITE);
            outtextxy(ox - 20, oy - 75, "HEADS");
            for (xr = 0; xr <= 50; ++xr)
            {
                setfillstyle(LTSLASH_FILL, RED);
                fillellipse(ox, oy, xr, yr);
                delay(30);
                setcolor(BLACK);
                setfillstyle(SOLID_FILL, BLACK);
                fillellipse(ox, oy, xr, yr);
                setcolor(WHITE);
            }
            for (xr = 50; xr >= 0; --xr)
            {
                setfillstyle(LTSLASH_FILL, RED);
                fillellipse(ox, oy, xr, yr);
                delay(30);
                setcolor(BLACK);
                setfillstyle(SOLID_FILL, BLACK);
                fillellipse(ox, oy, xr, yr);
                setcolor(WHITE);
            }
            side = !side;
        }
        else
        {
            setcolor(BLACK);
            outtextxy(ox - 20, oy - 75, "HEADS");
            setcolor(WHITE);
            outtextxy(ox - 20, oy - 75, "TAILS");
            for (xr = 0; xr <= 50; ++xr)
            {
                setfillstyle(LTBKSLASH_FILL, YELLOW);
                fillellipse(ox, oy, xr, yr);
                delay(30);
                setcolor(BLACK);
                setfillstyle(SOLID_FILL, BLACK);
                fillellipse(ox, oy, xr, yr);
                setcolor(WHITE);
            }
            for (xr = 50; xr >= 0; --xr)
            {
                setfillstyle(LTBKSLASH_FILL, YELLOW);
                fillellipse(ox, oy, xr, yr);
                delay(30);
                setcolor(BLACK);
                setfillstyle(SOLID_FILL, BLACK);
                fillellipse(ox, oy, xr, yr);
                setcolor(WHITE);
            }
            side = !side;
        }
    }
}
```




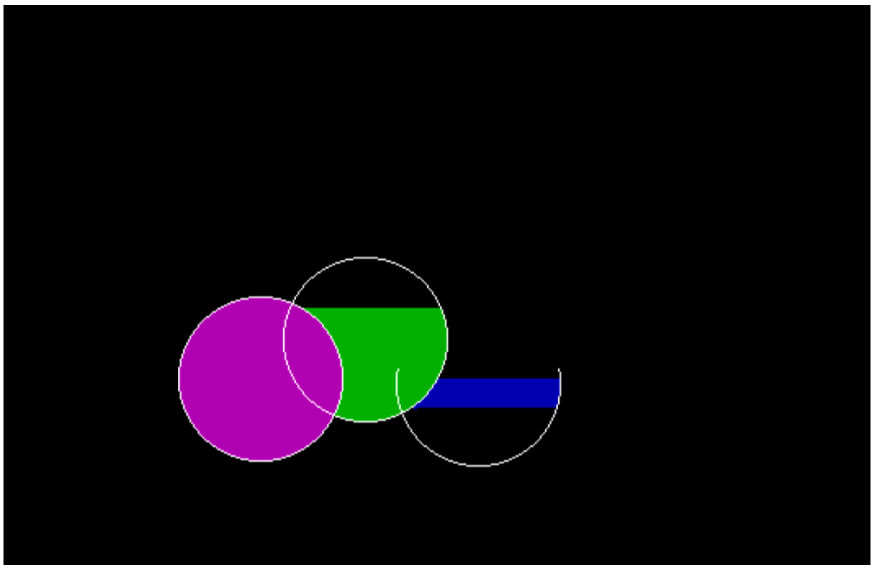
17. Cohen-Sutherland Algorithm

```cpp
#include <bits/stdc++.h>
#include <graphics.h>
using namespace std;

int xmin, xmax, ymin, ymax;

struct lines {
    int x1, y1, x2, y2;
};

int sign(int x)
{
    if (x > 0)
        return 1;
    else
        return 0;
}

void clip(struct lines mylines)
{
    int bits[4], bite[4], i, var;
    setcolor(RED);

    bits[0] = sign(xmin - mylines.x1);
    bite[0] = sign(xmin - mylines.x2);
    bits[1] = sign(mylines.x1 - xmax);
    bite[1] = sign(mylines.x2 - xmax);
    bits[2] = sign(ymin - mylines.y1);
    bite[2] = sign(ymin - mylines.y2);
    bits[3] = sign(mylines.y1 - ymax);
    bite[3] = sign(mylines.y2 - ymax);

    string initial = "", end = "", temp = "";

    for (i = 0; i < 4; i++) {
        if (bits[i] == 0)
            initial += '0';
        else
            initial += '1';
    }
    for (i = 0; i < 4; i++) {
        if (bite[i] == 0)
            end += '0';
        else
            end += '1';
    }


    float m = (mylines.y2 - mylines.y1) / (float)(mylines.x2 - mylines.x1);
    float c = mylines.y1 - m * mylines.x1;

    if (initial == end && end == "0000") {
        line(mylines.x1, mylines.y1, mylines.x2, mylines.y2);
        return;
    }

    else {
        for (i = 0; i < 4; i++) {

            int val = (bits[i] & bite[i]);
            if (val == 0)
                temp += '0';
            else
                temp += '1';
        }
        if (temp != "0000")
            return;

        for (i = 0; i < 4; i++) {
            if (bits[i] == bite[i])
                continue;
            if (i == 0 && bits[i] == 1) {
                var = round(m * xmin + c);
                mylines.y1 = var;
                mylines.x1 = xmin;
            }
            if (i == 0 && bite[i] == 1) {
                var = round(m * xmin + c);
                mylines.y2 = var;
                mylines.x2 = xmin;
            }
            if (i == 1 && bits[i] == 1) {
                var = round(m * xmax + c);
                mylines.y1 = var;
                mylines.x1 = xmax;
            }
            if (i == 1 && bite[i] == 1) {
                var = round(m * xmax + c);
                mylines.y2 = var;
                mylines.x2 = xmax;
            }
            if (i == 2 && bits[i] == 1) {
                var = round((float)(ymin - c) / m);
                mylines.y1 = ymin;
                mylines.x1 = var;
            }
            if (i == 2 && bite[i] == 1) {
                var = round((float)(ymin - c) / m);
                mylines.y2 = ymin;
                mylines.x2 = var;
            }
            if (i == 3 && bits[i] == 1) {
                var = round((float)(ymax - c) / m);
                mylines.y1 = ymax;
                mylines.x1 = var;
            }
            if (i == 3 && bite[i] == 1) {
                var = round((float)(ymax - c) / m);
                mylines.y2 = ymax;
                mylines.x2 = var;
            }
            bits[0] = sign(xmin - mylines.x1);
            bite[0] = sign(xmin - mylines.x2);
            bits[1] = sign(mylines.x1 - xmax);
            bite[1] = sign(mylines.x2 - xmax);
            bits[2] = sign(ymin - mylines.y1);
            bite[2] = sign(ymin - mylines.y2);
            bits[3] = sign(mylines.y1 - ymax);
            bite[3] = sign(mylines.y2 - ymax);
        }
        initial = "", end = "";
        for (i = 0; i < 4; i++) {
            if (bits[i] == 0)
                initial += '0';
            else
                initial += '1';
        }
        for (i = 0; i < 4; i++) {
            if (bite[i] == 0)
                end += '0';
            else
                end += '1';
        }
        if (initial == end && end == "0000") {
            line(mylines.x1, mylines.y1, mylines.x2, mylines.y2);
            return;
        }
        else
            return;
    }
}

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

    xmin = 40;
    xmax = 100;
    ymin = 40;
    ymax = 80;

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

    line(xmin, ymin, xmax, ymin);
    line(xmax, ymin, xmax, ymax);
    line(xmax, ymax, xmin, ymax);
    line(xmin, ymax, xmin, ymin);

    struct lines mylines[4];
    mylines[0].x1 = 30;
    mylines[0].y1 = 65;
    mylines[0].x2 = 55;
    mylines[0].y2 = 30;

    mylines[1].x1 = 60;
    mylines[1].y1 = 20;
    mylines[1].x2 = 100;
    mylines[1].y2 = 90;

    mylines[2].x1 = 60;
    mylines[2].y1 = 100;
    mylines[2].x2 = 80;
    mylines[2].y2 = 70;

    mylines[3].x1 = 85;
    mylines[3].y1 = 50;
    mylines[3].x2 = 120;
    mylines[3].y2 = 75;

    for (int i = 0; i < 4; i++) {
        line(mylines[i].x1, mylines[i].y1,
             mylines[i].x2, mylines[i].y2);
        delay(1000);
    }

    for (int i = 0; i < 4; i++) {
        clip(mylines[i]);
        delay(1000);
    }
    delay(4000);
    getch();
    closegraph();
    return 0;
}
```



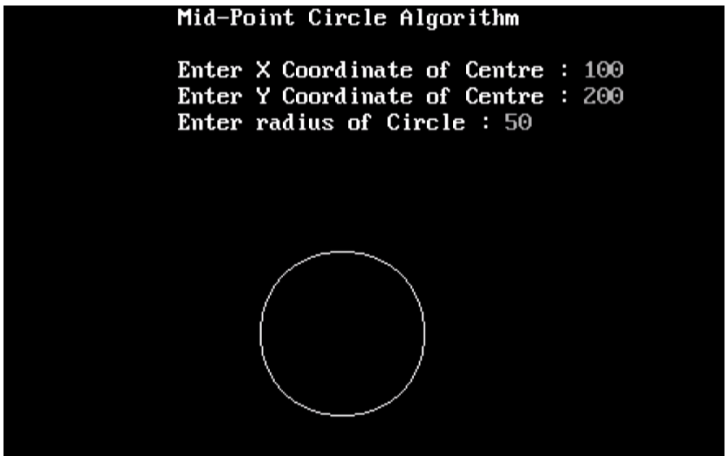
18. Program to draw Flying Balloons

```
#include <graphics.h>
#include <conio.h>
#include <dos.h>
int main()
{
    int gd = DETECT, gm, i, j;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI\\");
    for (j = 0; j < 5; j++)
    {
        for (i = 0; i < 600; i++)
        {
            setfillstyle(SOLID_FILL, MAGENTA);
            circle(50, 390 - i, 50);
            floodfill(50, 390 - i, WHITE);
            setfillstyle(SOLID_FILL, GREEN);
            circle(90 + i, 390 - 2 * i, 50);
            floodfill(90 + i, 390 - 2 * i, WHITE);
            setfillstyle(SOLID_FILL, BLUE);
            circle(135 + 2 * i, 393 - i, 50);
            floodfill(130 + 2 * i, 390 - i, WHITE);
            setfillstyle(SOLID_FILL, WHITE);
            circle(195 + 2 * i, 393 - 3 * i, 50);
            floodfill(195 + 2 * i, 393 - 3 * i, WHITE);
            delay(5);
            cleardevice();
        }
    }
    getch();
}
```



19. Mid-Point Circle Algorithm

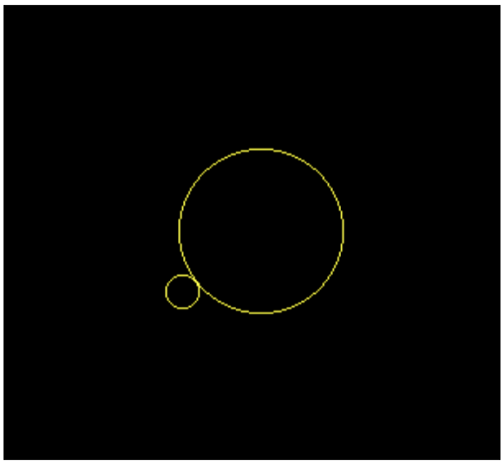
```c
#include <graphics.h>
#include <conio.h>
#include <stdio.h>

void main()
{

    int gd = DETECT, gm, x, y, a, b, r, p;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

    printf("Mid-Point Circle Algorithm\n\n");
    printf("Enter X Coordinate of Centre : ");
    scanf("%d", &a);
    printf("Enter Y Coordinate of Centre : ");
    scanf("%d", &b);
    printf("Enter radius of Circle : ");
    scanf("%d", &r);

    x = 0;
    y = r;
    p = 1 - r;

    while (x <= y)
    {
        putpixel(a + x, b - y, WHITE);
        putpixel(a - x, b + y, WHITE);
        putpixel(a + x, b + y, WHITE);
        putpixel(a - x, b - y, WHITE);
        putpixel(a + y, b - x, WHITE);
        putpixel(a - y, b + x, WHITE);
        putpixel(a + y, b + x, WHITE);
        putpixel(a - y, b - x, WHITE);

        if (p < 0)
        {
            p = p + (2 * x) + 1;
        }
        else
        {
            p = p + (2 * (x - y)) + 1;
            y--;
        }
        x++;
    }
    getch();
    closegraph();
}
```



```
Mid-Point Circle Algorithm

Enter X Coordinate of Centre : 100
Enter Y Coordinate of Centre : 200
Enter radius of Circle : 50
```

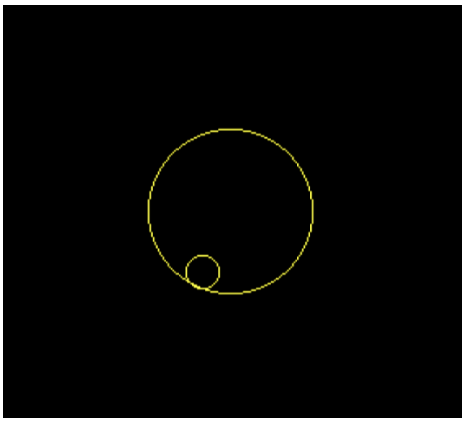
20. Program to rotate a circle outside another circle

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

#define SIN(x) sin(x * 3.14159 / 180)
#define COS(x) cos(x * 3.14159 / 180)

#define xc (getmaxx() / 2)
#define yc (getmaxy() / 2)

void rotate_circle(int x, int y, int s)
{
    int x1 = floor((x)*COS(s) - (y)*SIN(s));
    int y1 = floor((y)*COS(s) + (x)*SIN(s));
    circle(x1 + xc, yc - y1, 10);
}

void main()
{
    double angle = 0, theta;
    int i, a, gd = DETECT, gm, r = 50;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

    while (!kbhit())
    {
        cleardevice();
        setcolor(YELLOW);
        circle(xc, yc, r);

        rotate_circle(60, 0, angle);
        angle++;
        delay(20);
    }
    getch();
    closegraph();
}
```



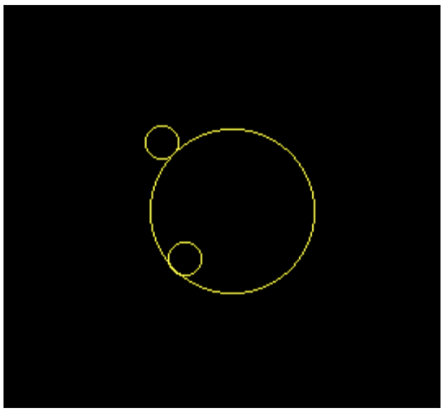
21. Program to rotate a circle inside another circle

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

#define SIN(x) sin(x * 3.14159 / 180)
#define COS(x) cos(x * 3.14159 / 180)

#define xc (getmaxx() / 2)
#define yc (getmaxy() / 2)

void rotate_circle(int x, int y, int s)
{
    int x1 = floor(x * COS(s) + y * SIN(s));
    int y1 = floor(y * COS(s) - x * SIN(s));
    circle(x1 + xc, yc - y1, 10);
}

void main()
{
    double angle = 0, theta;
    int i, a, gd = DETECT, gm, r = 50;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

    while (!kbhit())
    {
        cleardevice();
        setcolor(YELLOW);
        circle(xc, yc, r);

        rotate_circle(40, 0, angle);
        angle++;
        delay(20);
    }
    getch();
    closegraph();
}
```



22. Program to rotate a circle inside and outside another circle alternatively

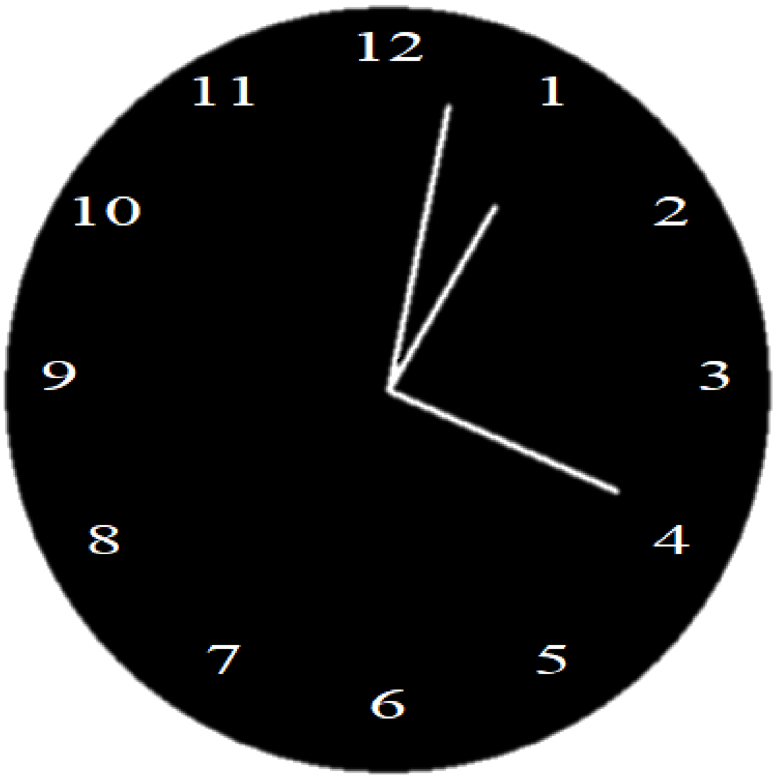
```c
#include <stdio.h>
#include <graphics.h>
#include <math.h>
#include <conio.h>

#define SIN(x) sin(x * 3.14159 / 180)
#define COS(x) cos(x * 3.14159 / 180)

#define xc (getmaxx() / 2)
#define yc (getmaxy() / 2)

void rotate_circle(int x, int y, int s)
{
    int x1 = floor(x * COS(s) + y * SIN(s));
    int y1 = floor(y * COS(s) - x * SIN(s));
    circle(x1 + xc, yc - y1, 10);
}

void main()
{
    double angle = 0, theta;
    int i, a, gd = DETECT, gm, r = 50;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

    while (!kbhit())
    {
        cleardevice();
        circle(xc, yc, r);
        setcolor(YELLOW);

        rotate_circle(40, 0, angle);
        rotate_circle(60, 0, -angle);
        angle++;
        delay(20);
    }
    getch();
    closegraph();
}
```



23. Analog Clock

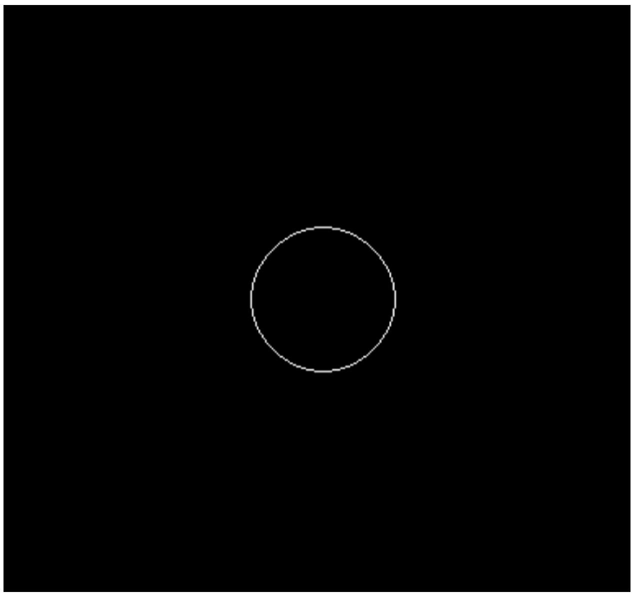
```c
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <math.h>
#include <time.h>
#include <graphics.h>
#include <dos.h>

void minSecCalc(int xrad, int midx, int midy, int x[60], int y[60])
{
    int i, j = 45;
    for (i = 360; i >= 0; i = i - 6)
    {
        x[j] = midx - (xrad * cos((i * 3.14) / 180));
        y[j--] = midy - (xrad * sin((i * 3.14) / 180));
        j = (j == -1) ? 59 : j;
    }
    return;
}

void calcPoints(int radius, int midx, int midy, int x[12], int y[12])
{
    int x1, y1;
    /* 90, 270, 0, 180 degrees */
    x[0] = midx, y[0] = midy - radius;
    x[6] = midx, y[6] = midy + radius;
    x[3] = midx + radius, y[3] = midy;
    x[9] = midx - radius, y[9] = midy;

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

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

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

    initgraph(&gdriver, &gmode, "C:/TURBOC3/BGI");
    err = graphresult();


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

    radius = 200;

    calcPoints(radius - 30, midx, midy, x, y);

    calcPoints(radius - 90, midx, midy, hrx, hry);

    minSecCalc(radius - 50, midx, midy, minx, miny);

    minSecCalc(radius - 70, midx, midy, secx, secy);

    while (!kbhit())
    {
        setlinestyle(SOLID_LINE, 1, 3);
        settextstyle(TRIPLEX_FONT, 0, 3);

        circle(midx, midy, radius);

        for (j = 0; j < 12; j++)
        {
            if (j == 0)
            {
                sprintf(str, "%d", 12);
            }
            else
            {
                sprintf(str, "%d", j);
            }
            settextjustify(CENTER_TEXT, CENTER_TEXT);
            moveto(x[j], y[j]);
            outtext(str);
        }

        t1 = time(NULL);
        data = localtime(&t1);

        sec = data->tm_sec % 60;
        line(midx, midy, secx[sec], secy[sec]);

        min = data->tm_min % 60;
        line(midx, midy, minx[min], miny[min]);
```

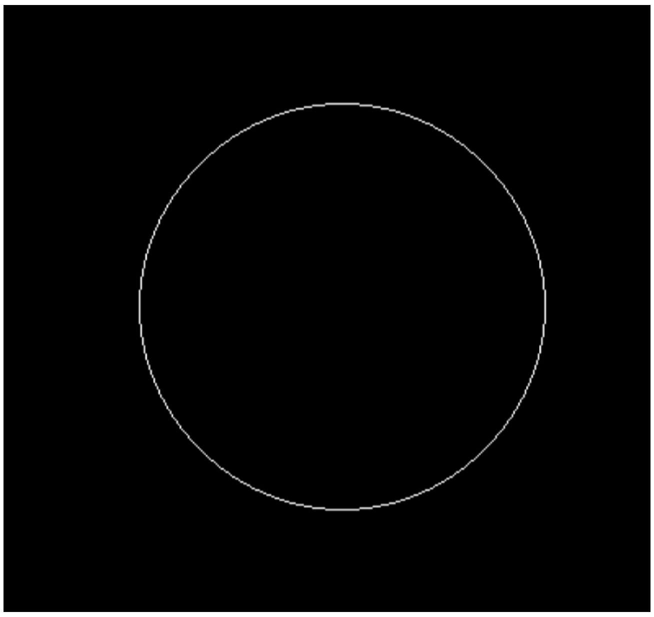
```
        hr = data->tm_hour % 12;
        line(midx, midy, hrx[hr], hry[hr]);
        delay(1000);
        cleardevice();
    }

    getch();

    closegraph();
    return 0;
}
```



24. Program to show changing radius of circle

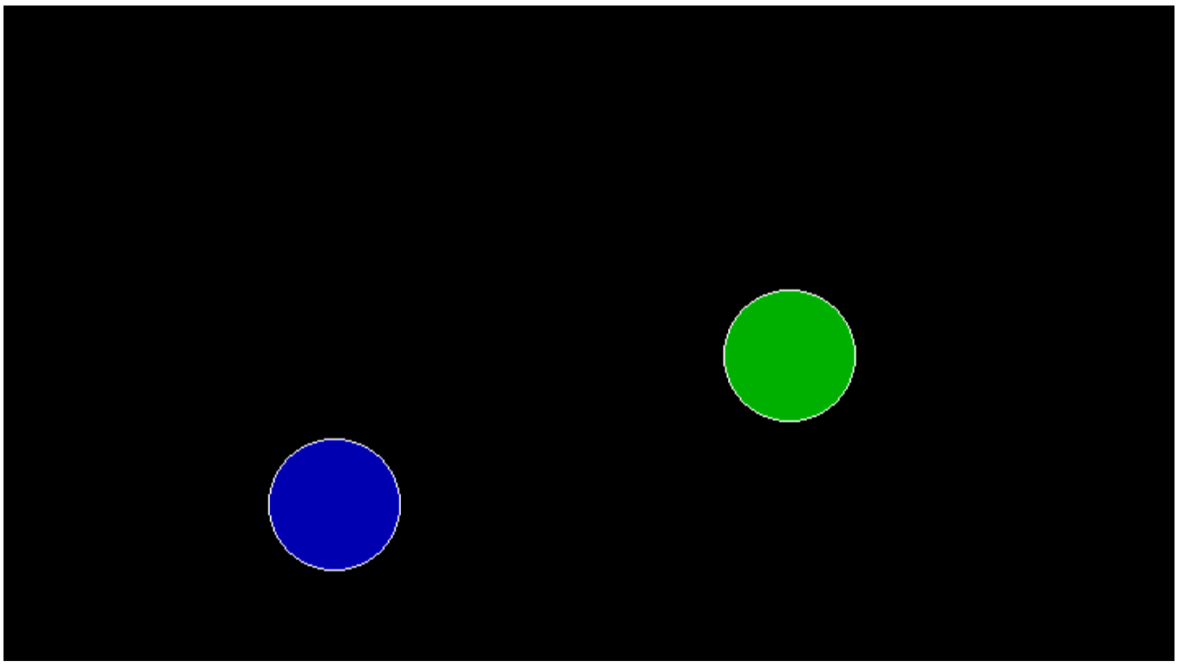
```
#include <conio.h>
#include <graphics.h>

void drawPixel(int xc, int yc, int x, int y, int c)
{
    putpixel(xc + x, yc + y, c);
    putpixel(xc + x, yc - y, c);
    putpixel(xc - x, yc + y, c);
    putpixel(xc - x, yc - y, c);
    putpixel(xc + y, yc + x, c);
    putpixel(xc - y, yc + x, c);
    putpixel(xc + y, yc - x, c);
    putpixel(xc - y, yc - x, c);
}

void printCircle(int xc, int yc, int r, int c)
{
    int x = 0, y = r, d;
    d = 3 - 2 * r;

    drawPixel(xc, yc, x, y, c);
    while (y >= x)
    {   x++;
        if (d > 0){
            y--;
            d = d + 4 * (x - y) + 10;
        }
        else{
            d = d + 4 * x + 6;
        }
        drawPixel(xc, yc, x, y, c);
    }
}

void main()
{
    int gd = DETECT, gm, r = 0, xc, yc, c = 1;
    initgraph(&gd, &gm, "C:\\Turboc3\\BGI");

    xc = getmaxx() / 2;
    yc = getmaxy() / 2;

    while (!kbhit())
    {
        if (c == 1){
            printCircle(xc, yc, r - 1, 0);
            printCircle(xc, yc, r, 15);
            r += 1;
        }
        else{
            printCircle(xc, yc, r + 1, 0);
            printCircle(xc, yc, r, 15);
            r -= 1;
        }

        if (r == 240) {  c = 0; }
        if (r == 1)   {  c = 1; }

        delay(50);
    }
    getch();
    closegraph();
}
```





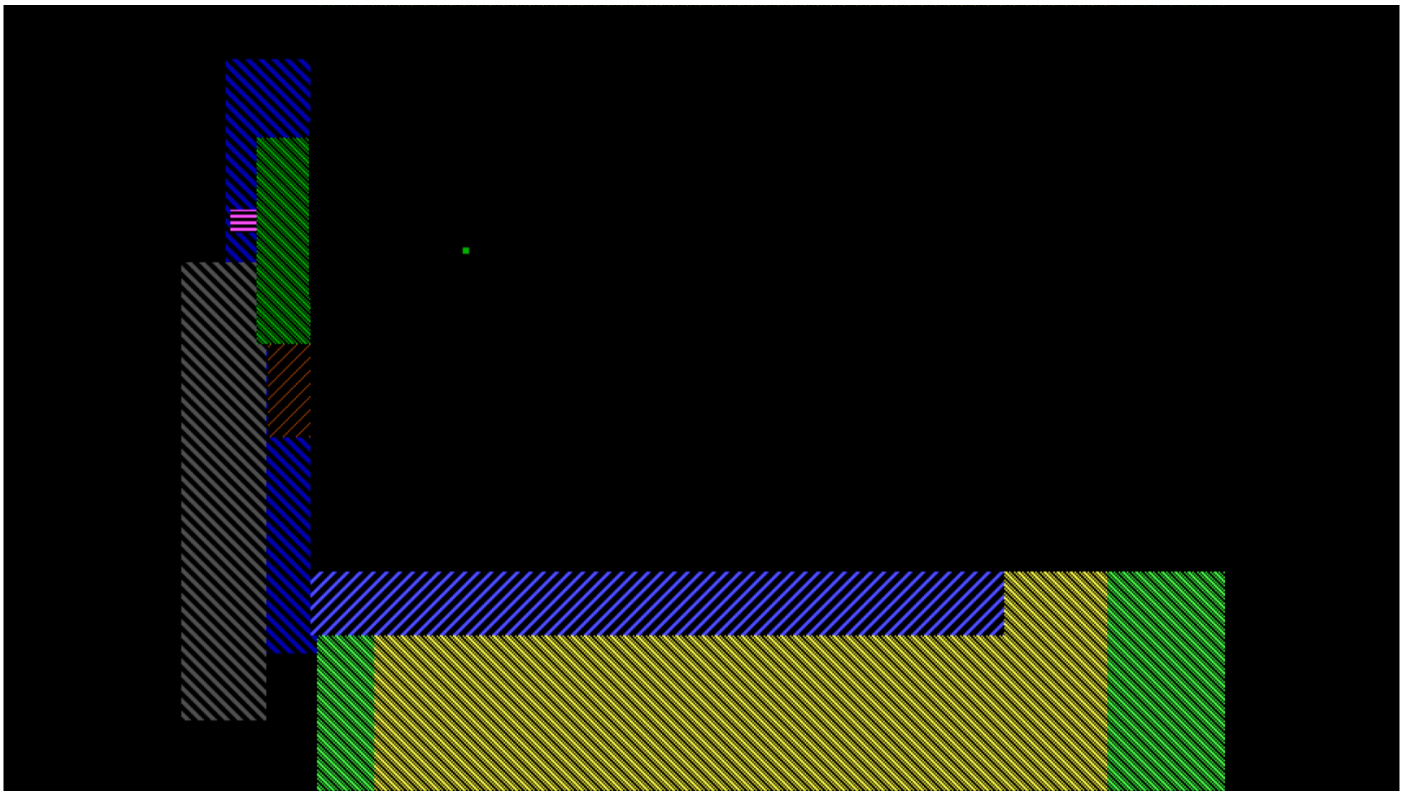
25. Program for diagonals bouncing ball

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

void main(){
    int gd=DETECT,gm,x1,y1,f1=1,x2,y2,f2=1;
    initgraph(&gd,&gm,"C:\\Turboc3\\BGI");

    x1=getmaxx()/2;
    y1=getmaxy()/2;
    x2=41;
    y2=getmaxy()-41;

    while(!kbhit()){
        if(y1>=getmaxy()-40 || y1<=40) f1=-f1;
        if(y2>=getmaxy()-40 || y2<=40) f2=-f2;

        setfillstyle(SOLID_FILL, GREEN);
        fillellipse(x1,y1,40,40);

        setfillstyle(SOLID_FILL, BLUE);
        fillellipse(x2,y2,40,40);

        delay(50);

        cleardevice();
        if(f1==1) { x1++;y1++;}
        else {x1--;y1--;}

        if(f2==1) { x2++;y2--;}
        else {x2--;y2++;}
    }
    getch();
    closegraph();
}
```



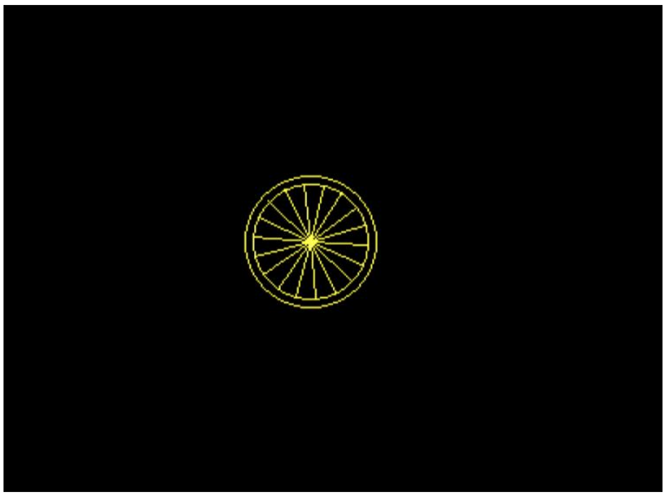
26. Program For screensaver

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

void main()
{
    int gd = DETECT, gm;
    int left = 200, top = 200, right = 700, bottom = 700, color = 15, pat = 8;
    initgraph(&gd, &gm, "C:\\Turboc3\\BGI");
    cleardevice();
    while (!kbhit())
    {
        setfillstyle(random(pat), random(color));
        bar(random(left), random(top), random(right), random(bottom));
        delay(250);
    }
    closegraph();
}
```



27. Program to display a rotating fan

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

int xc = 50, yc = 200, r = 35;
int x[15], y[15];
void drawcircles()
{
        setcolor(YELLOW);
        circle(xc, yc, r);
        circle(xc, yc, r + 5);
}
void main()
{
        double angle = 0, theta;
        int i, a;
        int gd = DETECT, gm;
        initgraph(&gd, &gm, "C:\\TURBOC3\\BGI\\");
        a = xc + r;
        while (!kbhit())
        {
                while (a <= 630)
                {
                        theta = M_PI * angle / 180;
                        cleardevice();
                        drawcircles();
                        for (i = 0; i < 18; i++)
                        {
                                theta = M_PI * angle / 180;
                                x[i] = xc + r * cos(theta);
                                y[i] = yc + r * sin(theta);
                                angle += 20;
                                line(xc, yc, x[i], y[i]);
                        }
                        angle += 2;
                        xc += 2;
                        a = xc + r;
                        delay(50);
                }
                xc = 50;
                r = 35;
                a = xc + r;
        }
        getch();
        closegraph();
}
```



28. Program to show a moving car

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

void main()
{

    int i, j = 0, gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI\\");

    for (i = 0; i <= 420; i = i + 10)
    {
        setcolor(RED);
        line(0 + i, 300, 210 + i, 300);
        line(50 + i, 300, 75 + i, 270);
        line(75 + i, 270, 150 + i, 270);
        line(150 + i, 270, 165 + i, 300);
        line(0 + i, 300, 0 + i, 330);
        line(210 + i, 300, 210 + i, 330);

        // For left wheel of car
        circle(65 + i, 330, 15);
        circle(65 + i, 330, 2);

        // For right wheel of car
        circle(145 + i, 330, 15);
        circle(145 + i, 330, 2);

        // Line left of left wheel
        line(0 + i, 330, 50 + i, 330);

        // Line middle of both wheel
        line(80 + i, 330, 130 + i, 330);

        // Line right of right wheel
        line(210 + i, 330, 160 + i, 330);

        delay(100);

        cleardevice();
    }

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