Harsh Kashiwal
2003085
C21

Page No.
Date

Experiment No. 13

Aim: Implement midpoint circle and midpoint ellipse drawing algorithm in C.

Theory:

Circle: Midpoint circle drawing algorithm works by finding out whether the midpoint of two pixels between which the circle passes through is above or below the circumference of the circle. Depending on that the selection of pixel is made. Also, eight-octant symmetry is used to draw the whole circle and only one octant's pixels are actually calculated.

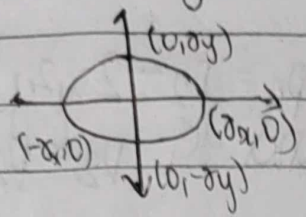
Algorithm:

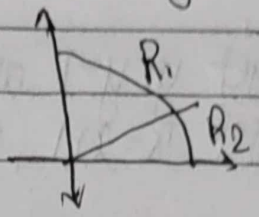
step 1: Take radius (r) and centre of circle (x_c, y_c) as input.
step 2: set $x=0, y=r$.
step 3: calculate initial decision parameter $p=1-r$.
step 4: Plot the 8 pixels in the octants that is,

(x_c+x, y_c+y), (x_c+x, y_c-y)
(x_c-x, y_c+y), (x_c-x, y_c-y)
(x_c+y, y_c+x), (x_c+y, y_c-x)
(x_c-y, y_c+x), (x_c-y, y_c-x).

step 5: If $p<0$
then set $x=x+1$
and set $p=p+2x+1$
else set $x=x+1$ and $y=y-1$
and set $p=p+2x+1-2y$.
step 6: Repeat steps 4-5 until $x \geq y$.

Ellipse: Ellipse is an elongated circle which can be described with a centre and two radii. The radius of the ellipse along x-axis (r_x) and radius of ellipse along y-axis (r_y). In midpoint ellipse drawing, we need to make use of 4 quadrant symmetry instead of 8-octant symmetry. We will call the region of ellipse with slope < -1 region 1 and slope > -1, region 2





Algorithm:

Step 1: Take centre of ellipse (x_c, y_c) and radii (r_x and r_y)

Step 2: Set $x = 0$, $y = r_y$

Step 3: Calculate initial decision parameter in region 1.

$$p_1 = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2$$

Step 4: Plot the 4 pixels in each quadrant, that is

$(x_c - x, y_c + y)$, $(x_c + x, y_c + y)$.

$(x_c - x, y_c - y)$, $(x_c + x, y_c - y)$.

Step 5: If $p_1 < 0$

then set $x = x + 1$ and set $p_1 = p_1 + 2r_y^2 x + r_y^2$

else set $x = x + 1$; $y = y - 1$ and set $p_1 = p_1 + 2r_y^2 x - 2r_x^2 y + r_y^2$

Step 6: Repeat steps 4-5 until $2r_y^2 x \geq 2r_x^2 y$.

Now the first region has been plotted.

Step 7: Calculate initial decision parameter in region 2 using x and y values reached in region 1.

$$p_2 = r_y^2\left(\frac{1}{2}+x\right)^2 + r_x^2(y-1)^2 - r_x^2 r_y^2$$

Step 8: Plot the 4 points in each quadrant, same as region 1.

Step 9: If $p_2 > 0$,

then set $y = y-1$ and set $p_2 = p_2 - 2r_y^2 y + r_x^2$

else set $x = x+1$, $y = y-1$ and set $p_2 = p_2 + 2r_y^2 x - 2r_x^2 y + r_x^2$

Step 10: Repeat steps 8-9 until $2r_y^2 x \geq 2r_x^2 y$

Now both regions 1 and 2 have been plotted and thus complete ellipse has been plotted.

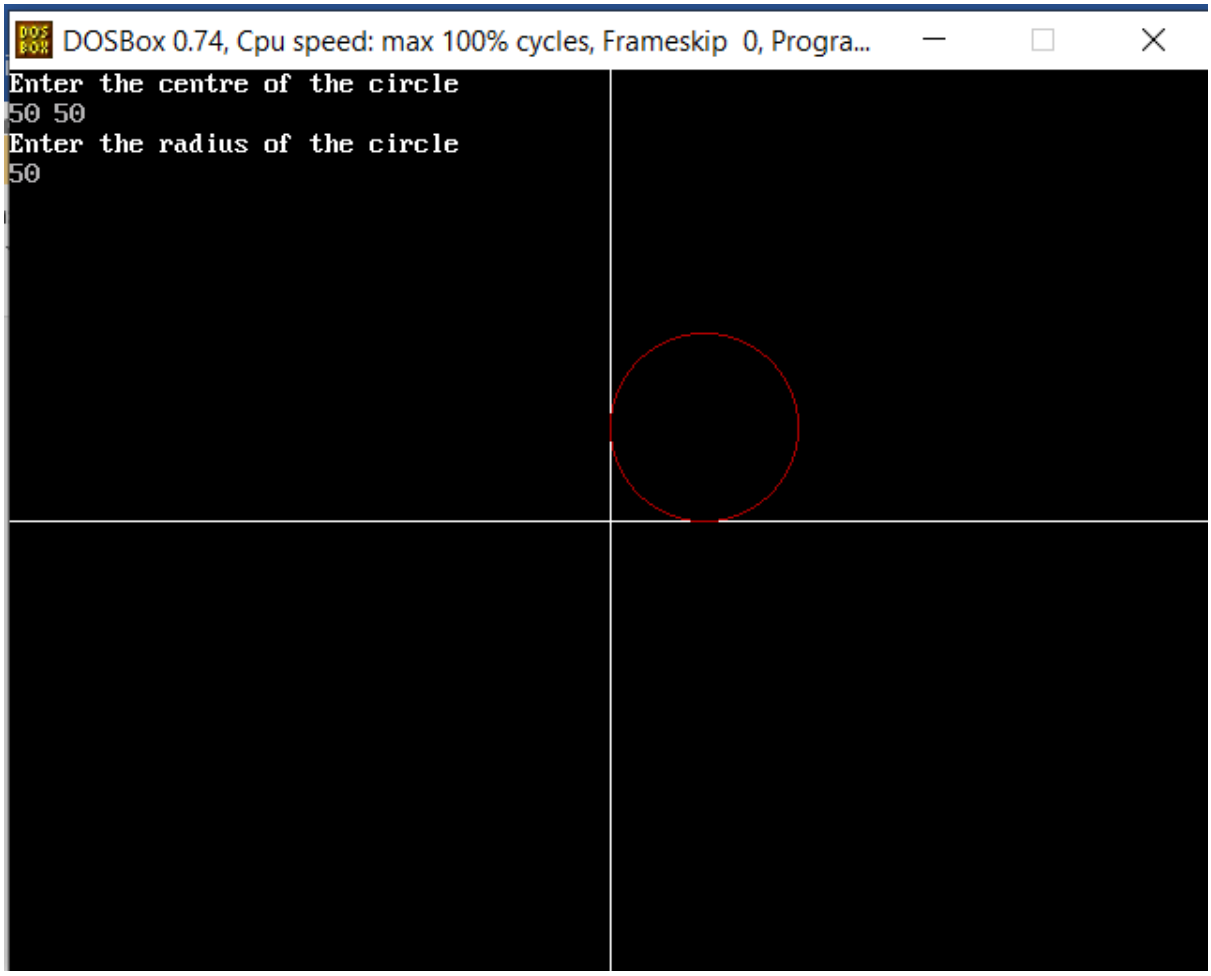
PROGRAM:

Code 1: Midpoint Circle

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
void plot(int, int, int, int, int);
void Circle(int, int, int, int);
void main()
{
 int gd = DETECT, gm;
 int xc, yc, r, col = RED;
 initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
 line(320, 0, 320, 480);
 line(0, 240, 640, 240);
 printf("Enter the centre of the circle\n");
 scanf("%d %d", &xc, &yc);
 printf("Enter the radius of the circle\n");
 scanf("%d", &r);
 Circle(xc, yc, r, col);
 getch();
}
void Circle(int xc, int yc, int r, int col)
{
 int x = 0, y = r, p = 1-r;
 while(x <= y)
 {
 plot(xc, yc, x, y, col);
 if(p < 0)
```

```
 {
 x++;
 p = p + 2*x + 1;
 }
 else
 {
 x++;
 y--;
 p = p + 2*(x-y) + 1;
 }
 }
}
void plot(int xc, int yc, int x, int y, int col)
{
 putpixel(320+xc+x, 240-(yc+y), col);
 putpixel(320+xc+x, 240-(yc-y), col);
 putpixel(320+xc-x, 240-(yc+y), col);
 putpixel(320+xc-x, 240-(yc-y), col);
 putpixel(320+xc+y, 240-(yc+x), col);
 putpixel(320+xc+y, 240-(yc-x), col);
 putpixel(320+xc-y, 240-(yc+x), col);
 putpixel(320+xc-y, 240-(yc-x), col);
}
```

OUTPUT:

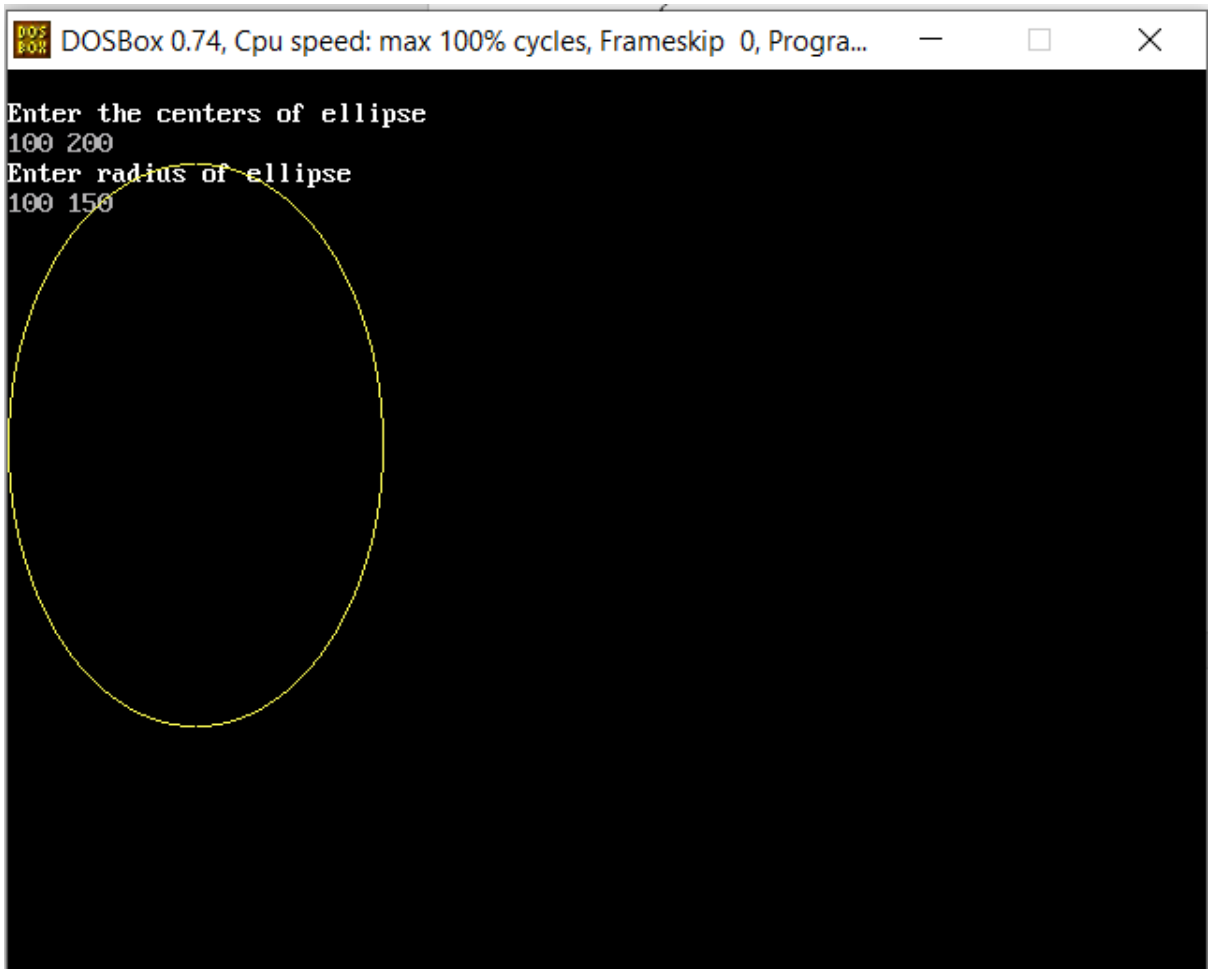

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...
Enter the centre of the circle
50 50
Enter the radius of the circle
50

Code 2: Midpoint Ellipse

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
#include<math.h>
void main(){
int gd=DETECT, gm;
float xc,yc,rx,ry,x,y,pk,p1k,p2k;
initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
printf("\nEnter the centers of ellipse\n");
scanf("%f%f", &xc, &yc);
printf("Enter radius of ellipse\n");
scanf("%f%f", &rx, &ry);
x=0; y=ry;
pk= (ry*ry)-(rx*rx*ry)+((rx*rx)/4);
while((2*ry*ry*x)<(2*rx*rx*y)){
        if(pk<=0){
        x++;
        p1k=pk+(2*ry*ry*x)+(ry*ry);
        }
        else{
        x++;
        y--;
        p1k=pk+(2*ry*ry*x)-(2*rx*rx*y)+(ry*ry);
        }
pk=p1k;
putpixel(xc+x, yc+y,14);
```

```
putpixel(xc-x, yc+y,14);
putpixel(xc+x, yc-y,14);
putpixel(xc-x, yc-y,14);
}
pk=((x+0.5)*(x+0.5)*ry*ry)+((y-1)*(y-1)*rx*rx)-(rx*rx*ry*ry);
while(y>0){
        if(pk>0){
        y=y-1;
        p2k=pk-(2*rx*rx*y)+(rx*rx);
        }
        else{
        x=x+1;
        y=y-1;
        p2k=pk+(2*ry*ry*x)-(2*rx*rx*y)+(rx*rx);
        }
pk=p2k;
putpixel(xc+x, yc+y,14);
putpixel(xc-x, yc+y,14);
putpixel(xc+x, yc-y,14);
putpixel(xc-x, yc-y,14);
}
getch();
}
```

OUTPUT:


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...
Enter the centers of ellipse
100 200
Enter radius of ellipse
100 150