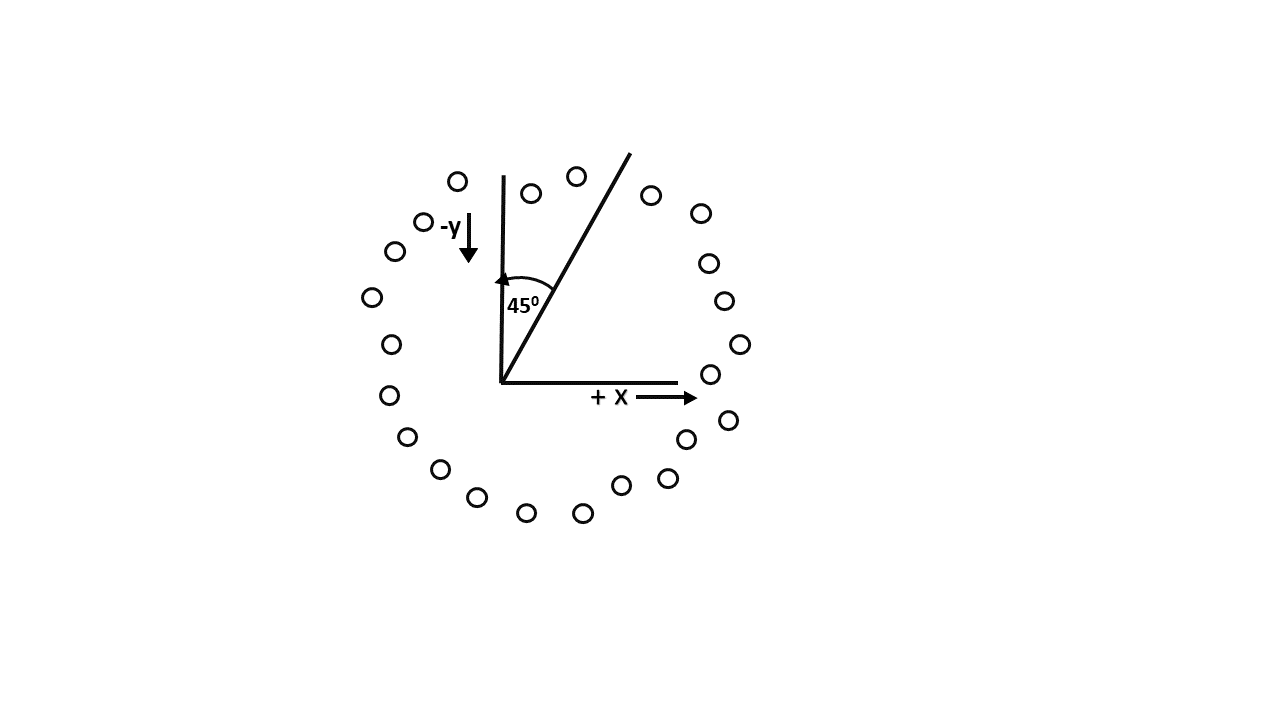
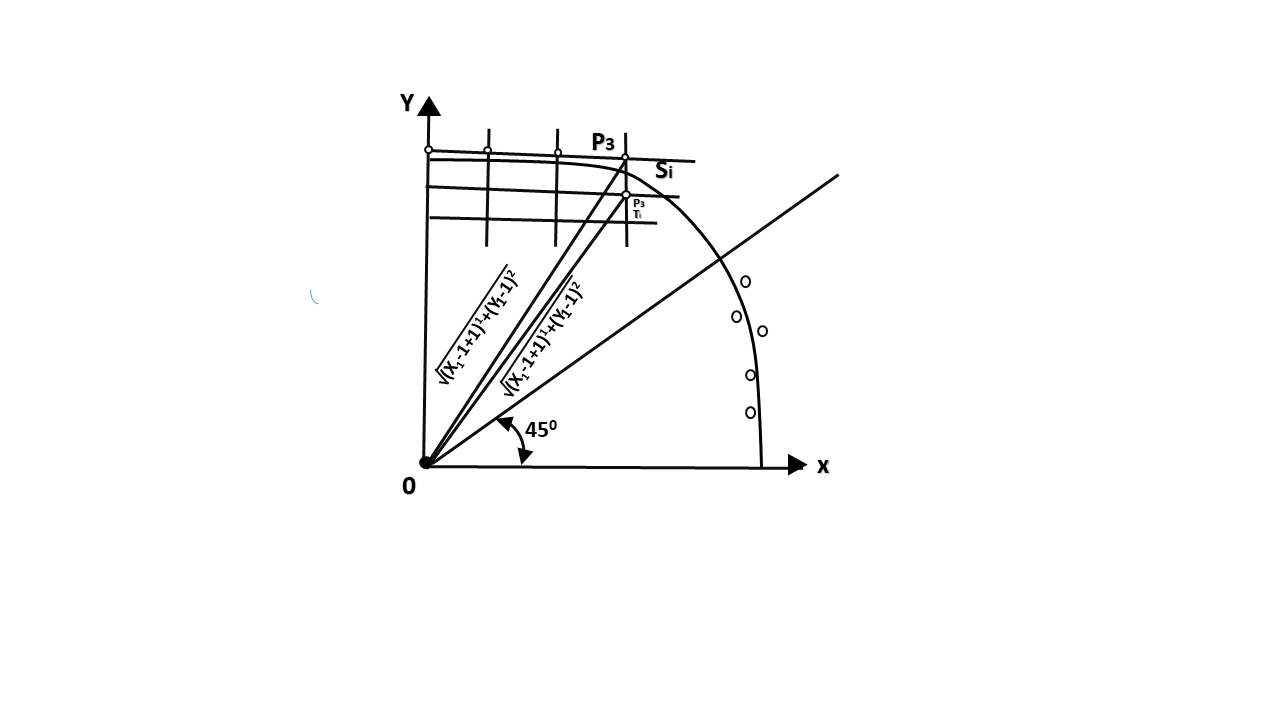
Bresenham's Circle Algorithm:

Scan-Converting a circle using Bresenham's algorithm works as follows: Points are generated from 90° to 45°, moves will be made only in the +x & -y directions as shown in fig:



The best approximation of the true circle will be described by those pixels in the raster that falls the least distance from the true circle. We want to generate the points from



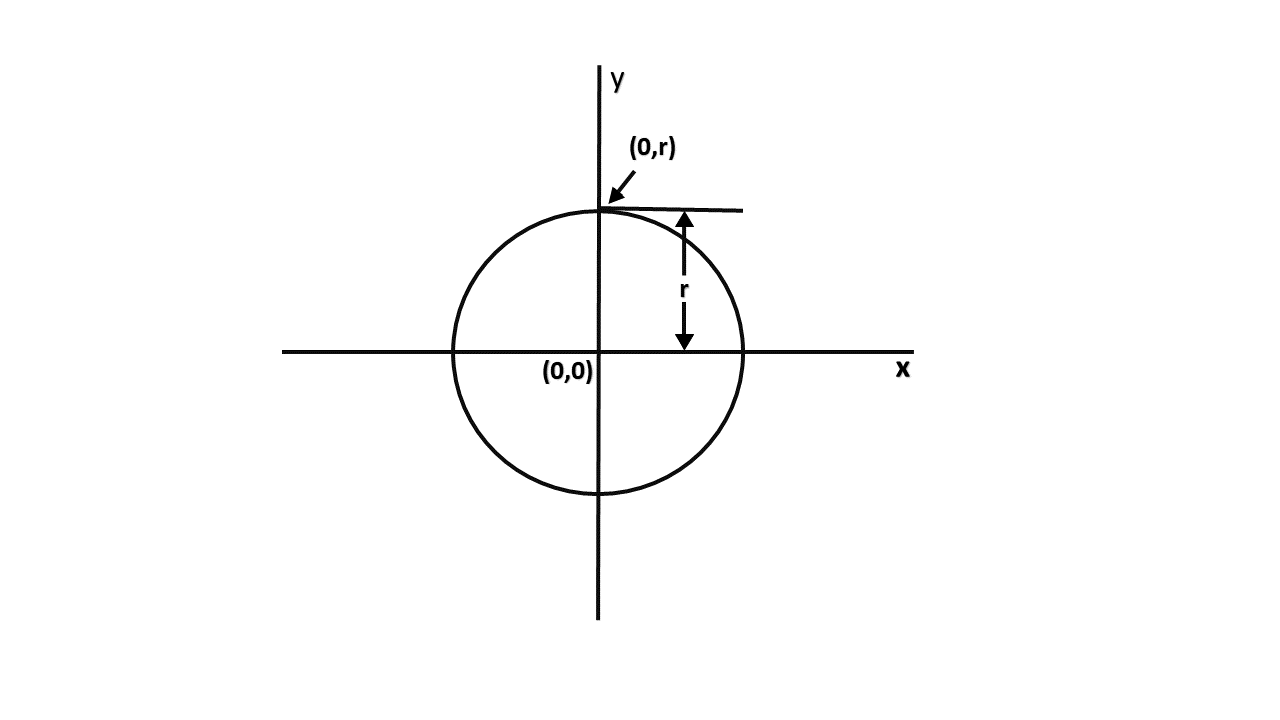
90° to 45°. Assume that the last scan-converted pixel is P1 as shown in fig. Each new point closest to the true circle can be found by taking either of two actions.

1. Move in the x-direction one unit or
2. Move in the x- direction one unit & move in the negative y-direction one unit.

Let D (Si) is the distance from the origin to the true circle squared minus the distance to point P3 squared. D (Ti) is the distance from the origin to the true circle squared minus the distance to point P2 squared. Therefore, the following expressions arise.

            D (Si)=(xi-1+1)2+ yi-12 -r2  
            D (Ti)=(xi-1+1)2+(yi-1 -1)2-r2

Since D (Si) will always be +ve & D (Ti) will always be -ve, a decision variable d may be defined as follows:



di=D (Si )+ D (Ti)

Therefore,  
di=(xi-1+1)2+ yi-12 -r2+(xi-1+1)2+(yi-1 -1)2-r2

From this equation, we can drive initial values of di as

If it is assumed that the circle is centered at the origin, then at the first step x = 0 & y = r.

Therefore,  
            di=(0+1)2+r2 -r2+(0+1)2+(r-1)2-r2  
            =1+1+r2-2r+1-r2  
            = 3 - 2r

Thereafter, if d\_i<0,then only x is incremented.

xi+1=xi+1         di+1=di+ 4xi+6

& if di≥0,then x & y are incremented  
xi+1=xi+1         yi+1 =yi+ 1  
di+1=di+ 4 (xi-yi)+10

Bresenham's Circle Algorithm:

**Step1:** Start Algorithm

**Step2:** Declare p, q, x, y, r, d variables  
        p, q are coordinates of the center of the circle  
        r is the radius of the circle

**Step3:** Enter the value of r

**Step4:** Calculate d = 3 - 2r

**Step5:** Initialize       x=0  
          &nbsy= r

**Step6:** Check if the whole circle is scan converted  
            If x > = y  
            Stop

**Step7:** Plot eight points by using concepts of eight-way symmetry. The center is at (p, q). Current active pixel is (x, y).  
                putpixel (x+p, y+q)  
                putpixel (y+p, x+q)  
                putpixel (-y+p, x+q)  
                putpixel (-x+p, y+q)  
                putpixel (-x+p, -y+q)  
                putpixel (-y+p, -x+q)  
                putpixel (y+p, -x+q)  
                putpixel (x+p, -y-q)

**Step8:** Find location of next pixels to be scanned  
            If d < 0  
            then d = d + 4x + 6  
            increment x = x + 1  
            If d ≥ 0  
            then d = d + 4 (x - y) + 10  
            increment x = x + 1  
            decrement y = y - 1

**Step9:** Go to step 6

**Step10:** Stop Algorithm

**Example:** Plot 6 points of circle using Bresenham Algorithm. When radius of circle is 10 units. The circle has centre (50, 50).

**Solution:** Let r = 10 (Given)

**Step1:** Take initial point (0, 10)  
                d = 3 - 2r  
                d = 3 - 2 \* 10 = -17  
                d < 0 ∴ d = d + 4x + 6  
                      = -17 + 4 (0) + 6  
                      = -11

**Step2:** Plot (1, 10)  
          d = d + 4x + 6 (∵ d < 0)  
                = -11 + 4 (1) + 6  
                = -1

**Step3:** Plot (2, 10)  
           d = d + 4x + 6 (∵ d < 0)  
                = -1 + 4 x 2 + 6  
                = 13

**Step4:** Plot (3, 9) d is > 0 so x = x + 1, y = y - 1  
                          d = d + 4 (x-y) + 10 (∵ d > 0)  
                = 13 + 4 (3-9) + 10  
                = 13 + 4 (-6) + 10  
                = 23-24=-1

**Step5:** Plot (4, 9)  
            d = -1 + 4x + 6  
                = -1 + 4 (4) + 6  
                = 21

**Step6:** Plot (5, 8)  
            d = d + 4 (x-y) + 10 (∵ d > 0)  
                = 21 + 4 (5-8) + 10  
                = 21-12 + 10 = 19

So P1 (0,0)⟹(50,50)  
            P2 (1,10)⟹(51,60)  
            P3 (2,10)⟹(52,60)  
            P4 (3,9)⟹(53,59)  
            P5 (4,9)⟹(54,59)  
            P6 (5,8)⟹(55,58)

Program to draw a circle using Bresenham's circle drawing algorithm:

1. #include <graphics.h>
2. #include <stdlib.h>
3. #include <stdio.h>
4. #include <conio.h>
5. #include <math.h>
7. **void**    EightWaySymmetricPlot(**int** xc,**int** yc,**int** x,**int** y)
8. {
9. putpixel(x+xc,y+yc,RED);
10. putpixel(x+xc,-y+yc,YELLOW);
11. putpixel(-x+xc,-y+yc,GREEN);
12. putpixel(-x+xc,y+yc,YELLOW);
13. putpixel(y+xc,x+yc,12);
14. putpixel(y+xc,-x+yc,14);
15. putpixel(-y+xc,-x+yc,15);
16. putpixel(-y+xc,x+yc,6);
17. }
19. **void** BresenhamCircle(**int** xc,**int** yc,**int** r)
20. {
21. **int** x=0,y=r,d=3-(2\*r);
22. EightWaySymmetricPlot(xc,yc,x,y);
24. **while**(x<=y)
25. {
26. **if**(d<=0)
27. {
28. d=d+(4\*x)+6;
29. }
30. **else**
31. {
32. d=d+(4\*x)-(4\*y)+10;
33. y=y-1;
34. }
35. x=x+1;
36. EightWaySymmetricPlot(xc,yc,x,y);
37. }
38. }
40. **int**  main(**void**)
41. {
42. /\* request auto detection \*/
43. **int** xc,yc,r,gdriver = DETECT, gmode, errorcode;
44. /\* initialize graphics and local variables \*/
45. initgraph(&gdriver, &gmode, "C:\\TURBOC3\\BGI");
47. /\* read result of initialization \*/
48. errorcode = graphresult();
50. **if** (errorcode != grOk)  /\* an error occurred \*/
51. {
52. printf("Graphics error: %s\n", grapherrormsg(errorcode));
53. printf("Press any key to halt:");
54. getch();
55. exit(1); /\* terminate with an error code \*/
56. }
57. printf("Enter the values of xc and yc :");
58. scanf("%d%d",&xc,&yc);
59. printf("Enter the value of radius  :");
60. scanf("%d",&r);
61. BresenhamCircle(xc,yc,r);
63. getch();
64. closegraph();
65. **return** 0;
66. }

**Output:**

