CIRCLE-GENERATING ALGORITHM

Drawing a circle on the screen is a little complex than drawing a line.

There are two popular algorithms for generating a circle

يعد رسم دائرة على الشاشة أمرًا معقدًا بعض الشيء من رسم خط. هناك نوعان من

الخوارزميات الشائعة لتوليد الدائرة

- **Bresenham's Algorithm**
- **Midpoint Circle Algorithm**.

- خوارزمية بريسنهام
- خوارزمية الدائرة الوسطى.

These algorithms are based on the idea of determining the subsequent points required to draw the circle.

تستند هذه الخوارزميات على فكرة تحديد النقاط اللاحقة المطلوبة لرسم الدائرة.

Let us discuss the algorithms in detail –

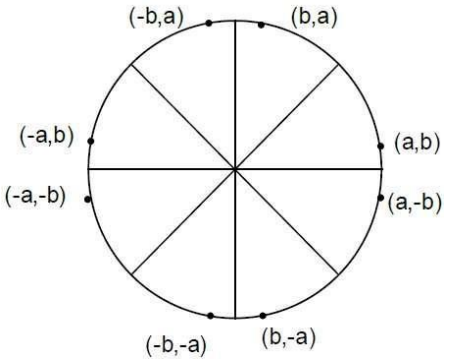
The equation of circle is

دعونا نناقش الخوارزميات بالتفصيل -

معادلة الدائرة هي

where r is radius.

$$x^2+y^2=r^2,$$

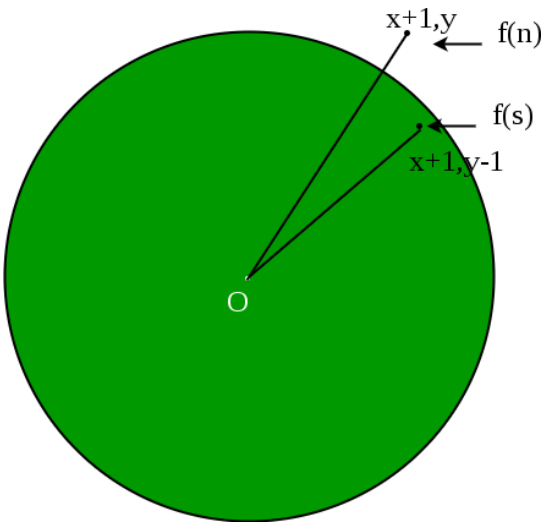


Bresenham's Algorithm

Now, we will see how to calculate the next pixel location from a previously known pixel location (x, y). In Bresenham's algorithm at any point (x, y) we have two options either to choose the next pixel in the east i.e. (x+1, y) or in the south east i.e. (x+1, y-1).

خوارزمية بريسنهام

الآن ، سنرى كيفية حساب موقع البكسل التالي من موقع بكسل معروف سابقًا (x، y). في خوارزمية Bresenham في أي نقطة (x، y) لدينا خياران إما لاختيار البكسل التالي في الشرق ، أي (x + 1، y) أو في الجنوب الشرقي ، أي (x + 1، y-1).


x+1,y
f(n)
f(s)
x+1,y-1
O

And this can be decided by using the decision parameter d as:

- If d > 0, then (x+1, y-1) is to be chosen as the next pixel as it will be closer to the arc.
- else (x+1, y) is to be chosen as next pixel.

ويمكن تحديد ذلك باستخدام معامل القرار δ على النحو التالي:

- إذا كانت $\delta > 0$ ، فسيتم اختيار $(\psi - 1، \xi + 1)$ على أنها البكسل التالي لأنها ستكون أقرب إلى القوس.
- سيتم اختيار $\xi + 1$) $\varepsilon\lambda\sigma\varepsilon$، ($\psi$ ليكون البكسل التالي.

Now to draw the circle for a given radius 'r' and centre (xc, yc) We will start from (0, r) and move in first quadrant till x=y (i.e., 45 degree). We should start from listed initial condition:

الآن لرسم دائرة لنصف قطر معين "r" والمركز (xc، yc) سنبدأ من (0، r) و
التحرك في الربع الأول حتى x = y (أي 45 درجة). يجب أن نبدأ من الحالة الأولية المدرجة:

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

Now for each pixel, we will do the following operations:

1. Set initial values of (xc, yc) and (x, y)
2. Set decision parameter d to d = 3 – (2 * r).
3. call drawCircle(xc, yc, x, y) function.
4. Repeat steps 5 to 8 until x < = y
5. Increment value of x.
6. If d < 0, set d = d + (4*x) + 6
7. Else, set d = d + 4 * (x – y) + 10 and decrement y by 1.
8. call drawCircle(int xc, int yc, int x, int y) function

1. قم بتعيين القيم الأولية لـ (xc، yc) و (x، y)
2. اضبط معامل القرار d على d = 3 - (2 * r).
3. استدعاء drawCircle (xc، yc، x، y) وظيفة.
4. كرر الخطوات من 5 إلى 8 حتى x <= y
5. زيادة قيمة x.
6. إذا كانت d <0 ، فقم بتعيين d = d + (4 * x) + 6
7. عدا ذلك ، قم بتعيين d = d + 4 * (x - y) + 10 وإنقاص y بمقدار 1.
8. استدعاء drawCircle (int xc، int yc، int x، int y) function

drawCircle() function:

```
// function to draw all other 7 pixels
// present at symmetric position
drawCircle(int xc, int yc, int x, int y)
```

دالة drawCircle ():

```
// وظيفة لرسم جميع البكسلات الـ 7
الأخرى
// موجود في وضع متماثل رسم دائرة ( int
xc، int yc، int x، int y)
{
   putpixel(xc+x, yc+y);
```

```
    putpixel(xc-x, yc+y);
    putpixel(xc+x, yc-y);
    putpixel(xc-x, yc-y);
    putpixel(xc+y, yc+x);
    putpixel(xc-y, yc+x);
    putpixel(xc+y, yc-x);
    putpixel(xc-y, yc-x);
}
```

Advantages

- It is a simple algorithm.
- It can be implemented easily
- It is totally based on the equation of circle i.e. x2 +y2 =r2

Disadvantages

- There is a problem of accuracy while generating points.
- This algorithm is not suitable for complex and high graphic images.

مزايا

- إنها خوارزمية بسيطة.
- يمكن تنفيذه بسهولة
- تقوم كليًا على معادلة الدائرة ، أي $\rho 2 = \psi 2 + \xi 2$

سلبيات

- وجود مشكلة في الدقة أثناء توليد النقاط.
- هذه الخوارزمية غير مناسبة للصور الرسومية المعقدة والعالية.

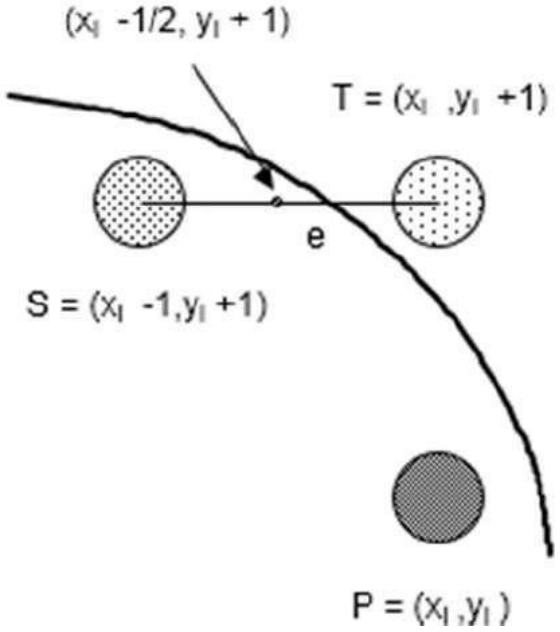
Mid-Point Algorithm

The mid-point circle drawing algorithm is an algorithm used to determine the points needed for rasterizing a circle.

We use the mid-point algorithm to calculate all the perimeter points of the circle in the first octant and then print them along with their mirror points in the other octants. This will work because a circle is symmetric about its centre.

خوارزمية النقطة المتوسطة

خوارزمية رسم الدائرة المتوسطة هي خوارزمية تُستخدم لتحديد النقاط المطلوبة لتنقيط الدائرة.
نستخدم خوارزمية النقطة الوسطى لحساب جميع نقاط محيط الدائرة في الثماني الأول ثم نطبعها مع نقاط المرآة الخاصة بها في الثمانيات الأخرى. سينجح هذا لأن الدائرة متناظرة حول مركزها.



Procedure-

Given-

Centre point of Circle = (X_0, Y_0)

Radius of Circle = R

The points generation using Mid-Point Circle Drawing Algorithm involves the following steps-

إجراء-

منح-

نقطة مركز الدائرة = (X0، Y0) نصف قطر الدائرة = R.

يتضمن إنشاء النقاط باستخدام خوارزمية رسم دائرة منتصف النقطة الخطوات التالية-

Step-01:

Assign the starting point coordinates (X_0, Y_0) as-

$X_0 = 0$

$Y_0 = R$

الخطوة 01:

قم بتعيين إحداثيات نقطة البداية (X0، Y0) كـ- X0 = 0

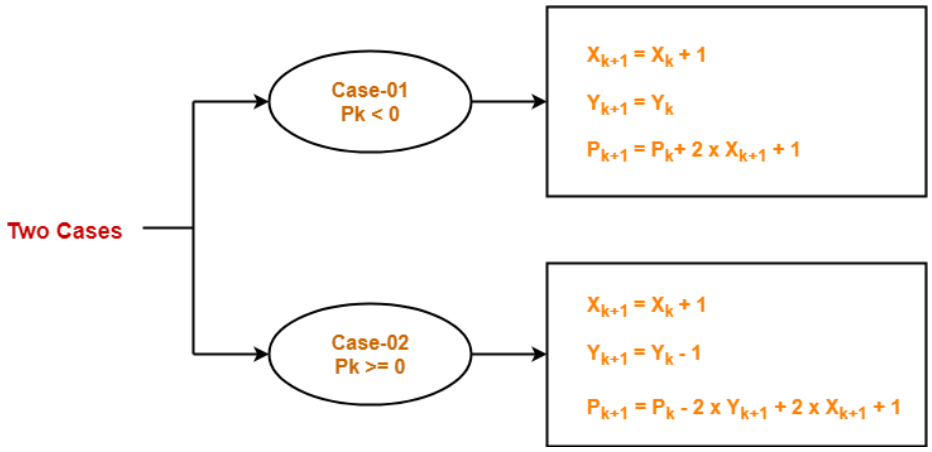
Y0 = ص

Step-02:
Calculate the value of initial decision parameter P0 as-
$P_0 = 1 – R$

Step-03:
Suppose the current point is (X_k, Y_k) and the next point is (X_{k+1}, Y_{k+1}).
Find the next point of the first octant depending on the value of decision parameter Pk.
Follow the below two cases-

الخطوة 02:
احسب قيمة معلمة القرار الأولي P0 as- P0 = 1 - R

الخطوة 03:
افترض أن النقطة الحالية هي (Xk، Yk) والنقطة التالية هي (Xk + 1، Yk + 1).
ابحث عن النقطة التالية من الثماني الأول اعتمادًا على قيمة معلمة القرار Pk. اتبع الحالتين التاليتين-



Step-04:

If the given centre point (X_0, Y_0) is not (0, 0), then do the following and plot the point-

- $X_{plot} = X_c + X_0$
- $Y_{plot} = Y_c + Y_0$

Here, (X_c, Y_c) denotes the current value of X and Y coordinates.

Step-05:

Keep repeating Step-03 and Step-04 until $X_{plot} >= Y_{plot}$.

Step-06:

Step-05 generates all the points for one octant.
To find the points for other seven octants, follow the eight-symmetry property of circle.

Examples:

Input: Centre = (0, 0), Radius = 3

Output: (3, 0) (3, 0) (0, 3) (0, 3)
(3, 1) (-3, 1) (3, -1) (-3, -1)
(1, 3) (-1, 3) (1, -3) (-1, -3)
(2, 2) (-2, 2) (2, -2) (-2, -2)

Input: Centre = (4, 4), Radius = 2

Output: (6, 4) (6, 4) (4, 6) (4, 6)
(6, 5) (2, 5) (6, 3) (2, 3)
(5, 6) (3, 6) (5, 2) (3, 2)