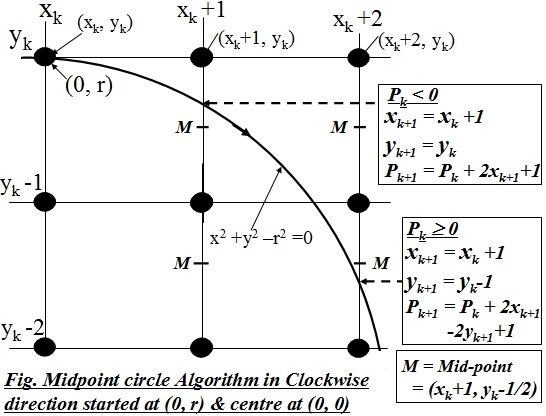
**TITLE: - MID POINT CIRCLE GENERATING ALGORITHM**

**OBJECTIVE: -**

* To implement Mid-point circle generating Algorithm

**THEORY: -**

The mid-point circle generating algorithm is an algorithm used to determine the points needed for rasterizing a circle. Since, circle is symmetric about its centre; using this algorithm the circumference points in the first octant are calculated. The points in remaining octants are obtained by symmetry.

To implement the mid-point method, we define a circle function as:

The relative position of any point can be determined by checking the circle function as:

Suppose, is the pixel plotted, the next pixel will be either or as shown in the figure. And, the mid-point is .

Thus, our decision parameter

* …………………………….…….{1}

Also the successive decision parameters are obtained using the incremental calculations. We obtain a recursive expression for the next decision parameter by evaluating the circle function at sampling position.

Thus, the next pixel to plot will either be or .

Also, the mid-point is: and thus, our decision parameter

* …… {2}

Now, subtracting {1} from {2}, we get;

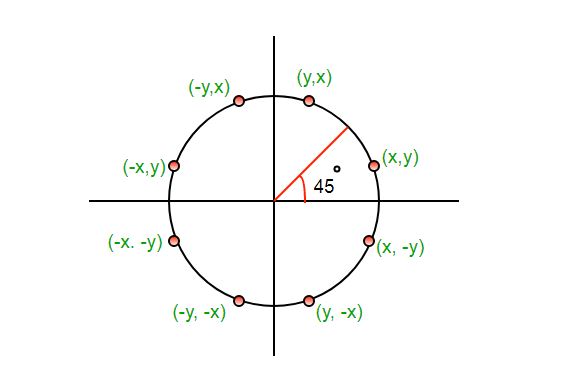
If, , the mid-point is inside the circle and the pixel on the scan line closer to the circle boundary. Otherwise, the mid-point is outside or on the boundary and we select the pixel on scan line .

**CASE I:**

**CASE II :**

For Initial decision parameter, our starting point is at the top, then the initial decision parameter is obtained by evaluating the circle at starting position

* **(eqvt.)**



**Mid-Point Circle Drawing Algorithm :**

Step 1: Input radius and circle centre and obtain the first point on the circumference of a circle centred on the origin as .

Step 2: Calculate the value of initial decision parameter as .

Step 3: At each position, starting at , perform following tasks:

1. If , next point along the circle centred at origin is and .
2. Else, the next point is and .

Step 4: Determine the symmetry points on the 7 octants.

Step 5: Move each calculated pixels position onto the circular path centred on and plot the coordinate values as and .

Step 6: Repeat step 3 through Step 5 until .

Note: For the circle starting from , but and the operations are performed accordingly.

**Some advantages of Mid-point Circle Drawing Algorithm:**

* Can be extended to draw ellipse.
* Non-floating Point arithmetic
* More efficient compared to other algorithms.
* Does not requires extensive memory usage.
* Relatively easy to understand and implement.

**Some Disadvantages of Mid-Point Circle Drawing Algorithm:**

* Jagged edges and distortion in the appearance of the circle.
* Just draw the circle outline. Does not support filling circle.
* Rounding floating points gives certain inaccuracies.
* Limitedly efficient for modern hardware.
* Needs extra consideration for off-centre circle and ellipse drawing.

**Program to implement Mid-Point Circle Generating Algorithm:**

#include <iostream>

#include <stdio.h>

#include <conio.h>

#include <graphics.h>

using namespace std;

void print\_points(int x, int y){

char p1[15];

sprintf(p1,"(%d, %d)",x,y);

outtextxy(x,y,p1);

}

int main(){

int r, x1, y1, xk, yk, pk;

cout << "Enter the radius of the circle:- ";

cin >> r;

cout << "Enter the center:- ";

cin >> x1>> y1;

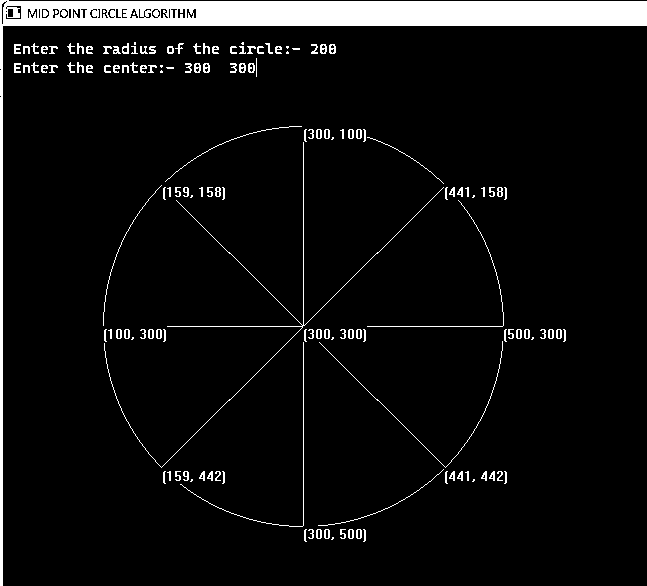
xk = 0;

yk = r;

pk = 1 - r;

int k = 0;

initwindow(getmaxwidth(), getmaxheight(),"MID POINT CIRCLE ALGORITHM");

 do{

putpixel(xk + x1, yk + y1, 15);

putpixel(yk + x1, xk + y1, 2);

putpixel(-xk + x1, yk + y1, 3);

putpixel(-yk + x1, xk + y1, 5);

putpixel(xk + x1, -yk + y1, 7);

putpixel(yk + x1, -xk + y1, 8);

putpixel(-xk + x1, -yk + y1, 4);

putpixel(-yk + x1, -xk + y1, 6);

if (pk < 0){

xk = xk + 1;

yk = yk;

pk = pk + 2 \* xk + 1;

}

else{

xk = xk + 1;

yk = yk - 1;

pk = pk + 2 \* xk + 1 - 2 \* yk;

}

k++;

}

while (xk <= yk);

line(x1 - xk, y1 - yk, xk + x1, yk + y1);

line(x1 - xk, y1 + yk, xk + x1, -yk + y1);

line(x1, r + y1, x1, y1 - r);

line(x1 - r, y1, x1 + r, y1);

print\_points(-r+x1, y1);

print\_points(yk + x1, -xk + y1);

print\_points(x1+r, y1);

print\_points(-yk + x1, -xk + y1);

print\_points(x1, -r + y1);

print\_points(yk + x1, xk + y1);

print\_points(x1, r + y1);

print\_points(-yk + x1, xk + y1);

print\_points(x1,y1);

getch();

closegraph();

}

**CONCLUSION**:

In this way, we implement Mid-point circle generating algorithm through writing code in C++ programming language and analysed its precision and way of calculation.