FUTURE VISION BIE

One Stop for All Study Materials

& Lab Programs

Future Vision

By K B Hemanth Raj

Scan the QR Code to Visit the Web Page

Or

Visit : https://hemanthrajhemu.github.io

Gain Access to All Study Materials according to VTU,
CSE – Computer Science Engineering,
ISE – Information Science Engineering,
ECE - Electronics and Communication Engineering
& MORE...

Join Telegram to get Instant Updates: https://bit.ly/VTU_TELEGRAM

Contact: MAIL: futurevisionbie@gmail.com

INSTAGRAM: www.instagram.com/hemanthraj_hemu/

INSTAGRAM: www.instagram.com/futurevisionbie/

WHATSAPP SHARE: https://bit.ly/FVBIESHARE

codeblocks with MinGW setup

1. CG Basics
2. OpenGL Basics
3. Three algorithms - DDA (Digital Differential Analyser), Bresenham's Line or midpoint line, Bresenham's circle or Midpoint circle

6.2.19

* Explain Raster-scan systems & Random-scan Displays.
* Explain CRT systems
Explain basic primitives with an example (points, lines, linestrip, lineloop)

Open GL Basics

gl.h - graphics library ; Primitive fncs how to draw line, circle

glu.h - graphics library utility ; contains fncs of gl.h & windowing fncs.

glut.h - graphics library utility Toolkit ; gl.h + glu.h + other properties

How to draw a point, line, linestrip, lineloop, polygon etc

First OpenGL program.

* Point

```
#include <GL/glut.h>
void display()
{   glPointSize(10);
    glBegin(GL_POINTS);
    glVertex2f(-0.5,-0.5);
    glVertex2f(0.5,-0.5);
    glVertex2f(0.5,0.5);
    glVertex2f(-0.5,-0.5);
    glEnd();
    glFlush();
}
void main(int argc, char **argv)
{   glutInit(&argc,argv);
    glutCreateWindow("Points Demo");
    glutDisplayFunc(display);
    glutMainLoop();
}
```

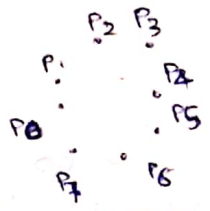
initializing openGL/graphics system so that system understands our graphics applications

OpenGL will create a window with Title 'Points Demo'

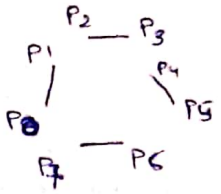
It is a call back function which renders pixels to be drawn on to screen. We need to register a function which actually draws/executes as a parameter to this glutDisplayFunc callback function.

→ Run the output forever

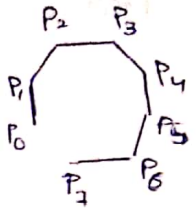
* Primitives



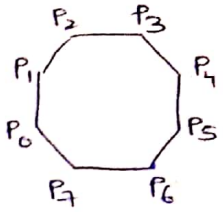
Points



Lines



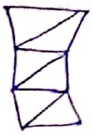
Linesstrip



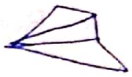
Lineloop

GL_Triangles

Triangle Strip



Triangle fan



GL_Polygon

Polygon with colour

GL quad

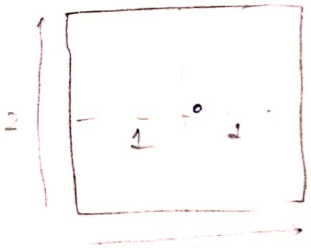
GL QuadStrip

→ In openGL programs default bgcolour is black
→ pixel colour is white by default
→ Entry point for any prog is int main()

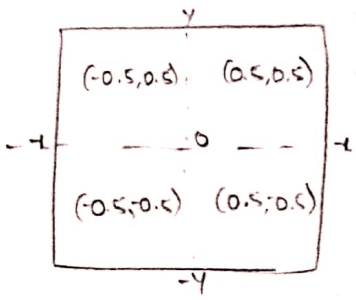
18-2-19

1st Program

→ canvas/window size 2×2 by default



Aim is to draw



* Applying custom background colour

1) glClearColor (0,0,1,1) // setting color
R G B A - Parameter
or
transparency parameter
It can be 1 or 0

2) glClear (Maskedbits); // applying color
↓
constant

In openGL, uppercase : constant

```
void display()
{
    glClearColor (0,0,1,1);
    glClear (GL_COLOR_BUFFER_BIT);
    glBegin (GL_POINTS);
    =
}
```

commenting either of the 2 stmts will leave window with black background only.

O/P Blue bg color
white pixel color

* Applying custom font color

```
glColor3f(1,0,1);
```

```
void display()
{
    glClearColor (0,0,1,1);
    glClear (GL_COLOR_BUFFER_BIT);
    glColor (1,1,0);
    =
}
```

O/P Blue Bg color
Yellow pixel color

LINES

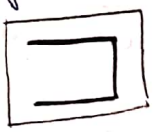
```
glBegin(GL_LINES);
glLineWidth(10); //thickness
```

pg1 O/P

LINE STRIP

```
glBegin(GL_LINE_STRIP);
```

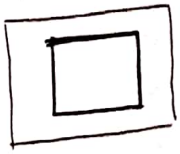
O/P



LINE LOOP

```
glBegin (GL_LINE_LOOP);
```

O/P



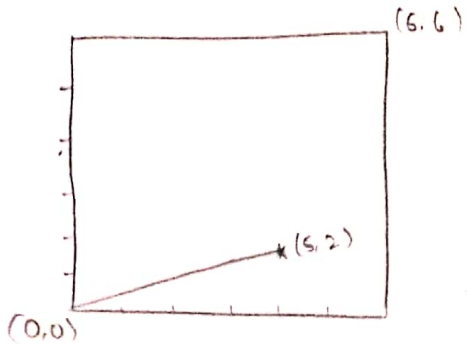
Algorithms

1) DDA
2) Bresenham's line
3) Bresenham's circle

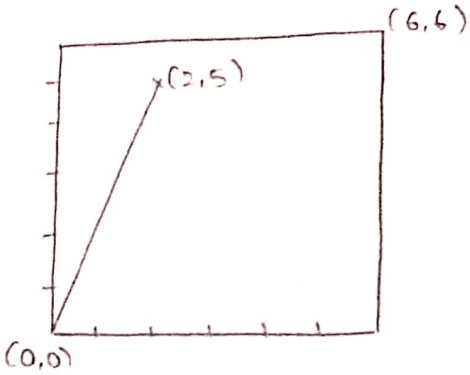
17 DDA Line Digital Differential Analyser

* screen is full of pixels
* aim : to determine intermediate pixels blw start & end point

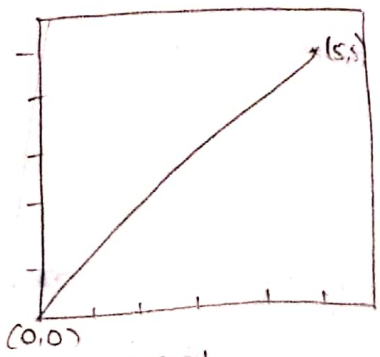
we have 3 cases



$m < 1$



$m > 1$



$m = 1$

we know, Line eq is $y = mx + c$ where $m = \frac{y_2 - y_1}{x_2 - x_1}$ — (1)

for any two points (x_1, y_1) & (x_2, y_2)

In general, let us assume

current pixel = (x_k, y_k)

$\therefore$ Next pixel = (x_{k+1}, y_{k+1})

$$m = \frac{y_{k+1} - y_k}{x_{k+1} - x_k} \quad — (2)$$

we have 3 cases,

case 1 : $m < 1$

x always gets changed unit wise.

$y = ?$

$$\boxed{x_{k+1} = x_k + 1}$$

$y_{k+1} = ?$

(2) $\Rightarrow$ $m = \frac{y_{k+1} - y_k}{1}$

$\therefore$ $\boxed{y_{k+1} = y_k + m}$

case 2 : $m > 1$

y always gets changed unit wise

$x = ?$

$$\boxed{y_{k+1} = y_k + 1}$$

$x_{k+1} = ?$

(2) $\Rightarrow$ $m = \frac{1}{x_{k+1} - x_k}$

$\therefore$ $\boxed{x_{k+1} = x_k + \frac{1}{m}}$

case 3: $m = 1$

y changes unit wise

x changes unit wise

$$x_{k+1} = x_k + 1$$
$$y_{k+1} = y_k + 1$$

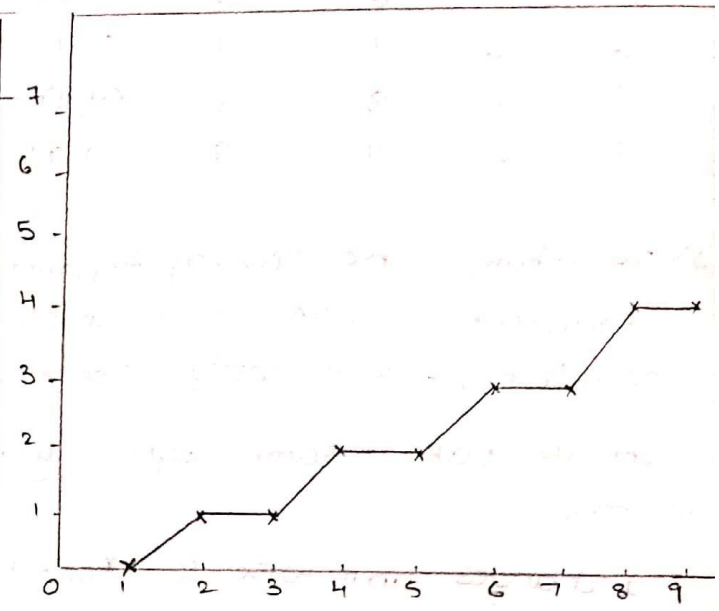
Example 1 : Draw a line between (1,0) to (9,4) using DDA.

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{4-0}{9-1} = \frac{1}{2} < 1$$

since $m < 1$,

$$x_{k+1} = x_k + 1$$
$$y_{k+1} = y_k + m$$

x_k	y_k	x_{k+1}	y_{k+1}	round (actual point (x,y))
1	0	1	0	(1,0)
1	0	2	0.5	(2,1)
2	0.5	3	1	(3,1)
3	1	4	1.5	(4,2)
4	1.5	5	2	(5,2)
5	2	6	2.5	(6,3)
6	2.5	7	3	(7,3)
7	3	8	3.5	(8,4)
8	3.5	9	4	(9,4)

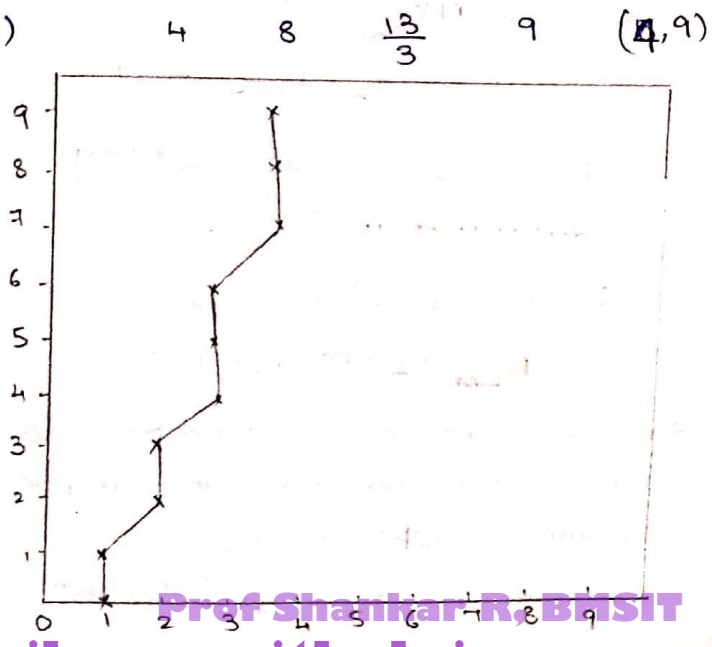


2: (1,0) to (4,9)

$m = 9/3 = 3 > 1$; $x_{k+1} = x_k + \frac{1}{m}$

$y_{k+1} = y_k + 1$

x_k	y_k	x_{k+1}	y_{k+1}	round(x,y)
1	0	1	0	(1,0)
1	0	4/3	1	(1,1)
4/3	1	5/3	2	(2,2)
5/3	2	2	3	(2,3)
2	3	8/3	4	(3,4)
8/3	4	3	5	(3,5)
3	5	10/3	6	(3,6)
10/3	6	11/3	7	(4,7)
11/3	7	4	8	(4,8)

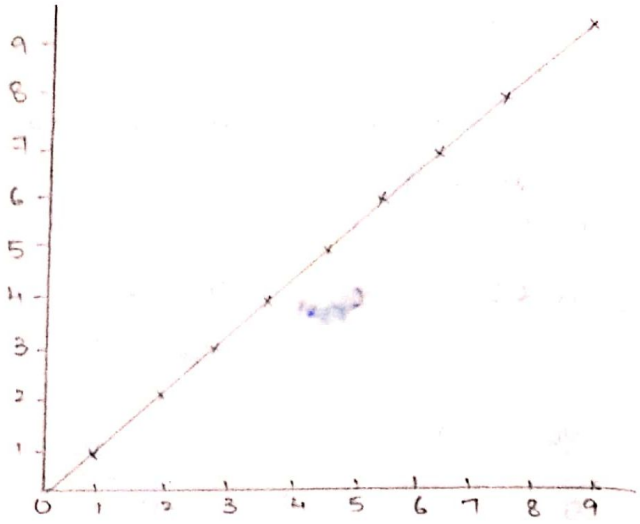


3) (1,1) to (9,9)

$m = 1$ $\quad x_{k+1} = x_k + 1$

$y_{k+1} = y_k + 1$

x_k	y_k	x_{k+1}	y_{k+1}	(x, y)
1	1	1	1	(1,1)
1	1	2	2	(2,2)
2	2	3	3	(3,3)
3	3	4	4	(4,4)
4	4	5	5	(5,5)
5	5	6	6	(6,6)
6	6	7	7	(7,7)
7	7	8	8	(8,8)
8	8	9	9	(9,9)



22/9/18

2) Bresenham's line Drawing Algorithm (Midpoint Line Algorithm)

Drawback of DDA is it involves rounding operation & floating pt operation which is costly. Hence we have Bresenham's line equation.

Let us take same eqn $y = mx + c$; 3 cases $m<1, m>1$ & $m=1$

i) $m < 1$

x changes unit wise, so $x_{k+1} = x_k + 1$ (no dilema)

$y_{k+1} = ?$ (there is a confusion between to choose y_{k+1} or y_k)

WKT $y = mx + c$, $m = \frac{\Delta y}{\Delta x}$

since it is $m<1$ case, we know $x_{k+1} = x_k + 1$

so., $y = m x_{k+1} + c$

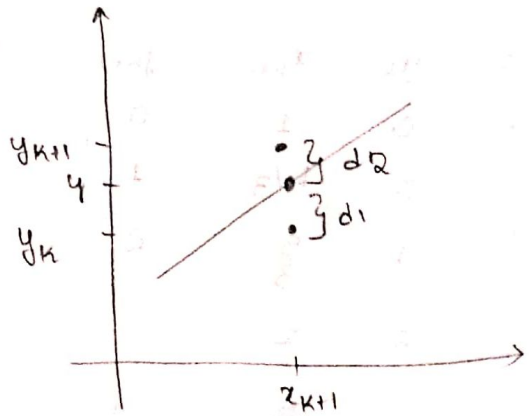
$y = m(x_k + 1) + c$ - ①

$d_1 = y - y_k$ $\quad d_2 = y_k + 1 - y$

substitute in ①

$d_1 = m(x_k + 1) + c - y_k$

$d_2 = y_k + 1 - m(x_k + 1) - c$



Since we are in a dilemma to choose blw $y_k + 1$ or y_k, let us (consider) calculate decision parameter p_k which helps us in deciding y_{k+1} or y_k.

$p_k = \Delta x (d_1 - d_2)$ - ②

Let us calculate $(d_1 - d_2)$

$d_1 - d_2 = m(x_k+1) + c - y_k - y_k + 1 + m(x_k+1) + c$

$= 2m(x_{k+1}) + 2c - 2y_k - 1$

substitute in ②,

$P_k = \Delta x [2m(x_k+1) - 2y_k + 2c - 1]$

$= \Delta x \left[2 \cdot \frac{\Delta y}{\Delta x}(x_k+1) - 2y_k + 2c - 1\right]$

$$\boxed{P_k = 2\Delta y(x_k+1) - 2\Delta x y_k + \Delta x(2c-1)} \quad -③$$

eqn ③ $\Rightarrow$ P_k is initial decision parameter. In order to find the continuous decisions, we have to find the next P_k.

$$P_{k+1} = 2\Delta y(x_{k+1}+1) - 2\Delta x y_{k+1} + \Delta x(2c-1) \quad -④$$

From now the next decision parameter will always be difference b/w P_{k+1} & P_k.

④ − ③ $\Rightarrow P_{k+1} - P_k = 2\Delta y(x_{k+1}+1) - 2\Delta x y_{k+1} + \Delta x(2c-1) - 2\Delta y(x_k+1) + 2\Delta x y_k - \Delta x(2c-1)$

$= 2\Delta y x_{k+1} + 2\Delta y - 2\Delta x y_{k+1} - 2\Delta y x_k - 2\Delta y + 2\Delta x y_k$

$= 2\Delta y(x_{k+1}) - 2\Delta x y_{k+1} - 2\Delta y(x_k) + 2\Delta x y_k$

$= 2\Delta y - 2\Delta x[y_{k+1} - y_k]$

$$\boxed{P_{k+1} = 2\Delta y - 2\Delta x(y_{k+1} - y_k) + P_k} \quad -⑤$$

The initial point to be plotted is (x_k, y_k)

Let us substitute (x_k, y_k) for (x, y) in initial decision parameter.

Eqn ③ $y = mx + c \Rightarrow c = y - mx$

$P_k = 2\Delta y(x_k+1) - 2\Delta x y_k + \Delta x(2c-1)$

$= 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + \Delta x(2(y - mx) - 1)$

$= 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + \Delta x\left(2\left(y_k - \frac{\Delta y}{\Delta x}x_k\right) - 1\right)$

$= 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + 2\Delta x y_k - 2\Delta y x_k - \Delta x$

$$\boxed{P_k = 2\Delta y - \Delta x} \rightarrow ⑥$$

Eqn ⑥ is is initial decision parameter

(1) $P_K = 2\Delta y - \Delta x$ // apply once initially

(2) $P_{K+1} = P_K + 2\Delta y - 2\Delta x\,(y_{K+1} - y_K)$ // From next iteration

Conclusion: If ($P_K \geq 0$)

{

$x_{K+1} = x_K + 1$

$y_{K+1} = y_K + 1$

}

else

{ $x_{K+1} = x_K + 1$

$y_{K+1} = y_K$

}

eg 1: Draw a line between (1,0), (9,4) using Bresenham's

$m = \frac{4-0}{9-1} = 0.5 < 1.$ $\Delta y = 4$, $\Delta x = 8$, $2\Delta y = 8$, $2\Delta x = 16$

$P_K = 2\Delta y - \Delta x = 8 - 8 = 0$

Let us calculate Initial Decision parameter P_K, since this IDP P_K happens to be zero, see the if loop

x_K	y_K	P_K	x_{K+1}	y_{K+1}	(x_{K+1}, y_{K+1})	
—	—	—	1	0	(1,0)	
1	0	(1): 8-8 0	2	1	(2,1)	
2	1	(2) 0+8-16(1-0) -8	3	1	(3,1)	$y_{K+1} = 1$ $y_K = 0$
3	1	0	4	2	(4,2)	-8+8-16(0)
4	2	-8	5	2	(5,2)	0+8-16(1)
5	2	0	6	3	(6,3)	-8+8-16(0)
6	3	-8	7	3	(7,3)	0+8-16(3-1)
7	3	0	8	4	(8,4)	-8+8-16(3-3)
8	4	-8	9	4	(9,4)	0+8-16(4-3)

2. (3,2) to (9,6)

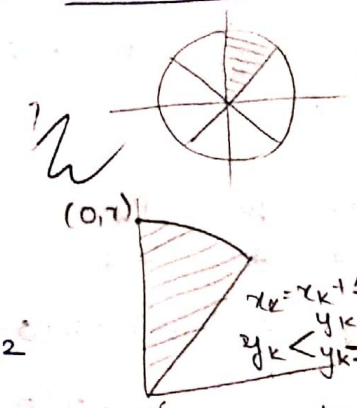
$m = \frac{6-2}{9-3} = \frac{4}{6} = \frac{2}{3} < 1$ $\quad \Delta y = 4, \Delta x = 6, 2\Delta y = 8, 2\Delta x = 12$

$P_K = 2(4) - 6 = 2$

x_k	y_k	P_K	x_{k+1}	y_{k+1}	(x_{k+1}, y_{k+1})	
—	—	—	3	2	(3,2)	
3	2	2	4	3	(4,3)	
4	3	-2	5	3	(5,3)	2+8-12(1)
5	3	+6	6	4	(6,4)	-2+8-12(0)
6	4	2	7	5	(7,5)	6+8-12(1)
7	5	-2	8	5	(8,5)	2+8-12(1)
8	5	6	9	6	(9,6)	-2+8-12(0)

3) Mid point circle [Bresenham's circle Algorithm]

Circle follows symmetrical property. As radius is same for all quadrants any one quadrant calculation is fine. In one quadrant one octant is fine



next coordinates may be (x_k+1, y_k) or (x_k+1, y_k-1)

mid point $= \frac{x_k+1+x_k+1}{2}, \frac{y_k+y_k-1}{2}$

$= \left(x_k+1, y_k - \frac{1}{2}\right)$

we have to apply this circle formula $x^2+y^2=r^2$

$$\boxed{P_K = (x_k+1)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2} \quad -(1)$$

This is Initial decision parameter

To find next, calculate P_{k+1}

From now on the next dp can be calculated by diff blw P_{k+1} & P_k

$P_{k+1} = (x_{k+1}+1)^2 + (y_{k+1} - \frac{1}{2})^2 - r^2$

$P_{k+1} - P_k = ((x_{k+1}+1)+1)^2 + (y_{k+1} - \frac{1}{2})^2 - (x_k+1)^2 - \left(y_k - \frac{1}{2}\right)^2$

$= (x_k+1)^2 + 1 + 2(x_k+1) + (y_{k+1}^2) + \frac{1}{4} - y_{k+1} - (x_k+1)^2 - y_k^2 - \frac{1}{4} + y_k$

$= 2(x_k+1) + y_{k+1}^2 - y_k^2 - (y_{k+1} - y_k) + 1$

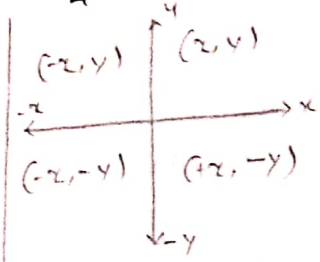
$$\boxed{\star\ P_{k+1} = P_k + 2(x_k+1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1}$$

Initial dp at (1) starting from (0, r)

$P_k = (0+1)^2 + \left(r - \frac{1}{2}\right)^2 - r^2 = 1 + r^2 + \frac{1}{4} - r^2 - r$

$= \frac{5}{4} - r$

★ $\boxed{P_k = 1 - r}$ → Initial dp



Quadrant 1 : calculate till (x, 0)
Quadrant 2 : write all values with '-' x
3 : -x, -y
4 : +x, -y

I Octant

Conclusion :

if ($P_k \geq 0$)

{
$x_{k+1} = x_k + 1$

$y_{k+1} = y_k - 1$
}

else

{
$x_{k+1} = x_k + 1$

$y_{k+1} = y_k$
}

Q. Draw a circle of radius 8 units using MPCircle algorithm.

II Octant

given r = 8

Initial dp $P_k = 1 - r = -7$

$P_{k+1} = P_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$

x_k	y_k	P_k	x_{k+1}	y_{k+1}
0	8	-7	1	8
1	8	$-7 + 2(0+1) + (8^2 - 8^2) - (8-8) + 1 = -7+2+1 = -4$	2	8
2	8	$-4 + 2(2) + (0) - (0) + 1 = -4+4+1 = 1$	3	7
3	7	$1 + 2(3) + (49 - 64) - (-1) + 1 = 1+6-15+1 = -7$	4	7
4	7	$-7 + 2(4) + (0) - 0 + 1 = -7+8+1 = 2$	5	6
5	6	$2 + 2(5) + (36 - 49) - (-1) + 1 = 2+10-13+1 = -9$	6	5
6	5	$-9 + 2(6) + 0 - 0 + 1 = -9+12+1 = 4$	7	3
7	3	$4 + 2(7) + 9 - 25 - 2 + 1 = 4+14-16-1 = 11$	8	2
8	2	$11 + 2(8) + 4 - 9 - 1 + 1 = 11+16-5 = 22$	8	1
8	1	$22 + 2(9) + 1 - 4 - 1 + 1 = 22+18-3 = 37$	8	0

9. $r = 12$

$P_k = 1 - 12 = -11$

x_k	y_k	P_k	x_{k+1}	y_{k+1}
0	12	-11	1	12
1	12	$-11 + 2(1) + (12^2 - 12^2) - (0) + 1 = -11 + 2 + 1 = -8$	2	12
2	12	$-8 + 2(2) + 0 - 0 + 1 = -8 + 4 + 1 = -3$	3	12
3	12	$-3 + 2(3) + 0 - 0 + 1 = -3 + 6 + 1 = 4$	4	11
4	11	$4 + 2(4) + 121 - 144 - (11 - 12) + 1 = 4 + 8 - 23 + 1 + 1 = -9$	5	11
5	11	$-9 + 2(5) + 0 - 0 + 1 = -9 + 10 + 1 = 2$	6	10
6	10	$2 + 2(6) + 100 - 121 - (-1) + 1 = 2 + 12 - 21 + 2 = -5$	7	10
7	10	$-5 + 2(7) + 0 - 0 + 1 = -5 + 14 + 1 = 10$	8	9
8	9	$10 + 2(8) + 81 - 100 + 1 + 1 = 10 + 16 - 19 + 2 = 9$	9	8
			10	7
			10	6
			11	5
			11	4
			12	3
			12	2
			12	1
			12	0

I Octant

II Octant

Quadrant e: x_{k+1} with $(-)$

3: $-x, -y$

4: $-x, -y$