Derivation

- Now, In this case our d_1 will be the distance between intersection point x and pixel x_k
  $d_1 = x - x_k$ By using ....(7)
  $d_1 = 1/m(y_{k+1} - c) - x_k$
  And similarly our d_2 will be the distance between intersection point x and
- pixel x_{k+1}
  $d_2 = x_{k+1} - x$ By using ....(7)
  $d_2 = x_{k+1} - 1/m(y_{k+1} - c)$
- Note: here y is always incrementing so we can write y_{k+1} as $y_k + 1$ and here x_{k+1} is next pixel so we can write it as $x_k + 1$.
  subtracting d_2 from d_1
  $d_1 - d_2 = 1/m(y_k + 1 - c) - x_k - [x_k + 1 - 1/m(y_k + 1 - c)]$
  $= 1/m(y_k + 1 - c) - x_k - x_k - 1 + 1/m(y_k + 1 - c)$
  $d_1 - d_2 = 2/m(y_k + 1 - c) - 2x_k - 1$
  $d_1 - d_2 = 2dx(y_k + 1 - c) - 2dyx_k - dy$

Lecture No 16 Topic: Derivation

- Multiplying both side by (dy)

  $dy(d_1-d_2) = 2dxy_k+2dx-2dxc-2dyx_k-dy$

- Now we need to find decision parameter P_K

  $P_K = dy(d_1-d_2)$

  $C = 2dx-2dxc-dy$ which is constant

  So new equation is

  $P_K = 2dxy_k-2dyx_k +C$ ......(8)

- Now our next parameter will be

  $P_{k+1} = 2dx(y_{k+1})-2dy(x_{k+1})+C$

- Substracting P_k from P_{k+1}

  $P_{k+1}-P_k = 2dx(y_{k+1}-y_k)-2dy(x_{k+1}-x_k)+C-C$

  Note: here x is always incrementing so we can write y_{k+1} as $y_k +1$

  $P_{k+1}-P_k = 2dx(y_k +1-y_k)-2dy(x_{k+1}-x_k)$

  $P_{k+1}=P_k+2dx-2dy(x_{k+1}-x_k)$

  when $P_k<0$ then $(d_1-d_2)<0$

Derivation

- So $d_1 < d_2$ then we will write x_{k+1} as x_k because current pixel's distance from intersection point x is smaller.so, we will have to choose current pixel.

  Then our formula will be:

  $P_{k+1} = P_k + 2dx - 2dy(x_k - x_k)$
  $P_{k+1} = P_k + 2dx$
  And when $P_k > 0$ then $(d_1 - d_2) > 0$
- So $d_1 > d_2$ then we will write x_{k+1} as $x_k + 1$ because current pixel's distance from intersection point x is Larger.so, we will have to choose next pixel.
- At that time our formula will be:
  $P_{k+1} = P_k + 2dx - 2dy(x_k + 1 - x_k)$
  $P_{k+1} = P_k + 2dx - 2dy$

Derivation

- For Initial decision parameter
  From 8th equation
  $P_K = 2dxy_k - 2dyx_k + C$
  $P_0 = 2dxy_0 + 2dx - 2dxc - 2dyx_0 - dy$
  By using 1st equation
  $y_k = m(x_k) + c$
- $c = y_k - m(x_k) P_0 = 2dxy_0 - 2dyx_0 + 2dx - 2dx(y_0 - m(x_0)) - dy$
  $P_0 = 2dxy_0 - 2dyx_0 + 2dx - 2dxy_0 + 2dyx_0 - dy$ (By using 2)
  $P_0 = 2dx - dy$

  hence formulas for bresenham is derived.

Bresenham`s Line Drawing Algorithm

```
Bresenham's Line Drawing Algorithm
#include<stdio.h>
#include<conio.h>
#include<graphics.h>

void main()
{
int gd =DETECT,gm;
initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
int x,x1,y,y1,dx,dy,pk;
printf("Enter first co-ordinates");
scanf(" %d%d",&x,&y);
printf("Enter second co-ordinates");
scanf(" %d%d",&x1,&y1);
dx=x1-x;
dy=y1-y;
pk=2*dy-dx;
while(x<x1)
    { x++;
   if(pk<0){
         pk+=2*dy;
          }
   else{y++;
   pk+=2*(dy-dx);
       }
   putpixel(x,y,9);
   }
getch();
closegraph();


}
```