Deriviation

DDA

Assume that any lines in octant 0,1 start at the point P0(x0,y0) and the next point P1(x1, y1). To calculate easily, dx = x1 - x0, dy = y1 - y0. And then the slope of this line, m = dy /dx.

P0(x0, y0) -> P1(x1, y1)

dx = x1 – x0, dy = y1 – y0

m = dy / dx

To draw a line using DDA, we use the slope calculated with mathematical line equation.

First, put the starting point P0 into mathematical line equation(y = mx + b),

y0 = mx0 + b. And the next point p1 also satisfy the equation : y1 = mx1 + b

y = mx + b

P0 : y0 = mx0 + b

P1 : y1 = mx1 + b

In pixel area, the interval between pixel’s x value and the right next pixel’s x value is 1. And we compute y value by stepping along X-axis.

So, y1 = mx1 + b is equal to y1 = m(x0 + 1) + b. Because y0 = mx0 + b, y1 = mx0 + m + b is equal to y1 = y0 + m.

y1 = m(x0 + 1(dx’s unit increment)) + b | dx = 1

y1 = mx0 + b + m

y1 = y0 + m

It means that to calculate next pixel, just add 1 to original pixel’s x value and m to original pixel’s y value.

P1(x1, y1) = P1(x0 + 1, y0 + m) // 0 <= m <= 1

If m > 1(it means dy < dx), we need to compute x by stepping along Y-axis instead of to compute y by stepping along X-axis.

P1(x1, y1) = P1(x0 + 1/m,y0 + 1) // m > 1

Pseudo-code1)

DrawLineOct0(float x0, float x1, float y0, float y1, Color I)

{  
 float dy = y1 – y0, y = y0;

Int x = ROUND(x0);

Int dx = ROUND(x1) –x0

float m = dy / dx;

Int xstep = 1;

float ystep = m;

Setpix(x,y,I);

While(dx--)

{

x += xstep;

y += ystep;

Setpix(x,y,I);

}

}

DrawLineOct1(float x0, float x1, float y0, float y1, Color I)

{

float dx = x1 – x0, x = x0;

Int y = ROUND(y0);

Int dy = ROUND(y1) –y0

float rm = dx / dy;

Int xstep = rm;

float ystep = 1;

Setpix(x,y,I);

While(dx--)

{

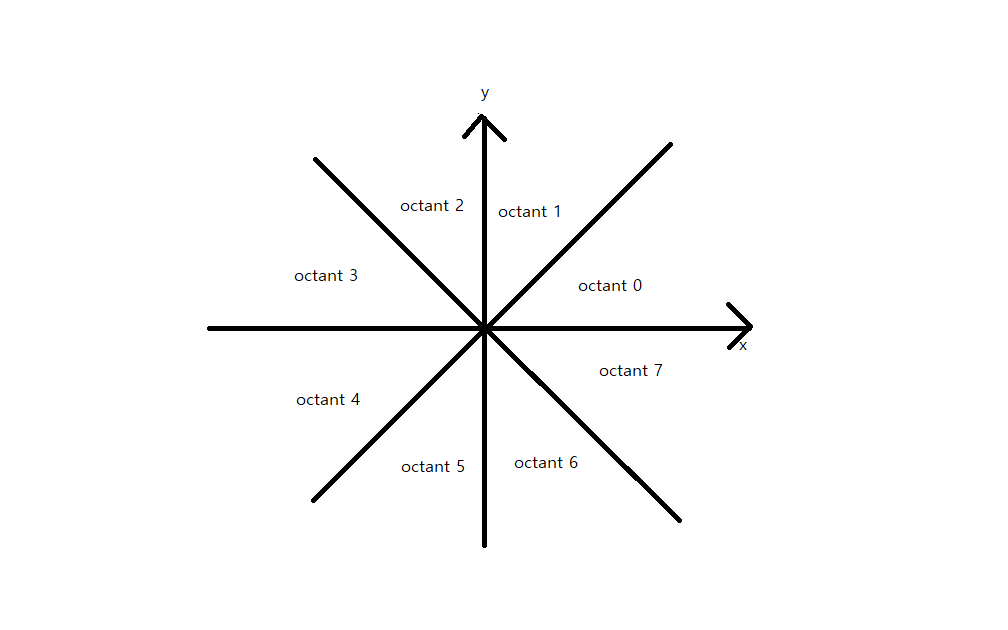
x += xstep;

y += ystep;

Setpix(x,y,I);

}

}



But it is only possible if m >= 0 and dx,dy > 0(octant 0, 1). For octant 4,5, dx and dy are negative. It means that the unit increment along x-axis is -1 and the unit increment along y-axis is -1.

(along x-axis) x1 = x0 - 1

(along y-axis) y1 = y0 - 1

And the sign of m determine that m is added or subtracted.

P1(x1, y1) = P1(x0 - 1, y0 - m) // 0 <= m <= 1

* y decrease in the picture on octant 5, so m need to be subtracted

P1(x1, y1) = P1(x0 - 1/m, y0 - 1) // m > 1

* x decrease in the picture on octant 6, so m need to be subtracted

So we need to figure out the sign of dx and dy that determine the sign of xstep and ystep and m that determine to add or subtract m to y, x value.

Octant0

P1(x1, y1) = P1(x0 + 1, y0 + m) // dx > 0, dy > 0, 0 <= m <= 1

Octant1

P1(x1, y1) = P1(x0 + 1/m,y0 + 1) // dx > 0, dy > 0, m > 1

Octant2

P1(x1, y1) = P1(x0 + 1/m,y0 + 1) // dx < 0, dy > 0, m < -1

Octant3

P1(x1, y1) = P1(x0 + 1, y0 - m) // dx < 0, dy > 0, 0 >= m >= -1

Octant4

P1(x1, y1) = P1(x0 - 1, y0 - m) // dx < 0, dy < 0, 0 <= m <= 1

Octant5

P1(x1, y1) = P1(x0 - 1/m,y0 - 1) // dx < 0, dy < 0 , m > 1

Octant6

P1(x1, y1) = P1(x0 - 1/m,y0 - 1) // dx > 0, dy < 0 , m < -1

Octant7

P1(x1, y1) = P1(x0 + 1, y0 + m) // dx > 0, dy < 0, 0 >= m >= -1

Pseudo-code2)

DrawLine (float x0, float x1, float y0, float y1, Color I)

{  
 float dy = y1 – y0, y = y0;

Int x = ROUND(x0);

Int dx = ROUND(x1) –x0

float m = dy / dx;

Int xstep = 1;

float ystep = m;

Setpix(x,y,I);

While(dx--)

{

x += xstep;

y += ystep;

Setpix(x,y,I);

}

}

DrawLine (float x0, float x1, float y0, float y1, Color I)

{

float dx = x1 – x0, x = x0;

Int y = ROUND(y0);

Int dy = ROUND(y1) –y0

float rm = dx / dy;

float ystep = (dy >= 0) 1: -1;

dy = (dy >= 0) ? dy: -dy;

rm = (rm >= 0.f) ? rm : -rm;

Int xstep = (dx >= 0.f) ? rm : -rm;

Setpix(x,y,I);

While(dx--)

{

x += xstep;

y += ystep;

Setpix(x,y,I);

}

}

Finally, y when 0 <= |m| <= 1 and x when 1 < |m| always round down, so we need to add 0.5 to round float value y and x.

float y = y0 + 0.5f

float x = x0 + 0.5f

Pseudo-code3)

DrawLine (float x0, float x1, float y0, float y1, Color I)

{  
 float dy = y1 – y0, y = y0 + 0.5f;

Int x = ROUND(x0);

Int dx = ROUND(x1) –x0

float m = dy / dx;

Int xstep = 1;

float ystep = m;

Setpix(x,y,I);

While(dx--)

{

x += xstep;

y += ystep;

Setpix(x,y,I);

}

}

DrawLine (float x0, float x1, float y0, float y1, Color I)

{

float dx = x1 – x0, x = x0 + 0.5f;

Int y = ROUND(y0);

Int dy = ROUND(y1) –y0

float rm = dx / dy;

float ystep = (dy >= 0) 1: -1;

dy = (dy >= 0) ? dy: -dy;

rm = (rm >= 0.f) ? rm : -rm;

Int xstep = (dx >= 0.f) ? rm : -rm;

Setpix(x,y,I);

While(dx--)

{

x += xstep;

y += ystep;

Setpix(x,y,I);

}

}

Example)

(0,0) -> (8, 4)

Octant 0

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| P1 | 0 | 0.5 | 1 | 1.5 | 2 | 2.5 | 3 | 3.5 | 4 |
| result | (0,0) | (1,0) | (2,1) | (3,1) | (4,2) | (5,2) | (6,3) | (7,3) | (8,4) |

(0,0) -> (4,8)

Octant 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P0 | 0 | 1 | 2 | 3 | 4 |
| P1 | 0 | 2 | 4 | 6 | 8 |
| result | (0,0) | (1,2) | (2,4) | (3,6) | (4,8) |

(0,0) -> (-4,8)

Octant 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P0 | 0 | -1 | -2 | -3 | -4 |
| P1 | 0 | 2 | 4 | 6 | 8 |
| result | (0,0) | (-1,2) | (-2,4) | (-3,6) | (-4,8) |

(0,0) -> (-8,4)

Octant 3

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P0 | 0 | -1 | -2 | -3 | -4 | -5 | -6 | -7 | -8 |
| P1 | 0 | 0.5 | 1 | 1.5 | 2 | 2.5 | 3 | 3.5 | 4 |
| result | (0,0) | (-1,0) | (-2,1) | (-3,1) | (-4,2) | (-5,2) | (-6,3) | (-7,3) | (-8,4) |

(0,0) -> (-8,-4)

Octant 4

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P0 | 0 | -1 | -2 | -3 | -4 | -5 | -6 | -7 | -8 |
| P1 | 0 | -0.5 | -1 | -1.5 | -2 | -2.5 | -3 | -3.5 | -4 |
| result | (0,0) | (-1,0) | (-2,-1) | (-3,-1) | (-4,-2) | (-5,-2) | (-6,-3) | (-7,-3) | (-8,-4) |

(0,0) -> (-4,-8)

Octant 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P0 | 0 | -1 | -2 | -3 | -4 |
| P1 | 0 | -2 | -4 | -6 | -8 |
| result | (0,0) | (-1,-2) | (-2,-4) | (-3,-6) | (-4,-8) |

(0,0) -> (4,-8)

Octant 6

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P0 | 0 | 1 | 2 | 3 | 4 |
| P1 | 0 | -2 | -4 | -6 | -8 |
| Result | (0,0) | (1,-2) | (-1,-4) | (3,-6) | (4,-8) |

(0,0) -> (8,-4)

Octant 7

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| P1 | 0 | -0.5 | -1 | -1.5 | -2 | -2.5 | -3 | -3.5 | -4 |
| result | (0,0) | (1,0) | (2,-1) | (3,-1) | (4,-2) | (5,-2) | (6,-3) | (7,-3) | (8,-4) |

Midpoint algorithm

Assume that any lines in octant 0,1 start at the point P0(x0,y0) and the next point P1(x1, y1). To calculate easily, dx = x1 - x0, dy = y1 - y0.

P0(x0, y0) -> P1(x1, y1)

dx = x1 – x0, dy = y1 – y0

In midpoint algorithm, we would fine the midpoint of next pixels.

y = dy/dx \* x + b

F(x,y) = dy \* x – dx \* y + b \* dx = 0

F(Midpoint of next pixel) = dy(x + 1) – dx(y + ½) + b\*dx

And then, determine the next pixel

F(East pixel) = dy(x + 1) – dx\*y + b\*dx

F(NorthEast pixel) = dy(x + 1) – dx(y + 1) + b\*dx

With the decision variable d

d(previous) = dy \* x – dx \* y – b\*dx = 0

d(x + 1, y + 1/2 (middlepoint)) = dy(x+1) – dx(y + ½) – b\*dx

d(Middlepoint) = dy\*x – dx \* y – b\*dx + dy -1/2dx

d(Middlepoint) = d(previous) + dy -1/2dx

d(Middlepoint) = dy -1/2dx

If d > 0) we choose NE(northeast pixel)

F(NorthEast pixel) = dy(x + 1) – dx(y + 1) + b\*dx

F(NorthEast pixel) = dy\*x – dx\*y + b\*dx + dy - dx

F(NorthEast pixel) = F(previous pixel) + dy - dx

F(NorthEast pixel) = dy – dx

F(previous) += F(NorthEast pixel)

If d <= 0) we choose E(east pixel)

F(East pixel) = dy(x + 1) – dx\*y + b\*dx

F(East pixel) = dy\*x – dx\*y + b\*dx + dx

F(East pixel) = F(previous pixel) + dx

F(East pixel) = dx

F(previous) += F(East pixel)

To eliminate fraction(1/2 in d), we need to double to original F(F that is previous above lecture)

2F(previous) = 2dy \* x – 2dx \* y – 2b \* dx = 0

2d(Middlepoint) = 2dy – dx

2F(NorthEast pixel) = 2dy -2dx

2F(East pixel) = 2dx

Like DDA, it needs to be changed so that the function is calculated in all octants.

d(x + 1, y +1/2) = dy(x+1) – dx(y + ½) – b\*dx // octant 0

d(x + 1, y -1/2) = dy(x+1) – dx(y - ½) – b\*dx // octant 7

d(x - 1, y +1/2) = dy(x-1) – dx(y + ½) – b\*dx // octant 3

d(x - 1, y -1/2) = dy(x-1) – dx(y - ½) – b\*dx // octant 4

Because the rest of octants(1,2,5,6), |dx| > |dy|, therefore change the value based on x-axis into the value based on y-axis.

d(x + 1/2, y +1) = dy(x+1) – dx(y + ½) – b\*dx // octant 1

d(x + 1/2, y -1) = dy(x+1) – dx(y - ½) – b\*dx // octant 6

d(x – 1/2, y +1) = dy(x-1) – dx(y + ½) – b\*dx // octant 2

d(x – 1/2, y -1) = dy(x-1) – dx(y - ½) – b\*dx // octant 5

Pseudo-code3)

DrawLine(int x0, int y0, int x1, int y1, Color I)

{

Int dx = x1 – x0, dy = y1 – y0, x = x0, y = y0;

Int ystep = (dy >= 0) ? 1: -1;

Int xstep = (dx >= 0) ? 1 : -1;

Dx = (dx >= 0) ? dx : -dx;

Dy = (dy >= 0) ? dy : dy;

Setpix(x,y,I);

Int d = 2 \* (dx – dy), dN = 2 \* dx, dNE = 2 \* (dx – dy);

While(dy--)

{

Y += ystep;

If(d>=0)

{

x += xstep;

d += dNE;

}

Setpix(x,y,I);

}

}  
  
  
example)

(0,0) -> (8, 4)

Octant 0

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| P1 | 0 | 0.5 | 1 | 1.5 | 2 | 2.5 | 3 | 3.5 | 4 |
| result | (0,0) | (1,0) | (2,1) | (3,1) | (4,2) | (5,2) | (6,3) | (7,3) | (8,4) |

(0,0) -> (4,8)

Octant 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P0 | 0 | 1 | 2 | 3 | 4 |
| P1 | 0 | 2 | 4 | 6 | 8 |
| result | (0,0) | (1,2) | (2,4) | (3,6) | (4,8) |

(0,0) -> (-4,8)

Octant 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P0 | 0 | -1 | -2 | -3 | -4 |
| P1 | 0 | 2 | 4 | 6 | 8 |
| result | (0,0) | (-1,2) | (-2,4) | (-3,6) | (-4,8) |

(0,0) -> (-8,4)

Octant 3

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P0 | 0 | -1 | -2 | -3 | -4 | -5 | -6 | -7 | -8 |
| P1 | 0 | 0.5 | 1 | 1.5 | 2 | 2.5 | 3 | 3.5 | 4 |
| result | (0,0) | (-1,0) | (-2,1) | (-3,1) | (-4,2) | (-5,2) | (-6,3) | (-7,3) | (-8,4) |

(0,0) -> (-8,-4)

Octant 4

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P0 | 0 | -1 | -2 | -3 | -4 | -5 | -6 | -7 | -8 |
| P1 | 0 | -0.5 | -1 | -1.5 | -2 | -2.5 | -3 | -3.5 | -4 |
| result | (0,0) | (-1,0) | (-2,-1) | (-3,-1) | (-4,-2) | (-5,-2) | (-6,-3) | (-7,-3) | (-8,-4) |

(0,0) -> (-4,-8)

Octant 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P0 | 0 | -1 | -2 | -3 | -4 |
| P1 | 0 | -2 | -4 | -6 | -8 |
| result | (0,0) | (-1,-2) | (-2,-4) | (-3,-6) | (-4,-8) |

(0,0) -> (4,-8)

Octant 6

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P0 | 0 | 1 | 2 | 3 | 4 |
| P1 | 0 | -2 | -4 | -6 | -8 |
| Result | (0,0) | (1,-2) | (-1,-4) | (3,-6) | (4,-8) |

(0,0) -> (8,-4)

Octant 7

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| P1 | 0 | -0.5 | -1 | -1.5 | -2 | -2.5 | -3 | -3.5 | -4 |
| result | (0,0) | (1,0) | (2,-1) | (3,-1) | (4,-2) | (5,-2) | (6,-3) | (7,-3) | (8,-4) |