Line and Circle Rasterization Algorithms

2D screen coordinate systems (0–based)

(1) pixels as grid cells

(2) pixels as intersection points

We use convention (2)
Line−Drawing Algorithm

rasterize_line_segment(x0, y0, x1, y1);
For a given pair of points \((x_0, y_0)\) and \((x_1, y_1)\), let 
\[dx = x_1 - x_0, \quad and \quad dy = y_1 - y_0\]
we can compute the slope as:
\[m = \frac{y_1 - y_0}{x_1 - x_0} = \frac{dy}{dx}\]
and
intercept \(b = y_1 - m \cdot x_1\)

Given the slope, we have:
\[dy = m \cdot dx\]
also
\[dx = dy \cdot \frac{1}{m}\]

We can use the above equation to compute a sequence of discrete points along the line segment:
Starting from \( X = x_0 \) \( Y = y_0 \)

If we choose the next point’s x value at:

\[ X' = X + \Delta X \]

Then we have:

\[ Y' = Y + ?? \]

Now consider the screen (discrete) space ...

our goal: finding the closest pixels (integer x and y) along the line path

To do this, we can set \( \Delta x = 1 \), and we will have:

\[
\begin{align*}
X_1 &= x_0 + 1 & X_k &= X_{k-1} + 1 \\
Y_1 &= y_0 + m & Y_k &= Y_{k-1} + m
\end{align*}
\]

plot \((X_1, \text{int}(Y_1))\) ... plot \((X_k, \text{int}(Y_k))\)
DDA Line–Drawing Algorithm (cont’d)

What about a line looks like this?

Can we still use $\Delta X = 1$ and compute $y$?

No! Because if we do so, we will end up with.....

Instead, we should increment $Y$ by 1 in this case:

Set $\Delta y = 1$, and we will have:

$X_1 = x_0 + 1/m$

$Y_1 = y_0 + 1$

$X_k = X_{k-1} + 1/m$

$Y_k = Y_{k-1} + 1$

plot (int($X_1$), $Y_1$)  ...  plot (int($X_k$), $Y_k$)
DDA Line–Drawing Algorithm (cont’d)

The above algorithm is called DDA (Digital Differential Analyzer) algorithm because it is based on $\Delta X$ and $\Delta Y$

So far in our discussions the slope $m$ is greater than 0, what to do if $m$ is negative?

Read page 87–88 in the textbook

DDA Algorithm has two problems:

1) Numerical errors (could be bad for long line segments)

2) Floating point operations --- Too slow
Bresenham’s Line Drawing Algorithm

Only incremental integer operations are used

In the case of \( 0 < m < 1 \), we sample the line at unit \( X \) intervals, what we really need to decide is --

Should we increment \( Y \) now?

: pixel has been drawn

: draw northeast pixel?

: draw east pixel?
Bresenham’s Line Drawing Algorithm (cont’d)

The key idea:

Should we increment $Y$?

If $d_1 > d_2$  
Use NE pixel ($y = Y' + 1$)

if $d_2 > d_1$  
Use E pixel ($y = Y'$)

Can we compute and compare $d_1$ and $d_2$ using pure integer operations?
Bresenham’s Line Drawing Algorithm (cont’d)

Assuming that the k–th pixels \((X_k, Y_k)\) along the line has been drawn, and \(0 < m < 1\):

Now consider the k+1 pixel \((X, Y)\)

We know: \(X = X_k + 1\)

We want to decide whether

\[
\begin{align*}
Y &= Y_k \quad (d2 > d1) \quad \text{or} \\
Y &= Y_k + 1 \quad (d1 > d2)
\end{align*}
\]

Remember the line equation: \(Y = mx + b\)

so we have \(Y = m(X_k + 1) + b\)

\[
\begin{align*}
d1 &= Y - Y_k = m(X_k + 1) + b - Y_k \\
d2 &= (Y_k + 1) - Y = Y_k + 1 - m(X_k + 1) - b
\end{align*}
\]

\[
d1 - d2 = 2m(X_k + 1) - 2Y_k + 2b - 1
\]

Substitute \(m = \frac{dy}{dx}\)

\[
d1 - d2 = 2\left(\frac{dy}{dx}\right)(X_k + 1) - 2Y_k + 2b - 1
\]

\[
dx(d1-d2) = 2dy(X_k + 1) - 2dxY_k + \\
dx(2b-1)
\]

let \(2dy + dx(2b-1) = c\) \ We have:

\[
dx(d1-d2) = 2dy \cdot X_k - 2dx \cdot Y_k + C
\]
Bresenham’s Line Drawing Algorithm (cont’d)

\[ dx \ (d1-d2) = 2dy \cdot X_k - 2 \ dx \cdot Y_k + C \]

Remember \( C = 2 \ dy + dx \ (2b-1) \) and

\[
\begin{align*}
dy &= y_1 - y_0 \text{ (y distance of two end points)} \\
dx &= x_1 - x_0 \text{ (x distance of two end points)}
\end{align*}
\]

Let \( P_k = dx \ (d1 - d2) \)

If we only consider the \( dx > 0 \) (draw the line from left to right):

then \( P_k \) has the same sign as \( (d1 - d2) \)

That is:

\[
\begin{align*}
\text{if} \quad P_k < 0 & \Rightarrow d1 < d2 \quad \Rightarrow \quad Y_{k+1} = Y_k \quad \text{(take the E Pixel)} \\
\text{else} \quad d1 > d2 & \Rightarrow \quad Y_{k+1} = Y_k + 1 \quad \text{(take the NE pixel)}
\end{align*}
\]
Given \( P_k \), we can compute \( P_{k+1} \) incrementally:

Consider

\[
P_k = 2 \, dy \cdot X_k - 2 \, dx \cdot Y_k + C
\]

We have:

\[
P_{k+1} = 2 \, dy \cdot X_{k+1} - 2 \, dx \cdot Y_{k+1} + C \quad \text{where}
\]

\((X_{k+1}, Y_{k+1})\) is the \((k+1)\)th pixel

\[
P_{k+1} - P_k = 2 \, dy \,(X_{k+1} - X_k) - 2 \, dx \,(Y_{k+1} - Y_k)
\]

\[
= 2 \, dy - 2 \, dx \,(Y_{k+1} - Y_k)
\]

So,

\[
P_{k+1} = P_k + 2 \, dy - 2 \, dx \,(Y_{k+1} - Y_k)
\]

The initial condition \( P_0 \):

\[
P_0 = 2 \, dy \cdot X_0 - 2 \, dx \cdot Y_0 + 2 \, dy
\]

\[
+ dx \,(2 \, Y_0 - 2 \, dy/dx \cdot X_0 - 1)
\]

\[
= 2 \, dy - d \, x
\]
Bresenham’s Line Algorithm

Given end points \((x_0, y_0) \ (x_1, y_1)\)
\[dx = x_1 - x_0, \ dy = y_1 - y_0\]

Starting with an end point \((x_0, y_0)\):
1. Compute \(P_0 = 2dy - dx\)
2. For each \(K\), starting with \(k = 0\)
   
   if \((P_k < 0)\)
   
   the next point is \((X_k + 1, Y_k)\) // E pixel
   \[P_{k+1} = P_k + 2dy\]
   
   else
   
   the next point is \((X_k + 1, Y_k + 1)\) // NE pixel
   \[P_{k+1} = P_k + 2(dy - dx)\]
3. Repeat step 2 \(x_1 - x_0\) times
Ask yourself ...

In the class, we assumed that $0 < m < 1$ and $dx > 0$, where $m$ is the slope of the line segment and $dx = x_1 - x_0$

Describe how we should modify the algorithm at the bottom of page 90 in the textbook for an arbitrary slope $m$ and $dx$?

(hint: read page 91, 92)