Rasterization

Converts
- lines and triangles
- with floating point vertices
- in viewport (screen) coordinates
into
- pixels
- with integer coordinates
- in viewport (screen) coordinates

pixels centered at grid vertices, not grid cells
Need to rasterize lines between any two clipped screen points, from \((x_0, y_0)\) to \((x_1, y_1)\)
Line Rasterization

Need to rasterize lines between any two clipped screen points, from \((x_0, y_0)\) to \((x_1, y_1)\)

Only rasterize lines from the origin to a point in the first octant
Line Rasterization

Need to rasterize lines between any two clipped screen points, from \((x_0, y_0)\) to \((x_1, y_1)\)

Translate \((x_0, y_0)\) to origin

Only rasterize lines from the origin to a point in the first octant
Line Rasterization

Need to rasterize lines between any two clipped screen points, from \((x_0, y_0)\) to \((x_1, y_1)\)

Translate \((x_0, y_0)\) to origin

\[
\begin{align*}
(x_0 - x_0, y_1 - y_0) \quad &\rightarrow \quad (0, y_1 - y_0) \\
(-x_1 - x_0, -y_1 + y_0) \quad &\rightarrow \quad (-x_1, y_1)
\end{align*}
\]

Only rasterize lines from the origin to a point in the first octant

Reflect \((x_1 - x_0, y_1 - y_0)\) to first octant
Line Rasterization

(-x, -y)
Line Rasterization

(-x, -y)
How to rasterize a line from (0,0) to (4,3)
- Pixel (0,0) and (4,3) easy
- One pixel for each integer x-coordinate
- Pixel’s y-coordinate closest to line
- If line equal distance between two pixels, pick on arbitrarily but consistently
Midpoint Algorithm

- Which pixel should be plotted next?
  - East?
  - Northeast?
Midpoint Algorithm

• Which pixel should be plotted next?
  – East?
  – Northeast?
• Line equation
  \[ y = mx + b \]
  \[ m = (y_1 - y_0)/(x_1 - x_0) \]
  \[ b = y_0 - mx_0 \]
  \[ f(x,y) = mx + b - y \]
Midpoint Algorithm

- Which pixel should be plotted next?
  - East?
  - Northeast?
- Line equation
  \[ y = mx + b \]
  \[ m = \frac{(y_1 - y_0)}{(x_1 - x_0)} \]
  \[ b = y_0 - mx_0 \]
  \[ f(x, y) = mx + b - y \]

\[ f > 0 \]
Midpoint Algorithm

- Which pixel should be plotted next?
  - East?
  - Northeast?
- Line equation
  \[ y = mx + b \]
  \[ m = \frac{(y_1 - y_0)}{(x_1 - x_0)} \]
  \[ b = y_0 - mx_0 \]
  \[ f(x, y) = mx + b - y \]
Midpoint Algorithm

- Which pixel should be plotted next?
  - East?
  - Northeast?
- Line equation

  \[ y = mx + b \]

  \[ m = \frac{y_1 - y_0}{x_1 - x_0} \]

  \[ b = y_0 - mx_0 \]

  \[ f(x, y) = mx + b - y \]

- \[ f(M) \geq 0 \rightarrow NE \]
- \[ f(M) < 0 \rightarrow E \]
Pixel Increments

\[ f(x,y) = mx + b - y \]

\[ M = P + (1, \frac{1}{2}) \]

\[ f(M) = f(x+1, y+\frac{1}{2}) \]

\[ = m(x+1) + b - (y+\frac{1}{2}) \]

\[ = mx + m + b - y - \frac{1}{2} \]

\[ = mx + b - y + m - \frac{1}{2} \]

\[ = f(P) + m - \frac{1}{2} \]

\[ f(0,0) = b \]

\[ = 0 \text{ if line starts at origin} \]
Midpoint Increments

\[ f(M) = f(P) + m - \frac{1}{2} \]

\[ f(M_E) = f(x+2, y+\frac{1}{2}) = m(x+2) + b - (y+\frac{1}{2}) = f(P) + 2m - \frac{1}{2} = f(M) + m \]

\[ f(M_{NE}) = f(x+2, y+1\frac{1}{2}) = m(x+2) + b - (y+1\frac{1}{2}) = f(P) + 2m - 1\frac{1}{2} = f(M) + m - 1 \]

\[ f(1, \frac{1}{2}) = m + b - \frac{1}{2} = m - \frac{1}{2} \text{ if line starts at origin} \]
\[ f(M_E) = f(M) + m \]
\[ f(M_{NE}) = f(M) + m - 1 \]
\[ f(1, \frac{1}{2}) = m + b - \frac{1}{2} \]
\[ b = 0 \]
\[ m = \frac{(y_1 - y_0)}{(x_1 - x_0)} = \Delta y / \Delta x \]

\[ \Delta x f(M_E) = \Delta x f(M) + \Delta y \]
\[ \Delta x f(M_{NE}) = \Delta x f(M) + \Delta y - \Delta x \]
\[ \Delta x f(1, \frac{1}{2}) = \Delta y - \frac{1}{2} \Delta x \]

\[ 2\Delta x f(M_E) = 2\Delta x f(M) + 2\Delta y \]
\[ 2\Delta x f(M_{NE}) = 2\Delta x f(M) + 2\Delta y - 2\Delta x \]
\[ 2\Delta x f(1, \frac{1}{2}) = 2\Delta y - \Delta x \]
2Δxf(M_E) = 2Δx f(M) + 2Δy
2Δxf(M_{NE}) = 2Δx f(M)+2Δy-2Δx
2Δxf(1, ½) = 2Δy − Δx

F(M_E) = F(M) + 2Δy
F(M_{NE}) = F(M) + 2Δy − 2Δx
F(1,½) = 2Δy − Δx
Integer Math

\[ F(M_E) = F(M) + 2\Delta y \]
\[ F(M_{NE}) = F(M) + 2\Delta y - 2\Delta x \]
\[ F(1, \frac{1}{2}) = 2\Delta y - \Delta x \]

The Bresenham Line Algorithm

```c
line(int x0, int y0, int x1, int y1) {
    int dx = x1 - x0;
    int dy = y1 - y0;
    int F = 2*dy - dx;
    int dFE = 2*dy;
    int dFNE = 2*dy - 2*dx;
    int y = y0;
    for (int x = x0, x < x1; x++) {
        plot(x,y);
        if (F < 0) {
            F += dFE;
        } else {
            F += dFNE; y++;
        }
    }
}
```