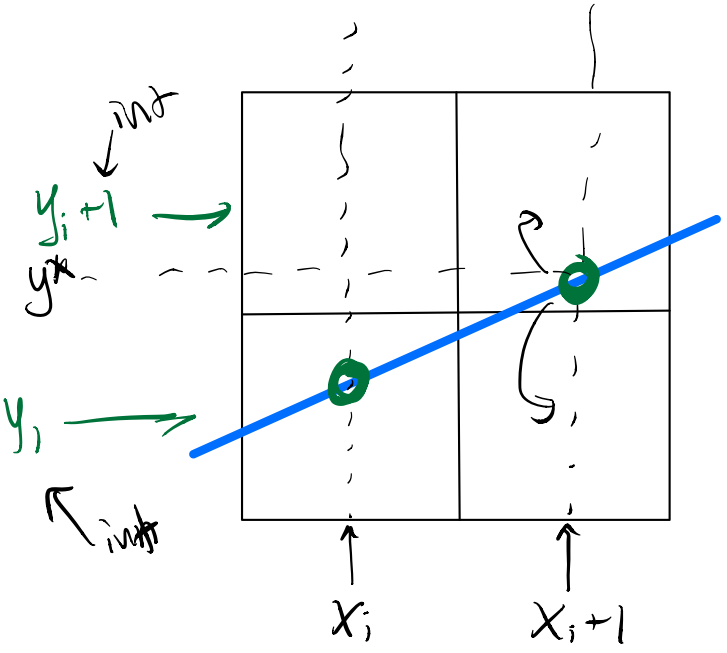
Midpoint Algorithm



Intuition: 1 pixel per col

pick pixel the line spends most time in

Equivalently: the pixel the line is in at integer x

Algorithm: $(y = mx+b)$

```
// compute m, b
for x = xmin : xmax
    y = m * x + b
    draw(x, round(y))
```

Efficiency?

3 flops

Faster Midpoint Algorithm

Original:

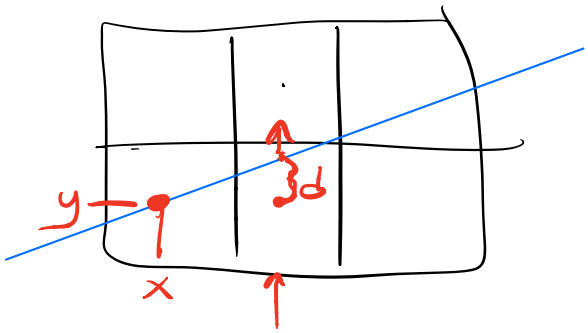
```
// compute m, b
for x = xmin : xmax:
    y ← b + m * x
    draw(x, round(y))
```

Faster:

```
// compute m, b
y = m * xmin + b
for x = xmin : xmax+1
    draw(x, round(y))
    y += m
```

Fasterer Midpoint Algorithm?!

```
//compute m, b
y = round(m * xmin + b)
for x = xmin : xmax
    draw(x, round(y))
    d = m(x+1) + b - y
    if(d > 0.5)
        y += 1
        d -= 1
```



← distance from line to current y

Strategy: incrementalize d as we did y above.

Flops : 1x >, 1x +, and 0 or 1x -

↑ shift by 0.5, This becomes a sign bit check!

$(d > 0.0)$

Interpolating Values

Given: P_1, P_2, V_1, V_2

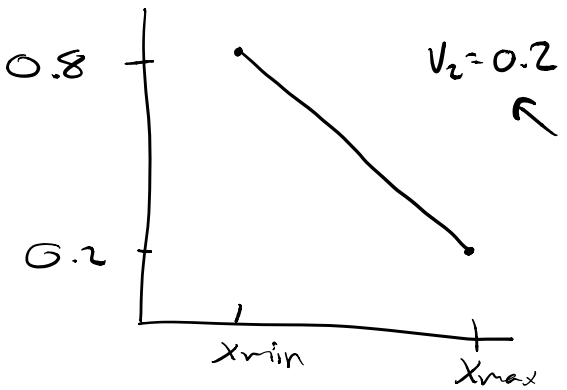
↑ endpoints

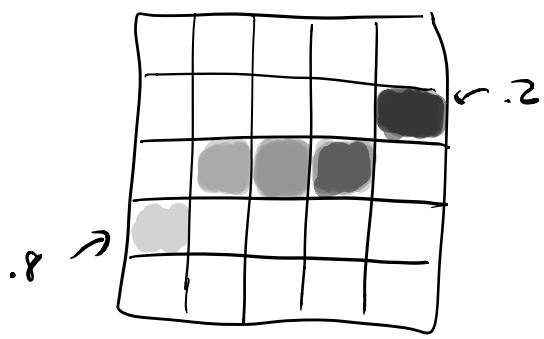
↖ values of same property at P_1, P_2

e.g.: color
any vertex data

Interpolate a V for each pixel

Example: $V_1 = 0.8$






```
// calc. m, b
y = m x_min + b
// calc vm, vb
V = Vm x_min + Vb
for x = x_min : x_max
    draw(x, round(y), V)
    y += m
    V += Vm
```

2/23 Announcements

- Setup FP repo by tomorrow night
  (GH classroom, push proposal + feedback)
- A3 due W night
  - Point lights have r^2 falloff
  - Ambient light $I_a^{(ka)}$ is a constant added to all pixel colors
- A2 grading feedback (except extensions)
  is on Github in the "Feedback" pull request
- Today's lab is completed in your A0 repo
  - push finished changes by F night.
  - 10 pts in written HW category