CG – T5 - Rasterization

L:CC, MI:ERSI

Miguel Tavares Coimbra

(course and slides designed by Verónica Costa Orvalho)

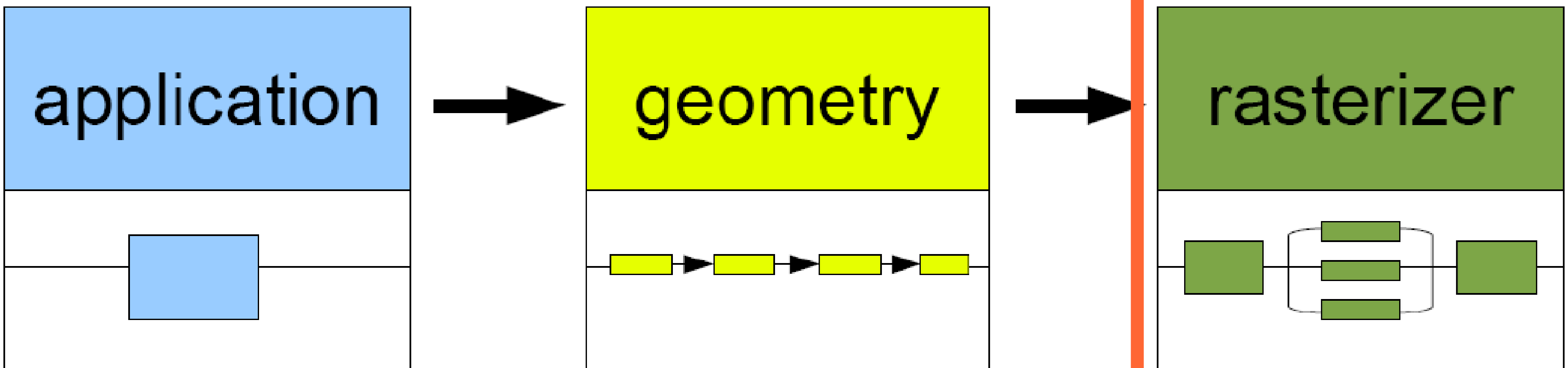
What is rasterization?

Basic steps for creating a 2D image out of a 3D world

- Create the 3D world
  - Vertexes and triangles in a 3D space
- Project it to a 2D 'camera'
  - Use perspective to transform coordinates into a 2D space
- Paint each pixel of the 2D image
  - Rasterization, shading, texturing
  - Will break this into smaller things later on
- Enjoy the super cool image you have created

Today

pipeline



. **collision** detection
. **animation** global acceleration
. **physics** simulation

process on CPU or GPU

. **transformation**
. **projection**

Computes:
. what is to be drawn
. how should be drawn
. where should be drawn

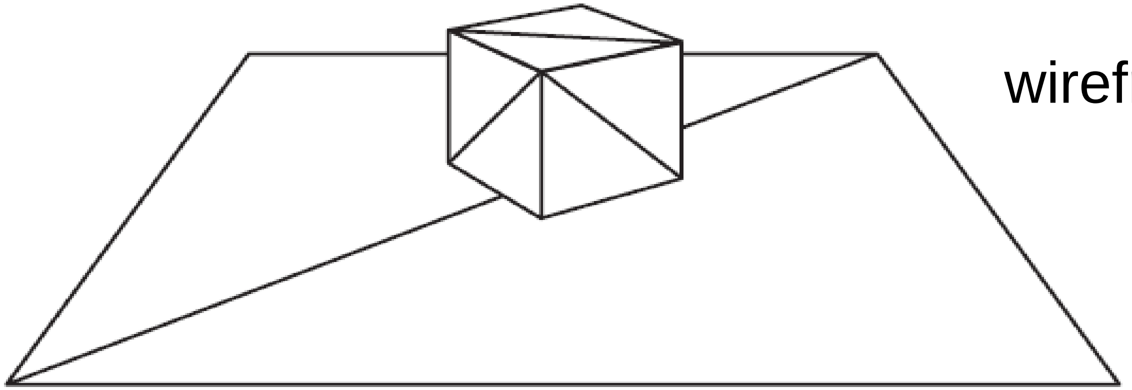
process on GPU

. **draws** images generated by **geometry stage**

process on GPU

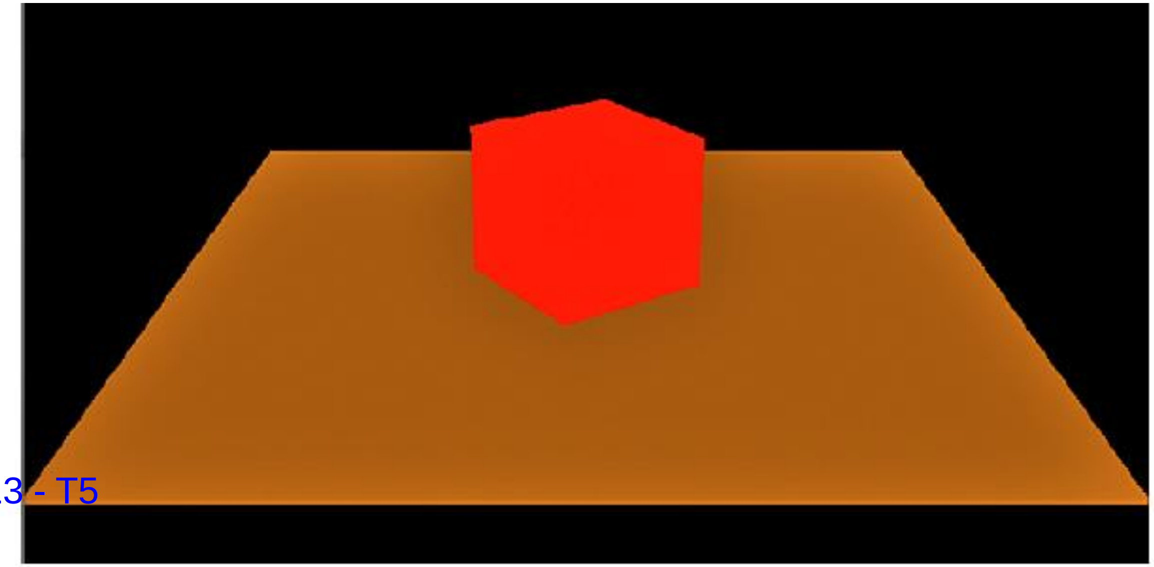
Rasterization

rasterization:

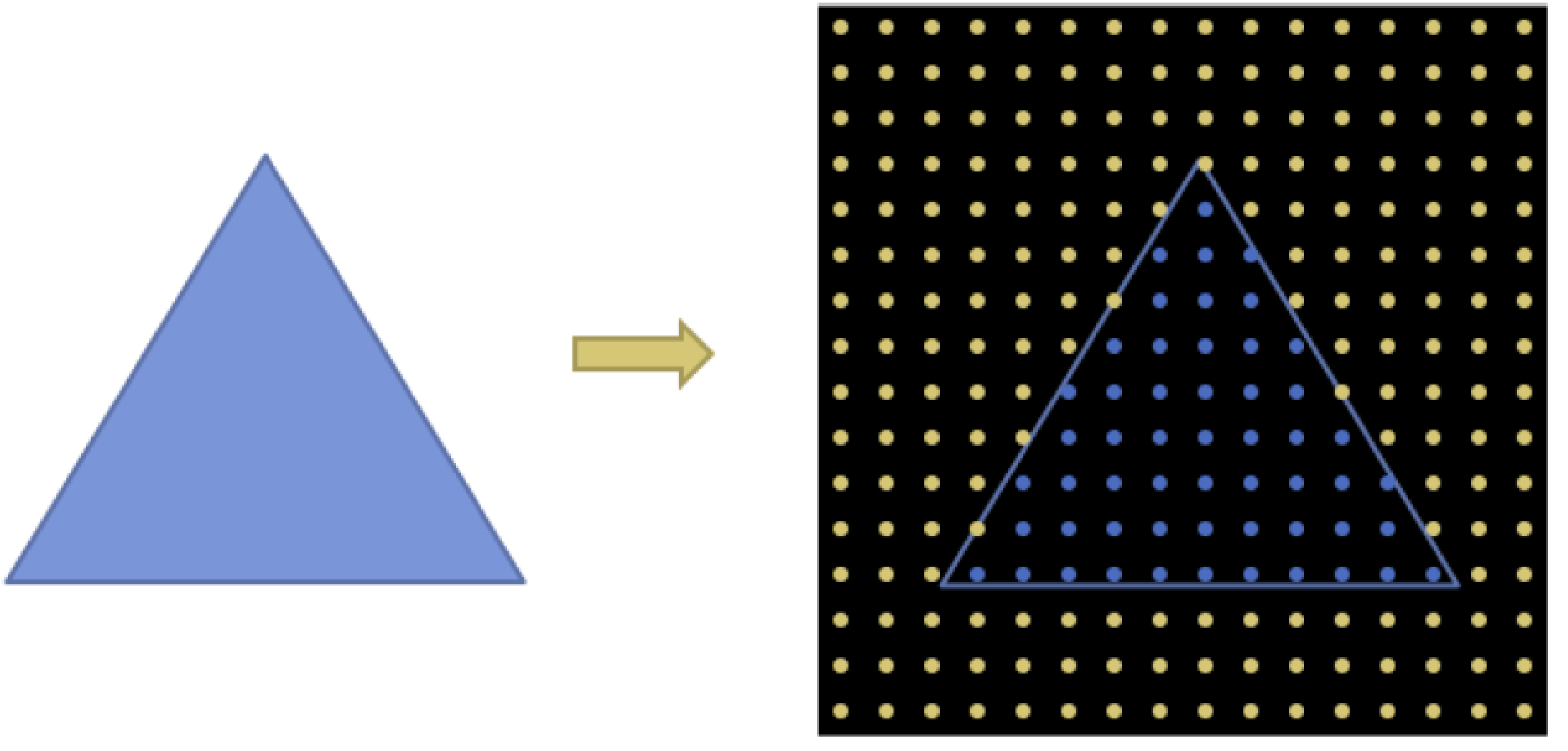


wireframe

filling with colors



Rasterization



Slide by Ron Fedkiw, Stanford University

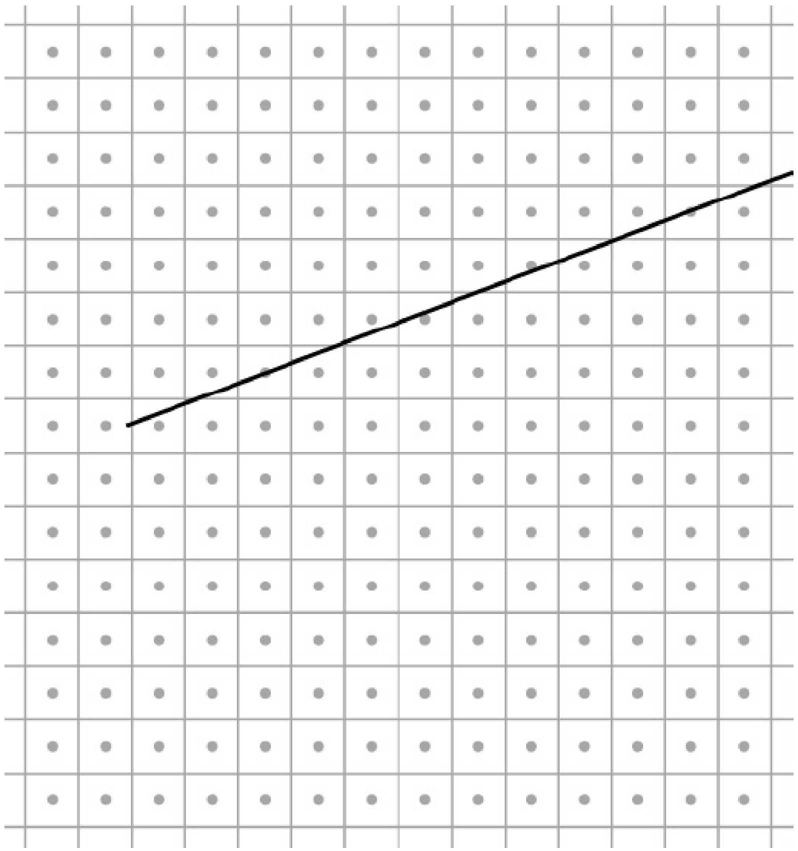
How do we do this?

Primitives

- Only three!
  - Points
  - Line segments
  - Triangles
- How do I rasterize them?
  - Points are simple
  - Lines?
  - Triangles?

Rasterizing lines

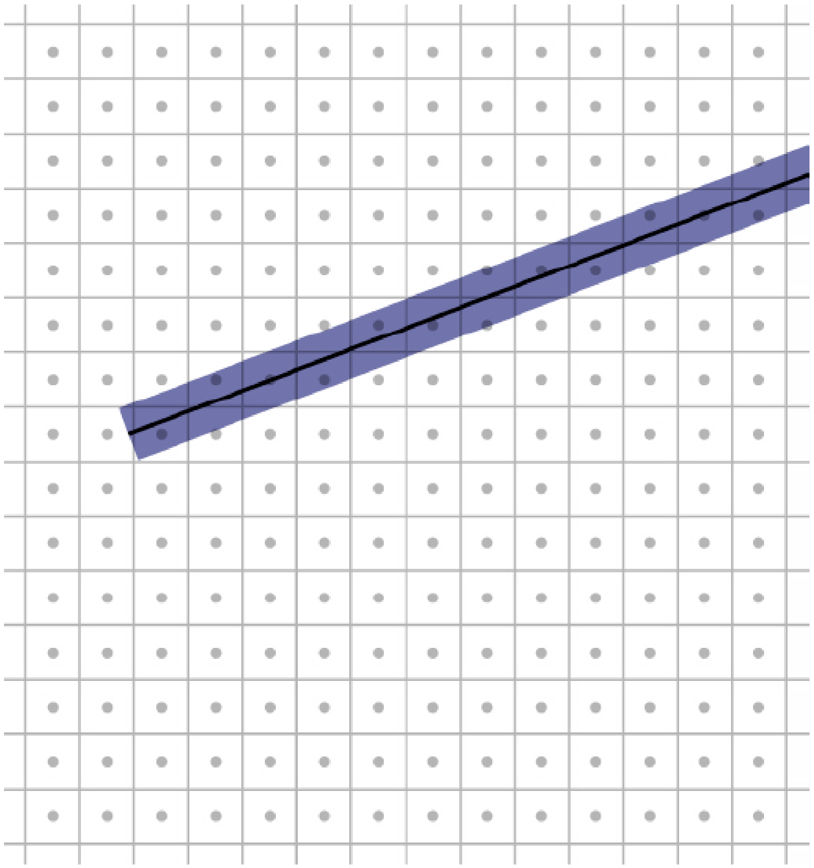
- Lines are defined by two points
  - Projected into my 2D screen from my 3D world
- Consider it a rectangle
  - So that it occupies a non-zero area



Adapted from Steve Marschner, Cornell University

Rasterizing lines

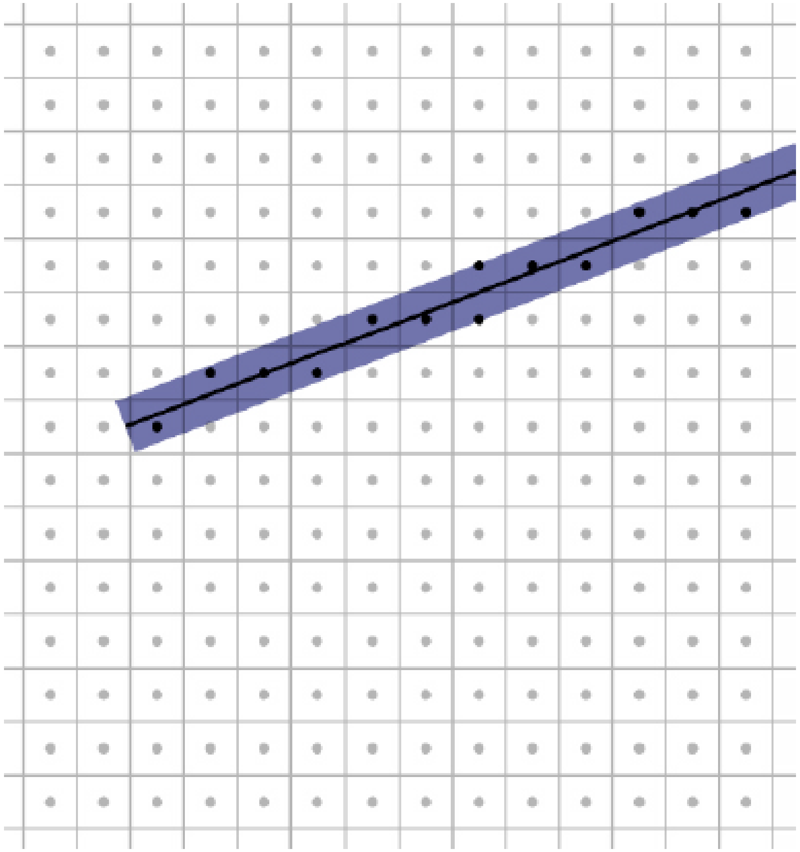
- Lines are defined by two points
  - Projected into my 2D screen from my 3D world
- Consider it a rectangle
  - So that it occupies a non-zero area



Adapted from Steve Marschner, Cornell University

Point sampling

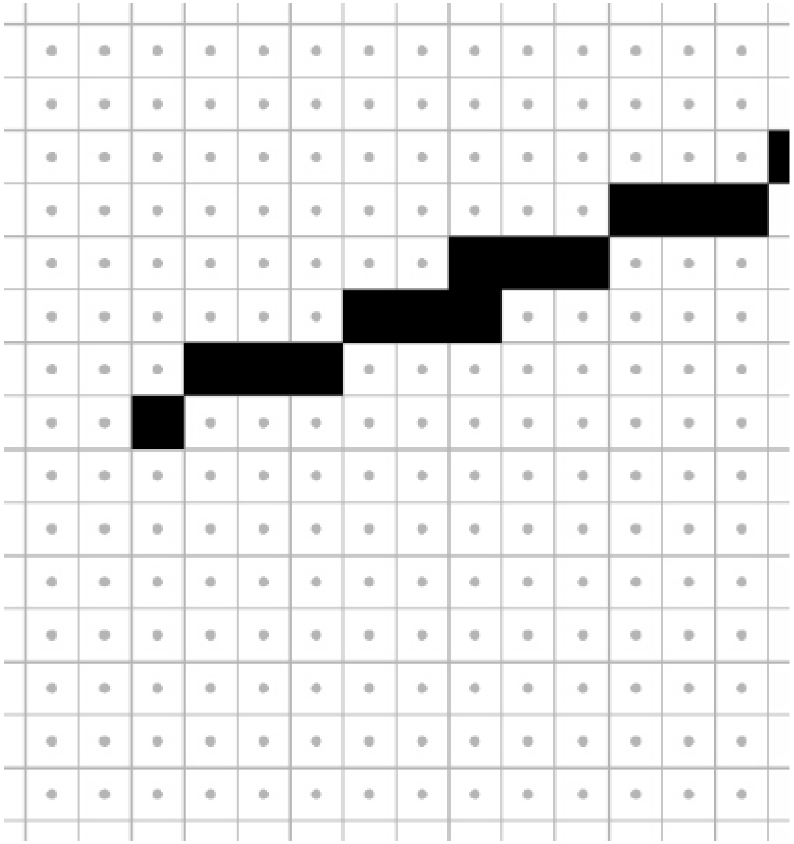
- Draw all the pixels whose centers fall within the rectangle
- It may draw undesired adjacent pixels…



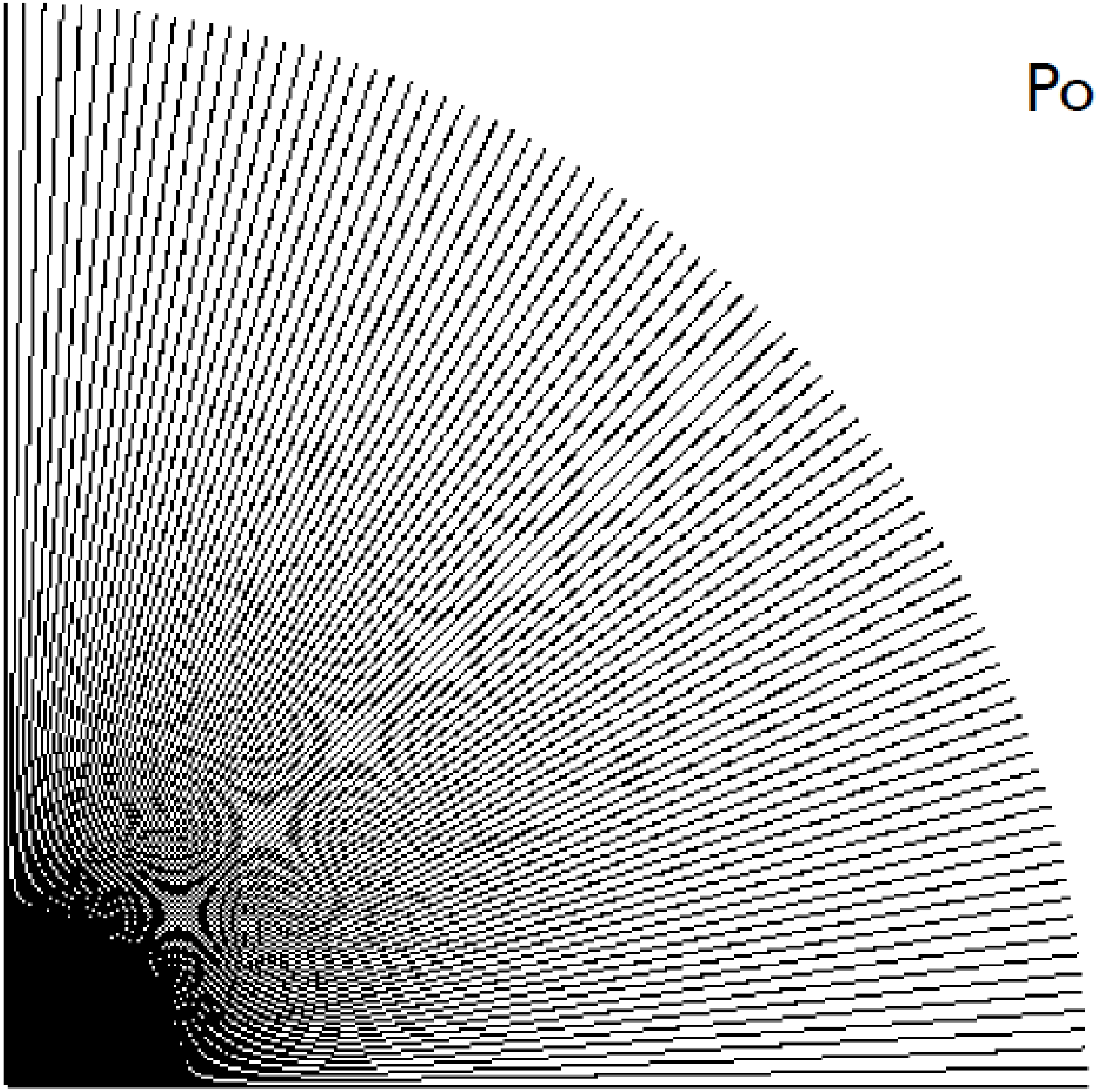
Adapted from Steve Marschner, Cornell University

Point sampling

- Draw all the pixels whose centers fall within the rectangle
- It may draw undesired adjacent pixels…



Adapted from Steve Marschner, Cornell University



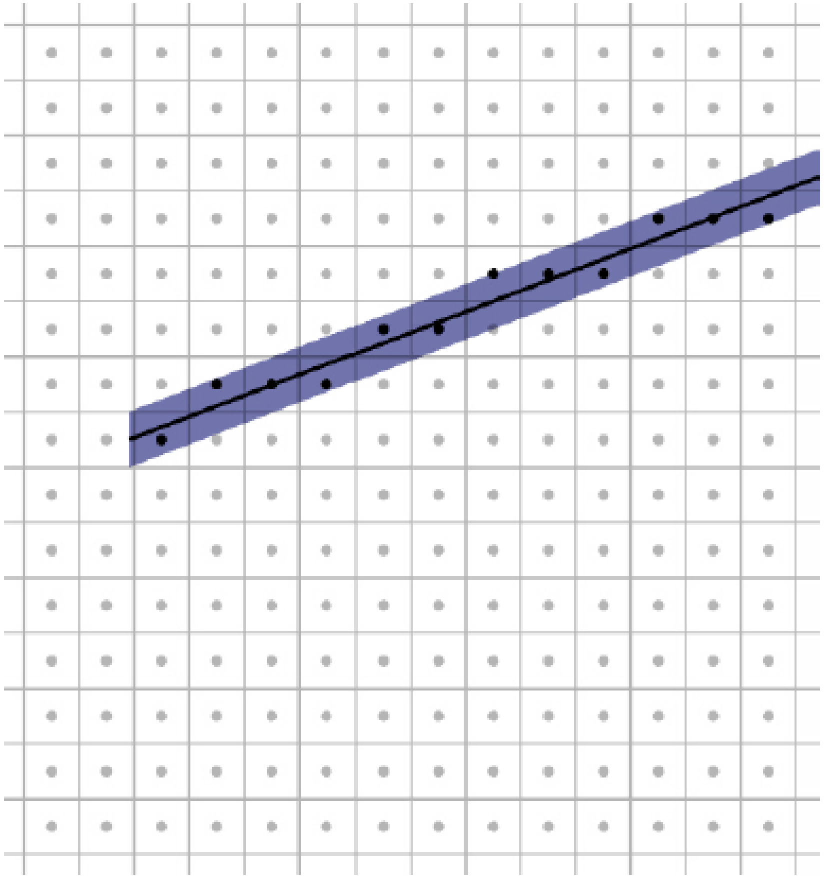
Point sampling in action

© 2008 Steve Marschner • 8

Adapted from Steve Marschner, Cornell University

Bresenham lines (midpoint alg.)

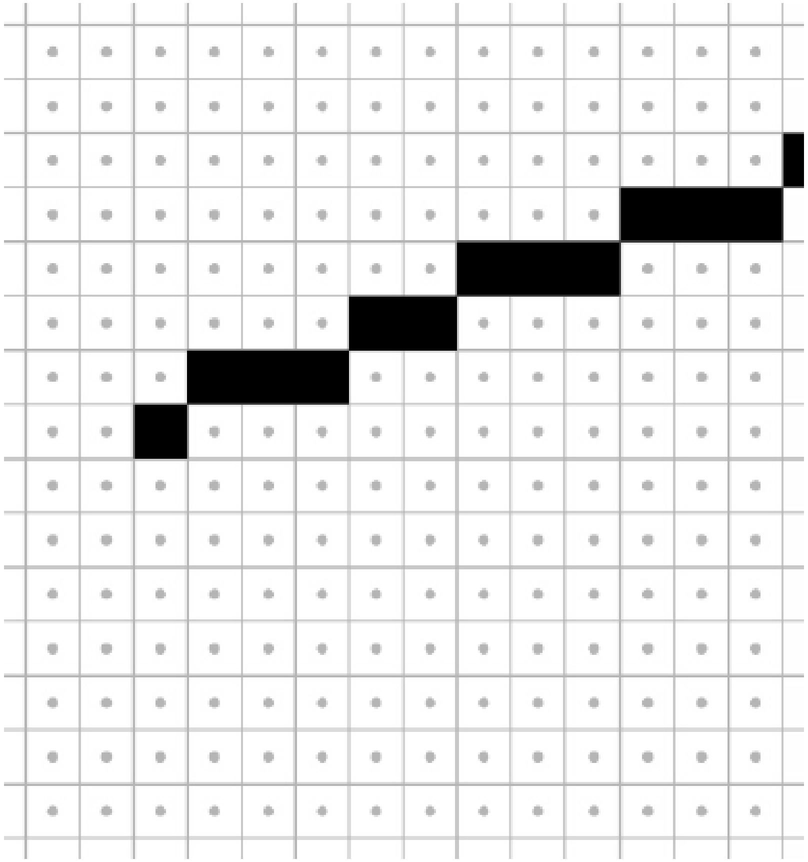
- Idea:
  - Define line width parallel to pixel grid
- What does this mean?
  - Turn on the single nearest pixel in each column



Adapted from Steve Marschner, Cornell University

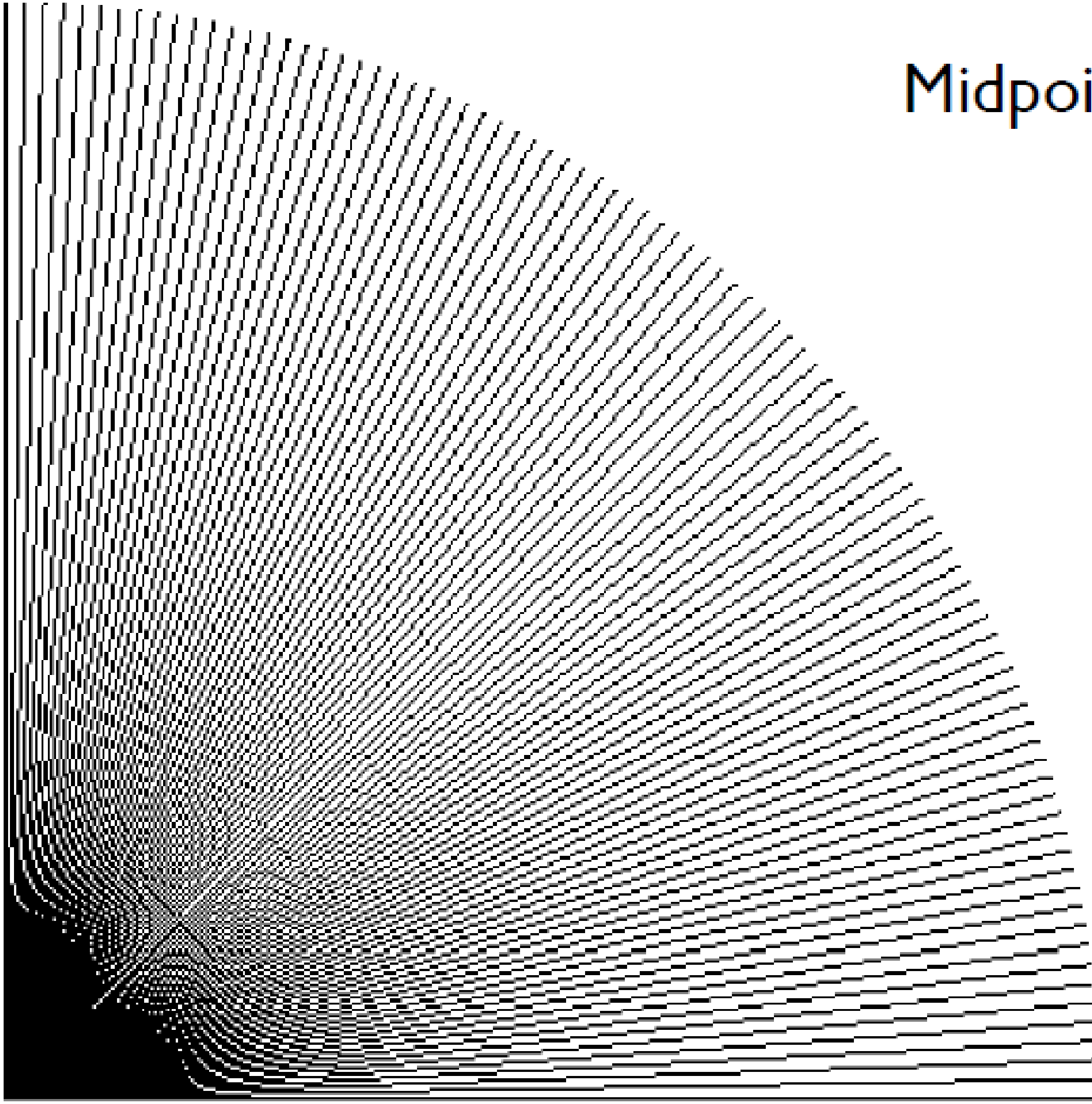
Bresenham lines (midpoint alg.)

- Idea:
  - Define line width parallel to pixel grid
- What does this mean?
  - Turn on the single nearest pixel in each column



Adapted from Steve Marschner, Cornell University

Midpoint algorithm in action



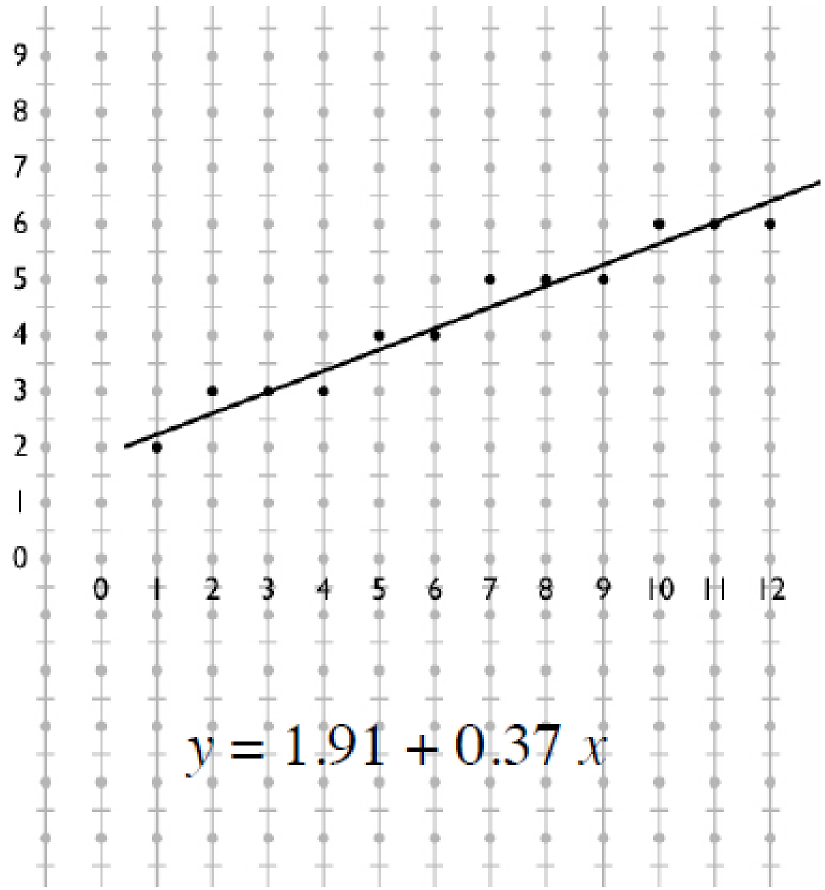
© 2008 Steve Marschner • 10

Adapted from Steve Marschner, Cornell University

Algorithms for drawing lines

- Simple
  - Evaluate line equation per column
- Line equation
  - y=b+m.x

```
for x = ceil(x0) to floor(x1)
    y = b + m*x
    output(x, round(y))
```

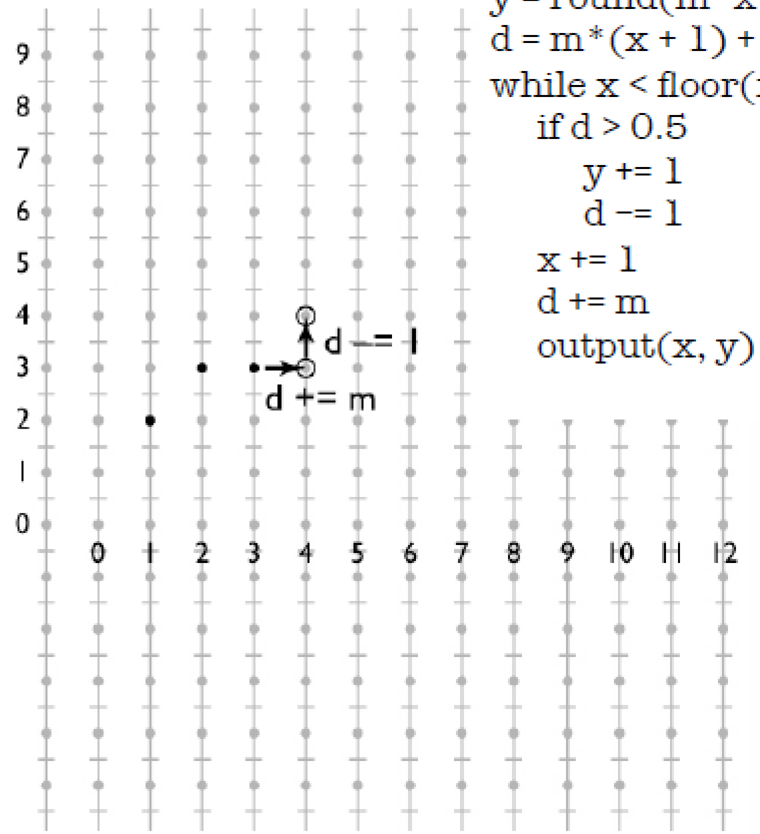


Adapted from Steve Marschner, Cornell University

Optimized line drawing

- Multiplying and rounding is slow
- We can add the vertical displacement to our previous vertical coordinate (d)
  - Initially: d = m(x+1)+b-y
  - Then: d+=m
- We call this DDA (digital differential analyzer)

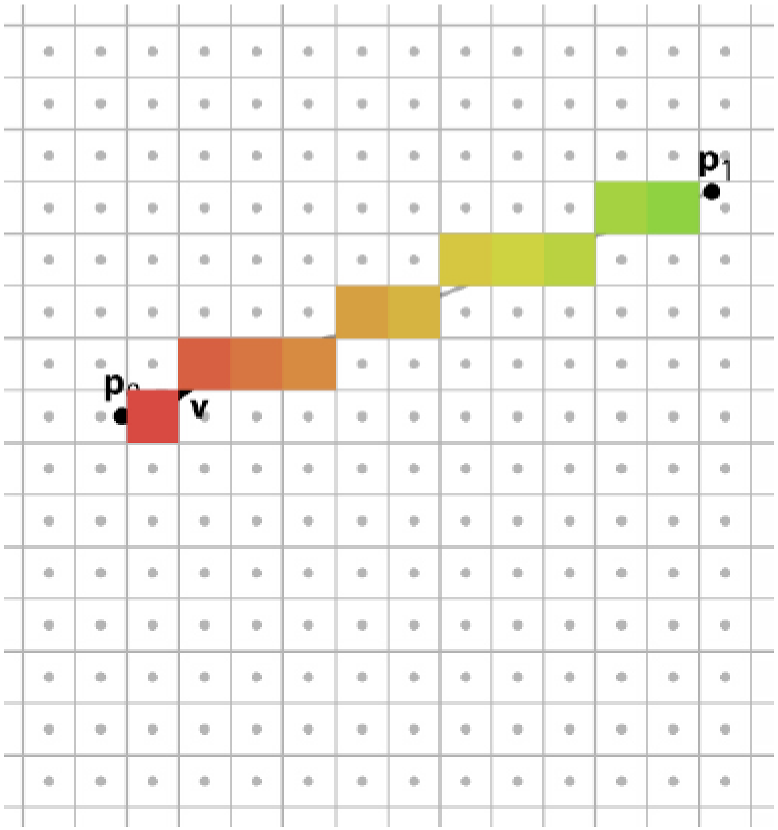
```
x = ceil(x0)
y = round(m*x + b)
d = m*(x + 1) + b − y
while x < floor(x1)
   if d > 0.5
      y += 1
      d −= 1
   x += 1
   d += m
   output(x, y)
```



Adapted from Steve Marschner, Cornell University

Interpolation along lines

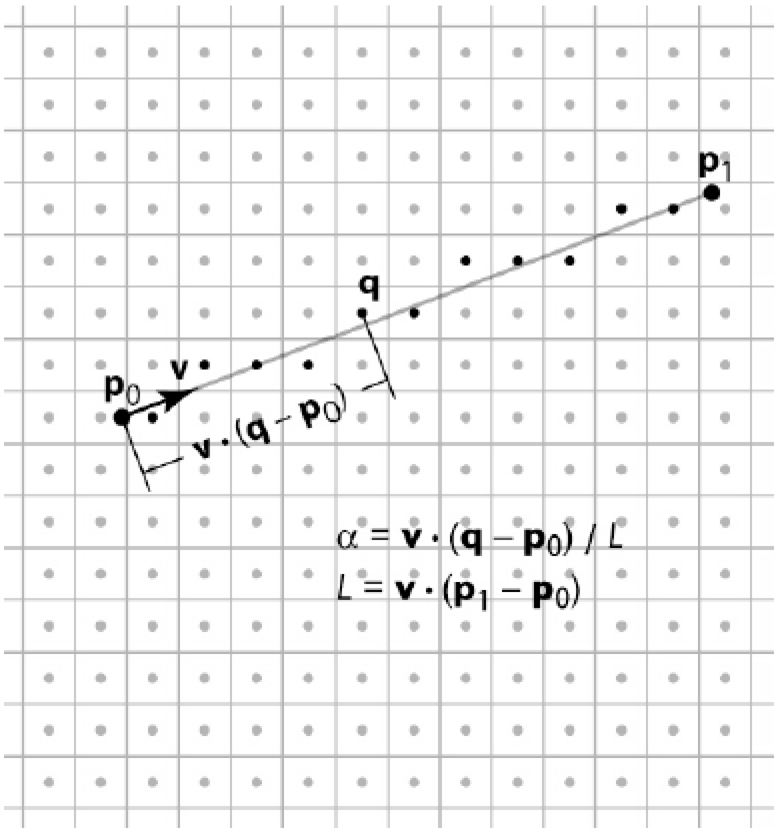
- We don't want to simply know which pixels are on the line
  - Boolean
- Vertexes hold attributes
  - Ex: Color
- We want these to vary smoothly along the line
  - Linear interpolation



Adapted from Steve Marschner, Cornell University

Linear interpolation

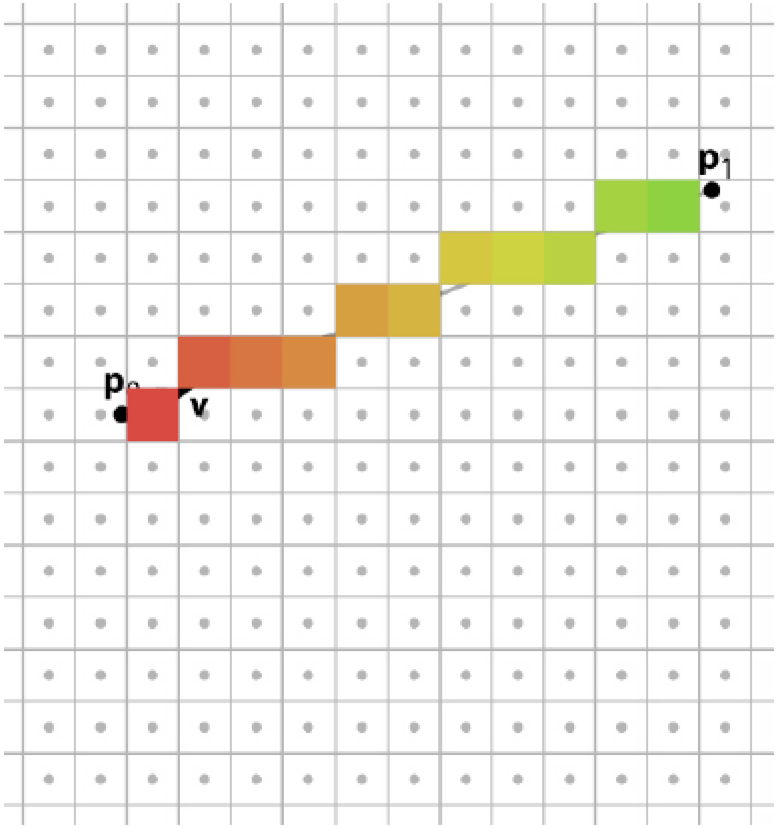
- Pixels are not exactly on the line
- Must project pixels on the line for the correct percentage
- We can use DDA!



Adapted from Steve Marschner, Cornell University

Linear interpolation

- Pixels are not exactly on the line
- Must project pixels on the line for the correct percentage
- We can use DDA!

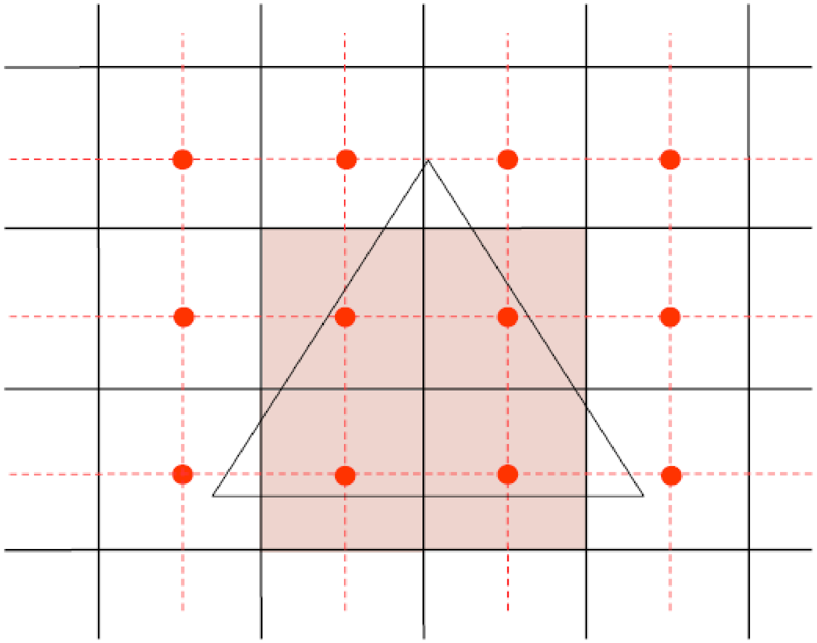


Adapted from Steve Marschner, Cornell University

What about triangles?

Rasterizing triangles

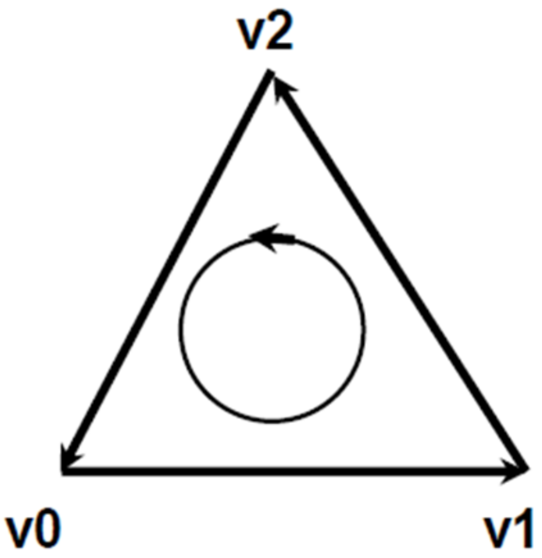
- Pixel belongs to the triangle if its center is inside the triangle
- Need two things:
  - – Which pixels belong to the triangle?
  - – How do we interpolate values from 3 vertexes?



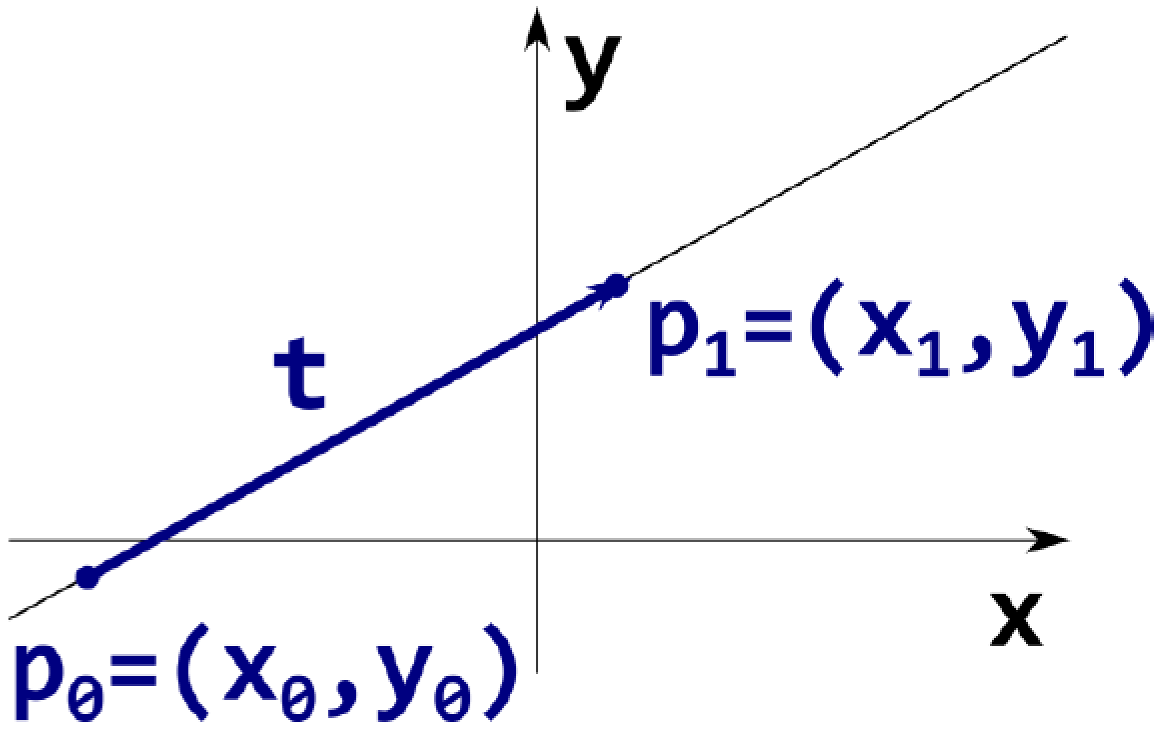
Adapted from Ron Fedkiw, Stanford University

Using directed lines

- Point is inside the triangle if it is <u>on the left of three directed lines</u>
  - They could be on the right too…
- How do we build a simple test for this?



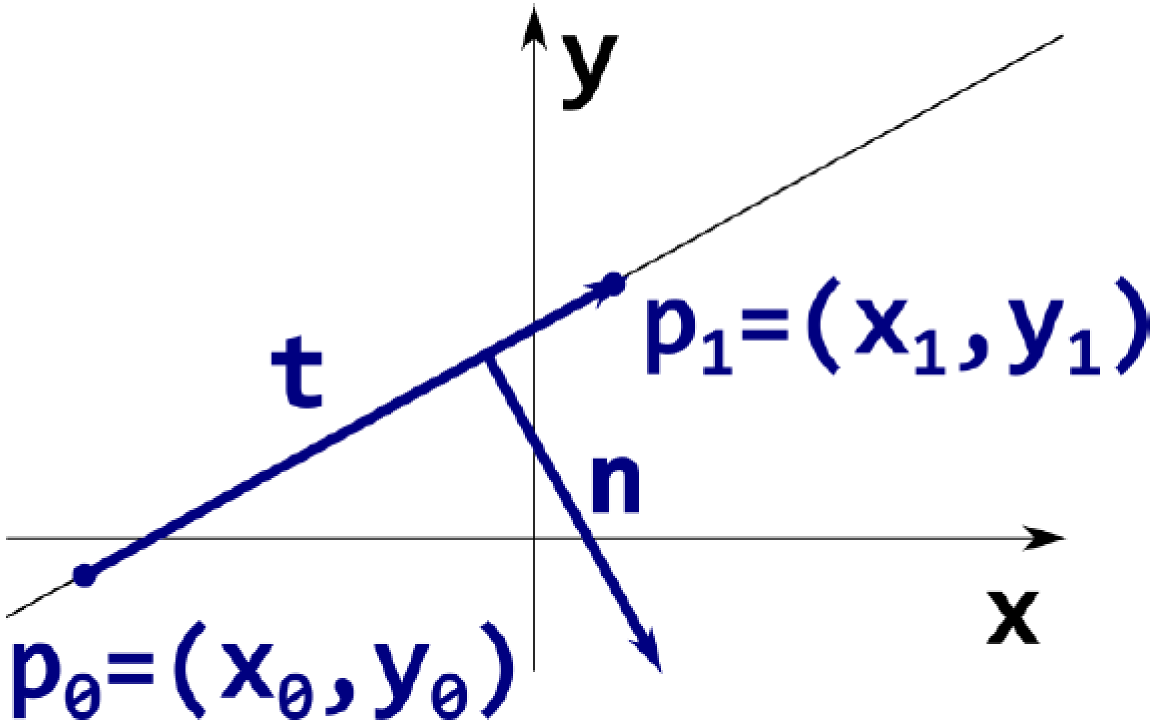
Start by defining a directed line



$$\mathbf{t} = \mathbf{p_1} - \mathbf{p_0} = (x_1 - x_0, y_1 - y_0)$$

Adapted from Ron Fedkiw, Stanford University

Easy to obtain its normal

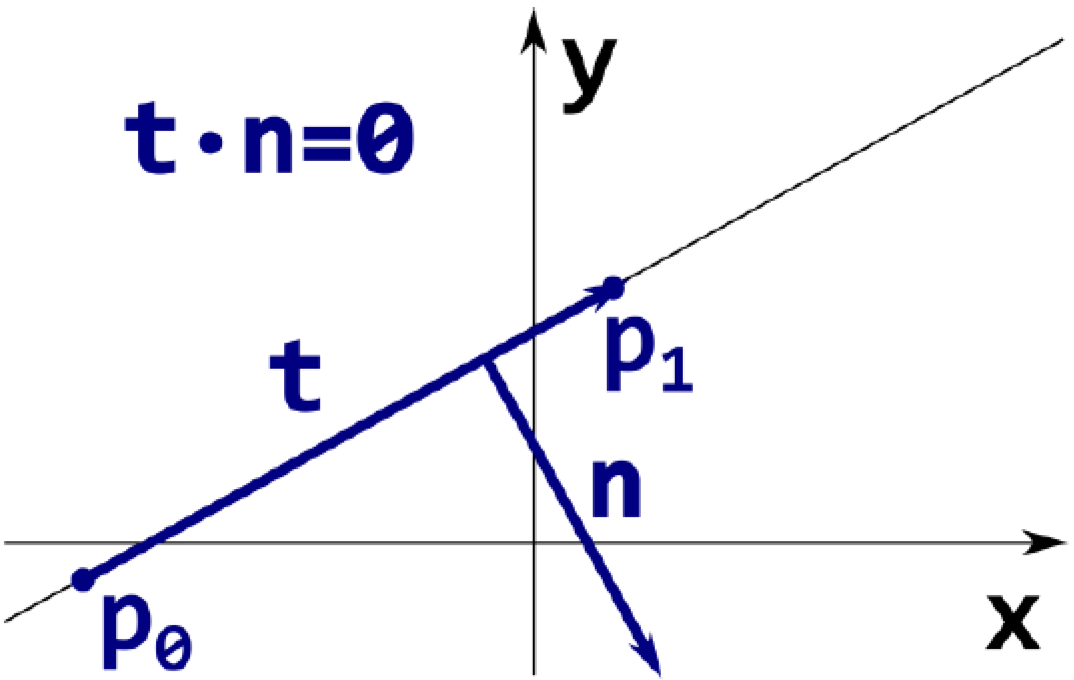


$$\mathbf{t} = \mathbf{p_1} - \mathbf{p_0} = (x_1 - x_0, y_1 - y_0)$$

$$\mathbf{n} = -\mathrm{Perp}(\mathbf{t}) = (y_1 - y_0, x_0 - x_1)$$

Adapted from Ron Fedkiw, Stanford University

Dot product gives us a simple test

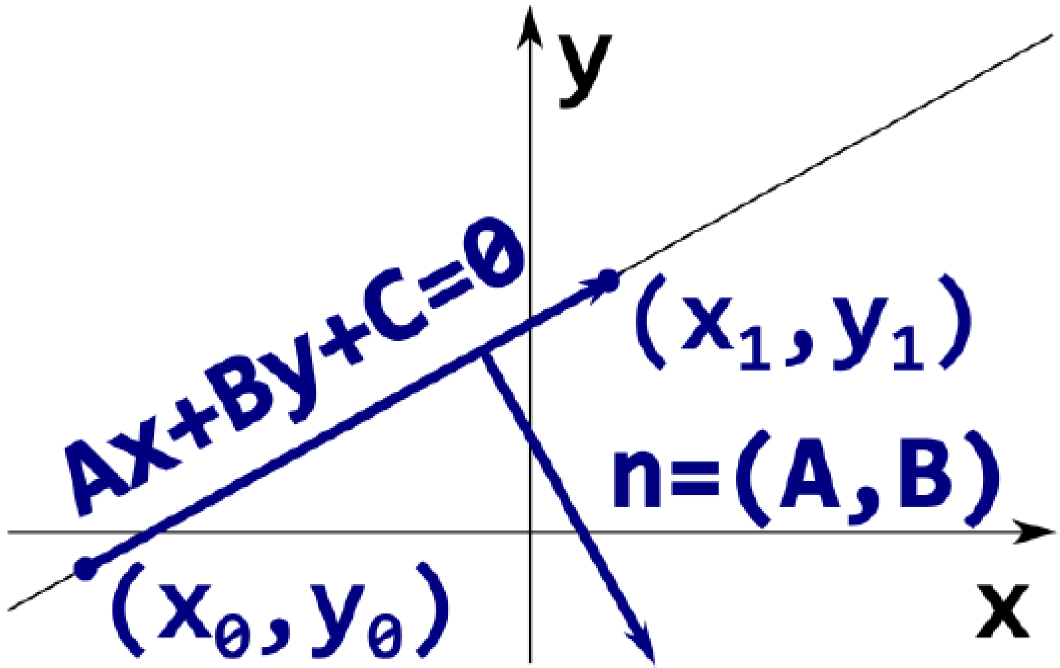


$$(\mathbf{p} - \mathbf{p}_0) \cdot \mathbf{n} = 0$$

This equation must be true for all point p on the line

Adapted from Ron Fedkiw, Stanford University

Using our coordinate system



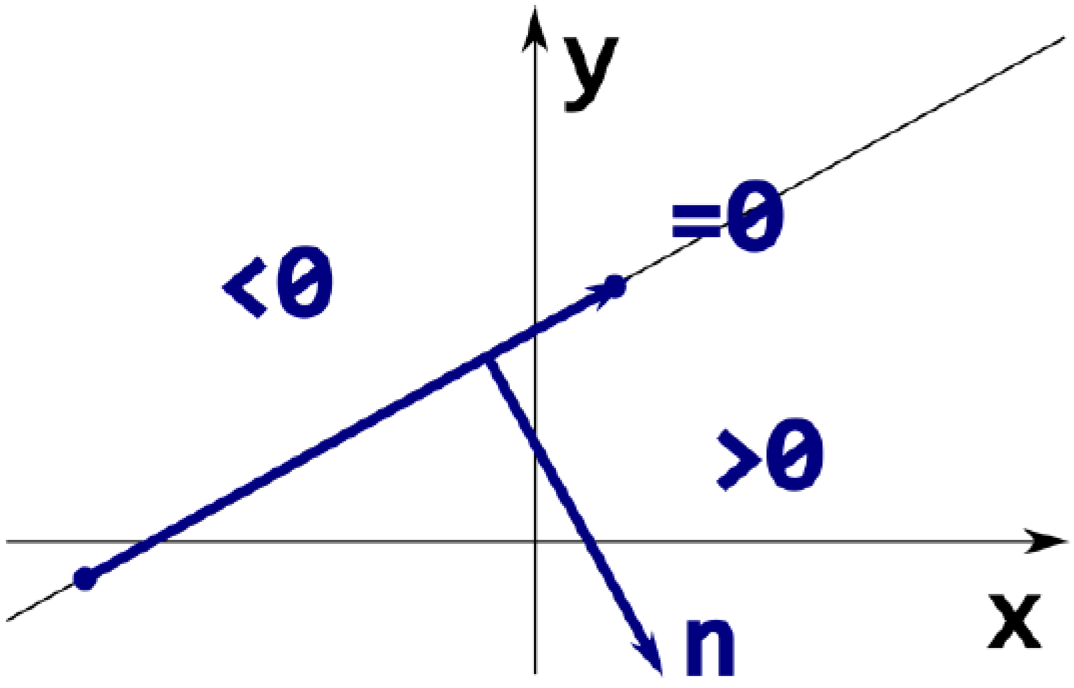
$$A = y_1 - y_0$$

$$B = x_0 - x_1$$

$$C = y_0 x_1 - x_0 y_1$$

Adapted from Ron Fedkiw, Stanford University

Line divides the plane in two

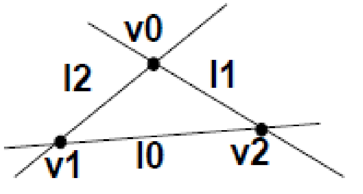


Normal n points to the right of the line
Inside (negative values) to the left

Adapted from Ron Fedkiw, Stanford University

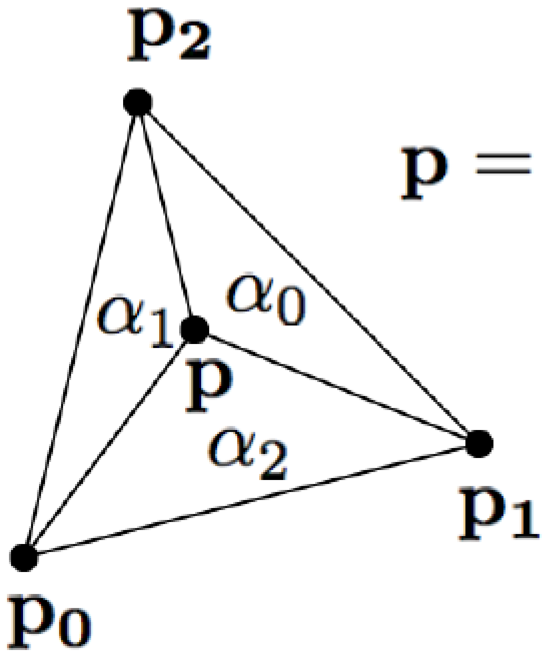
Point inside triangle test

```
makeline( vert& v0, vert& v1, line& l )
{
  l.a = v1.y - v0.y;
  l.b = v0.x - v1.x;
  l.c = -(l.a * v0.x + l.b * v0.y);
}
rasterize( vert v[3] )
{
  line l0, l1, l2;
  makeline(v[0],v[1],l2);
  makeline(v[1],v[2],l0);
  makeline(v[2],v[0],l1);
  for( y=0; y<YRES; y++ ) {
    for( x=0; x<XRES; x++ ) {
      e0 = l0.a * x + l0.b * y + l0.c;
      e1 = l1.a * x + l1.b * y + l1.c;
      e2 = l2.a * x + l2.b * y + l2.c;
      if( e0<=0 && e1<=0 && e2<=0 )
        fragment(x,y);
    }
  }
}
```



Barycentric interpolation

Triangle



$$\mathbf{p} = \alpha_0\,\mathbf{p_0} + \alpha_1\,\mathbf{p_1} + \alpha_2\,\mathbf{p_2}$$

$$\alpha_0 + \alpha_1 + \alpha_2 = 1$$

Advanced topic!
Check the ref. book

Adapted from Ron Fedkiw, Stanford University

Summary

- Rasterization
  - Which pixels belong to the primitive
  - How do I interpolate vertex atributes?
- Lines
  - Consider them rectangles
  - Linear interpolation
- Triangles
  - Use three directed lines
  - Barycentric interpolation