Rasterization and Aliasing

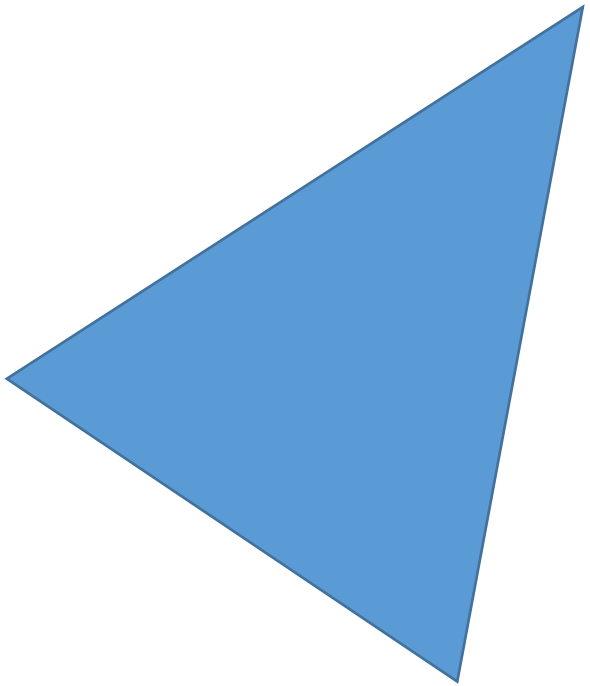
Lecture 27

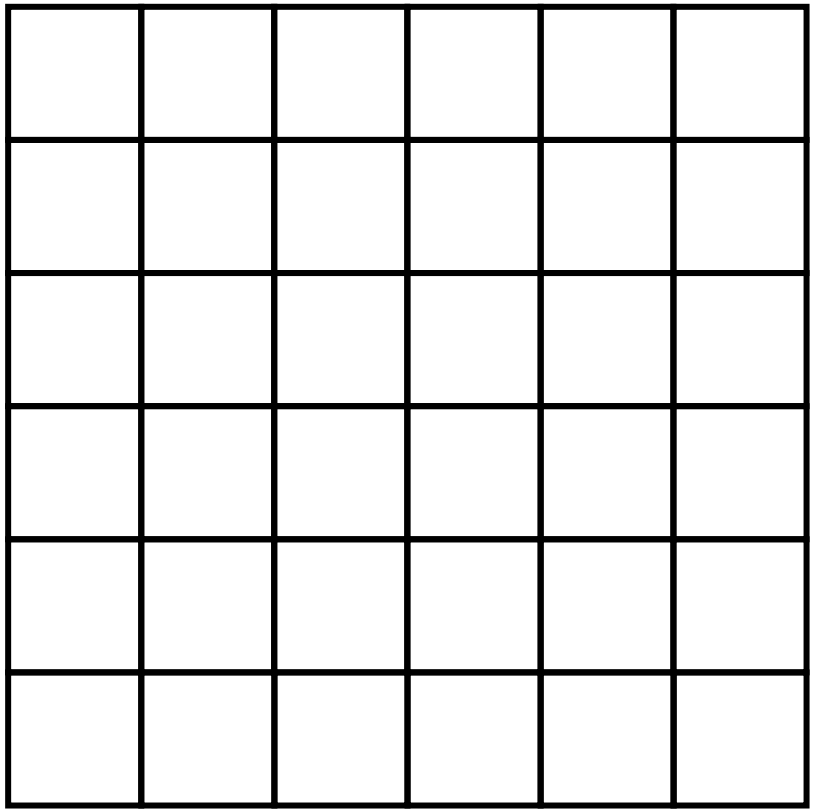
Rasterization

Figure out which pixels a primitive “covers”

Turns primitives into pixels

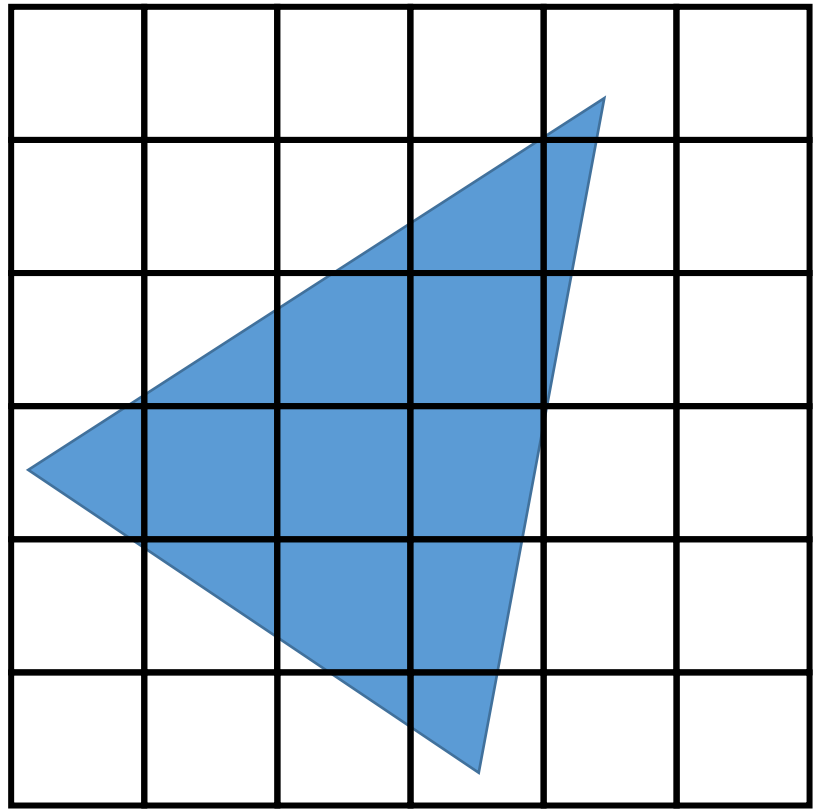
Rasterization





Rasterization

Convert primitives to pixels



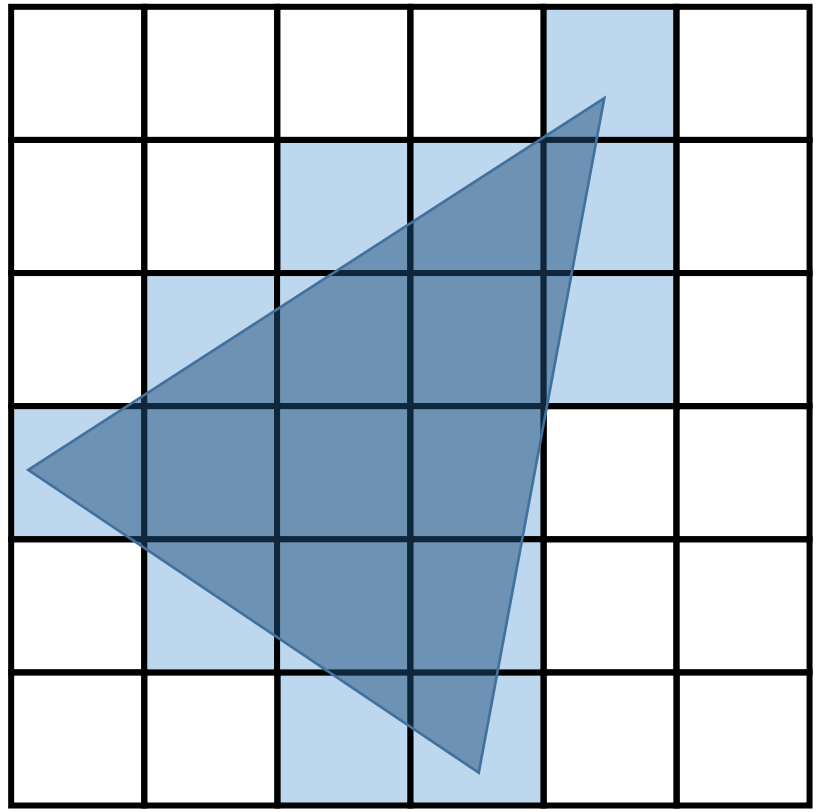
Rasterization

Convert primitives to pixels

For now consider coverage

Yes/No decision

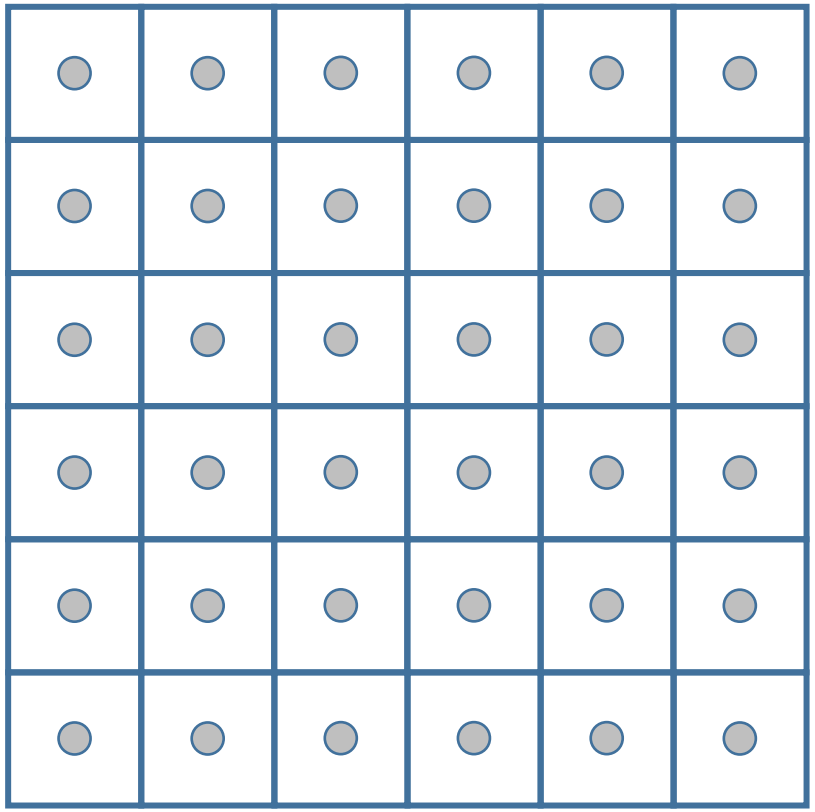
Partial fill later



A pixel is a little square?

Think of pixels as points

The center of the square



Repeat after me…

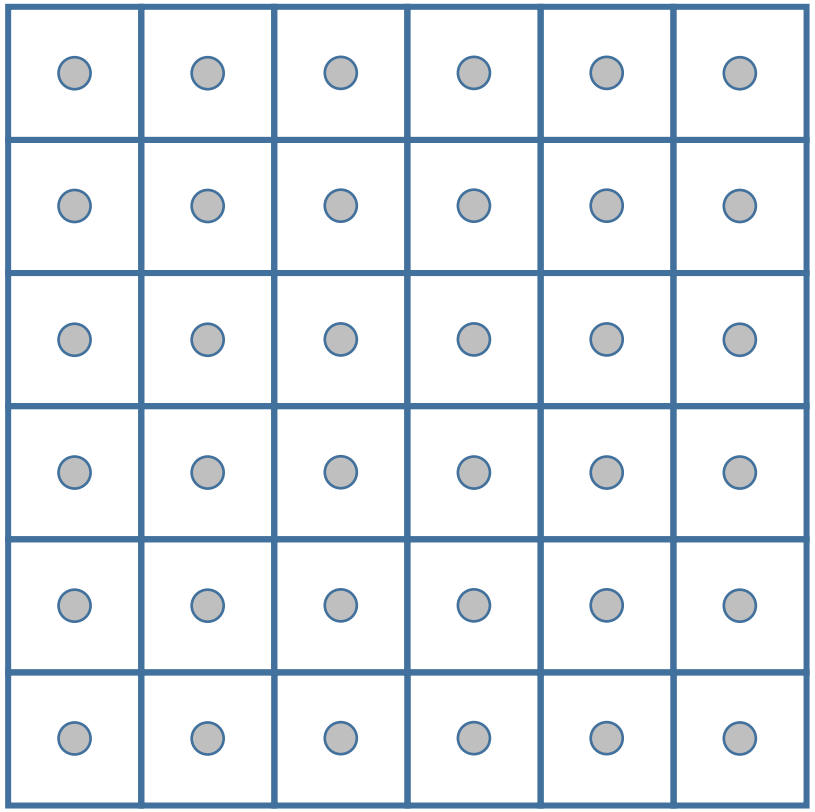
A pixel is not a little square

Pixels are Point Samples

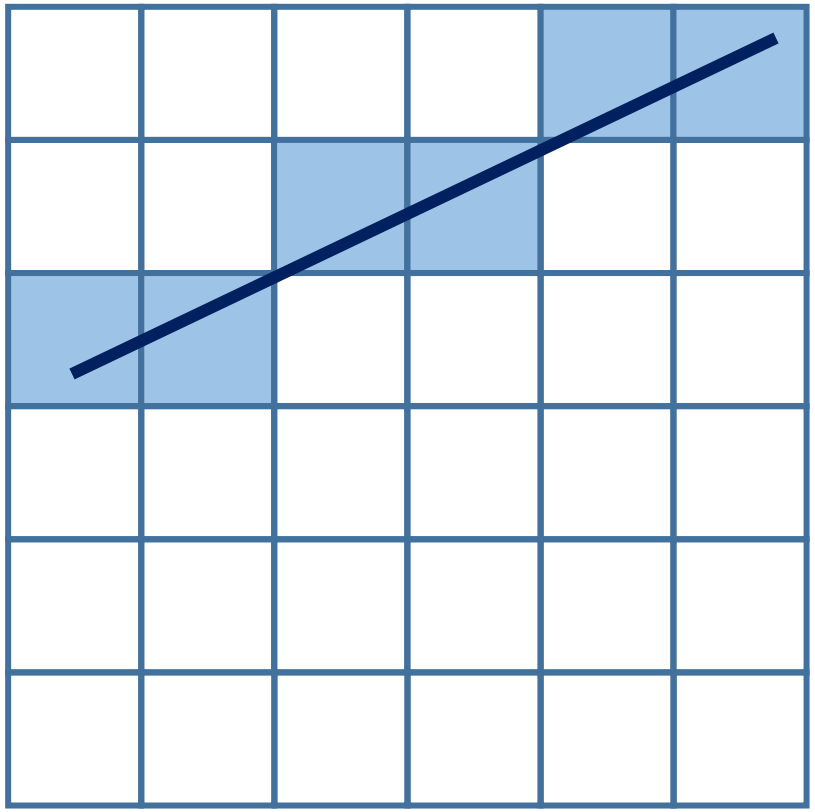
Specific positions

The center of the square (convention)

Makes everything more consistent



Lines

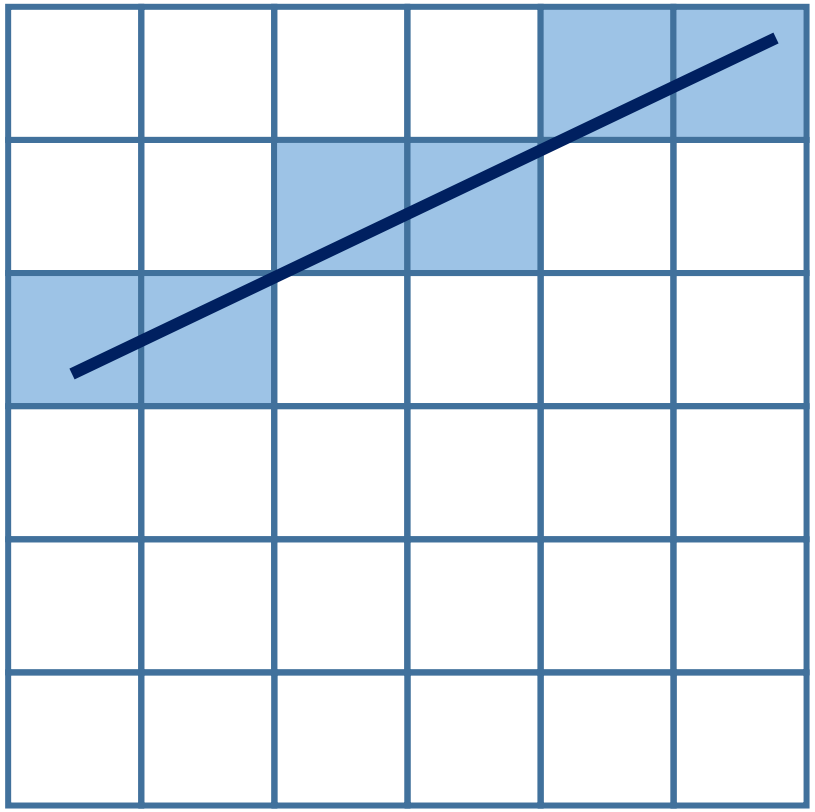


Drawing Lines

No gaps

1 pixel per column

(for slopes -1 to 1)



Brezenham / Midpoint Algorithm

Each octant is special

(this is for 1st octant)

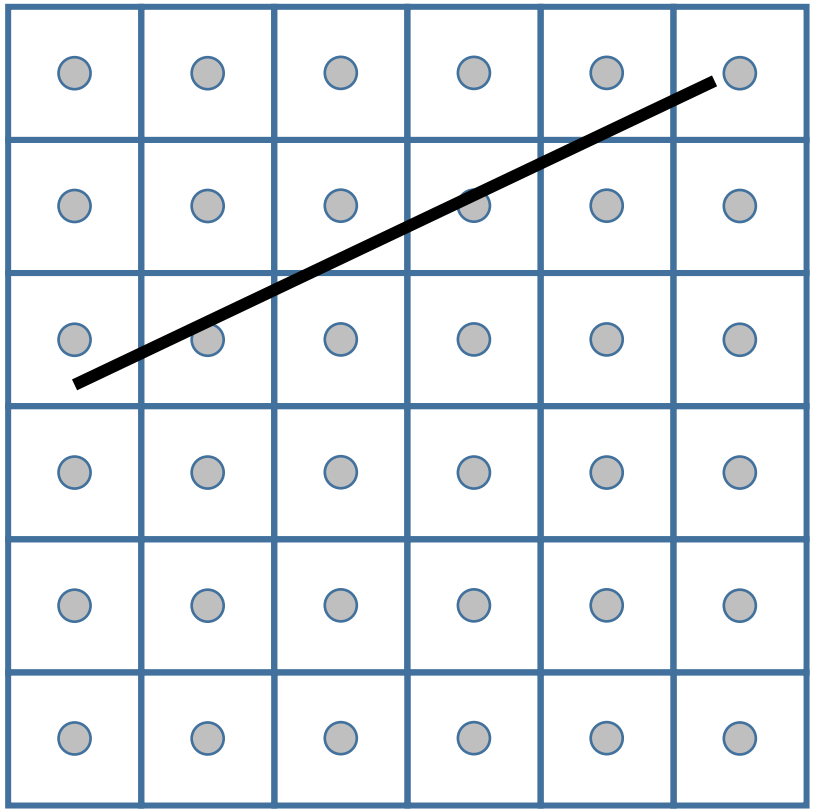
For each column:

pick closest point

As you move right to left…

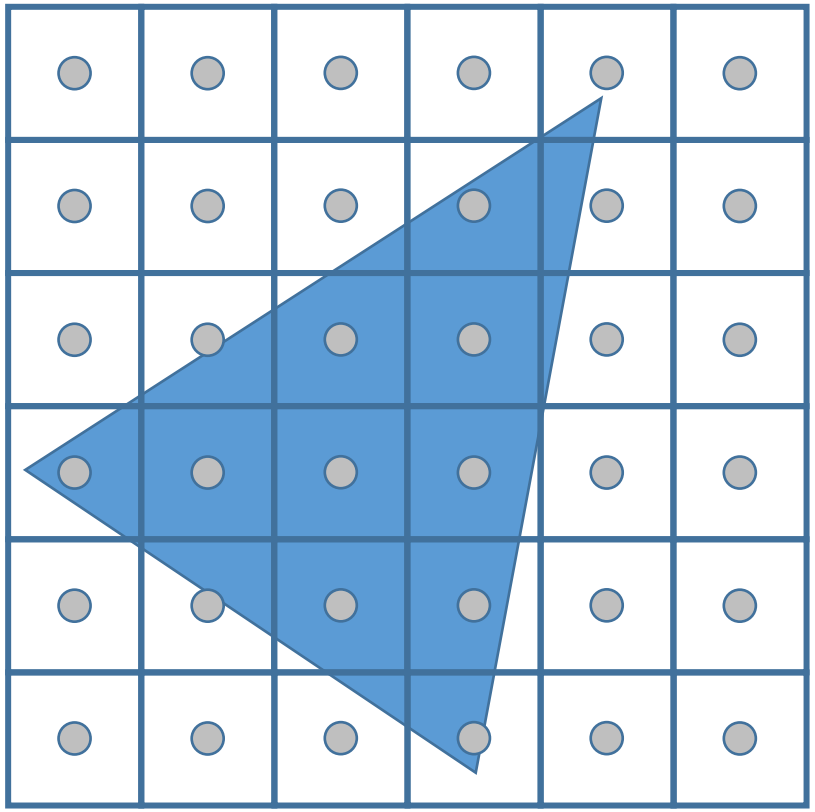
stay the same

move up



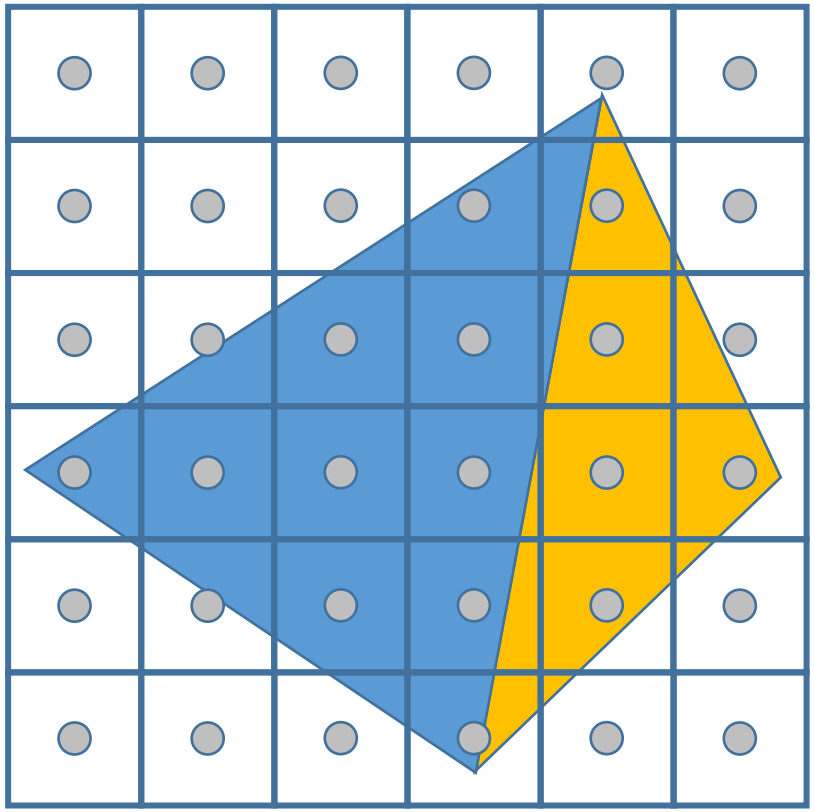
Triangles

Does the triangle cover the sample point?



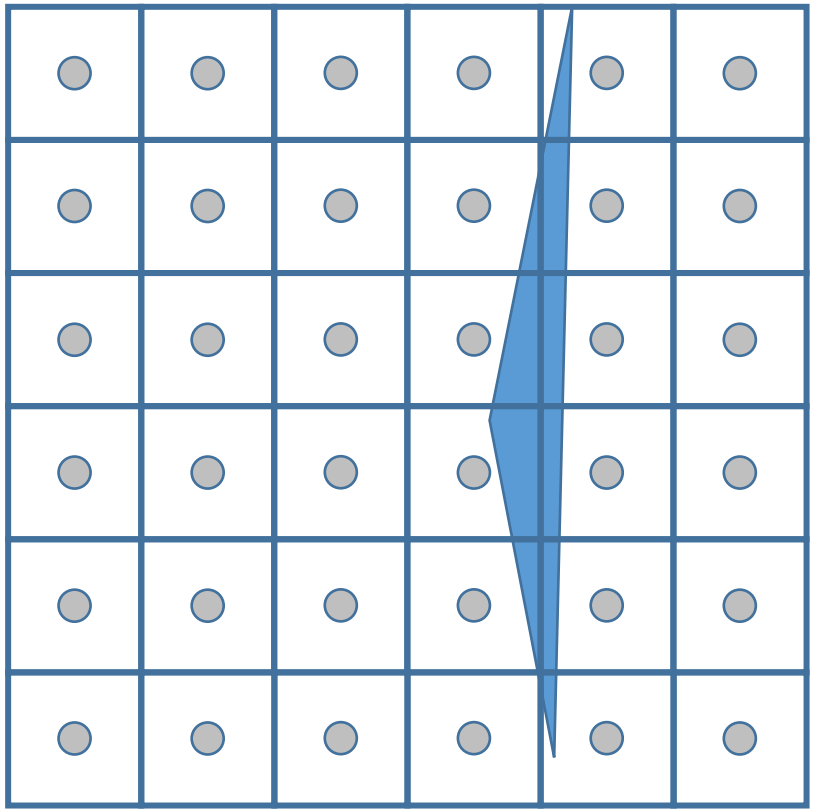
Point Coverage

Handles edges consistently



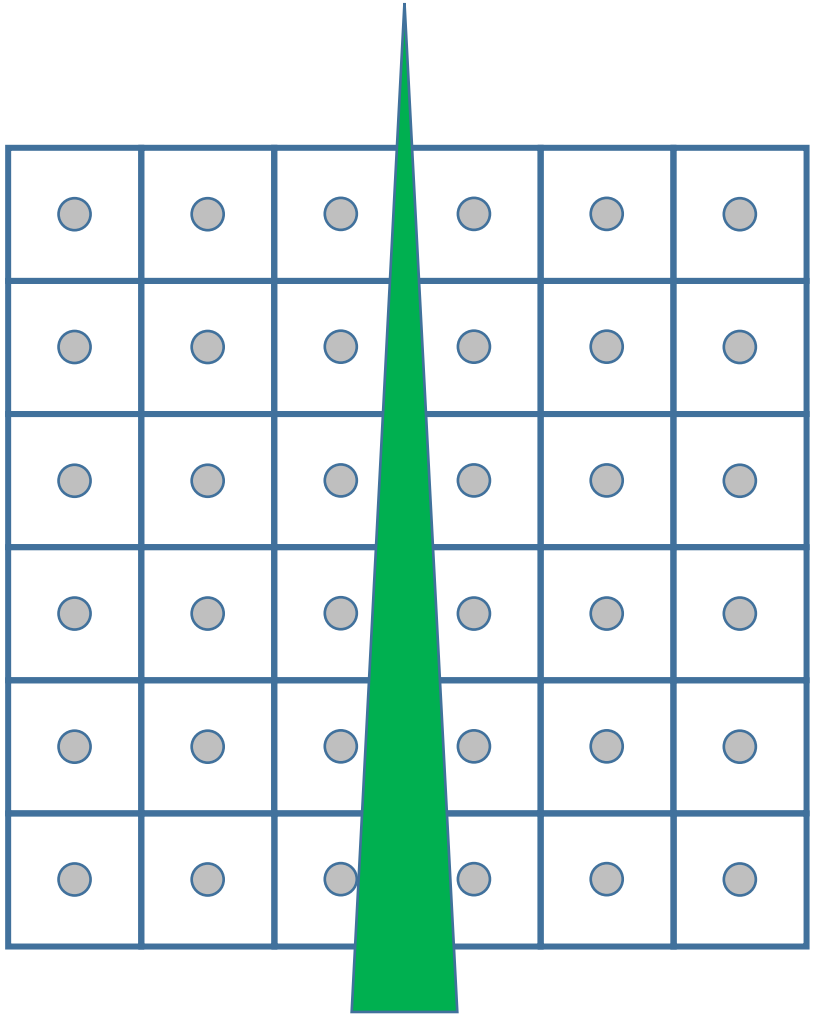
Triangles

A triangle could get lost between pixels



Triangles

A triangle could get lost between pixels



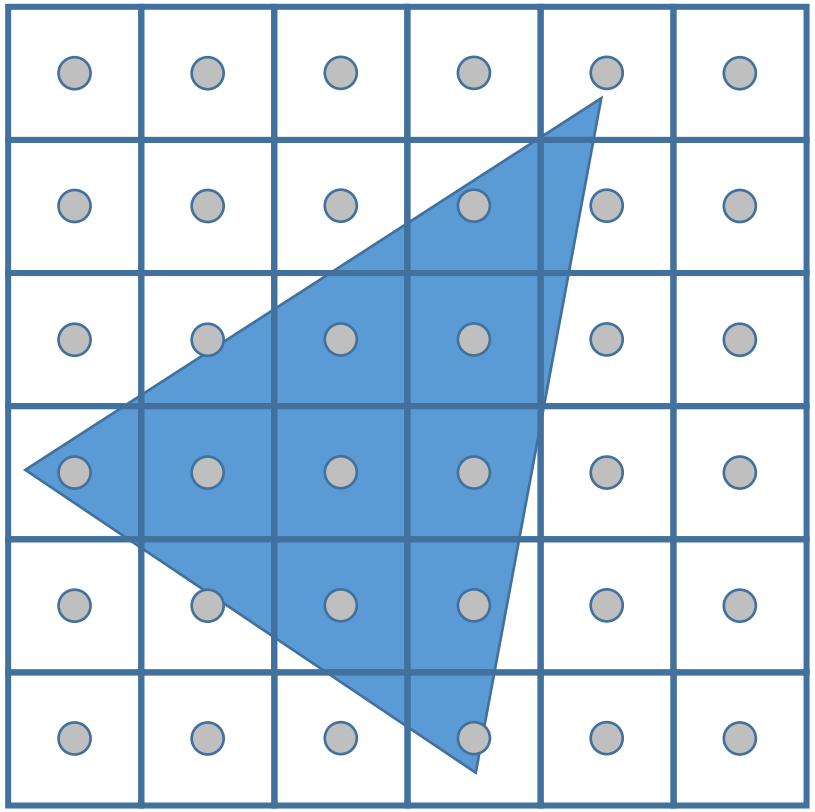
Scanline Algorithm

For each row

find left and right

fill

Use line drawing to get edges



Hardware triangle drawing

Barycentric Coordinates

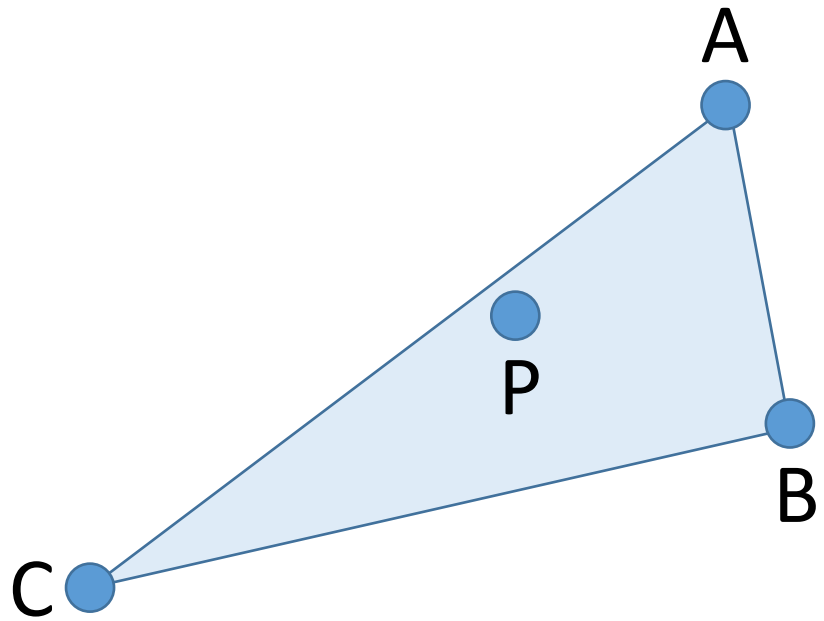
Any point in the plane is a convex combination of the vertices of the triangle

$P=\alpha A+\beta B+\gamma C$

$\alpha+\beta+\gamma=1$

Inside triangle

$0 <= \alpha, \beta, \gamma <= 1$



Barycentric Coords are Useful!

Every point in plane has a coordinate

$(\alpha\ \beta\ \gamma)$ such that: $\alpha + \beta + \gamma = 1$

Easy test inside the triangle

$0 <= \alpha, \beta, \gamma <= 1$

Interpolate values across triangles

$$\mathbf{x}_p = \alpha\, \mathbf{x}_1 + \beta\, \mathbf{x}_2 + \gamma\, \mathbf{x}_3$$

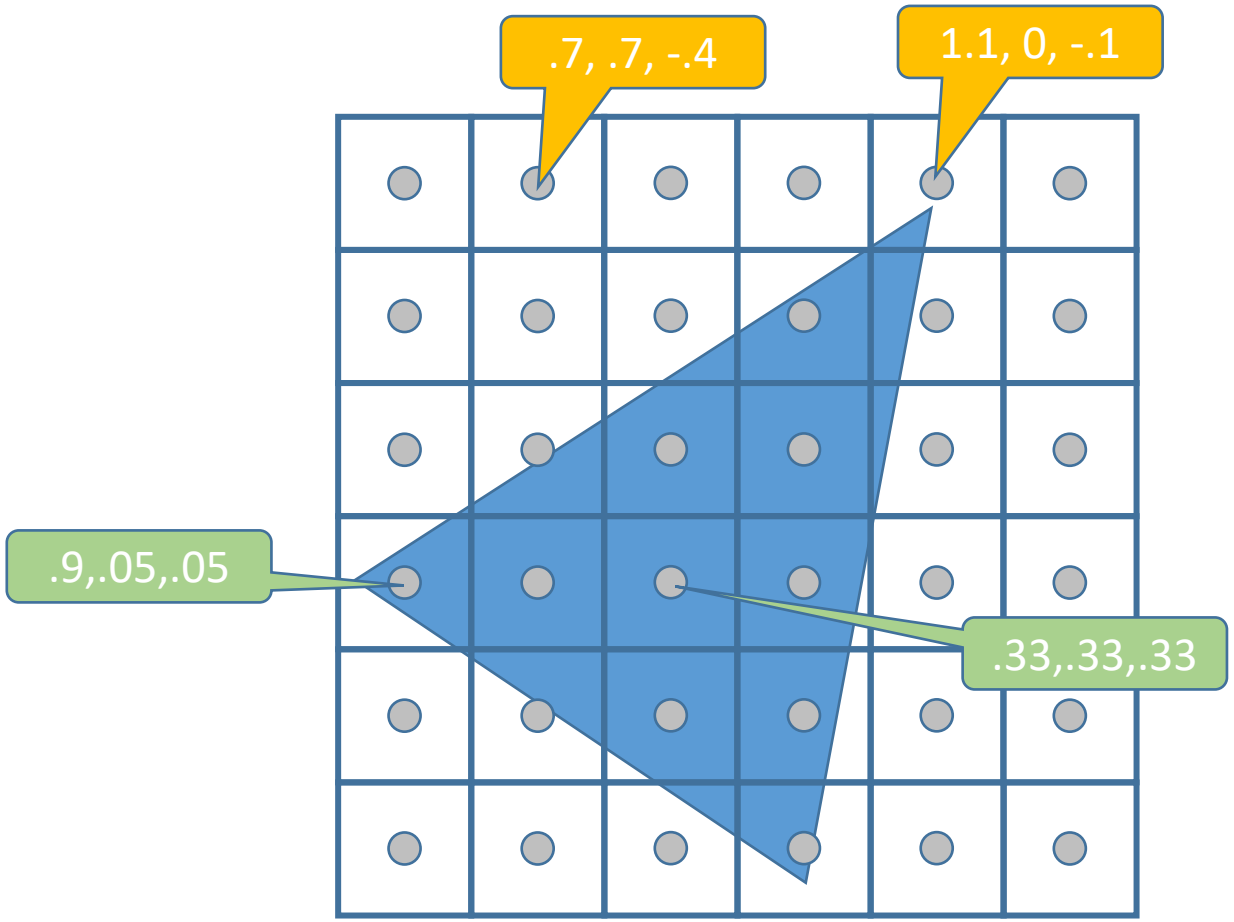
$$\mathbf{c}_p = \alpha\, \mathbf{c}_1 + \beta\, \mathbf{c}_2 + \gamma\, \mathbf{c}_3$$

Hardware Rasterization

For each point:

Compute barycentric coords

Decide if in or out



Wasteful?

Can do all points in parallel

We want the coordinates (for interpolation)

Does the right things for touching triangles

Each point in 1 triangle

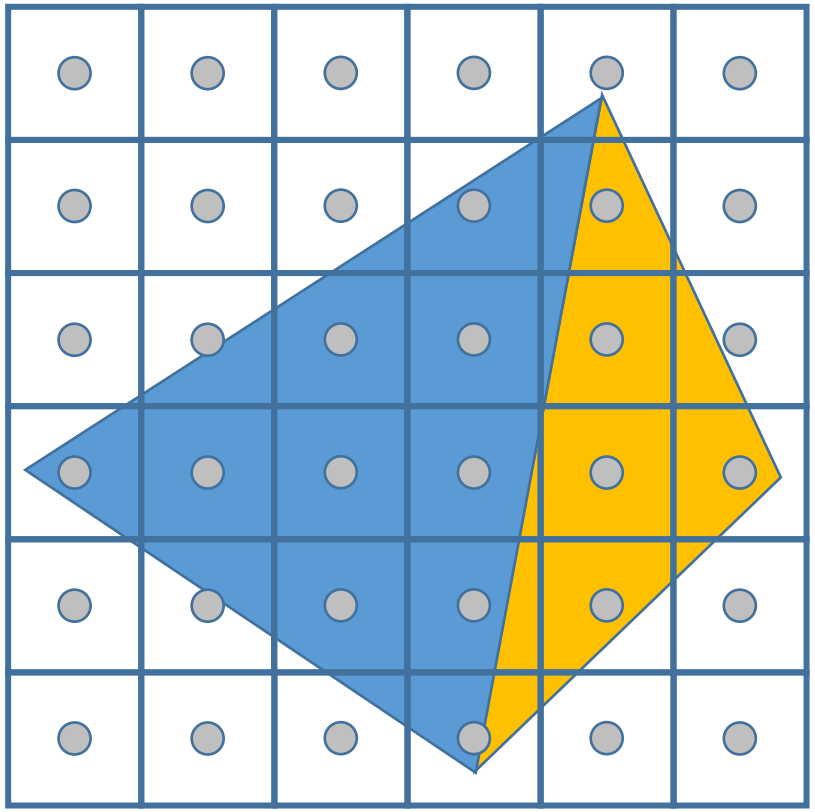
Hardware Rasterization

Each center point in one triangle

If we choose consistently for “on-the-edge” cases

Over simplified version:

$0 <= a,b,c < 1$



Aliasing and Anti-Aliasing

Living with the problems of finite representations

Aliasing: The problem

Continuous World

Infinite information!

Discrete Computer

Finite representation

Aliasing

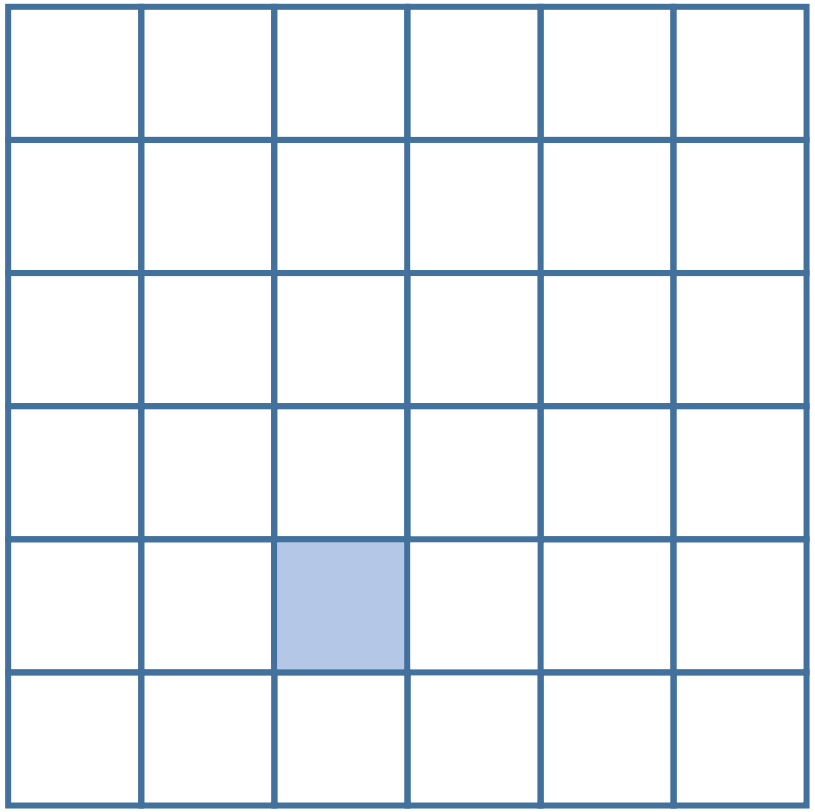
Any representation may have multiple interpretations

We don’t know which one is right

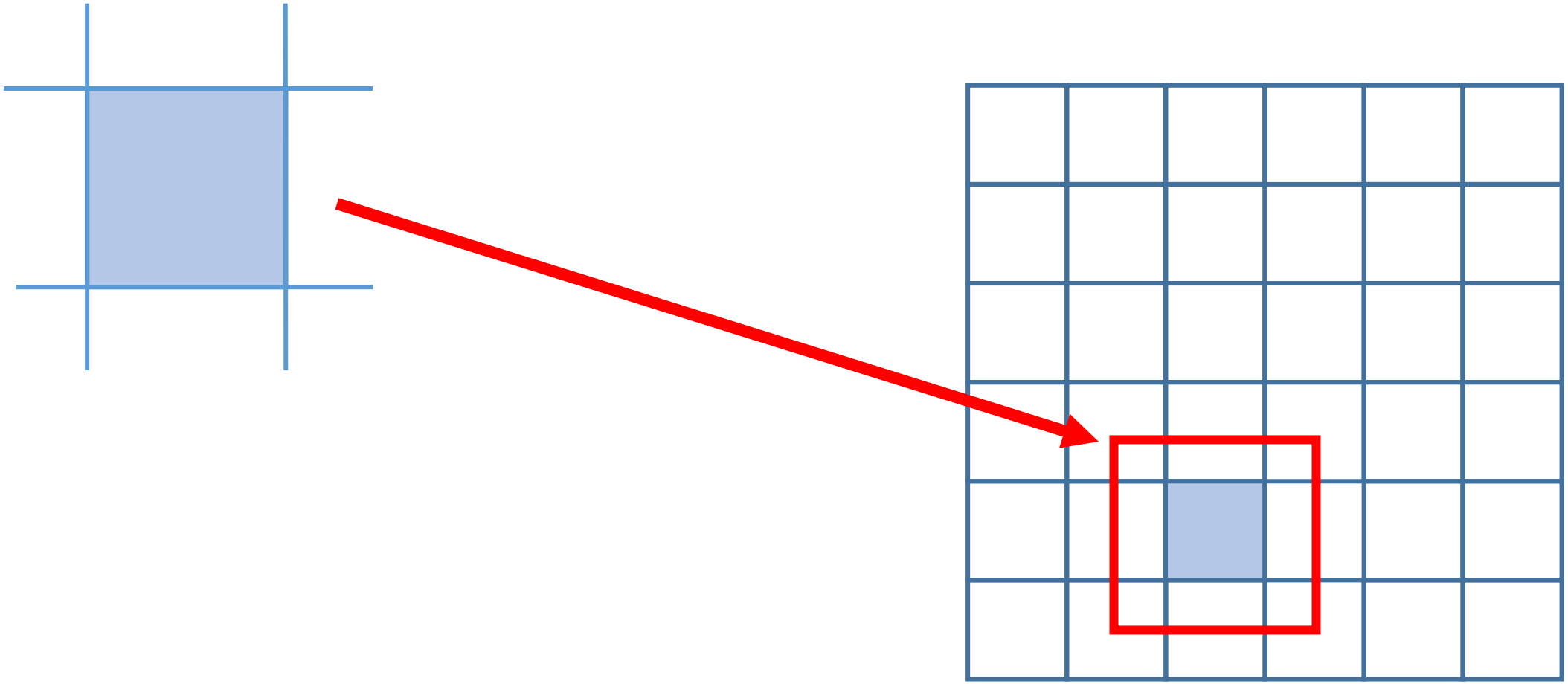
They are aliases of each oteher

Aliasing
It also happens in the “little square model”

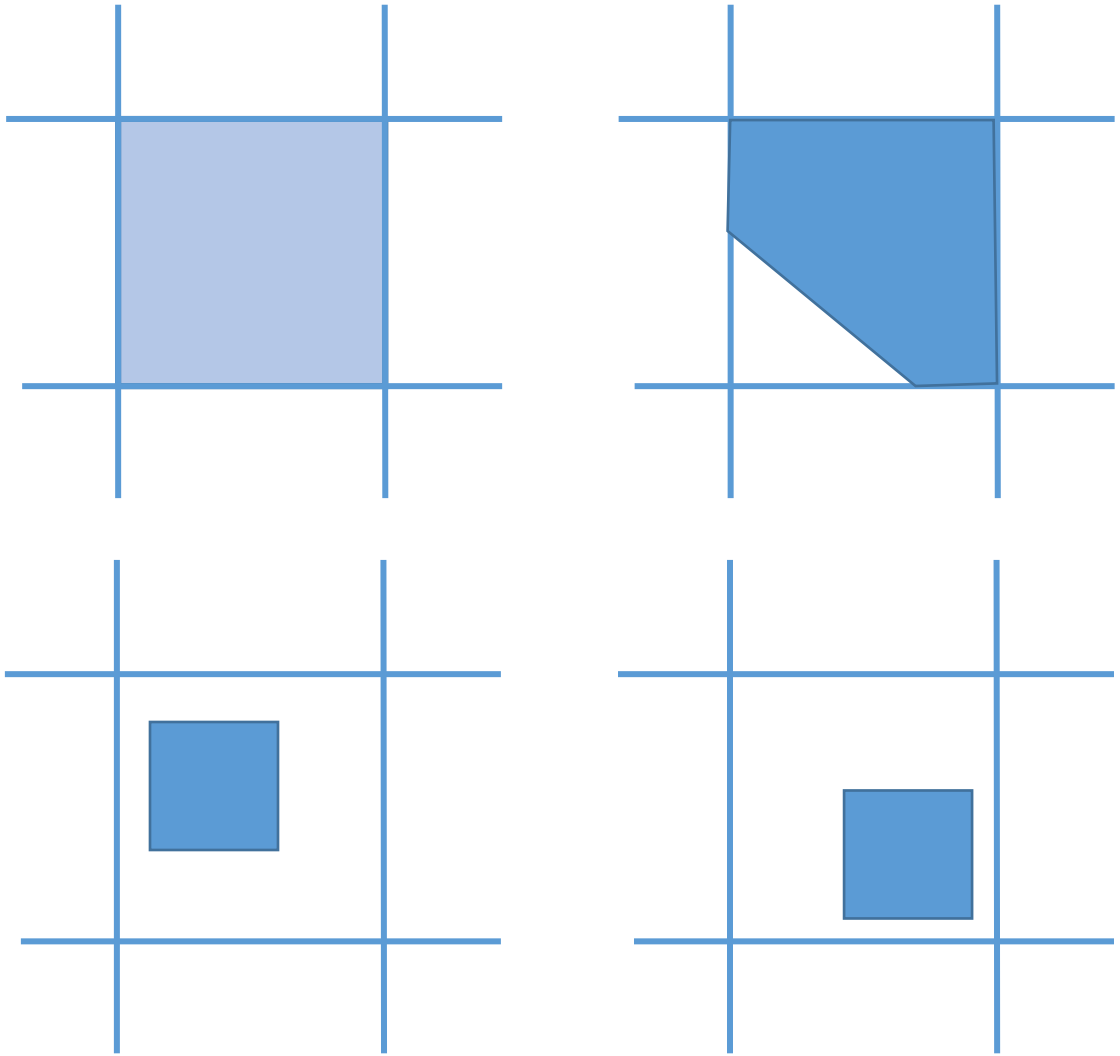
A pixel has 1 color

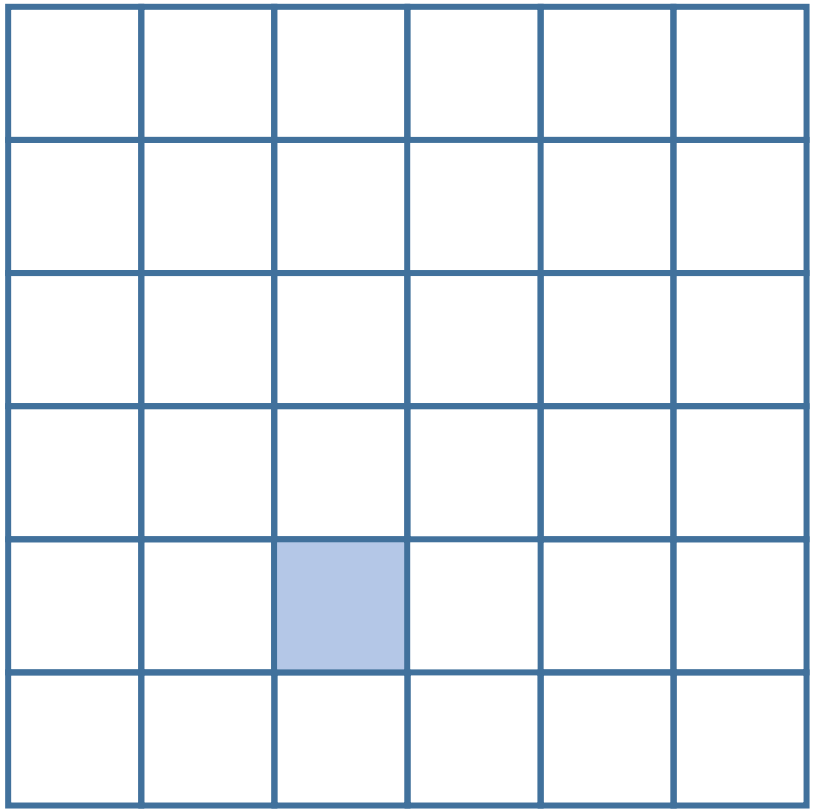


Which one is it?



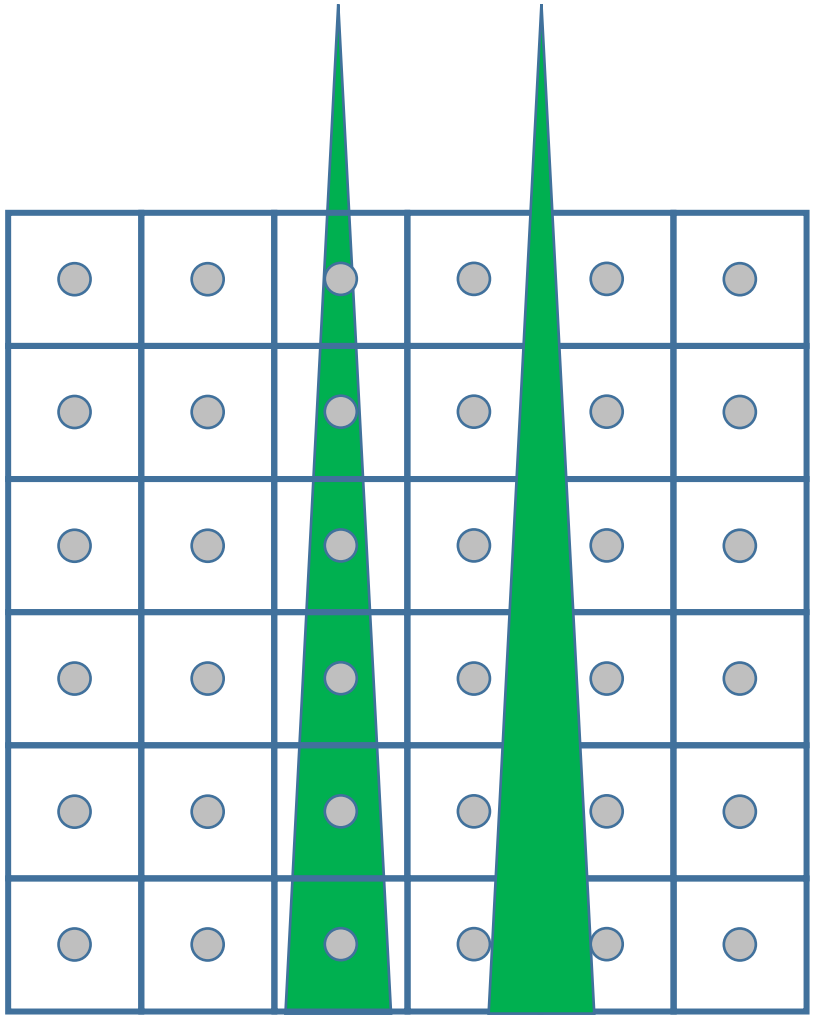
Which one is it?





No Squares

Same thing – we only know what happens at the sample!



Many possible original signals
Same discrete representation

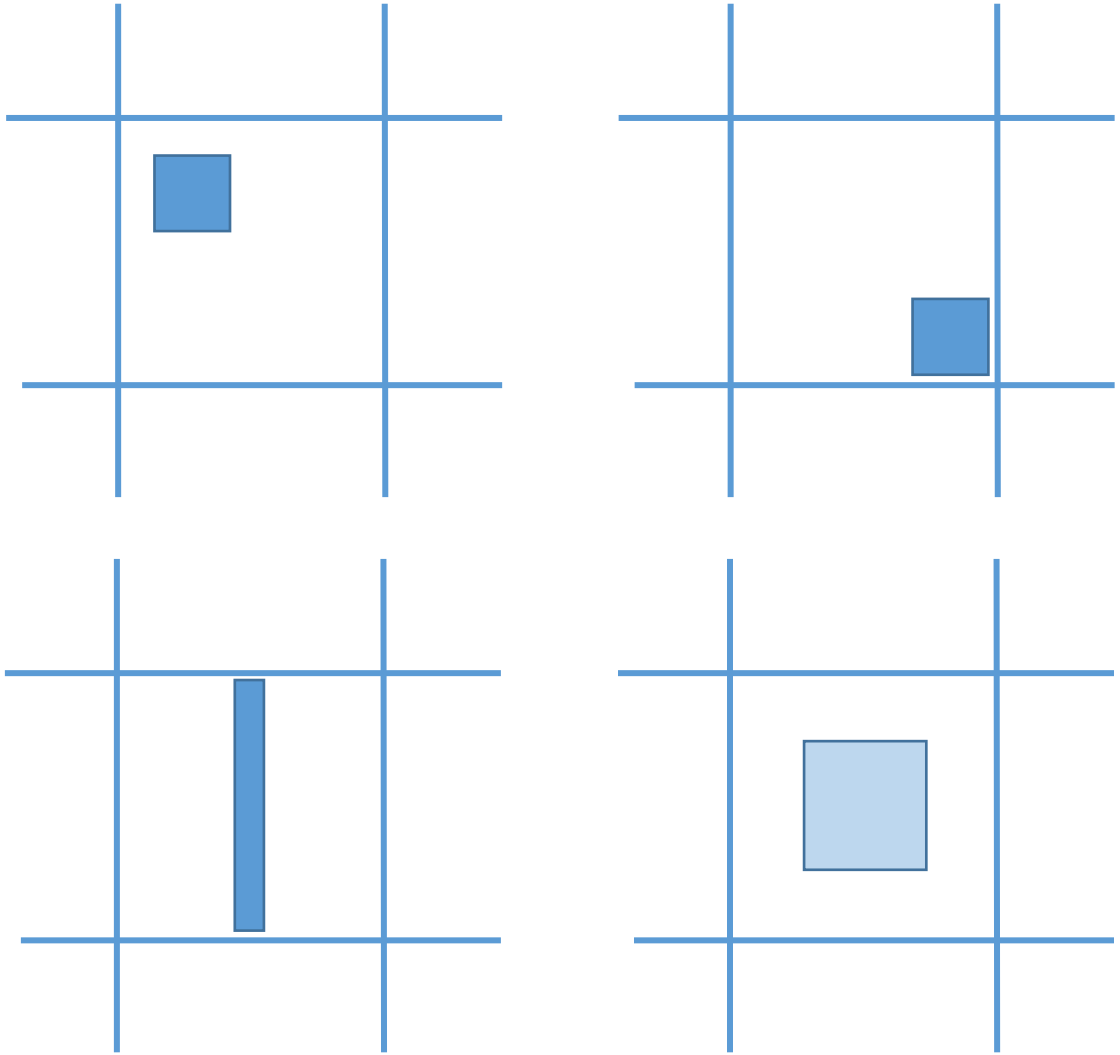
Aliasing

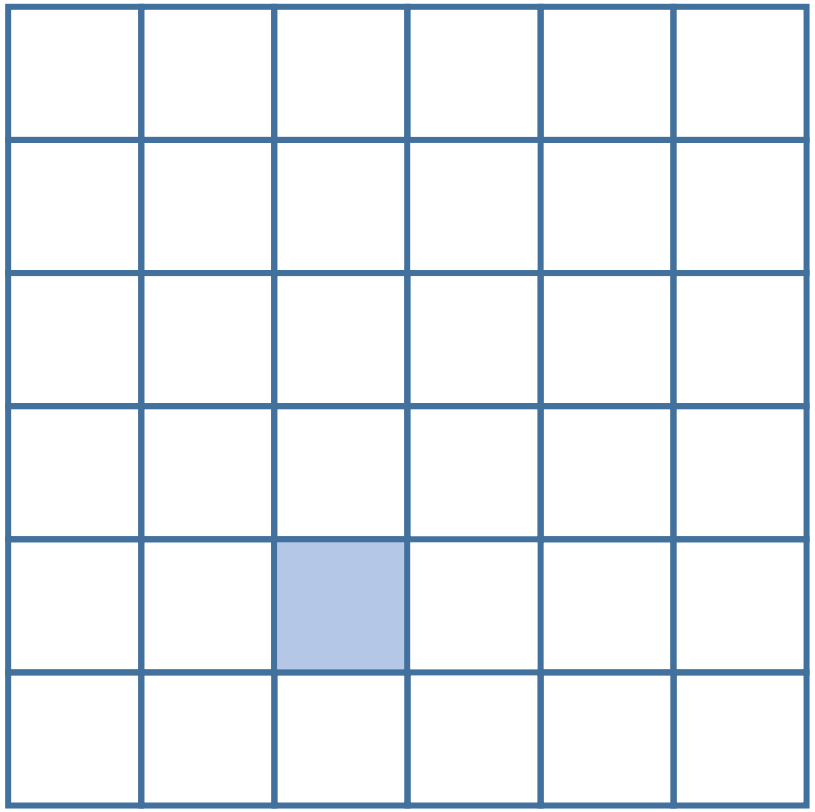
Many possible “signals”

Same discrete representation

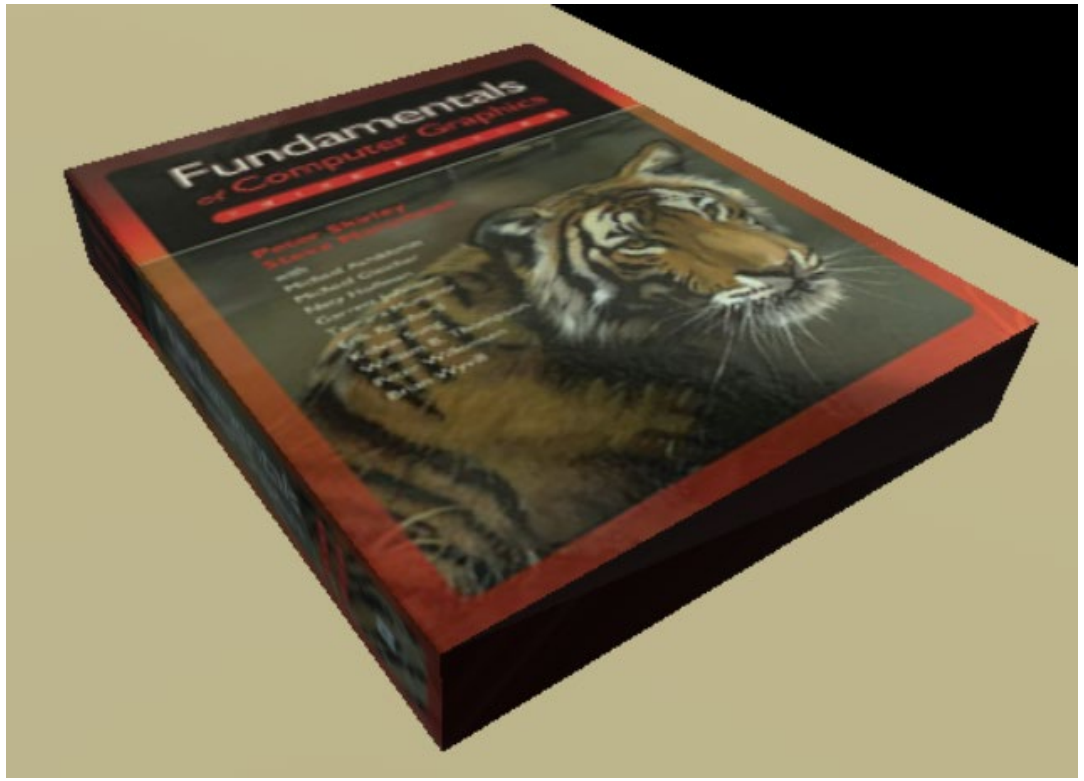
This problem comes up everywhere...

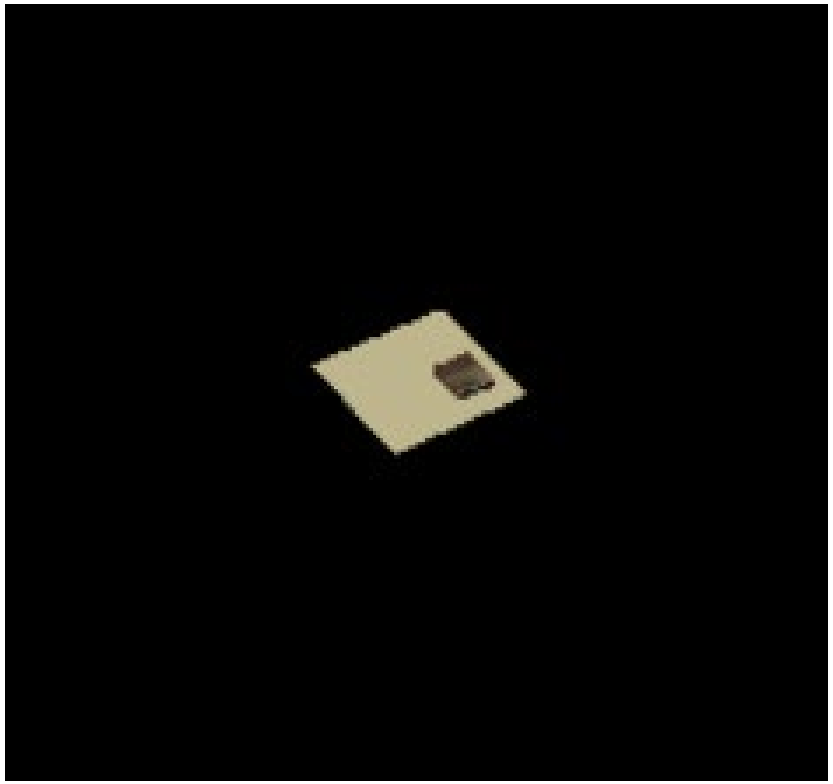
Point positions





Texture Sampling



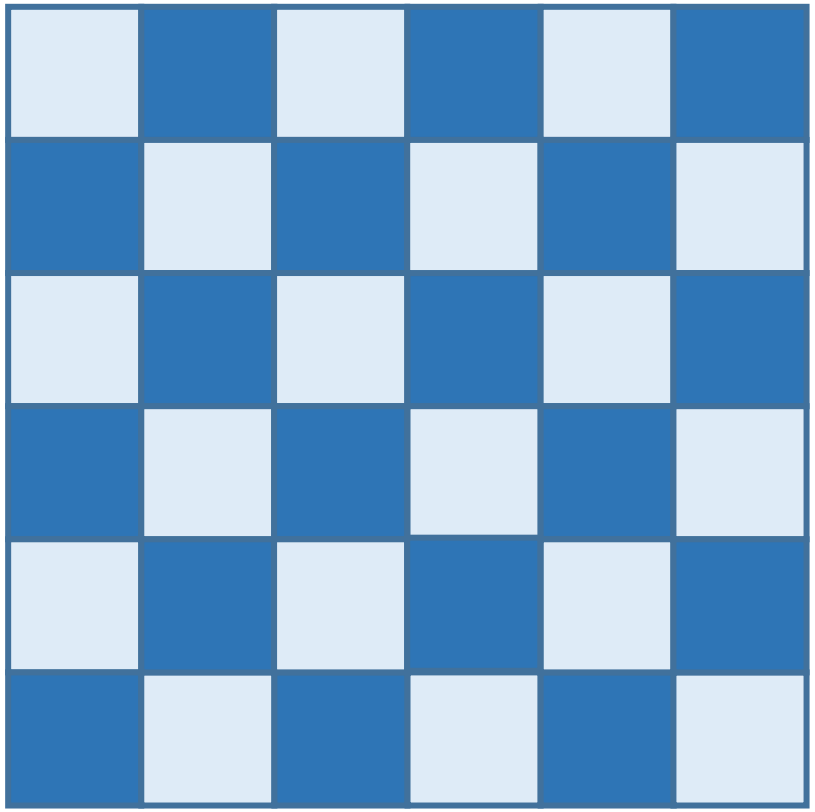


Make an image (e.g. texture) smaller…

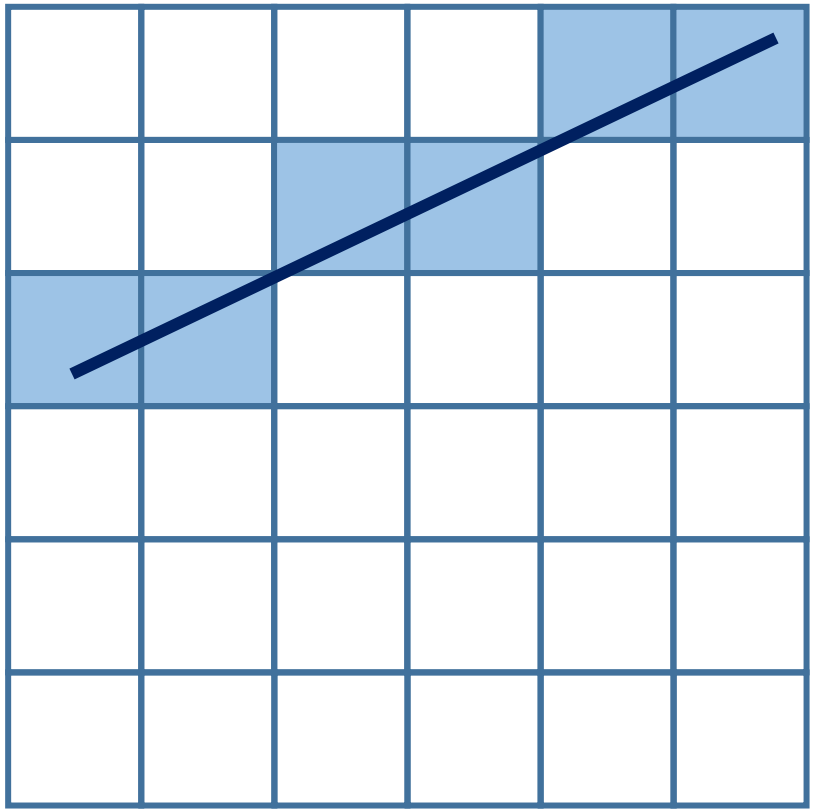
Cut by a factor of 2

Even or odd

Gives very different results

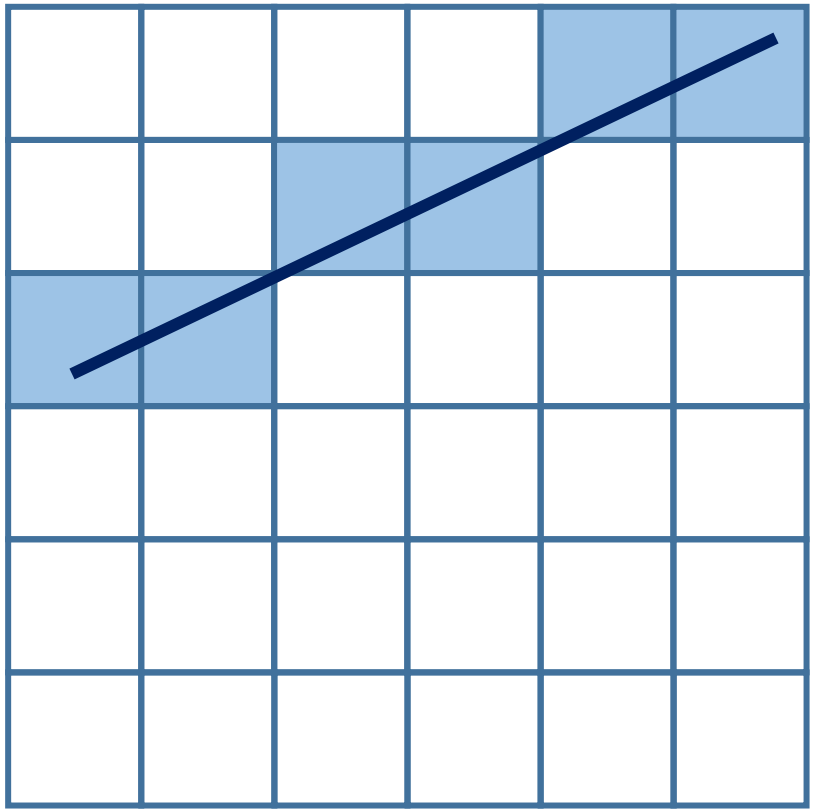


Lines



Lines

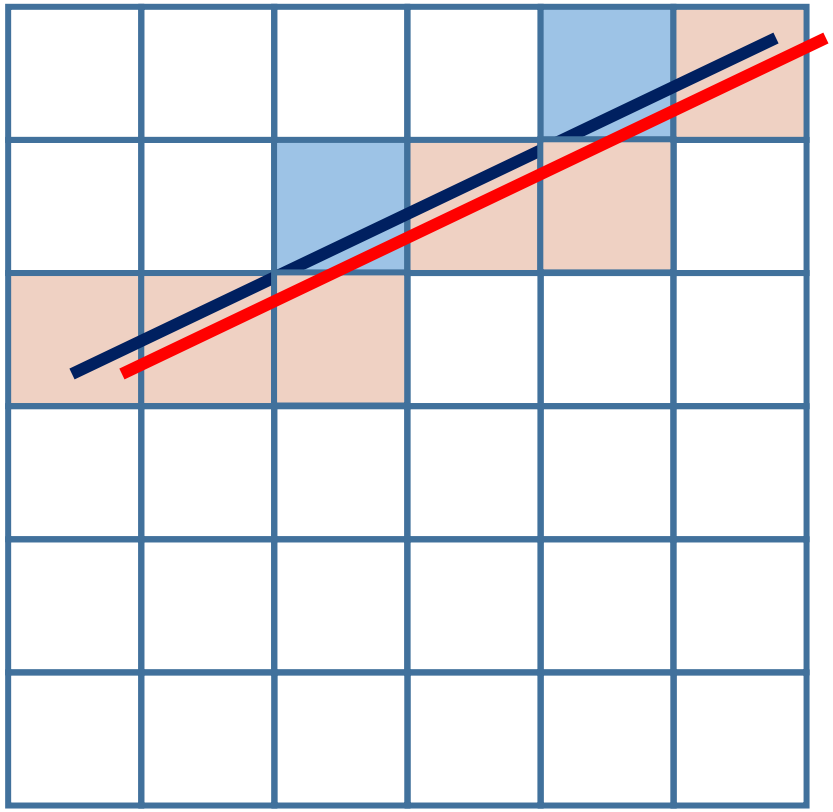
Jaggies



Lines

Crawlies

(mainly if edge)

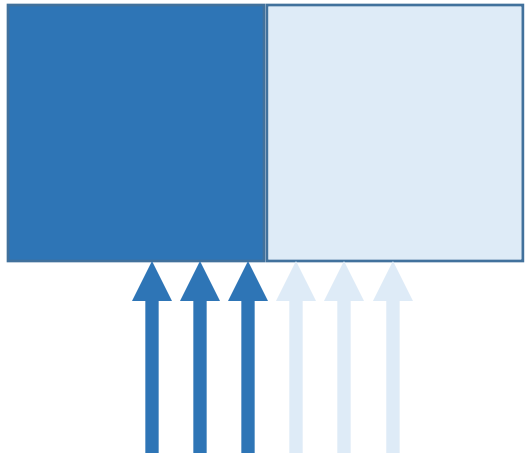


Intuition: Sharp Edges are bad

Sharp Edge:

Small change in position

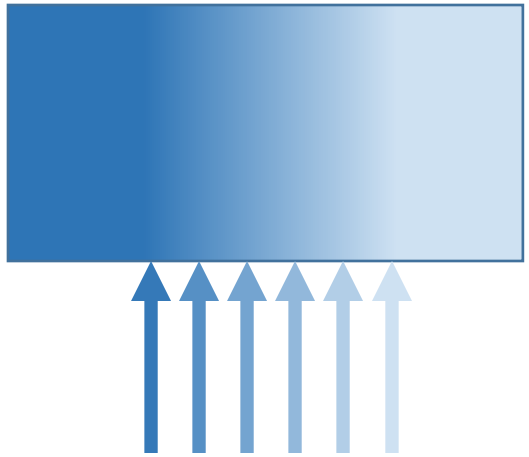
Big change in value



Smoother “Edge:”

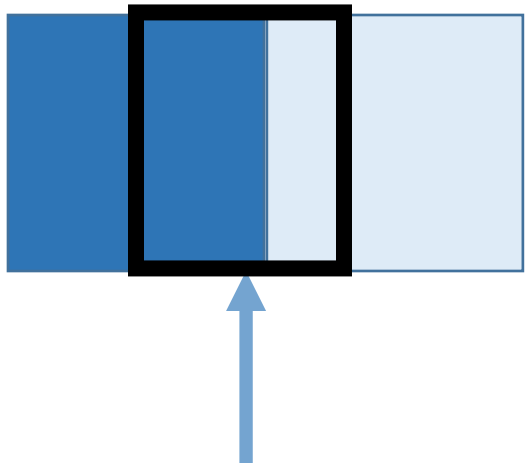
Small change in position

Doesn’t matter (that much)



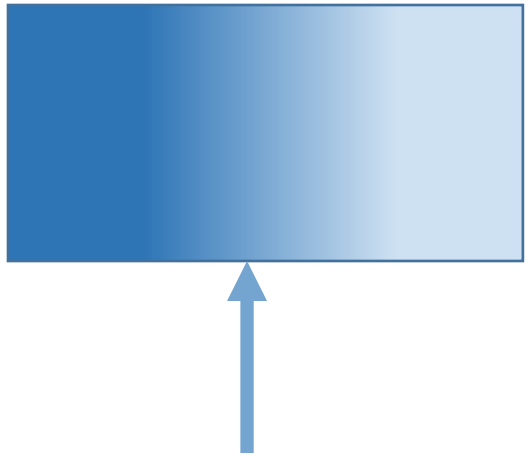
Idea: Average over the region

Average over the region

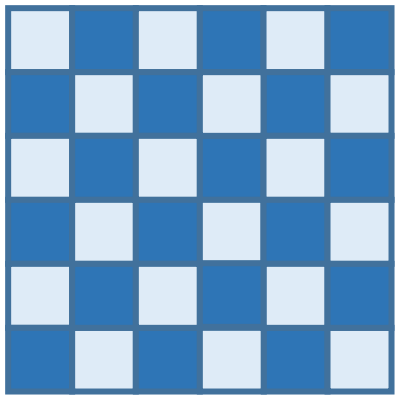


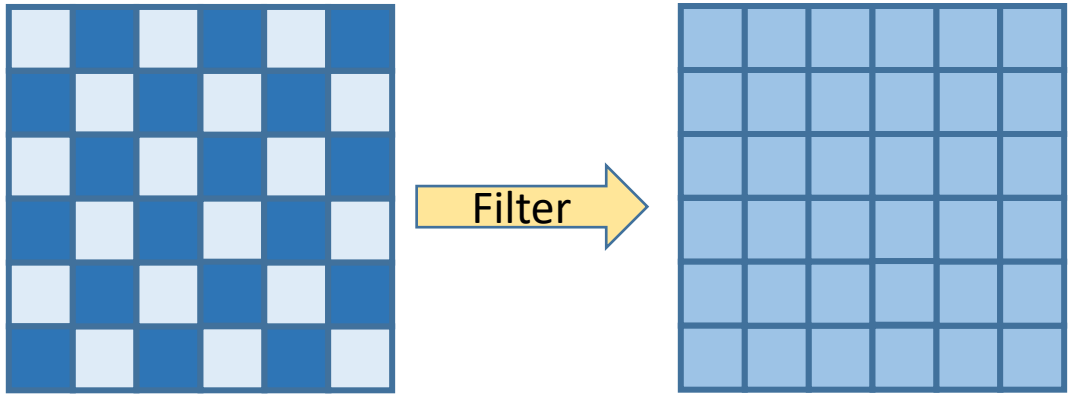
Pre-filter (blur)

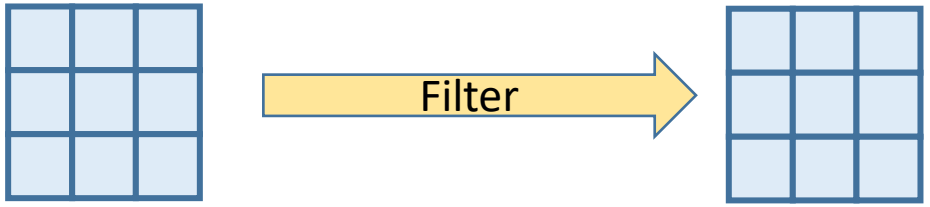
Then point sample

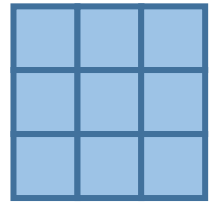


Important: it is PRE-filtering
you must filter before Aliasing occurs!









Anti-Aliasing

You cannot fix aliasing after it happens!

Take steps beforehand to avoid the worst problems

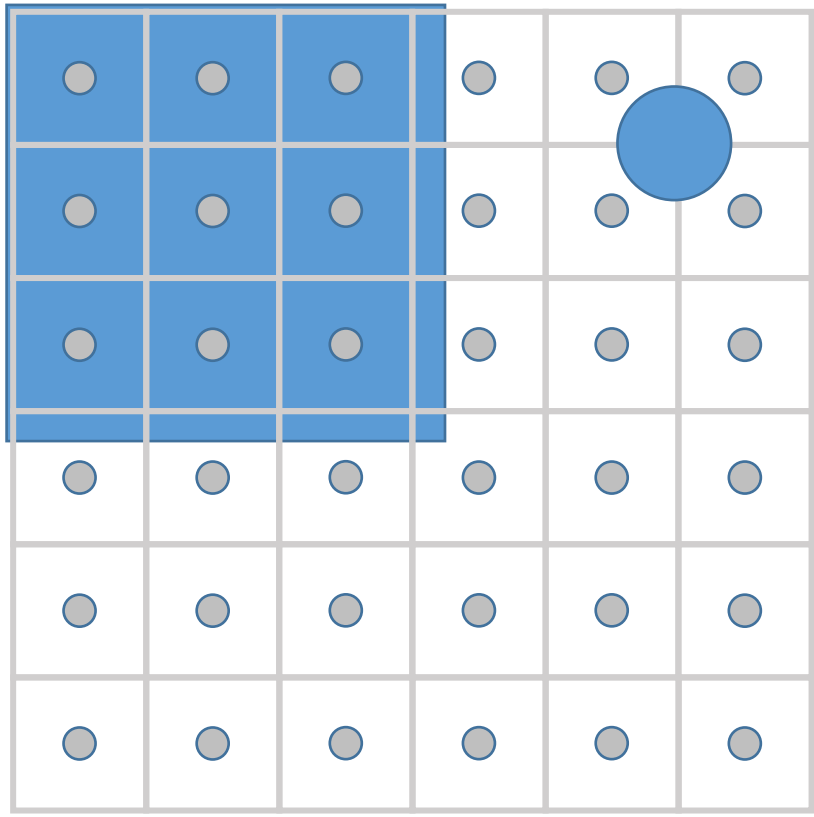
Blurry is better than wrong

In point sample land

Small things are bad

Sharp edges are bad

Worse when things move



Blur!

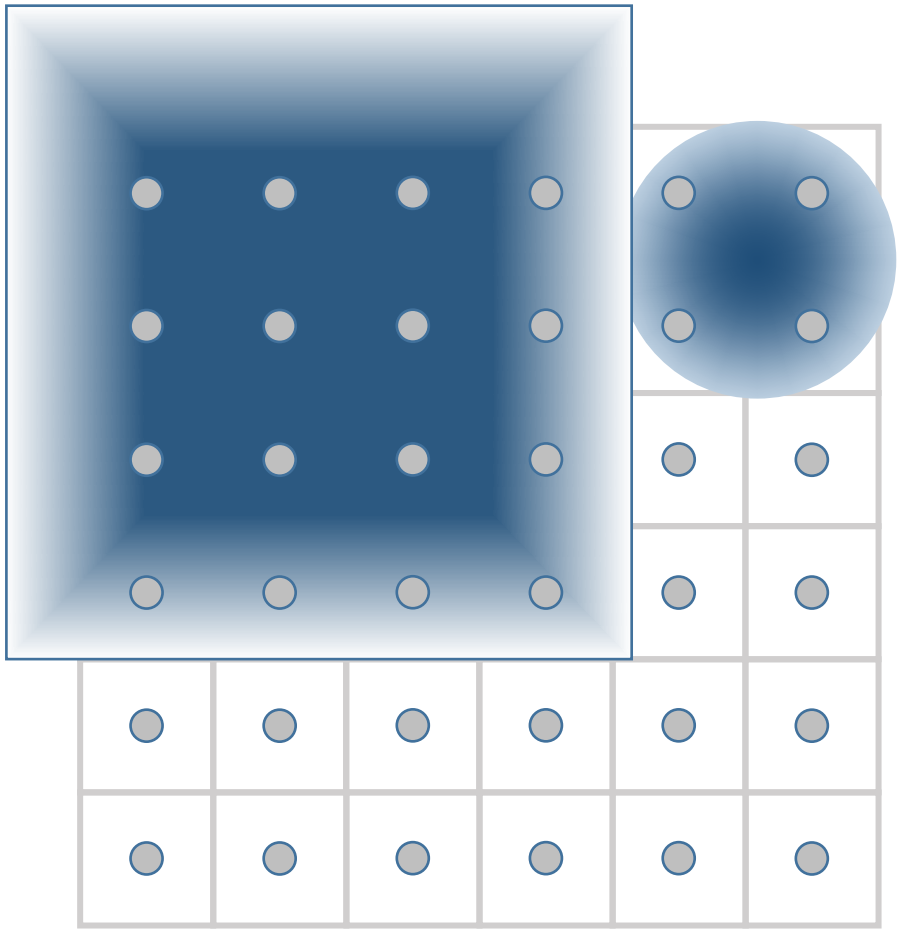
Small things are bad

Make them bigger!

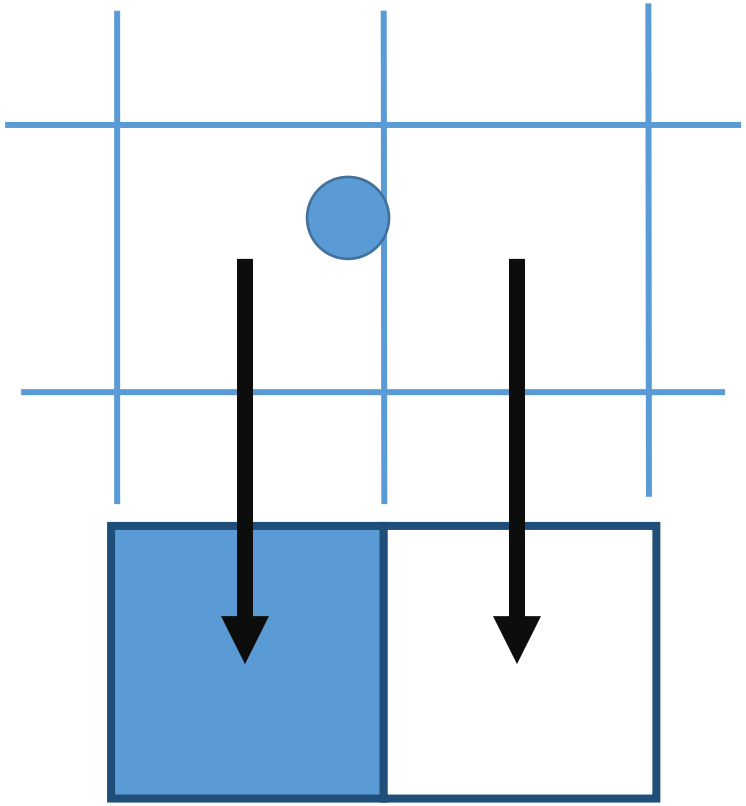
Sharp edges are bad

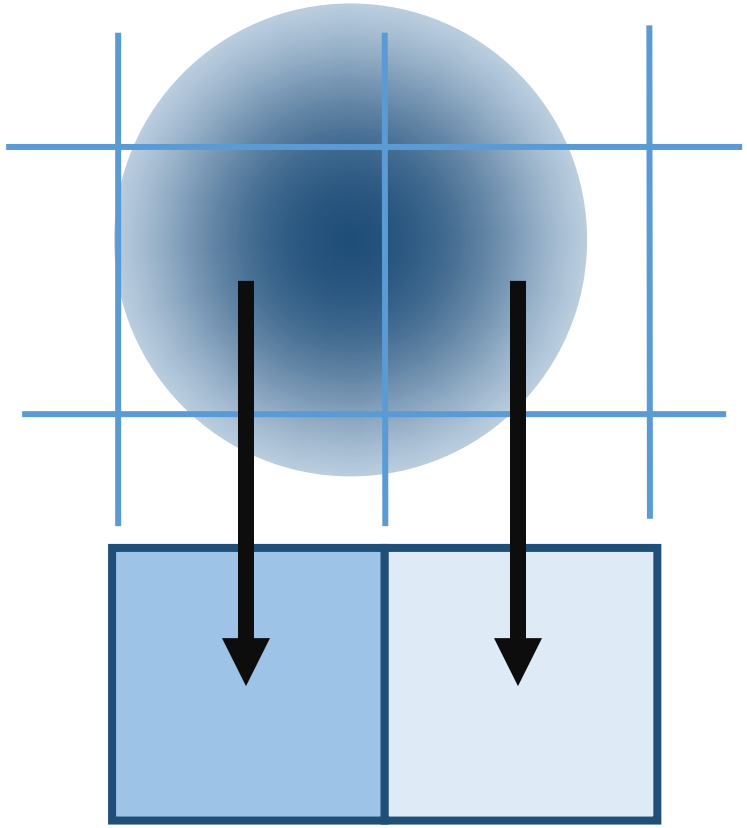
Make them smooth!

Blurry is predictable!

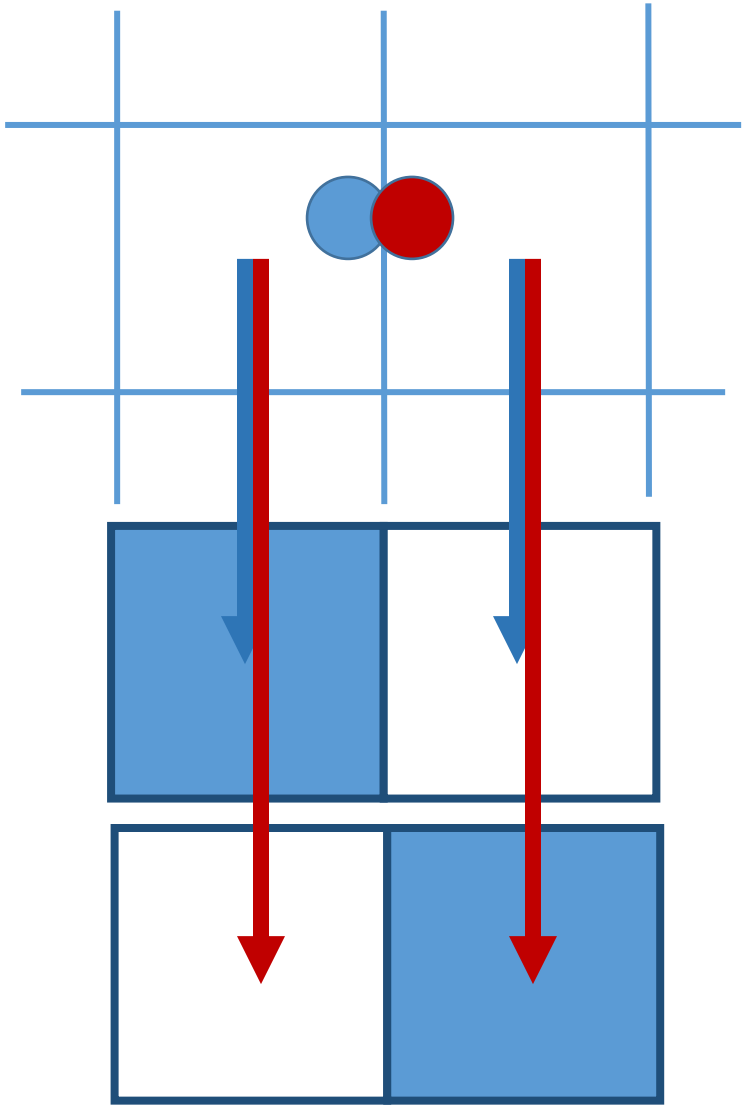


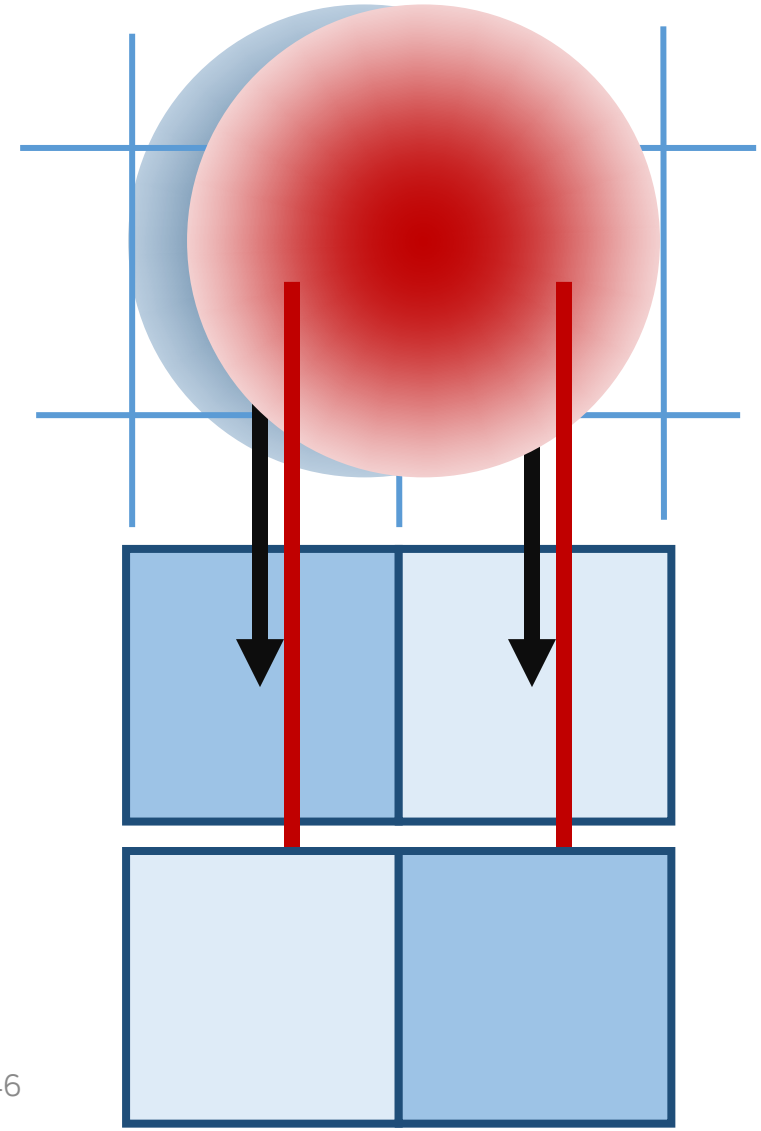
How does this help the point problem?





How does this help the point problem?



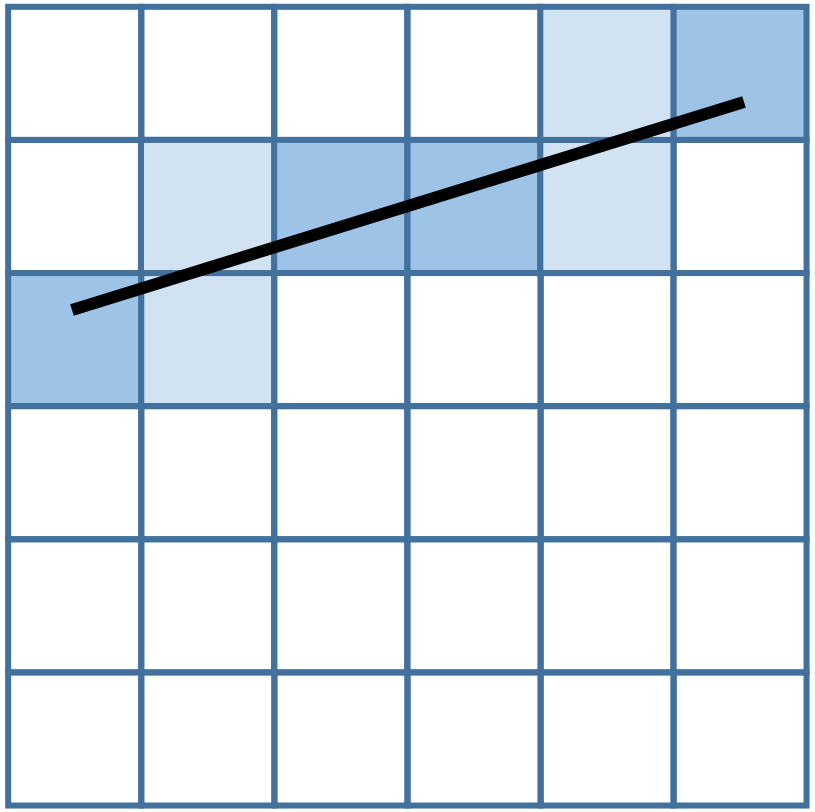


Lines

Area coverage

thick line

Partial Fill



Anti-Aliasing Primitives

It can’t be a binary yes/no decision

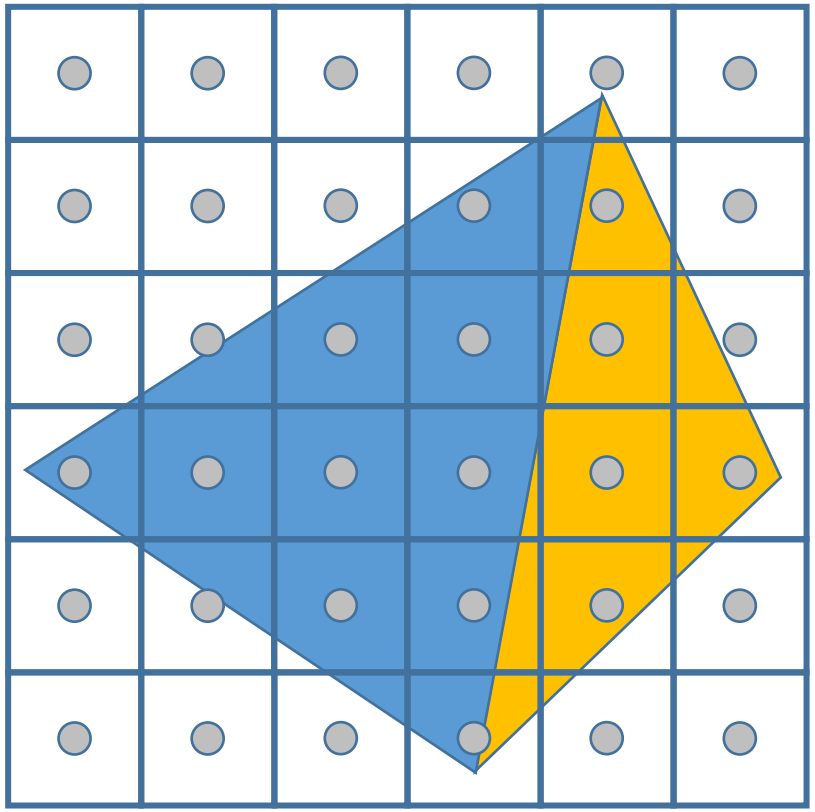
Primitive can partially fill a pixel

Blurred primitive can partially cover the sample point

Problem: what fills the other part?

Partial Fill and Triangles

A pixel may involve two (or more) triangles!



Anti-Aliasing Triangle Edges

Have pixels keep track of multiple triangles?

Hard (lots to store per pixel)

We want to keep triangles independent

Keeping Primitives Independent

Partial fill against background

Alpha channel (transparency)

Drawing order matters

Anti-Aliasing Within a Primitive

Shading computation can be aware of the whole primitive

Fragment shader – can consider an arbitrary range
Image-based shader – can consider a region

In Practice…

Anti-aliasing triangle edges is problematic

Use transparency (partial filling) when possible

requires back to front (OK for 2D)

Problem is worse when we consider visibility

Textures are easier to filter – use big triangles with texture

High quality rendering considers anti-aliasing