3D Computer Graphics

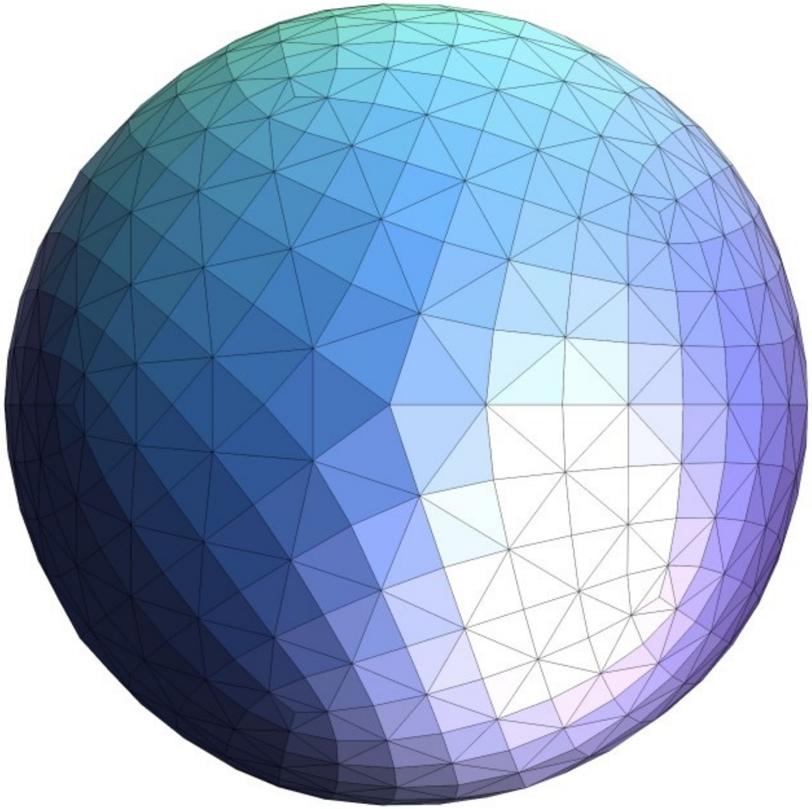
Rasterization

Dr. Zhigang Deng

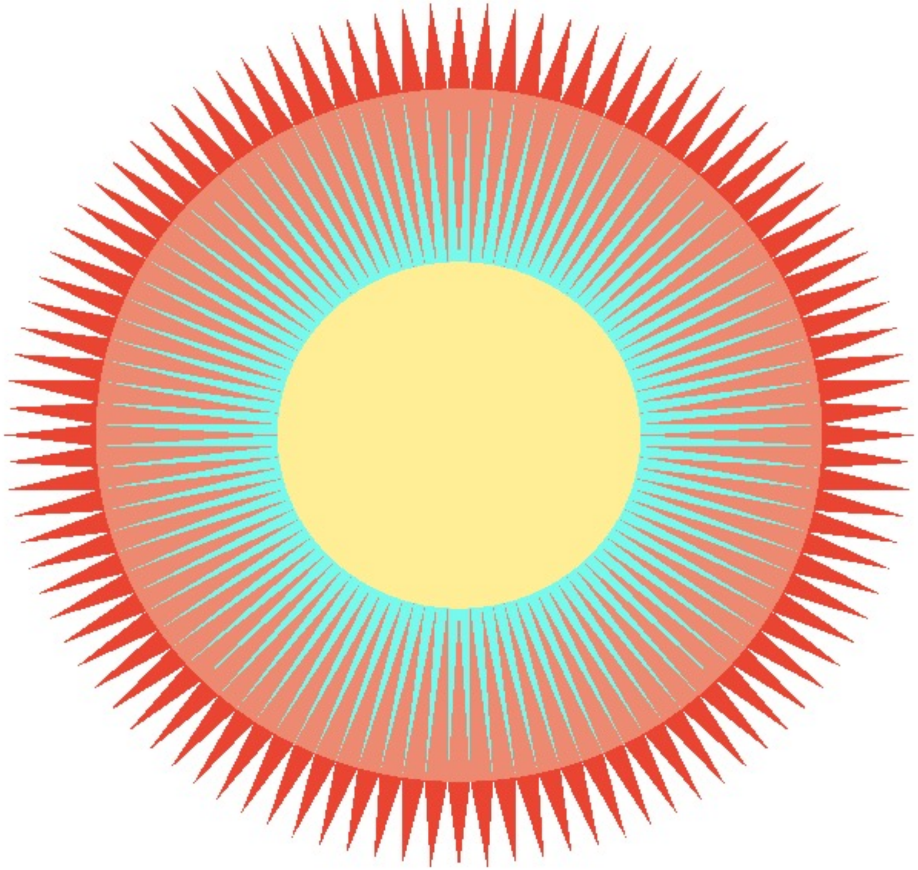
Polygon Meshes

Life of Pi (2012)

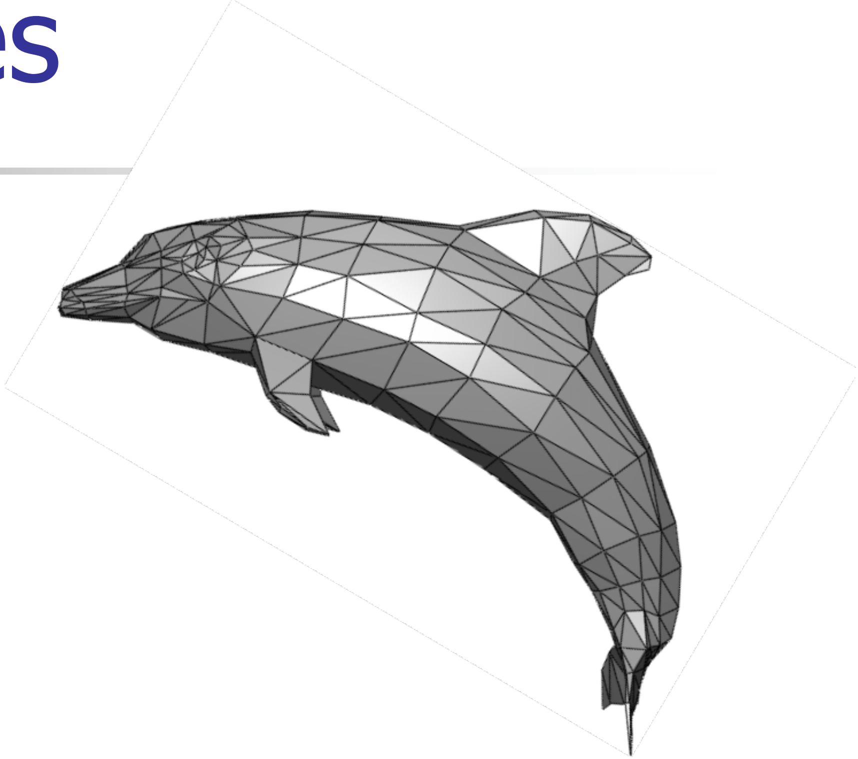
Triangle Meshes



Triangle Meshes



Triangles - Fundamental Shape Primitives

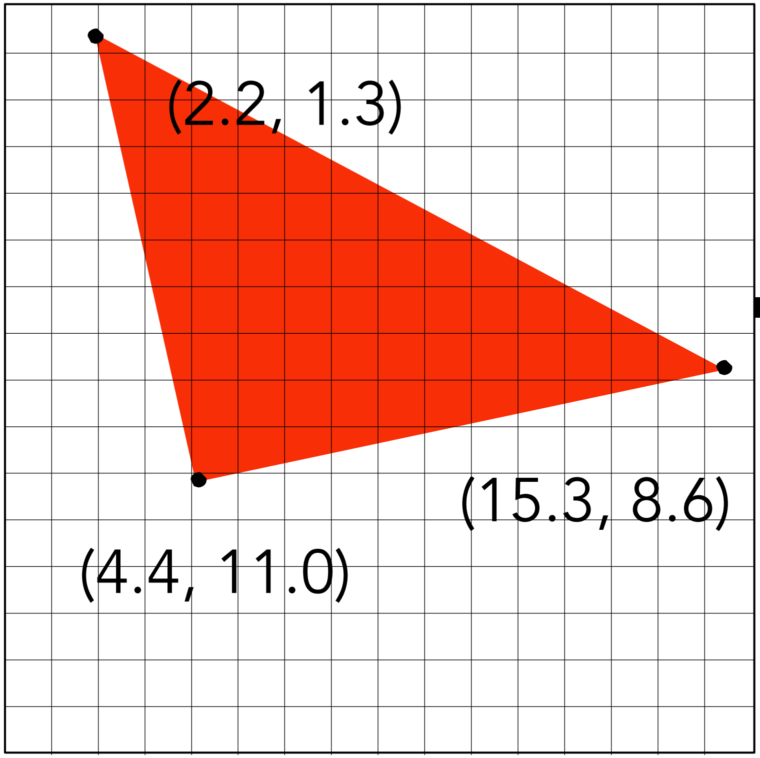


Why triangles?

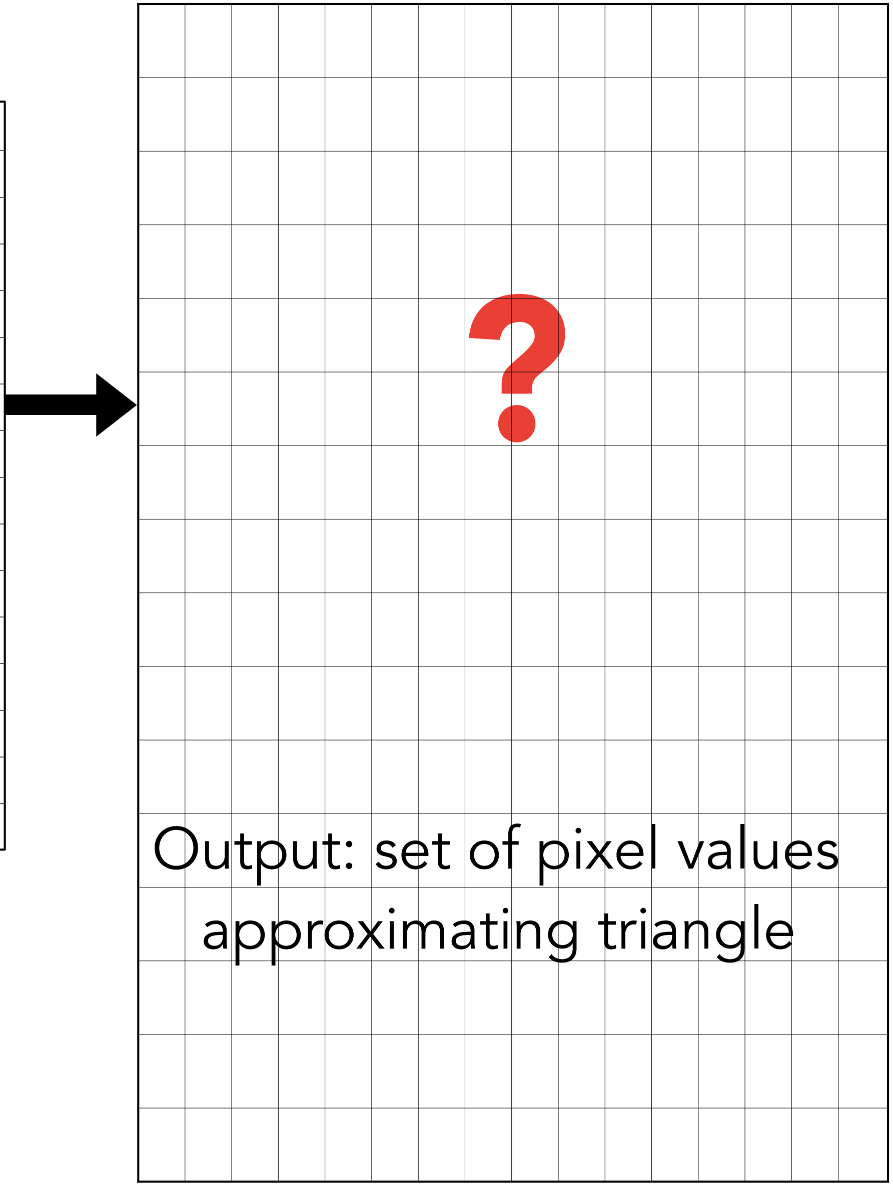
- Most basic polygon
  - Break up other polygons

- Unique properties
  - Guaranteed to be planar
  - Well-defined interior
  - Well-defined method for interpolating values at vertices over triangle (barycentric interpolation)

What Pixel Values Approximate a Triangle?



Input: position of triangle vertices projected on screen



Output: set of pixel values approximating triangle

A Simple Approach: Sampling

Sampling a Function

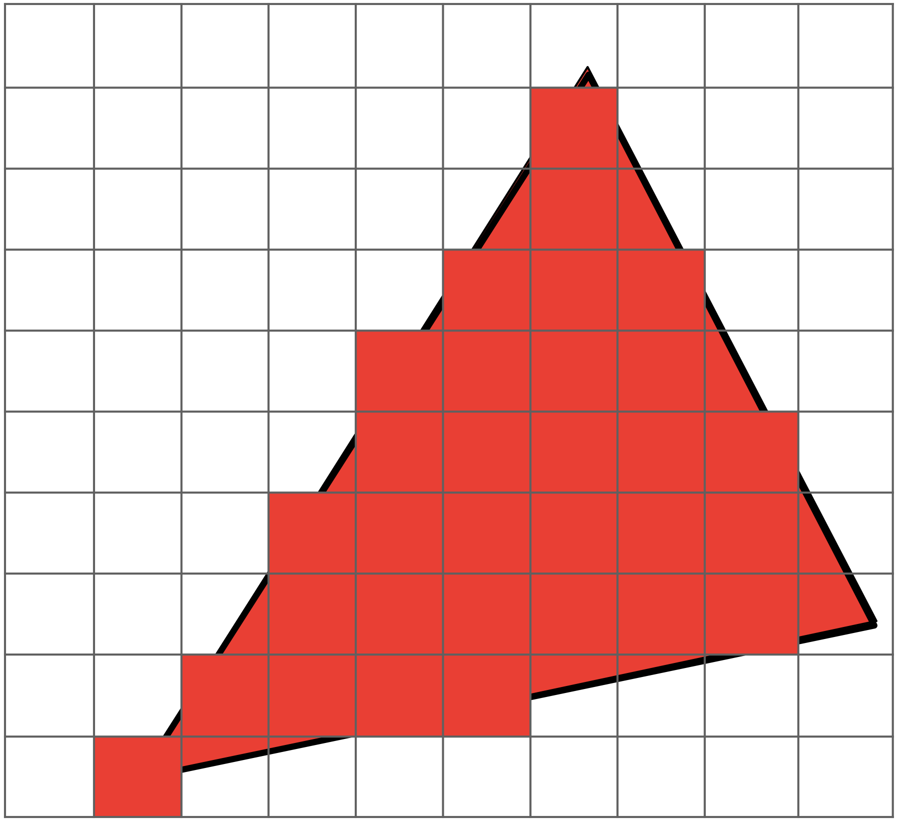
Evaluating a function at a point is sampling.

We can discretize a function by sampling.

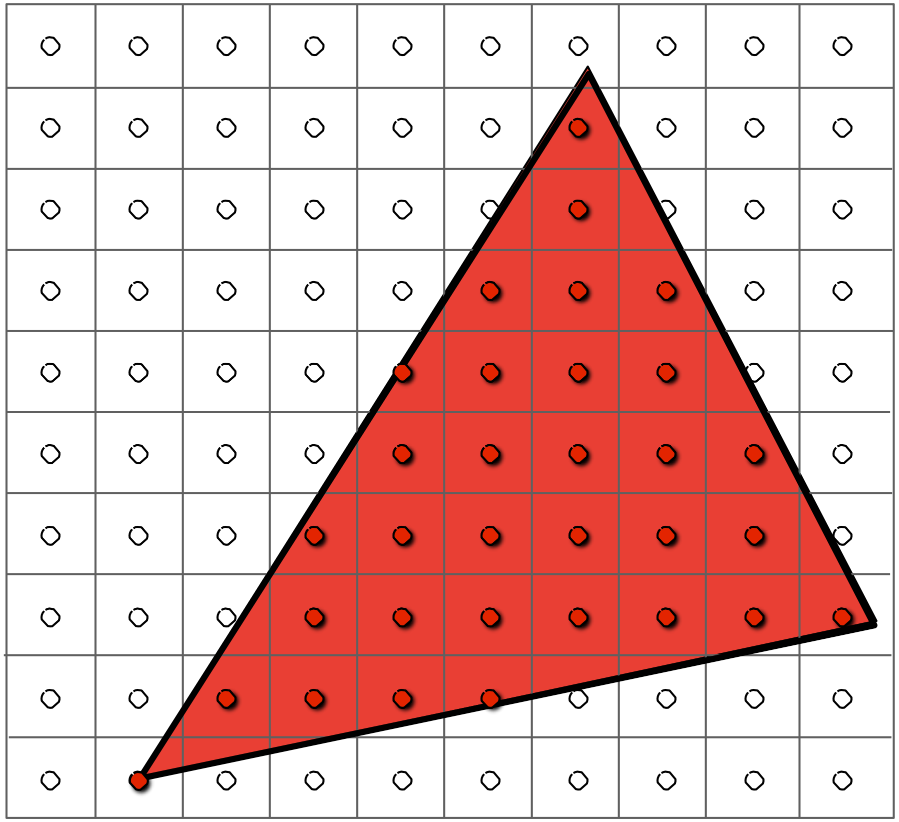
```
for (int x = 0; x < xmax; ++x)
    output[x] = f(x);
```

Sampling is a core idea in graphics.
We sample time (1D), area (2D), direction (2D), volume (3D) …

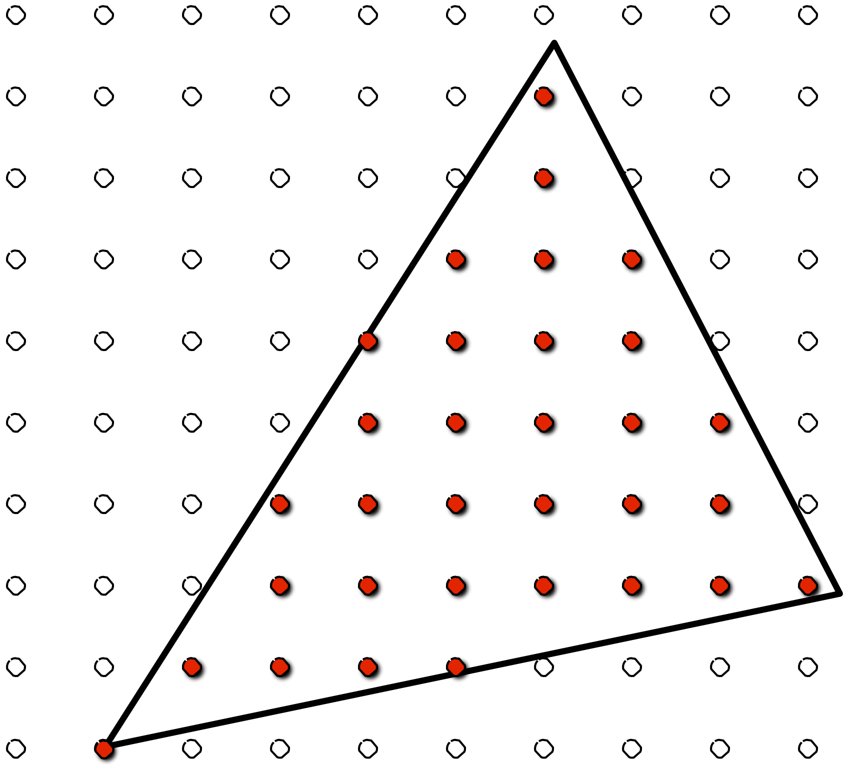
Rasterization As 2D Sampling



Sample If Each Pixel Center Is Inside Triangle

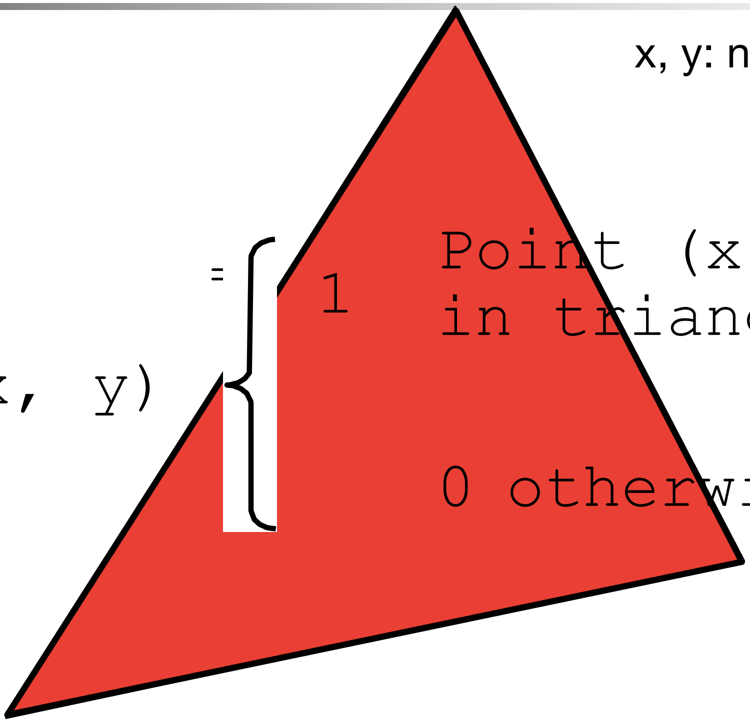


Sample If Each Pixel Center Is Inside Triangle



Define Binary Function: `inside(tri,x, y)`

x, y: not necessarily integers

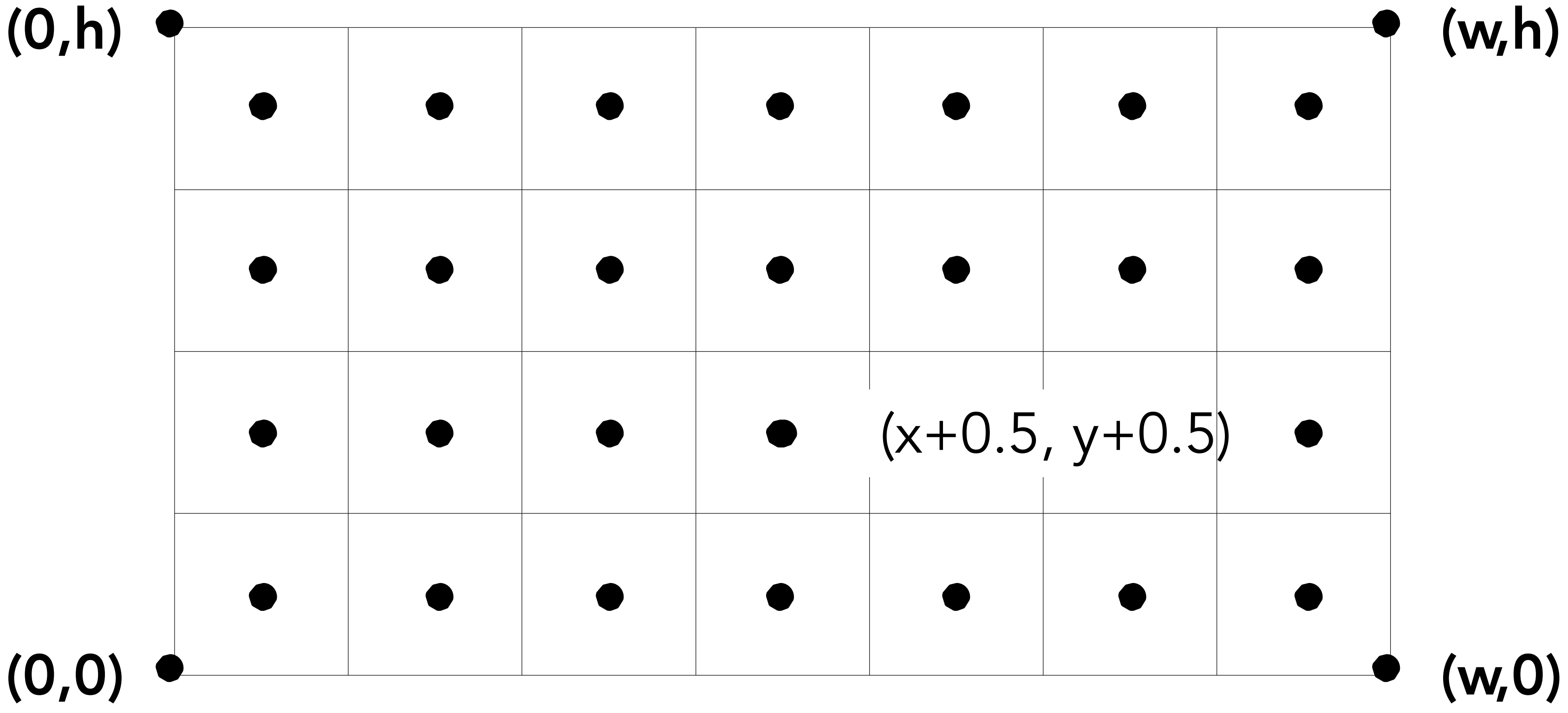


$$\texttt{inside(t, x, y)} = \begin{cases} 1 & \texttt{Point (x, y) in triangle t} \\ 0 & \texttt{otherwise} \end{cases}$$

Rasterization = Sampling A 2D Indicator Function

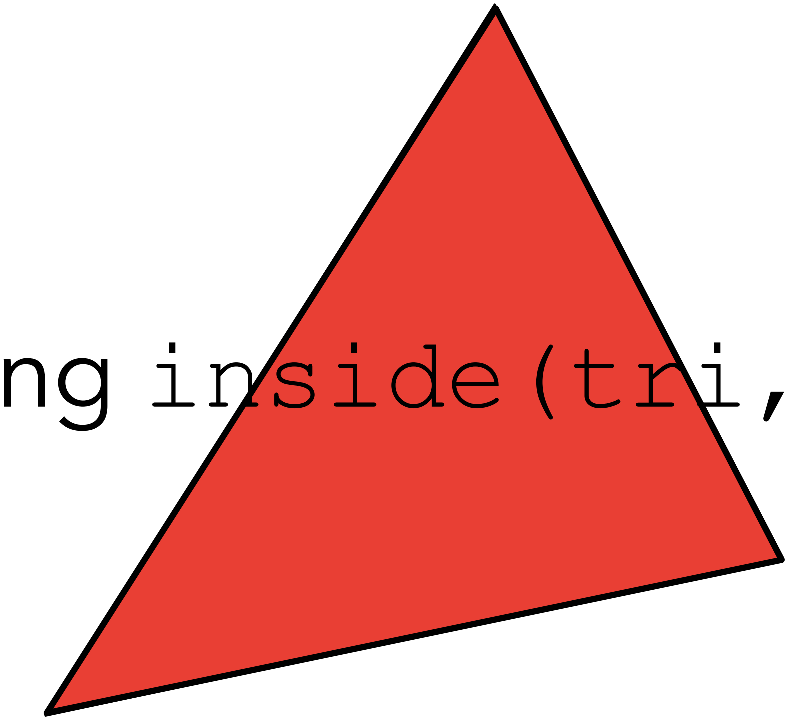
```
for (int x = 0; x < xmax; ++x)
  for (int y = 0; y < ymax; ++y)
    image[x][y] = inside(tri,
                         x + 0.5,
                         y + 0.5);
```

Recall: Sample Locations

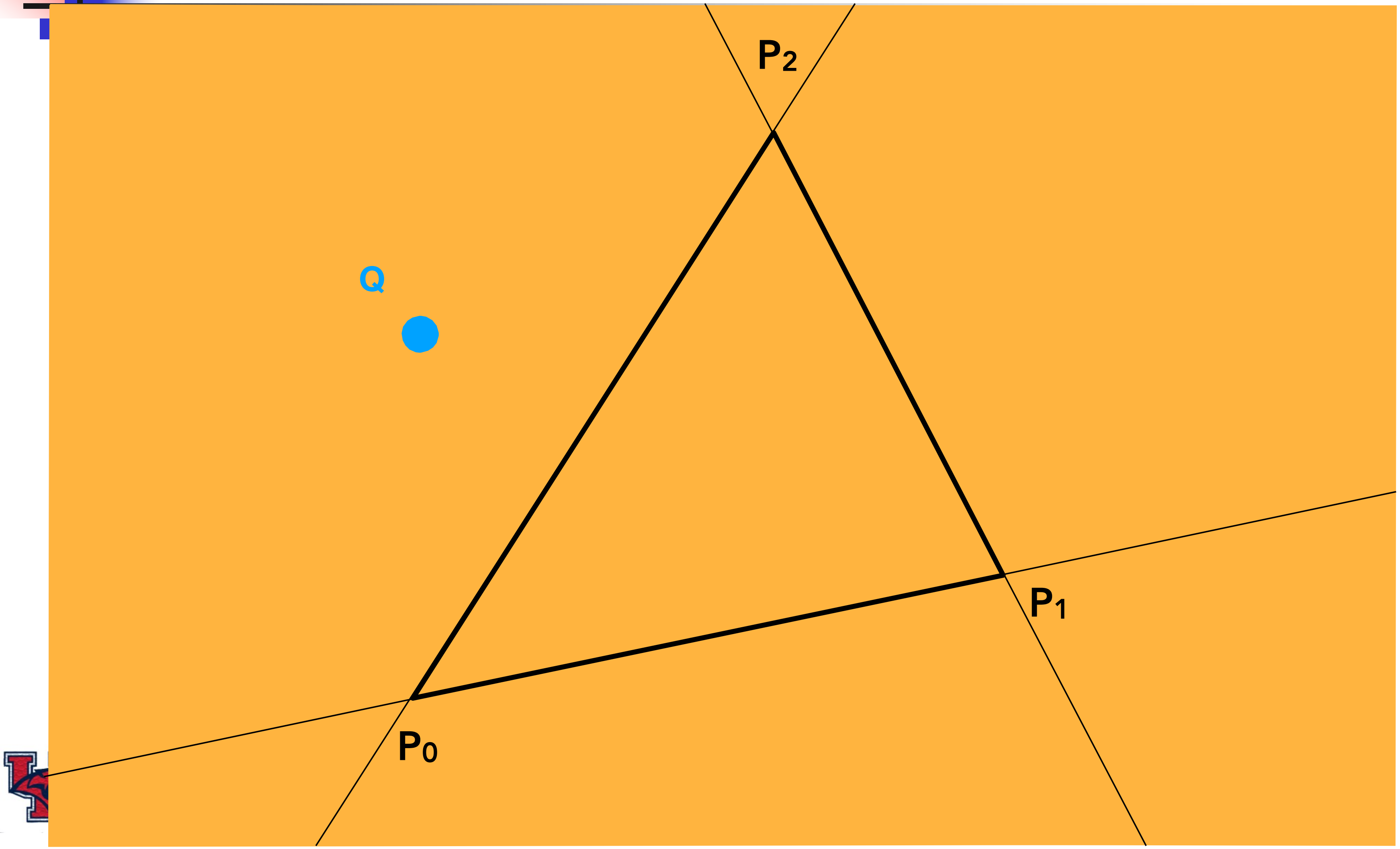


Sample location for pixel (x, y)

Evaluating `inside(tri, x, y)`

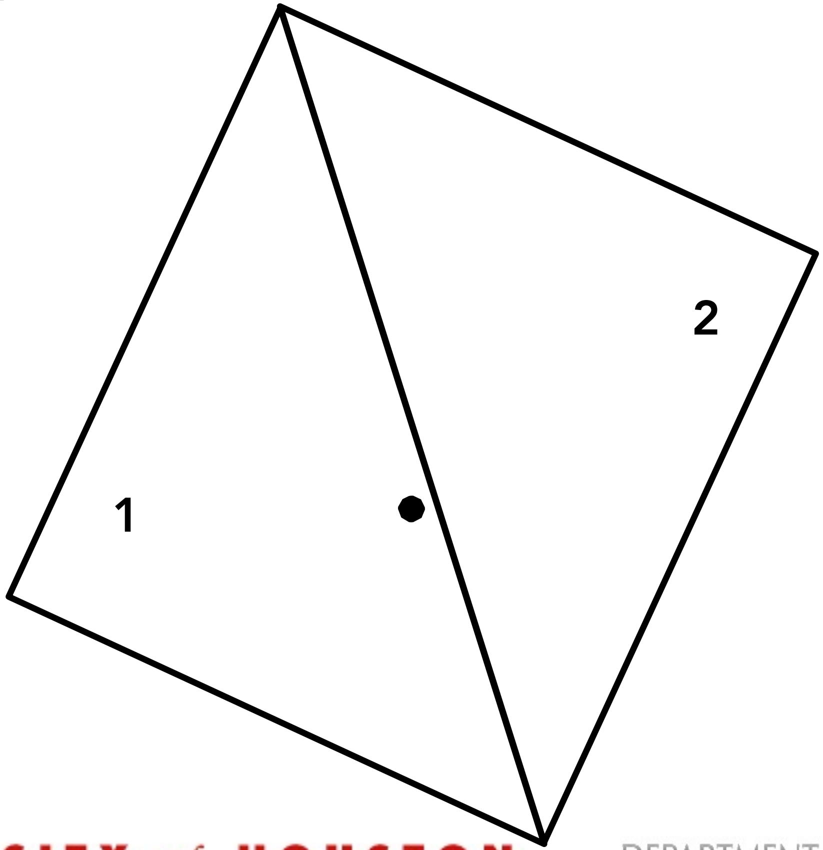


Inside? Recall: Three Cross Products!



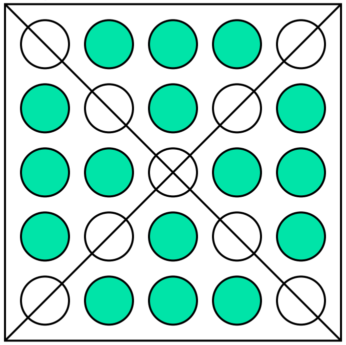
Edge Cases (Literally)

Is this sample point covered by triangle 1, triangle 2, or both?

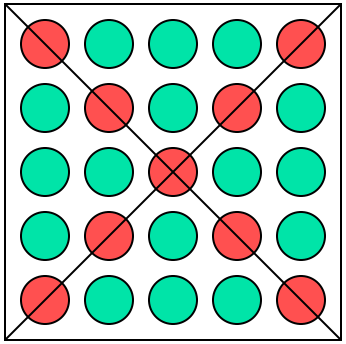


Handling Edge Pixels

Don’t use edges (e==0) – missing pixels

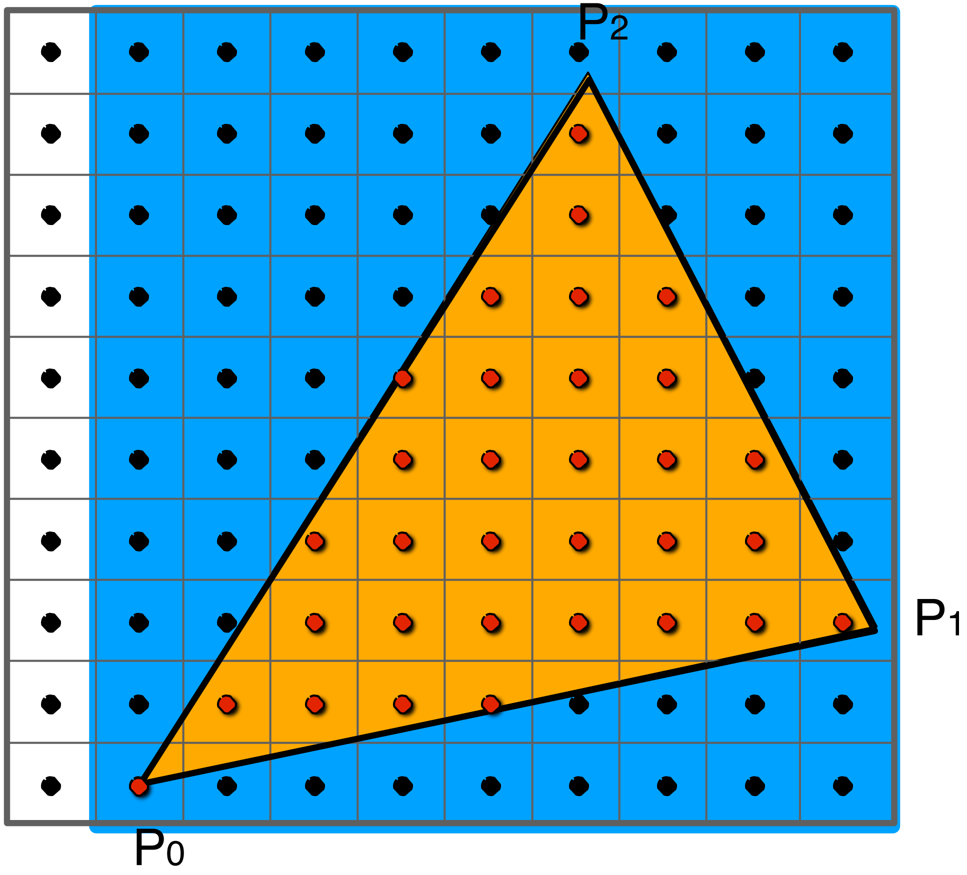


Always use edges (e==0) – waste & flicker



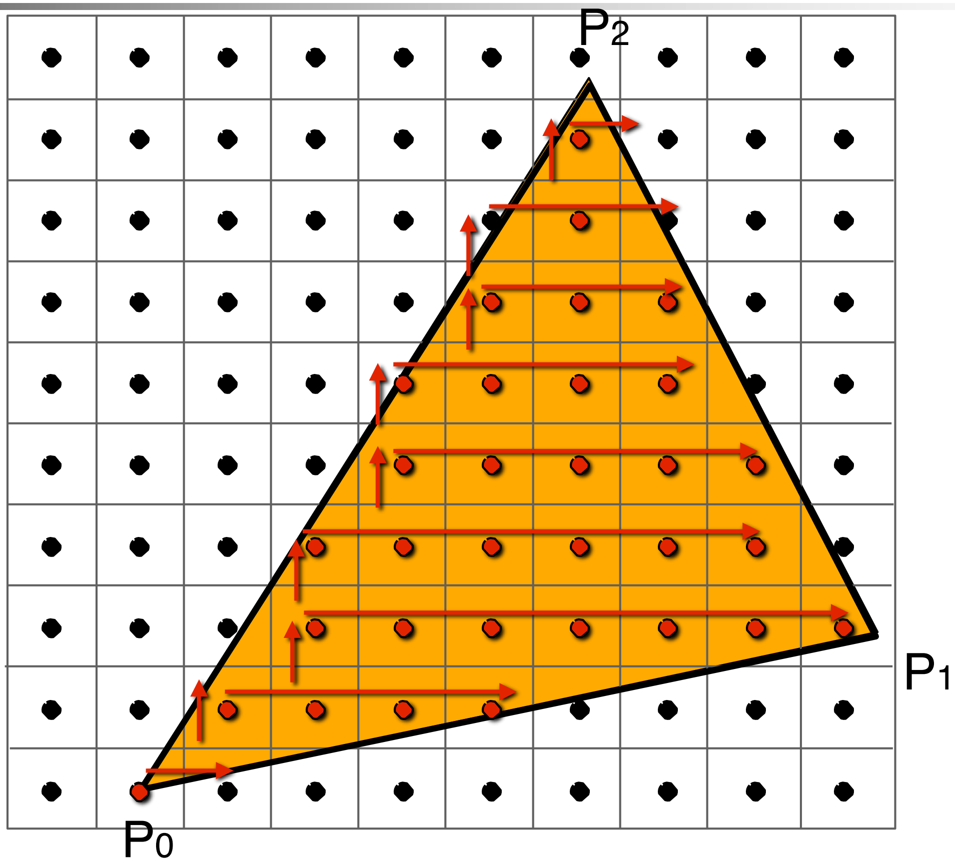
Need to include edge pixels on left or right edges (also determined by sort) to avoid <u>pinholes</u> between tris.

Checking All Pixels on the Screen?



Use a **Bounding Box!**

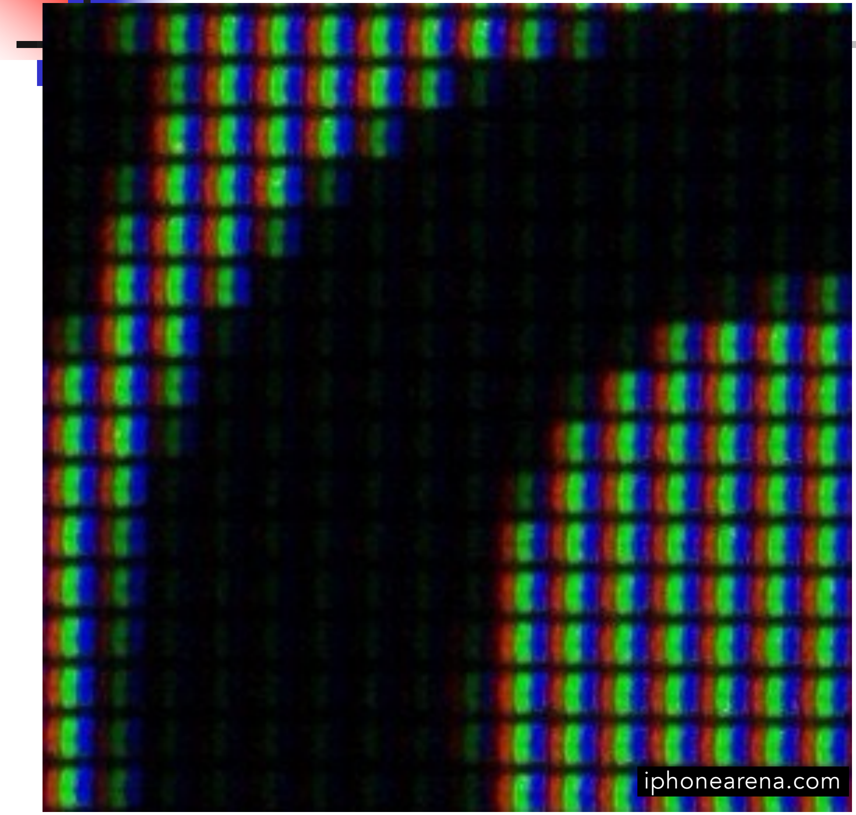
Incremental Triangle Traversal (Faster?)



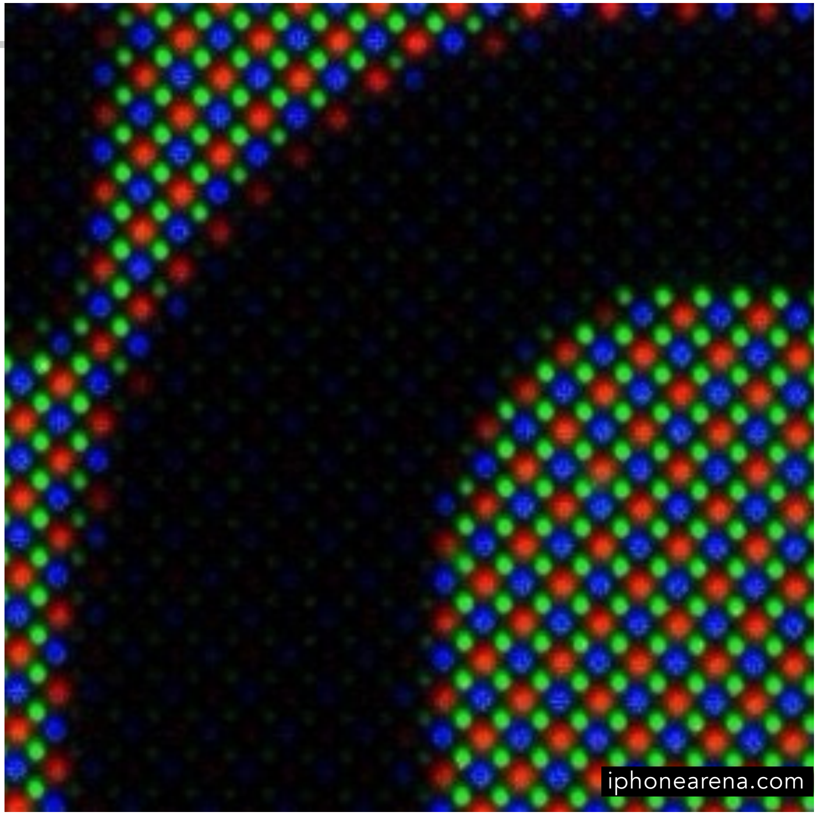
suitable for thin and rotated triangles

Rasterization on Real Displays

Real LCD Screen Pixels (Closeup)



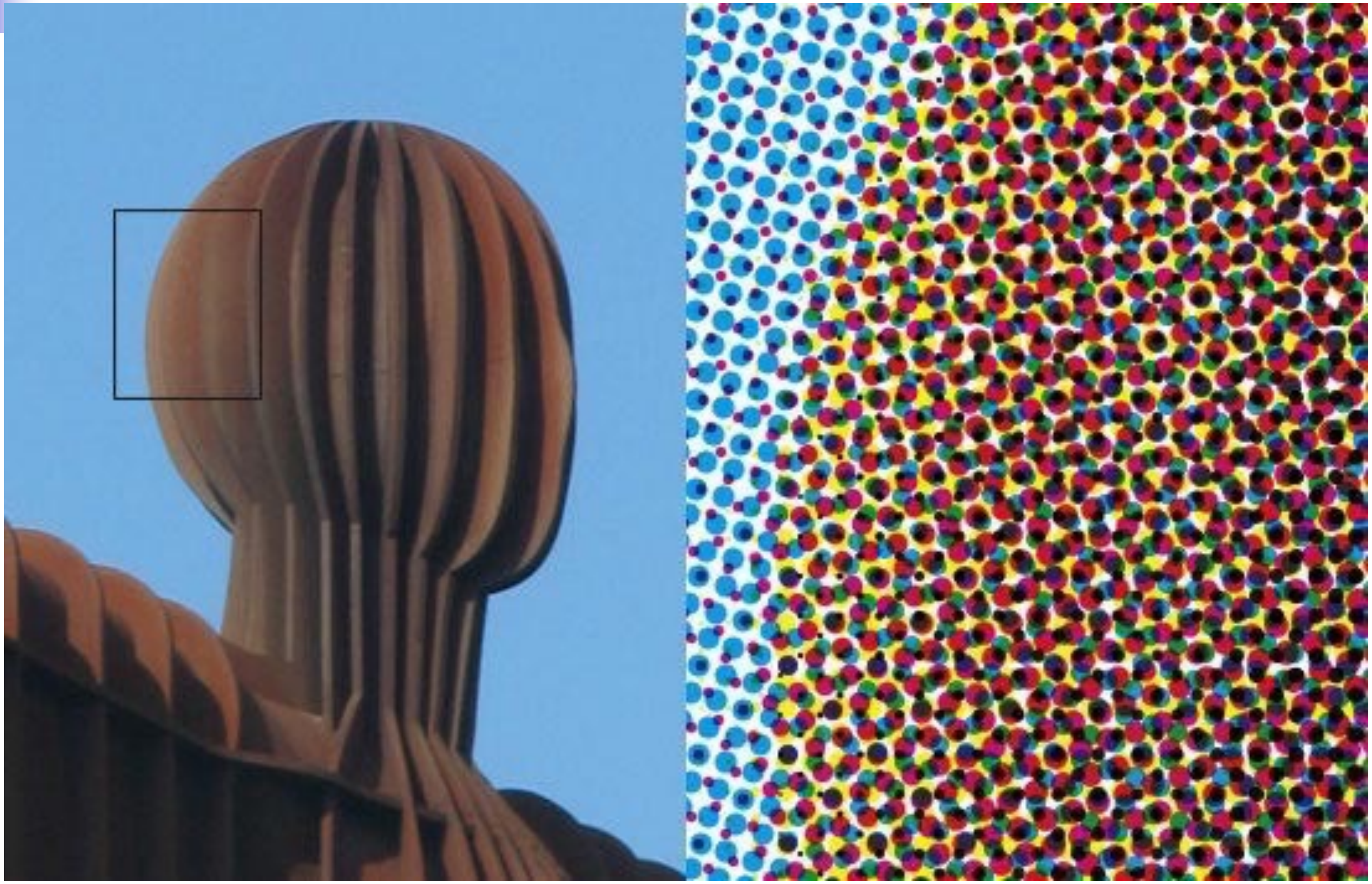
iPhone 6S



Galaxy S5

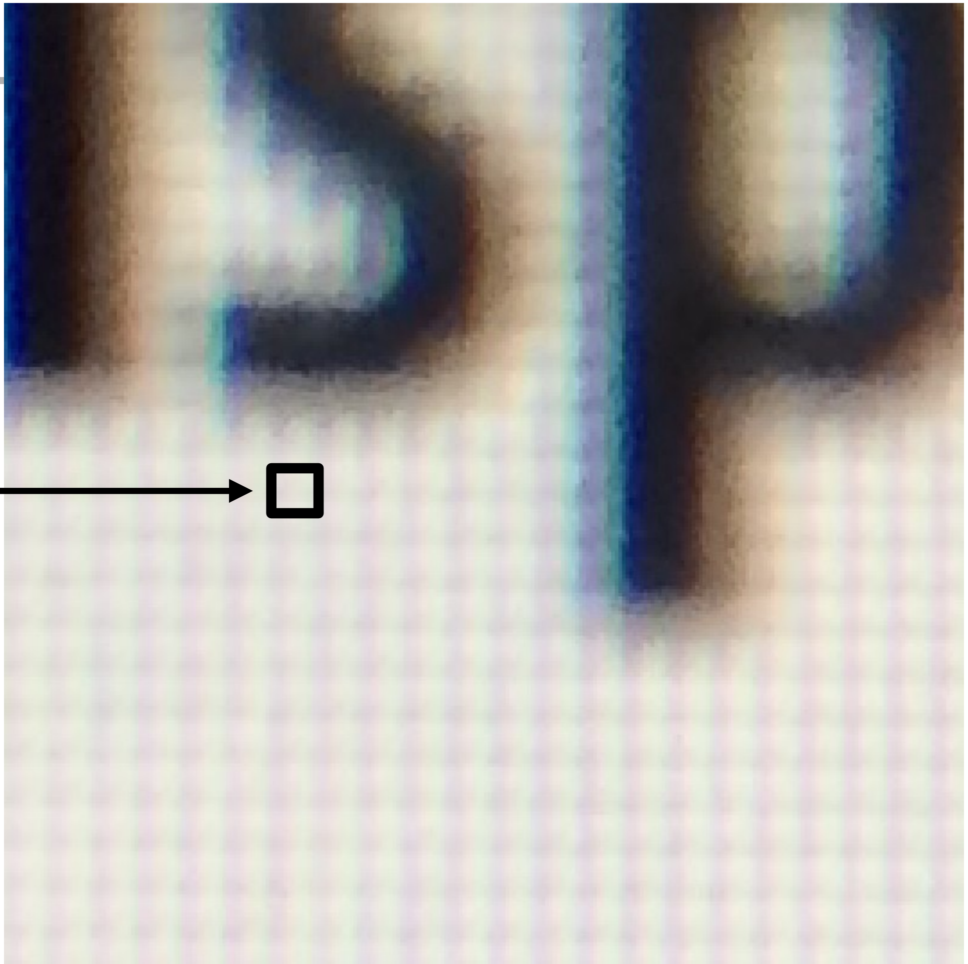
Notice R,G,B pixel geometry! But in this class, we will assume a colored square full-color pixel.

Aside: What About Other Display Methods?



Color print: observe half-tone pattern

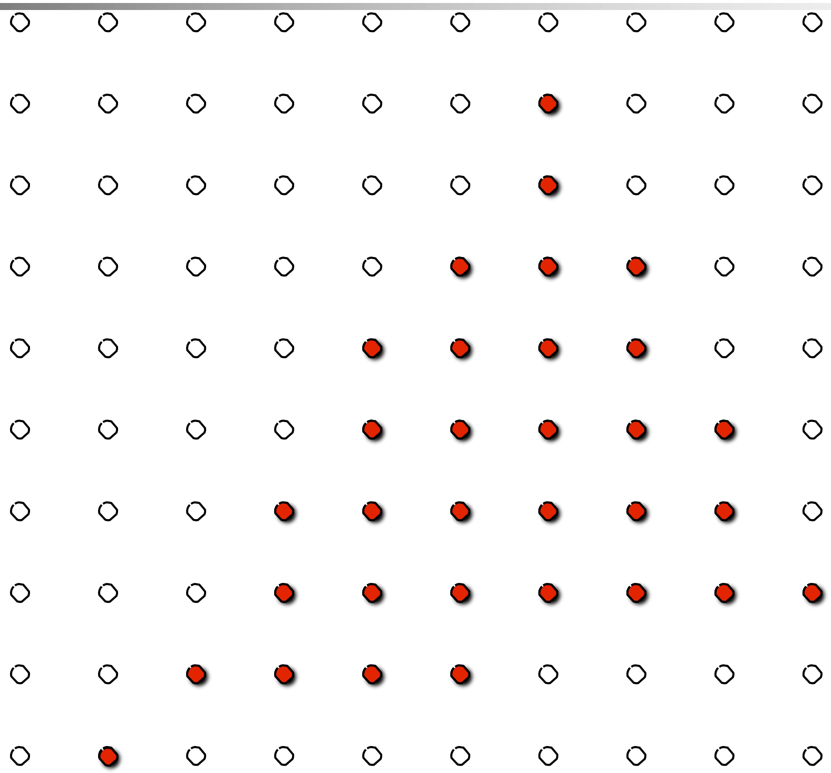
Assume Display Pixels Emit Square of Light



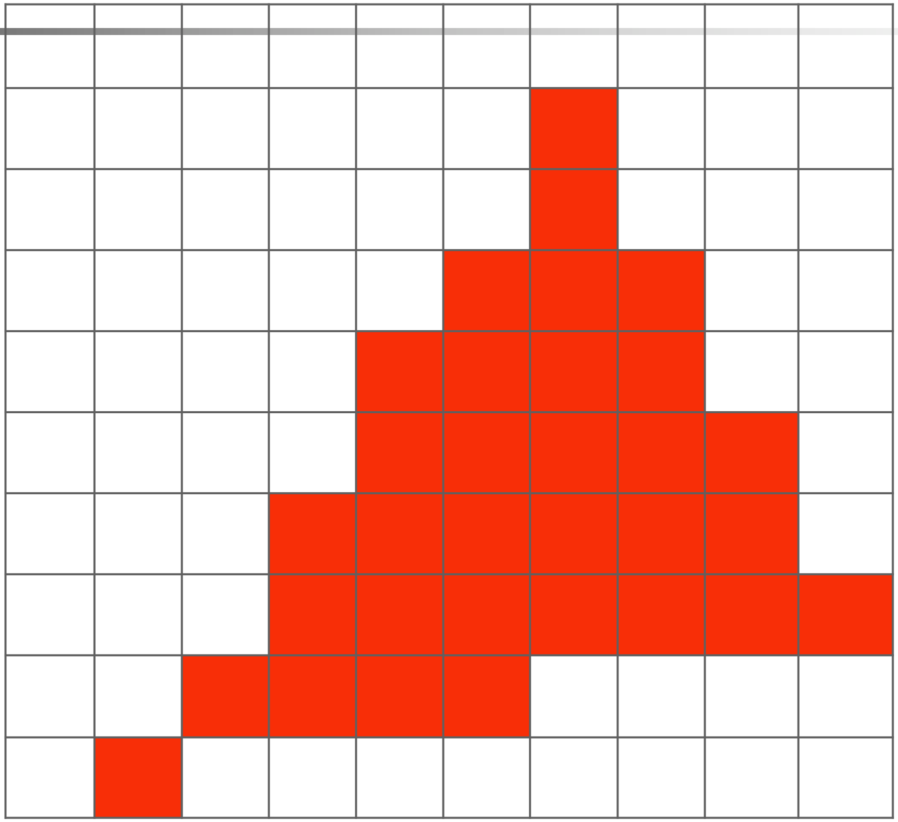
LCD pixel on laptop

* LCD pixels do not actually emit light in a square of uniform color, but this approximation suffices for our current discussion

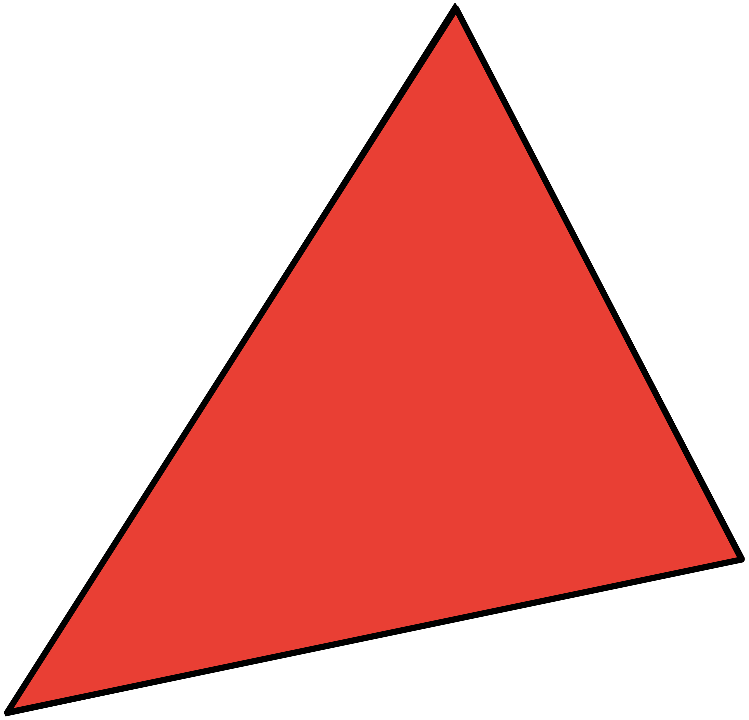
So, If We Send the Display the Sampled Signal



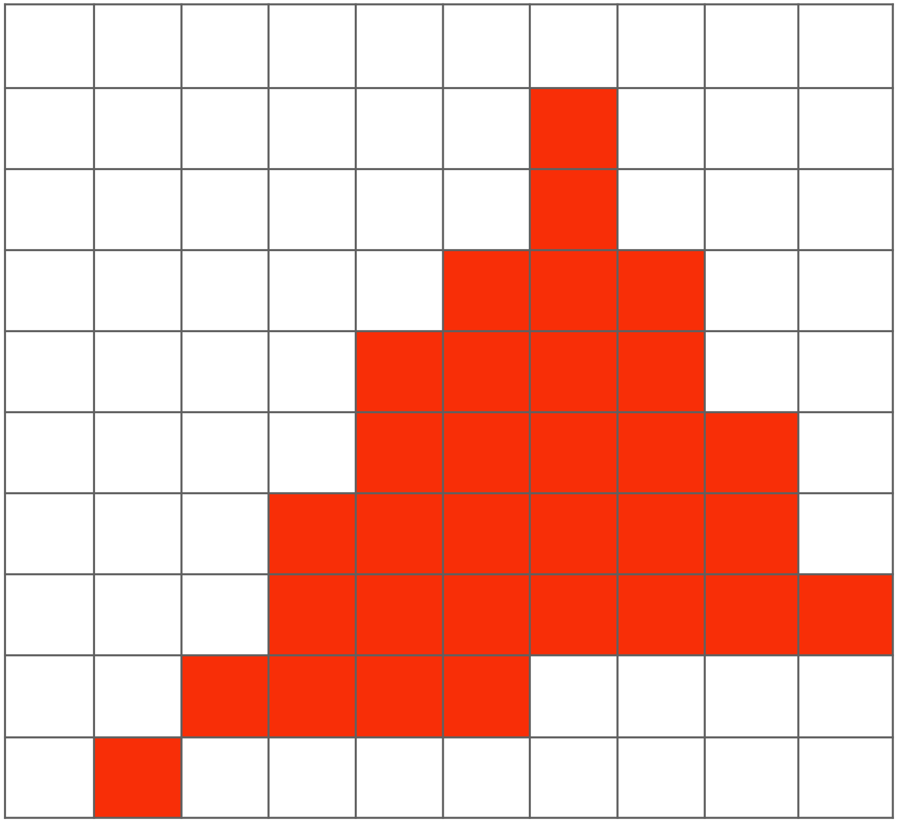
The Display Physically Emits This Signal



Compare: The Continuous Triangle Function

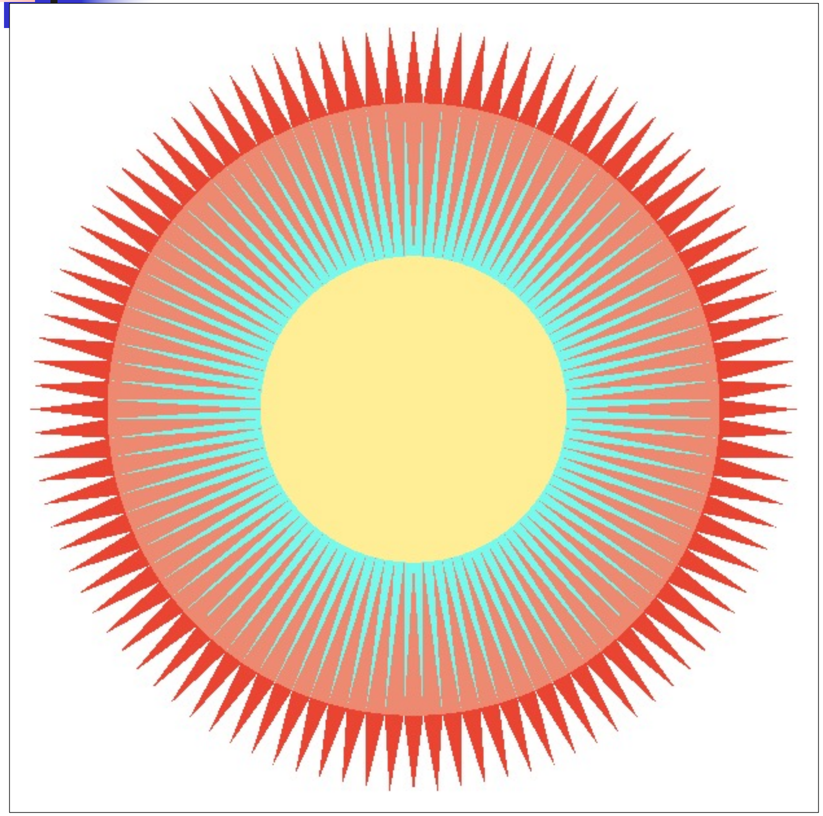


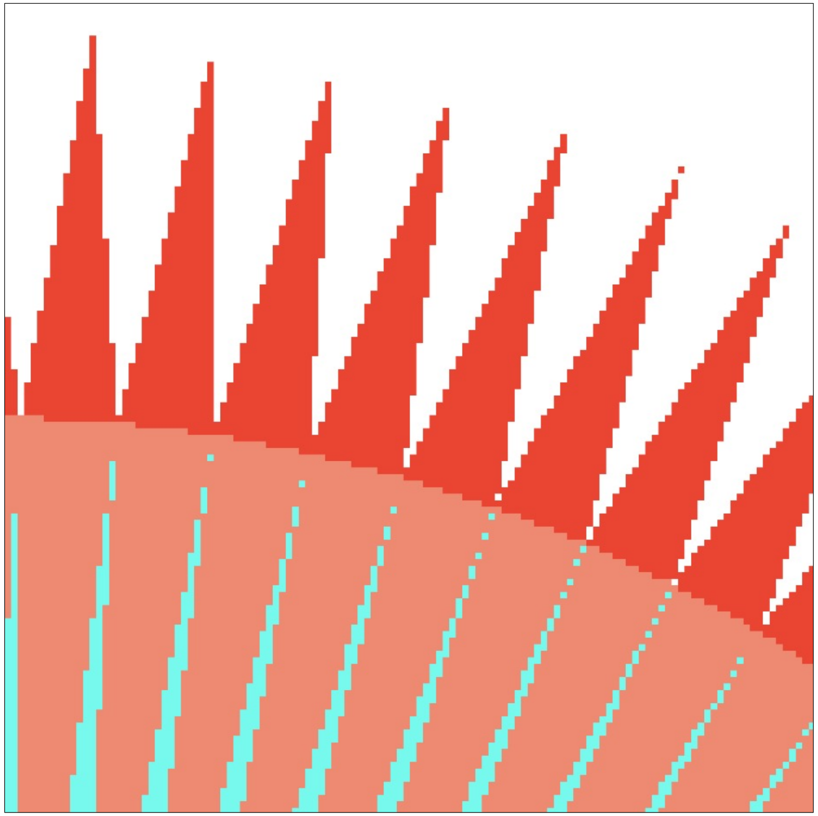
What's Wrong With This Picture?



Jaggies!

Aliasing (Jaggies)





Is this the best we can do?

Thank you!

(And thank Prof. lingqi Yan, Prof. Ravi Ramamoorthi and Prof. Ren Ng for many of the slides!)