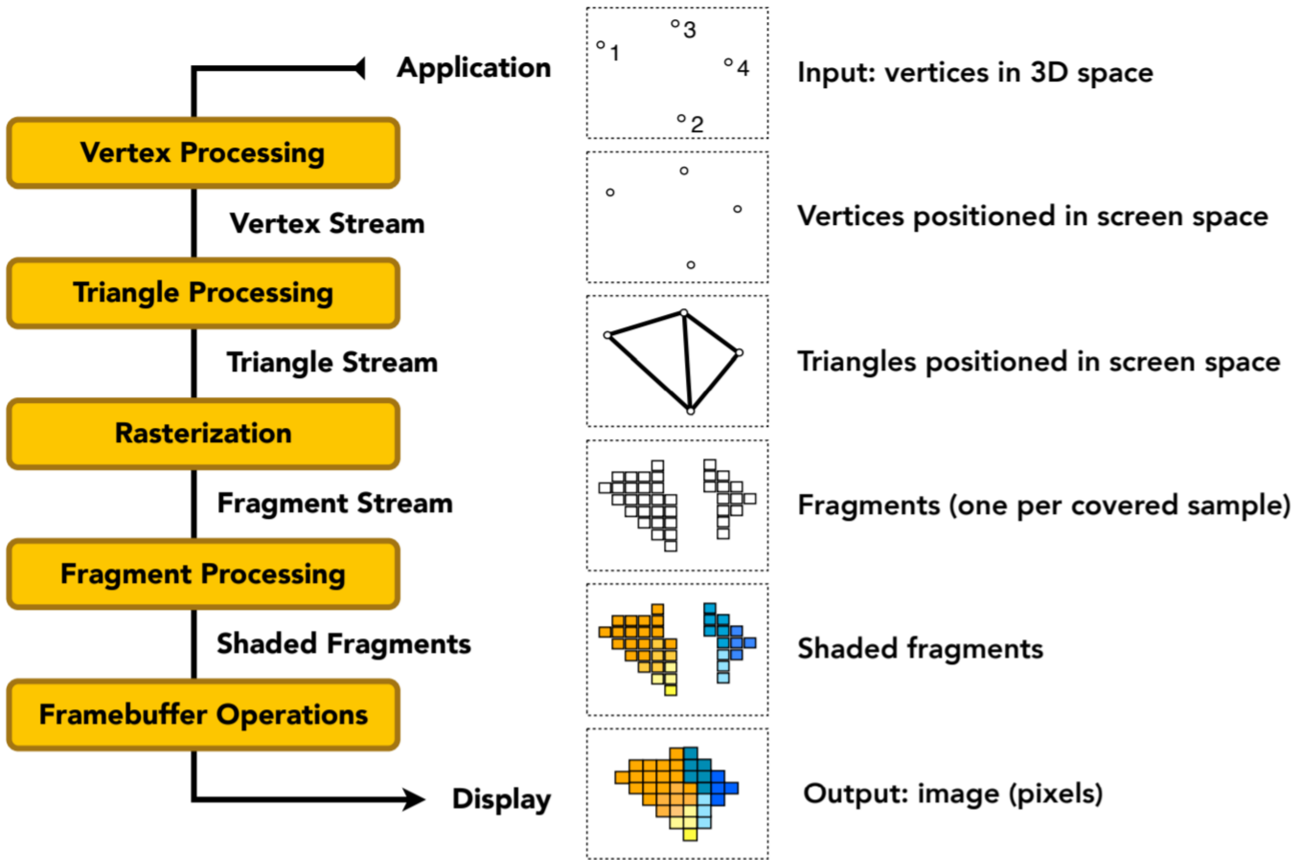
Rasterization and Anti-aliasing

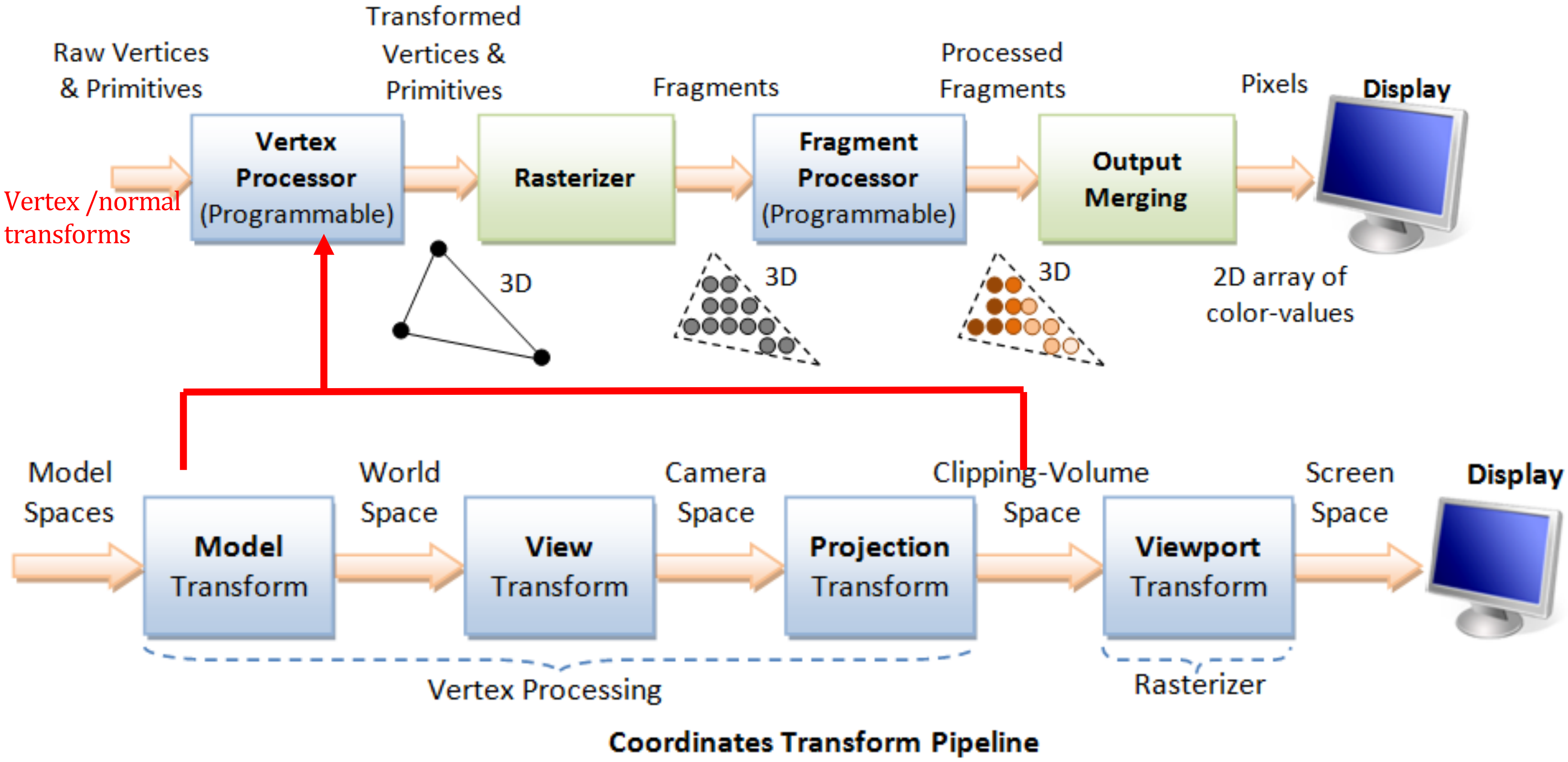
Previous

- Basic Transformation
- Viewing and Perspective
  - View (视图)/ Camera Transformation
  - Projection (投影) Transformation
- Viewport Transformation
- Geometry
  - Curves and Surfaces
  - Solid Modeling
  - Meshes

Graphics rendering pipeline



Graphics rendering pipeline



Today's Topic

- Rasterization
  - Rasterizing Triangles into Pixels
  - Aliasing and Antialiasing
  - Sampling Theory

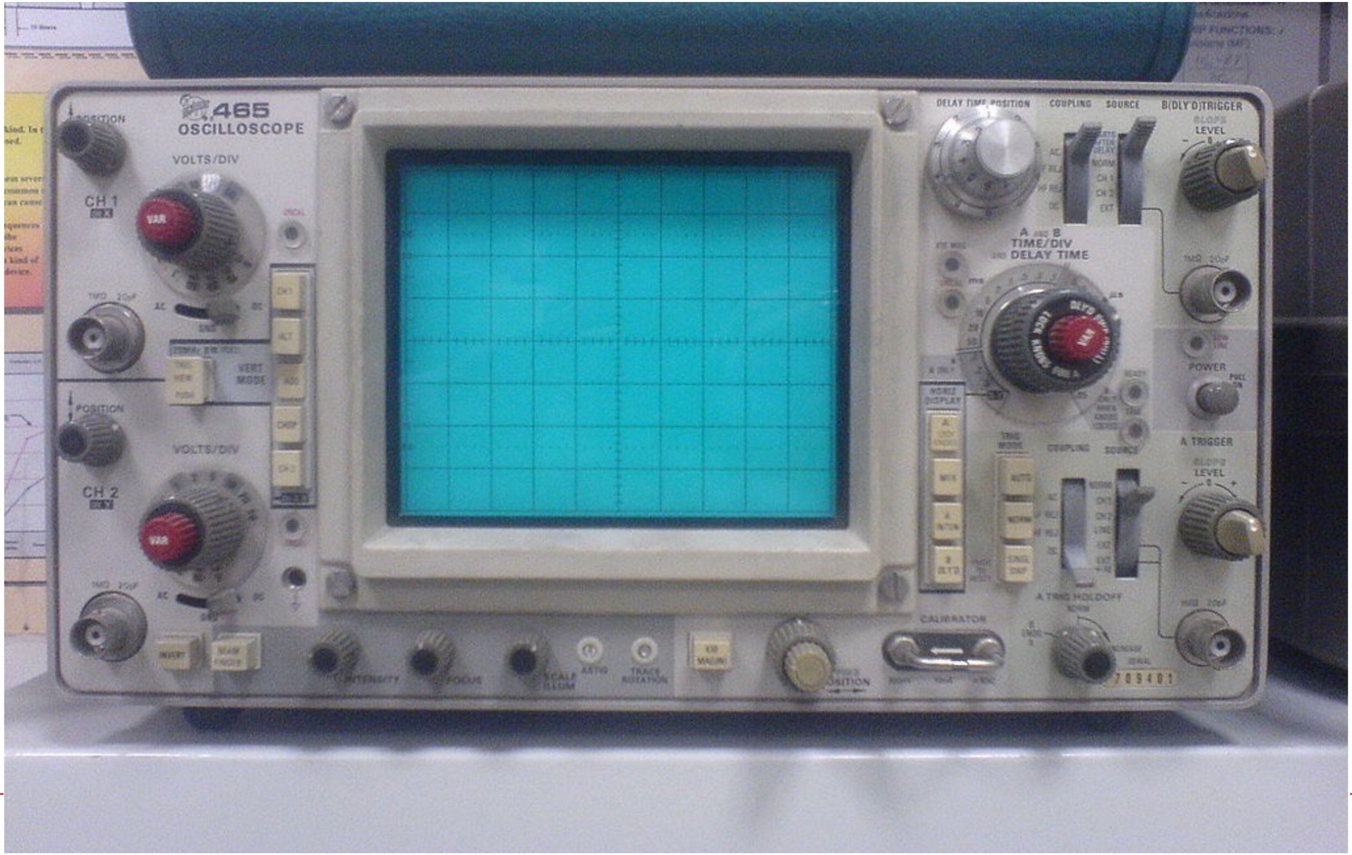
Today's Topic

- Rasterization
  - Rasterizing Triangles into Pixels
  - Aliasing and Antialiasing
  - Sampling Theory

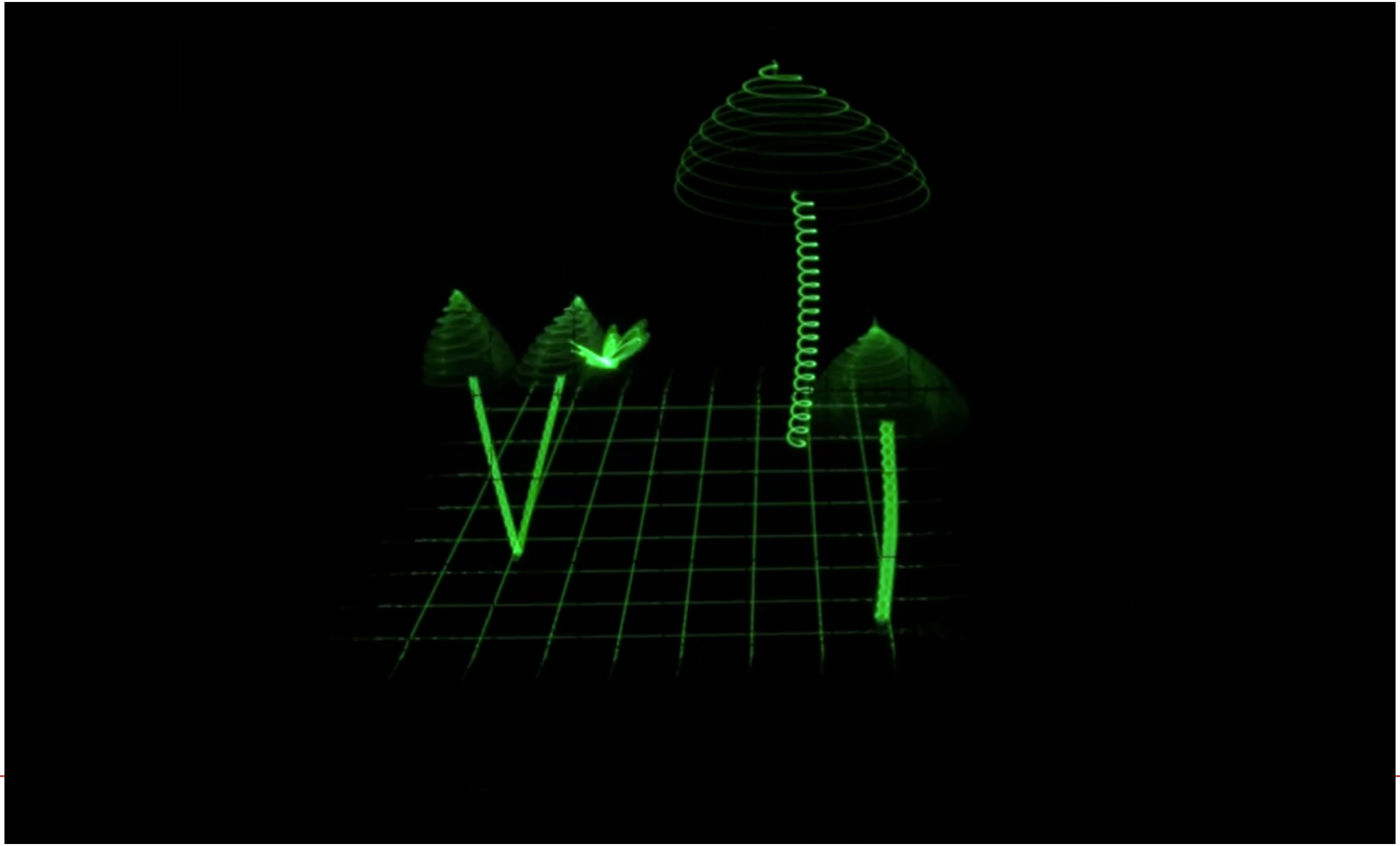
Rasterizing Triangles into Pixels

Different Raster Displays

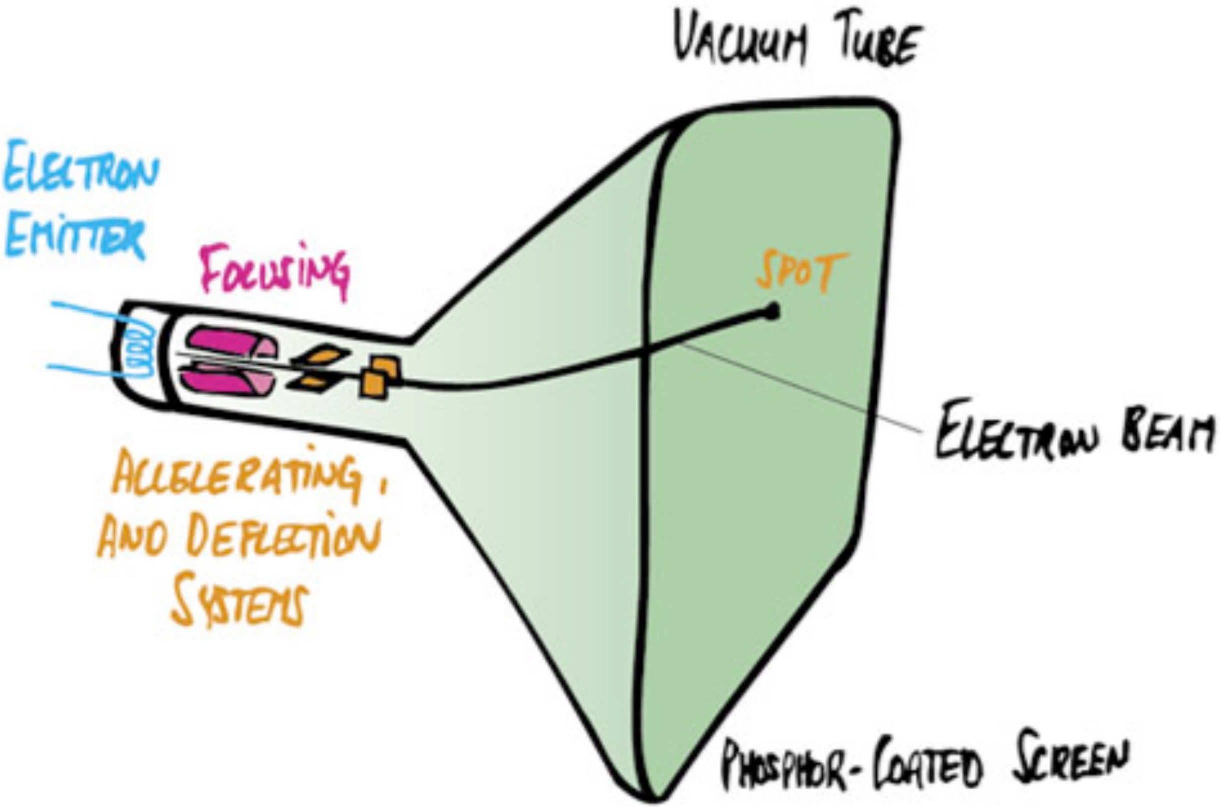
Osilloscope



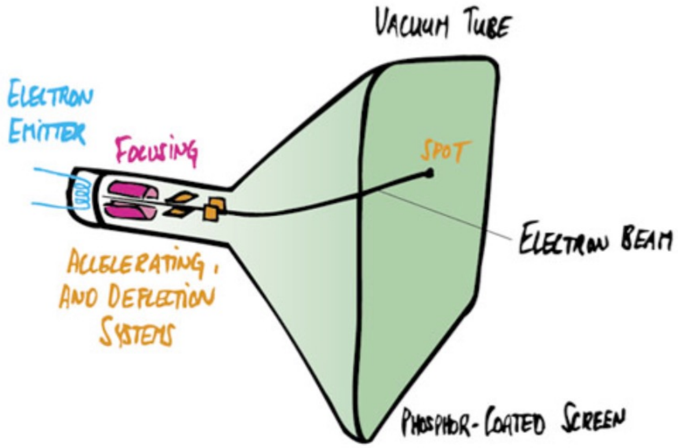
Osilloscope Art



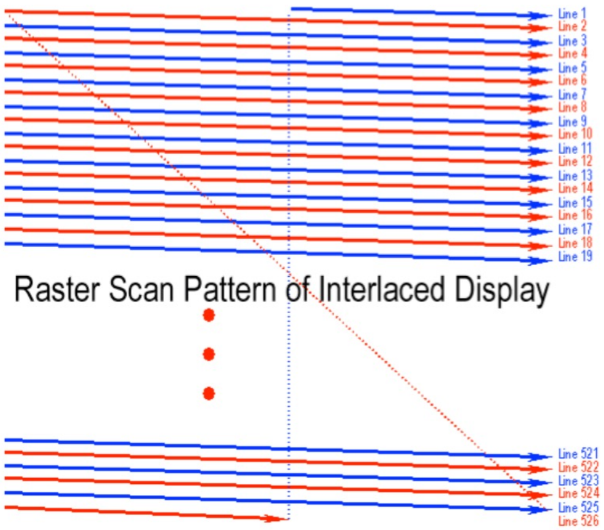
Cathode Ray Cube



Television - Raster Display CRT



Cathode Ray Tube



Raster Scan
(modulate intensity)

Frame Buffer: Memory for a Raster Display

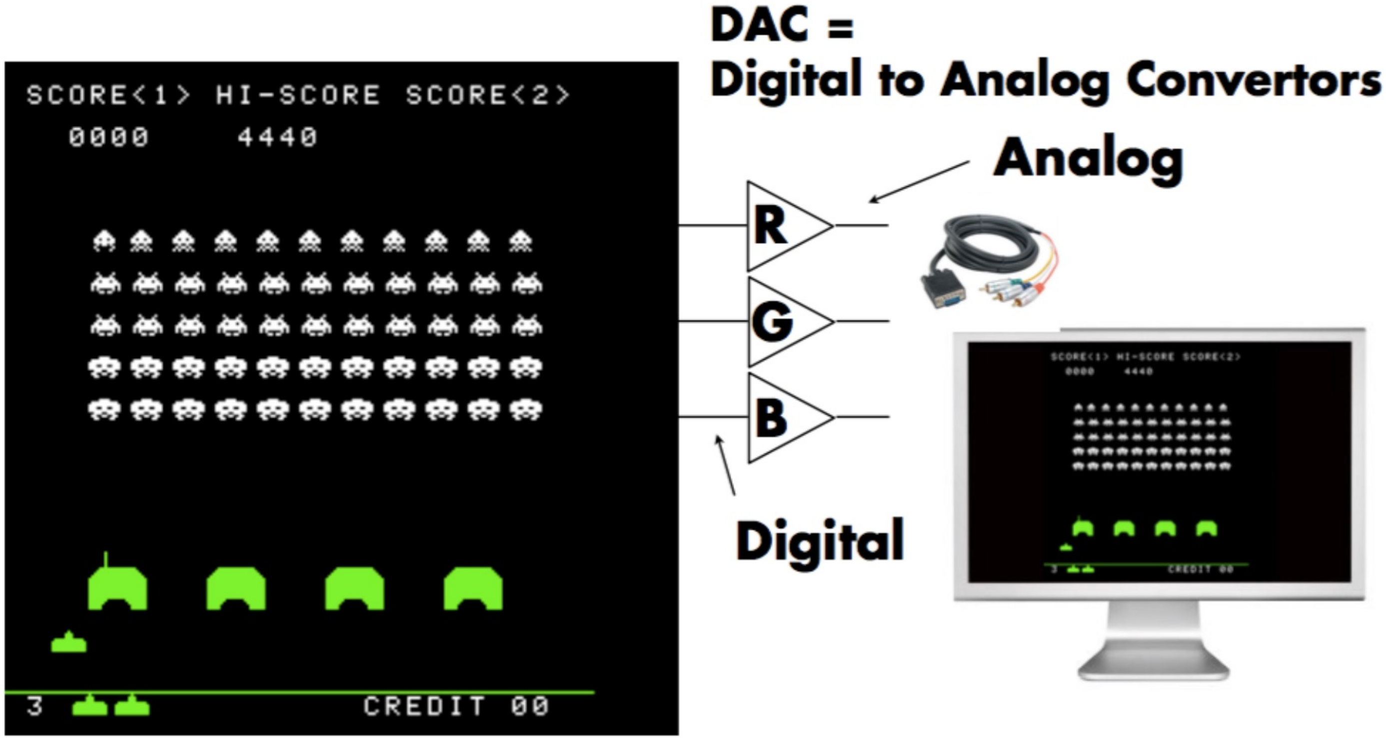


Image = 2D array of colors

Flat Panel Displays

Low-Res LCD Display

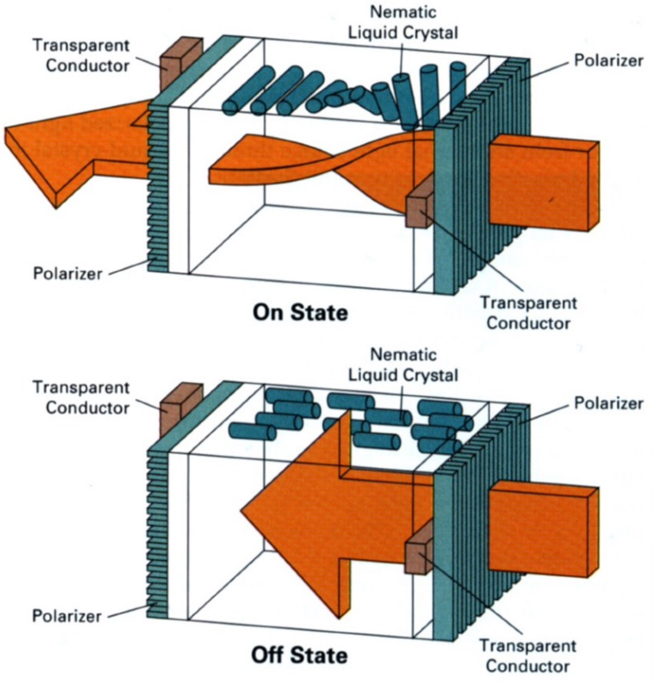
Color LCD, OLED, …

LCD (Liquid Crystal Display) Pixel

Principle: block or transmit light by twisting polarization

Illumination from backlight (e.g. fluorescent or LED)

Intermediate intensity levels by partial twist

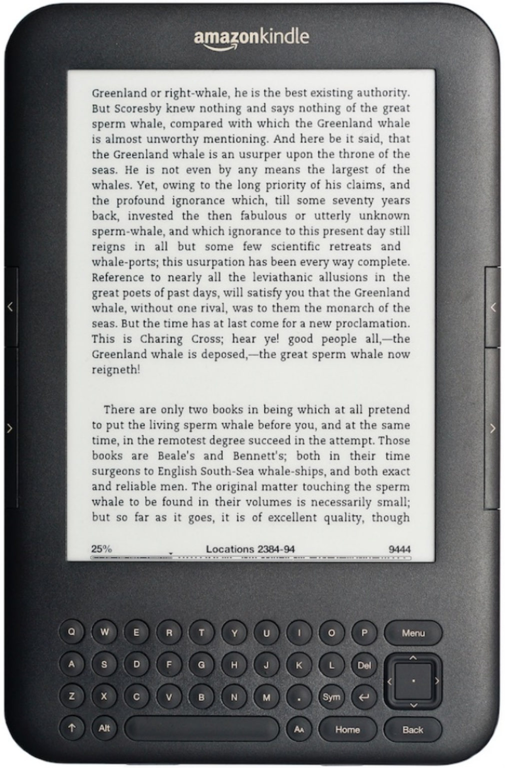


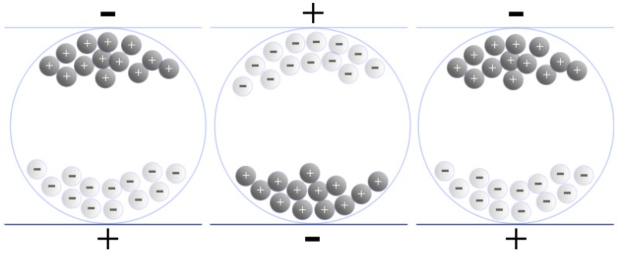
[H&B fig. 2-16]

LED Array Display

Light emitting diode array

Electrophoretic (Electronic Ink) Display



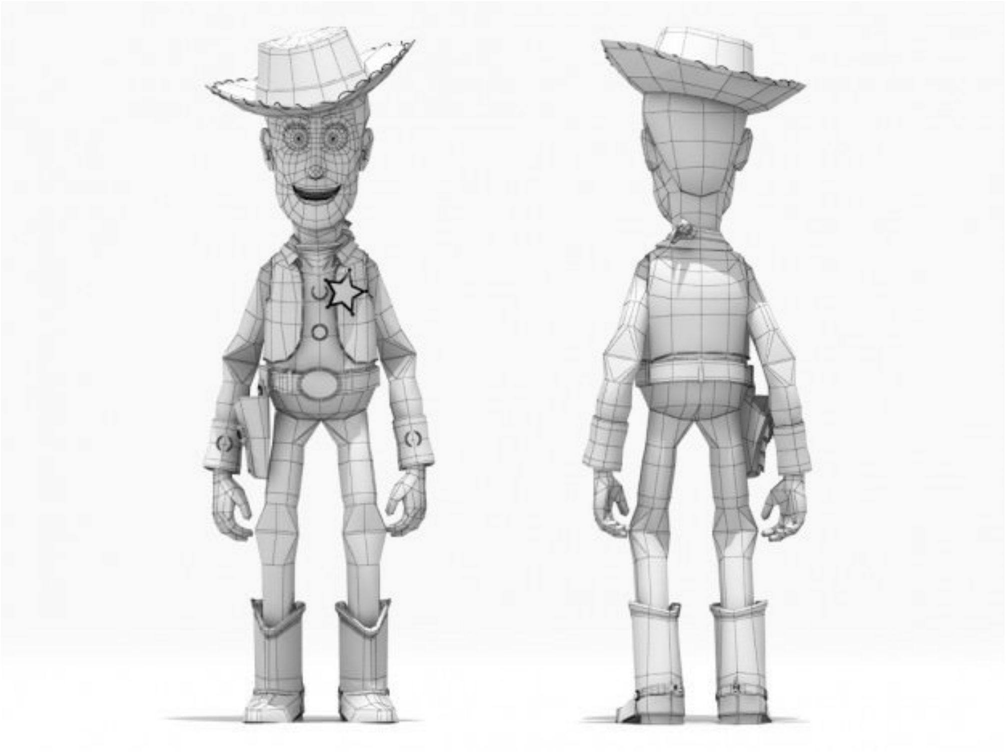


[Wikimedia Commons —Senarclens]

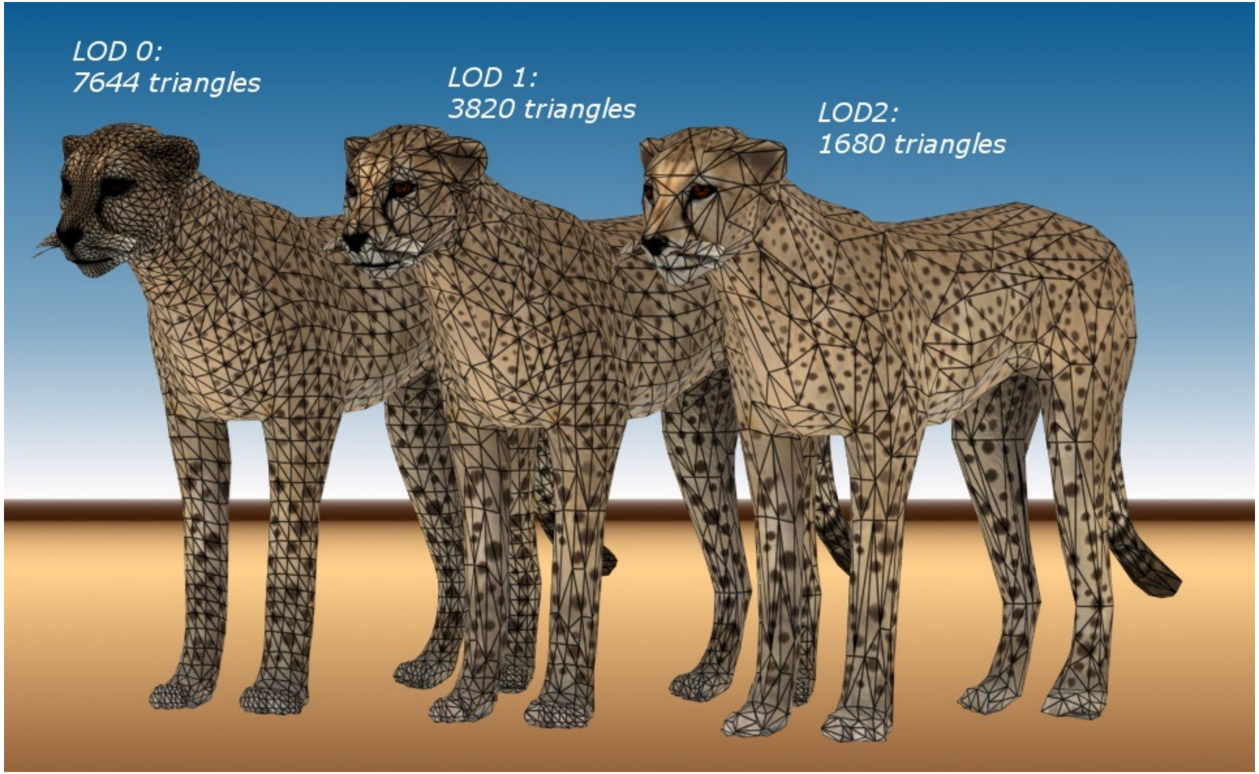
Rasterization:
Drawing to Raster Displays

Polygon Meshes

Polygon Meshes

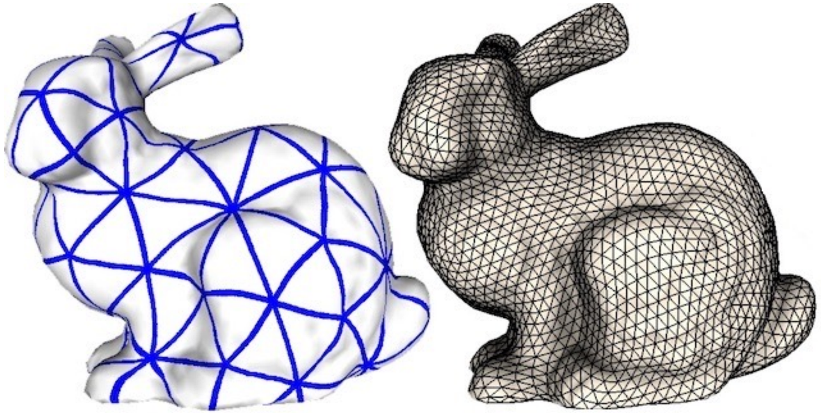


Triangle Meshes

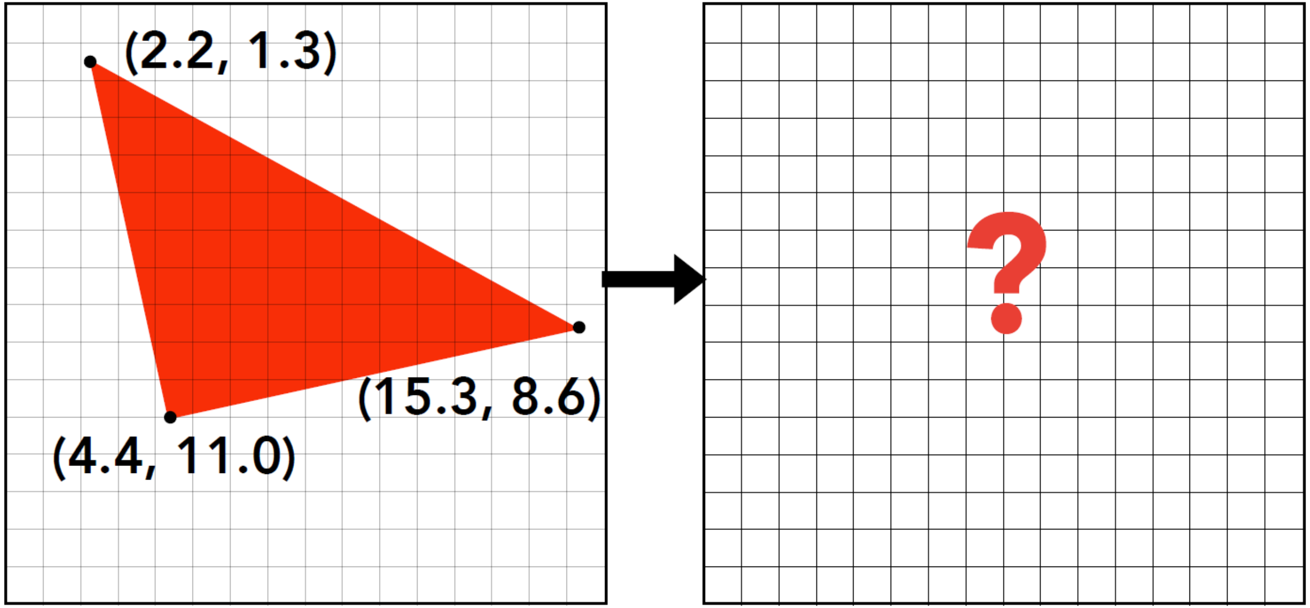


Triangles——Fundamental Area Primitive

- 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

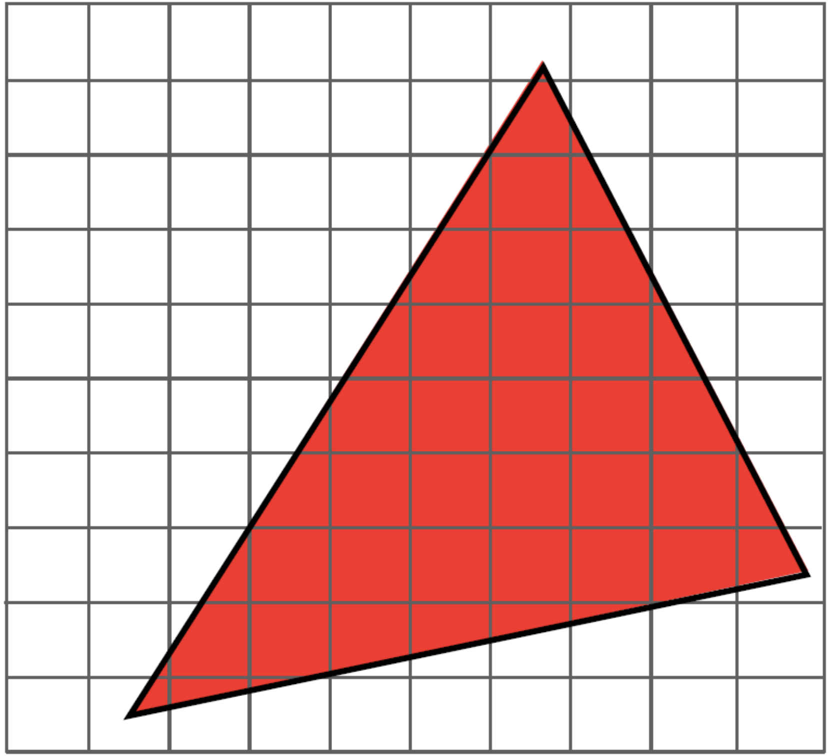
Sampling a Function

- Evaluating a function at a point is sampling.
- We can discretize a function by sampling

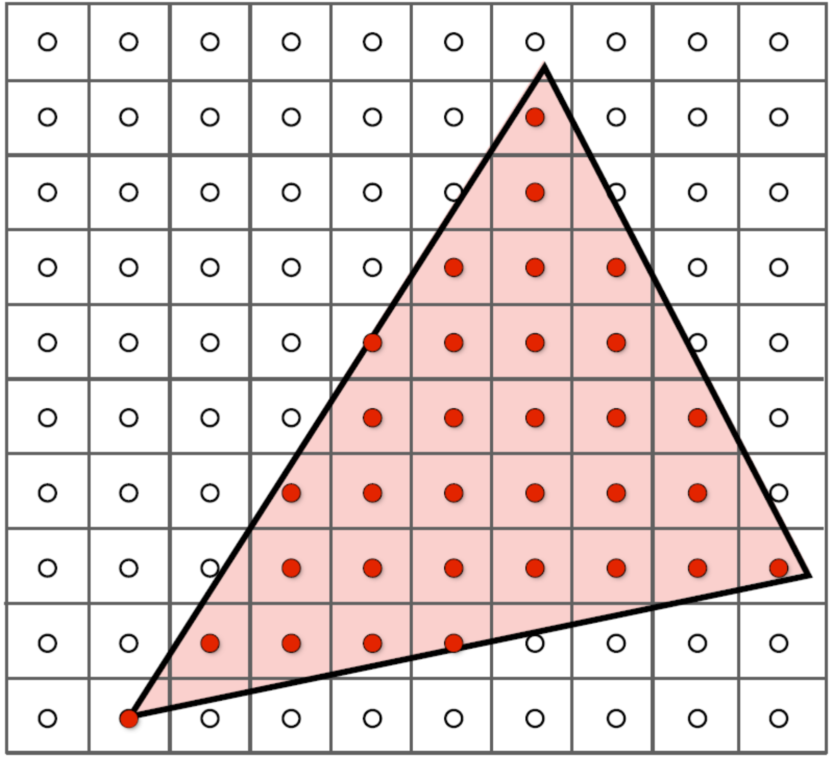
```
for (int x = 0; x < xmax; ++x)
    output[x] = f(x);
```

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

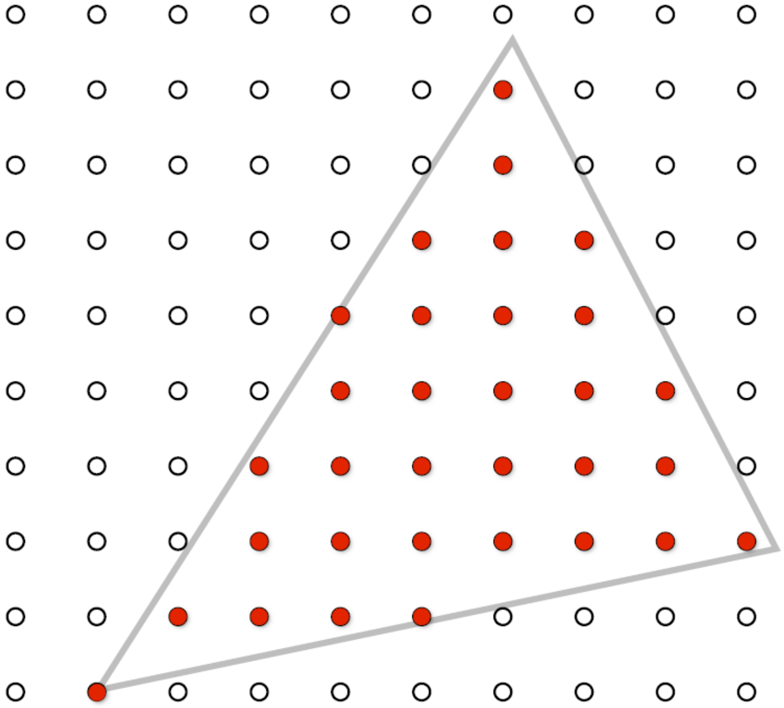
Rasterization As 2D Sampling



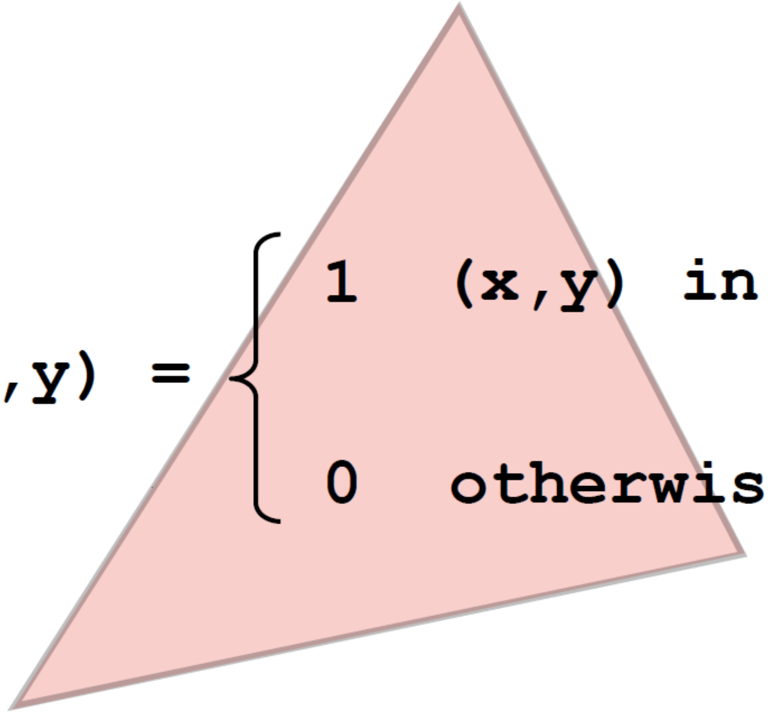
Sampling If Each Pixel Center is Inside Triangle



Sampling If Each Pixel Center is Inside Triangle



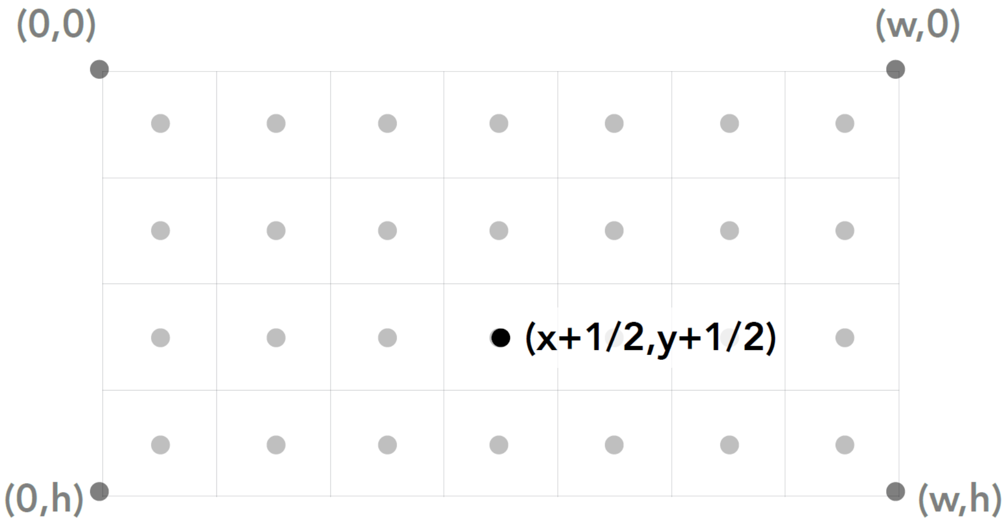
Define Binary Function: inside(tri,x,y)



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

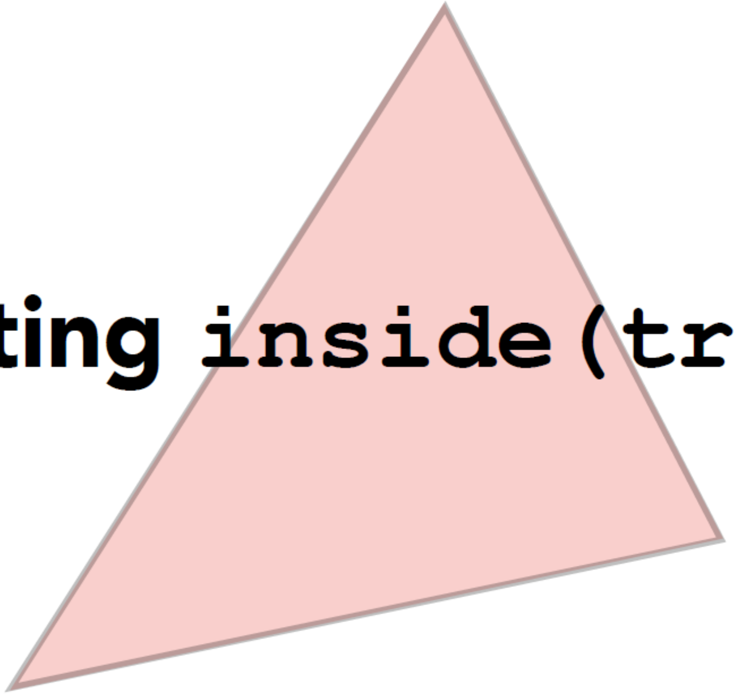
Rasterization=Sampling A 2D Indicator Function

```
for( int x = 0; x < xmax; x++ )
  for( int y = 0; y < ymax; y++ )
    Image[x][y] = f(x + 0.5, y + 0.5);
```



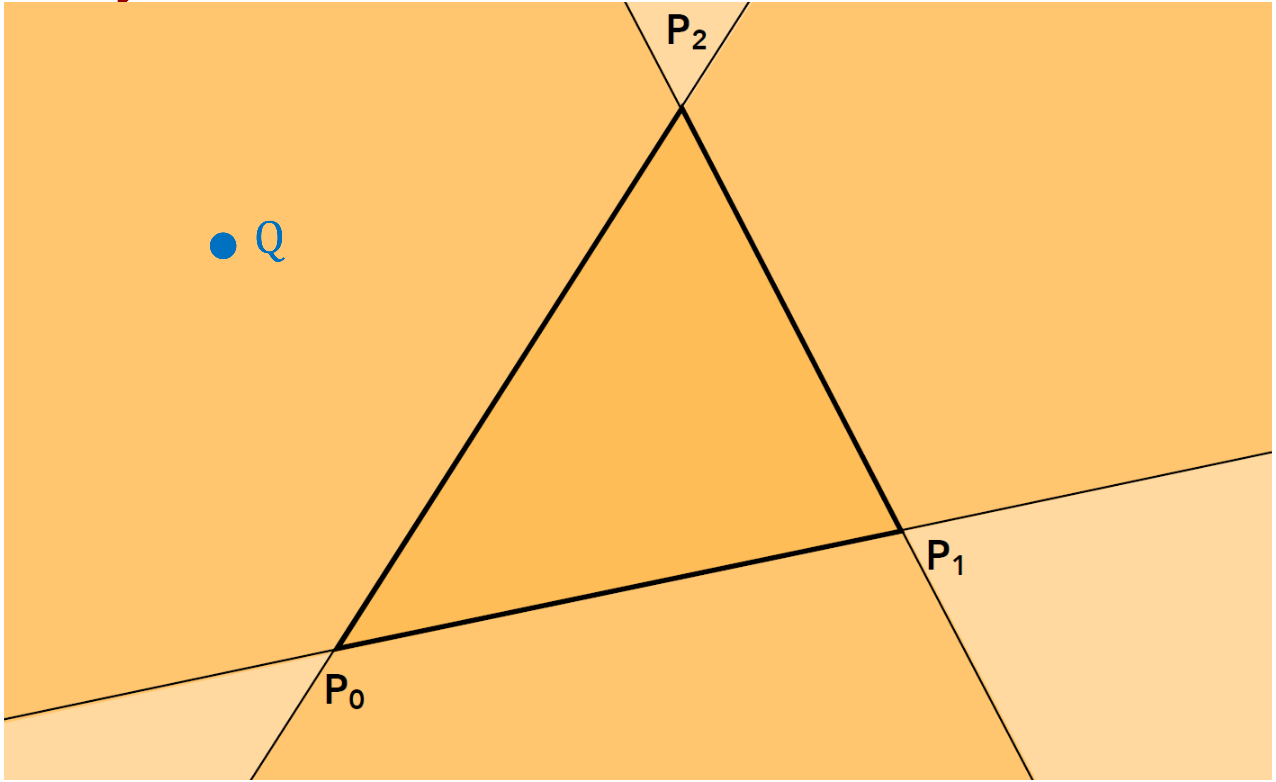
Sample location for pixel (x,y)

Rasterization=Sampling A 2D Indicator Function

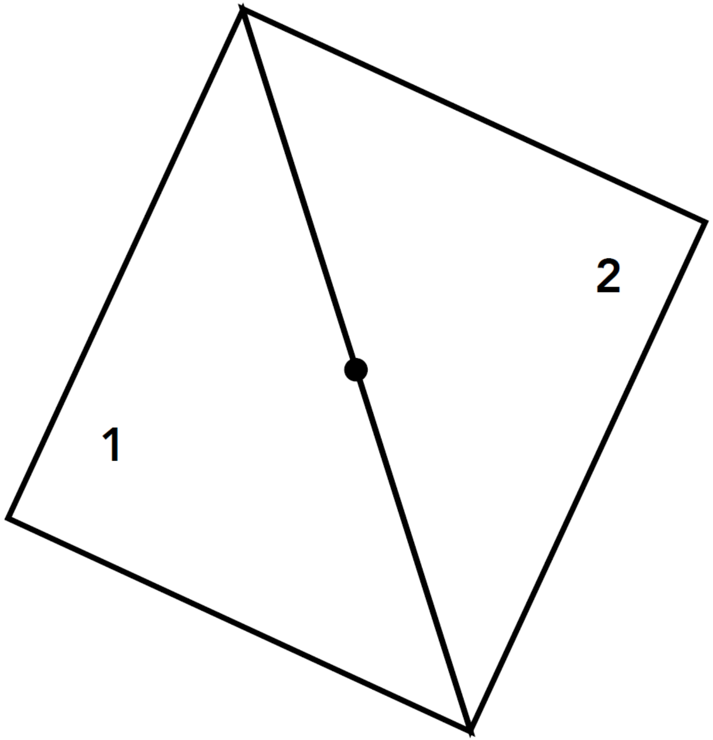


Evaluating `inside(tri,x,y)`

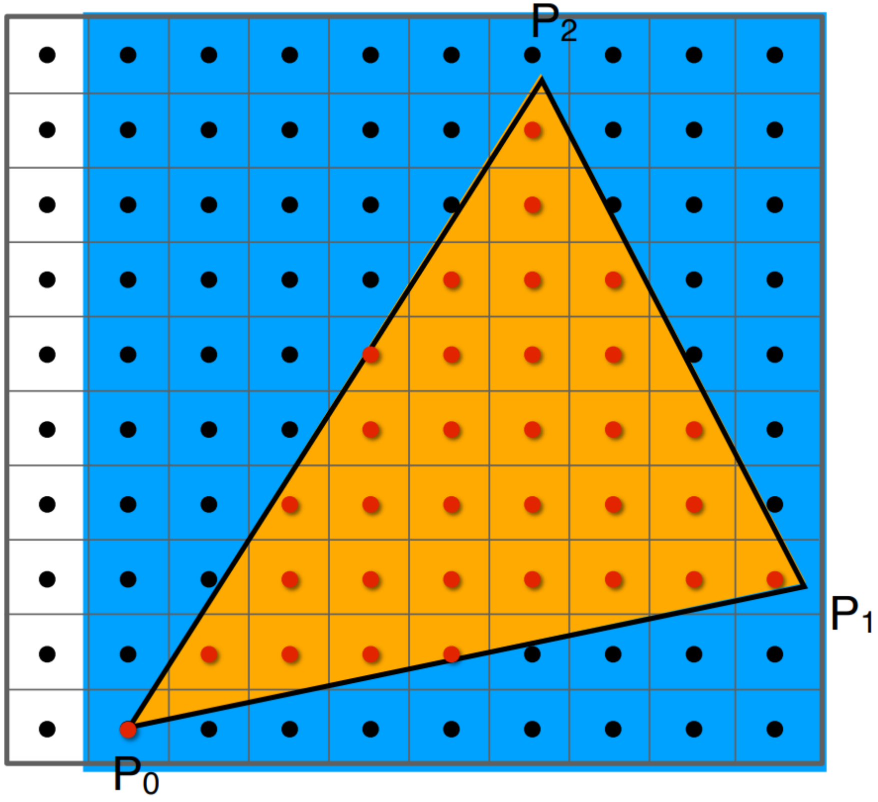
Inside? Try——Three Cross Products!



Edge Cases

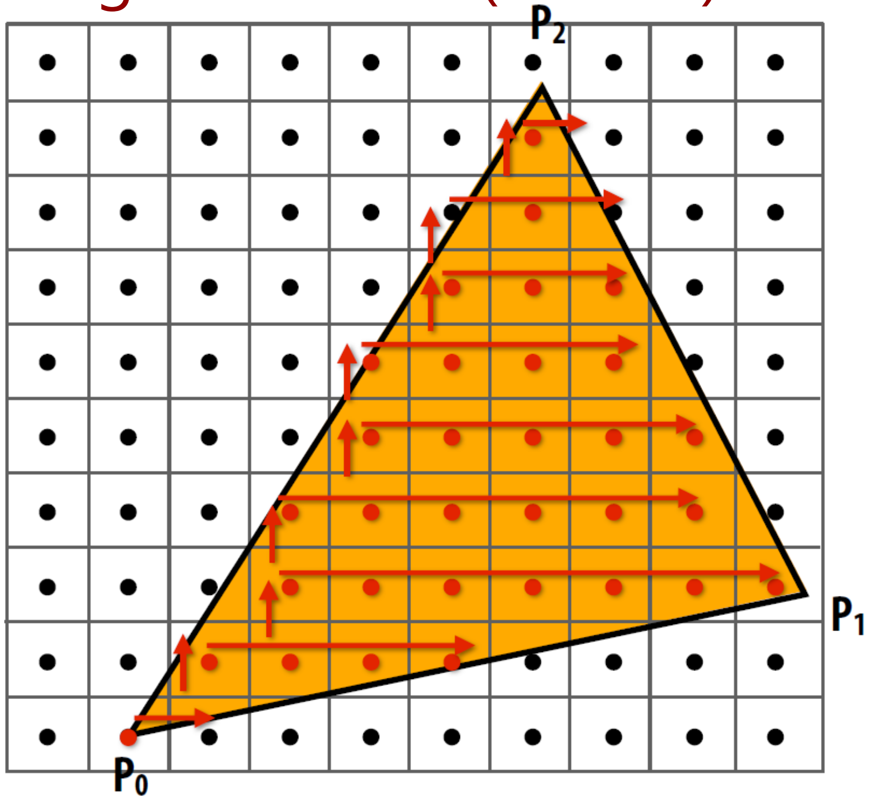


Checking All Pixels on the Screen?



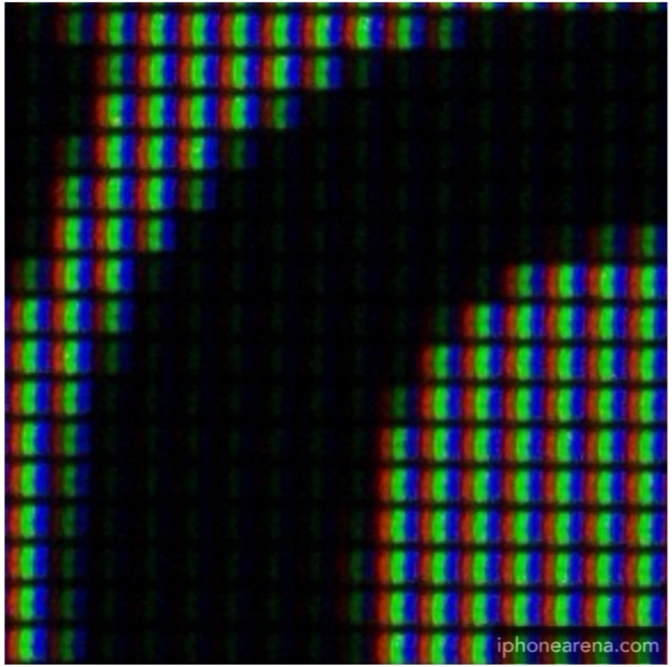
Use a **Bounding Box!**

Incremental Triangle Traversal (Faster?)

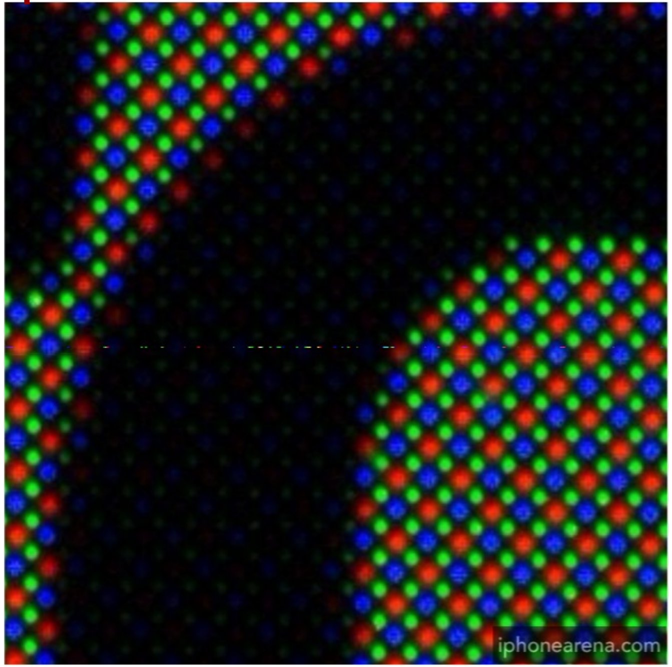


Rasterization on Real Displays

Real LCD Screen Pixels(Closeup)



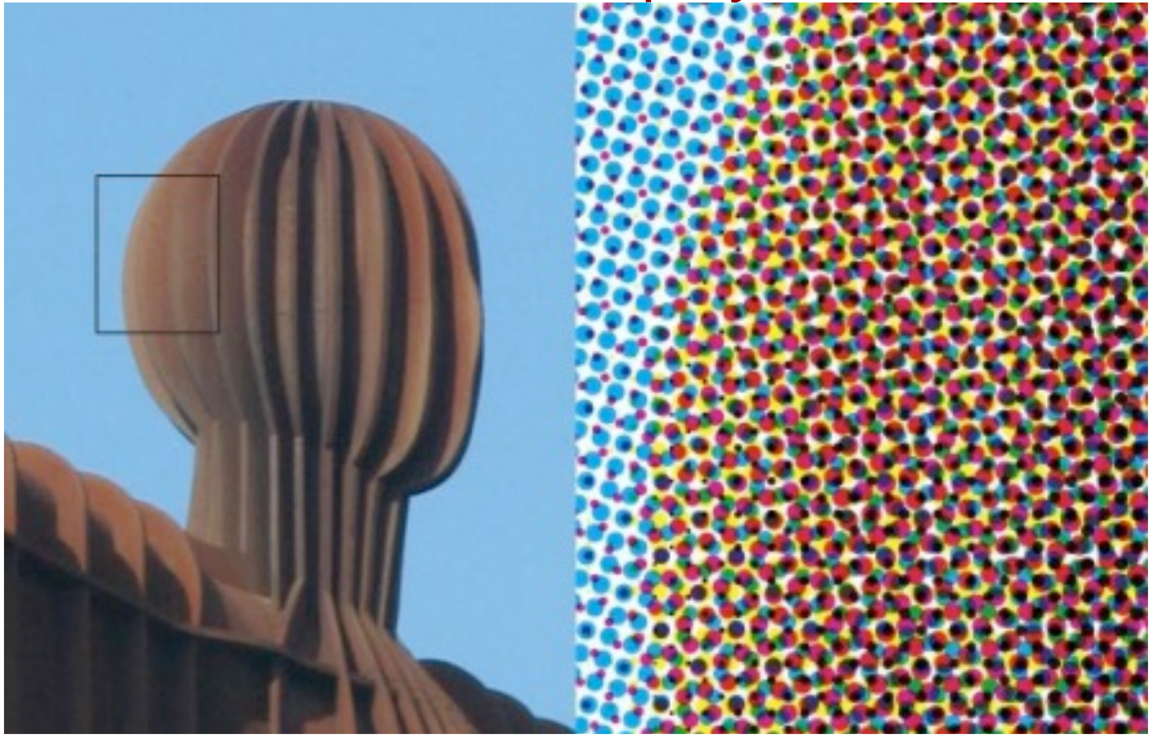
iPhone 6S



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?

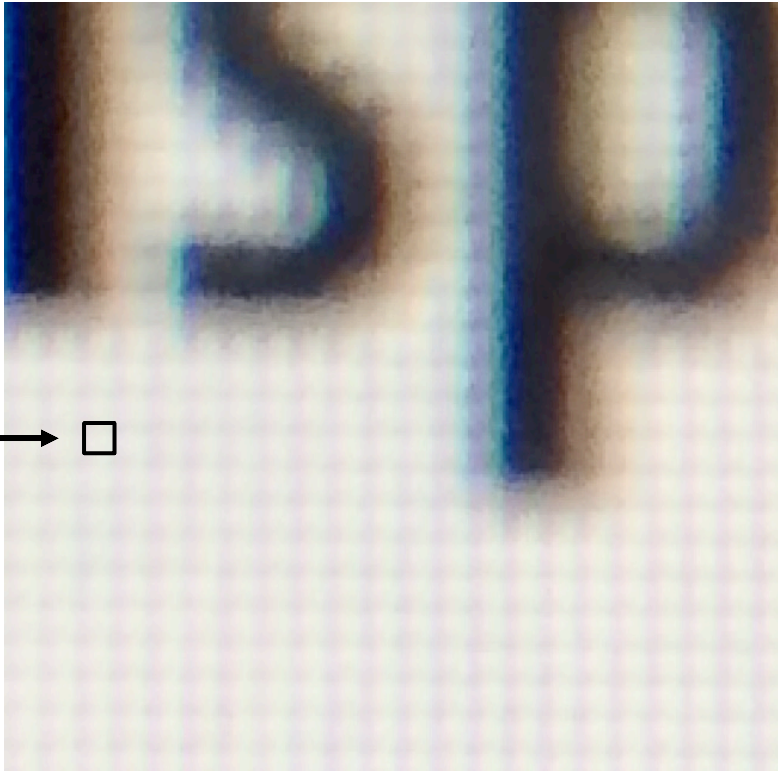


Color print: observe half-tone pattern

Assume Display Pixels Emit Square of Light

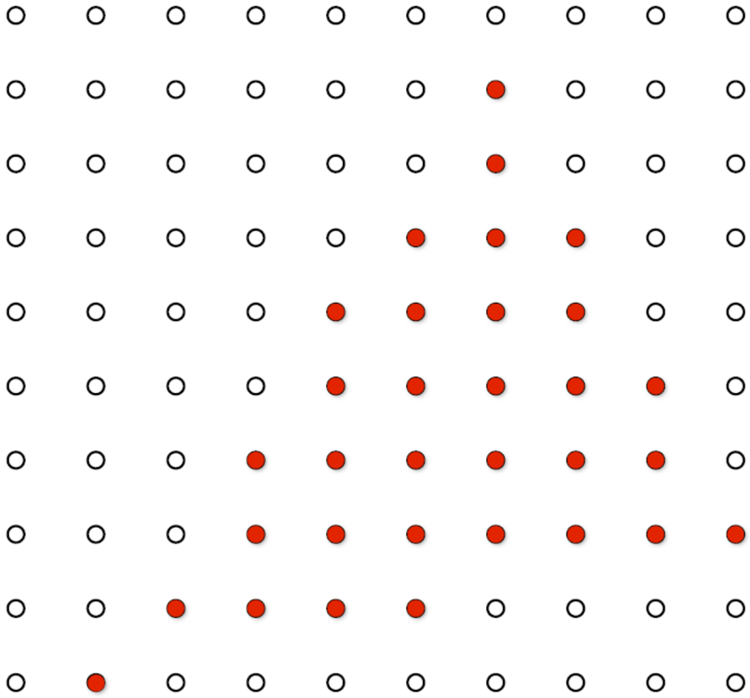
- Each image sample sent to the display is converted into a little square of light of the appropriate color: (a pixel = picture element)

LCD pixel on laptop

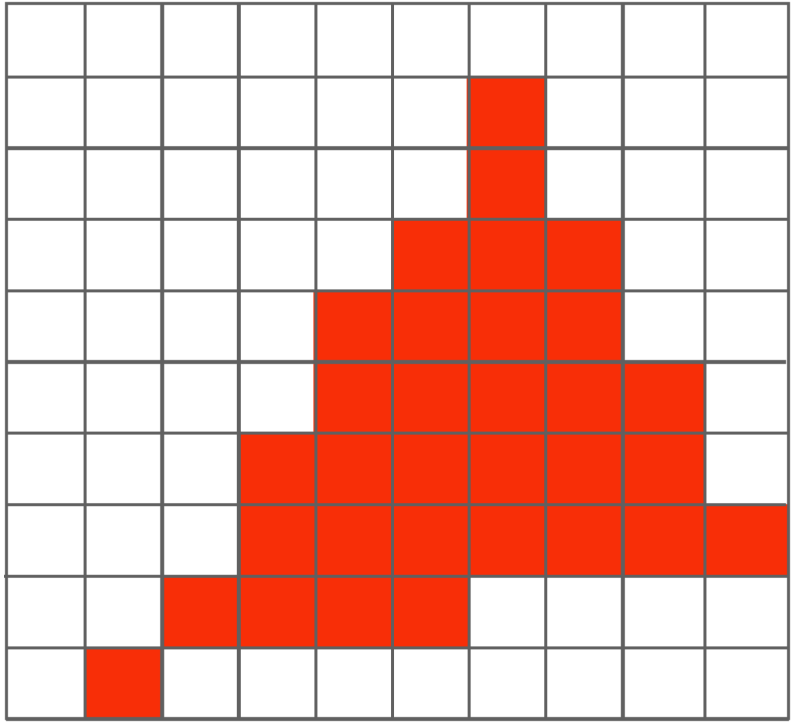


- * LCD pixels do not actually emit light in a square of uniform color, but this approximation suffices for our current discussion

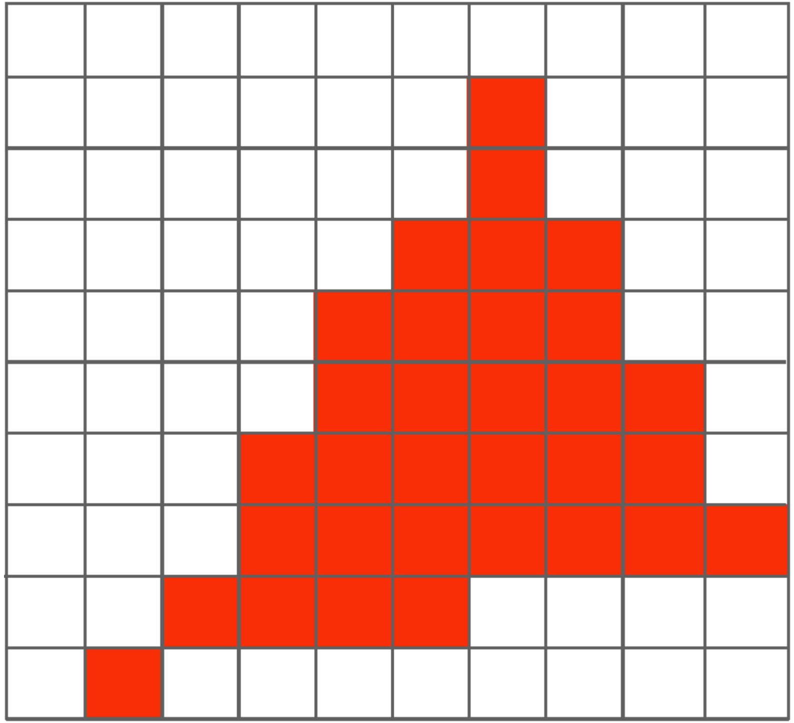
So, If We Send The Display This Sampled Signal

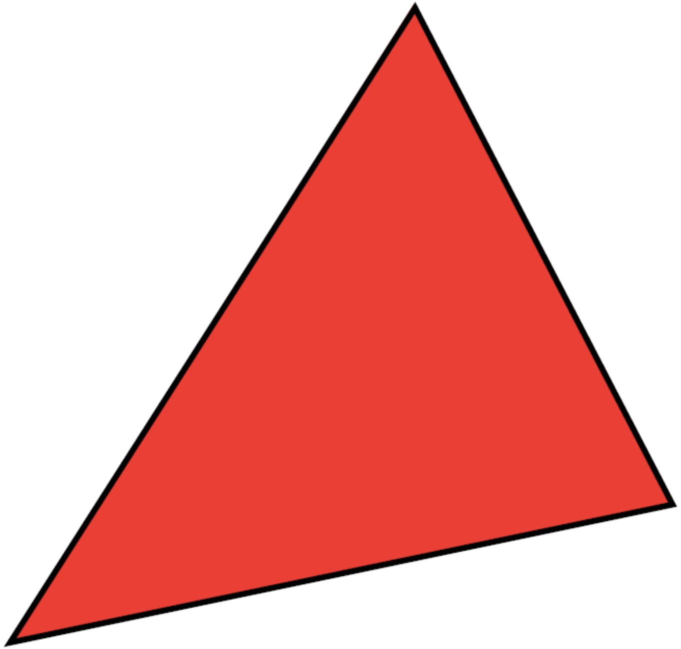


The Display Physically Emits This Signal

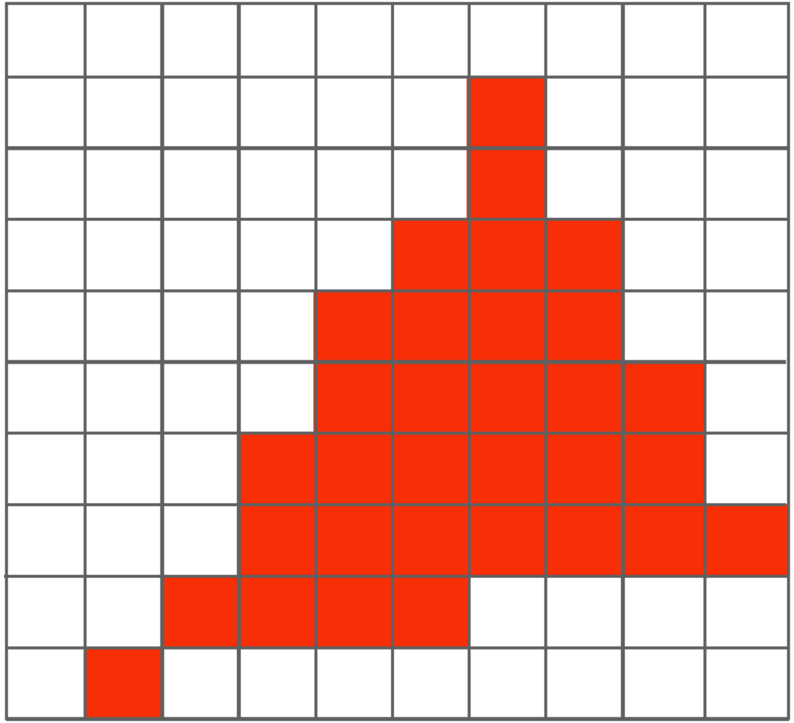


Compare: The Continuous Triangle Function





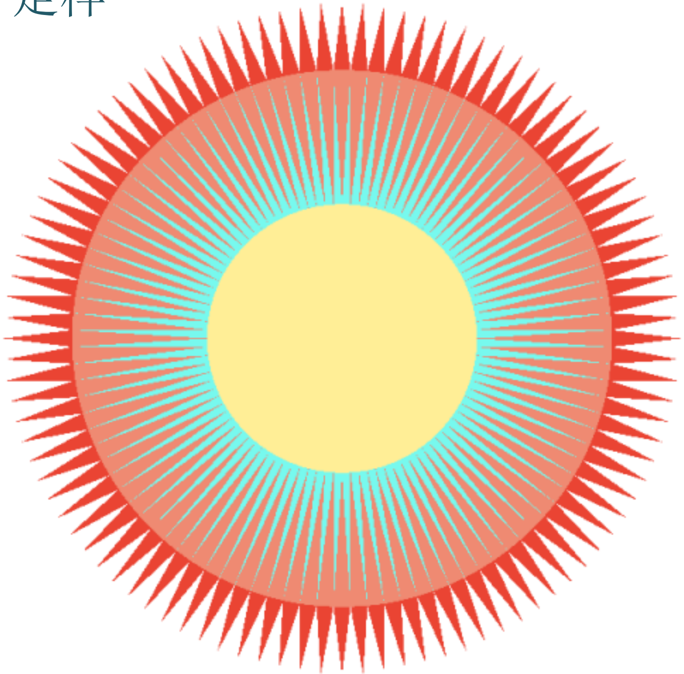
What's Wrong With This Picture?

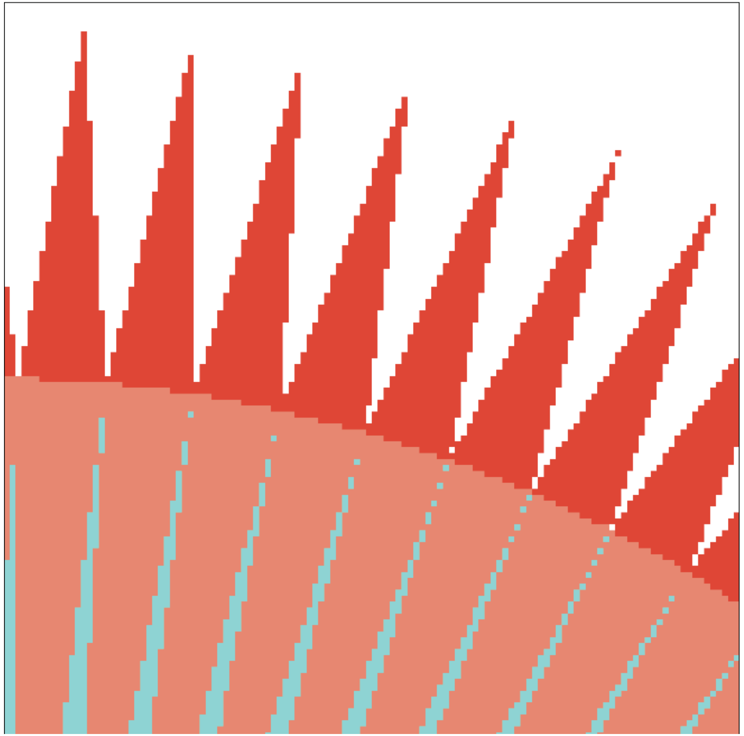


Jaggies!

Aliasing (Jaggies)

走样



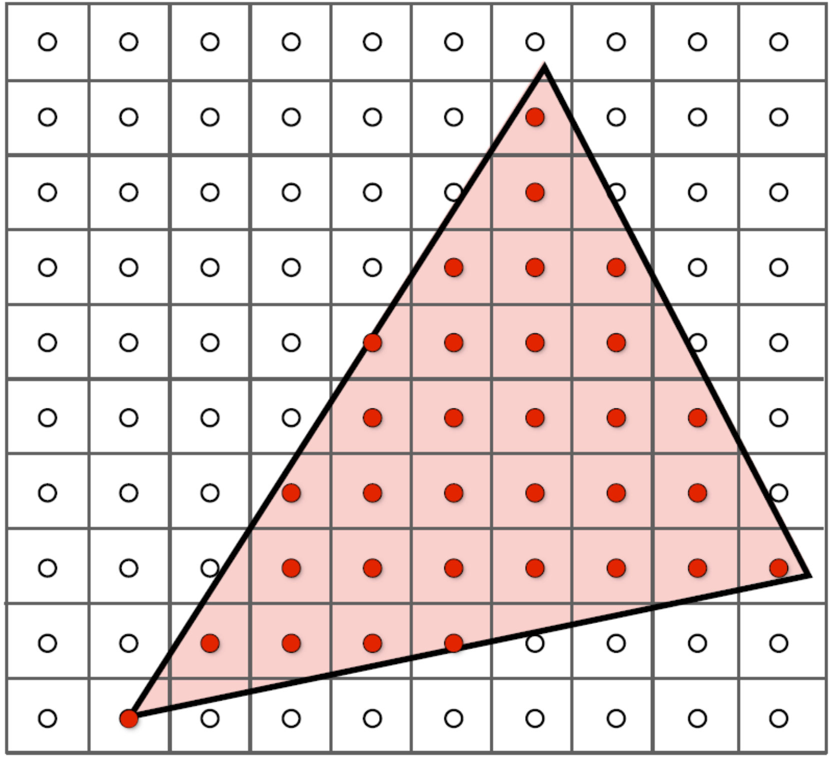


Today’s Topic

- Rasterization
  - Rasterizing Triangles into Pixels
  - Aliasing and Antialiasing
  - Sampling Theory

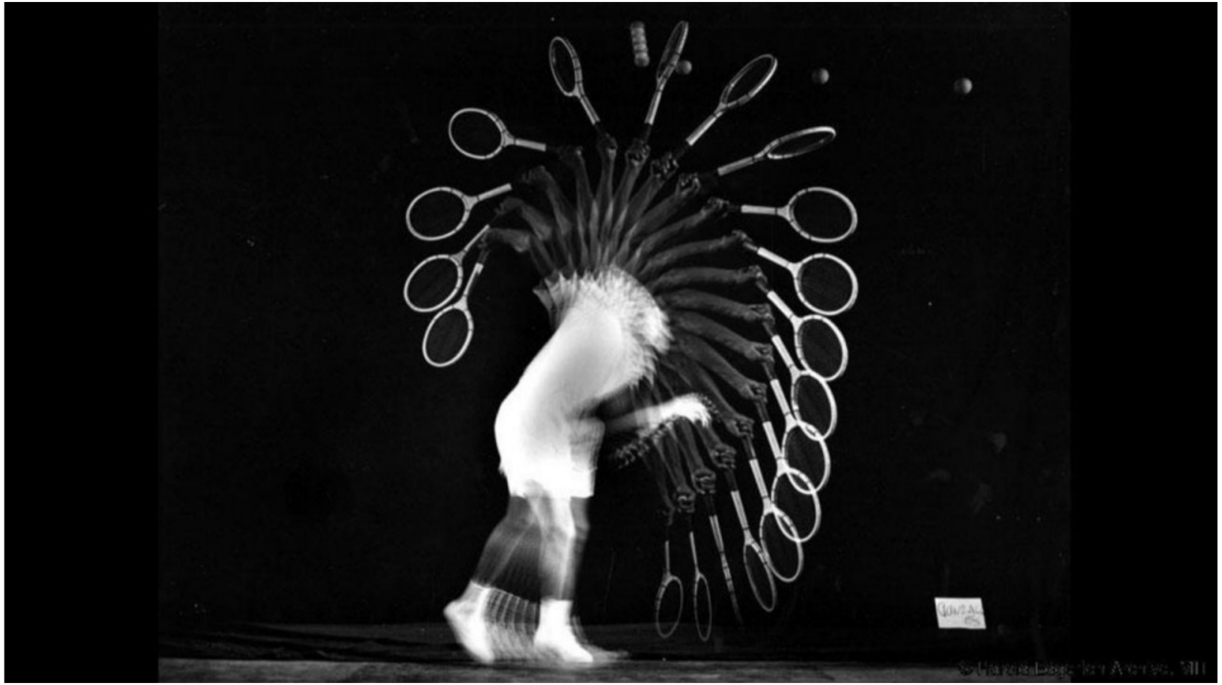
Sampling is Ubiquitous in Computer Graphics

Rasterization = Sampling 2D Positions



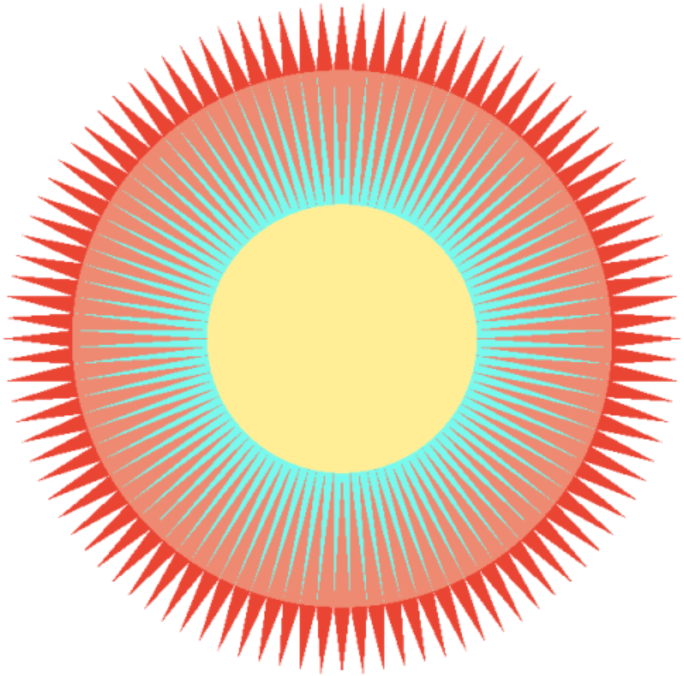
Photograph = Sample Image Sensor Plane

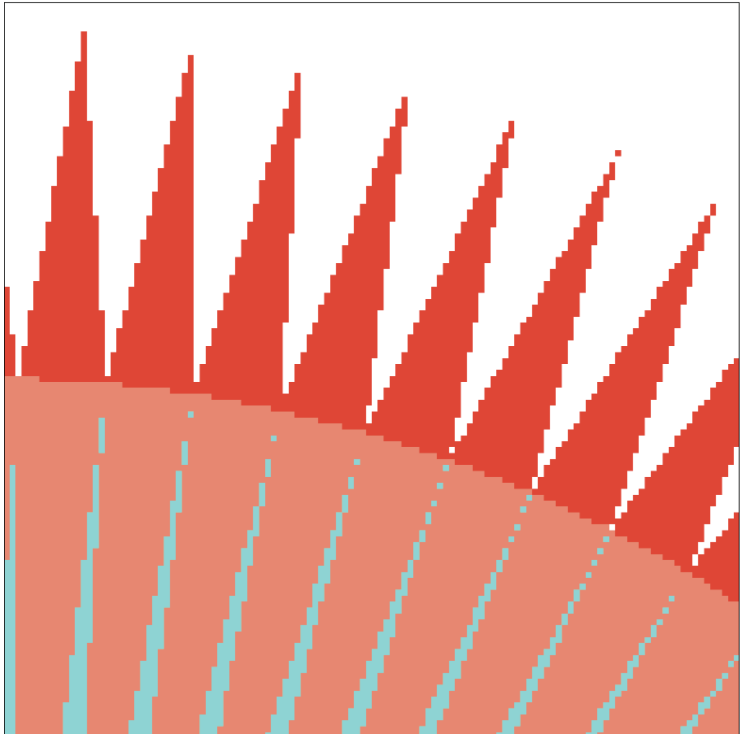
Video = Sample Time



Sampling Artifacts (Errors/Mistakes/Inaccuracies) in Computer Graphics

Jaggies (Staircase Pattern)





This is also an example of “aliasing” – a sampling error

Moiré Patterns in Imaging (Sampling Example)

Read every sensor pixel

Skip odd rows and columns

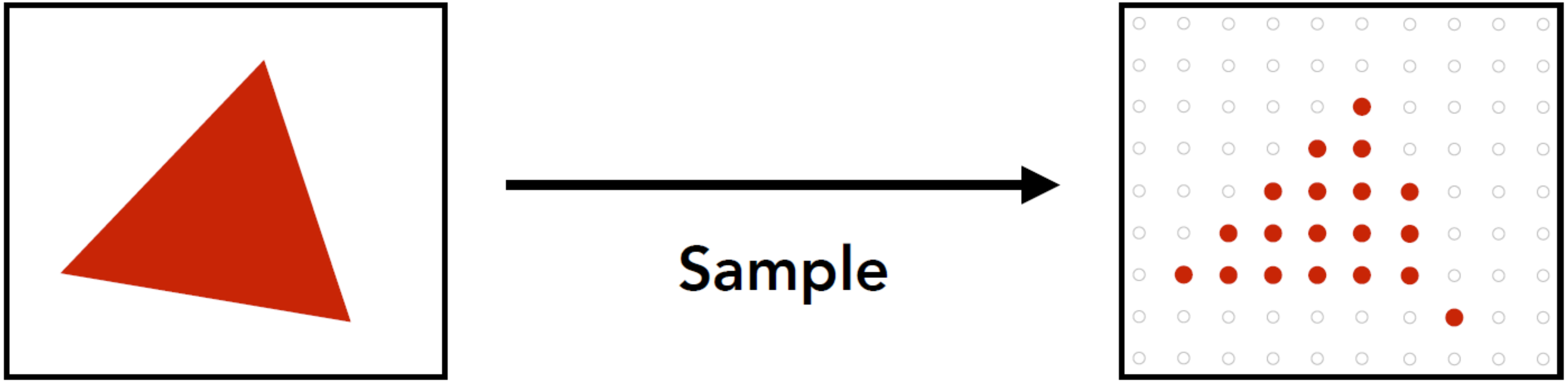
Wagon Wheel Illusion (False Motion) Video

Sampling Artifacts in Computer Graphics

- Artifacts due to (under)sampling - “Aliasing”
  - Jaggies – sampling in space
  - Moiré – undersampling images (and texture maps)
  - Wagon wheel effect – sampling in time
  - [Many more]…
- Behinds the Aliasing Artifacts
  - Signals are changing too fast(high frequency) but sampled too slowly

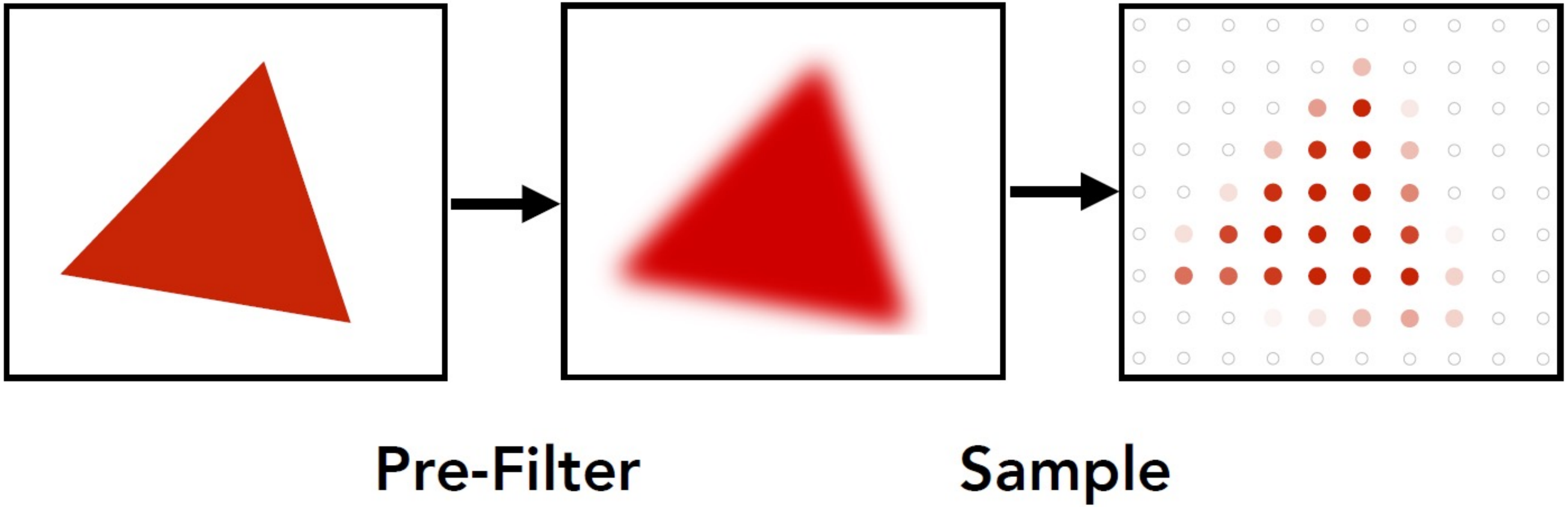
Antialiasing Idea:
Blurring (Pre-Filtering) Before Sampling

Regular Point Sampling in Space



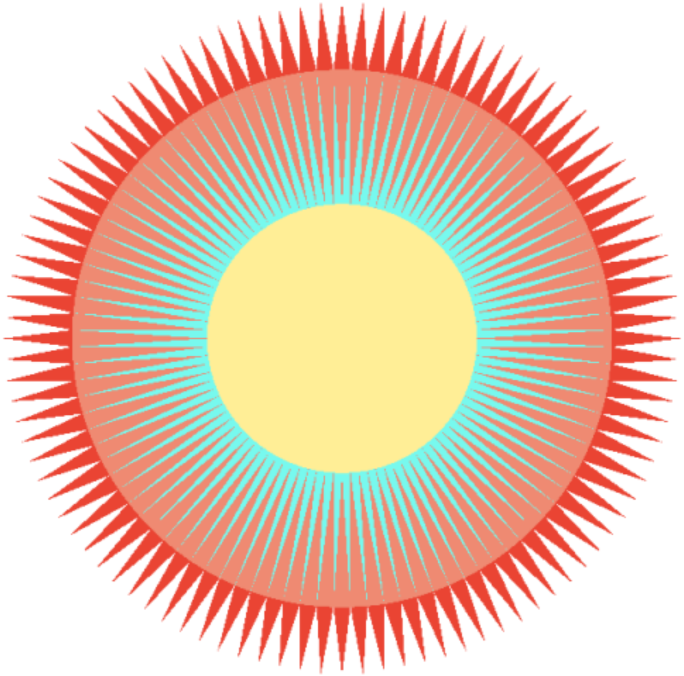
Note jaggies in rasterized triangle
where pixel values are pure red or white

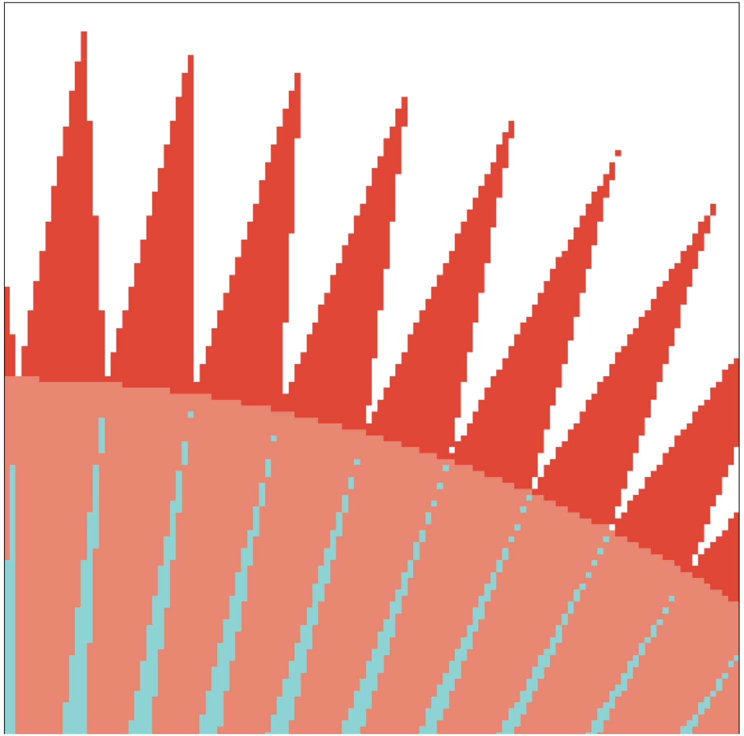
Antialiased Sampling



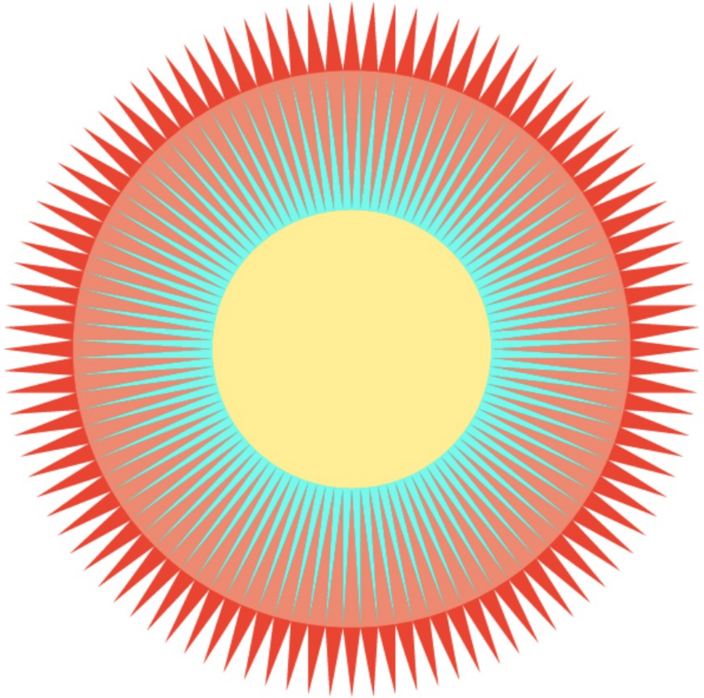
Note antialiased edges in rasterized triangle
where pixel values take intermediate values

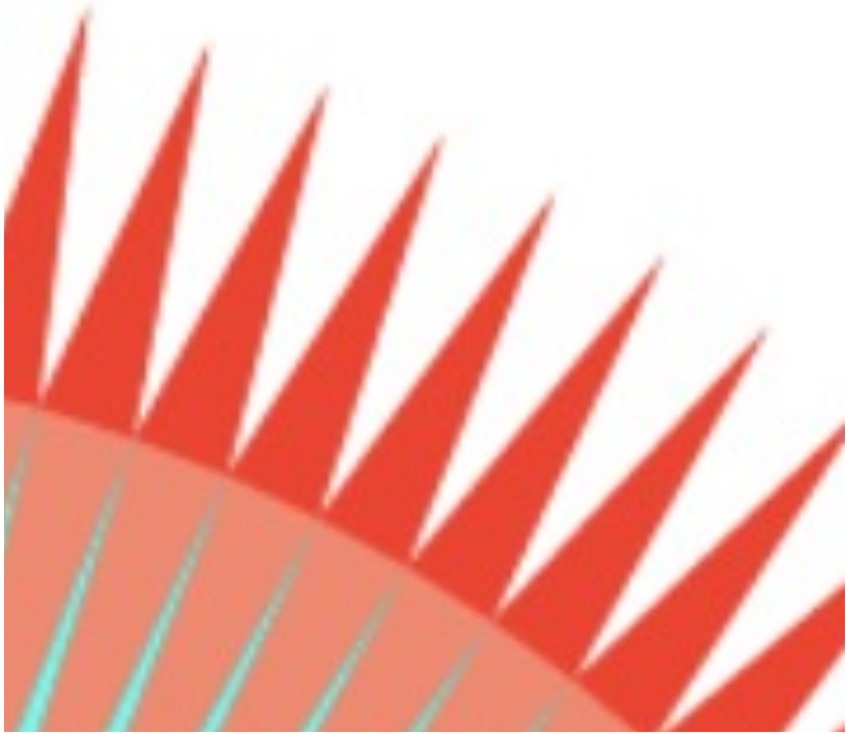
Point Sampling



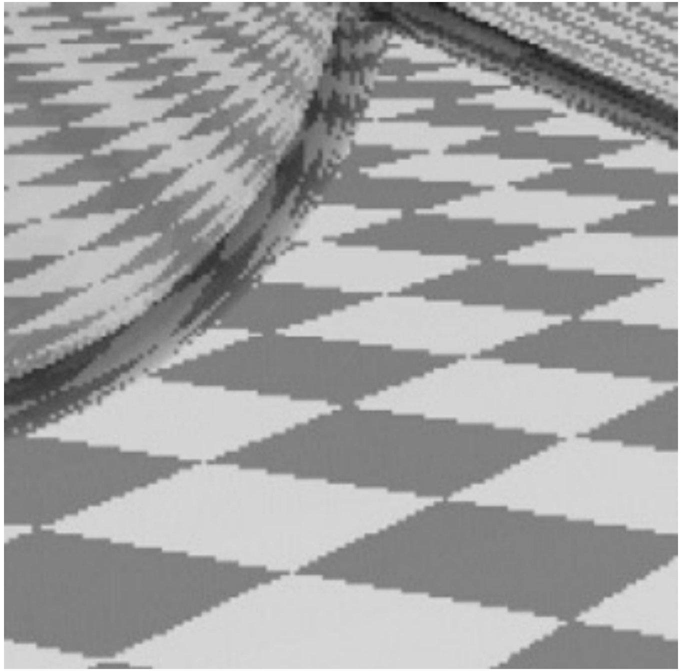


Antialiasing

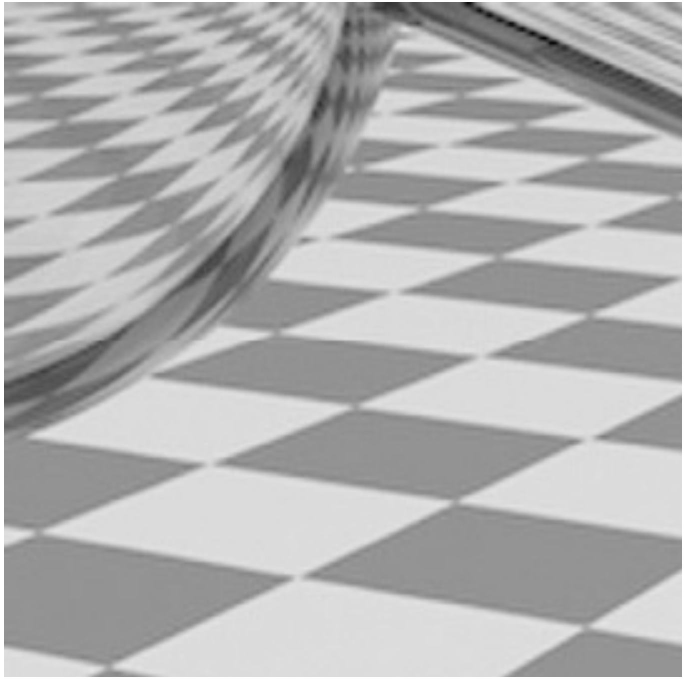




Point Sampling vs Antialiasing

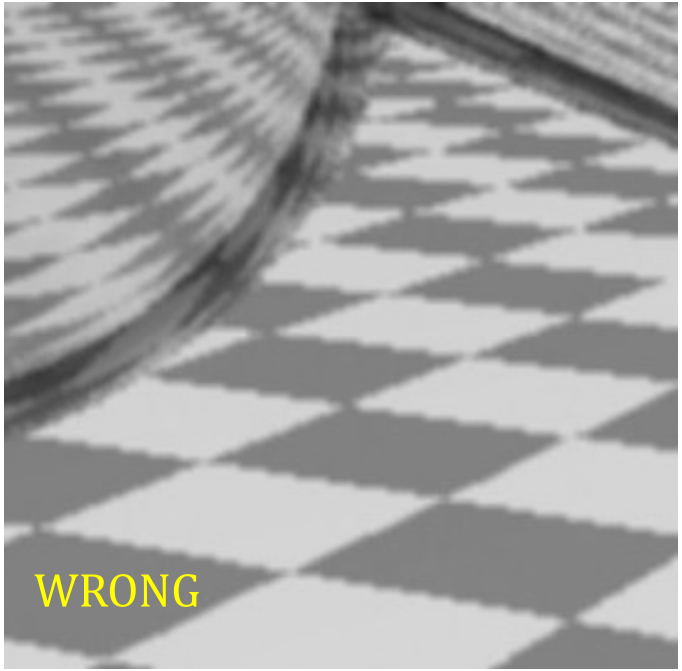


Jaggies

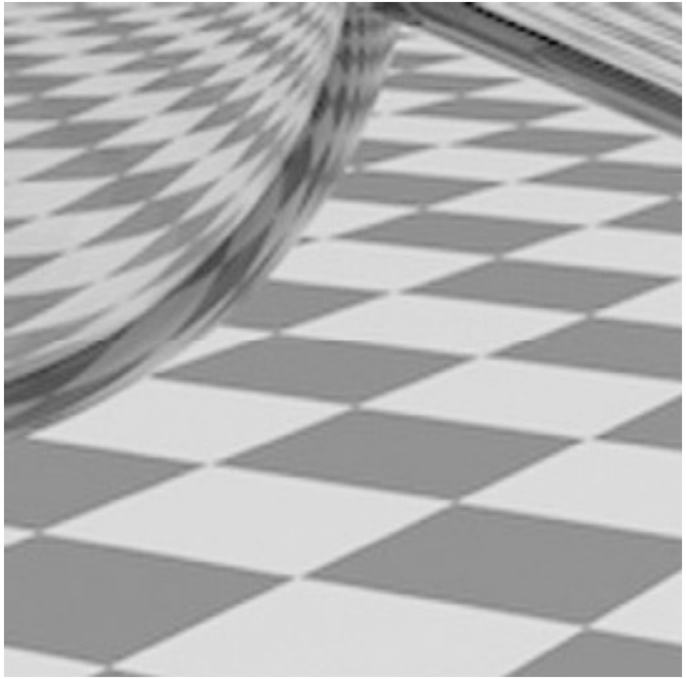


Pre-Filtered

Antialiasing vs Blurred Aliasing



Blurred Jaggies (Sample then filter)



Pre-Filtered(Filter then sample)

But Why???

- 1. Why undersampling introduces aliasing?
- 2. Why pre-filtering then sampling can do antialiasing?

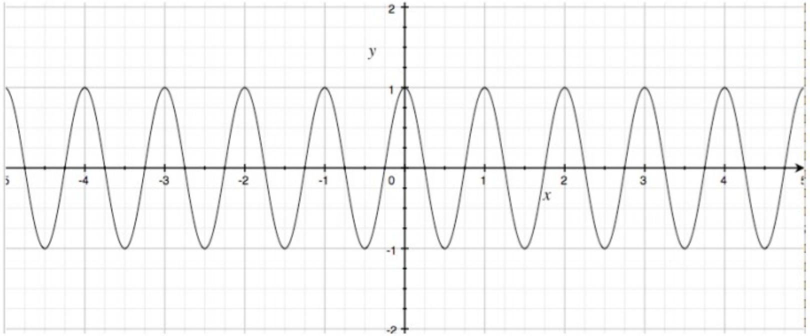
Let's dig into fundamental reasons And look at how to implement antialiased rasterization

Today's Topic

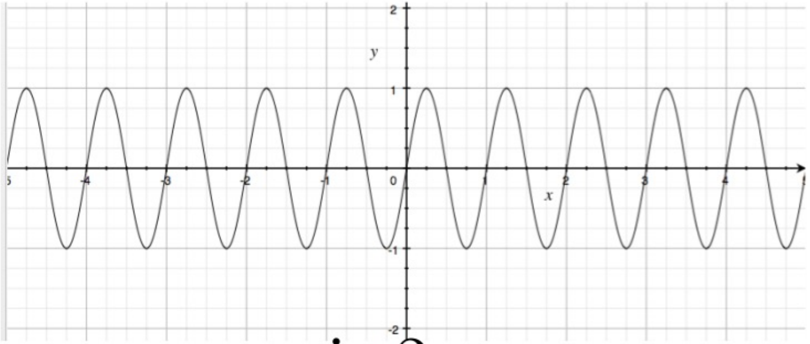
- Rasterization
  - Rasterizing Triangles into Pixels
  - Aliasing and Antialiasing
  - Sampling Theory

Frequency Domain

Sines and Cosines



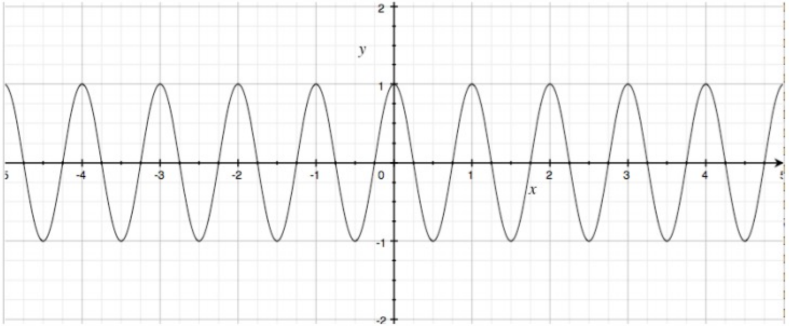
$\cos 2\pi x$



$\sin 2\pi x$

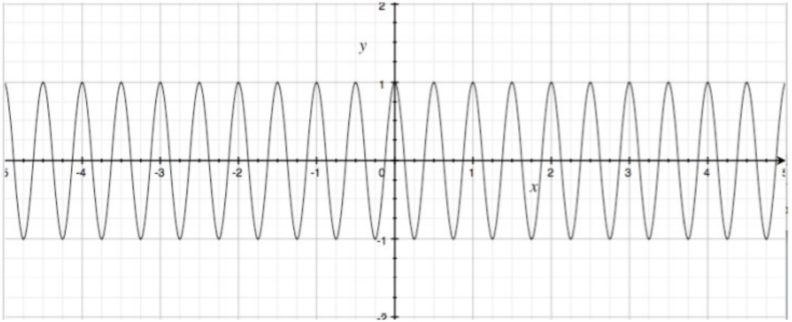
Frequencies $\cos 2\pi f x$

$$f = \frac{1}{T}$$



$f = 1$

$\cos 2\pi x$



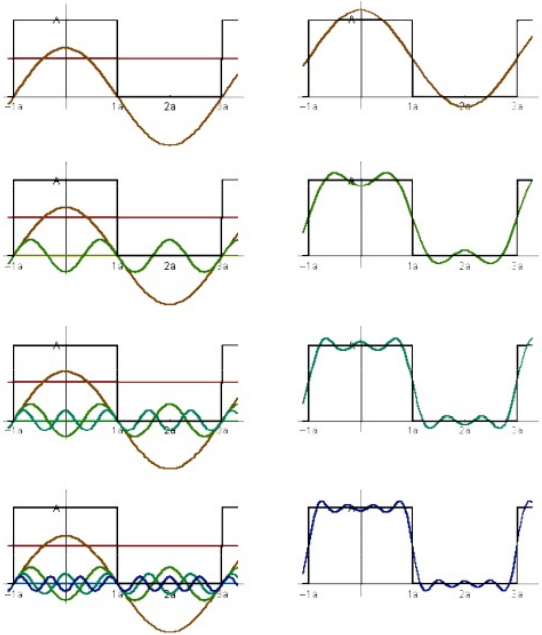
$f = 2$

$\cos 4\pi x$

Fourier Transform

Represent a function as a weighted sum of sines and cosines

Joseph Fourier 1768 - 1830



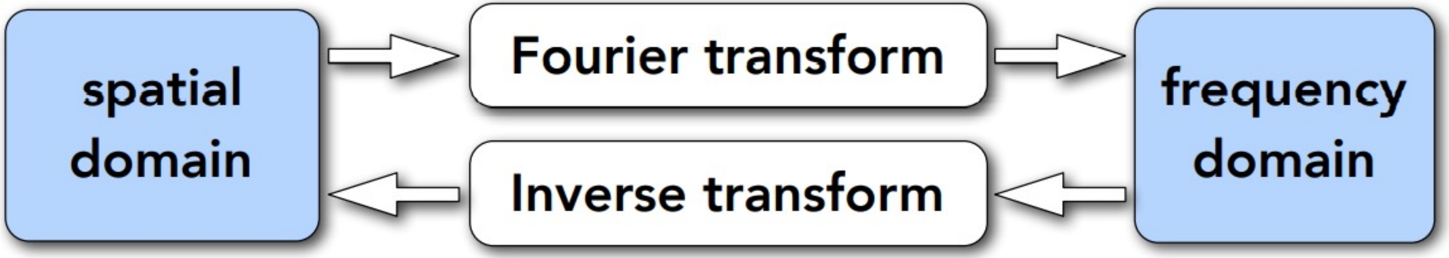
$$f(x) = \frac{A}{2} + \frac{2A\cos(t\omega)}{\pi} - \frac{2A\cos(3t\omega)}{3\pi} + \frac{2A\cos(5t\omega)}{5\pi} - \frac{2A\cos(7t\omega)}{7\pi} + \cdots$$

Fourier Transform Decomposes A Signal Into Frequencies

$f(x)$

$$F(\omega) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i \omega x} dx$$

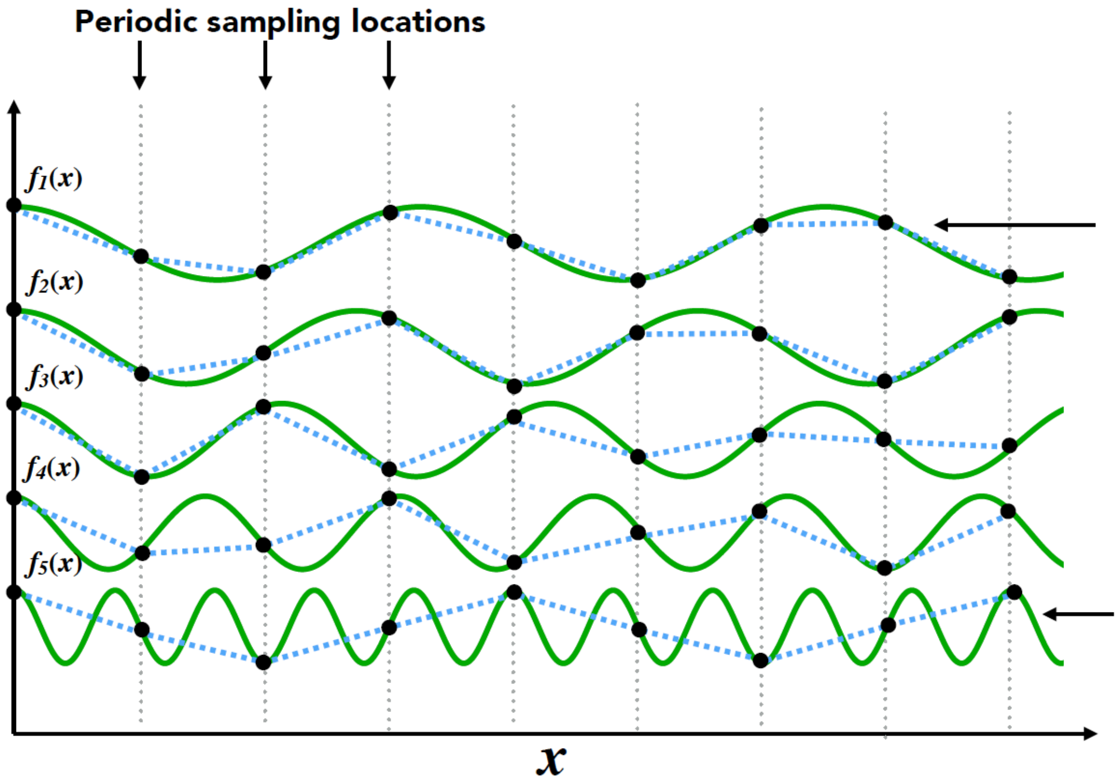
$F(\omega)$



$$f(x) = \int_{-\infty}^{\infty} F(\omega) e^{2\pi i \omega x} d\omega$$

Recall $e^{ix} = \cos x + i \sin x$

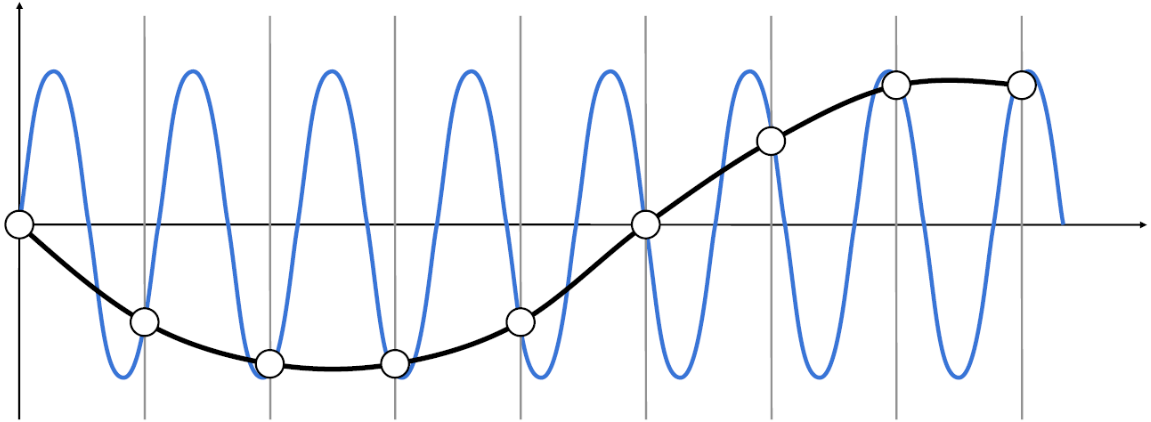
Higher Frequencies Need Faster Sampling



Low-frequency signal: sampled adequately for reasonable reconstruction

High-frequency signal is insufficiently sampled: reconstruction incorrectly appears to be from a low frequency signal

Undersampling Creates Frequency Aliases

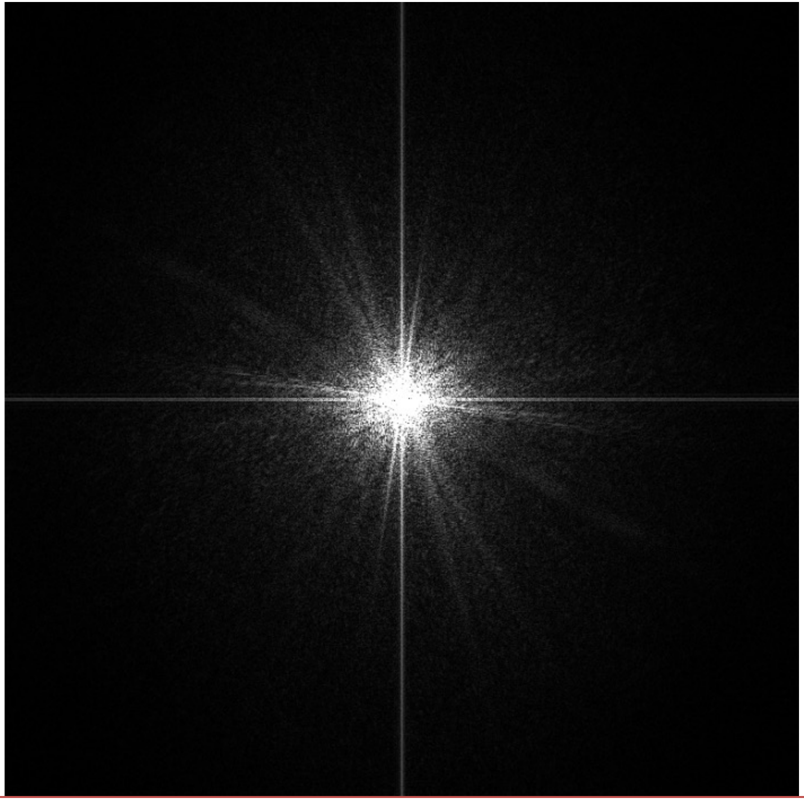


High-frequency signal is insufficiently sampled: samples erroneously appear to be from a low-frequency signal

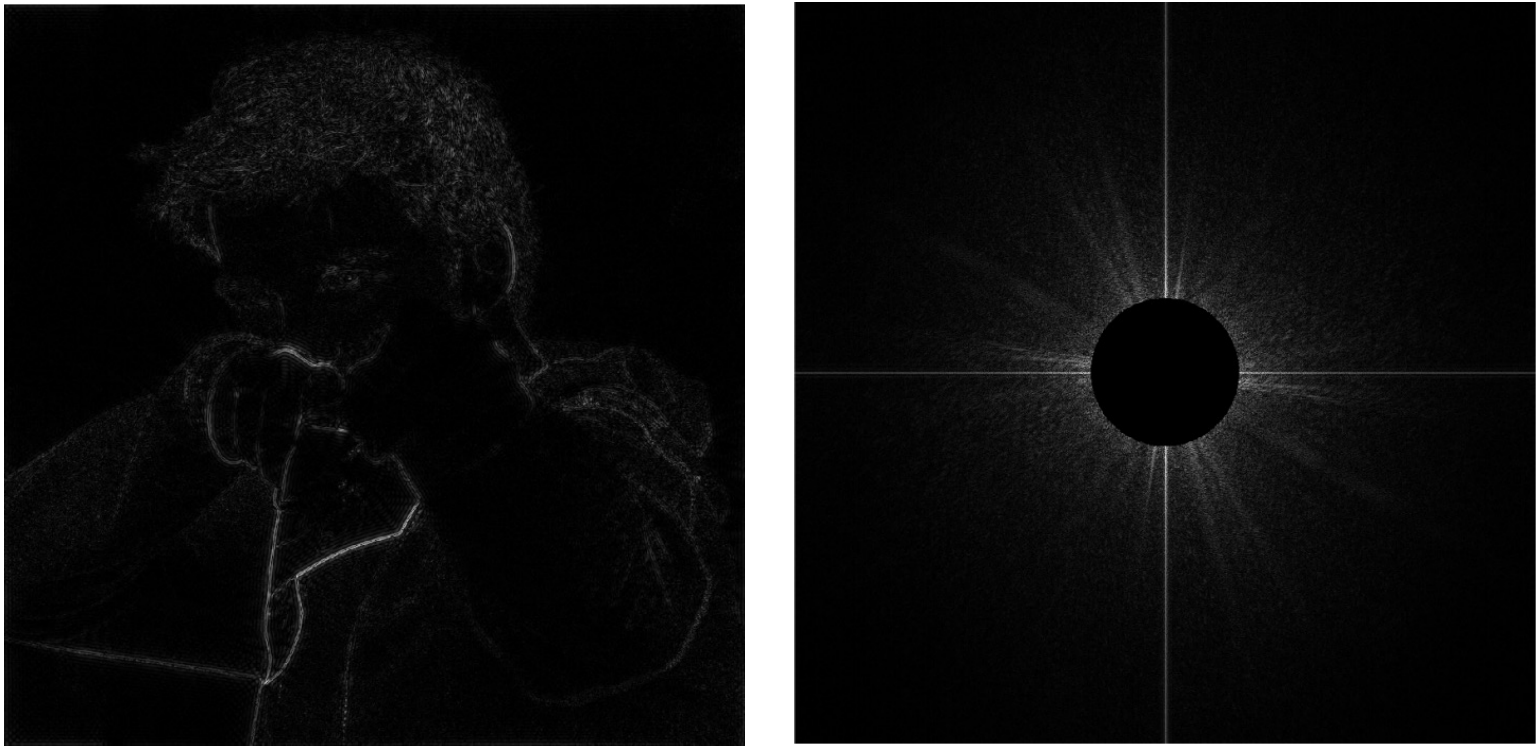
Two frequencies that are indistinguishable at a given sampling rate are called “aliases”

Filtering = Getting rid of certain frequency contents

Visualizing Image Frequency Content

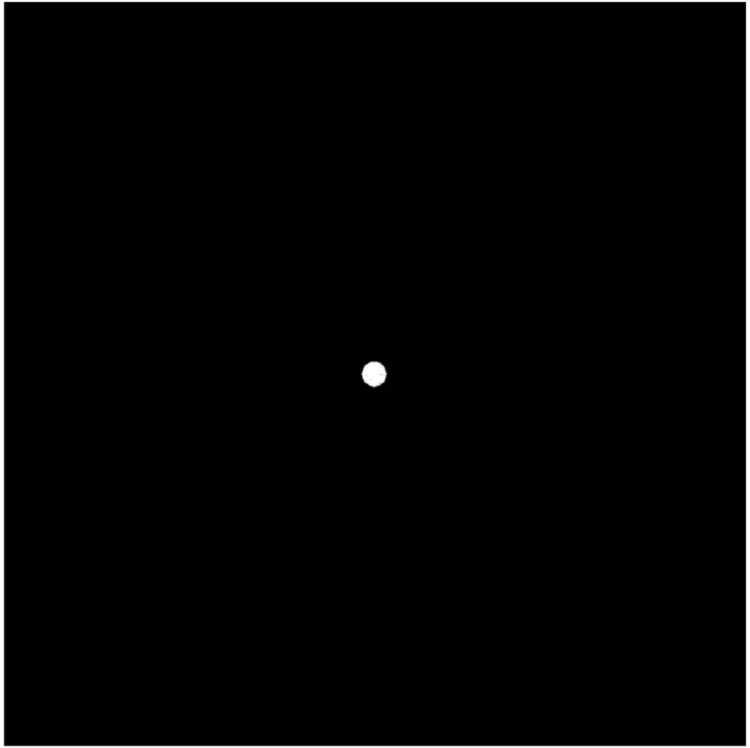


Filter Out Low Frequencies Only (Edges)



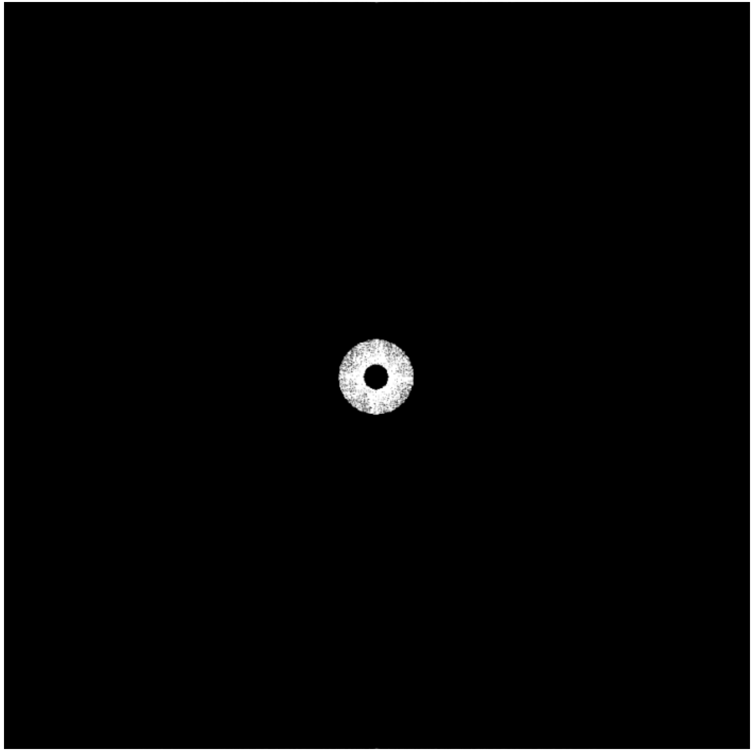
High-pass filter

Filter Out Low Frequencies Only (Blur)

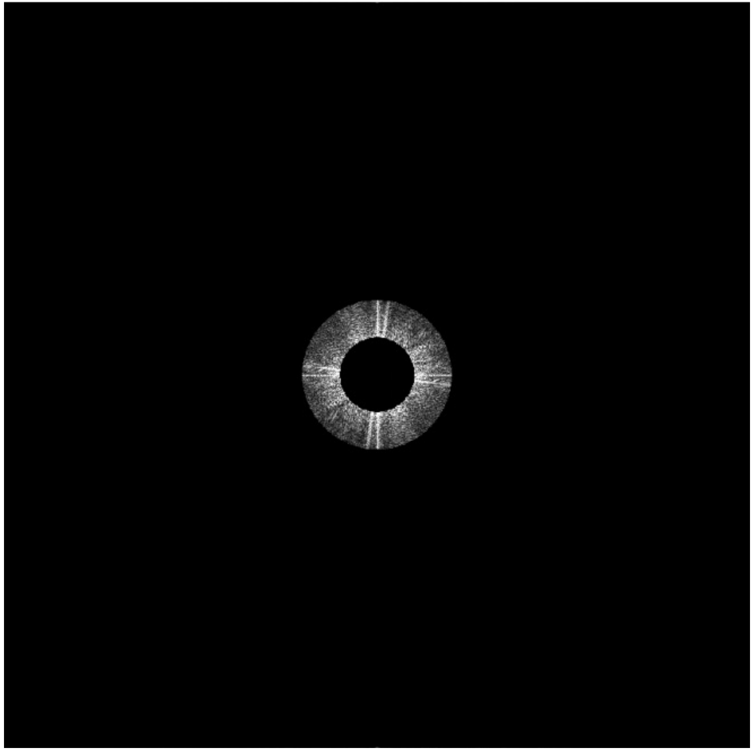


Low-pass filter

Filter Out Low and High Frequencies



Filter Out Low and High Frequencies



Filtering = Convolution (=Averaging)

Convolution

Signal	1	3	5	3	7	1	3	8	6	4

Filter	1/4	1/2	1/4

Point-wise local averaging in a “sliding window”

Convolution

Signal

1	3	5	3	7	1	3	8	6	4

Filter

1/4	1/2	1/4

$1 \times (1/4) + 3 \times (1/2) + 5 \times (1/4) = 3$

Result

	3								

Convolution

Signal

1	3	5	3	7	1	3	8	6	4

Filter

1/4	1/2	1/4

$$3 \times (1/4) + 5 \times (1/2) + 3 \times (1/4) = 4$$

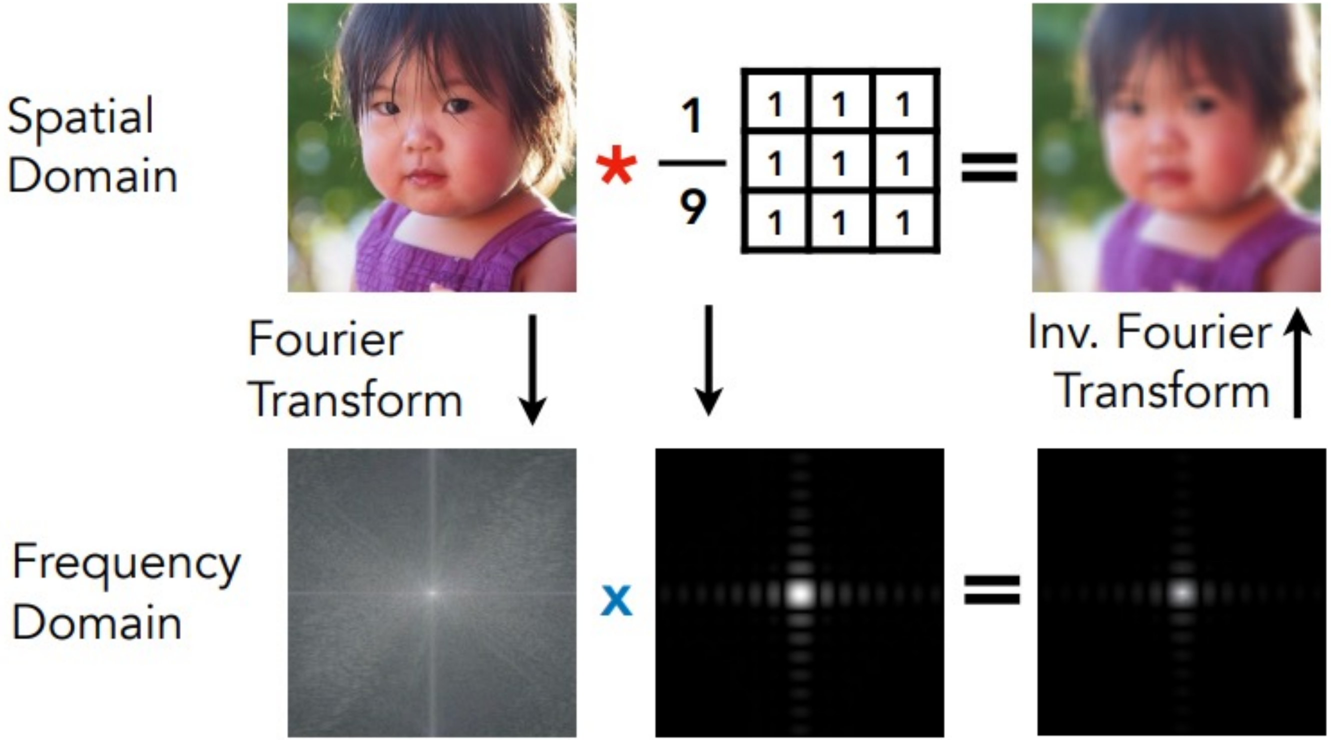
Result

	3	4							

Convolution Theorem

- Convolution in the spatial domain is equal to multiplication in the frequency domain, and vice versa
- Option 1:
  - Filter by convolution in the spatial domain
- Option 2:
  - Transform to frequency domain (Fourier transform)
  - Multiply by Fourier transform of convolution kernel
  - Transform back to spatial domain (inverse Fourier)

Convolution Theorem

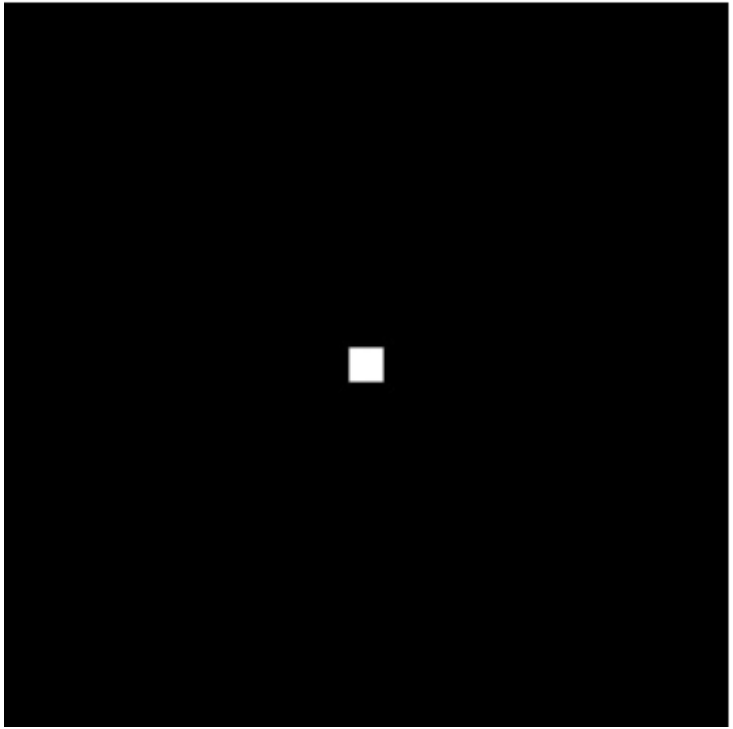


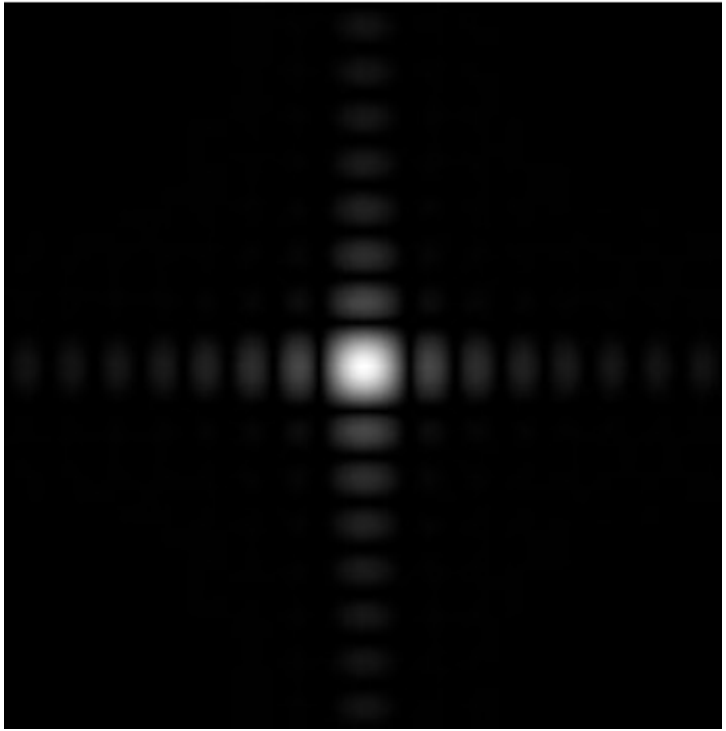
Box Filter

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array}$$

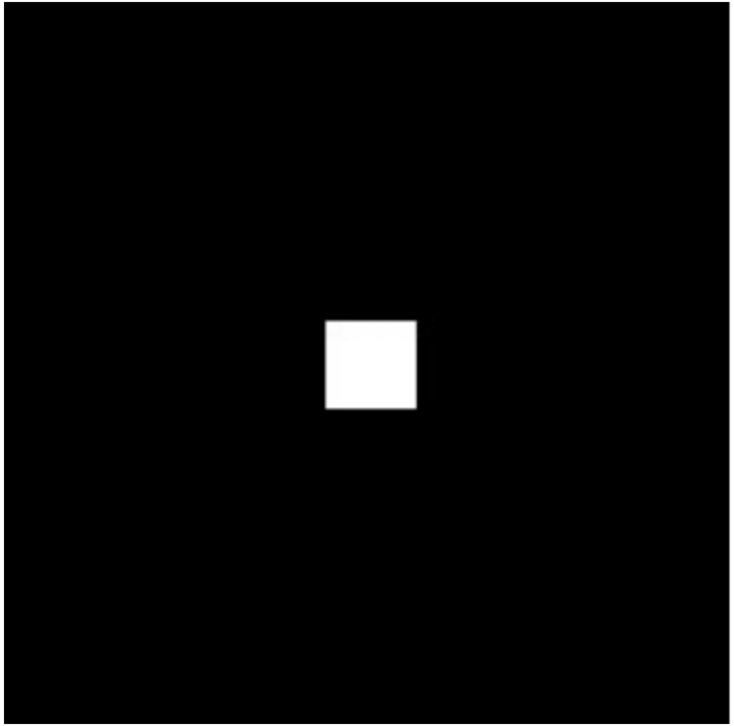
Example: 3x3 box filter

Box Function = “Low Pass” Filter



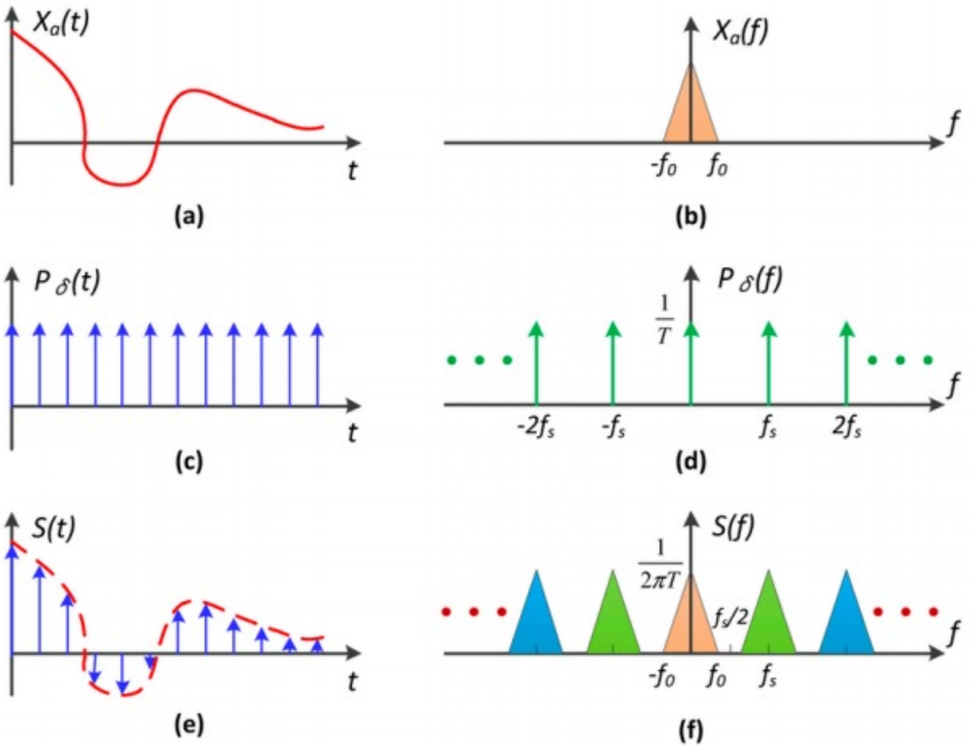


Wider Filter Kernel = Lower Frequencies



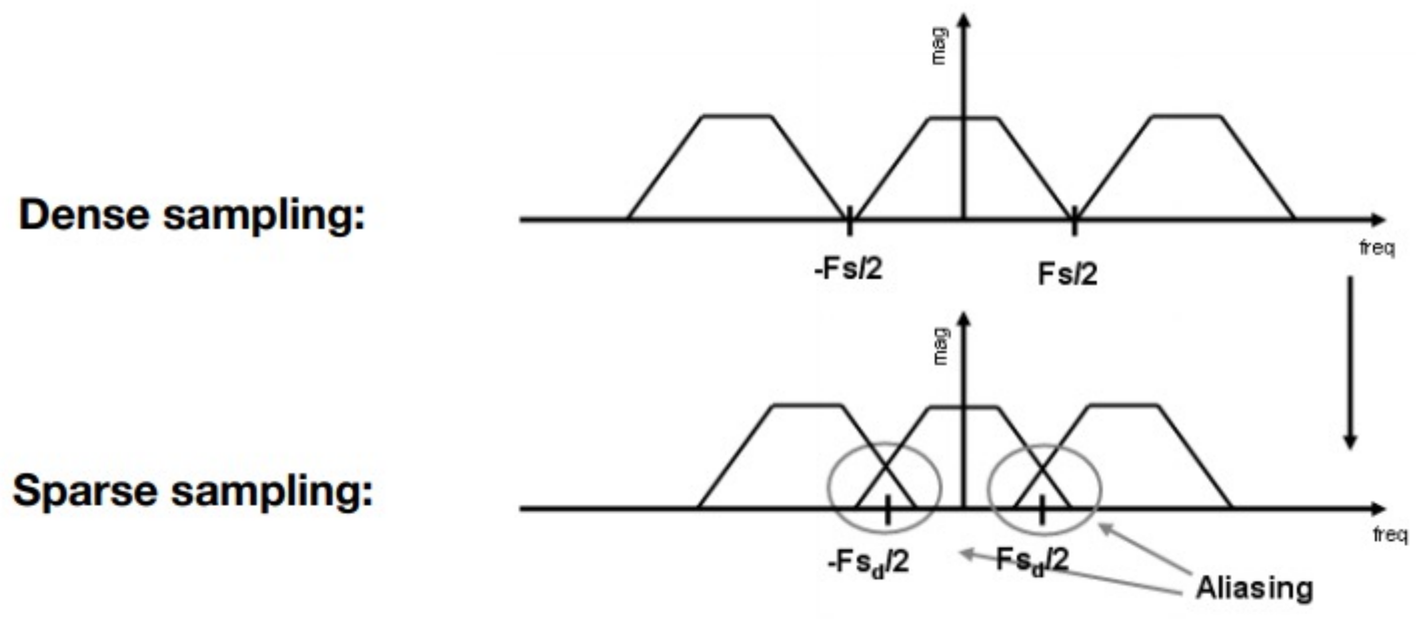
Sampling = Repeating Frequency Contents

Sampling = Repeating Frequency Contents



https://www.researchgate.net/figure/The-evolution-of-sampling-theorem-a-The-time-domain-of-the-band-limited-signal-and-b_fig5_301556095

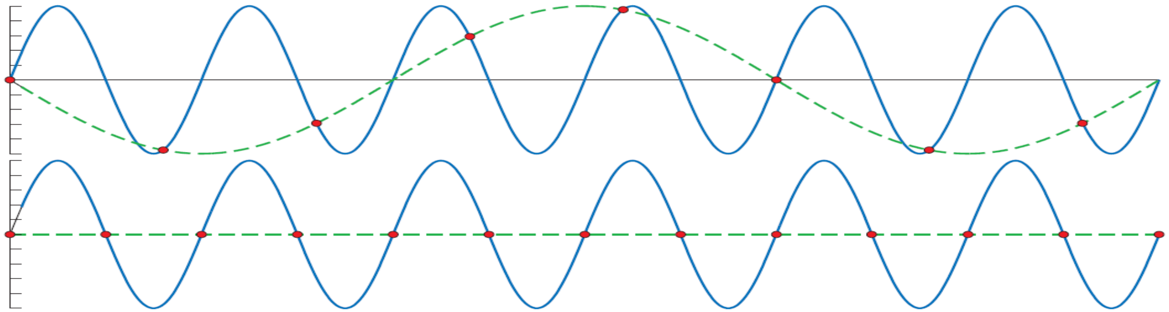
Aliasing = Mixed Frequency Contents



Antialiasing

How Can We Reduce Aliasing Error?

- Option 1: Increase sampling rate
  - Sampling theorem——The sampling frequency has to be **more than twice** the maximum frequency of the signal to be sampled



  - Higher resolution displays, sensors, framebuffers…
  - But: costly & may need very high resolution

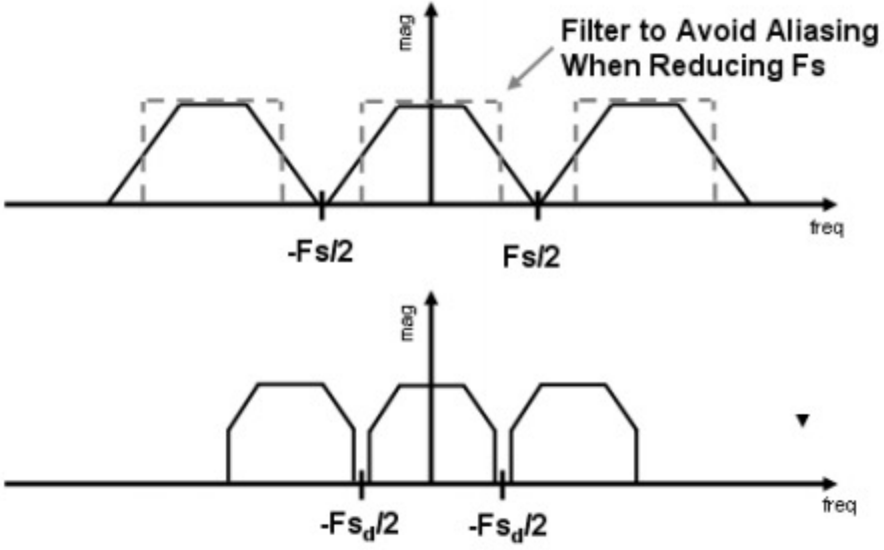
How Can We Reduce Aliasing Error?

- Option 2: **Antialiasing**
  - Making Fourier contents “narrower” before repeating
  - **i.e. Filtering out high frequencies before sampling**

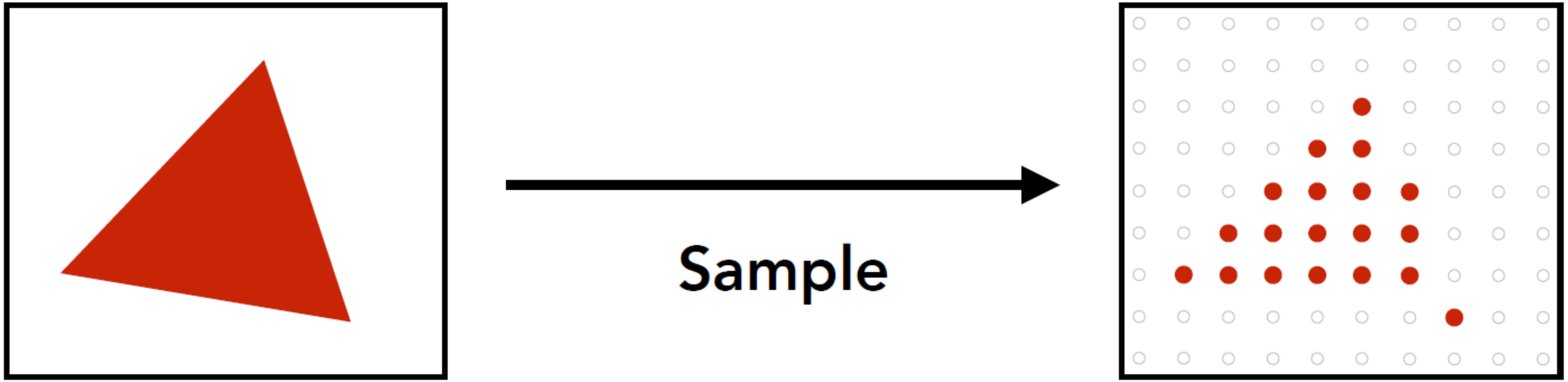
Antialiasing = Limiting, then repeating

Filtering

Then sparse sampling

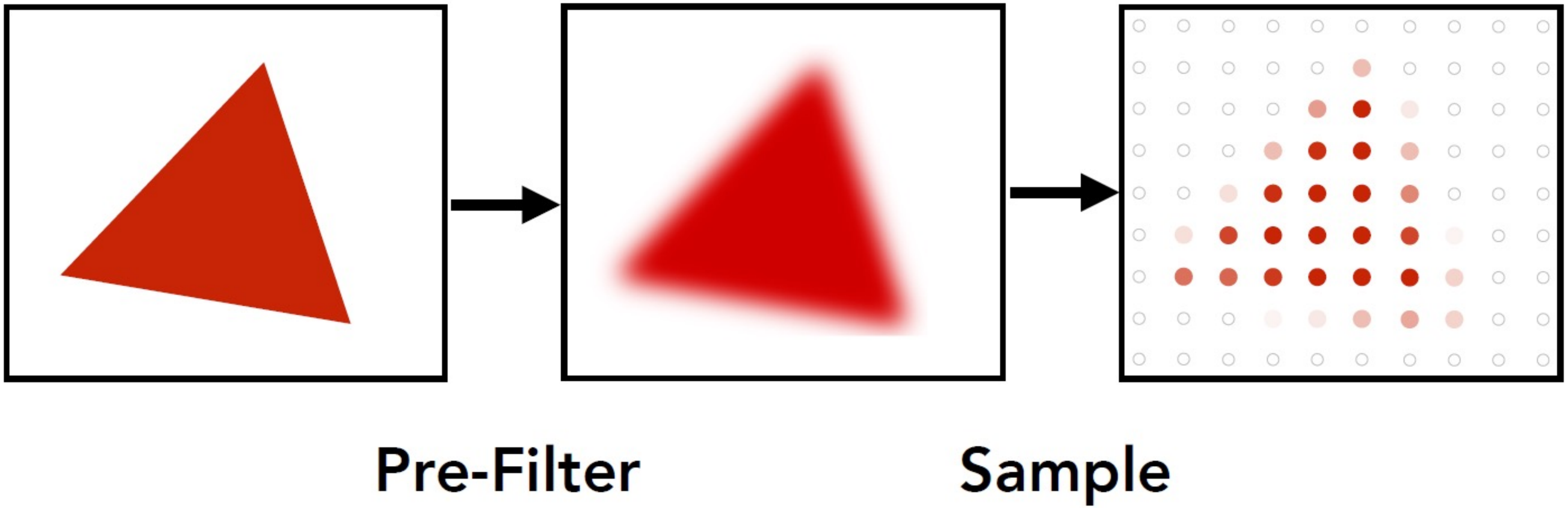


Regular Sampling



Note jaggies in rasterized triangle
where pixel values are pure red or white

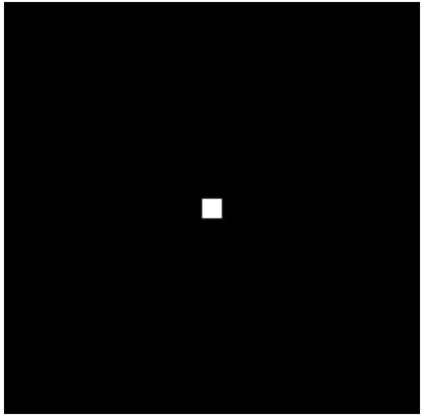
Antialiased Sampling



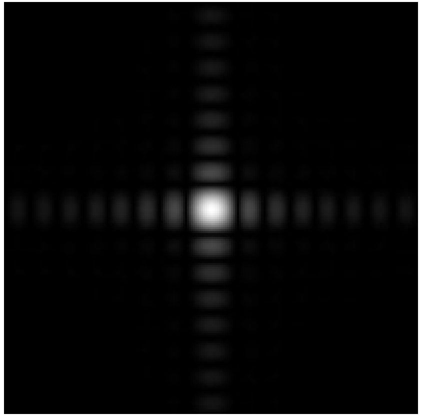
Note antialiased edges in rasterized triangle
where pixel values take intermediate values

A Practical Pre-Filter

A 1 pixel-width box filter (low pass, blurring)



Spatial Domain



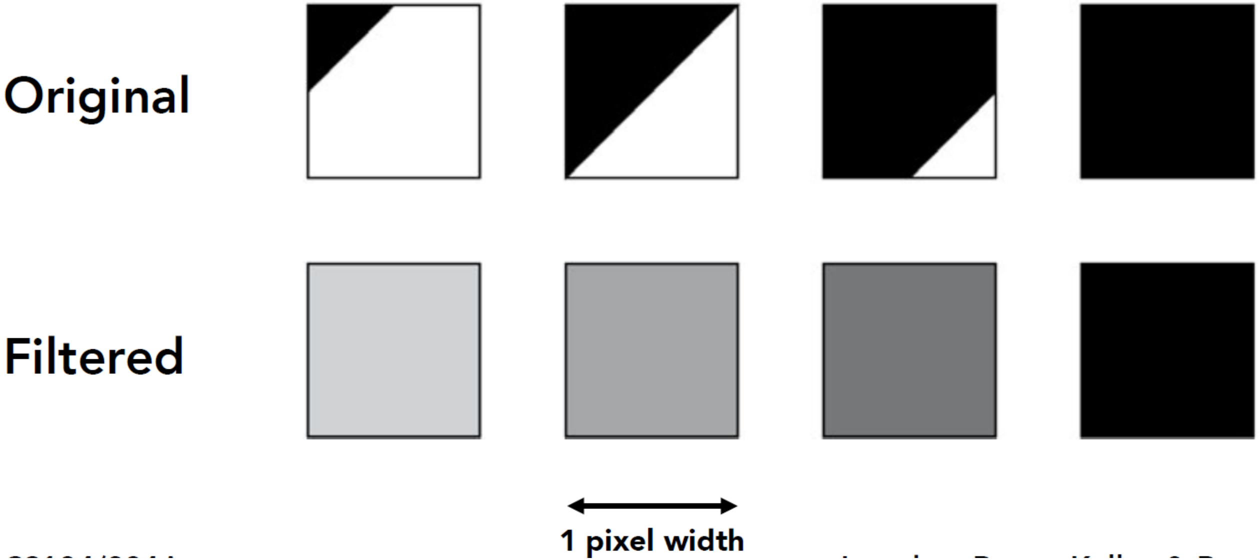
Frequency Domain

A Antialiasing By Averaging Values in Pixel Area

- Solution:
  1. **Convolve** f(x,y) by a 1-pixel box-blur
     - - Recall: convolving = filtering = averaging
  2. Then **sample** at every pixel's center

Antialiasing By Averaging Values in Pixel Area

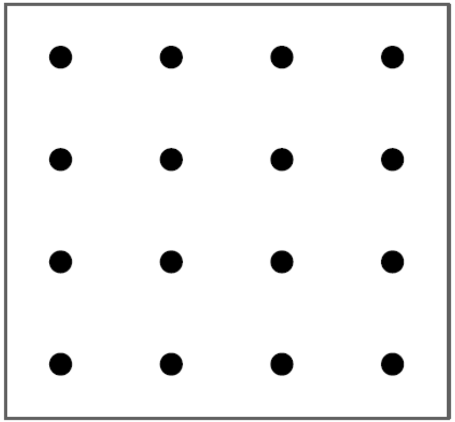
- The average value inside a pixel area of f(x,y) = inside(triangle,x,y) is equal to the area of the pixel covered by the triangle



Antialiasing By Supersampling (MSAA)

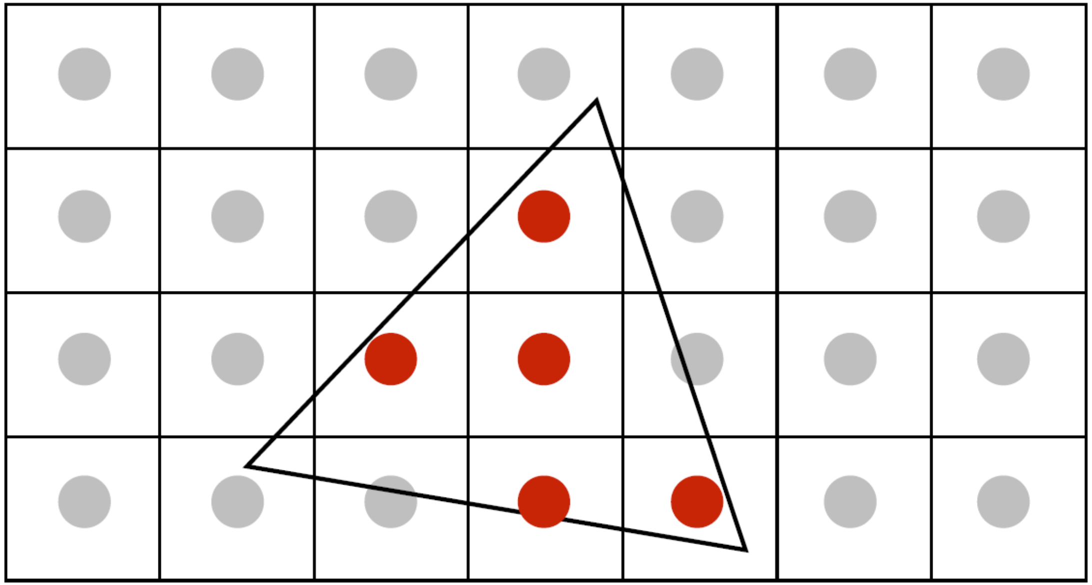
Antialiasing By Supersampling (MSAA)

- MSAA, multisampling antialiasing
- Approximate the effect of the 1-pixel box filter by sampling multiple locations within a pixel and averaging their values:



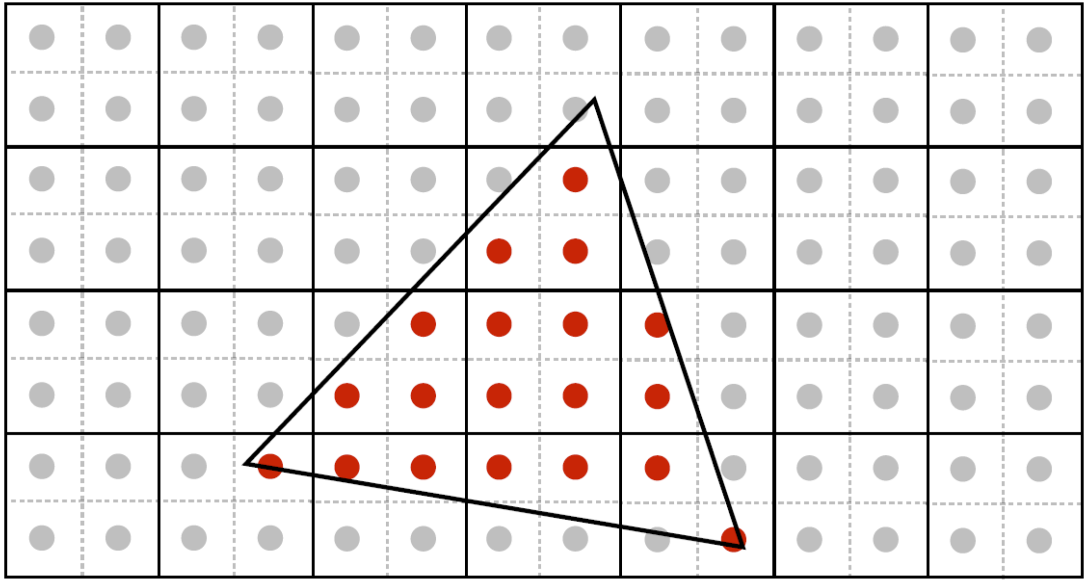
4x4 supersampling

Point Sampling: One Sample Per Pixel



Supersampling: Step 1

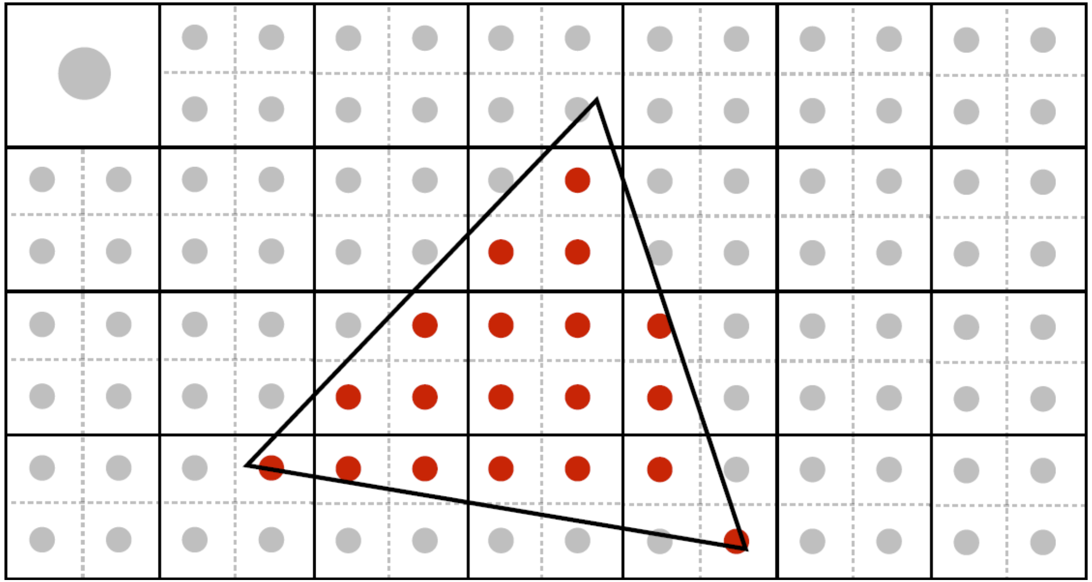
- Take NxN samples in each pixel.



2x2 supersampling

Supersampling: Step 2

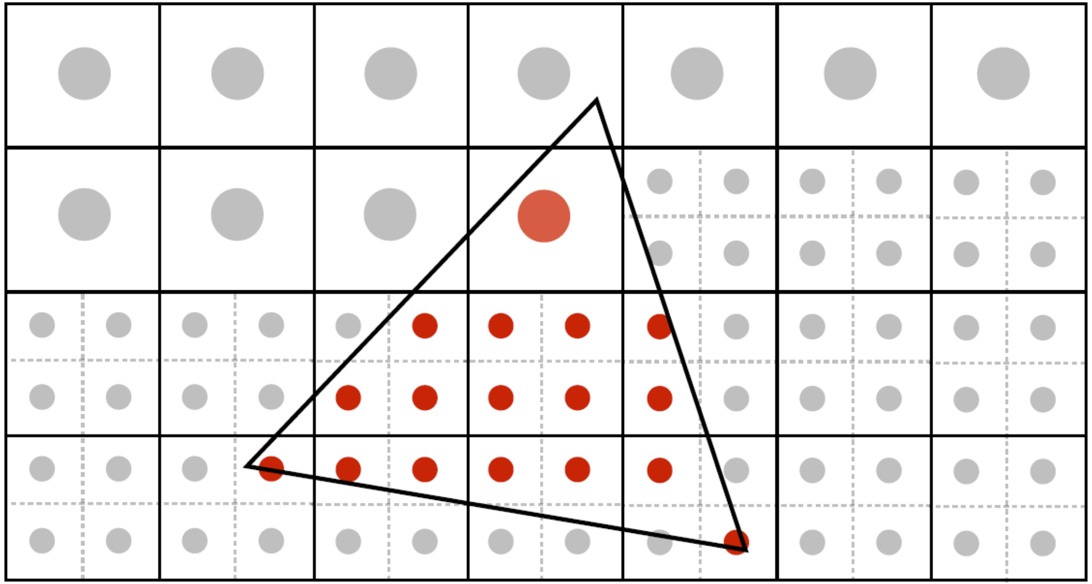
- Average the NxN samples “inside” each pixel



Averaging down

Supersampling: Step 2

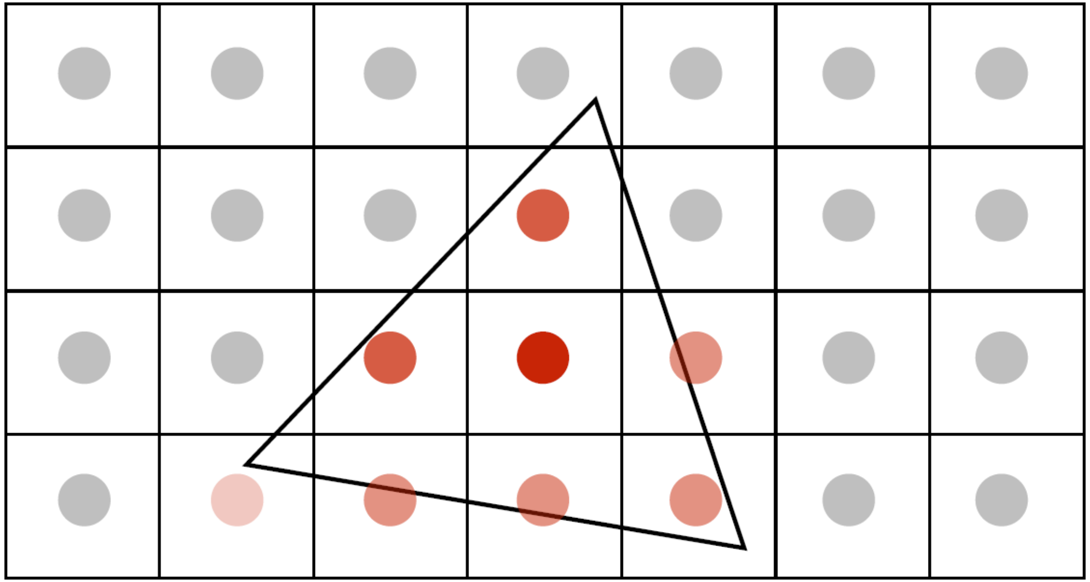
- Average the NxN samples "inside" each pixel



Averaging down

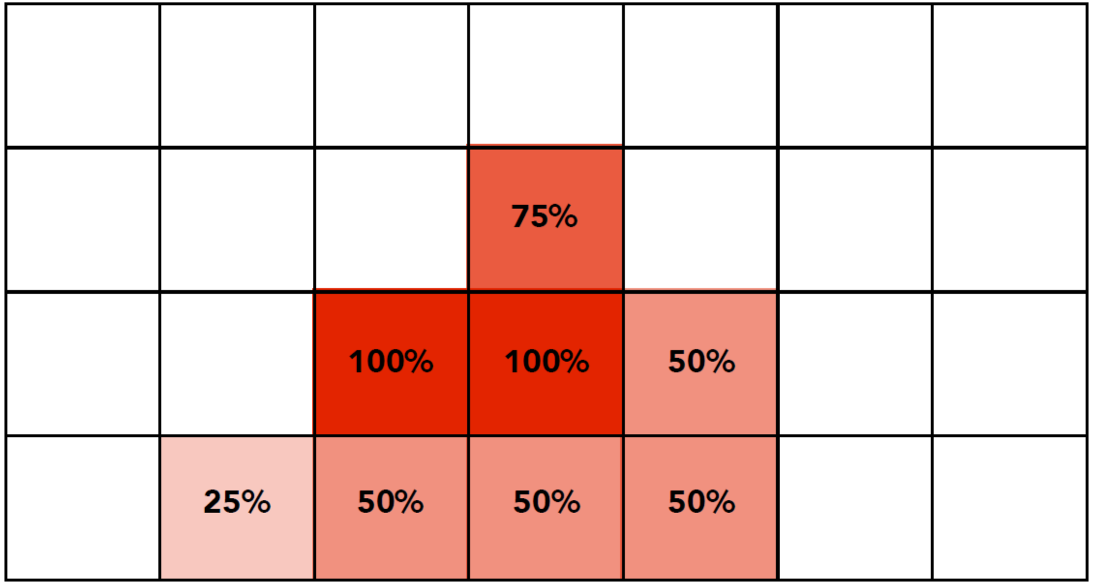
Supersampling: Step 2

- Average the NxN samples “inside” each pixel

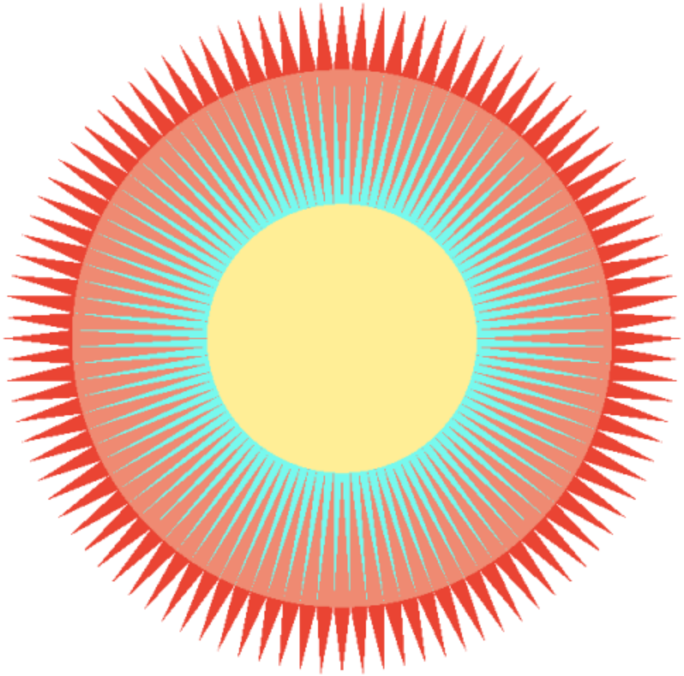


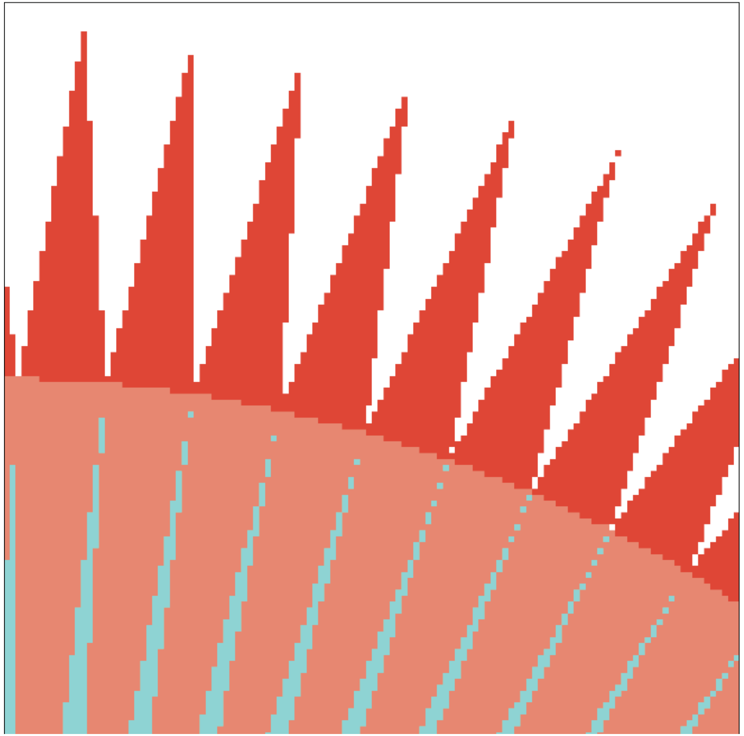
Supersampling: Result

- This is the corresponding signal emitted by the display

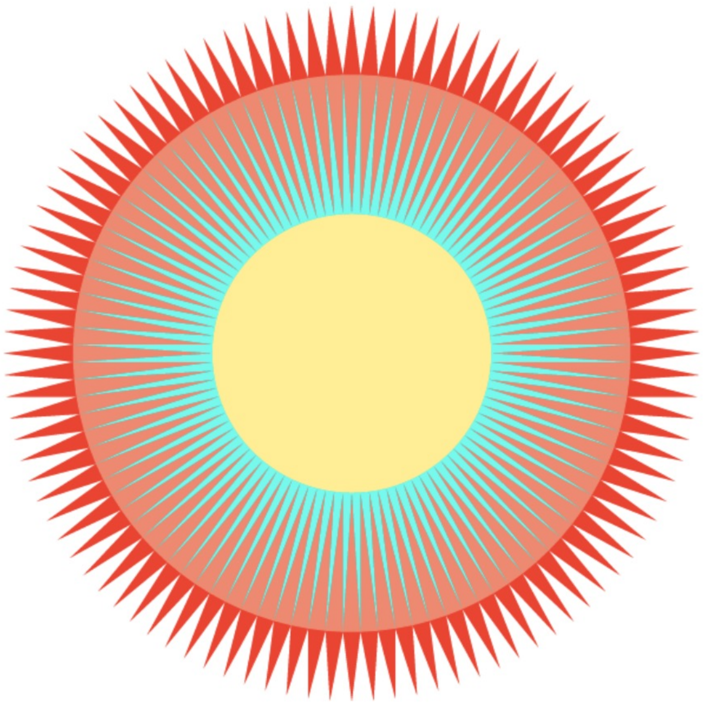


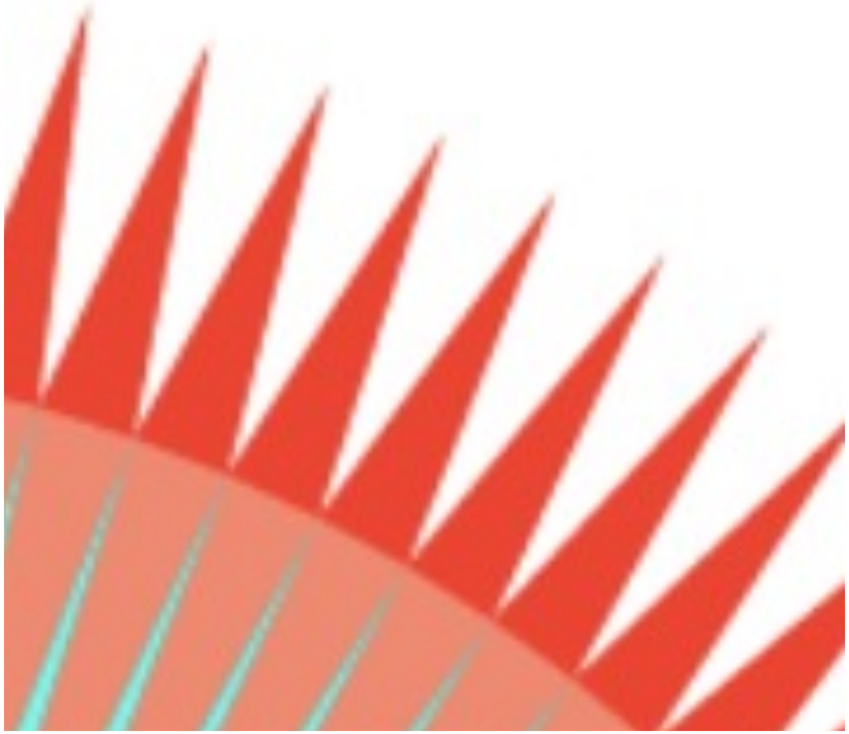
Aliasing——Point Sampling





Antialiasing——4x4 Supersampling





Antialiasing Summary

- No free lunch!
  - What's the cost of MSAA?
- Milestones (personal idea)
  - FXAA (Fast Approximate AA)
  - TAA (Temporal AA)
- Super resolution / super sampling
  - From low resolution to high resolution
  - Essentially still "not enough samples" problem
  - DLSS (Deep Learning Super Sampling)

Thank you!