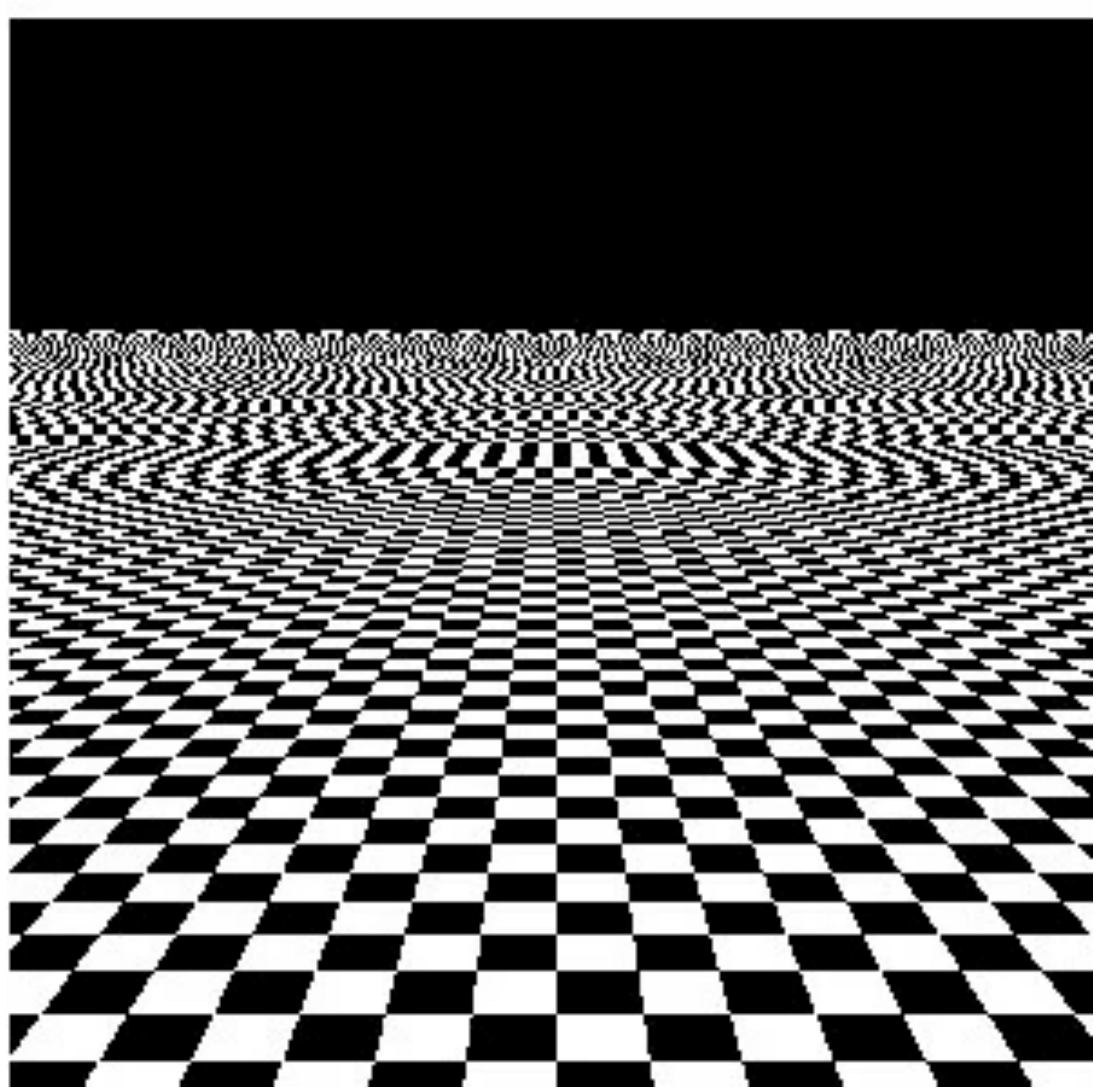
Antialiasing

CS 385 - Class 25
26 April 2022

The Problem

Aliasing

- The classic checkerboard rendering issue
- What is the problem here?

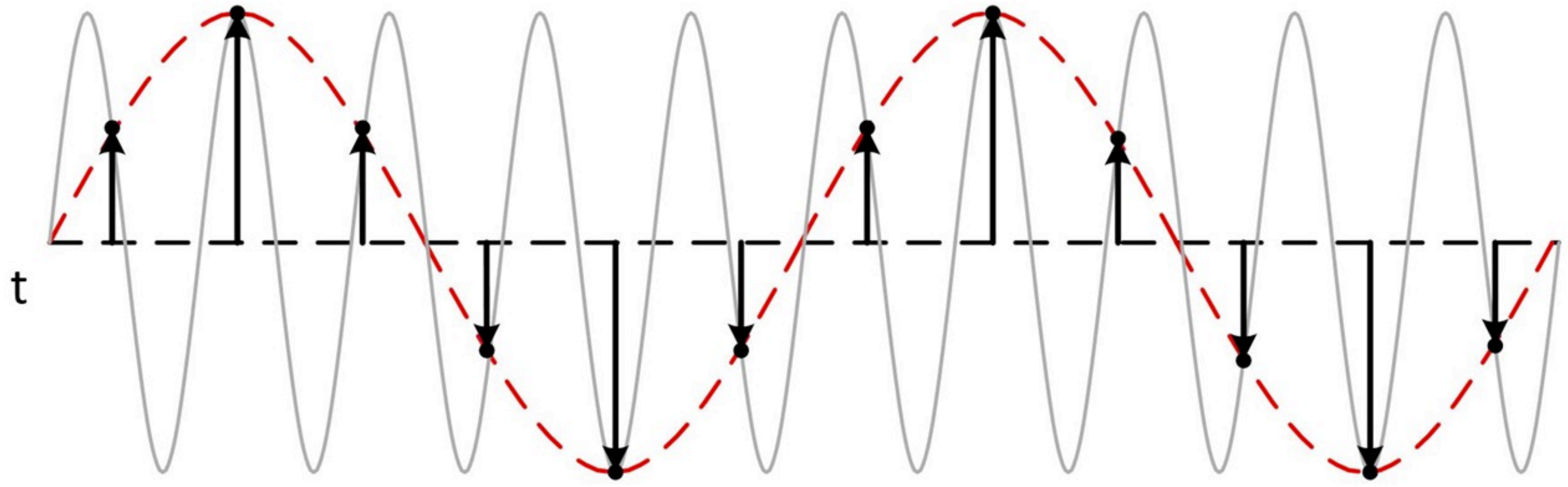


First, Some Terminology

- A *signal* is just a function – think $f(\overrightarrow{x})$
  - $\overrightarrow{x}$ is just shorthand for any number of dimensions
  - signal could be an audio (1D), an image (2D or 3D), or whatever
- A *sample* is the value of the function at a particular point
- The *sampling rate* is how many samples per measuring unit
  - usually specified as a *frequency*, measured in *Hertz* (Hz)
  - For example, CD audio is sampled at 44kHz
- *Reconstructing* a signal is an attempt to determine what the original signal was from a number of samples

Sampling (and Undersampling) a Signal

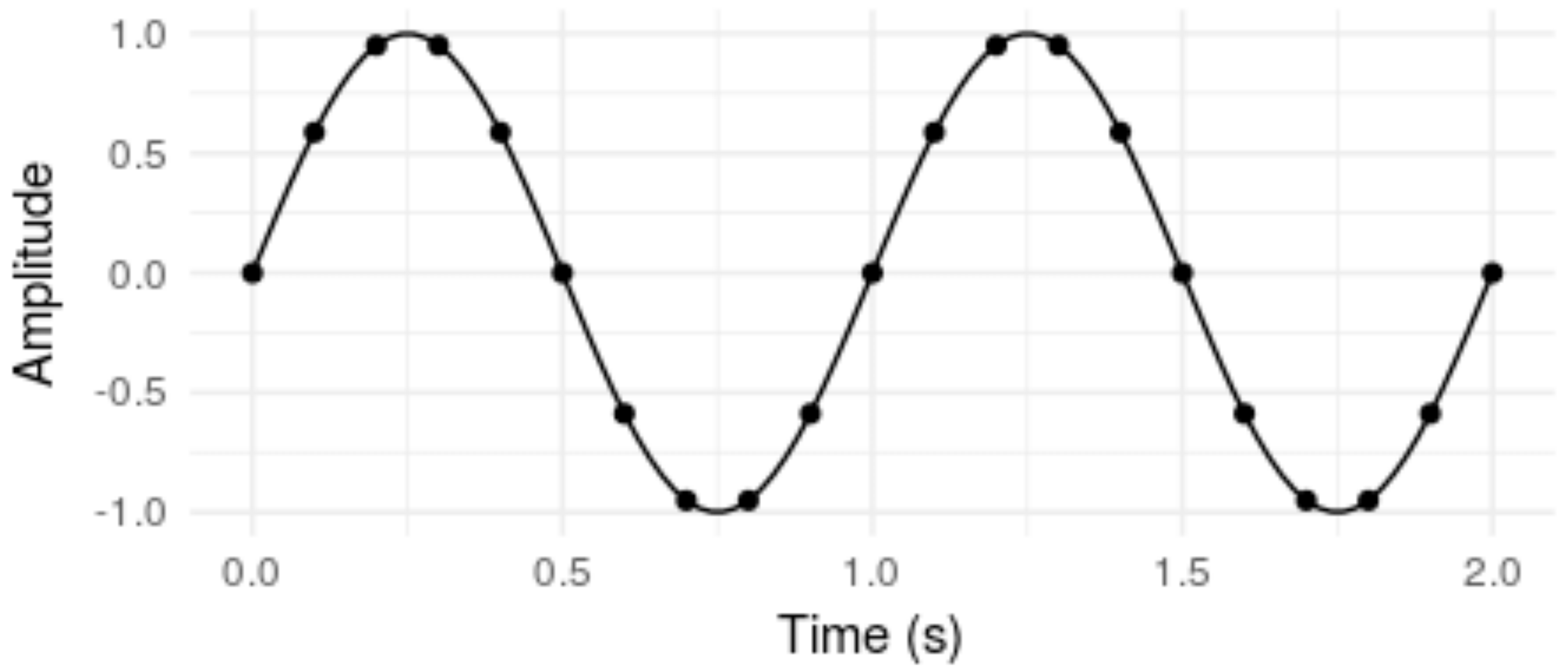
- The sampling rate controls how well we can reconstruct the signal



- Sampling rate is much below the frequency of the signal
- Consequently, the reconstructed signal doesn't look like the original signal
- This is the basis of *aliasing*

Nyquist Rate

- In order to accurately reconstruct a signal, need to sample at twice the highest frequency



$f_{signal} = 1$ Hz $\qquad f_{sampling} = 10$ Hz

Our hero:
Harry Nyquist

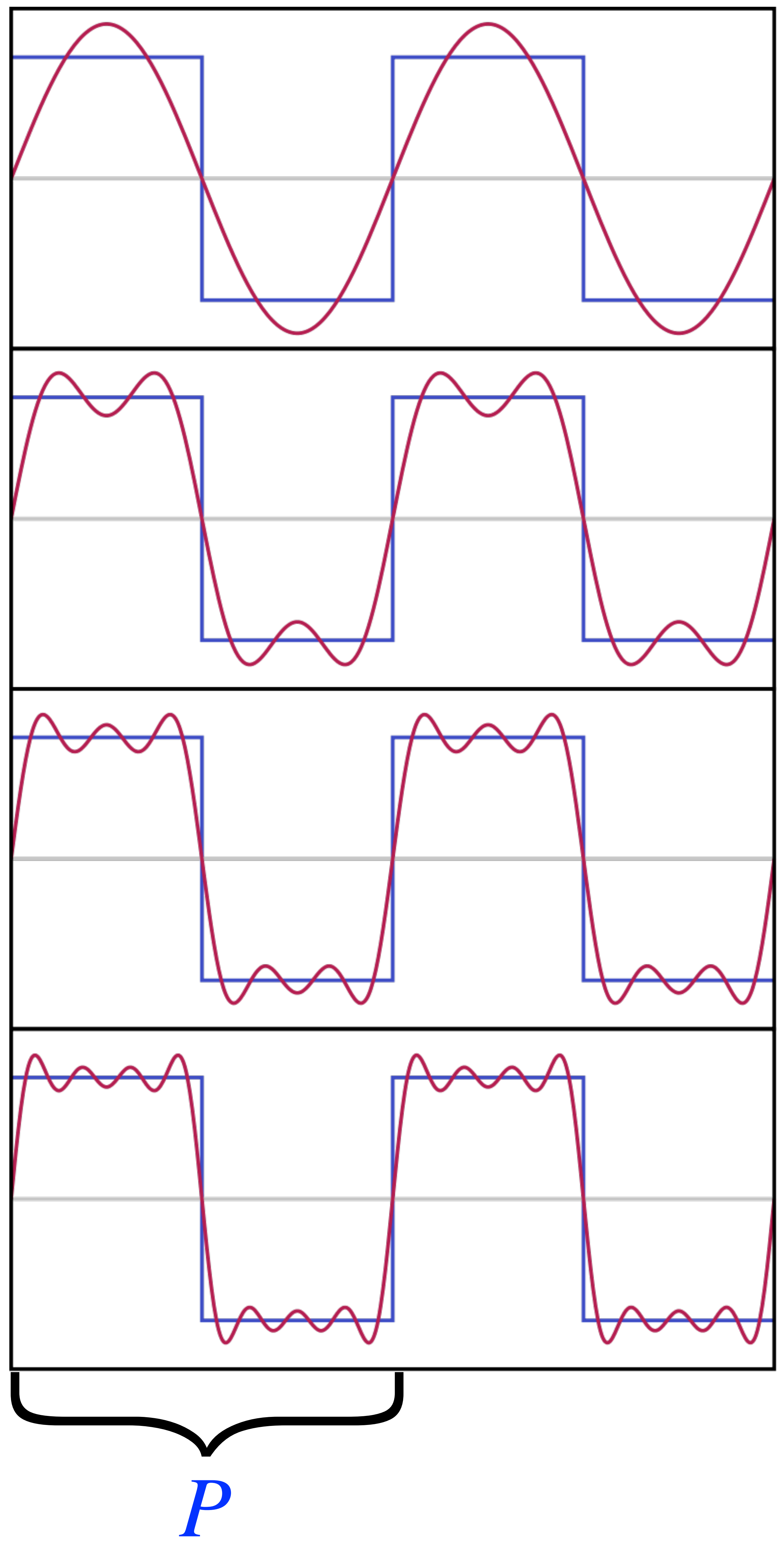
$$f_{sampling} \geq 2\, f_{signal}$$

Fourier Series

- Approximate a function using a sum of periodic (cosine, in this case) functions

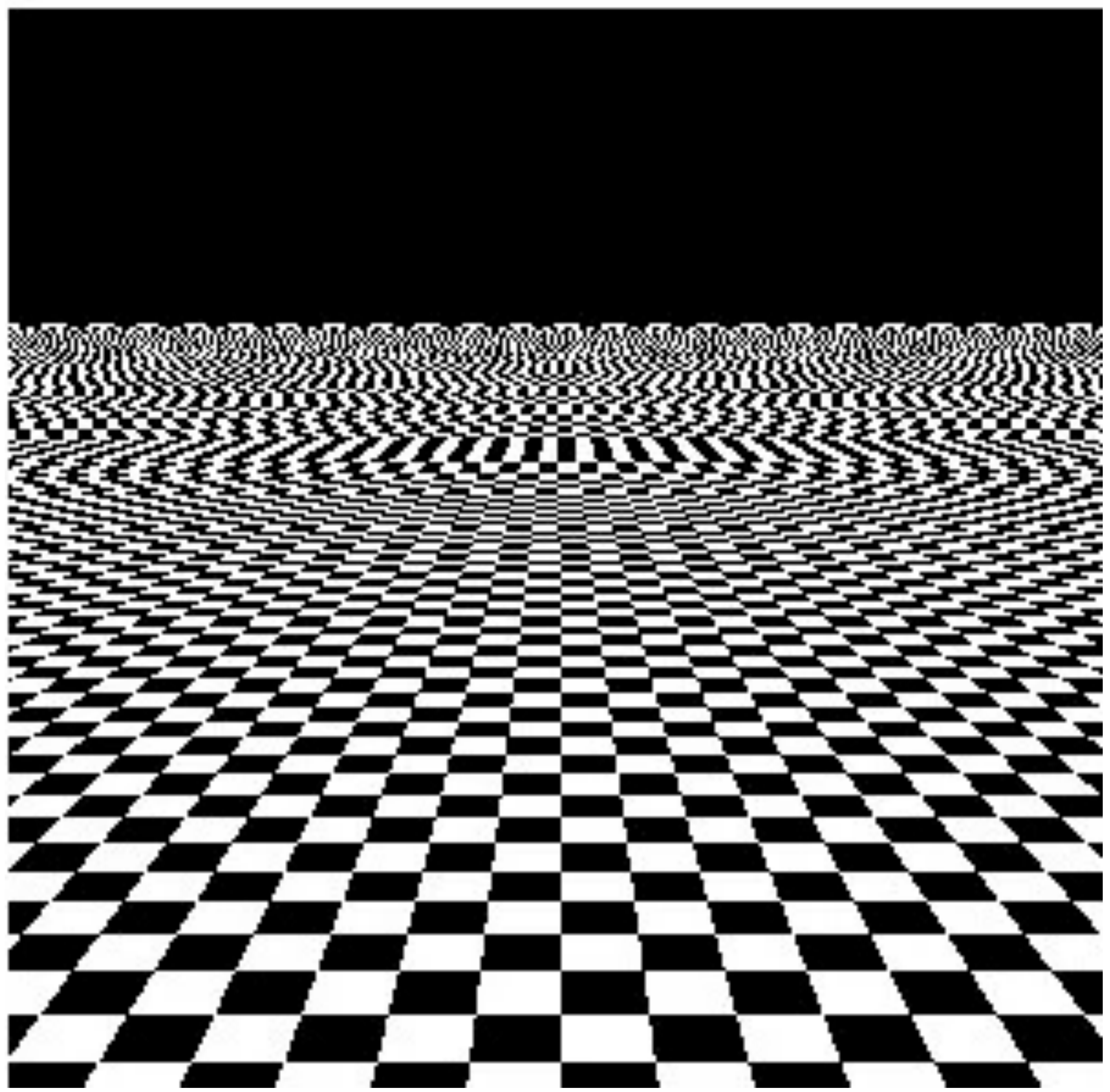
$$f(x) = \frac{A_0}{2} + \sum_{n=1}^{\infty} A_n \cos\left(\frac{2\pi}{P}nx - \phi_n\right)$$

- Original signal repeats every P cycles
- $f_n = \frac{2\pi}{P}n$ is the *frequency* of term n



Aliasing

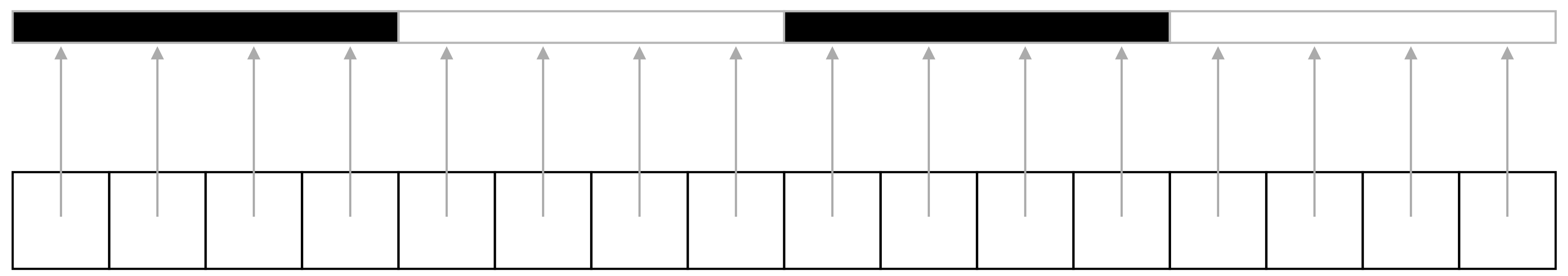
- The classic checkerboard rendering issue
- What is the problem here?
- *The signal's under sampled for some pixels*

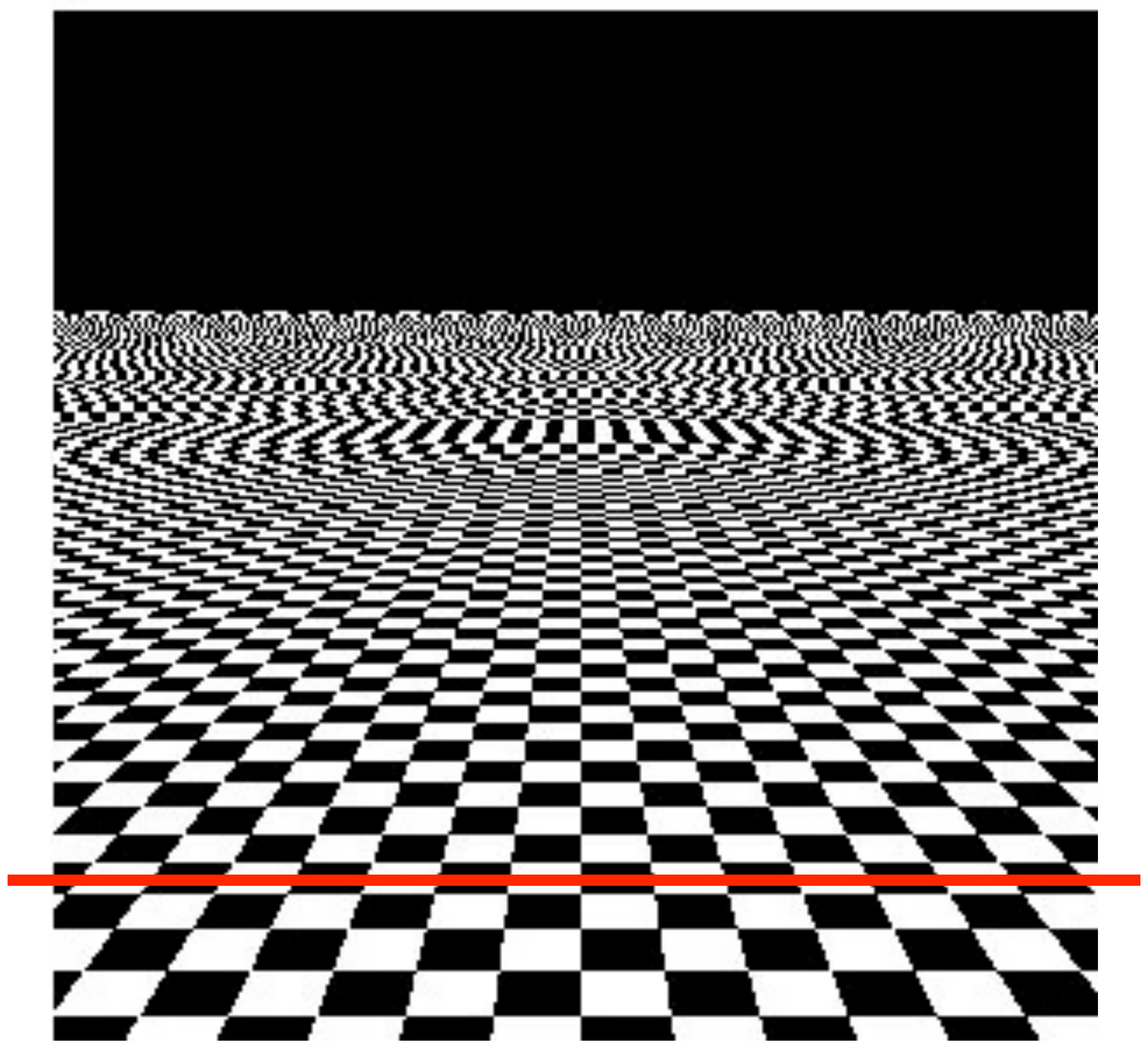


Texture Sampling

Texturing & Sampling

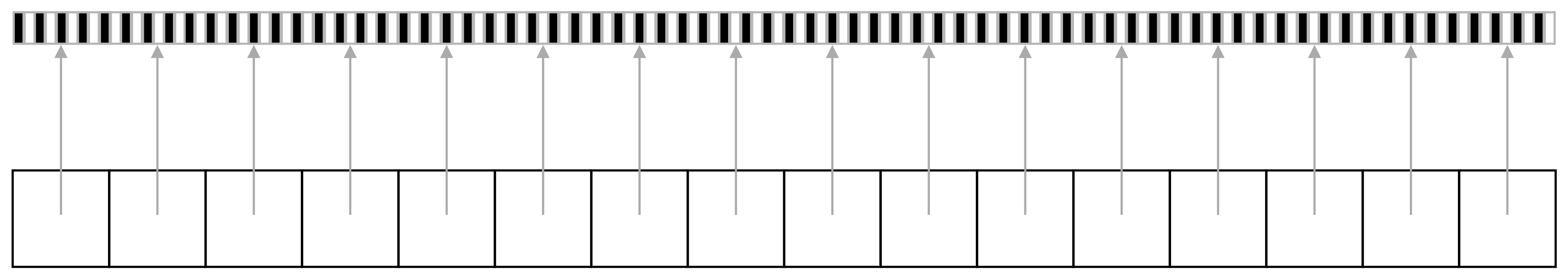
- Sampling rate is greater than the texture's frequency
  - All good!

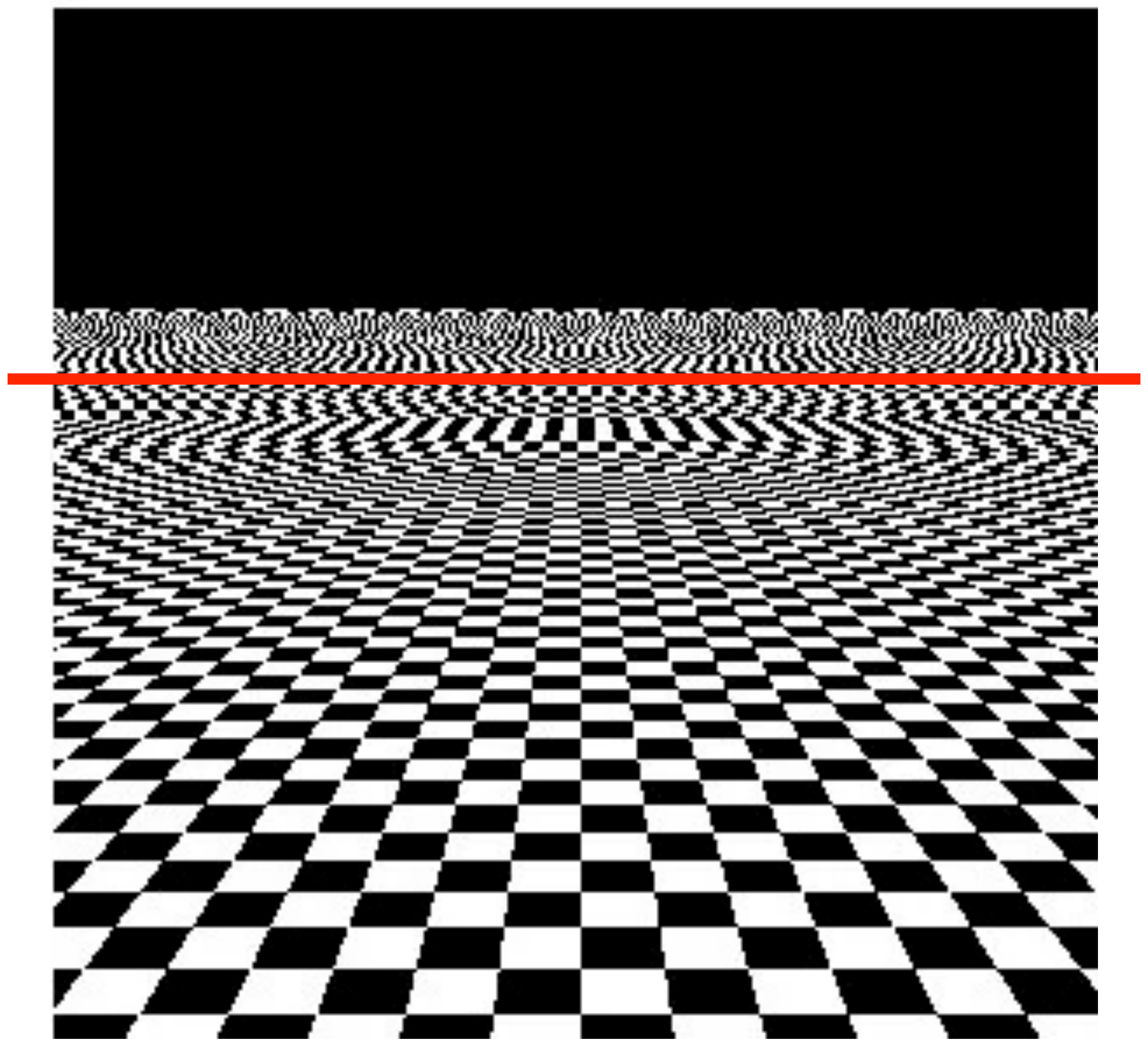




Texturing & Sampling

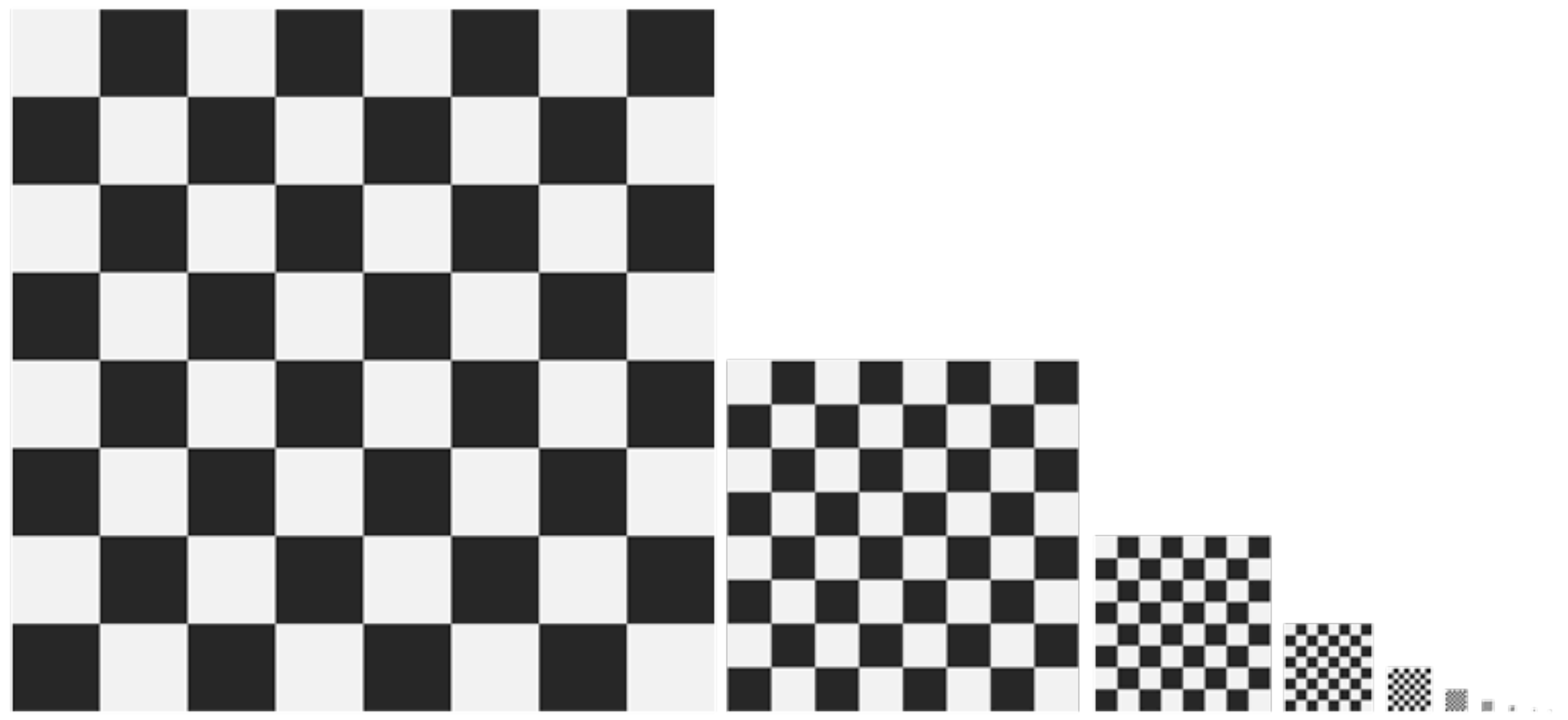
- Sampling rate is too small compared to the texture's frequency
  - Aliasing!





The Return of Mipmaps

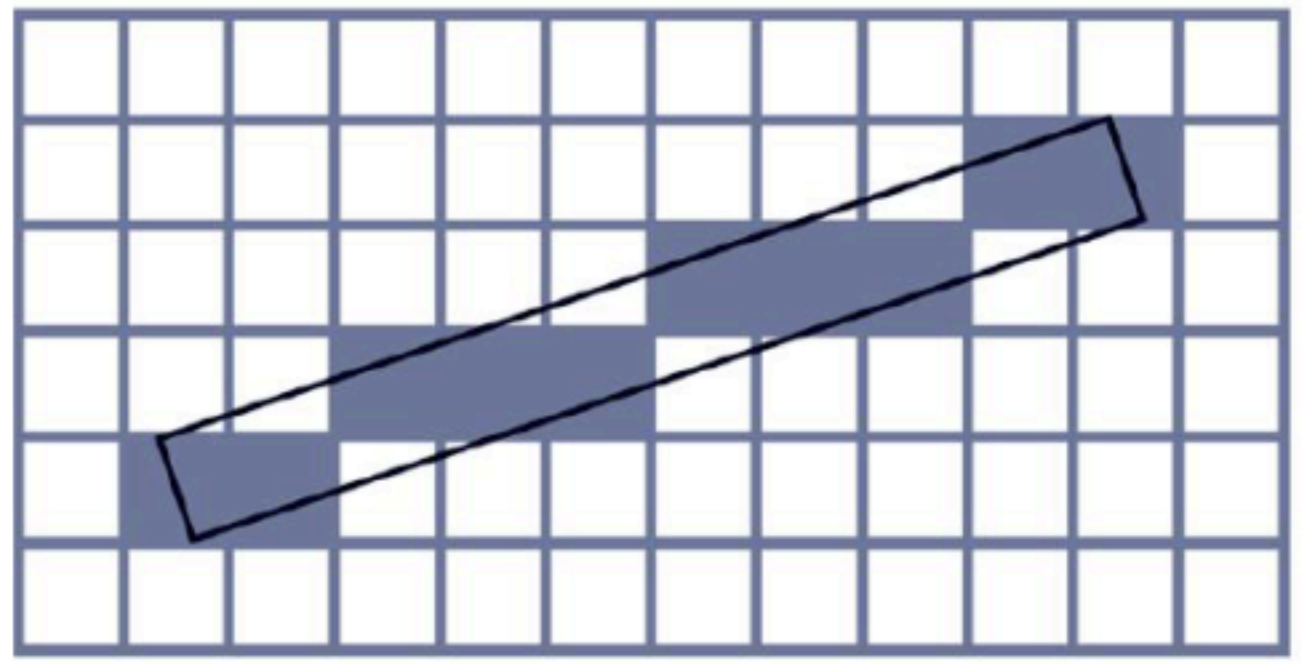
- Recall *mipmaps* from the texture mapping class
  - generate small versions of the original texture (mips) to better match the sampling rate during texturing
- However, the checkerboard is diabolically evil
  - transitions between light and dark require an infinite number of terms in the Fourier series
    - no real way to meet Nyquist in that situation



Geometric Antialiaising

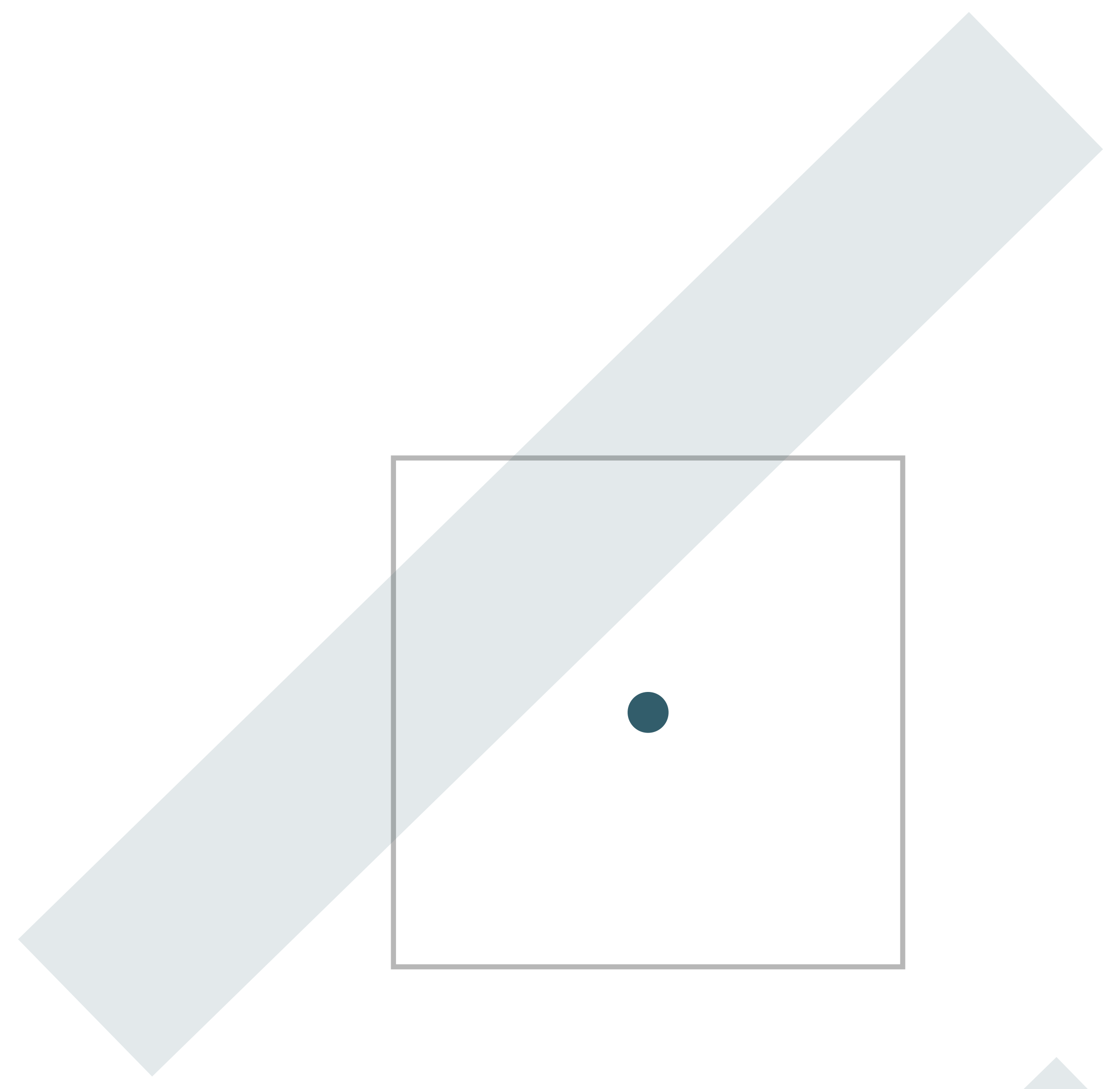
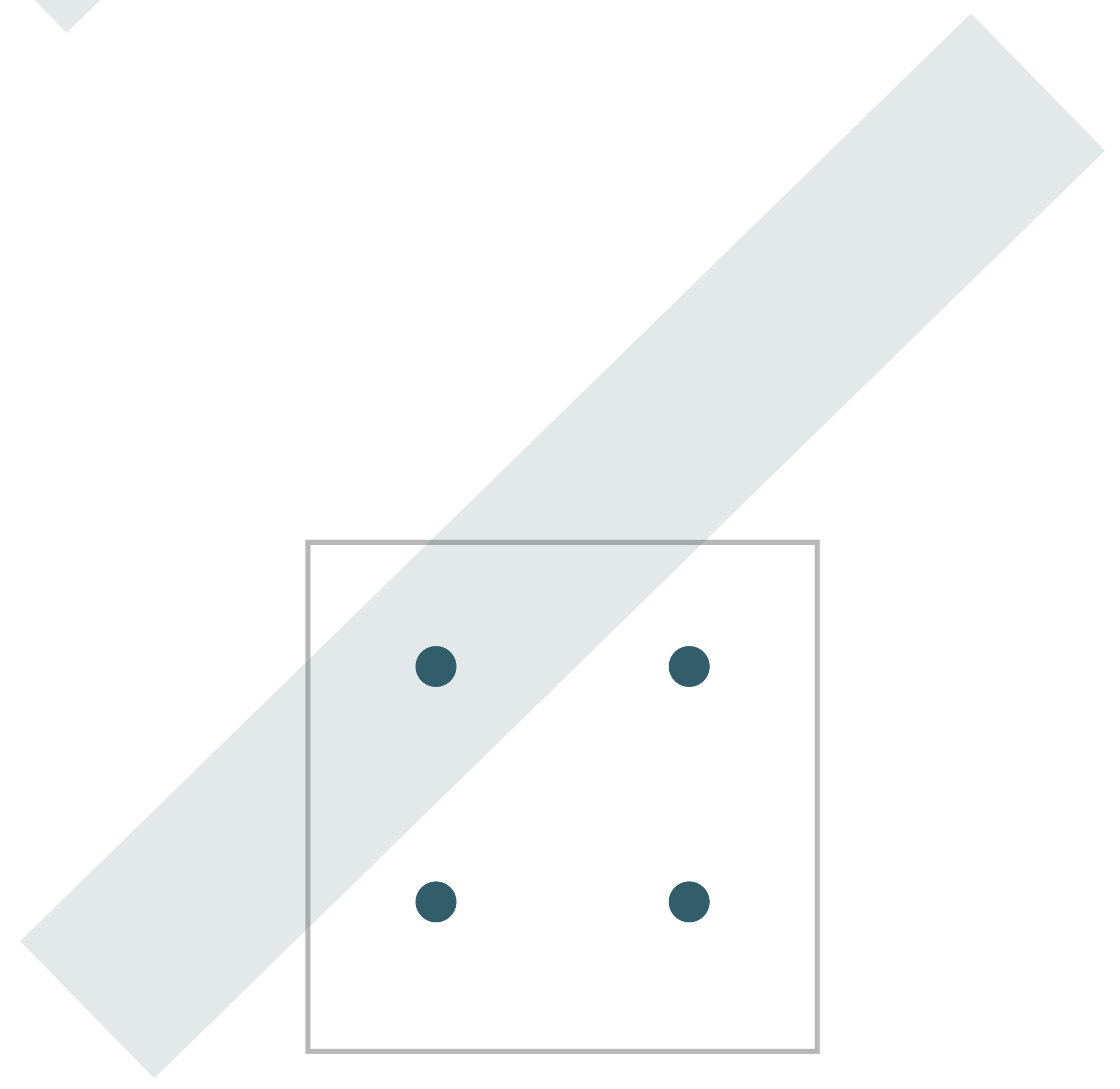
Rasterization & Sampling

- Pixels sample geometry at their pixel centers
  - if the center isn't in the primitive, no fragment is generated
- Results in the *jaggies*
  - yup, that's the technical term



Multisampling

- We solved antialiasing by sampling more
- Can we do that per pixel?
- Enter: *multisampling*
- Sample (rasterize) at more than just the pixel center
- Each *sample* (can) get rasterized just like the pixel center
  - *Supersampling*
- or just compute coverage and assign same color, depth, and stencil to each sample

Enabling Multisampling

- Not Exam Material!
- Several steps involved
  1. Create a multisampled render buffer (part of a framebuffer object)
  2. Bind FBO with multisampled buffer
  3. Render
  4. Bind FBO from 2. as the *read framebuffer*
  5. Bind another FBO (or default FBO) as the *write framebuffer*
  6. Blit to copy and *resolve* to a single-sample buffer/texture