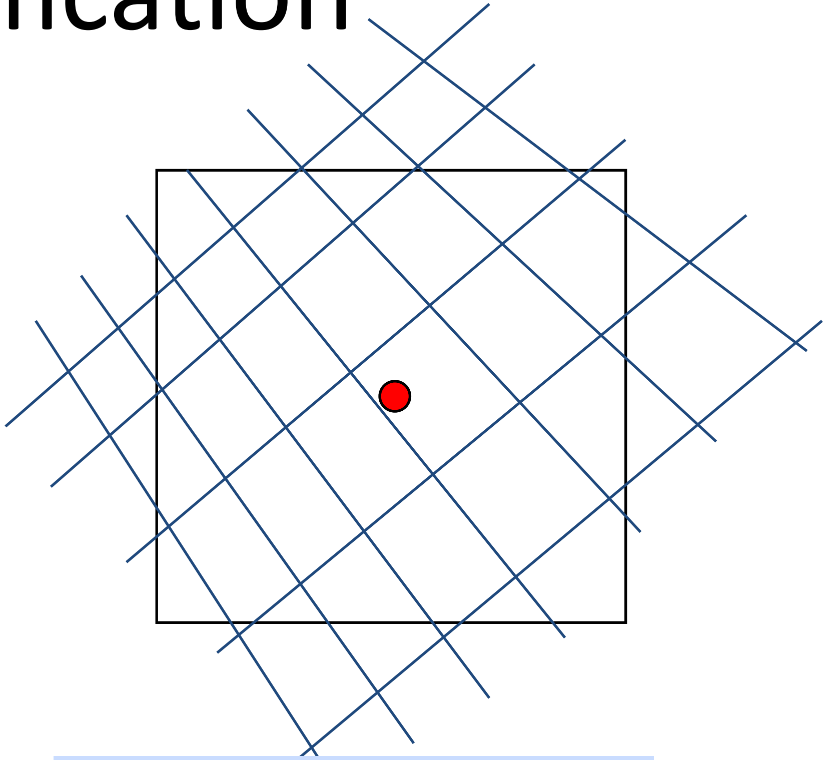
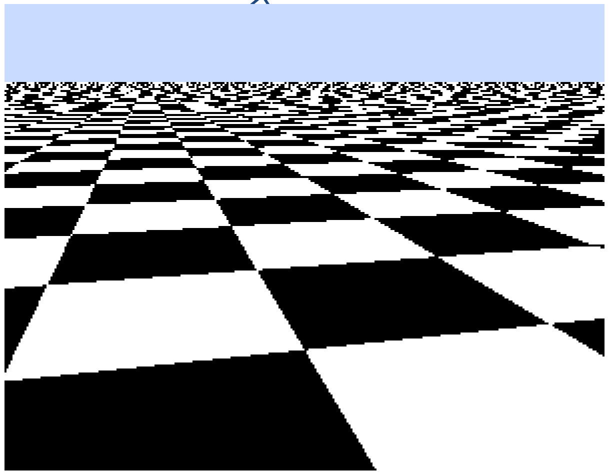
Antialiasing

Steve Rotenberg
CSE168: Rendering Algorithms
UCSD, Spring 2014

Texture Minification

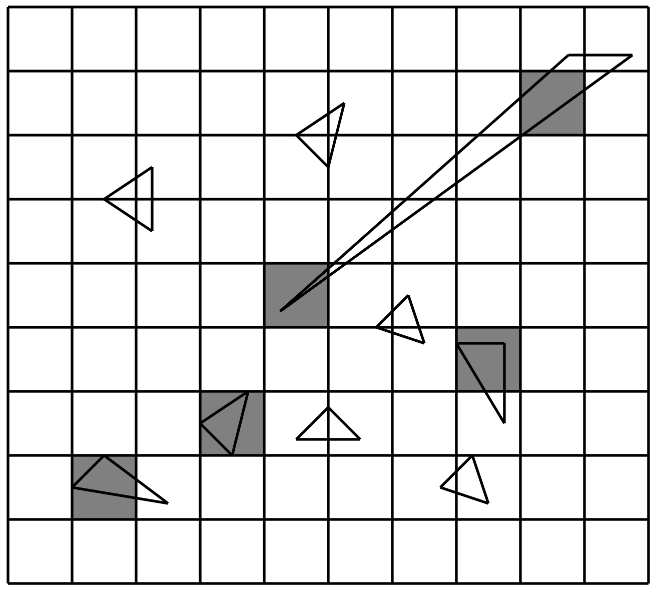
- Consider a texture mapped triangle
- Assume that we *point sample* our texture so that we use the nearest texel to the center of the pixel to get our color
- If we are far enough away from the triangle so that individual texels in the texture end up being smaller than a single pixel in the framebuffer, we run into a potential problem
- If the object (or camera) moves a tiny amount, we may see drastic changes in the pixel color, as different texels will rapidly pass in front of the pixel center
- This causes a flickering problem known as *shimmering* or *buzzing*
- Texture buzzing is an example of *aliasing*





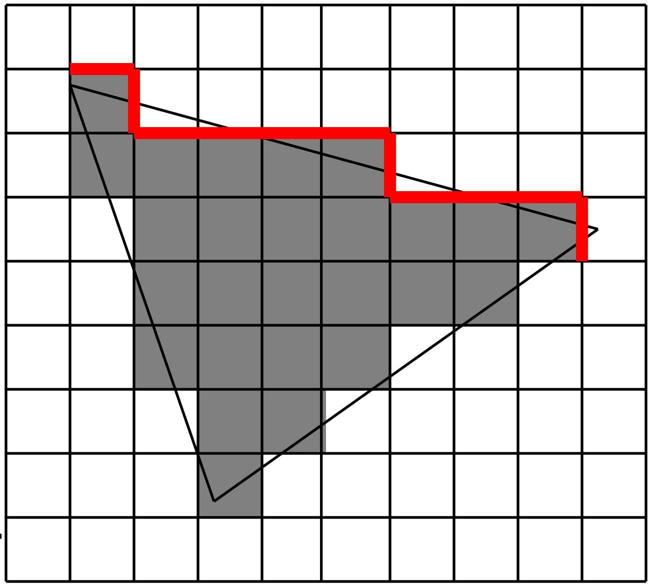
Small Triangles

- A similar problem happens with very small triangles
- If we shoot our a single ray right through the center of a pixel, then we are essentially *point sampling* the image
- This has the potential to miss small triangles
- If we have small, moving triangles, they may cause pixels to flicker on and off as they cross the pixel centers
- A related problem can be seen when very thin triangles cause pixel gaps
- These are more examples of *aliasing* problems



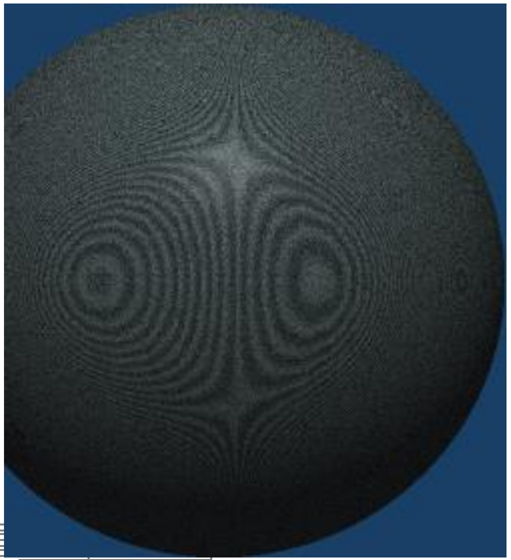
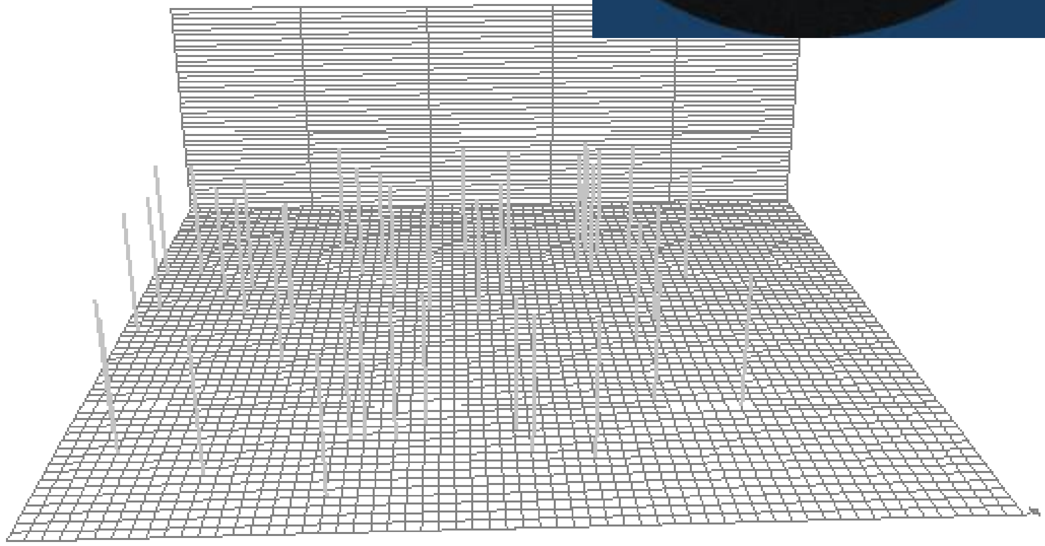
Stairstepping

- What about the jagged right angle patterns we see at the edges of triangles?
- This is known as the *stairstepping* problem, also affectionately known as "*the jaggies*"
- These can be visually distracting, especially for high contrast edges near horizontal or vertical
- Stairstepping is another form of *aliasing*



Moiré Patterns

- When we try to render high detail patterns with a lot of regularity (like a grid), we occasionally see strange concentric curve patterns forming
- These are known as Moiré patterns and are another form of aliasing
- You can actually see these in real life if you hold two window screens in front of each other

The Propeller Problem

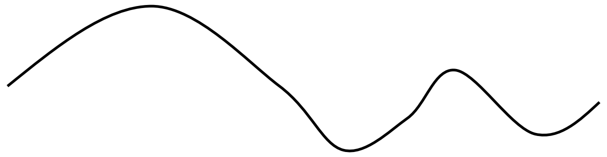
- Consider an animation of a spinning propeller, that is rendering at 30 frames per second
- If the propeller is spinning at 1 rotation per second, then each image shows the propeller rotated an additional 12 degrees, resulting in the appearance of correct motion
- If the propeller is now spinning at 30 rotations per second, each image shows the propeller rotated an additional 360 degrees from the previous image, resulting in the appearance of the propeller sitting still!
- If it is spinning at 29 rotations per second, it will actually look like it is slowly turning backwards
- These are known as *strobing* problems and are another form of *aliasing*

Aliasing

- These examples cover a wide range of problems, but they all result from essentially the same thing
- In each situation, we are starting with a *continuous signal*
- We then *sample* the signal at *discreet points*
- Those samples are then used to *reconstruct* a new signal, that is intended to represent the original signal
- However, the reconstructed signals are a false representation of the original signals
- In the English language, when a person uses a false name, that is known as an *alias*, and so it was adapted in *signal analysis* to apply to falsely represented signals
- Aliasing in computer graphics usually results in visually distracting *artifacts*, and a lot of effort goes into trying to stop it. This is known as *antialiasing*

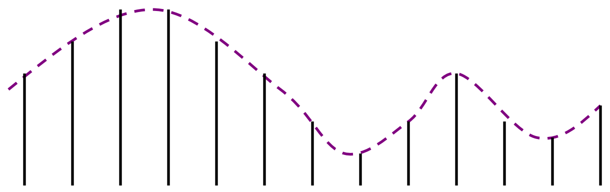
Signals

- The term *signal* is pretty abstract, and has been borrowed from the science of *signal analysis*
- Signal analysis is very important to several areas of engineering, especially electrical, audio, and communications
- Signal analysis includes a variety of mathematical methods for examining signals such as Fourier analysis, filters, sampling theory, digital signal processing (DSP), and more
- In electronics, a one dimensional signal can refer to a voltage changing over time. In audio, it can refer to the sound pressure changing over time
- In computer graphics, a one dimensional signal could refer to a horizontal or vertical line in our image. Notice that in this case, the signal doesn't have to change over time, instead it varies over space (the x or y coordinate)
- Often signals are treated as functions of one variable and examples are given in the 1D case, however the concepts of signal analysis extend to multidimensional signals as well, and so we can think of our entire 2D image as a signal



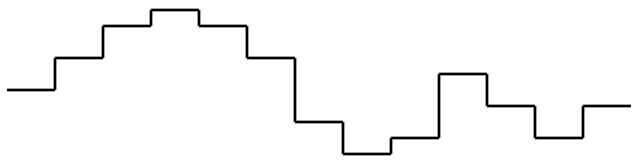
Sampling

- If we think of our image as a bunch of perfect triangles in continuous (floating point) device space, then we are thinking of our image as a *continuous signal*
- This continuous signal can have essentially infinite resolution if necessary, as the edges of triangles are perfect straight lines
- To render this image onto a regular grid of pixels, we must employ some sort of *discreet sampling* technique
- In essence, we take our original continuous image and sample it onto a finite resolution grid of pixels
- If our signal represents the red intensity of our virtual scene along some horizontal line, then the sampled version consists of a row of discreet 8 bit red values
- This is similar to what happens when a continuous analog sound signal is digitally sampled onto a CD



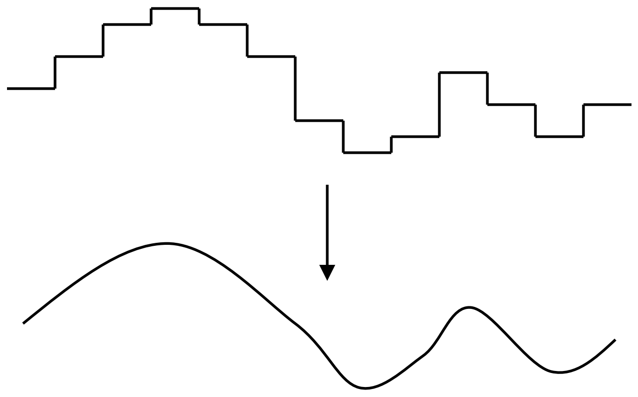
Reconstruction

- Once we have our sampled signal, we then *reconstruct* it
- In the case of computer graphics, this reconstruction takes place as a bunch of colored pixels on a monitor
- In the case of CD audio, the reconstruction happens in a DAC (digital to analog converter) and then finally in the physical movements of the speaker itself



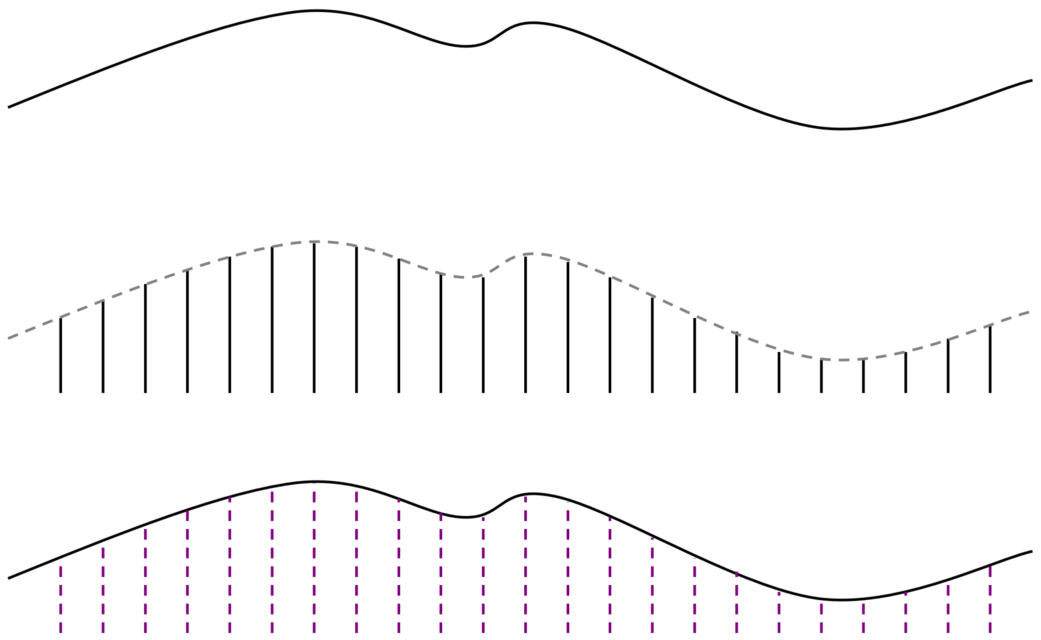
Reconstruction Filters

- Normally, there is some sort of additional *filtration* that happens at the reconstruction phase
- In other words, the actual pixels on the monitor are not perfect squares of uniform color. Instead they will have some sort of color distribution
- Additional filtration happens in the human eye so that the grid of pixels appears to be a continuous image
- In audio, the perfect digital signal is filtered first by the analog electronic circuitry and then by the physical limitations of the speaker movement



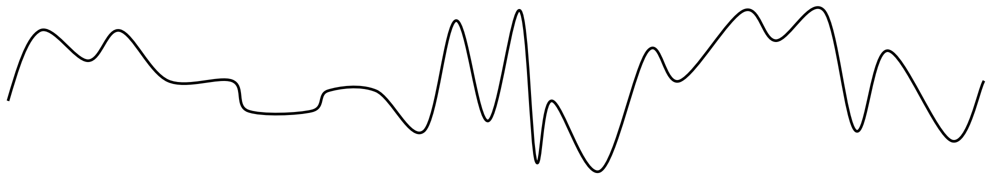
Low Frequency Signals

- Original signal
- Point sampled at relatively high frequency
- Reconstructed signal

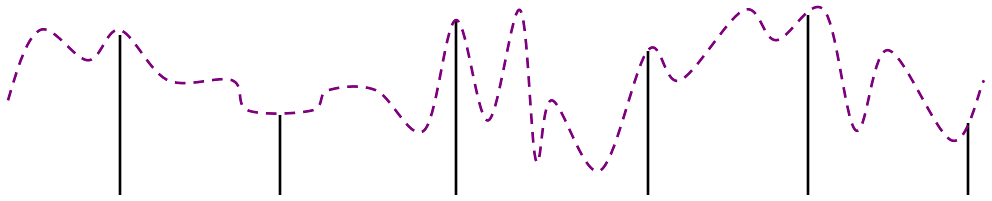


High Frequency Signals

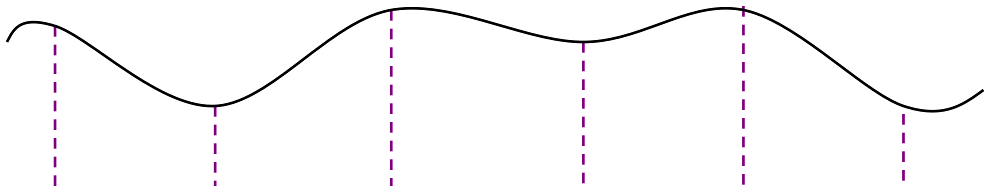
- Original signal



- Point sampled at relatively low frequency

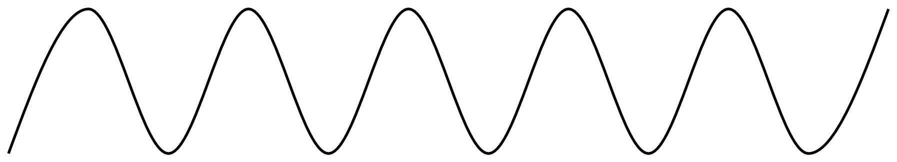


- Reconstructed signal

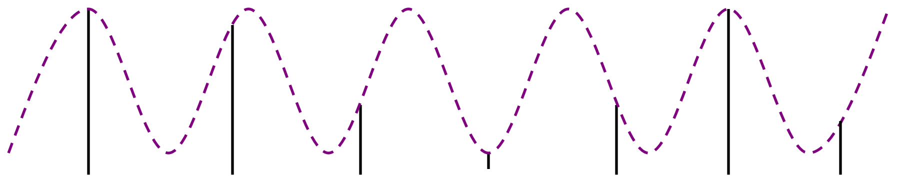


Regular Signals

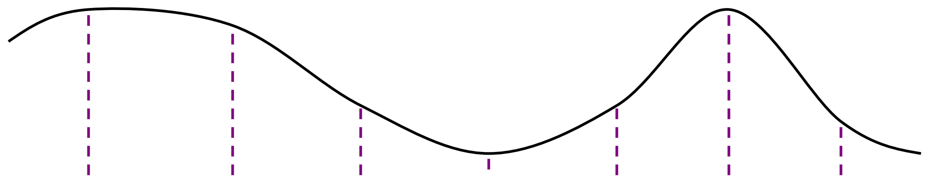
- Original repeating signal



- Point sampled at relatively low frequency



- Reconstructed signal repeating at incorrect frequency



Nyquist Frequency

- Theoretically, in order to adequately reconstruct a signal of frequency *x*, the original signal must be sampled with a frequency of greater than 2*x*
- This is known as the *Nyquist frequency* or *Nyquist limit*
- However, this is assuming that we are doing a somewhat idealized sampling and reconstruction
- In practice, it’s probably a better idea to sample signals at a minimum of 4*x*

Aliasing Problems

- Shimmering / Buzzing:

  Rapid pixel color changes (flickering) caused by high detail textures or high detail geometry. Ultimately due to point sampling of high frequency color changes at low frequency pixel intervals

- Stairstepping / Jaggies:

  Noticeable stairstep edges on high contrast edges that are nearly horizontal or vertical. Due to point sampling of effectively infinite frequency color changes (step gradient at edge of triangle)

- Moiré patterns:

  Strange concentric curve features that show up on regular patterns. Due to sampling of regular patterns on a regular pixel grid

- Strobing:

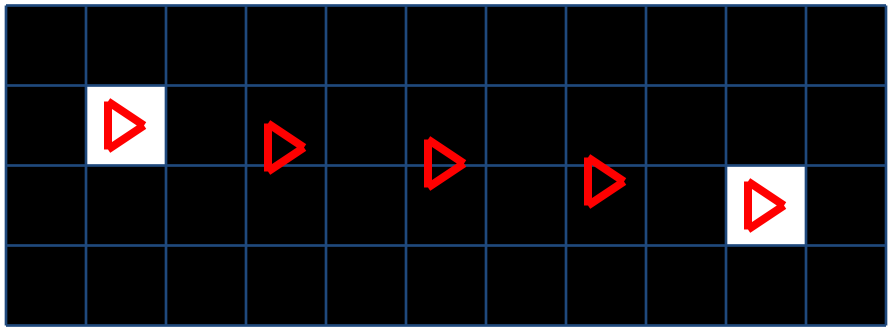
  Incorrect or discontinuous motion in fast moving animated objects. Due to low frequency sampling of regular motion in regular time intervals

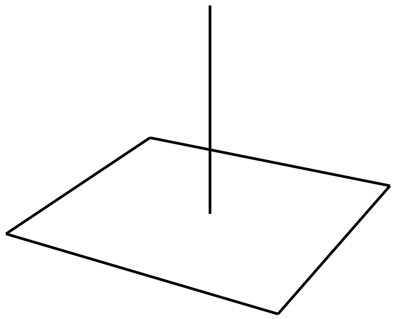
Spatial / Temporal Aliasing

- Aliasing shows up in a variety of forms, but usually those can be separated into either *spatial* or *temporal* aliasing
- Spatial aliasing refers to aliasing problems based on regular sampling in space. This usually implies device space, but we see other forms of spatial aliasing as well
- Temporal aliasing refers to aliasing problems based on regular sampling in time
- The antialiasing techniques used to fix these two things tend to be very different, although they are based on the same fundamental principles

Point Sampling

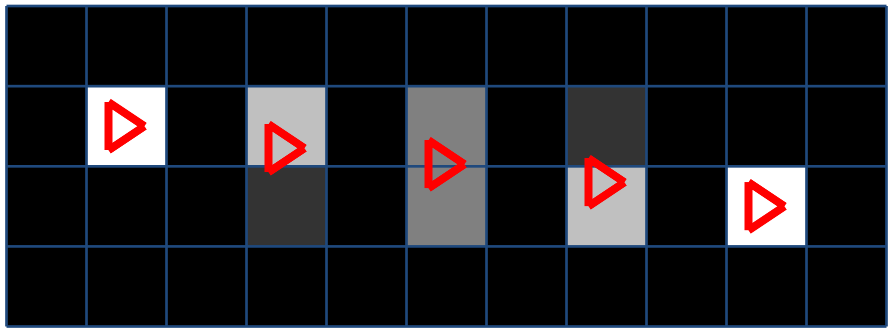
- The aliasing problems we’ve seen are due to low frequency *point sampling* of high frequency information
- With point sampling, we sample the original signal at precise points (pixel centers, etc.)
- Is there a better way to sample continuous signals?

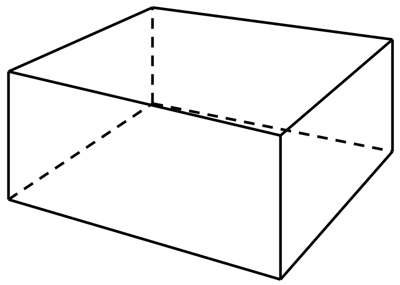




Box Sampling

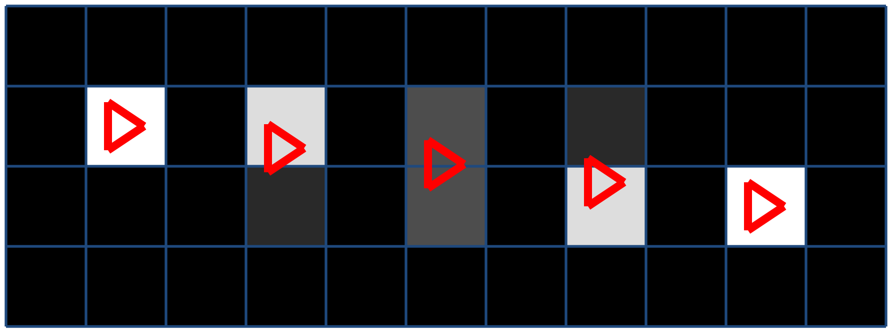
- We could also do a hypothetical *box sampling* (or *box filter*) of our image
- In this method, each triangle contributes to the pixel color based on the area of the triangle within the pixel
- The area is equally weighted across the pixel

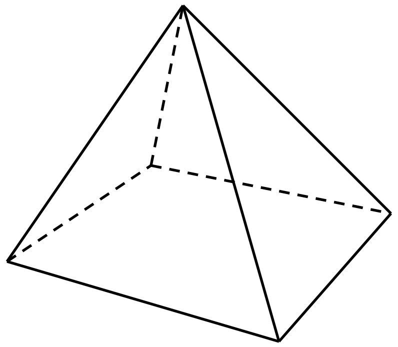




Pyramid Sampling

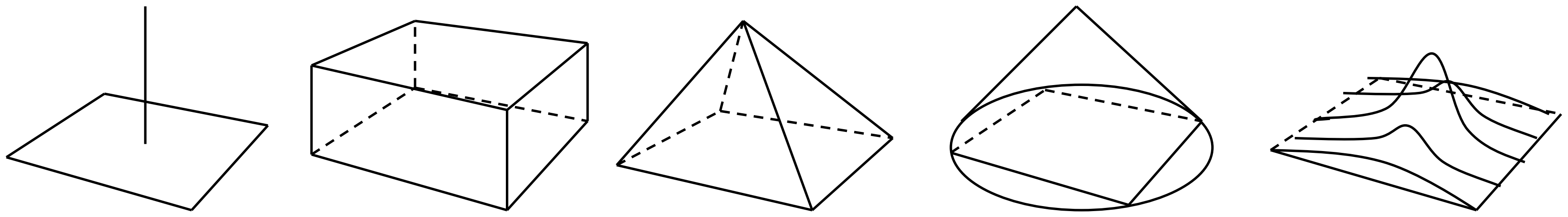
- Alternately, we could use a weighted sampling filter such as a pyramid filter
- The pyramid filter considers the area of triangles in the pixel, but weights them according to how close they are to the center of the pixel



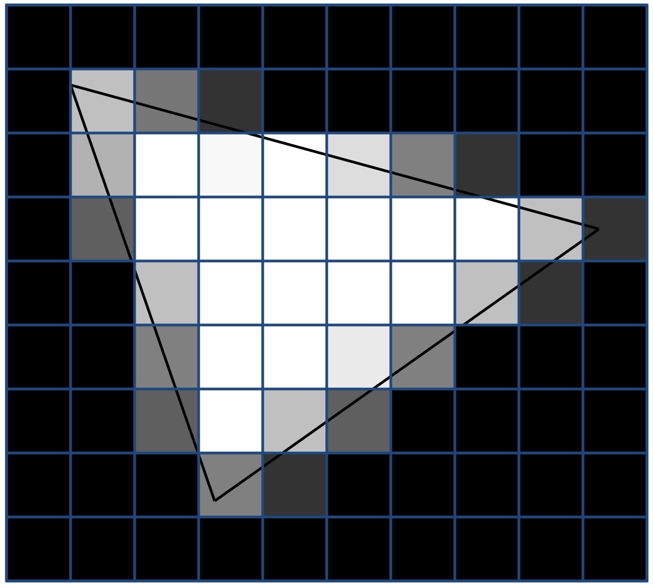


Sampling Filters

- We could potentially use any one of several different sampling filters
- Common options include the point, box, pyramid, cone, and Gaussian filters
- Different filters will perform differently in different situations, but the best all around sampling filters tend to be Gaussian in shape
- The filters aren't necessarily limited to cover only pixel. It is possible, and not uncommon to use filters that extend slightly outside of the pixel, thus overlapping with the neighboring pixels. Filters that cover less than the square pixel, however, tend to suffer from similar problems as point sampling



Edge Antialiasing

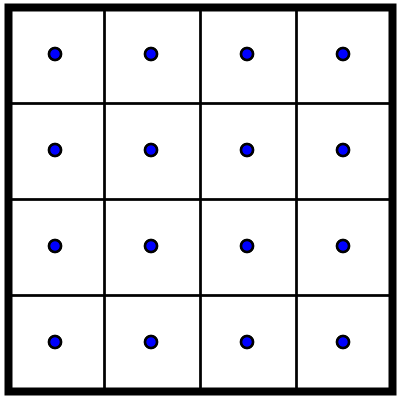


Supersampling

- The easiest way to improve the antialiasing in a ray tracer is to trace more than one ray per pixel
- Instead of just point sampling the center of the pixel, we can trace several
- We refer to this process as *supersampling*
- For high quality edge-antialiasing, it is not uncommon to use 16 or more samples per pixel

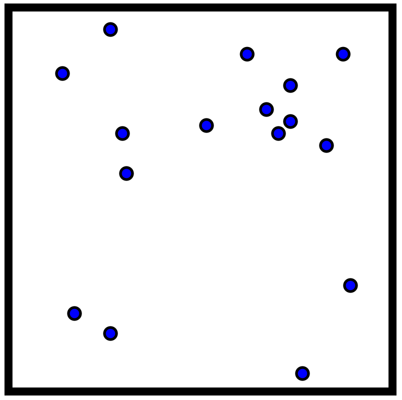
Uniform Sampling

- With *uniform sampling*, the pixel is divided into a uniform grid of *subpixels*
- Uniform supersampling should certainly generate better quality images than single point sampling
- It will filter out some high frequency information, but may still suffer from Moiré problems with highly repetitive signals



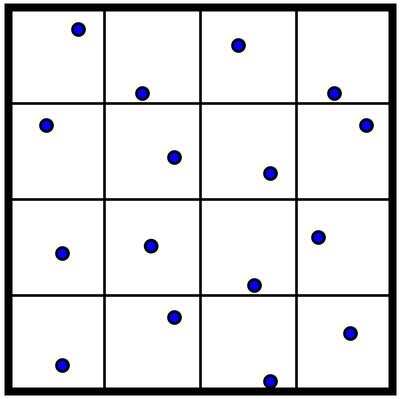
Random Sampling

- With *random sampling*, the pixel is supersampled at several randomly located points
- Random sampling has the advantage of breaking up repeating signals, and so can completely eliminate Moiré patterns. It does, however, trade the regular patterns with random *noise* in the image, which tends to be less annoying to the viewer
- It also suffers from potential clustering and gaps of the samples

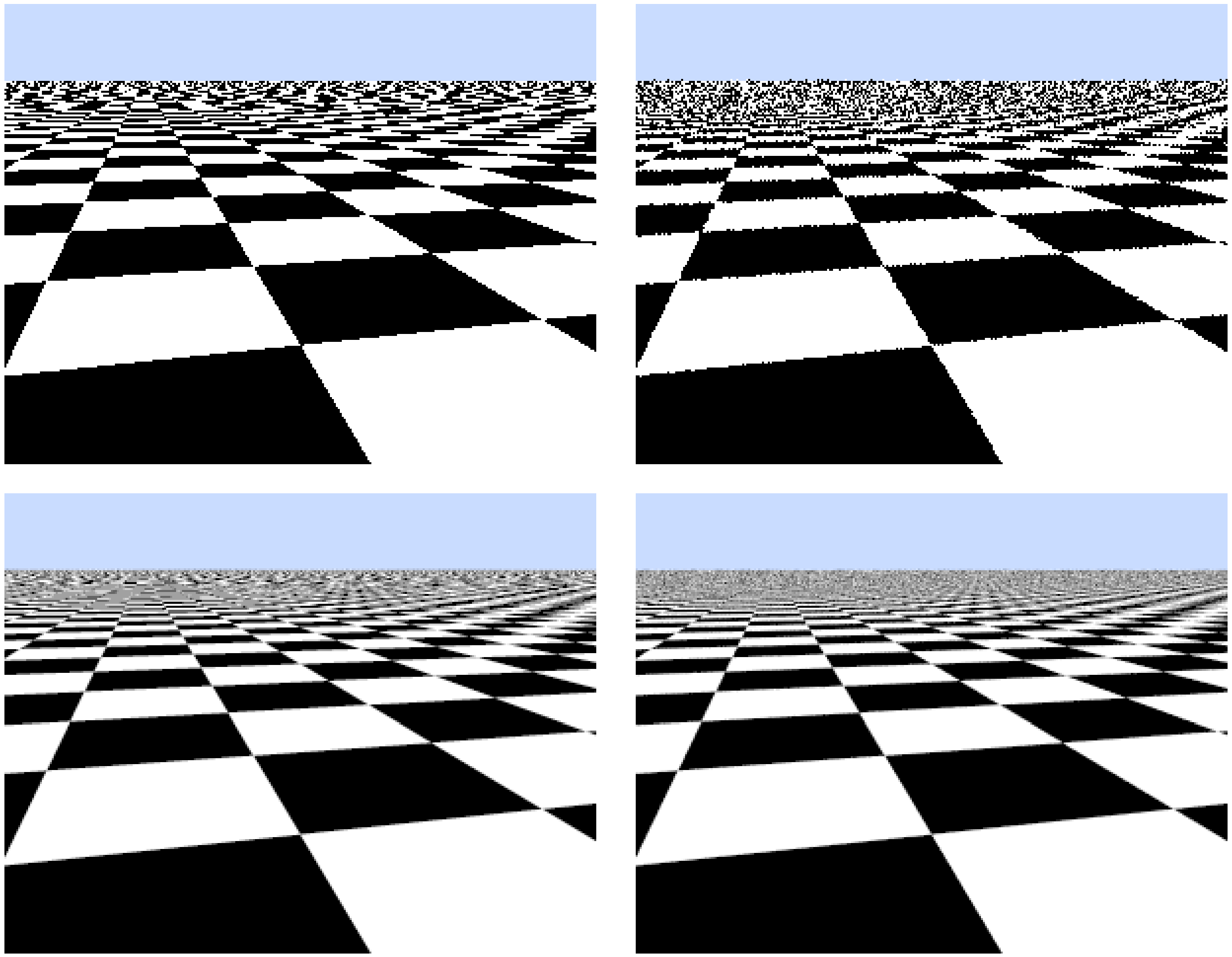


Jittered Sampling

- With *jittered* or *stratified sampling*, the pixel is divided into a grid of *subpixels*, but the subpixels themselves are sampled at a random location within the subpixel
- This combines the advantages of both uniform and random sampling

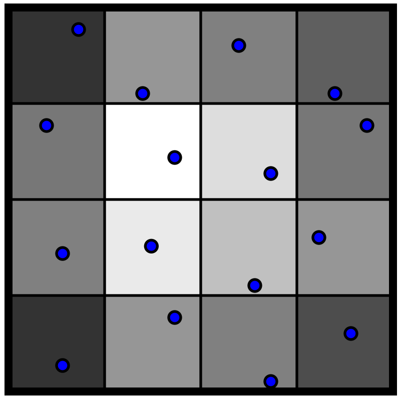


Jittered vs. Non-Jittered



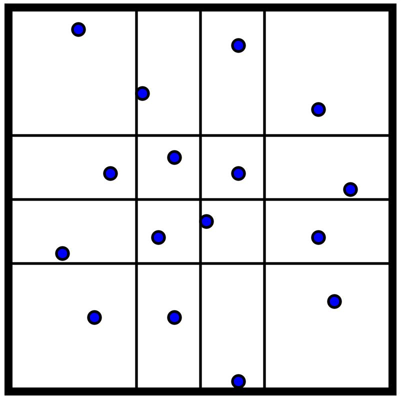
Weighted Sampling

- If we average all of the samples equally to get the final pixel color, we are essentially performing a box filter on the samples
- We can also perform a weighted average of the samples to achieve other shaped filters
- For example, we can weight the samples according to a box, cone, pyramid, or Gaussian shape if desired
- We can apply weighting to uniform, random, or jittered supersamples with little additional work



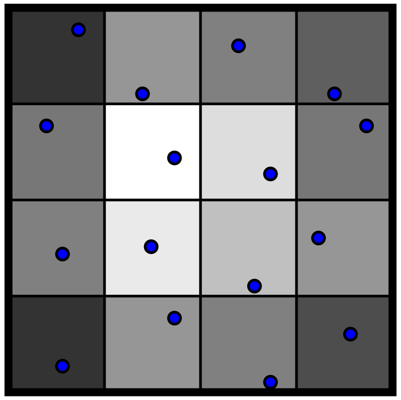
Weighted Distribution

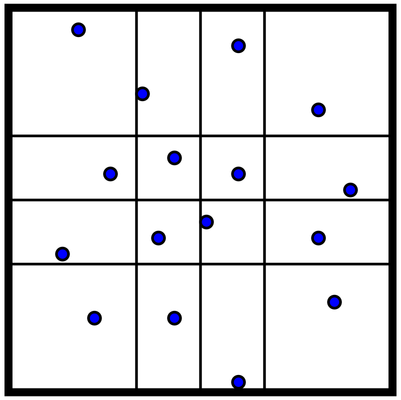
- Another option to achieve apply a weight (such as Gaussian) to a pixel is to modify the *distribution* of the samples
- In the case of a Gaussian weight, we could either take a uniform distribution and weight the individual samples or we could apply the weighing to the spacing of the samples and weight the samples themselves equally
- Which approach is better?



Weighted Samples vs. Distributions

- If we look at the 16 samples in the left image, we see that some are much more important than others, yet they all have the same computational cost
- In other words, the 4 samples in the center of the grid might have more total weight than the other 12 samples around the perimeter
- By adjusting our distribution so there are more samples in the higher valued areas, we can achieve the benefits of jittered and weighted sampling while maintaining efficiency by treating all samples equally
- This is an important concept that we will come back to again and again…





Weighted Distributions

- Let's say we start with two random numbers *s* and *t* that are uniformly distributed in the [0…1] interval
- We want to generate two new random numbers *s'* and *t'* in the same interval but with some type of weighting towards the center
- Gaussian:

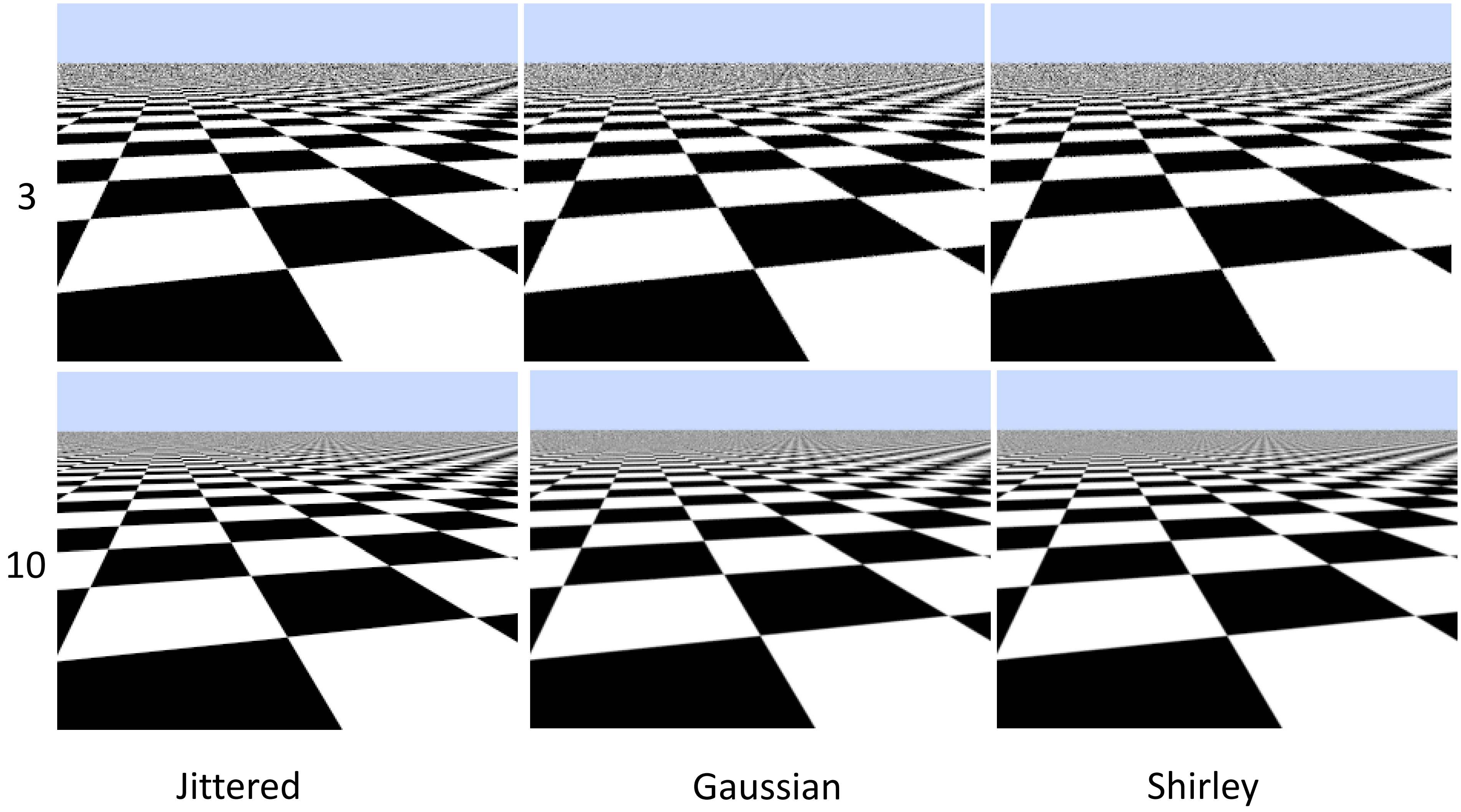
$$\begin{aligned} a &= \mu\sqrt{-2\log s} \qquad (\mu \approx 0.4) \\ b &= 2\pi t \\ s' &= 0.5 + a \cdot \sin b \\ t' &= 0.5 + a \cdot \cos b \end{aligned}$$

- Shirley:

$$s' = \begin{cases} -0.5 + \sqrt{2s} & if\ s < 0.5 \\ 1.5 - \sqrt{2 - 2s} & if\ s \geq 0.5 \end{cases}$$

$$t' = \begin{cases} -0.5 + \sqrt{2t} & if\ t < 0.5 \\ 1.5 - \sqrt{2 - 2t} & if\ t \geq 0.5 \end{cases}$$

Weighted Distributions



Adaptive Sampling

- Another approach to the problem is to perform *adaptive sampling*
- With this scheme, we start with a small number of samples and analyze their statistical variation
- It the colors are all similar, we accept that we have an accurate sampling
- If we find that the colors have a large variation, we continue to take further samples until we have reduced the statistical error to an acceptable tolerance
- Adaptive sampling schemes can be very efficient compared to brute force supersampling, but can also suffer from an important problem called *bias*

Sampling & Bias

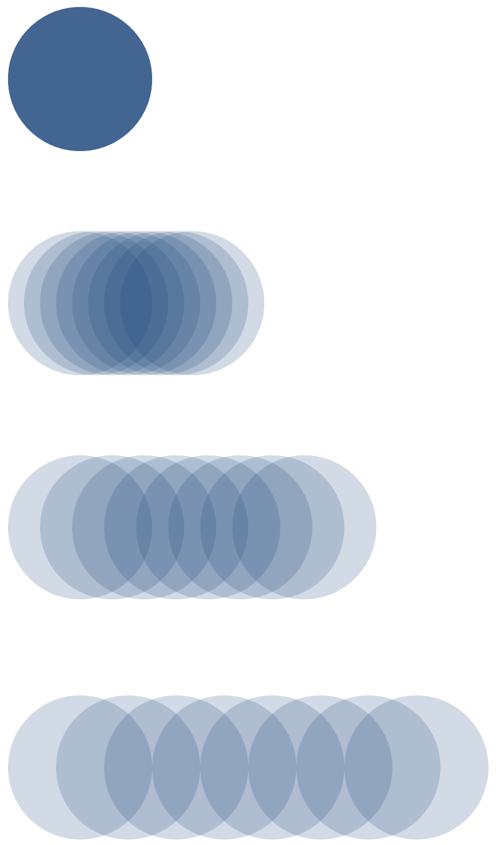
- When we do random sampling for antialiasing (and for other things like area lights), we are essentially estimating the value of a function
- The more random samples we take, the better our estimate of the result
- We say that as we take more random samples, our estimate *converges* to the actual value
- By taking a fixed number of supersamples, we generate an *unbiased* estimate that will converge to the correct value
- However, when we add decisions based on the results from previous samples (such as with the adaptive sampling methods), we can add a statistical *bias* to the estimate, resulting in a tendency towards either over or underestimating the resulting value
- Many rendering optimizations can add bias to the resulting image but its usually small enough to not be a problem, and the resulting performance improvements typically justify it
- There is however a tendency towards more brute force unbiased renderers these days as GPUs get faster and faster

Temporal Aliasing

- Properly tuned supersampling techniques address the spatial aliasing problems pretty well
- We still may run into temporal aliasing or strobing problems when we are generating animations
- Just as the spatial antialiasing techniques apply a certain blurring at the pixel level, temporal antialiasing techniques apply bluring at the frame level
- In other words, the approach to temporal antialiasing is to add *morion blur* to the image

Motion Blur

- Motion blur can be a tricky subject, and several different approaches exist to address the issue
- One way to incorporate it into a ray tracer is to randomly distribute the rays in time!
- Let’s assume that every object in the scene has two matrices- one for its position/orientation at the beginning of the frame and one for the end of the frame
- The camera has an initial and final matrix as well
- Each ray shot from the camera is randomly distributed in time and computes the appropriate interpolated matrix when it traverses through the scene
- This can actually be done pretty efficiently by modifying the Instance and Camera classes



Combining Antialiasing Techniques

- We can even combine the techniques of pixel antialiasing and motion blur without exponentially increasing the work
- This can be done by rendering a fixed number of supersamples total, each one spread in time and jittered at the pixel level
- We can also combine this with the area lighting samples from an earlier lecture. Previously, we needed hundreds of shadow rays per camera ray, but as we are now generating dozens of camera rays, we can reduce this to just a few shadow rays per camera ray
- This overall approach offers a powerful foundation for other blurry effects such as: soft shadows (penumbrae), lens focus (depth of field), color separation (dispersion), glossy reflections, diffuse interreflections, etc.