

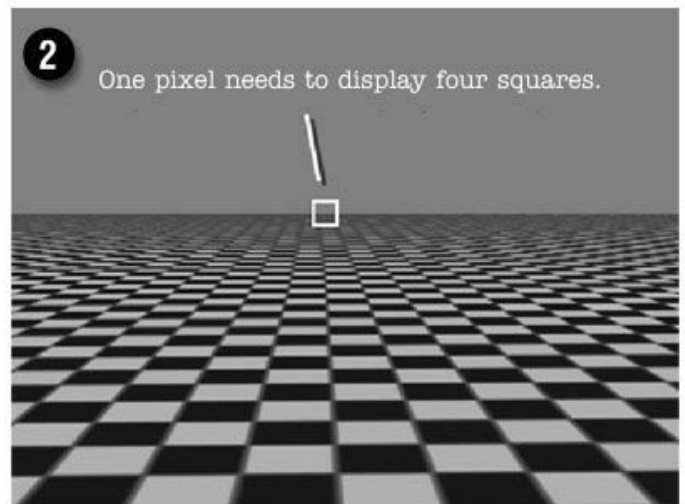
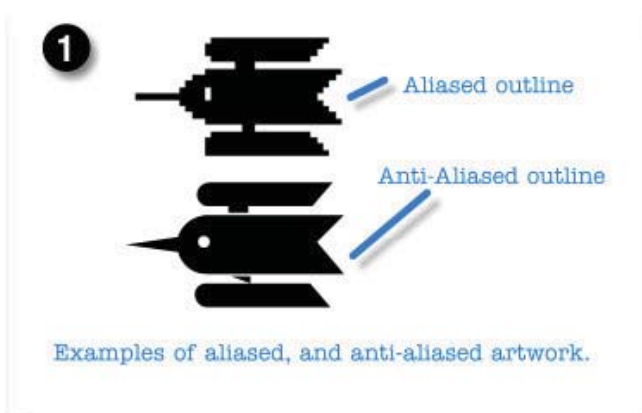
Anti-Aliasing and Resampling

The fundamental fundamentals of bitmap images and anti-aliasing are a fair enough topic for beginners and it's not a bad read for those of us who've been around the pixel once or twice. First in this article, lets discuss the procedure by which art applications disguise the reality that diagonal lines and curves cannot be faithfully, artistically represented using rectangular pixels – "helper pixels" are required around the edge of such shapes. This information sheet will take you through how anti-aliasing works, whether you're drawing a line or resizing a bitmap graphic.

"Aliasing" is the false presentation of visual data and is the result of an application rendering an image area without enough visual information. In the first illustration, you can see a spaceship at the top with an aliased outline – there are supposed to be curved and smooth diagonal lines around its outline, but instead there are stairsteps. "Anti-aliasing", shown applied to the same spaceship at the bottom, is a method for accurately representing image data.

Image Resolution and Granularity

To better demonstrate the effects of aliasing and anti-aliasing, next there is a visual example of a checkerboard that extends into the horizon. To remove user input and simply show how an application handles anti-aliasing, let's say that this checkerboard scene is being rendered in a modeling application, in which you define the scene and the application does the rendering work. The squares that are closest to the viewer are white or black; there is no ambiguity about the color of the large squares. However, as the squares diminish in size towards the horizon, each square is perceived using a smaller number of photo-receptors in the eye, until the horizon appears to be a solid tone instead of alternating colors. Your eye cannot distinguish clearly which squares are white and which are black, because the "granularity" – the number of photo-receptors in your eye – is a fixed amount.



How can a single image pixel faithfully represent more than one color?

When this checkerboard scene is rendered by a computer application to bitmap format, the application performing the rendering has two choices to make: to alias the horizon – to choose either white or black at any given pixel – or to anti-alias – to average the color for a given pixel, to create an image area closer to the way a human eye would see the actual scene. In the illustration above, imagine that the square next to the call-out is a single pixel, a pixel whose size cannot change and can contain only one color.

The visual content at this point in the picture, however, consists of *more* than one color – there can be a number of white and black squares close to the horizon – with only a single image pixel to represent them. So, what's it going to be: a black or a white pixel? To make such a decision is called aliasing; if you fill this pixel with white, you're negating the black squares within this sample area. To reconcile the impossibility of filling a single pixel with more than one color, anti-aliasing of the scene fills the pixel with a shade of black, because in reality the pixel sample area should contain a blend of both black and white sample information. Anti-aliasing can be added to artwork by most graphics applications on three occasions:

1. When you make a brush stroke.
2. When a brush stroke is made for you, as with a modeling/rendering application.
3. When pixels are added or deleted from an image. This is called "resampling".

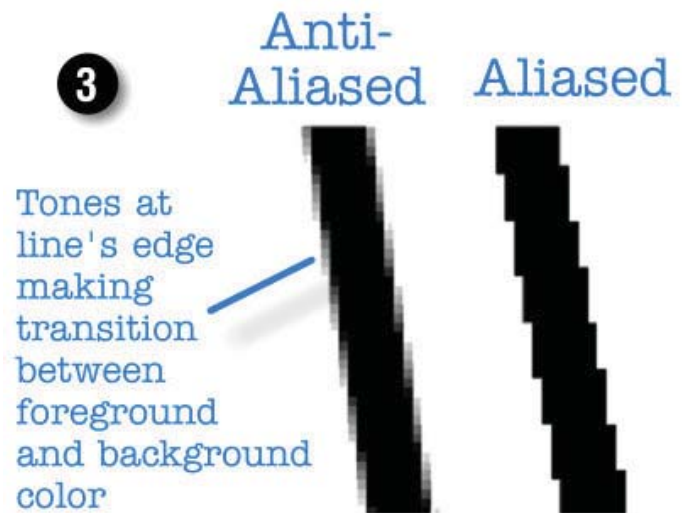
Anti-aliasing does more than simply reconcile pixel colors when the image information is too large to fit in a single pixel. Let's now take a look at how curves and diagonal lines – geometry that cannot realistically be displayed on a monitor – can be made smooth in appearance by anti-aliasing. Note: "super-sampling" is a term used in modeling applications to describe yet another type of anti-aliasing. The mechanism for super-sampling works like this: the user defines a specific size for the scene to be rendered. The application images the scene at two

times the requested size, holds the image in memory, and then creates the image at the requested size while averaging tones for the image's pixels from the larger image in memory. This super-sampling process can use more than one image in memory, so you could request that an 8x image, a 4x and a 2x image should be used to average and calculate final pixel colors.

Anti-aliasing and Brush Strokes

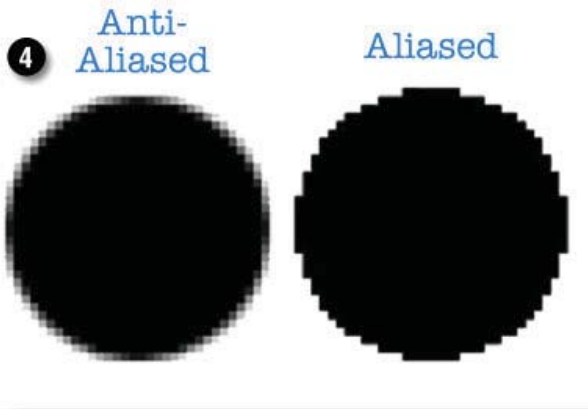
Curves and diagonal lines are particularly difficult for a monitor to display with image fidelity, because a monitor, and the pixel framework of digital images, have no mechanism for displaying anything beyond a rectangular quiltwork of image elements. To keep the edges of these geometric shapes smooth in appearance, anti-aliasing is used by applications to place pixels of different opacity along "problem areas" of curves and diagonal lines.

In the illustration below you can see a pair of diagonal lines; the one at left has several pixels of different opacity that "fill in" the abrupt edges where the line is not perfectly parallel to the grid of pixels that make up the image. At right, an aliased version of the diagonal line shows harsh, unappealing "stairsteps".



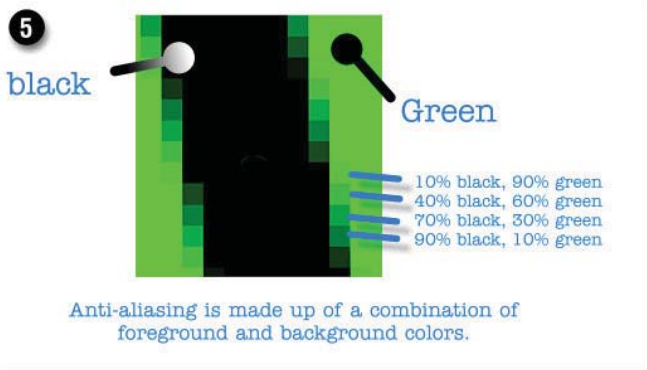
A fair question at this point would be, "How does the application *know* where to put the different anti-aliased pixels? The answer lies in averaging image area tones, and interpolating the correct shade of pixel

to lie on the edge of the line or curve. In the next illustration you can see a close-up of a rounded shape, with and without anti-aliasing. At a close view, the outline of the anti-aliased shape looks fuzzy, but at 1:1 viewing resolution the curve is both crisp and soft.



Anti-aliasing interpolates—averages—the correct edge color for round shapes.

If you zoom very closely on a diagonal line with anti-aliased edges, you will observe the following phenomenon: pixels at the edge of the line gradually are composed of less line color and more image background color the farther a pixel is from the line. Below you can see callouts for the percentage mixture of anti-aliased pixels on the edge of a vertical line.



Anti-aliasing is made up of a combination of foreground and background colors.

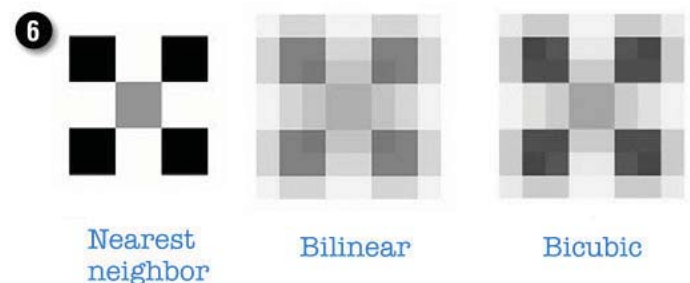
The idea behind anti-aliasing a shape you create is to make a smooth transition between the interior of a shape and its background. Besides the anti-aliasing that goes on in modeling and painting programs, there is a third type of anti-aliasing that occurs when you resize a bitmap image. Next month I'll cover the different sorts of interpolation

("interpretation") graphics apps use to make the resizing of an image come out more smoothly. I'll also tell you what sort of image editor you might want to buy, to make sure that product X can do bicubic interpolation, for example.

Interpolation and Averaging

Suppose you have a beautiful, miniature painting that is made up of only nine pixels, three pixels on a side. You decide that you want to make the image twice its original size (six pixels on a side), making it 36 pixels in total. There are three methods by which an application can "think up" new pixels to go into your image:

1. by creating pixels that are the nearest neighbor in color to the original pixel.
2. by sampling surrounding pixels in both a horizontal and vertical direction, and then creating a color average of the total sums for new pixels.
3. by sampling pixels in horizontal, vertical, and diagonal directions, and using a weighted average of the total colors for any given new pixel. Adobe Photoshop calls these three methods of interpolation *nearest neighbor*, *bilinear*, and *bicubic interpolation*. Other applications might have different names for these interpolations, but these are the three names we'll use here, as shown in the illustration below.



Nearest neighbor

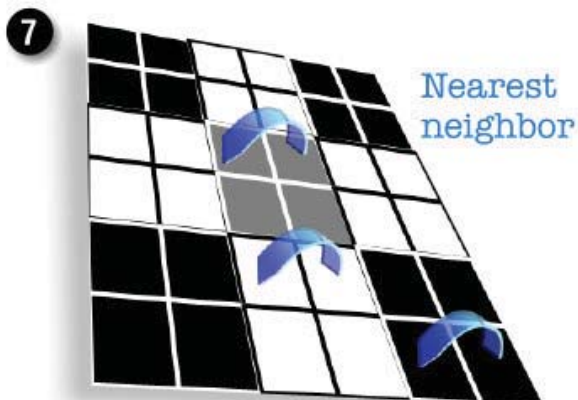
Bilinear

Bicubic

Nearest Neighbor

Nearest neighbor calculation isn't actually a method of interpolation. An application chooses the same color value for neighboring pixels in an enlarged image as the original color found at any given pixel. Therefore, if the center of our hypothetical 3 by 3-pixel

painting is 50% black, and we enlarge the painting 200% through nearest neighbor calculation, the center of the painting would contain 4 pixels that are 50% black. No anti-aliasing is produced using nearest neighbor calculations because no color averaging – no new pixel colors – are added to the new image. You can see how this calculation is performed by a program in the illustration below. Nearest neighbor calculation is perfectly fine if your composition is rectangular in content, and you enlarge the image in multiples of two (200%, 400%, and so on). However, a more sophisticated procedure is required when working with large images that contain visual content which is organic and/or photographic, and you want to enlarge or reduce the image to, say, 148% of its original size. Most modern image editing applications offer something called *bilinear interpolation*, and the following section describes how it works.

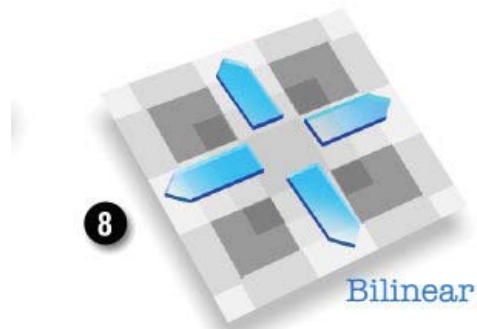


7 Nearest Neighbor calculation for new pixels uses the simplest of math for reassigning picture elements.

Bilinear Interpolation

A single pixel in a painting that is enlarged using bilinear filtering gets its destination color from the top, left, right and bottom of its original position. These color readings of neighboring pixels are added together, divided by four, and the resulting color is applied to the new pixels in the image. In the next illustration, you can see how bilinear interpolation “looks” for new color information in our 3 by 3-pixel painting. Bilinear interpolation produces anti-aliased pixels within the new, enlarged image, because whole number values for new pixels are not possible. The resulting new pixels should, from an artistic as well as a mathematical point

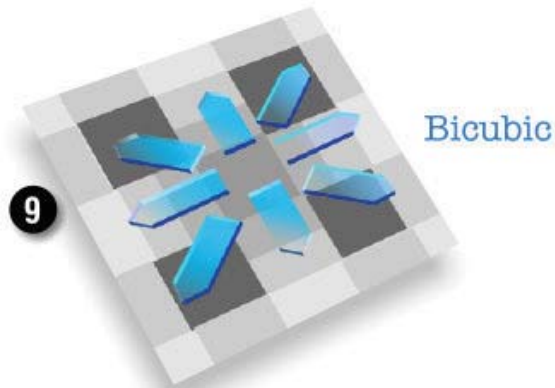
of view, represent *blends* or percentages, of neighboring pixels. Although bilinear interpolation produces good anti-aliased pixels, it is not the most sophisticated interpreting method for resizing images. *Bicubic interpolation* is covered in the following section.



8 Bilinear Interpolation creates new pixels based on color averages from both the horizontal and vertical neighbors of the area to be resized.

Bicubic Interpolation

If we think of the nearest neighbor assignment of pixels as being a one-dimensional sampling technique, then bilinear interpolation would be a square function – it looks across two dimensions for pixel data. Bicubic interpolation goes one step further in calculating new pixels. In this case, a pixel is assigned a new value based upon information taken in horizontal, vertical and diagonal directions, and then the sum is averaged with a preference for the predominating tones in the area, resulting in a weighted average. Bicubic interpolation is the most processor-intensive resizing method because it involves the most calculations, but the aesthetic results are also the most faithful to the original image. Whenever you resample an image, there will be some loss of focus within the image, but bicubic interpolation provides the sharpest of any method for creating new image data, or deleting and reassigning pixel colors. The illustration below shows how bicubic interpolation works. Of the three methods for creating or deleting pixels during resizing, only bilinear and bicubic interpolation produce anti-aliasing. If your computer has the horsepower, and your application offers interpolation choices, choose bicubic for the best results and then apply minor sharpening filtering, if you feel it’s needed.



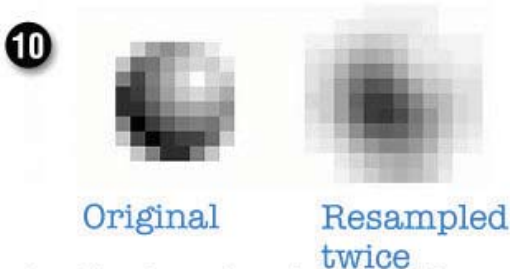
Bicubic interpolation is the most artistically pleasing method for resampling an image.

Summary

Anti-aliasing is necessary for creating refined artwork, because pixels are rectangular in shape while the content of your artwork most likely is not rectangular. Anti-aliasing also reconciles the impossibility of a single pixel having more than one color value, by averaging the colors into a composite tone. Finally, anti-aliasing intelligently reassigns image pixel colors when you enlarge or reduce an image, so there are no abrupt color transitions creating artifacts in your work.

Progressive Changes and Anti-Aliasing

There is a problem with resampling an image, or image area, too many times, and this only partially has to do with anti-aliasing. Pixel-based images are constructed of a finite number of placeholders and whenever you change the total number of pixels, you create a change in the artwork. This type of change is a progressive one – as you resample, you build change upon change and there is no real path back to your original design. In the next illustration, you can see a close-up of a very small sphere design at the left. At the right, the sphere has been resized (resampled) twice and the result is an out-of-focus blob. You will not witness such a great amount of deterioration when you resample larger images, but the change in pixel count will still tend to throw the design out of focus, due to the averaging the application performs to add or discard pixels. The best strategy for keeping images you resample crisp in appearance is to know in advance what the final size of the image should be, and allow the application to interpolate the selected area only once.



Regardless of resampling techniques, multiple resampling progressively deteriorates the sharpness of an image.