IV.5 WHAT ARE THE COORDINATES OF A PIXEL?

Paul S. Heckbert
University of California
Berkeley, California

Did you ever get confused deciding whether pixels were centered at integer coordinates or centered halfway between the integers? We present here a consistent answer to this question.

When modeling, we use real numbers, but pixels have integer coordinates, so somewhere during rendering we must quantize the coordinates to integer values. How, exactly, do we perform this mapping? *Do we round or truncate?* Consistency is vital, but making the better choice is also important. This may seem like a petty question, but failure to address it can lead to misalignment, gaps or overlap between objects, or edge effects at the screen border. The question is especially important if we are anti-aliasing.

To clarify the problem, we distinguish between discrete images and continuous images, and also between discrete coordinates and continuous coordinates. A *discrete image is* an array of pixels, the sort of image with which we're familiar in computer graphics and image processing, and a *continuous image is* a function defined over a continuous domain, as in optics or the real world. In computer graphics we take a geometric description of a continuous image (for example, a list of polygons with floating point coordinates) and sample it at a discrete array of points to create a discrete image. The discrete image we render is an approximation of the continuous image. We call the coordinates in the discrete image *discrete coordinates* and the coordinates in the continuous image *continuous coordinates.* Discrete coordinates take on integer values at the sample points, which are the pixel centers. The mapping question is now reduced to a choice of phase (displacement) between continuous and discrete coordinates.

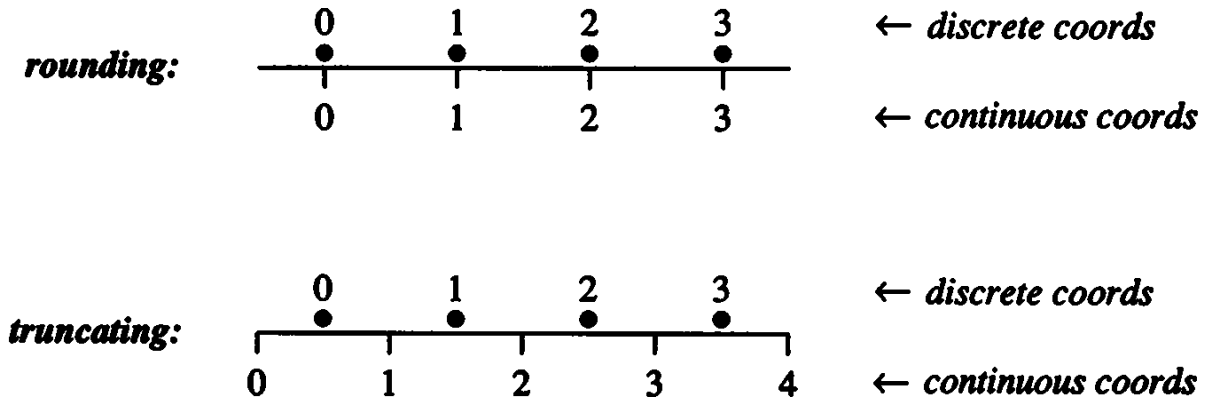

Figure 1. Rounding and truncating schemes for coordinate conversion.

If we round when converting floating point continuous coordinates to discrete coordinates, this is equivalent to aligning the continuous and discrete axes. Figure 1 shows the rounding mapping at top, for a hypothetical frame buffer four pixels on a side, with pixel centers marked by bullets. Rounding seems attractive at first, because continuous coordinates and discrete coordinates are equal. Unfortunately, the continuous range corresponding to our hypothetical frame buffer, using rounding, is the awkward range of – .5 to 3.5.

The better mapping choice is truncation, or more accurately, flooring, where continuous coordinates are converted to discrete coordinates by taking the floor function. In this scheme, there is a half-pixel phase shift between continuous coordinates and discrete coordinates as can be seen in Fig. 1, bottom. Continuous coordinates take on integer values halfway between pixels. The pixel with discrete coordinates (x, y) has its center at continuous coordinates (x + 1/2, y + 1/2). Assuming as a first approximation that we reconstruct using a one-pixel-wide box filter, the continuous coordinate domain of pixel (x, y) is from x to x + 1 in x and from y to y + 1 in y. For our hypothetical frame buffer, the continuous coordinate range using truncation is 0 to 4—simpler numbers than with rounding. The simplicity of the coordinate range facilitates image scaling and other transformations.

In summary, both continuous and discrete coordinates have their place. Continuous coordinates are most appropriate when modeling, that is, when one is concerned with geometry, not with pixels. Discrete coordinates are most useful when working close to the pixel level, as in scan

conversion or image processing. Note that discrete coordinates are not always integers: it is often useful to use floating point variables for discrete coordinates. When writing graphics software it is vital to be conscious of whether you are using continuous coordinates or discrete coordinates.

To convert from continuous to discrete or vice versa, where c is the continuous coordinate and d is the discrete coordinate,

$$d = floor\ (c)$$

$$c = d = \tfrac{1}{2}.$$

I developed the above dualist view of pixel coordinates while working on an image zoom algorithm at Xerox PARC in 1988. Thanks also to Alvy Ray Smith at Pixar for reinforcing my reverence for the pixel.

See also A Digital “Dissolve” Effect (221); Circles of Integral Radius on Integer Lattices (57); Precalculating Addresses for Fast Fills, Circles, and Lines (285)