The visual variables are used for communication, by encoding data and drawing distinctions between visual

elements. But the visual variables have different characteristics. Before you choose a visual variable to express

some distinction, you should make sure that the visual variable’s properties match your communication. For

example, you could display a temperature using any of the dimensions: position on a scale, length of a bar,

color of an indicator, or shape of an icon (a happy sun or a chilly icicle). Your choice of visual variable will

strongly affect how your users will be able to perceive and use the displayed data.

Two characteristics of visual variables are the kind of

scale

and the

length

of the scale.

A

nominal

scale is just a list of categories. Only comparison for equality is supported by a nominal scale.

Different values have no ordering relationship. The shape variable is purely nominal. Hue is also purely

nominal, at least as a

perceptual

variable. Although the wavelength of light assigns an ordering to colors, the

human perceptual system takes no notice of it. Likewise, there may be some cultural ordering imposed on hue

(red is “hotter” than blue), but it’s weak, doesn’t relate all the hues, and is processed at a higher cognitive level.

An

ordered

scale adds an ordering to the values of the variable. Position, size, value, and to some extent

texture (with respect to the grain size of the texture) are all ordered.

With a

quantitative

variable, you can perceive the

amount

of difference in the ordering. Position is

quantitative. You can look at two points on a graph and tell that one is twice as high as the other. Size is also

quantitative, but note that we are far better at perceiving quantitative differences in one dimension (i.e., length)

than in two dimensions (area). Value is not quantitative; we can’t easily perceive that one shade is twice as

dark as another shade.

The

length

of a variable is the number of distinguishable values that can be perceived. We can recognize a

nearly infinite variety of shapes, so the shape variable is very long, but purely nominal. Position is also long,

and particularly fine-grained. Orientation, by contrast, is very short; only a handful of different orientations

can be perceived in a display before confusion starts to set in. The other variables lie somewhere in between,

with roughly 10 useful levels of distinction, although size and hue are somewhat longer than value and texture

Finally, simplify your designs by aligning elements horizontally and vertically. Alignment contributes to the

simplicity of a design. Fewer alignment positions means a simpler design. The dialog box shown has totally

haphazard alignment, which makes it seem more complicated than it really is.

Labels

(e.g. “Wait” and “Retry after”). There are two schools of thought about label alignment: one school

says that the left edges of labels should be aligned, and the other school says that their right edges (i.e., the

colon following each label) should be aligned. Both approaches work, and experimental studies haven’t found

any significant differences between them. Both approaches also fail when long labels and short labels are used

in the same display. You’ll get best results if you can make all your labels about the same size, or else break

long labels into multiple lines.

Controls

(e.g., text fields, combo boxes, checkboxes). A column of controls should be aligned on both the left

and the right. Sometimes this seems unreasonable -- should a short date field be expanded to the same length

as a filename? It doesn’t hurt the date to be larger than necessary, except perhaps for reducing its perceived

affordance for receiving a date. You can also solve these kinds of problems by rearranging the display, moving

the date elsewhere, although be careful of disrupting your design’s functional grouping or the expectations of

your user.

So far we’ve only discussed left-to-right alignment. Vertically, you should ensure that labels and controls on

the same row share the same

text baseline

. Java Swing components are designed so that text baselines are

aligned if the components are centered vertically with respect to each other, but not if the components’ tops or

bottoms are aligned. Java AWT components are virtually impossible to align on their baselines. The dialog

shown here has baseline alignment problems, particularly among the controls in the last row: the checkbox

“Use custom editor”, the text field, and the Browse button