This palette is a collaborative effort and contains work done by James Heiser (jheiser@adobe.com) and Don Yacktman (Don_Yacktman@byu.edu). Both of us had played a little bit with the example in Chapter 18 of the NEXTSTEP manual **Development Tools and Techniques**. We each added our own little twists to the object and palette provided there; James Heiser added the tick marks (MiscProgressBar class) and the Pie version like those seen in Workspace and other NeXT apps (MiscProgressPie class), my changes were to support color and a few more target/action methods (in the class MiscProgressView, which is superclass to both of James' objects). We have decided to combine our code and provide it as a part of the MiscKit.

One major change Don made is to split up the rendering process into separate methods. Why? Granted this is slightly less efficient, it adds a lot of flexibility for subclassing since you can choose to keep parts of the rendering while changing or removing other parts of the rendering process. By doing this, it is possible to insert extra rendering steps in the process, as well. If the rendering were not broken up like this, some of the rendering code in MiscProgressView would have to be repeated in the MiscProgressBar class, for example. (Take a look and you'll see what I mean; the ticks can be inserted into the rendering chain in one of two spots, user selectable.) In this object, because it is so simply rendered, you might find this to be a bit of overkill, but it provides a very simple illustration of why an object might be designed in a particular way; here we are aiming to exploit the advantages of OOP, and are willing to sacrifice a little bit of performance. (And you are really only losing a handful of CPU cycles, so it's not anything significant to worry about!)

Note that you might never use the MiscProgressView class directlyÐthere's no reason that you can't, but the MiscProgressBar is a superset of the functionality, so there's no need to use the MiscProgressViewÐbut it does serve a useful purpose: it provides a way for both the MiscProgressBar and the MiscProgressPie to inherit

shared code, without having to duplicate the code in both classes. Consider it a sort of abstract superclass, if you will¼

You can test out the palette by loading the Test.nib file and playing with it in InterfaceBuilder. It contains several examples of each of the objects, with various different settings in place. Test.nib was put together by Don.

This palette project was created by using Scott Anguish's MiscClockView as a model as far as creating a palette, lib, and using a subproj to hold the palettized objects. Although Don created the initial project and one of the objects, he has turned all control over development to James, who will be considered owner and maintainer of this palette and all the objects of which it is composed.