Putting It All Together

We've studied drag-and-drop enabled applications from two points of view, that of a drag source and that of a drop site. But for most applications, you'll want a window to act as both a drag source *and* a drop site. As it turns out, this is just a simple combination of the techniques we've already seen in the previous sections.

If we use our completed drag source example program from the previous section as a starting point, and then compare it to the drop site example from the first section, it is apparent that the only "pieces" missing are:

- A call to `dropEnable`, to make `DragSource`'s canvas drop-enabled;
- A handler for the `SEL_DND_MOTION` message; and,
- A handler for the `SEL_DND_DROP` message.

If you merge those three short bits of code from *dropsite.rb* into *dragsource.rb*, you should end up with a program that can act as both a drag source and a drop site. To test this program, you should be able to start up two separate copies side-by-side and then drag from one to the other. Since, as written, both copies will start up with a red background, you might want to modify one of them to have a different initial backgroud color. Another interesting possibility is to start up a third program (such as an FXRuby example that has displays a color dialog) and drag colors back and forth between all three programs. You could spend days doing this and never leave the house.

The complete program is listed below, and is included in the *examples* directory under the file name *dragdrop.rb*.

```
require 'fox16'

include Fox

class DragDropWindow < FXMainWindow
  def initialize(anApp)
    # Initialize base class
    super(anApp, "Drag and Drop", :opts => DECOR_ALL, :width => 400, :height => 300)

    # Fill main window with canvas
    @canvas = FXCanvas.new(self, :opts => LAYOUT_FILL_X|LAYOUT_FILL_Y)
    @canvas.backColor = "red"

    # Enable canvas for drag-and-drop messages
    @canvas.dropEnable

    # Handle expose events on the canvas
    @canvas.connect(SEL_PAINT) do |sender, sel, event|
      FXDCWindow.new(@canvas, event) do |dc|
        dc.foreground = @canvas.backColor
        dc.fillRectangle(event.rect.x, event.rect.y, event.rect.w, event.rect.h)
      end
    end

    # Handle left button press
    @canvas.connect(SEL_LEFTBUTTONPRESS) do
      #
      # Capture (grab) the mouse when the button goes down, so that all future
      # mouse events will be reported to this widget, even if those events occur
      # outside of this widget.
      #
      @canvas.grab
```

```
      # Advertise which drag types we can offer
      dragTypes = [FXWindow.colorType]
      @canvas.beginDrag(dragTypes)
    end

    # Handle mouse motion events
    @canvas.connect(SEL_MOTION) do |sender, sel, event|
      if @canvas.dragging?
        @canvas.handleDrag(event.root_x, event.root_y)
        unless @canvas.didAccept == DRAG_REJECT
          @canvas.dragCursor = getApp().getDefaultCursor(DEF_SWATCH_CURSOR)
        else
          @canvas.dragCursor = getApp().getDefaultCursor(DEF_DNDSTOP_CURSOR)
        end
      end
    end

    # Handle SEL_DND_MOTION messages from the canvas
    @canvas.connect(SEL_DND_MOTION) do
      if @canvas.offeredDNDType?(FROM_DRAGNDROP, FXWindow.colorType)
        @canvas.acceptDrop
      end
    end

    # Handle left button release
    @canvas.connect(SEL_LEFTBUTTONRELEASE) do
      @canvas.ungrab
      @canvas.endDrag
    end

    # Handle SEL_DND_DROP message from the canvas
    @canvas.connect(SEL_DND_DROP) do
      # Try to obtain the data as color values first
      data = @canvas.getDNDData(FROM_DRAGNDROP, FXWindow.colorType)
      unless data.nil?
        # Update canvas background color
        @canvas.backColor = Fox.fxdecodeColorData(data)
      end
    end

    # Handle request for DND data
    @canvas.connect(SEL_DND_REQUEST) do |sender, sel, event|
      if event.target == FXWindow.colorType
        @canvas.setDNDData(FROM_DRAGNDROP, FXWindow.colorType, Fox.fxencodeColorData(@canvas.backColor))
      end
    end
  end

  def create
    # Create the main window and canvas
    super

    # Register the drag type for colors
    FXWindow.colorType = getApp().registerDragType(FXWindow.colorTypeName)

    # Show the main window
    show(PLACEMENT_SCREEN)
  end
end

if __FILE__ == $0
  FXApp.new("DragDrop", "FXRuby") do |theApp|
    DragDropWindow.new(theApp)
    theApp.create
    theApp.run
```

```
  end
end
```