MorphicDraw

Stephan J.C. Eggermont, Sensus

April 27, 2015

MorphicDraw is a drawing application demonstrating some of the power of Morphic. Morphic is a powerful graphics environment, used in Self, Squeak, Cuis and Pharo. In an iterative and incremental process we'll build up an application that supports drawing connected figures.

1 A Morphic Application

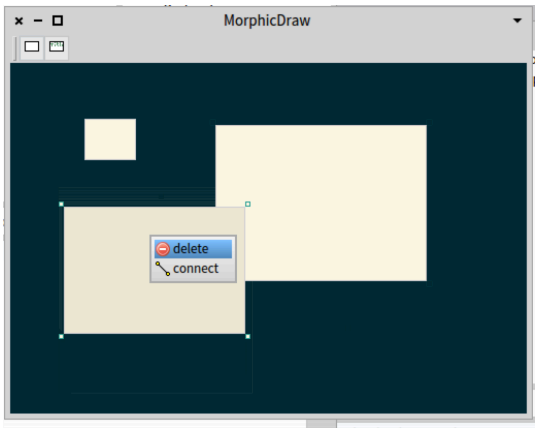

Figure 1: A first iteration of the main window of MorphicDraw

The first iteration (Figure 1) shows an application window with a toolbar and a drawing area. In the drawing area there are three graphical shapes, one of which is selected. A context menu for the selected shape shows options to delete it and to connect it.

1.1 An application with a window

Add a class that represents the application. It has instance variables for the different parts.

```
Object subclass: #MorphicDraw
    instanceVariableNames: 'window tools dock'
```

```
    classVariableNames: ''
    category: 'MorphicDraw-Model'
```

Creating a window in Morphic is simple. Open a workspace and DoIt

```
StandardWindow new openInWorld
```

This creates a window and opens it on the screen. It already has default behaviour for closing and resizing, and a default title. All graphical elements in Morphic are subclasses of Morph, and the World is a container for all of them. Opening a Morph in the world positions it and makes it visible. An alternative to opening it directly is to add it to the (mouse) cursor. In Morphic this is called the hand. DoIt:

```
StandardWindow new openInHand
```

The window is then positioned by clicking.
The MorphicDraw application uses the first, but needs to change the window title and default size.

```
MorphicDraw>>createWindow
    window := StandardWindow new
        setLabel: 'MorphicDraw';
        extent: 400@400;
        yourself.
```

StandardWindow is part of PolyMorph. PolyMorph makes the Window responsible for adding predefined user interface widgets to the application Window. For that it uses the TEasilyThemed trait. It adds a lot (163 in my current image) of convenience methods.
In Morphic, a toolbar in a window has buttons on it. This iteration of MorphicDraw uses two buttons to be able to create two different graphical shapes.

```
MorphicDraw>>createNewCardButton
    ^ window
        newButtonFor: self
        getState: nil
        action: #newCard
        arguments: nil
        getEnabled: nil
        labelForm: MDIcons default cardIcon
        help: 'New Card' translated
```

The help text is shown when hovering the mouse over the button. The button is always enabled, and sends the #newCard message without any arguments to self when it is pressed. It has no state-dependent behaviour or shape. The icon for the button is provided by MDIcons default cardIcon.
The button for the other shape is similar:

```
MorphicDraw>>createNewRectangleButton
    ^ window
        newButtonFor: self
        getState: nil
        action: #newRectangle
        arguments: nil
        getEnabled: nil
        labelForm: MDIcons default rectangleIcon
        help: 'New Rectangle' translated
```

A toolbar in Morphic consists of two parts, a ToolDockingBar and a Toolbar. The Toolbar is added

```
MorphicDraw>>createToolBar
   tools := window newToolbar: (Array with: self createNewRectangleButton with: self createNewCardBut
    dock := window newToolDockingBar.
    dock addMorphBack: tools
```

At the class side add a method to open the application

```
MorphicDraw>>open
    ^self new open
```

2 Shapes and PasteUpMorph

3 Toolbar

4 Connecting

5 Selection and resizing