Wayne Beaton Dwight Deugo

Visual Programming and Reusable Parts: The Marquee Part

Constructing reusable parts is a difficult task. You have to know what tools are available to use, and you must be familiar with existing reusable parts. Most importantly, you have to know the process to construct them. The task is not impossible, and with experience it becomes easier.

In this column, we will help you gain experience creating reusable parts by discussing the creation of a marquee part. A marquee is a relatively simple part designed to reflect the operation of a light board appearing in the front of many sport facilities. The marquee scrolls text that is too large to display statically. Our implementation scrolls right-to-left by repeatedly removing the first character of its message, and tacking it on to the end.

CONSTRUCTING A MARQUEE

It is simple to construct a window containing a Scrolling label. We constructed one called a MarqueeDemonstrationWindow, using both a label part and a timer part. Two connections between the window and the timer ensured that the timer started when the window opened and stopped when the window closed. Another connection updated the contents of the label when the timer ticked. Figure 1 shows the appearance of the MarqueeDemonstrationWindow in the Composition Editor.

IBM's VisualAge for Smalltalk Version 3.0 provides a timer part in the Multimedia Package. Version 2.0 includes the same part with an exercise. The timer has several features that can be set using the Settings Editor, shown in Figure 2. These features include the timer's period in milliseconds, and whether or not the timer should fire repeatedly. In our example, the timer is set to repeatedly fire every 100 milliseconds.

To support the scrolling mechanism in the Marquee DemonstrationWindow, we made the following connections in the Composition Editor. First, we connected the window part's openedWidget event to the timer's start action, and then connected the window's closedWidget event to the timer's stop action. With these connections, the timer fires one hundred milliseconds after the window opens, and continues to fire every one hundred milliseconds until the window closes. Finally, we connected the timer's timerFired event to the following script titled scroll:

```
scroll
    | oldText newText |
    oldText := (self subpartNamed: 'Label1') labelString.
    newText := (oldText copyFrom: 2 to: oldText size),
      (String with: oldText first).
    (self subpartNamed: 'Label1') labelString: newText
```

The script retrieves the label's current string, constructs a new one containing the current string's first character removed and appended to the end of it, and then updates the label with the new string. As this script executes ten times a second, the label's string cycles continuously.

We made further embellishments to permit the user to dynamically change the label's string. We added a push button and an entry field to our window so that the user could click the push button to set the label's string to the entry field's current string. To support this functionality, we connected the push button's clicked event to the label's labelString attribute and connected the clicked/labelString connection's value attribute to the entry field's object attribute.

WIDGET LAYOUT

Part of constructing a complete application includes specifying how parts react when a window is resized. Figure 3 shows the MarqueeDemonstrationWindow in action. When the window is resized, the entry field clings to the top of the window and the scrolling label clings to the bottom. The left and right sides of each part remain constant distances from the sides of the window.

Specifying a part's layout attachments in VisualAge is simple. In the a part's Settings Editor there is a page labeled Layout. On this page, you can specify layout information for the part's top, bottom, left, and right sides. For the input field part, we specified that its top

Dwight Deugo and Wayne Beaton are senior members of the development and educational staff at The Object People, in Ottawa, Ontario. Dwight (dwight@objectpeople.on.ca) has immersed himself in objects for more than 10 years and has helped clients with their object immersions as a project mentor and as a course instructor. Wayne (wayne@objectpeople.on.ca) is the coordinator of course construction and a software developer.

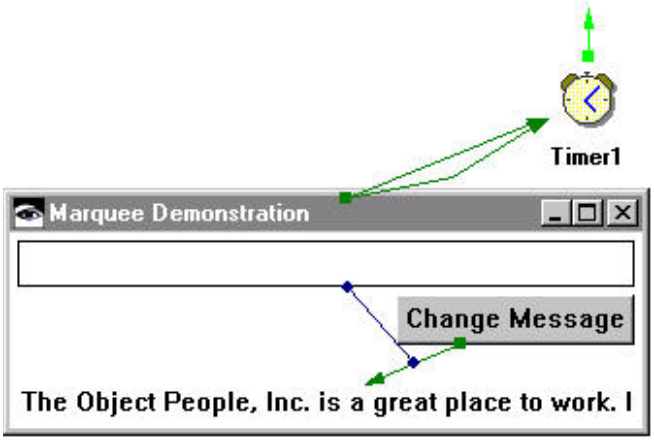

Figure 1. The MarqueeDemonstrationWindow as it appears in the Composition Editor.

edge is attached to the "Parent top edge," and that there is a four-pixel separation between them (the top edge of the input field part is a constant four pixels away from the top edge of its parent). The parent of any part is the canvas (typically a window) where the part is situated. If the part is positioned on a form, then the form is its parent. Similar attachments were specified for the input field's left and right edges (connecting them to the parent left and right edges, respectively).

The input field part's bottom edge used a different attachment. Specifying "No" attachment for any part indicates to VisualAge that the part should use whatever amount of space is appropriate. In the case of an input field part, the appropriate amount of space depends on its current font—the bottom edge will be far enough away from the top edge to accommodate the font, plus an appropriate amount of gutter space. Be aware that the "No attachment" attachment type makes sense only if the opposite edge is attached. For example, it is not possible to have both the top and the bottom unattached.

The "Target attachment" type is also very powerful. It permits you to constrain one part's edge to another part's edge. In Figure 3, the push button's top edge is attached to the input field's bottom edge. By using all of these arrangements, it is possible to design a window in which the parts always resize correctly, because they are dependent on the positioning of other parts, the current font, and the monitor resolution—fonts sometimes have different metrics in different monitor resolutions.

MAKING A REUSABLE PART

TheMarqueeDemonstrationWindow demonstrates one possibility in VisualAge: a scrolling marquee. However, anyone wanting the same marquee feature must drop the appropriated parts and make the correct connections. Since the marquee feature has potential for reuse, we built a reusable Marquee part for others to use, in which its construction was transparent to them.

The construction of the Marquee part began by creating a new visual part named Marquee. In the Composition Editor, we removed the default window and copied to it the label and timer parts from the MarqueeDemonstrationWindow, as shown in Figure 4. The script was also copied to the new part.

Next, we restored the timer's timerFired: event connection to the script named scroll. However, we had to change the way the timer started and stopped—there was no longer a window to trigger openedWidget and closedWidget events. These events still exist because all visual parts give notice, via the triggering of events, that they have opened. The Marquee part is no different. The context menu on the free-form surface area (the white space in the Composition Editor) contains the connect menu for the Marquee part. We connected the openedWidget event from the Marquee part to the timer's start action, and connected the closedWidget event to the timer's stop action.

Our implementation scrolls right-to-left by repeatedly removing the first character of its message, and tacking it on to the end.

This new Marquee part was now ready to use in the MarqueeDemonstrationWindow. First, we removed the old label and the timer parts from the window. Since the new Marquee part was not accessible in the palette, we added it using the Composition Editor's Options menu entry "Add" part. We placed the Marquee part in the window, opened the window, and it worked!

PRIMARY PARTS

Visual parts do not necessarily contain a window. However, all parts created with the Composition Editor have a Primary Part. The primary part is the first thing the

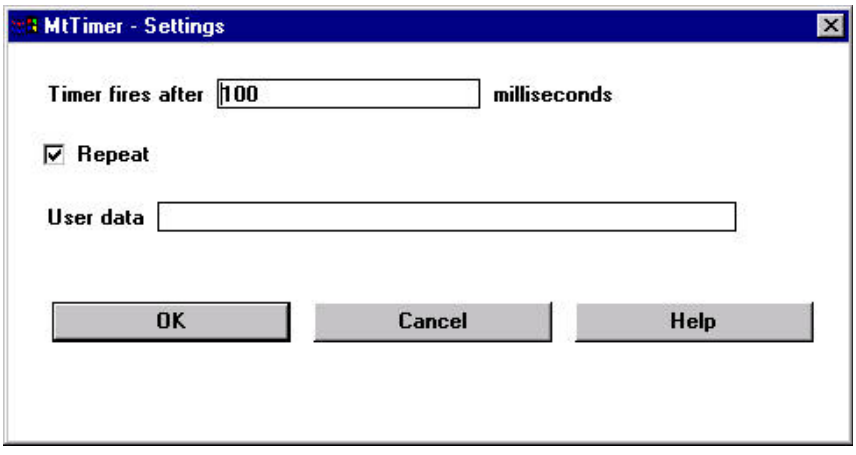

Figure 2. The Settings Editor for a Timer part.

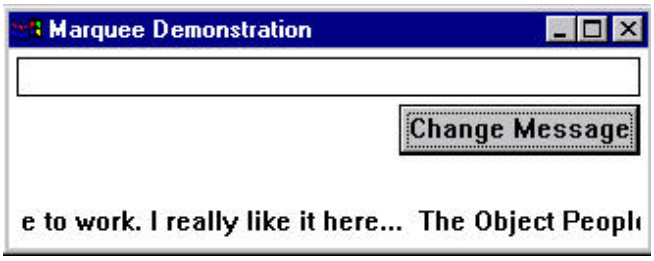

Figure 3. The MarqueeDemonstrationWindow in action.

user sees, which is usually a window. Sometimes the primary part can be another visual part. In our Marquee part, the primary part is the label. There is no special reason to have a window part as default primary part, short of windows being useful.

In the Composition Editor all visual parts that are not the primary part have an entry "Become" primary part in their context menu. We used this menu item to change the label to the new primary part.

In the original MarqueeDemonstrationWindow, the user could change the contents of the marquee. That ability is still available in our new Marquee part. Every part created using the Composition Editor has the same public features as those of its primary part. Therefore, the Marquee part has the same features as the label, including the labelString attribute. In fact, if you open the settings for the Marquee part, you will see entries for most of the label's attributes, including labelString.

The notion of a primary part permits one to have more than one window in a single Composition Editor.

As with the previous version of the MarqueeDemonstrationWindow, we connected the push button's clicked event to the Marquee part's labelString attribute and connected the value attribute of the resulting connection to the object attribute of the text field.

PROMOTE PART FEATURE

The connection between the push button and the Marquee part's labelString attribute identifies several problems with the Marquee part. First, should a Marquee part actually have a labelString attribute? We tend to think of a marquee as having a message that is displayed. Second, the name labelString implies something about the internals of the Marquee part that should be irrelevant, especially if those internals may change at some point in the future.

Two improvements are required. We need a new name for the labelString attribute (perhaps message). Also, the labelString attribute, along with the other inappropriate features, should be removed from the Marquee's public interface. The first problem is easy to correct, the second is not. We will now address the first problem and defer the second problem for a future discussion.

VisualAge can promote a part's features to features of its parent part. In our case, we wanted the label's labelString attribute promoted as a message attribute of the Marquee part. The benefit is that users of the Marquee part no longer know about the Marquee's internal representation, and, if at a later time we decide to change the implementation, we only have to ensure that the message feature persists.

In the current version of VisualAge, features of the **primary part** cannot be promoted. We assume that the designers built this into the product because all those features are made available anyway. Apparently, the designers did not have our reasoning. One workaround was to make another part become the primary part, promote the required feature of the label, and then change the primary part back to the label.

COMPOSITION EDITOR VARIABLES

Another workaround requires the use of variables. First, create a variable and connect its self attribute to the label's labelString attribute with an attribute-to-attribute connection. With this connection in place, the contents of the labelString attribute will always be reflected in the variable, and vice-versa. Rather than "fudging" the feature promotion as before, you can now directly promote the variable's self as the feature message. The self attribute of a variable actually refers to the contents of the variable, in this case a string. With the connection between the variable and the label, promoting the variable's self attribute has the same effect as promoting the label's labelString attribute.

VisualAge makes these tasks easy. The pop-up menu for the label part contains an entry "Tear-Off Attribute." When this entry is selected for a part such as a label, VisualAge prompts you to select an attribute from the label's attribute list. Based on your selection, VisualAge creates an appropriate variable and an attribute-to-

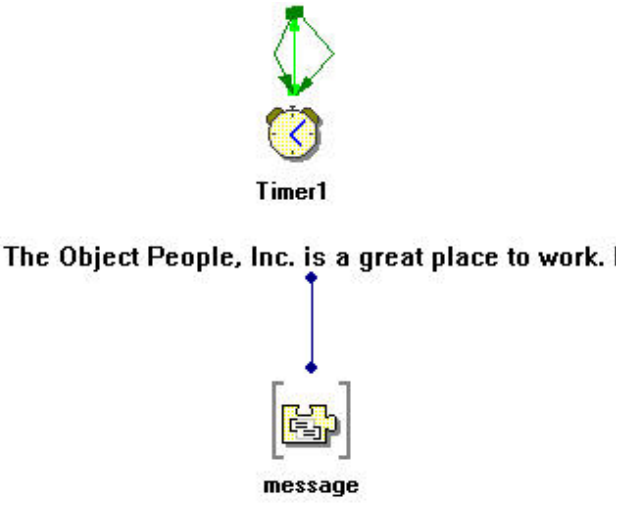

Figure 4. The Marquee part in the Composition Editor.

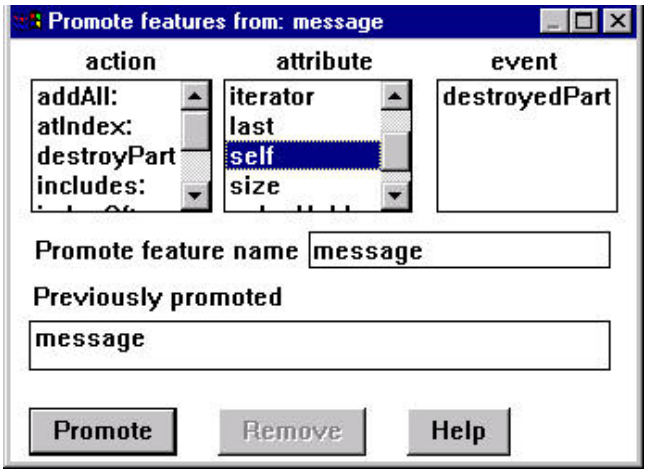

Figure 5. The Promote Part Features window.

attribute connection between the label's selected attribute and the variable's self attribute.

To promote the variable's self attribute as the Marquee's message attribute, position the mouse over the newly created variable in the Marquee part, pop up the menu and then select "Promote Part Features…." In the promote window, shown in Figure 5, select the self attribute and provide message as the feature name. Finally, click the Promote button to add the feature to the public interface of the Marquee part.

After saving the Marquee part, its feature list will include the promoted feature message. Further, the Marquee part's Settings Editor automatically includes the promoted message feature for you to initialize.

SUMMARY

We've discussed the construction of one reusable part: the Marquee part. Although its use is limited, what is interesting about it is the process and the VisualAge's features used for its construction. The notion of a primary part permits one to have more than one window in a single Composition Editor. It also makes it possible to have a primary part that is not a window (perhaps a push button that opens a window might find some application as a reusable part). Promoting part features is a wonderful mechanism for selectively granting access to the internals of objects. The Composition Editor's variables are useful for passing values from one part to another. They also permit access to parts of parts.

In future columns we intend to discuss more about each of these mechanisms, and many others, in the contexts of other interesting reusable parts we have created.

THE CODE

The code presented in this column and in future columns is available at http://www.objectpeople.com. The code is presented as an IBM Smalltalk library file, containing two versions of the software. Version one provides the single-class implementation of the Marquee; version two presents the reusable form with an example. S