FrameScript: Loops and Lists

by Rick Quatro
EMAIL

Last time, we introduced a simple FrameScript loop to process all of the paragraph formats in a document. In this lesson, we will introduce a variation of the loop, and the important concept of "linked lists" of FrameMaker objects.

The general syntax for the `Loop ForEach` command is

```
Loop ForEach(Object) In(Object) LoopVar(Variable)
...(perform tasks here)...
EndLoop
```

There are times, however, when this kind of loop will not work. For instance, you cannot use `Loop ForEach` to loop through all of the graphics in the active document; see page 84 in the *Scriptwriter's Reference* for a list of valid parameters. In addition, some commands will not work inside the loop. Here is an example. Open a blank, portrait document, choose FrameScript > Script Window, and type the following in the Edit Script Window:

```
Loop ForEach(PgfFmt) In(ActiveDoc) LoopVar(vPgfFmt)
  Delete Object(vPgfFmt);
EndLoop
```

Open the document's Paragraph Catalog, and click the Run button. We would expect the script to delete all of the paragraph formats in the document, but it doesn't. Each time you click Run, the script deletes one format and stops. To understand why this happens, let's introduce a different kind of loop and examine how a loops work in general.

First, here is a loop from the last lesson that we know works.

```
Loop ForEach(PgfFmt) In(ActiveDoc) LoopVar(vPgfFmt)
  Display vPgfFmt.Name;
EndLoop
```

Now, we will introduce another loop that performs the same task.

```
Set vPgfFmt = ActiveDoc.FirstPgfFmtInDoc;
Loop While(vPgfFmt)
  Display vPgfFmt.Name;
  Set vPgfFmt = vPgfFmt.NextPgfFmtInDoc;
EndLoop
```

Here is a summary of what the second loop does.

1. Sets the `vPgfFmt` variable equal to the document's first paragraph format.
2. Begins a loop that will continue as long as `vPgfFmt` is "true." In other words, the loop will continue as long as `vPgfFmt` "exists." If there are no paragraph formats in the document, the script will never enter the loop, because `vPgfFmt` would be "false" (does not exist).
3. Once inside the loop, the script will display the current paragraph format's name.
4. The second line inside the loop is the key. The value of `vPgfFmt` is changed to the next paragraph format in the document. If it exists, the script stays in the loop, if not, the script exits the loop.

While the syntax of the loops is different, it is important to understand that the two loops operate the same. The first loop is simply a shorthand version of the second.

The basic purpose of a loop is to perform a task for each member of a list of objects. You go from one member of the list to the next, performing the task on each member, until you reach the bottom of the list. FrameMaker maintains separate "linked lists" of each object type, such as paragraph formats, paragraphs, graphics, and tables. Each object represents a member of its list. *Understanding linked lists is the key to writing scripts!* This may seem like an overstatement, but you need to be able to access and traverse these lists (move from member to member) in order to manipulate them with scripts.

To illustrate linked lists, go to page 145 of the *Scriptwriter's Reference* where Document Properties are listed. This list contains Document properties, but it also contains *links* to other lists of properties. To access the other lists, the Document object gives us a "link" to the first member of each list. In our script, we use dot notation to help us reach the first member in the list of paragraph formats.

```
Set vPgfFmt = ActiveDoc.FirstPgfFmtInDoc;
```

What if we want to reach the second paragraph format in the document? Can we use `Set vPgfFmt = ActiveDoc.SecondPgfFmtInDoc;` or `Set vPgfFmt = ActiveDoc.NextPgfFmtInDoc;`? Neither of these will work, because the Document object only gives us access to the *first* member of the list. The Document object says, if effect, "I'll get you to the first member of the list, but after that, you are on your own."

The responsibility of moving from member-to-member in a list falls on each of the list members. Each `PgfFmt` object has a `NextPgfFmtInDoc` property. Each `Pgf` object has a `PrevPgfInDoc` property and a `NextPgfInDoc` property. The Document object "passes the baton" to the first member of the list, and the first member passes it the next, etc. This what our script is doing inside the loop.

```
Set vPgfFmt = vPgfFmt.NextPgfFmtInDoc;
```

When the script gets to the last member of the list, its `NextPgfFmtInDoc` property returns zero because the `NextPgfFmtInDoc` object does not exist. This causes the `vPgfFmt` variable to be "false" and the loop exits.

Our first loop uses the same method to access the document's paragraph formats, but the "baton passing" is hidden. Let's use the second form of the loop (and our knowledge of linked lists) to solve the problem of the script stopping before it deletes all the formats. Here is the revised form of the script to delete all of the paragraph formats in the document.

```
Set vPgfFmt = ActiveDoc.FirstPgfFmtInDoc;
Loop While(vPgfFmt)
  Delete Object(vPgfFmt);
  Set vPgfFmt = vPgfFmt.NextPgfFmtInDoc;
EndLoop
```

Copy this script into your Script Edit Window and click Run. Like before, it only deletes one format, and this time, it gives an error. Can you figure out why it only deletes one format? If we use the runner with the baton metaphor, the answer is simple. The line

```
Delete Object(vPgfFmt);
```

deletes the `vPgfFmt` object before it has a chance to "pass the baton" to the next paragraph format in the document. To put it bluntly, we killed the runner before he could pass the baton! The next line in the script gives an error because the `vPgfFmt` object no longer exists.

Here is the solution:

```
Set vPgfFmt = ActiveDoc.FirstPgfFmtInDoc;
Loop While(vPgfFmt)
  Set vPgfFmtToDelete = vPgfFmt;
  Set vPgfFmt = vPgfFmt.NextPgfFmtInDoc;
  Delete Object(vPgfFmtToDelete);
EndLoop
```

What we are doing is setting another variable to the current paragraph format with the line

```
Set vPgfFmtToDelete = vPgfFmt;
```

The next line passes the baton but now we have a variable, `vPgfFmtToDelete`, that remembers the previous runner that had the baton. It is now safe to delete the previous paragraph format, because `vPgfFmt` already represents the next one in the linked list.

If you don't quite understand this lesson yet, don't be too concerned. Keep working with the mechanics of the scripts, and the concepts will become clearer. As a reward for your endurance, I will finish (and, hopefully, reinforce) the lesson with a very useful script. This script will delete all of the paragraph formats from the catalog that are not in use in the document. This is a good way to clean up a document before importing formats from a template. The script will also introduce another loop and some new concepts, such as error checking and conditional statements.

Before beginning the script, it is best to figure out how we are going to attack the problem. At this point, we want to determine the "logic" of the script without worrying about the syntax. I like to verbalize the overall solution in plain English, and then make a list of the individual tasks. Here is the list form of my solution.

1. Make a list of all of the paragraph formats in the catalog.
2. For each paragraph in document, determine what paragraph format it

uses.
3. If the paragraph's format is in the list, remove it from the list.
4. After testing all of the paragraphs, see if there are any paragraph formats left in the list.
5. If there are, delete them from the document's catalog.

Below is the complete listing of the script. As an exercise, try to follow the script and figure out which lines of code correspond with the list items above. Try to add comments to the script. Can you think of any useful features to add? In the next lesson, we will analyze the script line-by-line.

```
// Delete all unused paragraph formats in the document.
If ActiveDoc = 0
  MsgBox `There is no active document.';
  LeaveSub;
Else
  Set vCurrentDoc = ActiveDoc;
EndIf

New StringList NewVar(vPgfFormatsInCatalog);

Set vPgfFmt = vCurrentDoc.FirstPgfFmtInDoc;
Loop While(vPgfFmt)
  Add Member(vPgfFmt.Name) To(vPgfFormatsInCatalog);
  Set vPgfFmt = vPgfFmt.NextPgfFmtInDoc;
EndLoop

Loop ForEach(Pgf) In(vCurrentDoc) LoopVar(vPgf);
  Find Member(vPgf.Name) InList(vPgfFormatsInCatalog)
    ReturnStatus(vFound);
  If vFound = 1
    Remove Member(vPgf.Name) From(vPgfFormatsInCatalog);
  EndIf
EndLoop

If vPgfFormatsInCatalog.Count > 0
  Loop While(vCounter <= vPgfFormatsInCatalog.Count)
   LoopVar(vCounter) Init(1) Incr(1)
   Get Member Number(vCounter) From(vPgfFormatsInCatalog)
    NewVar(vPgfFormat);
   Get Object Type(PgfFmt) Name(vPgfFormat) NewVar(vPgfFmt);
   Delete Object(vPgfFmt);
  EndLoop
EndIf
```

Features | Tips | Reviews | Case Studies | Adobe Speaks