Gerald's Column
by Gerald Fitton

Over the last few months I have been gently asking what sort of things you might want me to write about. One group of topics which has featured in many emails to me is a request for examples of some of the more difficult or obscure features of PipeDream and Fireworkz. Some have asked me for articles describing how to use Impression and Ovation more efficiently; I shall leave tutorials for those packages to others.

Last month I mentioned that a PipeDream (or Fireworkz) custom function could be used to execute a case sensitive sort. Consequently I have been asked to explain what custom functions are and how they work.

This month you will be treated to a custom function tutorial. Let's see how it goes down.

Some History

Custom Functions were a new feature introduced in PipeDream 4. They were intended to replace and improve on a specialised feature of PipeDream 3 which allowed users to integrate PipeDream with a BBC BASIC program. Access to BASIC still exists in PipeDream 4 but it was not documented in the hope that its use would 'fade away'.

All PipeDream format files will load into Fireworkz and PipeDream Custom Functions are no exception. If you have a PipeDream Custom Function then it will load directly into Fireworkz and work in exactly the same way that it does in PipeDream.

The Library

If PipeDream and Fireworkz are unable to find a referenced file then they look first at files which are in the same directory as the file which is calling the referenced file. If it is not found then the next place searched is a subdirectory located at !PipeDream.User.Library or in !Fireworkz.User.Library.

I have a few custom functions in my Library. These include custom functions for finding all sorts of unusual statistical parameters and one for the inversion of large matrices.

Command Files and Custom Functions

One thing to clear up from the start. Custom functions are different from command files.

Custom functions did not exist in PipeDream 3 but command files (called 'macros') did.

Command files consist of a sequence of PipeDream commands such as <Ctrl CGS> (Cursor Goto Slot) and <Ctrl BM> (Block Move); they can be used to add or delete rows and columns, even to load and save documents.

Custom functions can not be used to move blocks around within a document nor indeed to change the 'shape' of a document in any way.

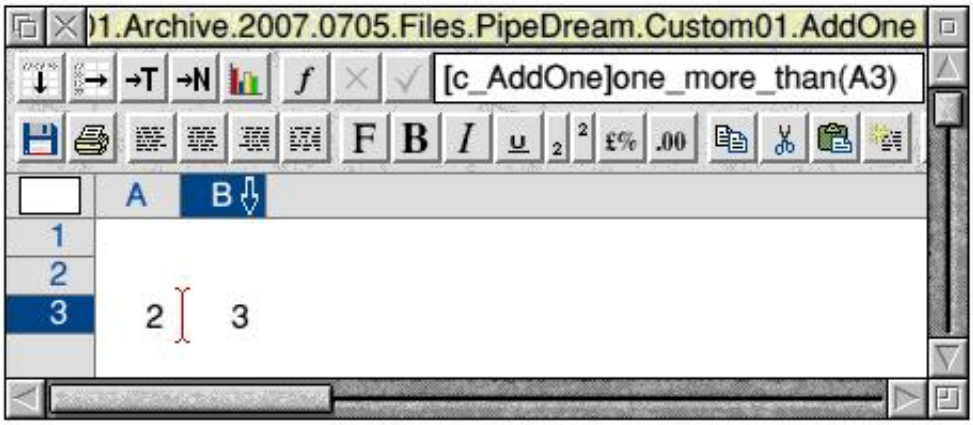
What you can do with custom functions is to process the data which already exists within the slots of a document (or set of documents) in more intricate ways than can be done simply with the 'standard' functions provided. Just like any standard function such as (A1 + A2) or lookup(A1,B1B10,C1C10), a custom function is 'called' from a slot and the 'result' of executing the function is returned to the slot from which it is called.

Using custom functions

Let's have a look at a custom function at work.

If you have the files on the monthly disc or elsewhere then double click on the file [AddOne] and you will find that you load to the screen not only [AddOne] but a dependent document which I have called [c_AddOne] containing a simple, two line, custom function.

The file [AddOne] looks like the screenshot below.



Because PipeDream 4 saves the position of a PipeDream document on the screen and the position of the cursor within the document, what you should find is that, just like the screenshot, [AddOne] is visible and alive (a yellow top bar) with the cursor in slot [AddOne]B3. In the formula line you will see the content of the cell B3, a formula. It is this formula which calls the custom function; we shall return to this later.

If you have the files then position the pointer in cell A3 (it should be the number 2 unless you've modified the file) and click the mouse select (left) button. You will see the number "2" in the formula line. Enter some other number, say 9, and <Return>. If all goes well then, besides [AddOne]A3 changing to 9, the number in [AddOne]B3 will change to 10.

Custom Function File Names

You can use any valid filing system file name for a custom function document but, so that I can recognise which PipeDream files are 'ordinary' documents and which are custom function documents, I have prefixed all my custom function documents with c_ (a lower case c followed by an underline). I suggest that you follow this convention; it was originally recommended by Colton Software.

Calling the Custom Function

The custom function 'called' from [AddOne]B3 is not a spectacular custom function. The 'result' returned to the slot [AddOne]B3 is (A3 + 1), one more than the number in the slot [AddOne]A3. Change the value in [AddOne]A3 a few times and convince yourself that the value in [AddOne]B3 is always 1 more than the value in [AddOne]A3.

Now let's have a look at the content of slot [AddOne]B3. Place the pointer over [AddOne]B3 and click select. The formula line does not show the simple, non custom function way of adding 1 which would be (A3 + 1); instead it shows the formula used to 'call' the custom function, namely [c_AddOne]one_more_than(A3).

Before looking at the custom function document, [c_AddOne], let's have a look at the formula used to call the custom function in more detail.

The 'calling' formula is in three parts:

The first part, [c_AddOne], is the name of the dependent document which contains the custom function. A custom function document such as [c_AddOne] can contain more than one custom function. Indeed, it is desirable to 'split up' large, exotic custom functions into a set of smaller custom functions (stored within the same 'c_' document) in the same way that it is usually desirable to split up a long program written in BASIC into more easily digested PROCs and FNs.

The second part, 'one_more_than', is the name of the custom function within [c_AddOne]. If you look at slot A7 of the document [c_AddOne] you will see that 'one_more_than', the name of the custom function, appears after the word 'function'. Another convention which I recommend to you is that the names of custom functions should be totally in lower case (no capitals). The reason is that lower case and upper case function names are equivalent (unlike BASIC procedures) and PipeDream converts upper case letters in the custom function names to lower case anyway!

The third part, '(A3)', is the single piece of data which is passed to the custom function for processing. To be more accurate, [AddOne]A3 is the slotref of the data passed to [c_AddOne] for processing. Data passed to a custom function this way is called a 'parameter'. A parameter is a variable which has a 'fixed' value within the custom function but can be varied outside the custom function. As we shall see in another tutorial, it is possible to pass more than one parameter to a custom function. You can pass as many different parameters as you wish to a custom function but, if you want to pass a large amount of data (for example a column of words which you wish to sort in a case sensitive manner) then, rather than use many parameters (a rather cumbersome method), it is better to pass a range of slots as an array so that it is passed as one single parameter, the name of the array.

Returning the result

'Calling' the custom function '[c_AddOne]one_more_than(A3)' from slot [AddOne]B3 returns the result of the processing to the slot from which it was called, [AddOne]B3.

Although the result can be returned to only a single slot this does not mean that you can return only one value from a custom function! You can return many values to one slot by returning an array to the calling slot. This array can then be expanded using (for example) set_value(range,slotref) where 'slotref' contains the array (and the calling function) and where 'range' is the range of slots containing the expanded array. We may return to examples of this tactic in another tutorial.

Programming in general

Before we look at the custom function [c_AddOne] in detail I am going to digress and discuss programming from a broader point of view.

Perhaps you are one of the few remaining RISC OS users who is familiar with writing short programs in BASIC. That may help or it may not! Perhaps you write 'well written' or perhaps 'badly written' BASIC programs. Both types usually work most of the time; both types sometimes fail. The difference between the two is often apparent only when you want to 'debug', modify, improve or extend the program you (or maybe someone else) has written.

Those programs which I class as 'good' are usually easier to 'debug' if they don't work first time and easier for others to modify if they wish to extend or improve them.

Short 'bad' programs can usually be written in a much shorter time so too many programmers with no formal training pick up some bad habits when they start programming. This is particularly true if they write short 'bad' programs which work. Early 'bad' habits learnt that way are often the most difficult to break.

Some languages encourage 'good' programming. Others permit 'bad' or even 'very bad' programming. I think I would put the PipeDream Custom Function Programming language in the 'fairly good' category!

Programming

Custom function documents such as '[c_AddOne]' contain custom functions such as 'one_more_than' written in the PipeDream Custom Function Programming language.

A custom function is a set of instructions which is designed to process similar data in a similar way. No matter what the actual value of the data, the same set of instructions are followed. For example, in the case of 'one_more_than' the data passed to it is a parameter representing a single number such as 2 and the pair of instructions which make up 'one_more_than' process the number represented by the parameter by adding 1 and returning the result.

Whatever the number which is passed as a parameter to 'one_more_than' the 'result' returned is one more than the original number. That original number is represented within the custom function as the parameter "parameter". Custom functions are written so that they can and most often do operate on variables (symbols representing any similar piece of data) rather than sets of instructions which process data directly.

There are many ways of classifying variables; one is by the way in which they retain or lose their identity when the overall program is broken down into smaller units. Using this classification a parameter is a 'sort of' variable; there are two more sorts called 'local' and 'global' variables which I shall introduce in another tutorial.

Within all but the simplest programming languages the manner of processing the data (best included as variables) is based on four concepts. We shall study these concepts under the headings: "Sequence", "Repetition" (such as "for - next" loops), "Decision" (such as "if - then" statements) and "Interrupt" (which will include the use of the commands 'input' and 'alert'). I shall limit this tutorial to a discussion of "Sequence" and the use of a parameter.

In other tutorials (if we get that far!) then we shall discover other sorts of variables as well as the concepts of "Repetition", "Decision" and "Interrupt".

Sequence

This topic is concerned with the order in which the commands are executed.

Every sequence must have a start and a finish.

In earlier spreadsheets such as PipeDream 2 and PipeDream 3 you had to choose whether to recalculate along the rows (one at a time, starting at the top and running from left to right along each row) or down the columns (one at a time starting with the left most column and working down each column from top to bottom). One consequence of this was that sometimes it was necessary to force the recalculation manually several times to ensure that all dependent cells had been recalculated.

In PipeDream 4 recalculation takes place in an order which is called "natural".

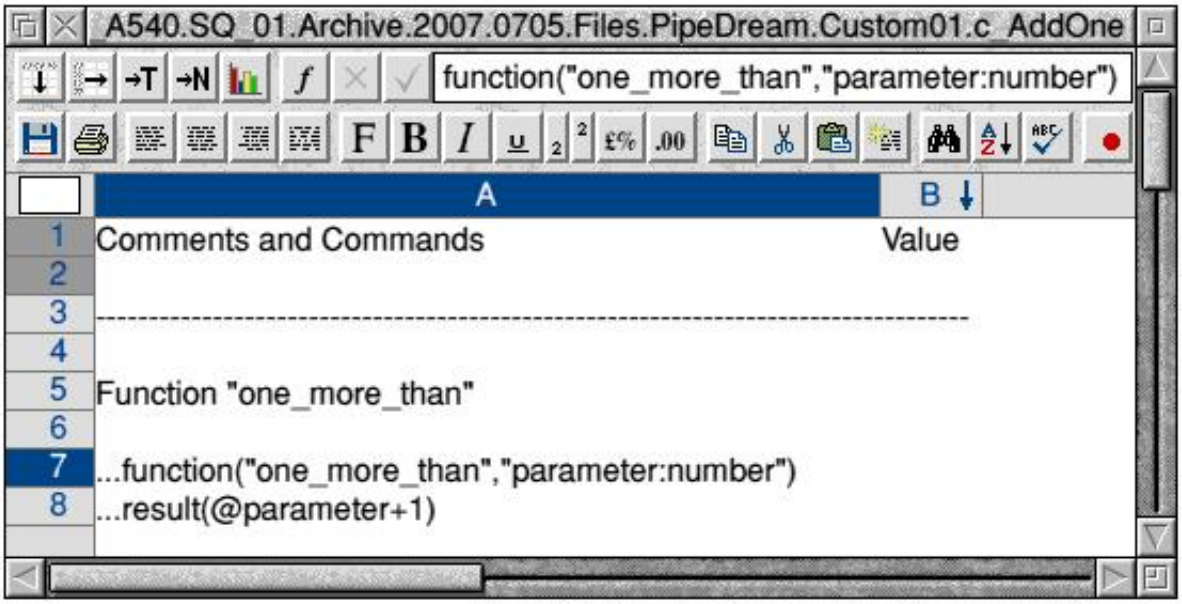
Here is not the place to explain in detail what is meant by "natural"; essentially, with "natural" recalculation, every cell can be regarded as part of a chain and the calculation proceeds along the chain in such a way that cells which depend on other cells are processed no matter where they are located within the document. When a slot is changed in PipeDream 4 only those chains which are affected are recalculated. The remainder of the spreadsheet is not recalculated.

In a PipeDream 4 custom function the default sequence is that commands are executed, one at a time, down a single column. The start is identified by the command 'function' and the end is identified by the command 'result'.

Details of the custom function

What we haven't looked at yet is the custom function itself to see how it adds one to the parameter and returns the 'result'. To do this we must bring the custom function '[c_AddOne]one_more_than' into view.

If you have the files from the disc or elsewhere then you can do this. If you don't then look at the screenshot below.



Rows 1 to 6 inclusive do not do anything; they are there like REM comments in a BASIC program or <!-- --> comments in an HTML document for explanation or comment.

Let's look at the commands in detail and see how they process data.

Row 7 '...function("one_more_than","parameter:number")'

In this custom function the command '...function ...' has two arguments.

The first argument is the name of the custom function, "one_more_than"; note that the name is included in inverted commas.

The second argument is the one and only parameter passed to the function. The name of the one parameter passed to '[c_AddOne]' is 'parameter' and its 'type' has been declared (after the colon) as a number. Because of this declaration, if you try to pass anything other than a number to this function then an error will be generated. Do not declare the type of variable if you want to pass a number sometimes and, say, a string on another occasion.

Try typing Fred (without and then with inverted commas) into [AddOne]A3 and you will find that the error message "String not expected" is returned to slot [AddOne]B3.

You will find the error message foreshortened (truncated) in the body of the [AddOne] document. If you want to read the full error message then click the pointer in [AddOne]B3, then on the formula button (the italic f just to the right of the PipeDream 4 logo) and finally run the pointer through the first menu option Slot 'B3' - Slot value and you will see the full error message displayed. This technique is particularly useful when long error messages are generated.

Row 8 '...result(@parameter+1)'

The command '...result' terminates the sequence of commands and returns a value to the slot from which it was called, [AddOne]B3.

Note the @ sign preceding the word 'parameter' and note that the inverted commas have gone. The @ sign 'goes with' the word 'parameter'. If you have the files then try '...return(1+@parameter)'; it gives the same result. Also, try changing '...return(@parameter+1)' to '...return(@parameter+2)'.

You have to do this in the formula line of [c_AddOne].

Even when you have modified the custom function, the number 3 in the slot [AddOne]B3 does not change. To make that change, and indeed if you want to check whether you have introduced any 'bugs' (errors in programming) in [c_AddOne] you must 'run' the custom function by 'calling' it again from the document [AddOne].

The simplest way of doing this is to place the cursor in slot [AddOne]B3, then move the pointer to the formula line and click on [c_AddOne]one_more_than(A3). Finally, click on the green tick to the left of the formula line; when you click on the green tick the custom function will be called and the new 'result' will be returned to [AddOne]B3.

Other points to note

It is advisable to set the <Ctrl O>, New slot format, to Numbers. Secondly, note that the three dots which precede the word "...function" appear automatically in slot [c_AddOne]A7 and do not appear in the formula line. You do not type the three dots anywhere; just type the expression 'function("one_more_than","parameter:number")' into a 'number' slot and tap <Return>! In case you really get stuck I have included on the monthly disc a file called [CustomFn] which you might like to load and then save with <Ctrl FI> as a custom function template.

Summary

A custom function is a sequence of commands which start with a 'function' command and end with a 'result' command. Once there a custom function is written into any cell in a document then the whole document is a custom function document.

Custom function documents behave differently from 'ordinary' documents. In particular the sequence in which the cells are evaluated is different so keep your custom functions separate from your data files. Also note that custom functions do not have a different file type so, in order to distinguish them from data files use the convention of including the "c_" at the front of the file name!

Communications

Please write to me at archive@abacusline.co.uk with your comments.