Stephen Pelc
MicroProcessor Engineering Ltd
133 Hill Lane
Southampton SO155AF
UK

http://www.mpeforth.com

stephen@mpeforth.com

*The VFX Forthv5.1 kernel incorporates dual-behaviour words and recognisers. This talk discusses our experience over the last year with these changes. Dual-behaviour words are a standards-compliant solution to needing words that have separate interpretation and compilation behaviour. Previous papers called these words NDCS words (non-default compilation semantics). Recognisers are a fashionable solution to providing a user-extensible text interpreter. Our experience converting two OOP packages to use recognisers is discussed, together with problems involved in supporting two floating point formats.*

## Introduction

VFX Forth v5.1 contains a different approach to compilation semantics in Forth and it incorporates recognisers.

Other Forths have different approaches to handling the general problem first defined in ANS Forth, that of "non-default compilation semantics" (NDCS) but in general the solutions involve defining separate actions for the interpret and compile time actions. For example, the definition of **DO** is:

```
: DO            \ Run: n1|u1 n2|u2 -- ; R: -- loop-sys ; 6.1.1240
 NoInterp  ;
ndcs: ( -- )  s_do,  3  ;
```

Our approach to this problem has been described in EuroForth 2018 and 2019 papers.

Recognisers are this decades's solution to the problem of providing a text interpreter that is extensible by the application.VFX Forth uses them for installing two different floating point handlers (NDP and SSE) and for handling two different OOP packages (ClassVFX and CIAO).

## Dual behaviour words

The definition below shown how the separate interpretation and compilation actions do not interfere with each other.

```
: ."            \ "ccc<quote>"  --
\ *G Output the text up to the closing double-quotes character.
  [char] " word $.  ;
ndcs: ( -- )  compile (.")  ",  ;
```

Since we introduced this notation, we have had no problems with it and the code passes all the tests we have tried.

## Recognisers

The text interpreter in VFX Forth is basically as below. It is assumed that **`recognize`** returns the address of a type structure holding the interpret, compile and postpone actions of the returned type.

```
: interpret \ --
  begin  ?stack parse-name dup
  while  forth-recognizer recognize  state @ abs cells + @ execute
  repeat
  2drop
;

: postpone  \ "<name>" -- ; POSTPONE <name> ; 6.1.2033
  parse-name forth-recognizer recognize  2 cells + @ execute
; immediate
```

For us, the big benefit of recognisers is to be able have fine grain control of the parsing. For example, NDP and SSE floats require different handlers; SSE is limited to 64 bit floats whereas NDP can handle 80 bit floats. In an environment which uses Forth source libraries, we may need to be able to switch between OOP packages. When switching to a new OOP package, we have to be able to remove float package notations completely. To do this we need vocabulary search order control as well as recogniser order control.

Recogniser order control is a relatively new idea, but we have been using wordlist search order control for decades. ANS Forth introduced a clumsy pair or words.

```
: GET-ORDER     \ -- widn...wid1 n  ; 16.6.1.1647
\ *G Return the list of WIDs which make up the current search-order.
\ ** The last value returned on top-of-stack is the number of WIDs
\ ** returned.

: SET-ORDER     \ widn...wid1 n -- ; unless n = -1 ; 16.6.1.2197
\ *G Set the new search-order. The top-of-stack is the number of WIDs
\ ** to place in the search-order. If N is -1 then the minimum search
\ ** order is inserted.
```

A more useful pair (invented at Forth Inc.) for daily use is:

```
: -ORDER   \ wid --
\ *G Remove all instances of the given wordlist from the *\fo{CONTEXT}
search order.

: +ORDER   \ wid --
\ *G The given wordlist becomes the top of the search order. Duplicate
\ ** entries are removed.
```

A similar pair can easily be designed for recognisers. I'm not going to get into the naming arguments that recognisers have suffered from for several years. Later recogniser proposals seem to have adopted the idea of a recogniser "stack" which is really just an ordered array. It is possible to use the same structure mechanism for both recognisers and wordlist search order control.

At present, most systems that use recognisers do it in very similar ways. However, the recogniser management words as present in most proposals are inadequate or ugly. For full use of recognisers, wordlist control is equally necessary. For example, if I have to switch between NDP and SSE floats as I may have to do when interfacing to macOS or Linux, I may need to select

```
  NDPfloats ( — )
  SSEfloats ( — )
  integers ( — )
```

These three words need to manipulate both recognisers and wordlists. Once we can do this in a sane way, we can then use the same techniques to switch between OOP packages. In turn this enables us to compile libraries that use different OOP packages without conflict. MPE supported five OOP packages in VFX Forth 32. It's a dreadful position to be in. Since none of the OOP designers in the Forth world seem prepared to compromise on a standard notation, the only solution I can see is for their parsers and compilers to be removable and installable at will. For MPE, this aspect of recognisers is the convincing use case; recognisers for literals did nothing for us as we already supported the common notations.

From the user's point of view, nobody really notices recognisers and they have caused no technical support except for the one or two users who wanted to expand notations. Once explained, they went away happy.

From a standards point of view, the original simple set of words has proven to be enough. We (MPE) are firmly convinced that trying to automate the postpone behaviour is potentially dangerous as we have no idea what clever Forth programmers will get up to with recognisers. The difficult part of recognisers is the set of recogniser management words.

## Converting two OOP packages to recognisers

There are two OOP packages in VFX that needed to be rock-solid with recognisers. CIAO (C Inspired Active Objects) was written over 20 years ago by a long-departed programmer to ease interfacing to C++. The package has its fans.

ClassVFX was designed by and is used by Construction Computer Software at the heart of a large application of 1.4 million lines of Forth source code.

As with many other complex and capable OOP systems in Forth, both packages take over 1000 lines of source code and contain ugly code. As a result, the original code was not thrown away for a new version, but the code was hacked to fit the parser and action model of recognisers. The parser actually performs all the required interpretation and compilation actions - yes, it remains state-smart as it always has been, and the action structure contains `NOOP`s for the interpretation and compilation actions, and a `THROW` for the `POSTPONE` action.

This rather brutal approach to code conversion was somewhat regrettable, but did prove the robustness and flexibility of recognisers. People with time on their hands are welcome to design a new implementation that manages to fully separate the parsing and action parts, but I haven't found a way yet.

CIAO previously used a hook in the text interpreter loop and took full advantage. To provide the same behaviour with recognisers, two parsers and action tables are required. One parser runs first in the recogniser sequence, and the other runs last.

Full source code is provided in the downloadable versions of VFX Forth.

## Acknowledgements

Gerald Wodni persuaded me that recognisers have merit.
Anton Ertl convinced me that all standards have bugs.
Willem Botha taught me about big Forth applications.
Pat Gillespie has tolerated my apparent refusal to retire.