Component-based Forth

By Joe O’Connor

Forth Day 11/15/2008

A bit of history

- I have been using Delphi as a RAD (Rapid Application Development) tool since about 1996.

- My familiarity with Forth precedes that by about 5 years (maybe more).

Forth History – early years

- First exposure – Tom Zimmer's FPC.
- Starting Forth and Thinking Forth.
- Became a member of the Forth Interest Group.
- To sum up, I was fascinated by the concept and simplicity of Forth but didn't do much with it at this early stage. (I am not a hardware/firmware programmer).

Forth – middle years

- Norman Smith's UNTIL – Write your Own Programming Language In C++.
- Smith extolled the virtues of using Forth as a simple macro language on top of a C/C++ application.
- Biggest advantage of using a form of Forth – no need for the complex baggage of lexical analysis and parsing.

Middle years (cont).

- Q: What does the lack of complex lexing/parsing allow?

- A: It allows the construction of a powerful, macro language in only a few thousand lines of code.

More middle years

- Q: Has anyone else had the bright idea of developing a macro language along similar lines besides Forthers?
- A: Tcl/Tk has a similar strategy of making elements of the language as commands with associated parameters. Its model more closely resembles the function application done by Lisp.

Modern Era Part I

- In 1998-1999 time frame, I had used Delphi for several RAD projects, and decided to try my hand at component programming.
- Delphi RAD development is **visual** in nature, and focuses on **using**, not creating components.
- Component development is a separate discipline. It's **nonvisual** in nature, and is aimed primarily in providing extra functionality to the programmer, not the end user.

Modern Era Part I cont.

- Inspiration for developing Creole Forth – UNTIL.
- Just as UNTIL can be piggybacked on top of a C/C++ program, any Delphi application can now have its own application language.
- All it requires is that the TCreole component be dropped onto a form and a few properties set.

First attempt – the good, bad, and ugly

- Used a form of threading probably unique in the history of Forth – “database threading”.
- The dictionary took the form of a Paradox database table.
- High level definitions were resolved down to primitives by repeated SQL queries.
- It is the opinion of this author that database threading should **remain** unique in the history of Forth – it’s horrendously inefficient.

Modern era – Part II

- Q1: What's the problem here?
- A1: A database is simply too heavyweight a solution to implementing a Forth type of lookup table.
- Q2: So what's the solution?
- A2: Find a lightweight data structure to serve as the dictionary.

Second pass

- Delphi's **TStringList** data type filled the bill nicely.
- An outer interpreter could use it as an associative array (aka hash).
- An inner interpreter could use integer indexes for lookup.
- Substituting the database with a TStringList sped up search and looping by a factor of **250,000** according to benchmarks.

Current form of Creole Forth

- Inherits from TComponent (previous incarnation inherited from TSQL).
- Has 5 stacks :
  - Parameter stack.
  - Return stack. This is implemented as an integer array for speed.
  - Vocabulary stack.
  - Prefilter stack.
  - Postfilter stack. Screens values to see if they belong to the allowed datatype. For example if ‘INTEGER’ is on this stack, only integer values are allowed on.

Current form of Creole Forth

- Encryption is used as a namespacing mechanism. Names of words are encrypted differently depending on the vocabulary they're in.
- No STATE variable. On compilation, the IMMEDIATE vocabulary is pushed on the stack. All words in this vocabulary are executed, others are compiled. Compilation ends when IMMEDIATE is popped off the stack.

Features of Creole Forth cont.

- Small and simple – less than 100 definitions.
- Now has networking primitives – found it hard to get along without them.

Examples – let’s create an application

- Open Delphi.
- Drop it on the form.
- Set the properties.

Examples – demo app.

- Web server stuff, etc.

Current applications

- Two-way web server.
  - Front end is a user-friendly interface that allows users to browse the directories of a remote Unix server and execute commands on the files there (ie. Check for duplicate lines or view a summary report).

More applications

- As an ActiveX control in Excel, pulls data from three different sources (Oracle calendar, a remote script that pulls email from a POP3 server, and the files on the server) and displays them in summary format.
- As a high-volume file transformer that takes Excel spreadsheets, transforms them into csv, and transfers them to a Unix server for loading.

Future directions?

- Lazarus and Free Pascal. Write once, compile anywhere. Pursuing this would put Creole Forth on the Linux map.
- Adapting the ‘Forth component’ idea to Java and Javabeans.