MI4 Metaprogramming in Forth

(Some notes and links are given here. Transcript of the presentation video begins on page 3.)

1. Besides MI4 being such a catchy name, we wish to use MI4 Metaprogramming in Forth *as a platform for Forthers of all ages to collaborate with the aim to create education and training materials to promote Forth and related topics to younger programmers.*

   Origin of "MI4": It may not be politically correct to admire James Bond 007 in a certain socialist oriented society in Asia. However, the effectiveness of the capitalist run movie industry in creating heroes and idols is something that should be admired. By the Chinese 矛盾 Maodun (spear and shield) philosophy, which states 以子之毛，攻子之盾 to use thy spear to attack thy shield, borrowing the deprecated agency name MI4 as the abbreviation Metaprogramming in Forth, is to exploit the popularity of James Bond 007's character to promote our cause.

   - https://en.wikipedia.org/wiki/MI4

2. My LinkedIn Profile:

   - bit.do/lsng
   - https://www.linkedin.com/in/%E4%BC%8D%E6%A8%91%E7%9B%9B-%E5%8D%9A%E5%A3%AB-liang-ng-ph-d-0502822b/

3. We used *ffmpeg*, a classic open source tool on Linux platform, extensively to prepare the video for this presentation. *ffmpeg* is (in)famous for its complicated parameters. As such, we have developed **Phoshell**, a very thin Bash shell wrapper based on Phoscript that can be used to ***simplify the execution of commands*** such as *ffmpeg* that have very complicated parameters, using our favourite Forth like Reverse Polish Notation.

   Unfortunately, it would take up extra presentation time to insert this new material in the current video. So we shall demonstrate it in a future presentation, while leaving readers and viewers with the link to Phoshell code and sample ffmpeg commands which may help you create presentation videos in the future.

   - https://github.com/udexon/Phoshell/blob/master/Phoshell/Phoshell.md

4. As a footnote to SymEngine (10:45 in in the presentation video), here are several useful links to computer algebra projects, an important aspect of Metaprogramming in Forth:

   - HOL Light https://www.cl.cam.ac.uk/~jrh13/hol-light/
   - SymPy https://www.sympy.org/en/index.html

- SageMath https://www.sagemath.org/

As you can see, computer algebra systems can benefit tremendously from the simplicity of the syntax of Reverse Polish Notation / Forth / Phoscript.

This has great consequences on the education of mathematics and programming for the next generation of students. Consider this:

- Can we teach computer algebra to teenagers just like how we teach them BASIC, LOGO and Spreadsheets, using RPN / Forth / Phoscript?

This, in our humble opinion, would be the trillion dollar question in the post 5G era.

5. Conclusions: (refer to screenshot at 09:50 of presentation video)

Phoscript introduces a novel concept which we call the **"Sandwich API Model"**. By "sandwich", we are referring to:

i. the top layer of bread as Phoscript;
ii. the bottom layer of bread as Forth or the stack machine interpreter of a programming language;
iii. the middle layer stuffing as the host programming language of Phoscript.

The Sandwich API Model can bring at least two major benefits to software development:

A. Ford style assembly line in software development:

Phoscript enables junior programmers in a team to focus on using Phoscript for coding, while senior programmers may develop Phoscript libraries to interface to the host programming language.

Further, when there are multiple programming languages or frameworks involved, Phoscript can be used as a universal interface script to interface them.

B. Phoscript can be used to interface to legacy systems written in obsolete programming languages, thus making it easier for young programmers to learn to maintain such systems.

Transcript for Presentation Video

Main reference (both links point to the same page):

- bit.do/Phoscript
- https://github.com/udexon/Multiweb/blob/master/Phoscript_Tutorials.md

(Additional links are given alongside the transcript.)

Video link: (Filename MI4_1103a.mp4)

- https://drive.google.com/file/d/1kgsHsiflm5qRAfDEdVeeP445d5b8HxV6/view?usp=sharing

03:35 (Minutes:Seconds)

Just as I was preparing this presentation, the organizer of this Zoom session Peter Forth posted a post comparing C with Forth, which I think is a nice introduction to what I am going to present today.

As you may agree, the majority of Forth programmers are what we may call hard Forth or low level hardware coders. However, in this presentation, I shall demonstrate that the metaprogramming techniques derived from Forth can be applicable to many high level programming languages and frameworks.

04:15

We use the term soft Forth to refer to Forth implementation on top of high level or the third generation programming languages. The idea of soft Forth is not new, starting with early implementations on top of C, then Java, as well as JavaScript, Rust, Haskell, Golang and even LISP. However, our implementation called Phoscript spelled P H O S C R I P T, taken from the Greek word meaning light, aims at integrating with the function libraries of the host programming language, instead of implementing the full vocabulary of Forth, in contrast to other implementations of soft Forth.

05:05

As you can see on the bottom of the screen, they are the PHP Phoscript library on the left and JavaScript Phoscript library on the right.

They essentially consist of these steps: firstly, splitting a space delimited string into tokens or words in Forth terminology; secondly, pushing non function word or token on to the stack; or executing the function in the host programming language by calling call_user_func in PHP at line 1272 on the screen or eval in JavaScript in line 595. Note that, we have introduced a colon suffix for function words that are mapped directly to the host programming language.

Due to time constraint, we shall not delve into colon definition words or aliases, branches, conditions and other more complex code structure. The code sample is available on our github account which is given together with the notes of this presentation.

06:25

So the first demo program for Phoscript is an AJAX script, which sends a simple addition script from the browser front end to the PHP back end, where PHP calculates the sum and returns the result to the front end.

Eventually, the same concept has been extended to cover React Redux and even HTML, which I suspect no one has attempted to convert into Reverse Polish Notation.

- http://phos.epizy.com/smashlet/

07:00

Here is simple demo on how Phoscript reverse polish notation generates HTML code.

At the bottom of the screen is a Linux terminal where the Phoscript commands are entered as php command line arguments.

The script simply produces a one cell table comprising the word "jack".

The HTML output is shown at the bottom of the screen.

- https://github.com/udexon/PhosChat

07:30

Three.js is a popular 3D animation library for WebGL.

We have modified the original demo to include an additional tank body following the original, as well as an additional target.

We then created two function words tank: and target:

A JavaScript statement m = new Phos () creates a object m so that we can access the internal functions of Phos ()

The function word tank: is used to create one additional tank body following the existing chain of bodies.

- https://github.com/udexon/GEISHA/tree/main/threejs/Phos

08:15

Jitsi Meet is a video conferencing app like Zoom but fully open source.

Its source code from github can produce a web app, Android app as well as iOS app.

The red text cvst: cube_cvst: and so on in the browser console on the left of the screen are Phoscript words which maps to WebGL canvas and Three.js functions.

- https://github.com/udexon/jitsi-meet

08:50

What you are seeing here, on the right hand side (of the screen), where the cursor is, at the back here, is a darkened video stream, where it has been processed using the Tensorflow Bodypix algorithm. As you can see, there is a human figure here. It is trying to detect the human figure and showing the results in white silhouette, and it darkens the background.

It also has a chatbox area, where we can use the message box here to enter Phoscript commands. We can output the Phoscript output (results) here within the chatbox. So as you can see it is fairly sophisticated.

09:50

After seeing what Phoscript can do with JavaScript and other mobile and web frameworks, we shall briefly discuss Fire4x (FireForth), a project to put Forth into the Firefox browser.

As you may know, WebAssembly is a major project in browser technology, which in my not so humble opinion, is simply LISP in your browser.

In 2018, I attempted to put jonesForth into Firefox, hence Fire4x. Eventually, I decided I should put my energy into educating younger programmers concerning using Forth in high level programming first. Then I shall let them decide if they want to pursue FireForth, which I believe may bridge the soft Forth and the hard Forth world.

10:45

SymEngine, in our opinion, is a modern attempt on computer algebra that began with SymPy, a project based on Python, which marked a departure from the traditional LISP based computer algebra systems.

We have forked SymEngine and named it as SymForth.

As shown at line 313 in the video, C++ allows us to define a stack of Reference Counted Pointers (RCP), a generic data structure that is used to represent algebraic terms.

Here we have an example of (x + 5) to the power 5, entered through command line in Reverse Polish Notation at the bottom of the screen, which yields 5 x y power 4 plus 10 x squared y cubed and so on as shown above the RPN expression.

- https://github.com/udexon/SymForth

11:55

Our next example is a fork of an underappreciated project called BTOSG by github user miguelleitao that combines the Bullet physics simulation engine with OpenSceneGraph. There have not been many attempts at combining physics simulation and scene graph at C++ levels, perhaps because it is more conveniently done at higher level such as Python, as demonstrated by PyBullet, a Python project utilizing the Bullet library.

In the original core car.cpp, there is only one car falling from the sky.

We have added a Phos word newcar: with some other dummy code to produce an additional four cars.

The code in the video shows that we use C++ template at the bottom of the screen and function mapping at line 234, to map Phoscript word newcar: to C++ function sm_newcar.

- https://github.com/udexon/Usegrammer/blob/master/Phos_BTOSG_I.md

13:15

Viewers with sharp eyes and mind might have noticed a little note that said "Inverse of Dijkstra's Shunting Yard Algorithm" earlier in the presentation, which we did not elaborate, as its significance is only apparent after we have shown examples in various programming languages, as listed on the screen, quite a comprehensive range of languages in our opinion, developed based on the very Shunting Yard Algorithm that Dijkstra himself pioneered.

The existence and applications of the Shunting Yard Algorithm and its inverse lead us to formalize homoiconism, a body of algorithms and models based on the unique feature of Forth and perhaps a selected few other languages, where data and code can be represented in a uniform syntax.

Question to ponder: Did Dijkstra know of Forth developments and would he concur the path (!!) I have taken to confront the problems he highlighted at the beginning of this presentation?

14:35

Having demonstrated that code written in any programming language can be homoiconified or transformed into Phoscript function word, it is of course tempting to consider keeping all these homoiconic functions in a graph database and to serve as a model for artificial intelligence.

Video: wikipedia tree figure.

We wish to propose one last novel idea, that is by traversing upwards from a node in a tree, we may then extract the key or edge connected to the node concerned. Further, such key or edge is essentially a representation of the relationship of the child node and its parent. The key, edge or relationship may then be used to trigger one or more homoiconic functions.

A homoiconic function may then produce the results as one or more nodes in the graph database.

Then the cycle or upward traversal repeats itself.

We will leave it to the viewer of this presentation to figure out if this model proposed above, called Graph Enhanced intelligent Search with Homoiconic Actions or abbreviated as GEISHA would serve as a realistic candidate to model the human mind.

- https://github.com/udexon/GEISHA

15:45

If a discussion on human level artificial intelligence sounds too far fetched, then GEISHA in its reduced form at degree zero or G zero, may have a very practical application: Geisha Zero is

practically a universal software management system, covering all conceivable programming languages, managed using a graph database, combining the features of Forth and other modern software engineering tools.

In our next presentation, we shall demonstrate how Geisha Zero can be used to manage and extend Jitsi Meet, together with other image processing libraries such as Tensorflow and OpenCV, with examples in Phoscript code that can look as simple and elegant as the legendary Logo programming language.