




Rainbow Forth


Brad Nelson

January 26th, 2008






Disclaimer

- This is my first Forth implementation
 - I have undoubtedly made bad design choices
 - Chuck would no doubt be horrified with the ugliness of my design
- 



Overview

- The **colorForth** Dialect
 - Design Choices
 - Tools
 - Implementation
 - Demo!
 - Language Exploration
 - Future Directions
- 




*The **color**Forth Dialect*

- Created by Chuck Moore circa 2000?

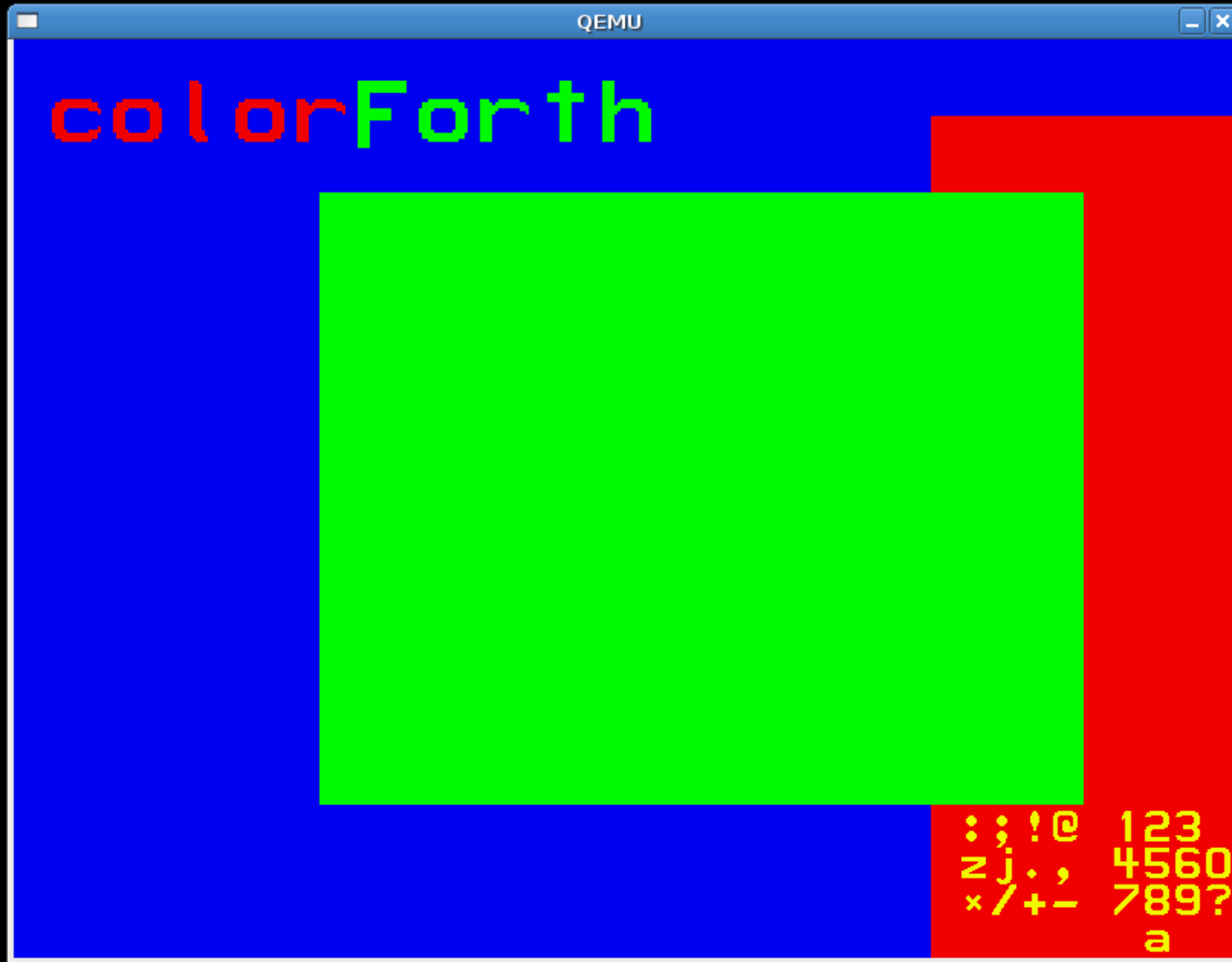




colorForth Features

- Stand-alone! Includes operating system.
 - Compact! 2K bytes for core software.
 - Fast! Optimized object code.
 - Simple! Applications stored as source. No object library.
 - Innovative! Text compressed and pre-parsed.
 - Unique! 27-key Dvorak keyboard.
- 

colorForth Startup Screenshot




colorForth Editor Screenshot

```
QEMU
macros
over ?dup 4468b 3, ;
push 50 1, drop ;
pop ?dup 58 1, ;
- dof7 2, ;
for push begin ;
*next swap
next 75240cff
0next , here - + 1, 4c483 3, ;
-next 79240cff 0next ;
i ?dup 24048b 3, ;
*end swap
end eb 1, here - + 1, ;
+! ?lit if ?lit if 581' 2, swap a, , ; th
en 501 2, a, drop ; then a! 950401' 3, 0
, drop ;
nop 90 1, ;
align here - 3 and drop if nop align ; t
hen ;
or! a! 950409 3, 0 , drop ;
* 6af0f 3, nip ;
*/ c88b 2, drop f9f72ef7 , nip ;Sct wrgx
/mod swap 99 1, 16893ef7 , ; ludr
/ /mod nip ; -mc+
26d /mod drop ; edit x.i
```




colorForth Colors

- comment
 - create
 - execute
 - compile
 - inline
 - variable
- 



colorForth vs Traditional Forth

- word --> (word)
 - word --> : word | create word
 - word --> [word]
 - word --> word
 - word --> ' word compile,
 - word --> variable word
- 




colorForth Advantages

- No look-ahead needed for parsing
- All places that create words are marked
- Fewer line noise characters
- Easier to understand compiler
- Color replaces parsing for both humans and compiler





colorForth Drawbacks

- Non-standard
 - Parser is less changeable than Forth
 - Incompatible with conventional operating systems
 - Dvorak scares off QWERTY users
- 




Introducing...

Rainbow Forth






Differences from colorForth

- 64x16 ASCII source blocks
 - Invisible color tags after each word
 - No shadow blocks
 - C compatible register choices
 - Bootstrapped from C instead of Assembly
 - More conventional word definitions
 - QWERTY keyboard input
 - literal is not called on yellow-green transition
 - Hexadecimals are marked with 'h'
 - Linux and Windows support (not bootable)
- 



Major Design Choices

- Source Block Format
 - Memory Layout
 - Dictionary Format
 - Registers
 - Bootstrapping Method
- 

Source Block Format


- 64 x 16 = 1,024 Blocks
- Spaces indicate color as a suffix
- 32 is white, 255-250 are red-magenta
- Spacial control is retained (and no waste)
- Example:

```
square n--n**2 dup * ;
```

```
square_ n--n**2_ dup_ *_ ;_
```




Memory Layout

- C Call Stack for RSTACK
 - Small DSTACK
 - Code Heap (for dictionary words)
 - Data Heap (for variables and allot)
- 




Dictionary Format

- Linked list of Dictionary Entries
 - Dictionary Entry
 - Next Word
 - Code Address
 - Data Address
 - Is Macro?
 - Smudged
 - Name Length
 - Name Characters
- 




Register Usage

- EAX - scratch
 - EBX - Top of Stack
 - ECX - scratch
 - EDX - scratch
 - ESI - unused
 - EDI - Execution Context Pointer
 - EBP - Data Stack Pointer
 - ESP - Return Stack Pointer
- 




Bootstrapping Method

- Implement editor in C
 - Implement minimal compiler harness in C
 - No assembler in compiler harness
 - Dictionary structure defined in C
 - Harness executes Forth words to generate code
 - Harness passes entry points for OS to Forth
- 




Tools

- Subversion
 - Debian Linux
 - Cygwin
 - Sourceforge.net
- 



Subversion

- Version control system
 - Successor to CVS and RCS
 - Keeps a snapshot of each change
 - Allows Internet collaboration
 - Checkout a copy with:
`svn checkout http://rainbowforth.svn.sourceforge.net/svnroot/rainbowforth`
 - Keep it up to date with:
`svn up`
 - Submit changes with:
`svn commit` (*requires sourceforge.net account*)
- 



Debian Linux

- Popular Linux Distribution
- Install developer tools with simple commands:

```
aptitude install gcc make ncurses-dev
```






Cygwin

- Unix-like environment on Windows
 - Includes GNU C compiler
 - Includes GNU Make
 - Includes Subversion client
 - Native Windows applications supported
 - Great for cross platform builds
- 




Sourceforge.net

- Free hosting for Open Source or Public Domain Projects
 - Bug Tracker
 - Subversion Server
- 



Implementation

- Overview
 - Color Console
 - Editor
 - Compiler Harness
 - System Call Access
- 




Implementation Overview

- < 2,000 total lines of C
- < 600 lines in compiler harness
- No register assumptions made
- Dictionary format is fixed
- 22 blocks in core system






Color Console

- Ncurses on Linux
 - WIN32 Console API on Windows
 - Wrap in a common interface
 - Supports Color, Underline/Reverse Video
- 



Editor

- Block at a time
 - Saves as you go
 - Arrow keys move cursor
 - PgUp and PgDn move between blocks
 - Ins, Del, Backspace
 - Cut (Ctrl-X), Copy (Ctrl-C), Paste (Ctrl-V)
 - Ctrl-G to just to a block number
 - Ctrl-R to run a block
 - F2-F8 to select color
 - Ctrl-Q / Escape to Quit
- 

Editor Interface

block=13 column=0 row=1

COMPARISON TO ZERO **macro** DEFINITIONS

```
_compare1 83h c, FBh c, 00h c,    cmp ebx,0
              0Fh c,                ; (first of of set?)
_compare2   c, FBh c,            (second half of set?)
              0Fh c, B6h c, DBh c, ; movzx ebx,bl
```

```
0= _compare1 94h _compare2 ;
0<> _compare1 95h _compare2 ;
0< _compare1 9Ch _compare2 ;
0> _compare1 9Fh _compare2 ;
0<= _compare1 9Eh _compare2 ;
0>= _compare1 9Dh _compare2 ;
```

forth


F2=create **F3=execute** **F4=compile** **F5=inline** **F6=lookup** **F7=variable** **F8=comment**

Ctrl-X=cut Ctrl-C=copy Ctrl-V=paste

Ctrl-G=goto Ctrl-R=run Ctrl-Q=quit



Compiler Harness

- C code compiles by calling a few Forth words
 - These Forth words are defined using a small set of built-in words
 - Small table of C entry points allow Forth to access system resources
- 




Harness Required Words

- execute-forth
- literal
- compile,
- variable






Harness Supplied Words

- macro
 - forth
 - smudge
 - unsmudge
 - b,
 - windows?
 - load
 - thru
 - heap-dump
 - word-dump
- 

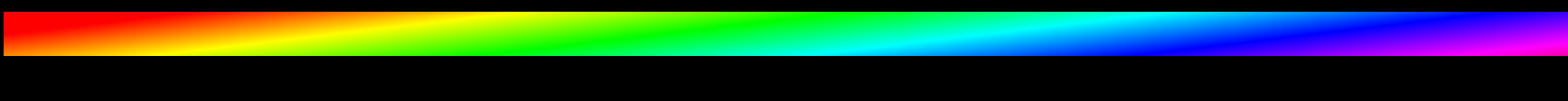


Harness Supplied Entry Points

- load
 - print_number
 - key
 - emit
 - library_load
 - library_symbol
- 




System Calls

- Windows Raw System Calls
 - Linux Raw System Calls
 - Windows Dynamic Libraries
 - Linux Dynamic Libraries
- 



Windows Raw System Calls

- How you call them changes in each version
 - It varies by processor version!
 - int 2Eh (intel pre-pentium II, amd pre-64-bit)
 - sysenter (intel 32/64-bit)
 - syscall (amd 64-bit)
 - Call number goes in EAX, but the numbers change!
 - NtWriteFile (0xC8 on WinNT4, 0xED on Win2000, 0x112 on WinXP, 0x11C on Win2003, 0x164 on Vista)
- 




Linux Raw System Calls

- int 80h call the system
- Call number goes in EAX
 - read=0, write=1, open=2, close=3, etc...
- Parameters go in ECX, EDX, ESI, EDI, EBP
- Return value in EAX





Windows Dynamic Libraries

- HANDLE LoadLibrary(const char *name);
GetProcAddress(HANDLE, const char *name);
 - For example:
 MessageBeep(int style);
 lives in USER32.DLL
- 



Linux Dynamic Libraries

- `void *dlopen(const char *name, int flags);`
`void *dlsym(void *h, const char *name);`
- For example:
 - `double cos(double);`
 - lives in `libm.so`






Demo

D**E****M****O**





Language Exploration

- Lookup color
 - Auto-literals
 - CREATE DOES>
 - Do we need macros/**cyan**?
 - Should macro be one-shot?
 - How to handle Strings?
 - push pop vs >R R>
- 



Lookup Color

- Need a way to get address of a word
- One primary/secondary color left **blue**
- Should go on compile time stack
- Example:

output-emit

to-printer printer-emit literal output-emit ! ;


to-screen screen-emit literal output-emit ! ;

emit output-emit @ execute ;





Auto-literals

- Should yellow-green transition compile a literal?
 - Example:
 - ~pi 22 7 / ; (colorForth)
 - ~pi 22 7 / **literal** ; (Rainbow Forth)
 - Assumes compile time calculations are more common than state changing words
 - Useful in 2 out of about 20 current uses
- 



CREATE DOES>

: defining-word create allocation does> action ;
defining-word instance-name

defining-word allocation does> action ;
instance-name defining-word





CREATE DOES> example


```
: pair create , , does> dup cell+ @ swap @ ;  
5 7 pair p
```

```
pair , , does> cell+ @ swap @ ;  
5 7 p pair
```






*Do we need macros/**cyan**?*

- Macros allow inline code without direct compiler support
 - Cyan highlights that something special is going on
 - Someone using a word doesn't need to guess what color (green usually is right)
 - Macros and regular words can coexist
 - `max 2dup < if nip else drop then ;`
is more awkward than
`max 2dup < if nip else drop then ;`
- 



Should macro be one-shot?

- **macro** and **forth** have state, much like **hex** and **decimal**
 - Maybe **macro** should be one-shot like **immediate**?
 - Or separate one-shot word **:macro**?
- 



How to handle constant Strings?

- Add a new color?
- Get strings only from blocks or data files?
- Parse quotes?
- Reuse red compile words !!!!
- Examples:


hi HelloWorld! last-name literal ;

XOpenDisplay 1 cimport return-value ;





push pop vs $>R R>$

- Looks less like line-noise
 - Matches assembler push/pop intuition
 - Doesn't look like a comparison
 - No shift key
 - But $>R R>$ is shorter
- 



Future Directions

- Peephole Optimizer
 - Rewrite editor in colorForth
 - Bootable Version
 - Cross Platform Graphics Wrapper
- 