Coming Back to Forth

Andrew McKewan

Silicon Valley FIG - Forth Day - 15 Nov 2025

1984: Learn Forth by typing…

fig-FORTH 8086/8088
ASSEMBLY SOURCE LISTING

RELEASE 1.0

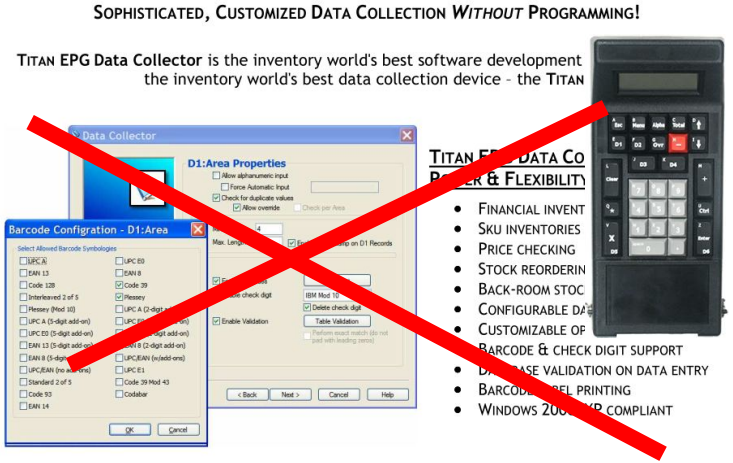
WITH COMPILER SECURITY
AND
VARIABLE LENGTH WORDS

MARCH 1981

1985: A Programmer! Discovered F83. Created the “Universal Data Langage” (aka 1806 Forth) so customers can program a hand-held data-collection device.

1802 Assembly Language

1806 Forth



And here we are in 2025

1988: California here we come! Dorado Systems (Bill R.), access-control readers and systems, 100% Forth tools, 68xx micros. FIG, FORML, community.

FEB 92 INSTALLATION MANUAL NA 3417 Iss 1
Model 611 Pass-Through Reader Page 1 of 7

1 IDENTIFYING THE PRODUCT

The Model 611 Pass Through Reader reads magnetic stripe security cards into access controllers. It accepts ABA format cards and transmits to the host in clock and data format.

A label located on the internal, rear surface carries part designations. This model has part number 3110-0611.

2 APPLICATION AND SPECIFICATIONS

The Model 611 card reader reports to any host controller that accepts ABA clock and data output format and supplies 5 volt power to the reader.

The reader is 6.00" L x 1.35" H x 1.75" W. Its footprint with the adapter plate is 7.00" H x 2.75" W. It operates over the temperature range of -35 to +66 degrees Celsius (-31 to 150 degrees Fahrenheit).

The Model 611 operates in indoor and sheltered outdoor conditions. The reader must be protected from rain and snow.

The cable distance from the host must not exceed 500' of #18 wire. Power at the reader must be in the range 4.75 to 5.25 volts. The Model 611 will operate on unshielded wire; some

1993: Finnigan (now Thermo Fisher), Win32Forth, large mass-spectrometer data system

Task: Convert a large (100s screens) GC/MS data collection and analysis application from a customized (by M.Perry) 16-bit MS-DOS F83 system to 32-bit Windows (not yet released). Hence Win32Forth.

2000: Automotive Infotainment (Harman/GM/Amazon), C, C++, Java, Rust, JS, Lua, ~~Forth~~

2025: Retired, now what?

WORDLE in Forth

Andrew McKewan

SVFIG Forth Day - 15 Nov 2025

Agenda

1. Introduction and motivation
2. Playing the game
3. Solving algorithms - guessing and pruning
4. Results
5. Source code: https://github.com/amckewan/wordle
6. Q&A

Wordle Solver in Forth

This is a Forth program that a provides a simple Wordle game and solver based on the online Wordle game that is now owned by the New York Times.

We use the original list of 12,972 5-letter words that are allowed as guesses. Game solutions are selected from a 2,315-word subset that we call the "hidden" list.

The goal is to solve all 2,315 puzzles, ideally without knowing the hidden word list.

There has been quite a bit of analysis done on Wordle and several people have written an optimal solver. According to The best strategies for Wordle, we know that we can solve all puzzles in 5 or fewer guesses with an average of about 3.4 guesses per puzzle.

Playing Wordle

Play nyt.com

Play forth game

Solving the Puzzle

Data types: words and scores

The Working set

Using the hidden list (2,315 words) or the full list (12,972 words)

Guessing - find the “best” word for each guess

Pruning - reducing the working set after each guess

Endgame strategy - getting close, time to shift gears

Wordle Words

```
5 constant len
: ww,  bl parse drop  here len  dup allot move ;

create wordle-words
  ww, aback  ww, abase  ww, abate ... ww, zonal ( #hidden )
  ww, aahed  ww, aalii  ww, aargh ... ww, zymic ( #words )

: ww ( w -- a )  len * wordle-words + ;

: find-word ( linear search ) ... ;
: w   ( -- w )  bl parse find-word 0= abort" Not in word list" ;
: [w]           w  postpone literal ; immediate

: w. ( w -- )    ww len type space ;
```

Scoring

```
\ Each letter can score grey, yellow or green, giving 3^5 possible scores.
\ Scores range from 0 (all grey) to 242 (all green), denoted by 's'.
\ We use '-' 'Y' and 'G' for input and display.

3 3 3 3 3 * * * * constant #scores

0 constant grey
1 constant yellow
2 constant green

\ literals
: color ( char -- color )  2 rshift  3 and  3 - negate ( ascii tricks ) ;
: >s ( a -- s )  0 1  len 0 do >r  swap count color  r@ * rot +  r> 3 * loop drop nip ;

: s   ( "w" -- s )  bl parse ?len >s ;
: [s] ( "w" -- s )  s postpone literal ; immediate

: s. ( s -- )  len 0 do  3 /mod swap S" -YG" drop + c@ emit  loop drop space ;

...
: score ( target guess -- score )
    init-scoring  score-greens  score-yellows ;
```

Working Set

```
\ The working set contains the words that could be the solution.
\ We start the game with all the wordle words then prune the set
\ after each guess by removing words that wouldn't get that score.
\
\ The working set always contains at least the secret (which is never pruned).
\
\ We implement the set as a linked list. Each entry in workset contains the
\ word # of the next entry or 0 to end the list.

create workset   #words cells allot

variable working    ( head of the working set linked list )

: all-words  ( add all words to the working set )
    0 working !  workset #words 1 do i over ! cell+ loop  0 swap ! ;

: next  ( w1 -- w2 )        cells workset + @ ;
: next? ( w1 -- w2 f | t )  next ?dup 0= ;

: remaining ( -- n )  0  working @ begin  swap 1+ swap  next? until ;

\ Prune the working set, removing words that wouldn't produce this score.
: prune ( -- )  ... ;
```

Guessing Algorithms

Simple - pick the first word from the working set

Random - pick a random word from the working set

Tally - use letter frequencies to find the best guess

Entropy - use entropy to find the best guess

And the winner is…

All puzzles solved with entropy guesser and endgame

```
use entropy-guess  hidden off  endgame on  solver


    0 Solved in 1
   13 Solved in 2
  283 Solved in 3
 1353 Solved in 4
  615 Solved in 5
   51 Solved in 6
    0 Failed

Average: 4.17    621.210 sec  ( about 50 ms per puzzle )
```

Solver Results

Guesser	Hidden	Average	Failed	Time (sec)
simple	on	4.21	52	3
	off	4.21	58	13
random	on	3.99	33	3
	off	4.33	162	11
tally	on	3.63	4	3
	off	4.24	44	14
entropy	on	3.62	12	695
	off	4.17	0	638

The entropy guesser

Algorithm from "Solving Wordle using Information Theory"
https://www.youtube.com/watch?v=v68zYyaEmEA

Information is measured in bits. If we can divide the solution in half, we have 1 bit of information. ¼ is 2 bits, ⅛ is 3 bits, etc.

Given target word T, compute the score you would give for every other word (in the working set) and record how many words got each of the 243 possible scores.

Let P = probability of a particular score (# words with that score / total words)

Let I = information bits = $\log_2(1/P)$ or $-\log_2(P)$

Let entropy = $\sum_{\text{scores}}(P * I)$

Find T with the largest entropy and use that as the next guess.

Endgame Strategy

We can get in a situation where we have 4 greens (one letter to guess) but more words remaining that we have guesses.

If this happens, we pick a word that has the most of the missing letters.

For example, on our fourth guess we get HOVER, and the working set contains LOVER, MOVER and ROVER. If we just pick at random we will fail a third of the time. Instead, we can use a guess like ALARM which has an L, M and R to narrow it down.

```
1 ABACK
2 DEFER
3 GIVER
4 HOVER
5 ALARM
6 ROVER Solved
```

Thank you!

- It was fun creating an application-specific vocabulary
- Managing > 2 stack items is difficult
- Interactivity is wonderful for exploring and developing ideas
- I improved my Wordle game
- I get to share what I did