Easy and appropriate to

[illegible]

To: P.Doctorow, M.Jabara, R.Foote, TopoTeam
From: Ed Wischmeyer
Date: July 9, 1983
Subj: Relationship of TopoSoft and TopoTeach

Introduction

Topoteach is generally agreed to be the software formerly known as Bonzo, comprising a number of features running on a non-Forth interpreter. Toposoft is currently envisioned as a collection of some of these features running with the MVP Forth interpreter. This memo documents the differences in the interpreters, and lists the features of Topoteach, some of which are appropriate for Toposoft.

Interpreters

The Topoteach interpreter differs significantly form the standard MVP Forth interpreter. The Topoteach interpreter allows
- forwards/backwards execution
- use of control constructs outside of definitions
- "define and do" operation
- single depth "stack" for parameter passing, including
  complete compile time error checking.

By comparison, the standard MVP Forth interpreter has
- branching constructs, including recursion
- deep parameter stack
- existing customer base and documentation.

The code for the Topoteach interpreter has been designed so that it will be easily compatible, in both philosophy and source code, with standard Forths. It is worth noting that MVP Forth was chosen for availability in the public domain in order to provide a quick, economical Forth system, and not for any technical merit. I hope that we will write an Androforth for our own purposes 4Q83.

Features in Topoteach

1. Menus
The menu structure provides a way of inputting characters to the Topoteach interpreter without typing. The menus are inappropriate for Toposoft because Toposoft will have a very large number of commands available. (M.Saari probably disagrees.)

2. Define & Do
Not transportable to Toposoft based on MVP Forth. Possibly doable on Androforth.

3. Speech
Easily and appropriately portable to Toposoft.

4. FROGraphics
This package is portable into Toposoft. Features are pen and background color selection, FROG and pen control, and simple drawing primitives (line, box, and dot). The FROG movement

semi-synchronized to 'bot movement should be portable.

5. Parameterized motion commands
Easy and appropriate to port to Toposoft.

6. Fixed Distance motion commands
Easy and appropriate to port to Toposoft.

7. Real time joystick motion commands
Easy and appropriate to port to Toposof.

8. Joystick teach
This requires mucking about with the input buffer. We can probably do this with little problem, but I wonder whether this should go into Toposoft.

9. Goto-it
This requires the FROGraphics in order to work. We could put this into Toposoft, but this would make the differences between the two rather minimal.

10. Topo Selection
This gives the user the ability to select which Topos respond to commands. This should go into Toposoft in some form.

11. Direct Topo control
This is the ability to directly formulate Topo commands on a byte-by-byte basis and ship them across. Topoteach does not have this feature, but Toposoft must have this capability.

12. Undo features
These Topoteach features will only work with the Topoteach interpreter.

13. Disk system
Toposoft should be able to read a Topoteach disk containing user-defined commands. (The source code has been defined for compatibility.) Similarly, a knowledgable user should be able to write and modify a Topoteach disk using just Toposoft, but this will not be very easy.

14. Joystick Editor
This requires full Topoteach disk capability. If Toposoft does not have full disk functionality, it cannot run the joystick editor.

15. Help Functions
These require a Topoteach disk to be mounted, and may require the Topoteach interpreter. (This part has not yet been defined).

16. Decompiler (unraveler)
This is designed to unravel Topoteach interpreter words only. It is possible to write a Forth decompiler, but we do not need to include that in Toposoft. We may choose to add it later.

17. Log file
Topoteach may allow the user to take a screen image, edit it, and save it on disk. This would be an outgrowth of the Topoteach disk system and editor. There seems to be little reason to transfer this to Toposoft.

18. Multi-processing

TopoTeach will support a primitive form of multi-processing. When one of the four hit switches on Topo is hit, one of four routines corresponding will be executed. This routine will run to completion, after which the interrupted process will resume. The four routines will have fixed names, and these fixed name routines will be definable only as aliases for existing routines (both system and user-defined). There will also be a wait-for-hit-switch command, which will cause the existing process to hold until any hit switch is pressed. The pressing of the hit switch will bypass the programmed hit-switch routines when a wait-for-hit-switch is pending. This simple multi-processing is easily portable to Toposoft.

Memory Budget

On the Apple, TopoTeach currently requires about 58K. Of this amount, the following is recoverable:

9k	headers in the hidden definition
3-5k	unused portions of MVP Forth
4k	ROM space on the Apple
1k	disk buffer
18k	total (appx), requiring 40k of memory

However, there is some amount of code which will be required to implement the disk software, editor software, and debugger. The bottom line is that there is a screaming chance of having a usable package on a 48k Apple.

When going to a machine other than the Apple, more space is recoverable:

8k	hi-res graphics on Apple
1.5k	lost space in low memory on the Apple

This means that there is a very remote possibility of having the full disk-base TopoTeach work on a Brand X machine with 32k. However, Brand X machines will also have quirks which will lead to unavoidable consumption of memory.

Fred Compatibility

TopoTeach should be easily compatible with Fred. There are a few areas which should be changed:

- drivers to talk to the IR link
- motion commands to reflect more limited capabilities of Fred (Fred can't translate and rotate simultaneously).
- more limited speech capability of Fred
- lack of hit-switches on Fred

Cassette and Cartridge Based Systems

These systems will require some major redesign in any case. A major effort will be required to properly implement a source code save and edit capability. There has been no effort expended on these systems, and there are not even any preliminary ideas available as yet.