# On the Design of Display Processors

Yelugoti Mohana Datta - IMT2016012

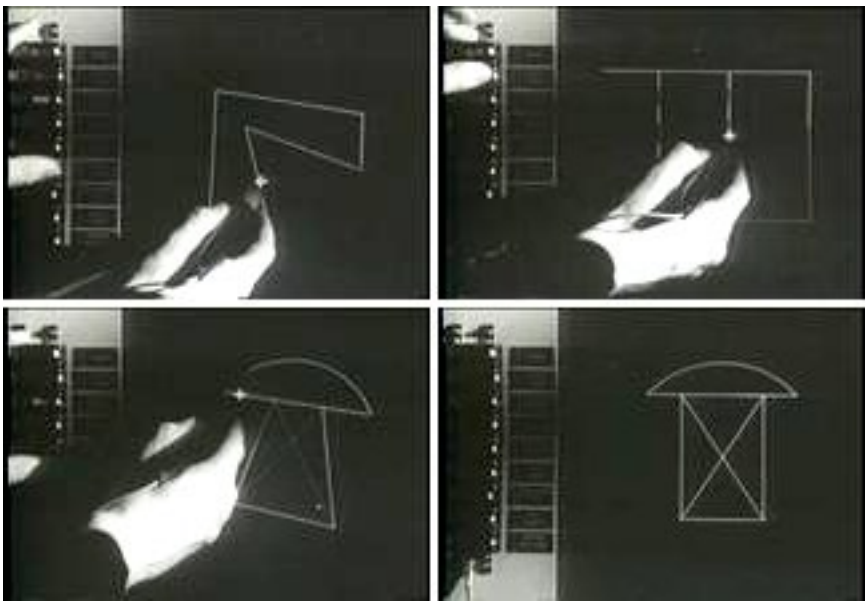
May 2, 2019

## 1 Authors

This paper was written by T.H Myer and I.E Sutherland with inputs from Bolt Beranek and Newman Inc.

- I.E Sutherland is a computer scientist and Internet pioneer, he was widely regarded as father of computer graphics.
- He received Turing award from Association for Computing Machinery in 1988 for invention of sketchpad, an early predecessor to GUI.

### 1.1 Sketchpad



## 2 Introduction

This paper talks about the flexibility and power needed in the data channel for computer display, it also looks at the design of the display processor (control part of the display) and how it was found that making successive improvements to design of display processor lie on a circular path. It also talks about the various challenges faced in associating the display with the core computer.

### 2.1 Context

- The display that the authors are designing is for SDS-940 time shared computer system.
  - SDS-940 was the first machine designed to directly support time-sharing. (Multiple users using single machine)

### 2.2 TX-O display

### 2.3 DEC-PDP

## 3 The Wheel of Reincarnation

When the authors decided to design the display processor, it got so complex and it resembled a full fledged computer with some special graphic features, and then a strange thing happened, the authors were compelled to add a second, subsidary processor, which itself began to grow in size, it continued and authors found that they were stuck in this never ending cyclic process. They called it wheel of Reincarnation.

### 3.1 Stages:

- Display from core's central registers.
- Introducing a data channel.
- Drawing lines and characters.
- Introducing
  - HALT command.
  - JUMP command.

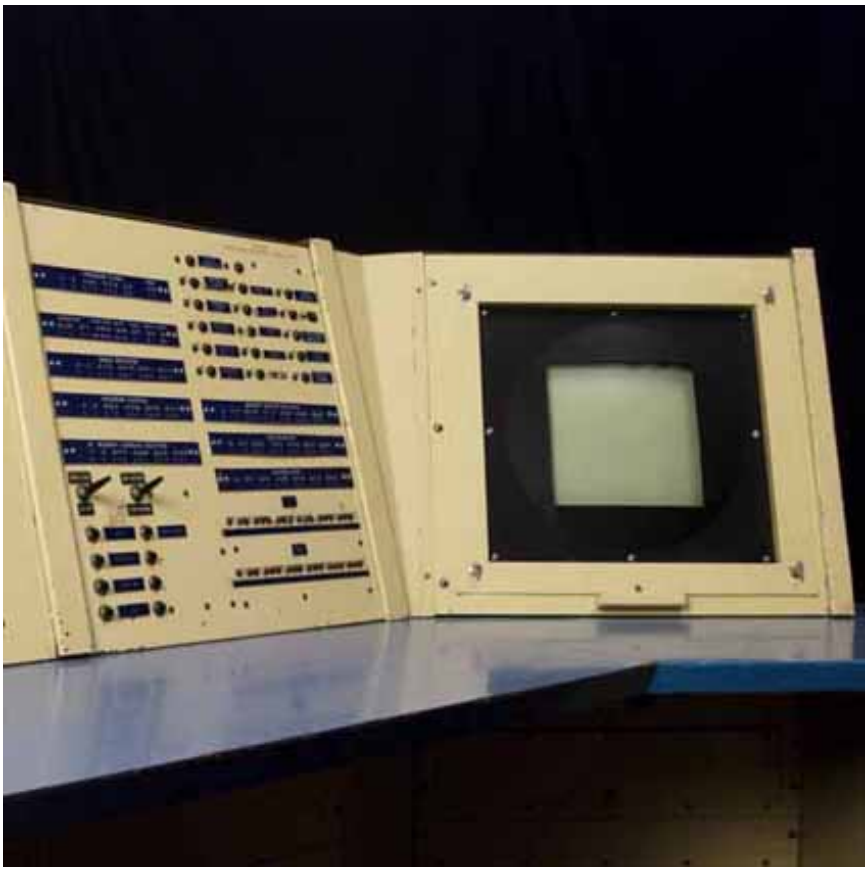


Figure 1: TX-0 display

Figure 2: DEC-PDP

- Subroutine feature for repetitive symbols.
- Store-exit command.

### 3.2 Realisation:

We should realize that display data channel is not a mere data channel, but is a processor.

Load Immediate and Flash (point)
Add Immediate and Flash (line)
Halt
Jump
Subroutine Jump
Store Subroutine Exit

### 3.3 Further improvements:

- Introducing stack.
- Adding conditional commands for interactive programs.
- Transparency by parameter passing.
- Global variables support.

### 3.4 Full-Blown processor

| | |
|---|---|
| Load Immediate and Flash | (point) |
| Add Immediate and Flash | (line) |
| Halt | |
| Jump | |
| Push-Jump | (subroutine) |
| Conditional Skip | (possibly more than one of these) |
| Push Parameters | (into stack) |
| Push X, Y Position | (into stack) |
| Pop | (restore top item from stack) |
| Add Immediate to Stack Pointer | |
| Load | (addressable: C (address) → X, Y) |
| Store | (addressable: X, Y → C (address)) |

Many of these commands would be included in a general purpose processor. In fact, to make the display processor general, for just a little more money, one can add:

| | |
|---|---|
| Execute | (addressable) |
| Complement | (for subtraction, and logic) |
| Shift | |
| Mask | (logical AND, OR, etc.) |

### 3.5

So, if we take a look at the design, we have built up display channel until it itself is a general purpose processor with a display. The display is tied directly to its processor, to generate a picture the display processor's central registers are used. In short, we have come exactly around once the wheel of reincarnation.

However, we have made significant developement, we have added support for many things in the process.

### 3.6

Now, we might argue that much of display processor's power is idle most of the time and that it is wasteful to tie up a general purpose processor merely to refresh a static display, therefore we might consider adding a *channel* to display processor, which can have some special commands to let it follow more complex data structures, **but** if we do this, we would be move into a second turn around the wheel.

Looking at commercial displays at that time, one can find many examples at various points around the wheel, but none stopping exactly at one revolution around the wheel.

## 4 General Questions

### 4.1 Questions:

Viewing this from a broader view, we see that there are mainly two questions:

- How closer should the display system be tied to parent systems?
- How much computing power should be included in the display processor?

The authors favor keeping the display close to the parent system, inspite of memory problems (if it had been far, it would have separate memory and we wouldn't face these).

### 4.2 How much computing power?

Depends on what we want to do with our display processor.

- The display processor must generate pictures from some form of internal representation, which may include multiple calls on display subroutines.
- Display element might generate pictures from computation rather than static representation in memory. (light-pens etc)
- Display processor might provide immediate feedback to user or handle simple interactive functions such as editing, light-pen tracking.

- The display processor should handle rotation, scaling etc when these are not handled by the display hardware.

### 4.3 Escaping the wheel

The view suggested by Daniel Bobrow that the display processor need not contain general purpose computing power, largely determined the design of the display processor.

General computing power should come from central resources of the system, if there are no enough resources to do particular thing it's fault of the system not the display. The display should mostly be concerned with displaying the points. This decision let's us escape from the wheel of reincarnation.

## 5 References:


- https://www.greatbigcanvas.com/view/dec-pdp-1-computer,1154199/
- https://en.wikipedia.org/wiki/History_of_computing_hardware
- https://archive.org/stream/sds-940/Image071017220051_djvu.txt
- https://en.wikipedia.org/wiki/Display_Data_Channel


**Thank you!**