From Commodore 64 to the Cloud

Lessons from 30 years of programming

Early 1980s

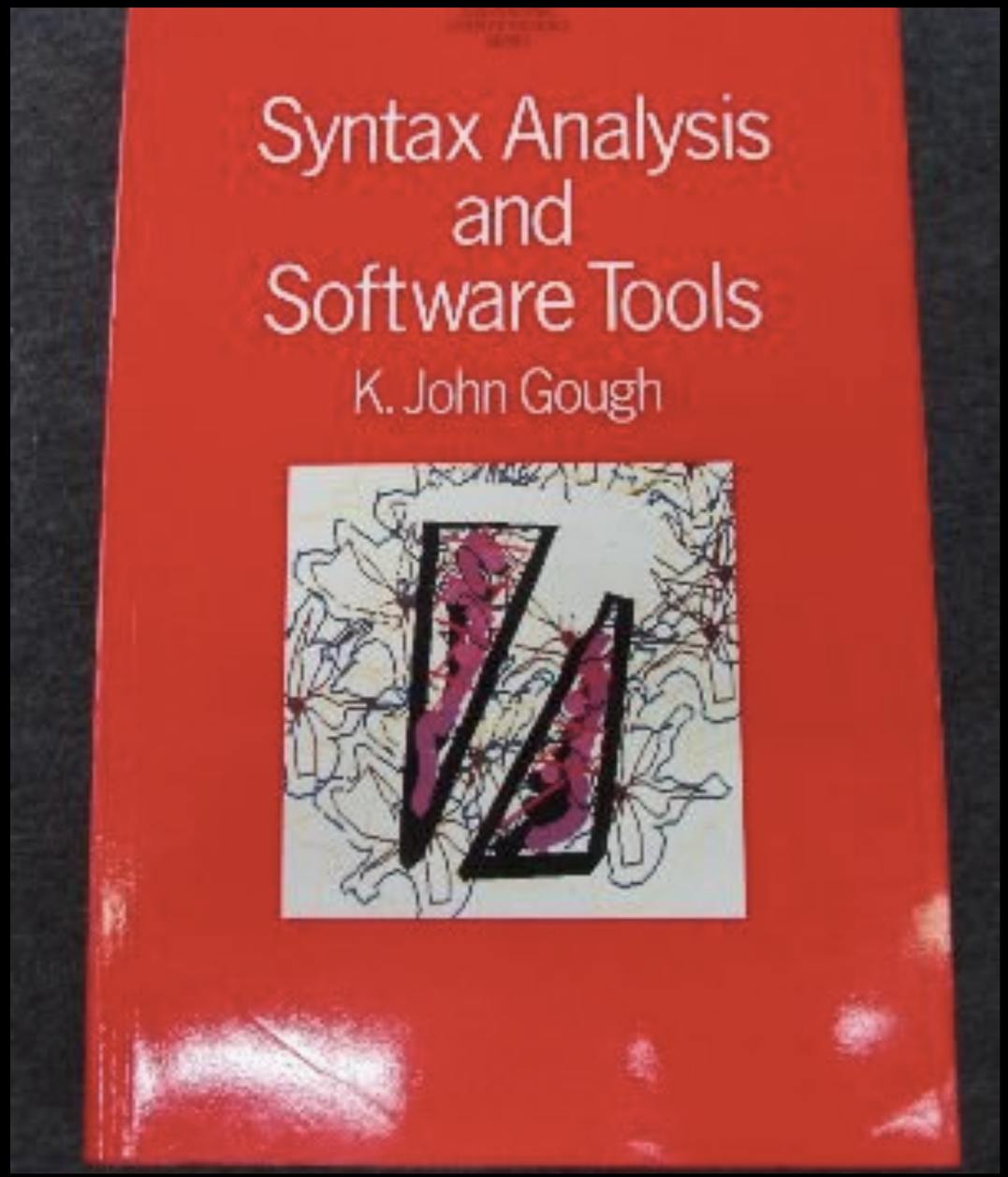
- Dick Smith Wizzard
- Commodore 64
- BASIC, peek/poke, machine code, assembly, graphics, games, electronic music, compilers.

Lesson - Fun!

Tell your kids to program games in JavaScript with HTML5/Canvas/WebGL
but ensure they understand that JavaScript is a low-level assembly language of modern machines

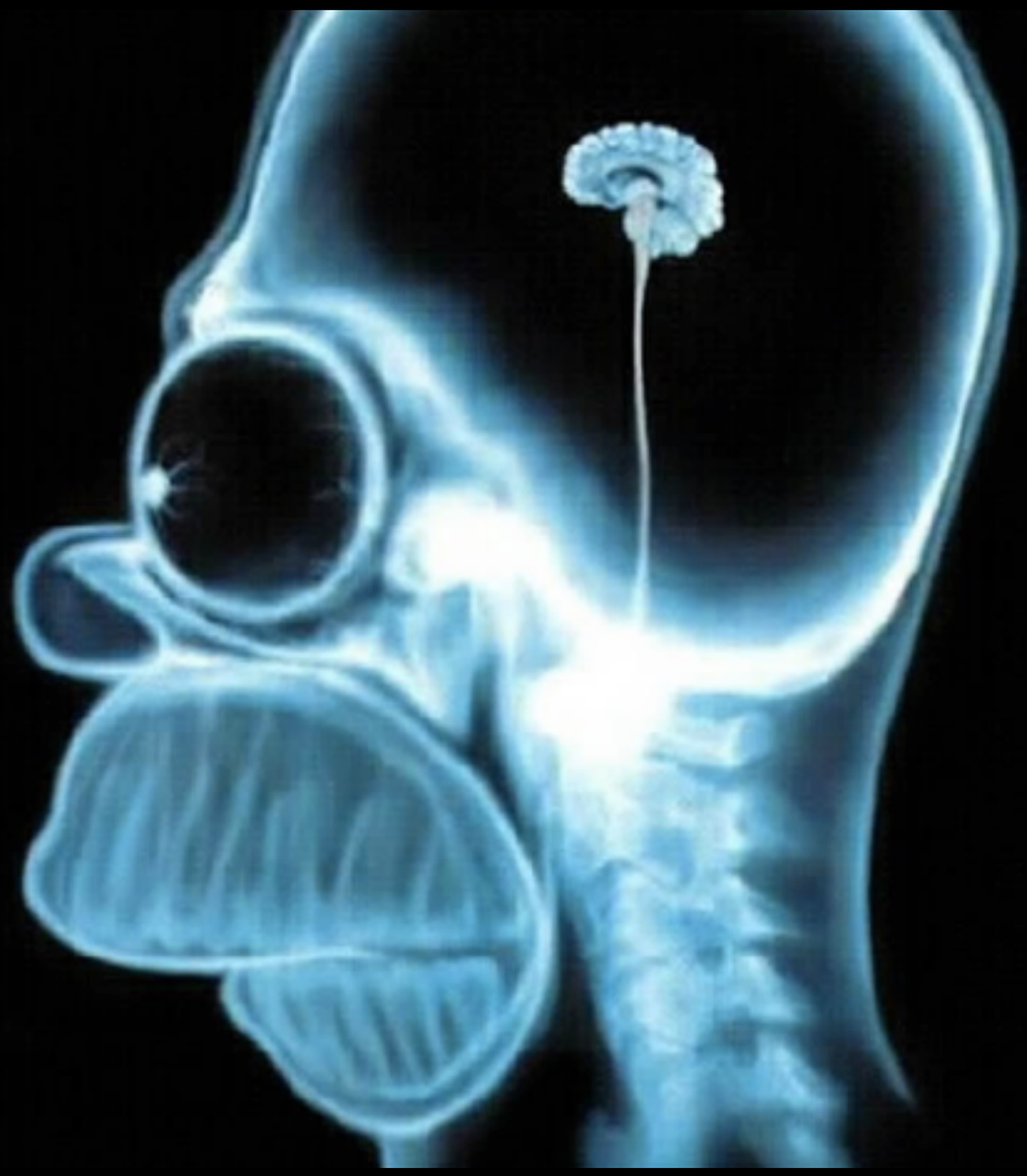
Late 1980s / Early 1990s

- QUT
- Modula-2 > Pascal
- VAX/VMS DCL
- Unix & C
- Gardens Point Beers Club



Lesson - Modules are important

Modula-2 modules are simple but useful. Sooo much better than C “header files” and preprocessor directives!



Lesson

Beer kills brain cells and lowers grades :)

Early 2000s

Python
Ruby
Smalltalk
Scheme
Common Lisp
Scheme
Dylan

Lesson - Macros

Macros are very useful for building extensible languages

Lesson - Continuations

Scheme’s first-class continuations tickle your brain

TODO: Read Oleg on why delimited continuations are the thing

Mid 2000s

Xavier Leroy on compilers, Caml Light, Caml, Objective Caml, OCaml

Lesson - High-level Programming Languages can be very efficient

i.e. OCaml is fast

but nag them about multicore ;)

Lesson - Pattern Matching

Algebraic datatypes are great for compilers

Lesson - Macros in ML family languages

camlp4 shows that macros aren't just for Lispers

Lesson - Better Modules

Modula-2 modules are great and simple. ML modules have yet to be improved upon (but check out MixML).

Late 2000s

Haskell, Laziness, Template Haskell

Lesson - Laziness for Modularity

John Hughes on “Why FP” is “Why Laziness”

Lesson - ad-hoc overloading

Can use type classes for CLOS style ad-hoc overloading

Early 2010s

- Proofs are programs
- Coq, Agda, Epigram, Idris, Cayenne, Ur
- Inductive datatypes
- Dependent types
- Module Systems

Lesson - The future of programming is dependently typed

- No longer need to statically analyse (aka infer or *guess*) the properties of your programs. Instead *state* them.
- TDD: Test Driven Development → Type Driven Development.
- Manuel Chakravarty calls for *Property Driven Development*. Yes, that would be *mathematical* properties.

Cloud

(aka “Cloud”)

i.e. Distributed Systems, Software Defined Infrastructure

Lesson - Reduce network hops

- A key lesson from distributed computing.
- Also applies to optimising the user-space to kernel-space interactions.
  Consider how the c10k problem is solved with libevent, libuv, NIO/NIO.2, over epoll, kqueue, AIO etc. This kind of efficient I/O system is baked into the runtime systems of Erlang, Haskell (via the Glorious GHC), Go, Rust, Java, Scala & Node.js.
- Single address-space Operating Systems help solve a similar problem when switching between threads/processes after a time-slice.
- L4 microkernel implements efficient inter-process message passing using hardware memory remapping techniques.
- Operating Systems as libraries (aka exokernels):
  - Mirage is OCaml on virtual metal
  - HaLVM is Haskell (via the Glorious GHC) on virtual metal

Lesson - Everything fails, deal with it!

… and everything will be fine 😀

These may help:

- replicas and redundancy
- app containers and PaaS, Docker, Kubernetes
- Raft, Zookeeper, etcd
- Erlang OTP

Questions?