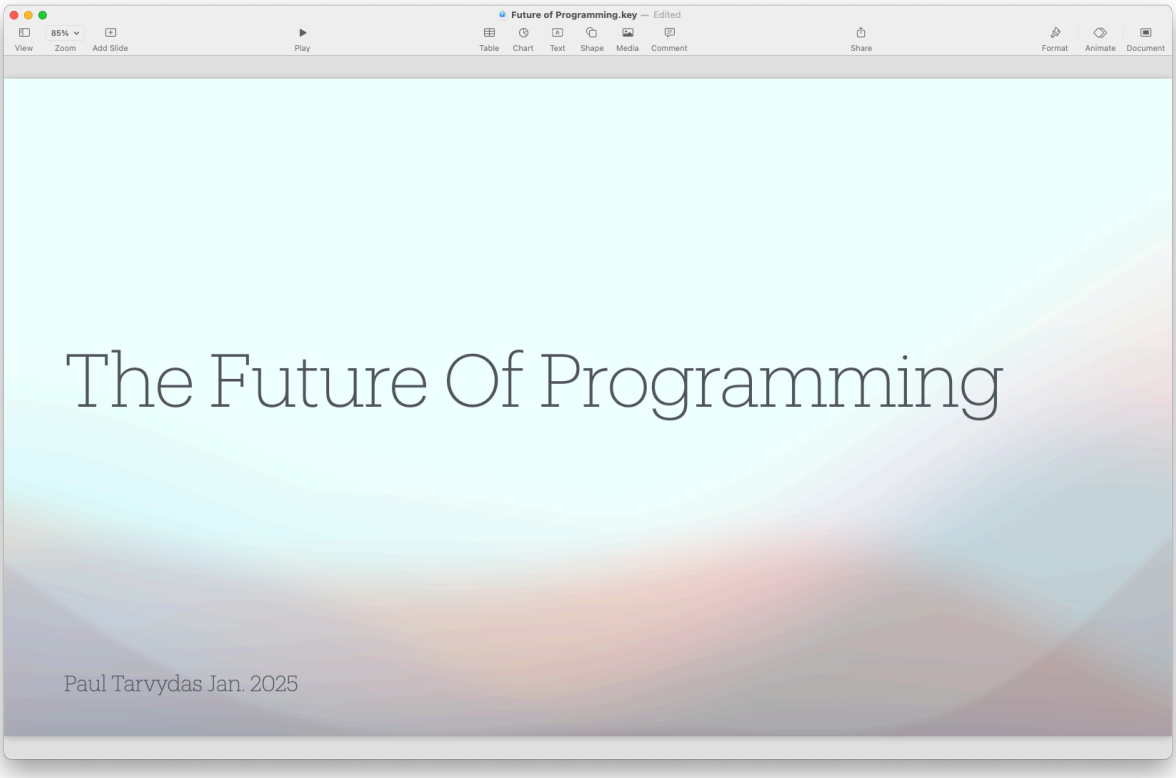
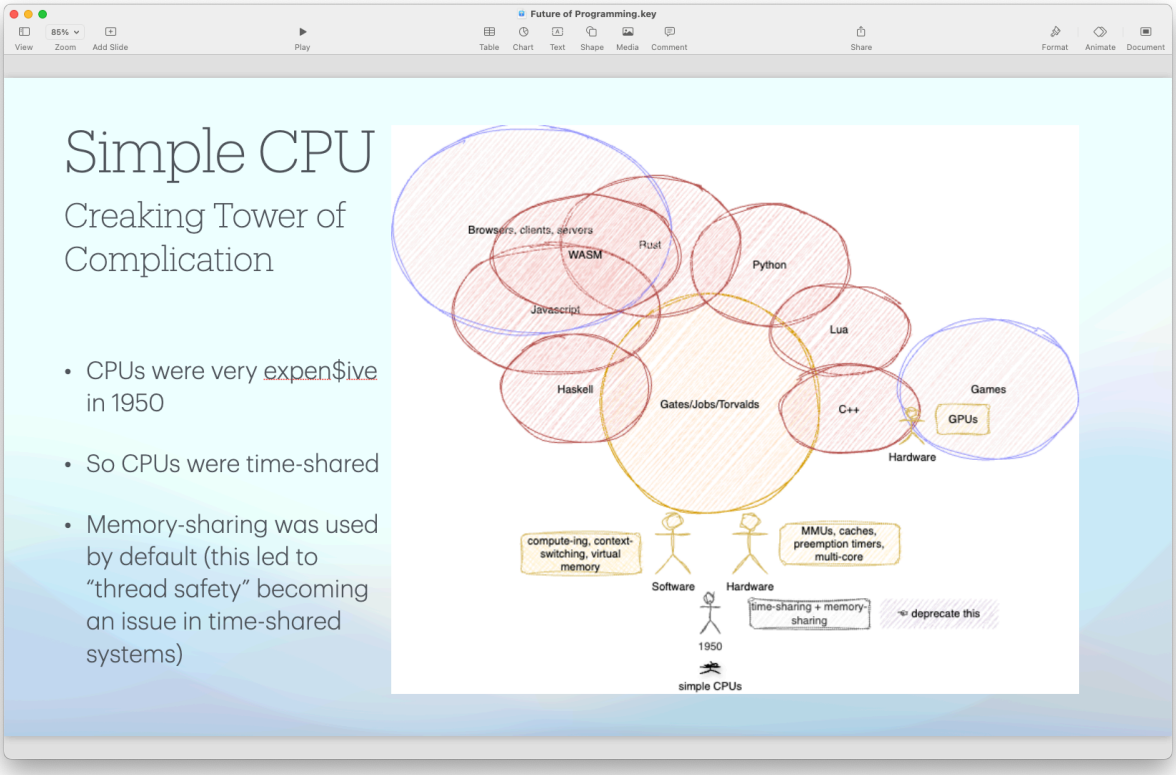
The Future of Programming

2025-01-02

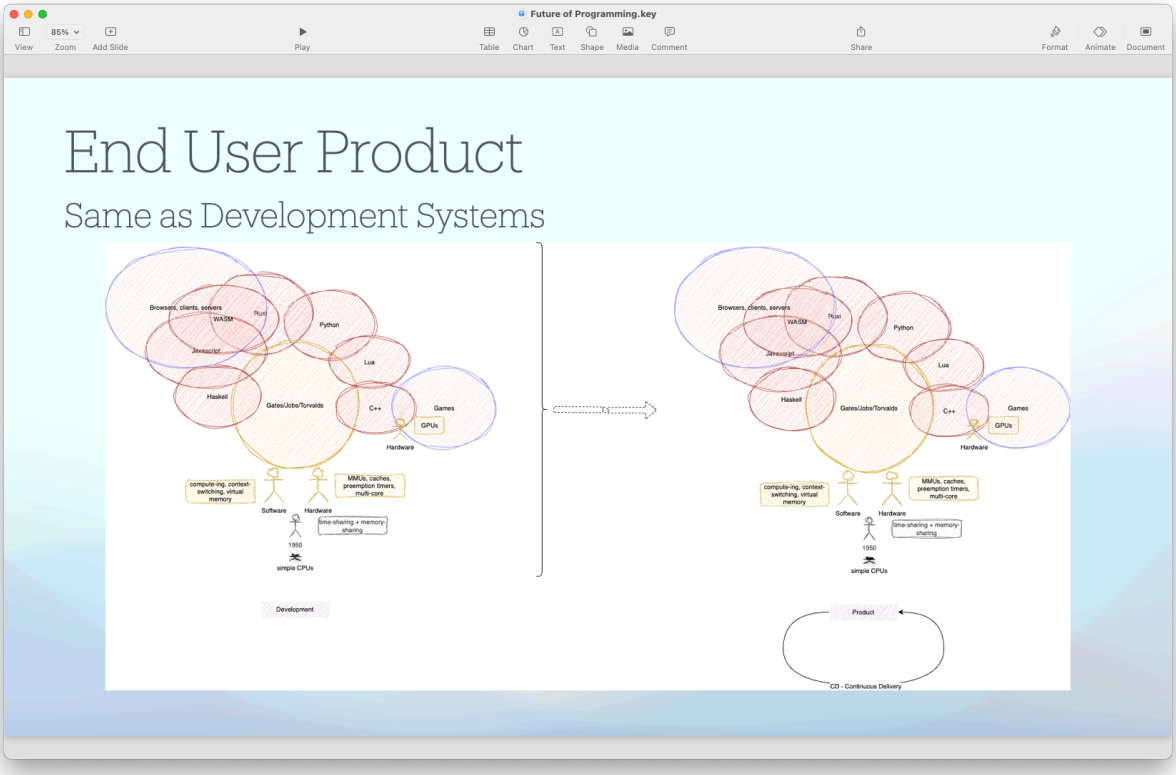


Slides from YouTube video (see reference below)

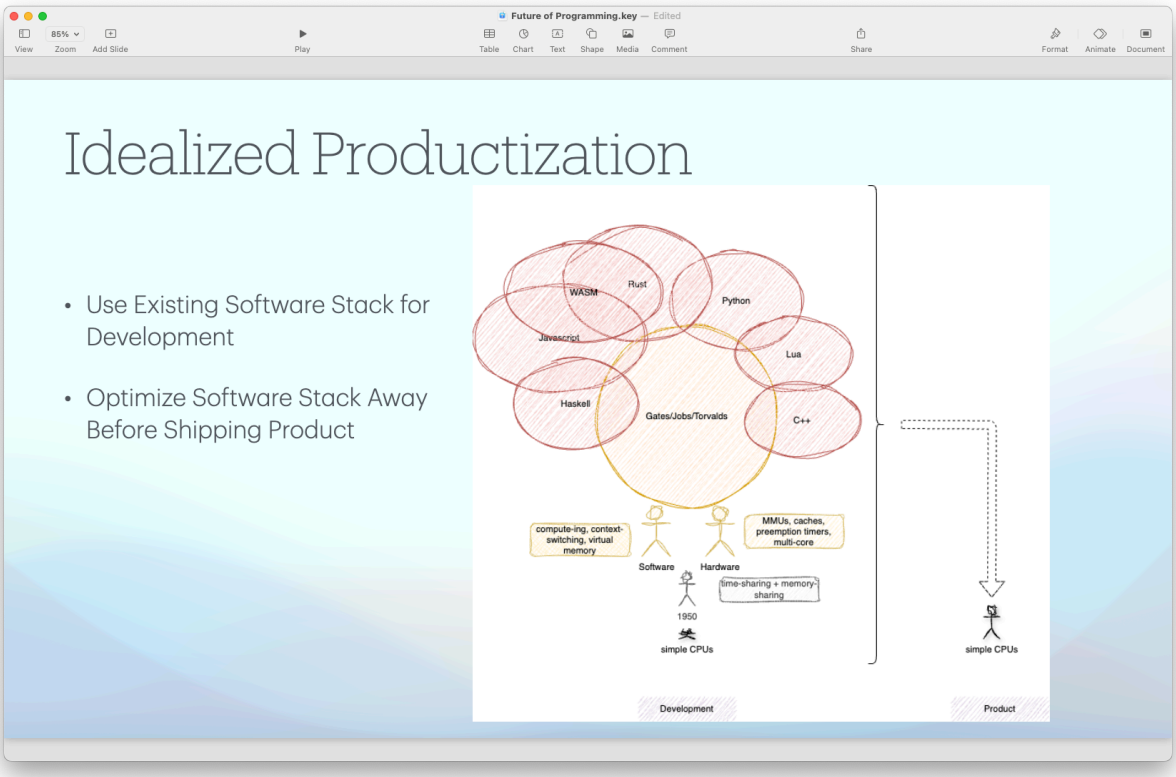
Simple CPUs


Future of Programming.key
Simple CPU
Creaking Tower of Complication
• CPUs were very expen$ive in 1950
• So CPUs were time-shared
• Memory-sharing was used by default (this led to "thread safety" becoming an issue in time-shared systems)
Browsers, clients, servers
WASM
Rust
Python
Javascript
Lua
Haskell
Gates/Jobs/Torvalds
C++
Games
GPUs
Hardware
compute-ing, context-switching, virtual memory
Software
Hardware
MMUs, caches, preemption timers, multi-core
time-sharing + memory-sharing
deprecate this
1950
simple CPUs

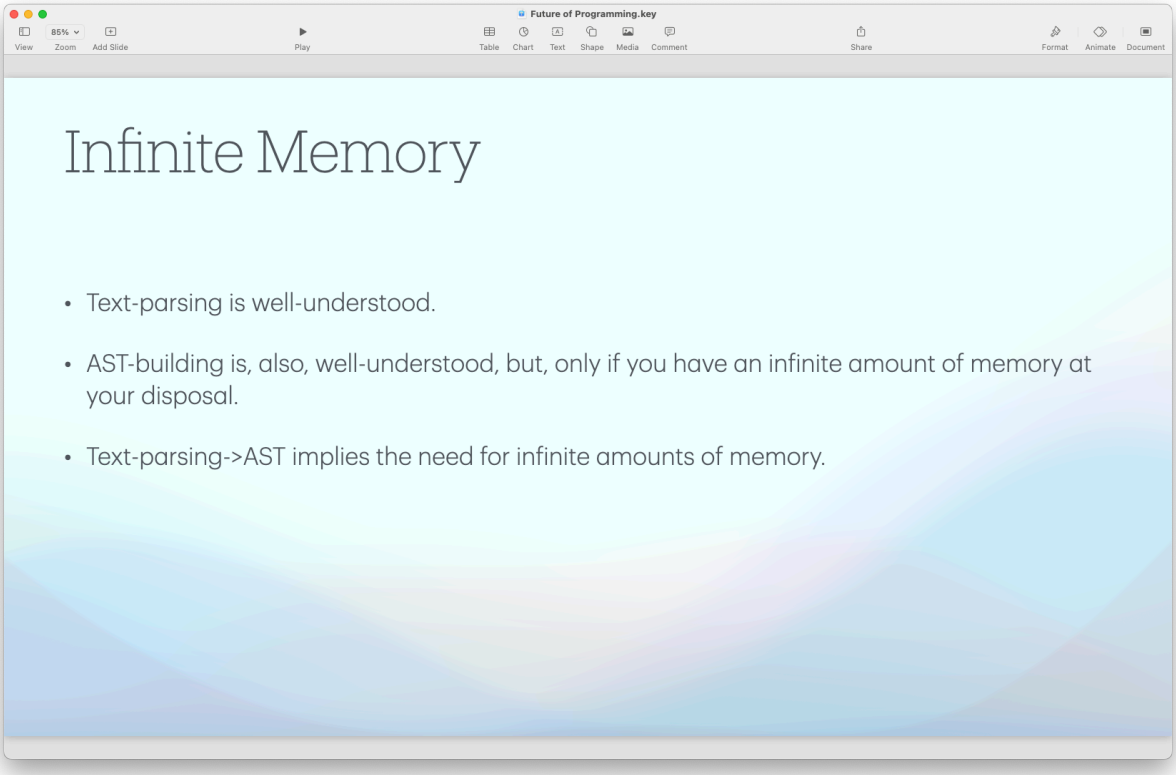
End User Product



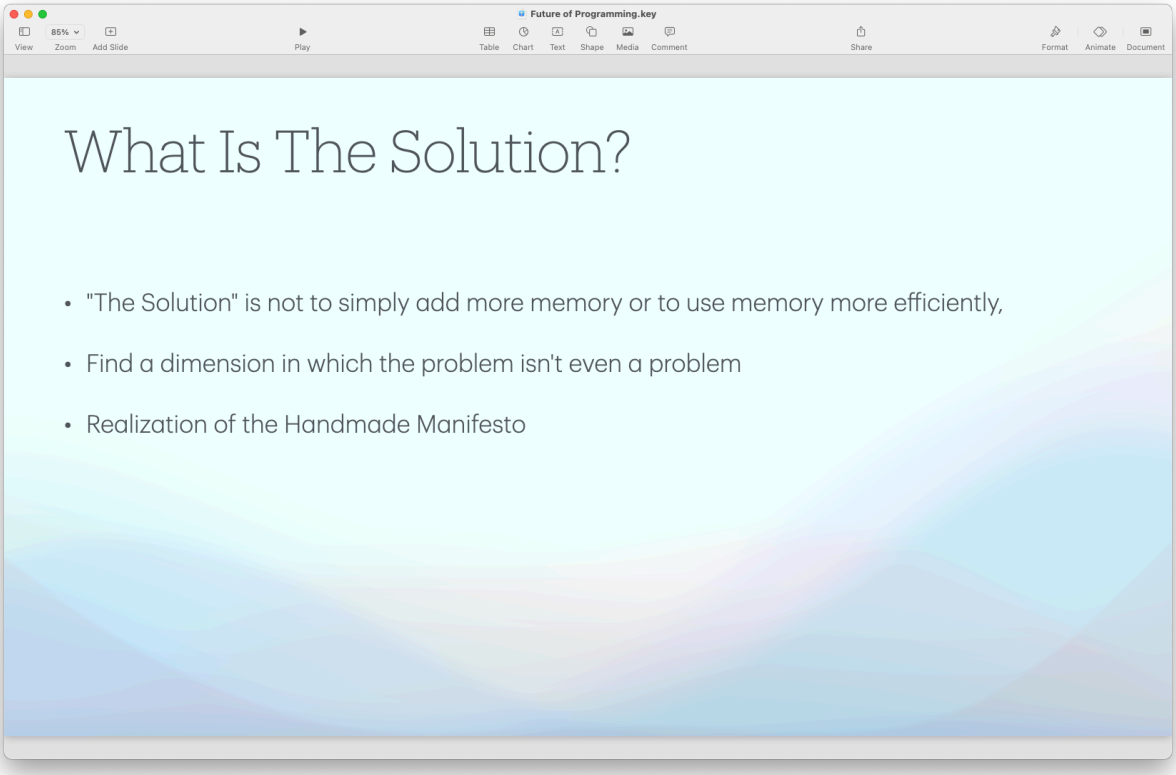
Idealized Productization



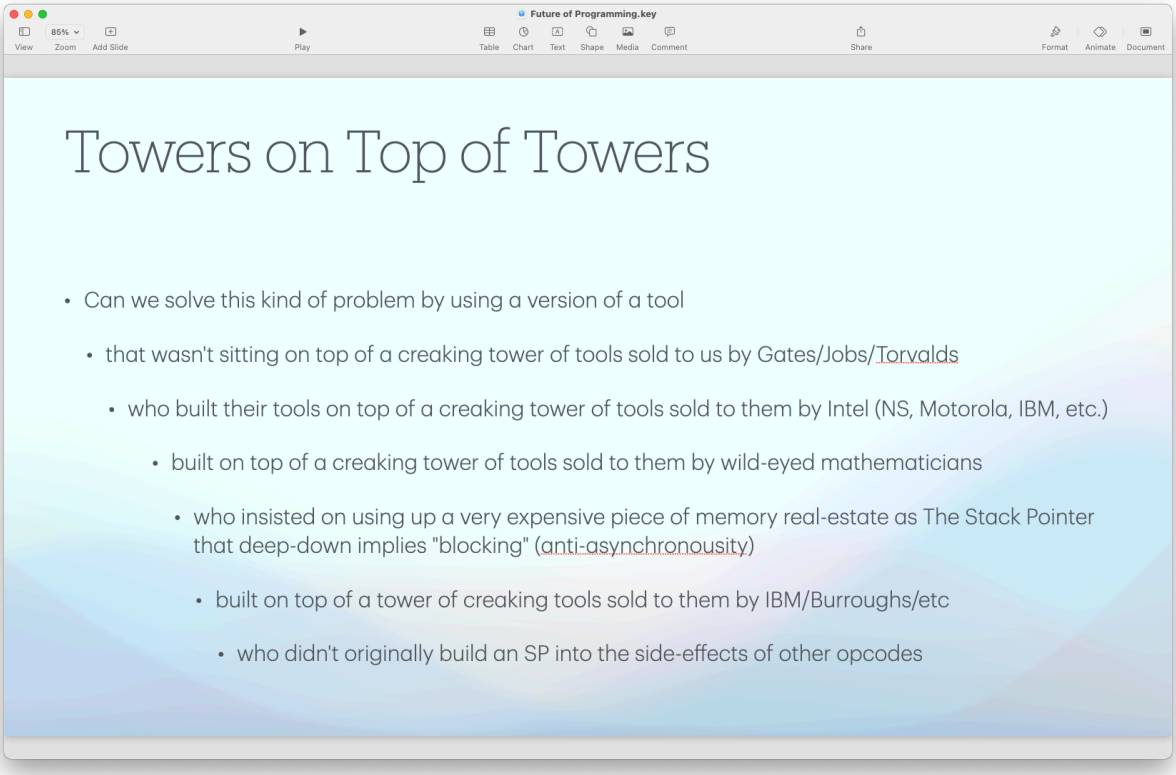
Infinite Memory



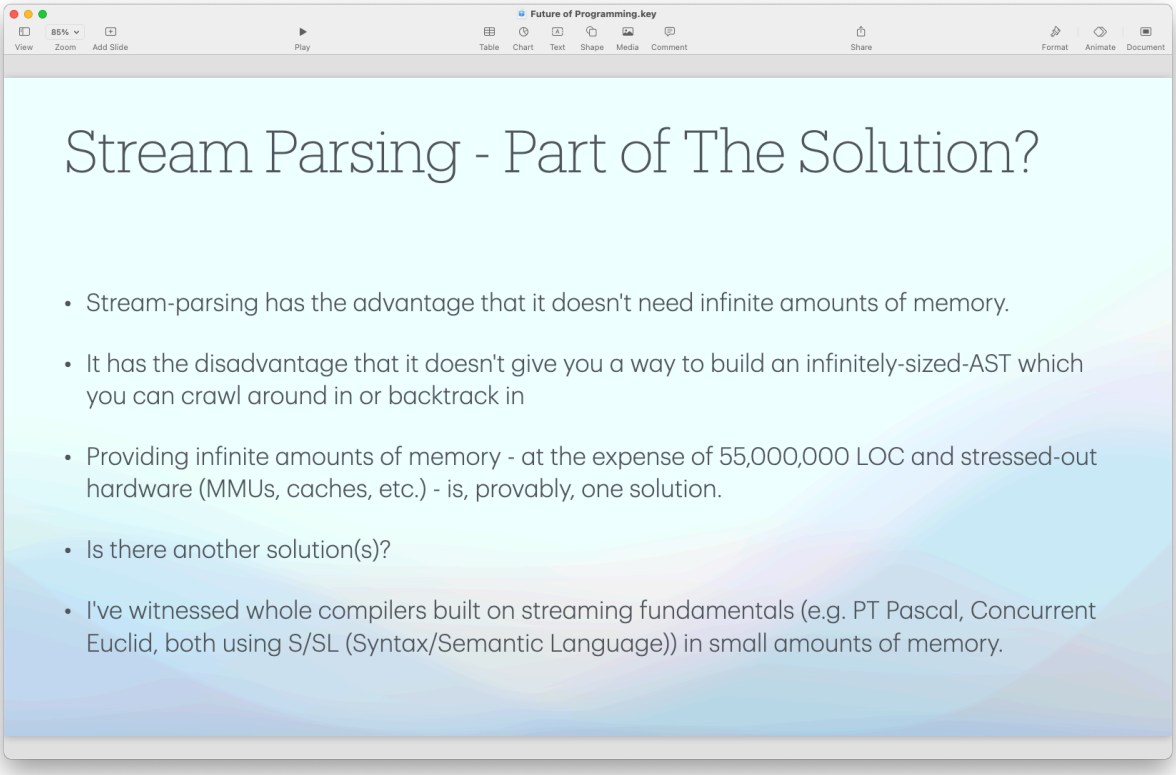
What Is The Solution?



Towers on Top of Towers

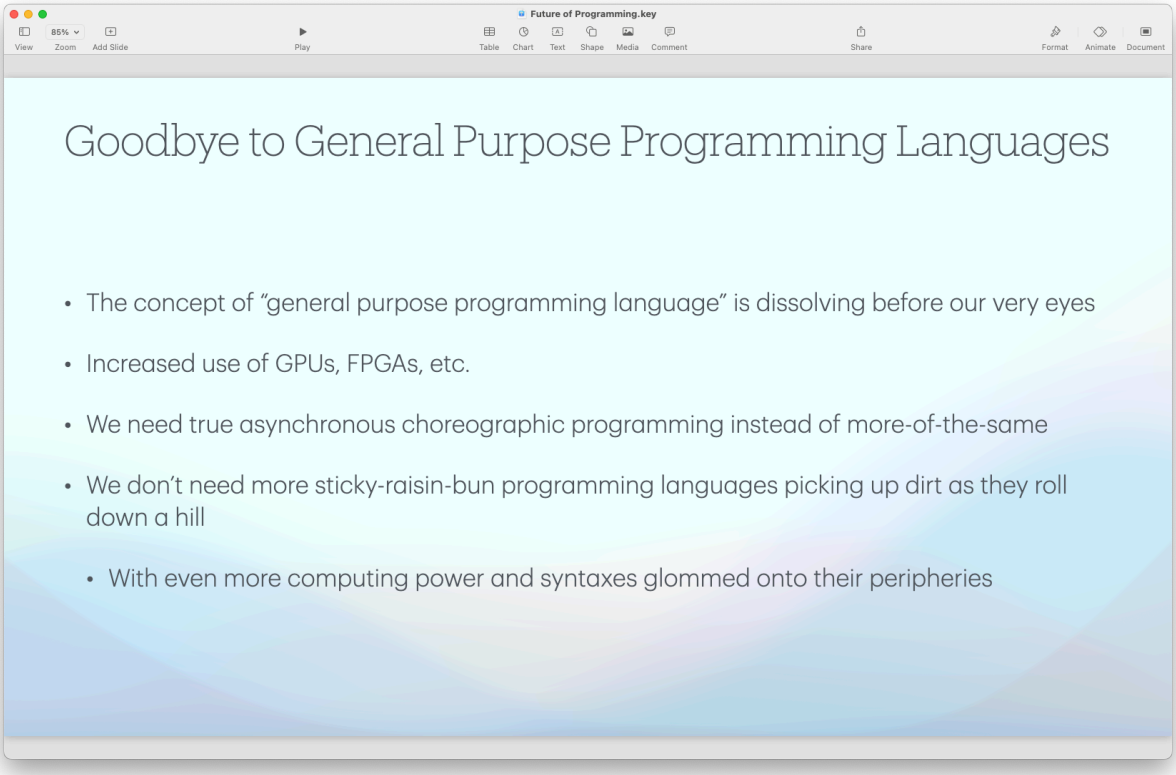


Stream Parsing - Part of the Solution?

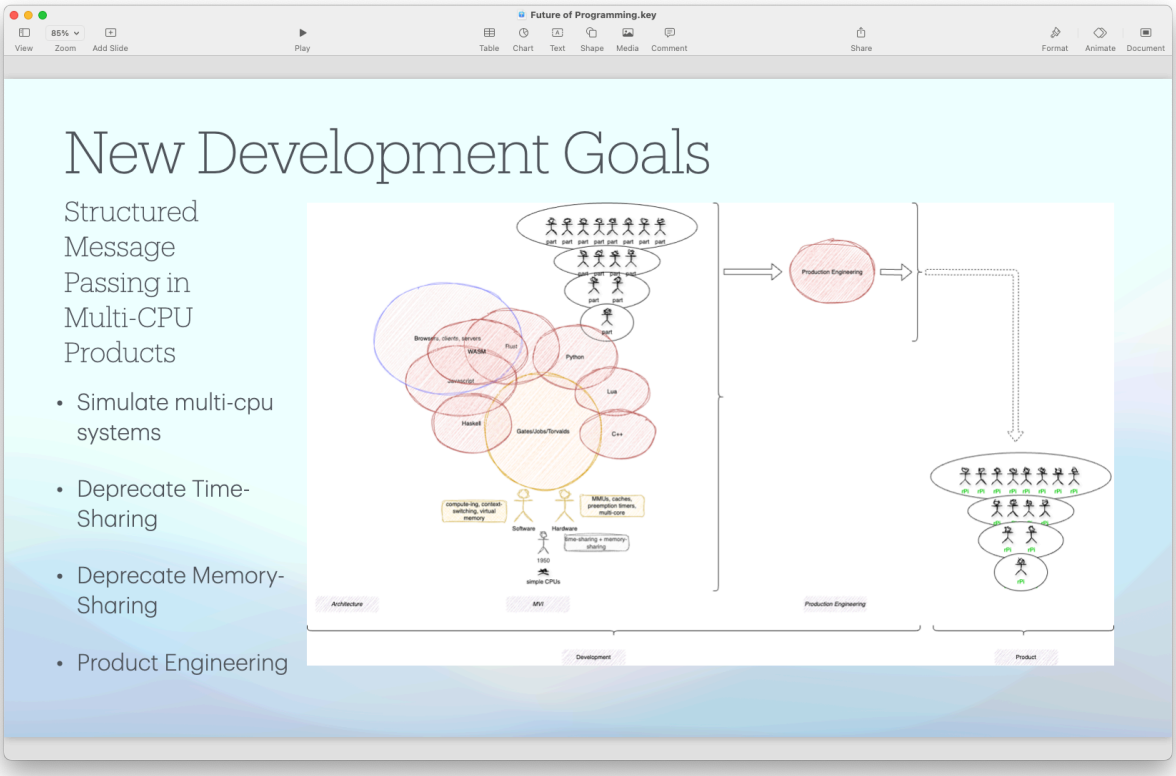


S/SL[1], PT Pascal[2]

Goodbye to General Purpose Programming Languages



New Development Goals

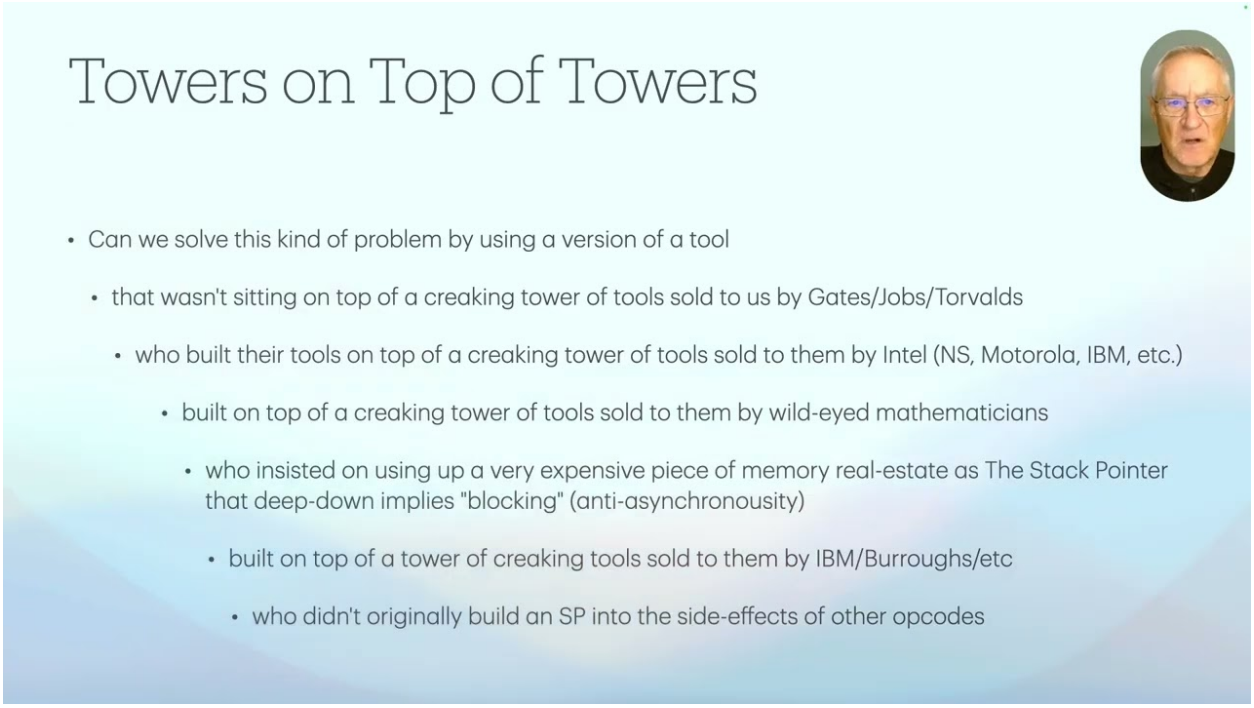


Bibliography

[1] S/SL (Syntax Semantic Language) from https://research.cs.queensu.ca/home/cordy/pub/downloads/ssl/
[2] PT Pascal from https://research.cs.queensu.ca/home/cordy/pub/downloads/ssl/

See Also

Same set of slides on YouTube:



References: https://guitarvydas.github.io/2024/01/06/References.html

Blog: guitarvydas.github.io

Videos: https://www.youtube.com/@programmingsimplicity2980

Discord: https://discord.gg/65YZUh6Jpq

Leanpub: [WIP] https://leanpub.com/u/paul-tarvydas

Gumroad: tarvydas.gumroad.com

Twitter: @paul_tarvydas