

A large satellite or space station module is shown in orbit against the black background of space. The satellite has a cylindrical body with various instruments and antennas. The words "UNITED STATES" are visible on its side. The Earth's surface is partially visible in the lower right corner.

Spaceships, Radiation Therapy, and Power Users

Blastoff Rails 2026

Nikky Southerland

Philadelphia via Seattle

Red or Green? Christmas

Lead Engineer @ Vehlo

Technology Director @
McSweeneys

Formerly Rick Steves' Europe



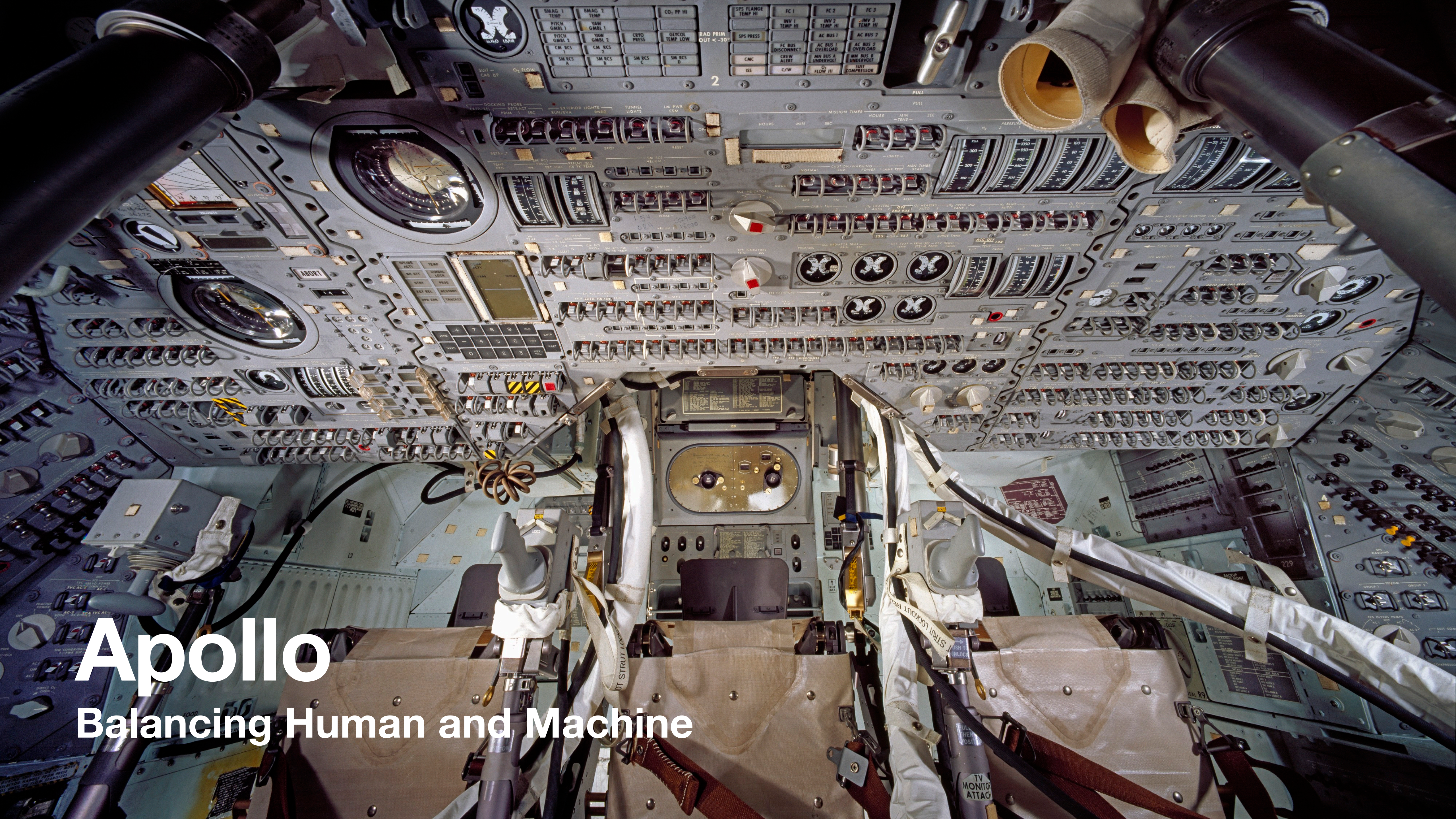
Our Journey Today

Humans Learning to Trust Computers

Humans Learning Computers May Want to Kill Us

Lessons for a Modern World

**Our Main Job: Designing for
Humans**



Apollo

Balancing Human and Machine

1962

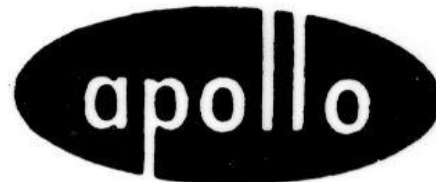
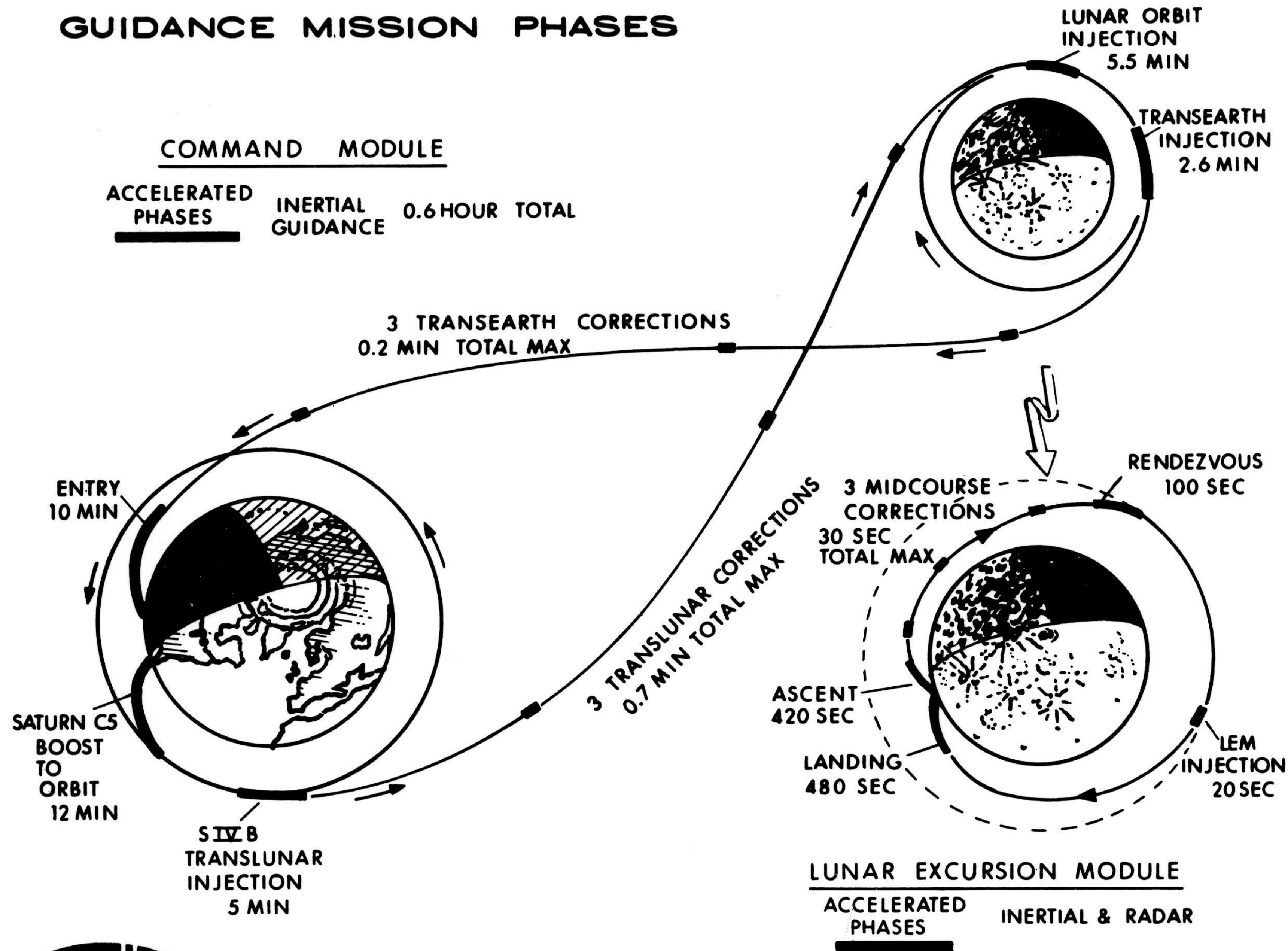
Destination: Moon



**We Weren't Sure
How To Do This**

(At all)

GUIDANCE MISSION PHASES



Navigational Challenges

- Orbital rendezvous (multiple times!)
- Maneuvering and orienting in a non-Earth gravity environment
- Rocket-assisted landing on the atmosphere-less moon



U.S. AIR FORCE
25016

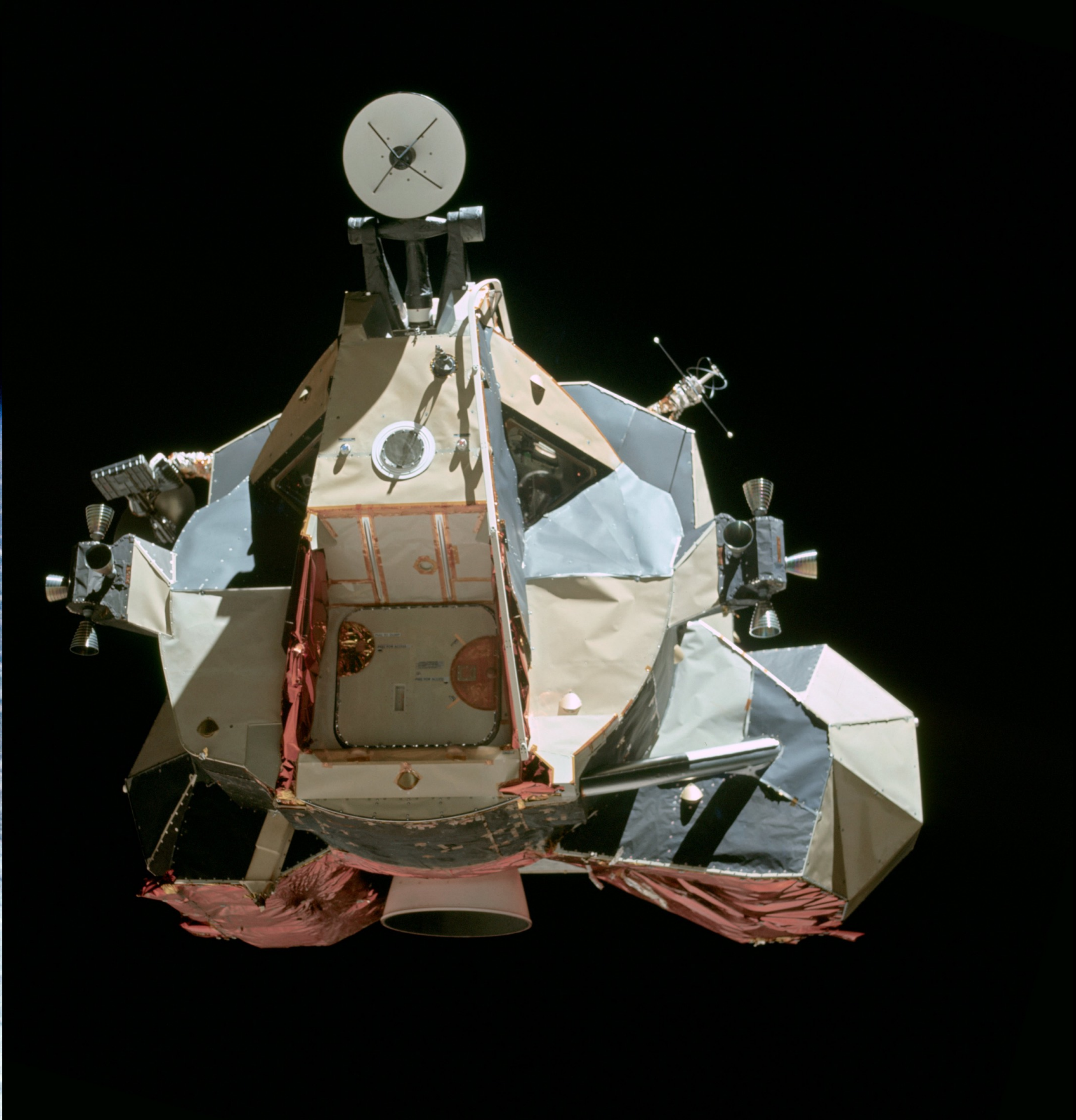
FU-016



North American

TF-86

TRANSONIC TRAINER



Who's in charge?

Humans?

Computers?

Apollo Guidance Computer



UPLINK
ACTY

TEMP

AUTO

GIMBAL
LOCK

HOLD

PROG

FREE

RESTART

NO ATT

TRACKER

STBY

KEY REL

OPR ERR

COMP
ACTY

PROG

VERB

NOUN

VERB

+

7

8

9

CLR

ENTR

-

4

5

6

STBY

RSET

NOUN

0

1

2

3

KEY
REL

Hybrid Approach

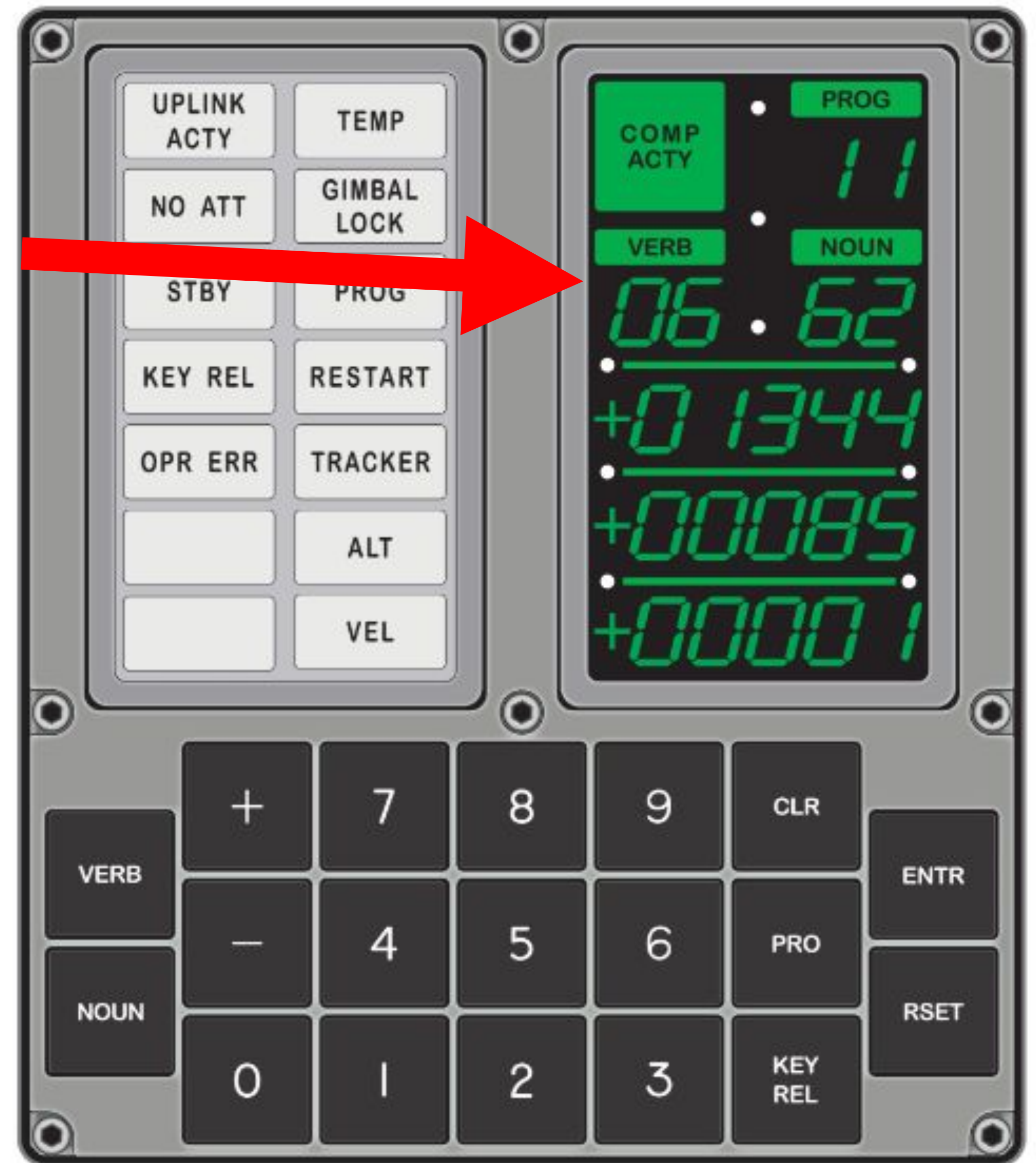
- Humans kept in the loop through displays and programs
- Joystick input runs through computer (early fly by wire!)
- Major commands only execute when instructed

Using the Computer

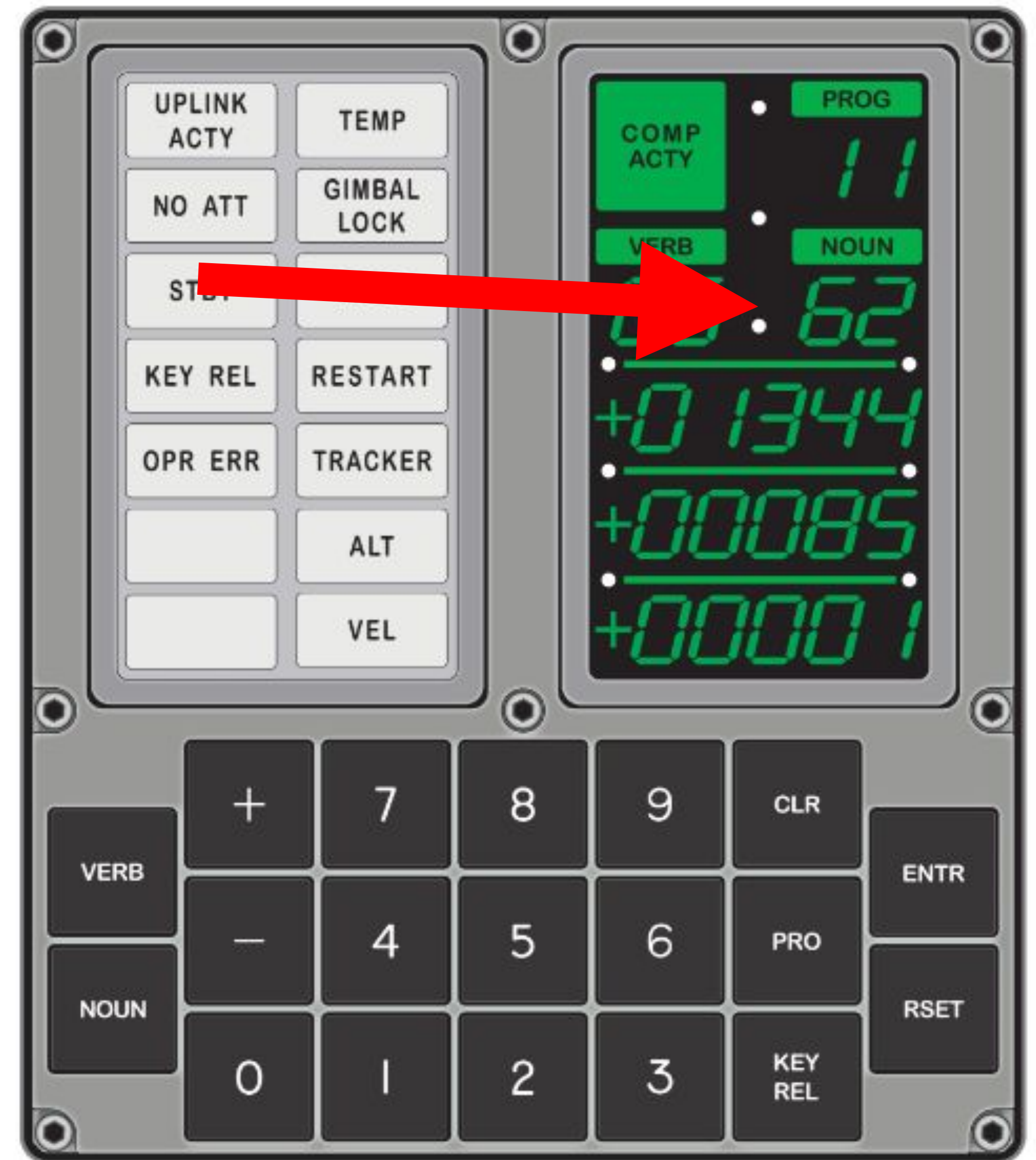
Program (App!)



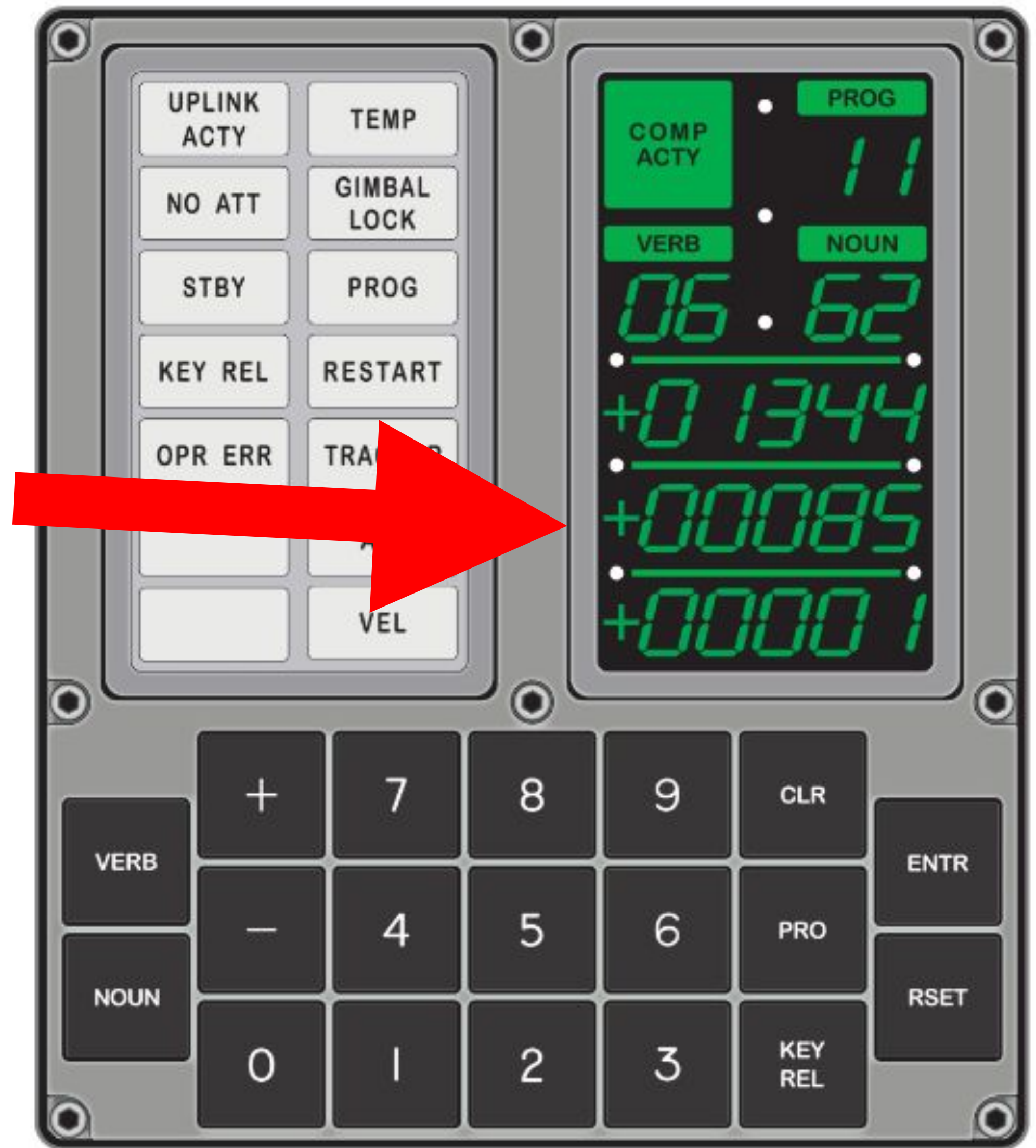
Verb (Function!)



Noun (Variable!)



Display (Output!)



Program: 15 (TLI Burn)
Verb: 16 (Monitor Data)
Noun: 68 (Δ H)

Or in Rails, `GET /tli/monitor?v=delta_h`

How did this interface come about?

They needed something for a demo

What did the astronauts think?

“...of course we’ll shut off the computer as soon as we’re up there...”

Anonymous Apollo Program Astronaut

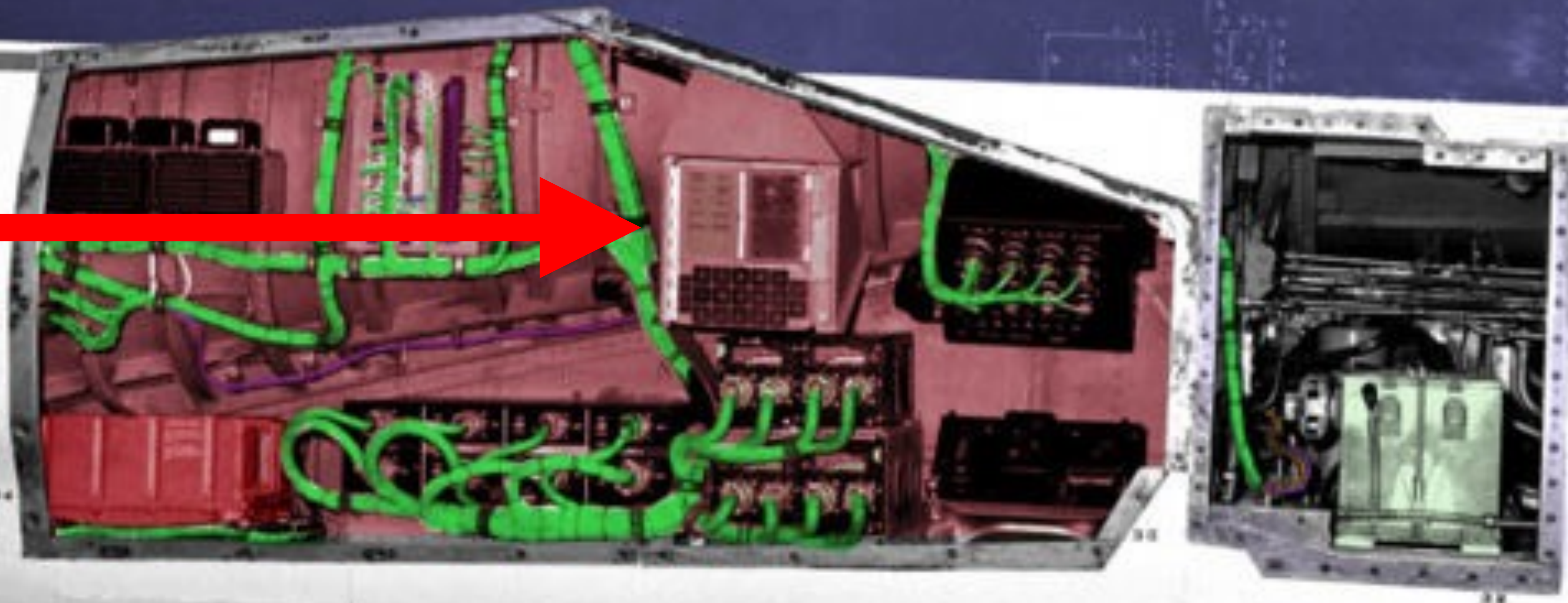
**“more purely administrative, or
supervisory” way of flying**

NASA Report

Apollo Lessons Learned

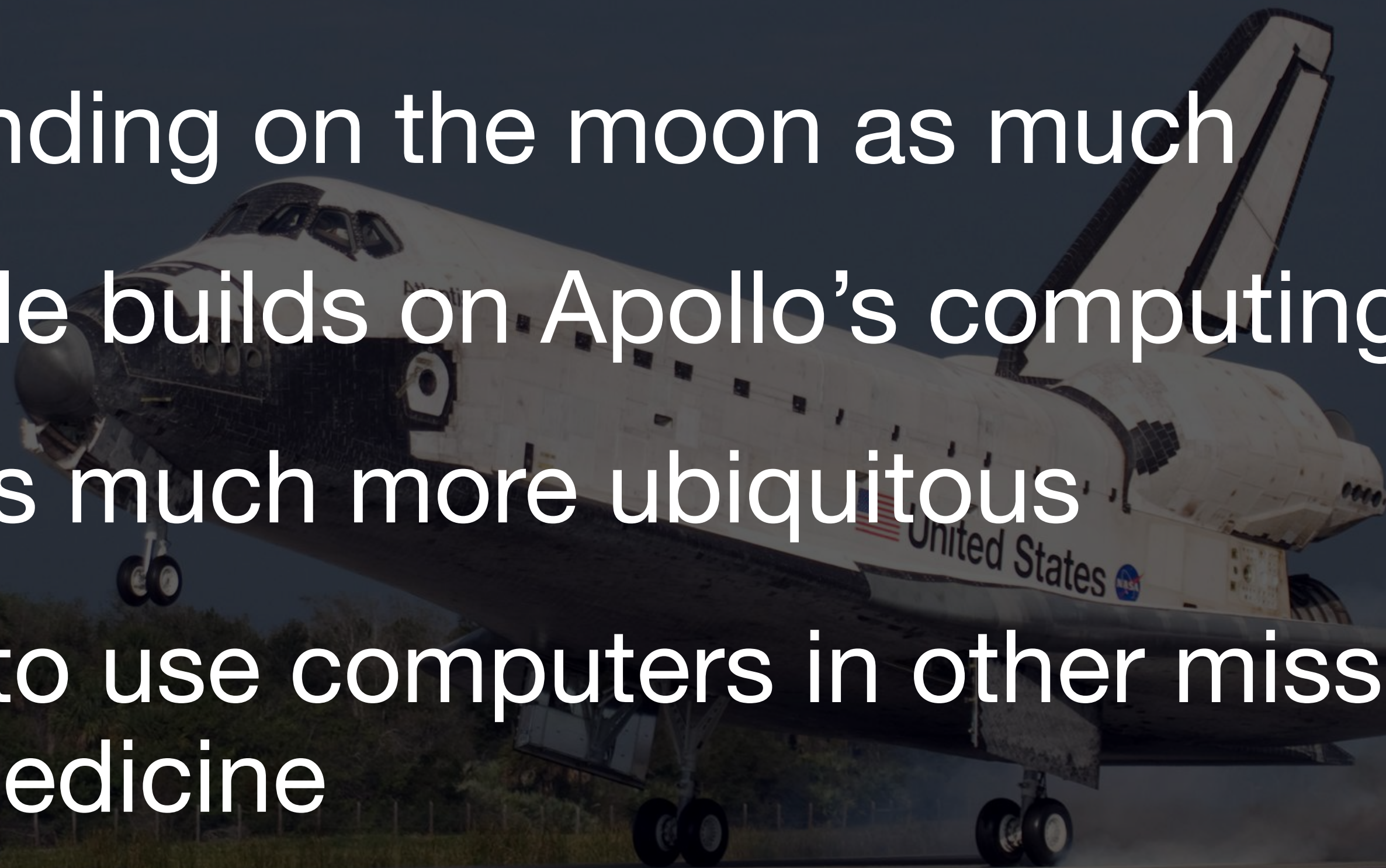
- Robust logic and defensive programming won over reluctant users
- Validated approach of having highly skilled users largely commanding computers

DIGITAL FLY-BY-WIRE



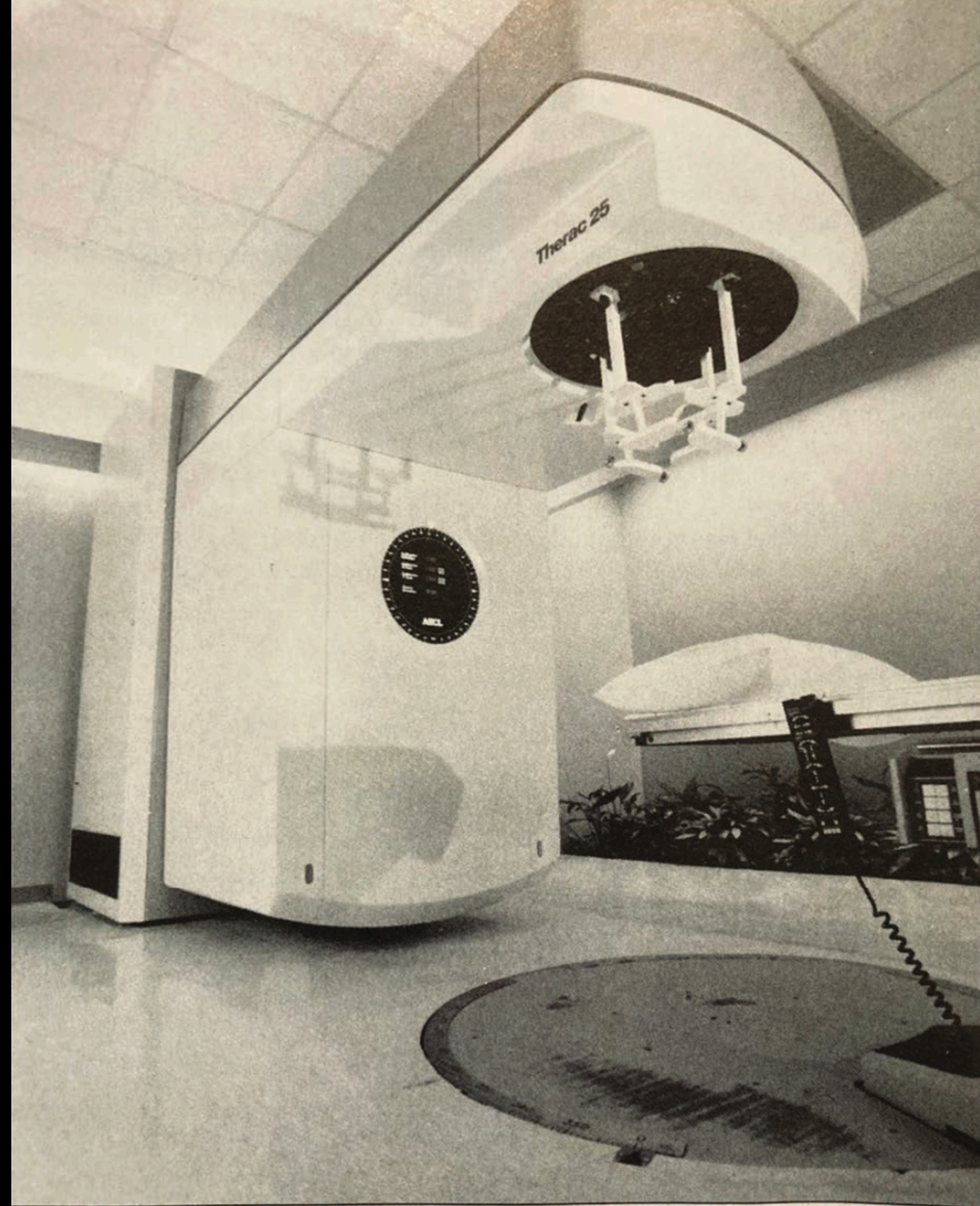
Skip Forward To The 1980s

- We're not landing on the moon as much
- Space Shuttle builds on Apollo's computing successes
- Computing is much more ubiquitous
- Confidence to use computers in other mission critical areas, like medicine



Therac-25

When Computers Try to Kill Us



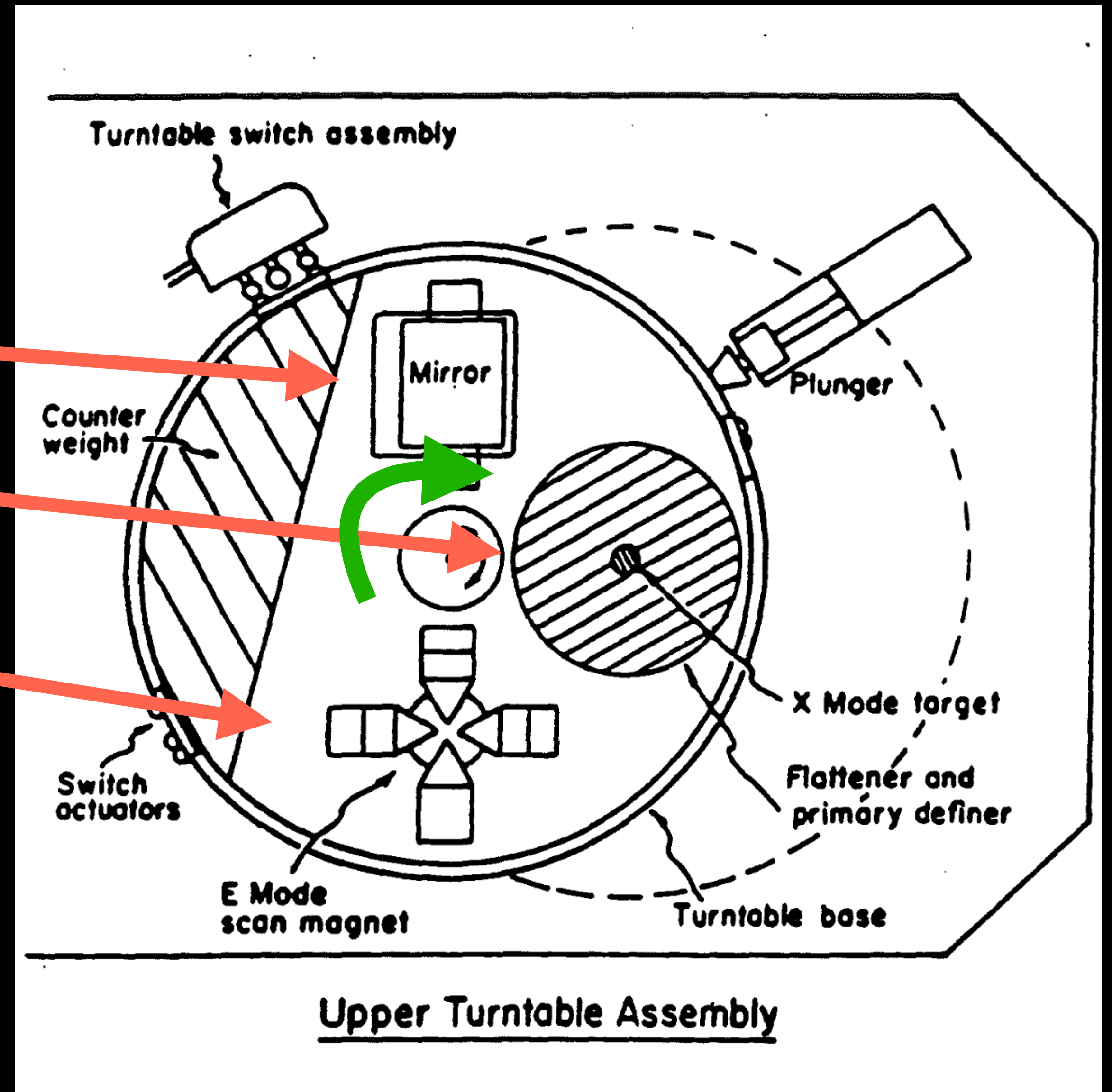
Therac Basics

- Radiation oncology device by AECL
- Development starts in mid-1970s
- Rolled out to clinics in 1982
- Based on earlier devices
- Software by a single developer



Safety Interlock

1. Mirror (visual mode)
2. Photon/X-Ray
3. Electron



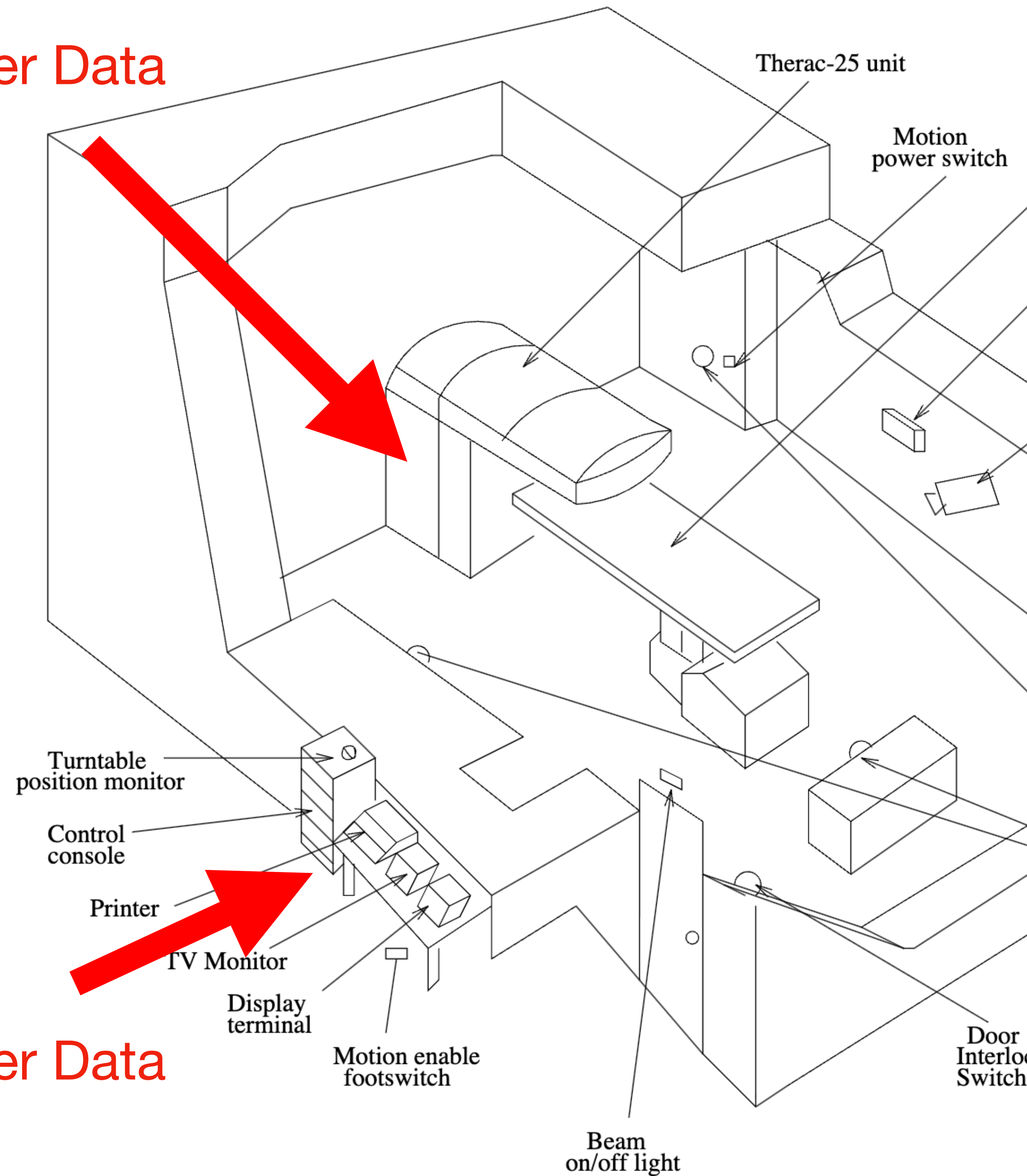
First Therac with Software Interlocks

Treatment Process

1. Operator places patient, sets treatment info on the machine
2. Operator walks to console, and enters the same info on console
3. If they match, treatment progresses

Enter Data

Enter Data

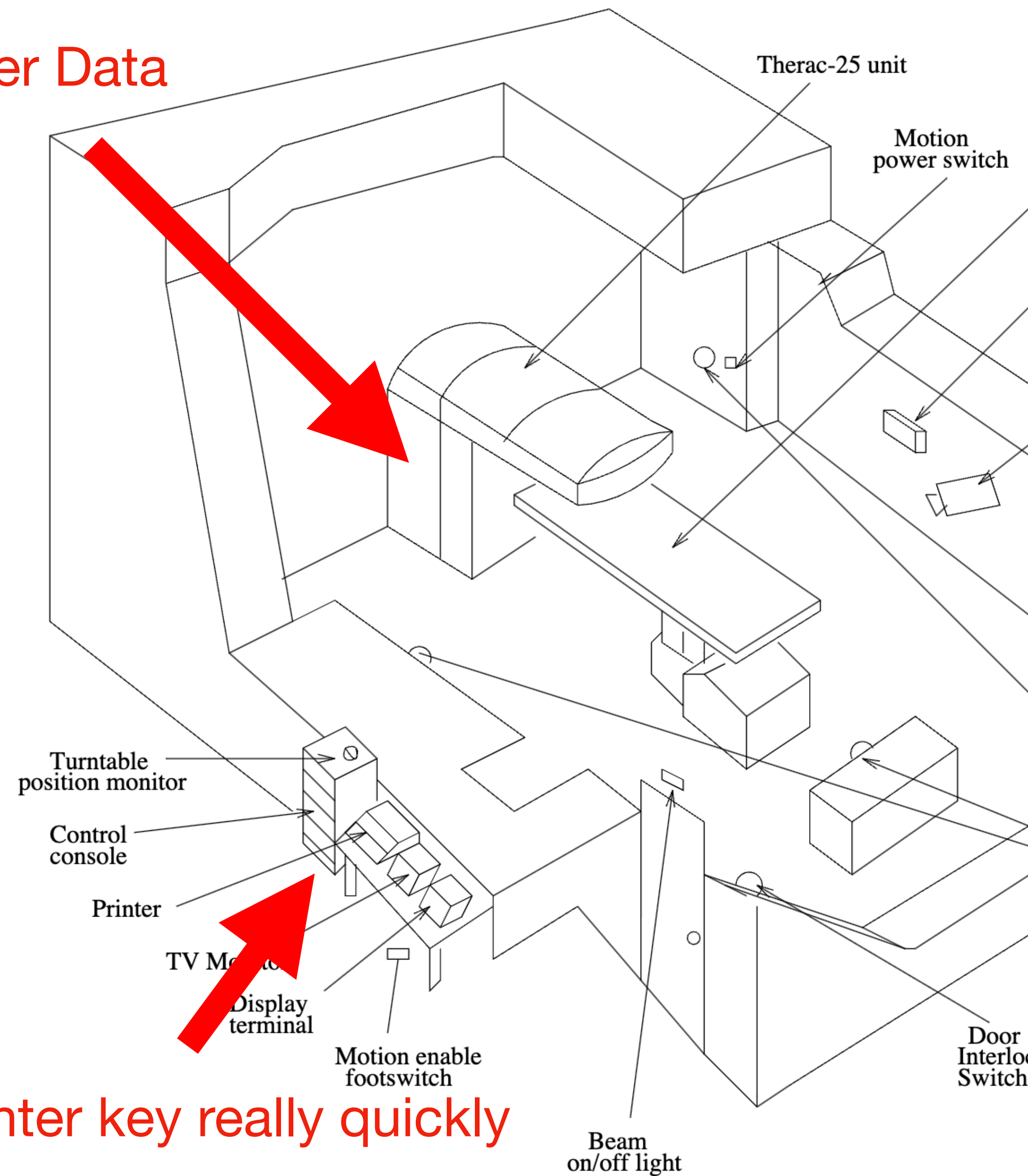


Users Complained

Modified Process

1. Operator enters information into machine
2. Operator can now just hit “enter” to copy from treatment site data
3. Treatment begins

Enter Data



Hit enter key really quickly

Errors

- Per AECL, machine cannot cause overdose
- 40+ malfunctions a day
- Operators learned to “proceed” on errors
- “Flatness”

Patient Harm

- June 1985: 15,000-20,000 rad in shoulder
- July 1985: Machine kept failing. Operator kept overriding. 17,000 rads
- December 1985: Skin burns in machine-like striped pattern
- March 1986: 16,000-25,000 rads in <1 second to 1cm square area; died 5 months later
- April 1986: Died three weeks later from overdose; visible brain stem damage

Troubleshooting

Problem Timeline

With a Power User

1. Tech enters in treatment params, hits “enter” quickly to confirm
2. Turntable begins to move to treatment mode. Starts 8 second internal timer
3. Tech realizes they entered the wrong treatment type (one that doesn’t need restriction)
4. Tech hits “Up” key to change, and then confirms within the 8 seconds. Mode shows as changed but the interlock is back at the original position
5. Hits “Dose” key; overdose occurs (updated dose + original interlock mode)

“To remove the accidental use, remove the cap and fix the [up key] contacts open with electrical tape”

AECL

One Fix Down; Many To Go

Therac-25 Lessons Learned

- Perhaps don't always give users what they're asking for
- Proficient operators can cause issues unanticipated during development
- Document all errors and make them as useful as possible
- Believe users

But really...

Who cares about ancient computers

Lessons We Can Use



Accessibility!

- Accessibility is for users of all ability levels, including the very skilled
- Bonus: well-structured content is LLM food

Prefer Keyboard Input

- Reliable
- Muscle Memory
- Efficient
- Caveat: change with caution!

Robust Validations Backstop Everything

- Good: Backend Validations (Rails- and Database-backed)
 - Rails Model Validations for Easy Error Handling
 - Database-backed for key areas
- Bad: No Validations
- Ugly: Frontend-only Validations

Unpredictable Behavior Kills

- It's easy to distrust computers
- It's very hard to learn to trust them again
- Don't surprise your users

It's Okay To Error!

- Good! Well-handled errors with descriptive error text
- Bad: Unhandled exceptions
- Ugly: Rescuing all exceptions and doing nothing with it

**Hide Unnecessary Complexity
(but reveal on demand)**

Keep Meticulous Logs

- Classic, but effective
- Cheap to help troubleshooting
- Spend the time to format and make searching fun
- Gems like Lograge that support key/value pairs are great

```
Jun 08 23:51:28 web-prod bash[872272]: [e0760206-3afd-4d49-b76d-a04df2640240] method=GET path=/articles/i-sisyphus-am-ninet
y-five-percent-of-the-way-there format=*/* controller=ArticlesController action=show status=200 allocations=6736 duration=4
8.00 view=37.10 db=7.13
Jun 08 23:51:29 web-prod bash[872272]: [9641211b-efdb-4c59-8b16-df7b8778fdc6] method=GET path=/articles/page-six-gets-extre
mely-personal format=html controller=ArticlesController action=show status=200 allocations=12150 duration=71.44 view=41.26
db=14.92
Jun 08 23:51:29 web-prod bash[872272]: [959e3a0a-6731-47be-9330-fb71e34126d8] method=GET path=/columns/open-letters-to-peop
le-or-entities-who-are-unlikely-to-respond format=*/* controller=ColumnsController action=show status=200 allocations=9405
duration=36.32 view=27.73 db=6.56
Jun 08 23:51:29 web-prod bash[872272]: [889f4aa4-a5b1-4b1c-9c87-18af3c8a0fcc] method=GET path=/articles/the-four-horsemen-o
f-gentrification format=html controller=ArticlesController action=show status=200 allocations=12906 duration=49.96 view=26.
83 db=20.58
```

Users Use Better Than You

It's true!

- Especially in the B2B and SAAS world, users are much faster
- They may spend 40+ hours a *week* in your interfaces
- Build for them
- Seconds matter
- If it's robust for power users, it's robust for everyone

Thanks + Questions

Resources! Links! Cool video!



<https://powerusers.nikkysoutherland.com>