

Beyond Bitcoin & FORTH (BBF004)

H3A Hilbertian Hashed Homoiconic Augmentation

Combining Phoscript Program Syntactic Simplification and Omnihash produces H3A Hilbertian Hashed Homoiconic Augmentation, whose ultimate goal is to create a "super google earth", where realtime user updates will be used to train AGI Artificial General Intelligence, whose code base is in turn being enhanced and modified by H3A Hilbertian Hashed Homoiconic Augmentation, fulfilling its homoiconic namesake by firstly producing a realtime user updated earth model and secondly modifying the code base of AGI Artificial General Intelligence in real time.

Liang Ng, May 2026

omnixtar.github.io/svfig

<https://www.linkedin.com/in/liang-ng>



Omni*Web + Omni*DOC



SVFIG (Silicon Valley FORTH Interest Group)

- [April 25 2026, BBF003 FORTH everywhere – even in AI llama.cpp](#)
- [February 28 2026, BBF002 AGI Self-Identity Using Hash of Public Key & Reselling AI Server Tokens w. Omnihash](#)
 - [Grok summary](#)
- [January 24 2026, BBF001 Hilbert Hotel with Infinite Cabinets](#)
 - [January 24 2026, I3P: <iframe> + I2P Invisible Internet Project](#)
- [November 15 2025, The roles of FORTH in blockchain technologies and beyond \(Summary & Updates\)](#)
- [October 25 2025, The roles of FORTH in blockchain technologies and beyond](#)
- [August 2025, Crypto-Metaprogramming: alternative to Model-View-Controller Architecture](#)
- [April 26 2025, How to share my latest FORTHish word with the world? DJSON + Phoscript = Omni*Web](#)

Beyond Bitcoin & FORTH

Bitcoin Address: Hash of Public Key

Omnihash User ID: Hash of
Public Key

Phoscript: extension of FORTH
on host programming languages

Omni*Web: ecosystem built with
Omnihash & Phoscript

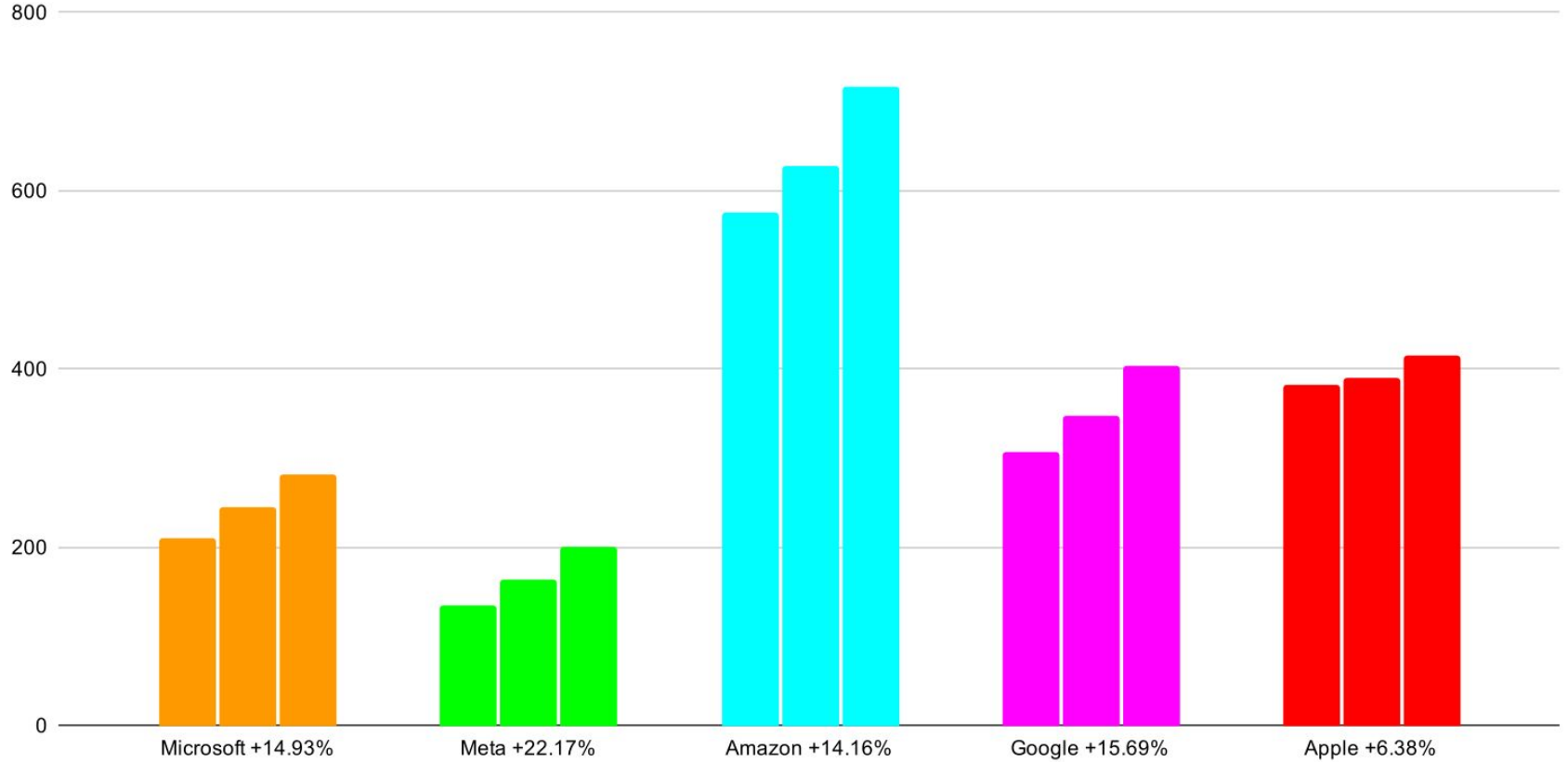
Omni: everything
*: everything

This series of presentations is called “Beyond Bitcoin & FORTH” as we explore crucial features of Bitcoin such as using the hash of public key as user identifier, and how FORTH has been deployed in Bitcoin;

- A. the hash of public key has been extended to Omnihash, a term to describe hashcode data structures for representing ANY kind of digital assets, including their ownerships, hence the prefix “Omni”.
- B. our variant of FORTH, Phoscript, has been deployed in various environments, including mobile and web front ends, PHP back end and Python Selenium web control modules.
- extensions of Omnihash and Phoscript will be explored.

MMAGA Revenues 2025/24/23 (USD billions)

Total 2025: USD 2.018 Trillion (+13.6%)



H3A: THE UNIFYING SYSTEM FOR GLOBAL EARTH SIMULATION AND AGI TRAINING

PHOSCRIPT

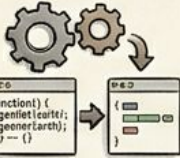
PROGRAM SYNTACTIC SIMPLIFICATION (PSS)

PSS converts users and programmers free time into very simple but powerful Phoscript code.

INPUT: USERS & PROGRAMMERS FREE TIME

PROCESS: PSS

OUTPUT: SIMPLE & POWERFUL PHOSCRIPT CODE



```
function() {
  scene: and(middle);
  com: liege(earth);
  vorp: at(earth);
  geom: (code);
  geom: code(each);
  ..? -- 4)
}
```

PSS converts users and programmers free time into very simple but powerful Phoscript code.

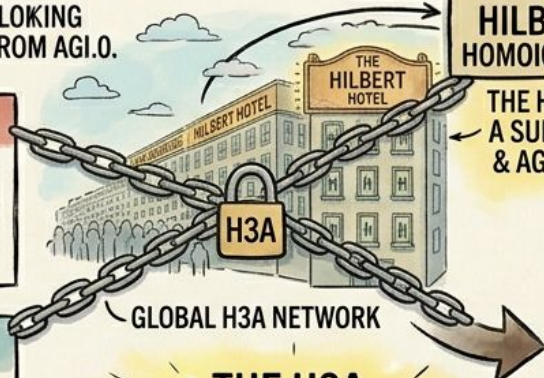
PSS AND UNBLUCSH ANBLOKING TWO TRUE ROADBLOCKS FROM AGI.0.

ROADBLOCK 1: OWNERSHIP

UNLOCKED

ROADBLOCK 2: SIMPLIFICATION

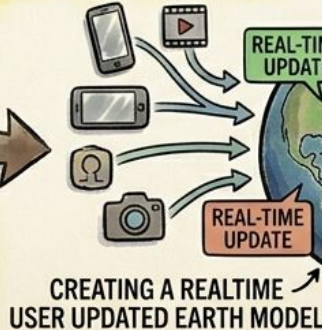
SOLVED



THE H3A ULTIMATE GOAL: A SUPER GOOGLE EARTH & AGI TRAINING ENGINE

H3A: HILBERTIAN HASHED HOMOICONIC AUGMENTATION

THE H3A ULTIMATE GOAL: A SUPER GOOGLE EARTH & AGI TRAINING ENGINE



OMNIHASH: VERIFIABLE DIGITAL CONTRACTS & OWNERSHIP

INPUT: PHOSCRIPT CODE

PROCESS: OMNIHASH

OUTPUT: VERIFIABLE DIGITAL CONTRACTS & ASSET OWNERSHIP

```
function() {
  scene: and(middle);
  com: liege(earth);
  vorp: at(earth);
  geom: (code);
  geom: code(each);
  ..? -- 4)
}
```



Omnihash converts Phoscript code into verifiable digital contracts, monetising users' digital assets and establishing the ownerships of digital assets.

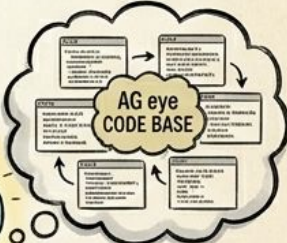
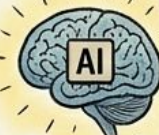


Essential for worldwide coordination and collaboration motivating participation.

3. H3A ENHANCEMENT

HOMOICONIC MODIFICATION

H3A ENHANCEMENT & HOMOICONIC MODIFICATION



AG eye AGI

(Artificial General Intelligence)

2. TRAINING AG EYE AGI

Realtime user updates are used to train AG eye AGI.

H3A REPEATING CYCLE

H3A: THE UNIFYING SYSTEM FOR GLOBAL EARTH SIMULATION AND AGI TRAINING

PHOSCRIPT

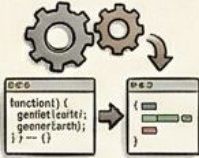
PROGRAM SYNTACTIC SIMPLIFICATION (PSS)

PSS converts users and programmers free time into very simple but powerful Phoscript code.

INPUT: USERS & PROGRAMMERS FREE TIME

PROCESS: PSS

OUTPUT: SIMPLE & POWERFUL PHOSCRIPT CODE



```
function() {  
  genier(middle);  
  connelegartrh);  
  vorpaysitlarth);  
  genier( code);  
  genier(codn(each);  
  ..?) -- 4)  
}
```

PHOSCRIPT CODE

PSS converts users and programmers free time into very simple but powerful Phoscript code.

PSS AND UNBLUCSH ANBLOKING TWO TRUE ROADBLOCKS FROM AGI.O.

OMNIHASH: VERIFIABLE DIGITAL CONTRACTS & OWNERSHIP

INPUT: PHOSCRIPT CODE

PROCESS: OMNIHASH

OUTPUT: VERIFIABLE DIGITAL CONTRACTS & ASSET OWNERSHIP



HASH

Omnihash converts Phoscript code into verifiable digital contracts, monetising users' digital assets and establishing the ownerships of digital assets.



Essential for worldwide coordination and collaboration motivating participation.

H3A:

HILBERTIAN HASHED HOMOICONIC AUGMENTATION

3. H3A ENHANCEMENT

HOMOICONIC MODIFICATION

The title of this presentation does not seem to involve Artificial Intelligence but it ultimately does. **Hilbertian**: to David Hilbert's Grand Hotel paradox which he introduced the concept of infinity and the unconventional outcome of arithmetic operations on infinity. **Hashed**: the use of hash numbers as room number of Hilbert Hotel, being a self generated mechanism to produce a label to represent the ownership of digital assets.

H3A: THE UNIFYING SYSTEM FOR GLOBAL EARTH SIMULATION AND AGI TRAINING

PHOSCRIPT

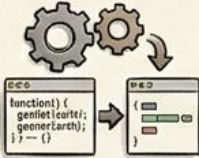
PROGRAM SYNTACTIC SIMPLIFICATION (PSS)

PSS converts users and programmers free time into very simple but powerful Phoscript code.

INPUT: USERS & PROGRAMMERS FREE TIME

PROCESS: PSS

OUTPUT: SIMPLE & POWERFUL PHOSCRIPT CODE



```
function() {
  scene: and(middle);
  coin: fige(gerth);
  vorpysat(gerth);
  geomet: (code);
  gener: oth(each);
  ..?? -- 4)
}
```

PHOSCRIPT CODE

OMNIHASH: VERIFIABLE DIGITAL CONTRACTS & OWNERSHIP

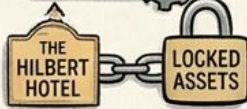
INPUT: PHOSCRIPT CODE

PROCESS: OMNIHASH

OUTPUT: VERIFIABLE DIGITAL CONTRACTS & ASSET OWNERSHIP



Omnihash converts Phoscript code into verifiable digital contracts, monetising users' digital assets and establishing the ownerships of digital assets.



Essential for worldwide coordination and collaboration motivating participation.

Homoiconic: using FORTH / Phoscript to simplify existing source code of free software projects, so that more people can participate – very important in AI era where redundancies worsen.

Augmentation: +1 – make one change, in code or data.

H3A: HILBERTIAN HASHED HOMOICONIC AUGMENTATION

THE H3A ULTIMATE GOAL: A SUPER GOOGLE EARTH & AGI TRAINING ENGINE

3. H3A ENHANCEMENT

H3A ENHANCEMENT & HOMOICONIC MODIFICATION

HOMOICONIC MODIFICATION



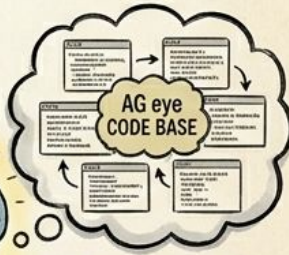
REAL-TIME UPDATE

UPDATED

UPDATED

REAL-TIME UPDATE

CREATING A REALTIME USER UPDATED EARTH MODEL



AG eye AGI

(Artificial General Intelligence)

2. TRAINING AG EYE AGI

Realtime user updates are used to train AG eye AGI.

DATA FOR TRAINING

H3A REPEATING CYCLE

H3A: THE UNIFYING SYSTEM FOR GLOBAL EARTH SIMULATION AND AGI TRAINING

PHOSCRIPT

PROGRAM SYNTACTIC SIMPLIFICATION (PSS)

PSS converts users and programmers free time into very simple but powerful Phoscript code.

INPUT: USERS & PROGRAMMERS

PROCESS: PSS

OUTPUT: SIMPLE & POWERFUL PT CODE

Just some tips for fellow programmers who may want to add some bells and whistles to your presentation materials:

1. Goggle Gemini (free) is quite good at generating comic / information drawing such as this.

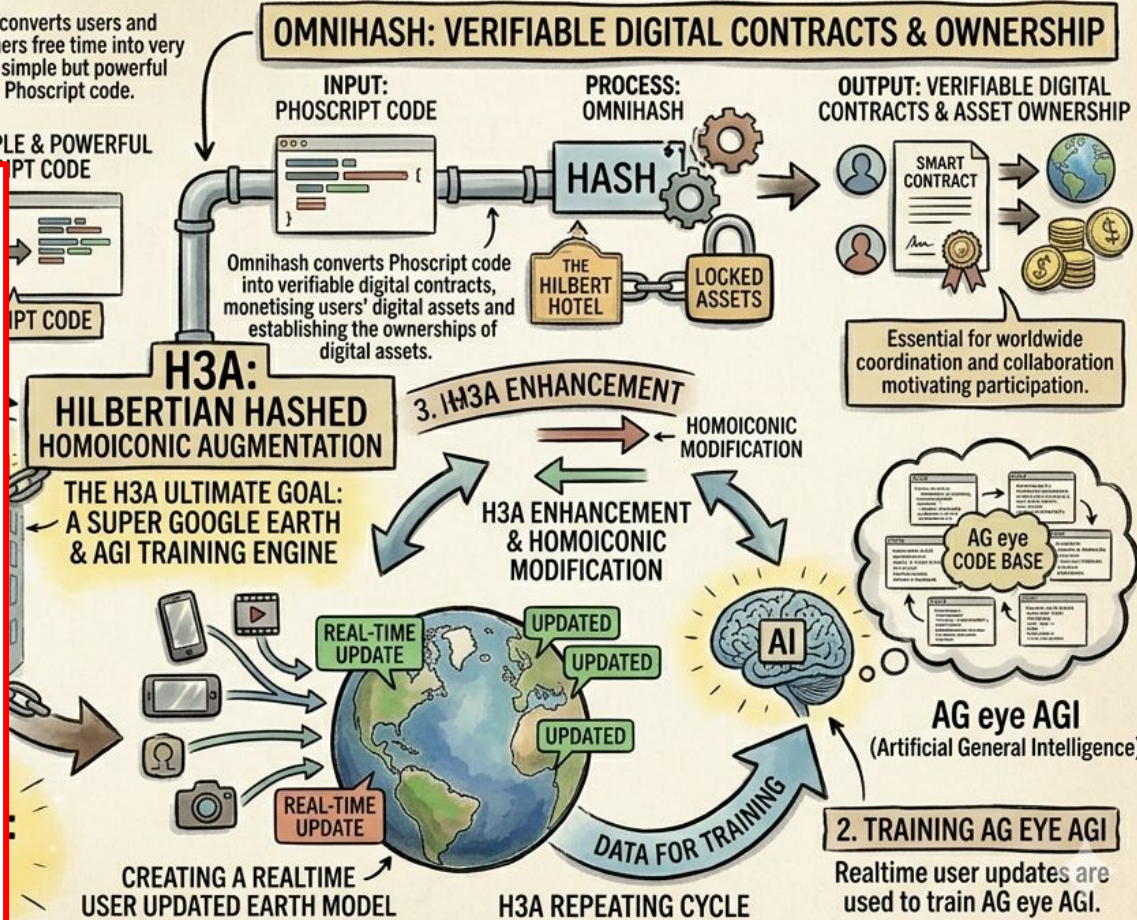
2. Many AI apps can generate SVG drawings, which you may edit using InkScape.

OMNIHASH: VERIFIABLE DIGITAL CONTRACTS & OWNERSHIP

INPUT: PHOSCRIPT CODE

PROCESS: OMNIHASH

OUTPUT: VERIFIABLE DIGITAL CONTRACTS & ASSET OWNERSHIP



H3A:

HILBERTIAN HASHED HOMOICONIC AUGMENTATION

THE H3A ULTIMATE GOAL:
A SUPER GOOGLE EARTH
& AGI TRAINING ENGINE

digital assets.

3. H3A ENHANCEMENT

HOMOICONIC
MODIFICATION

H3A ENHANCEMENT
& HOMOICONIC
MODIFICATION

Essential for worldwide
coordination and collaboration
motivating participation.



REAL-TIME
UPDATE

UPDATED

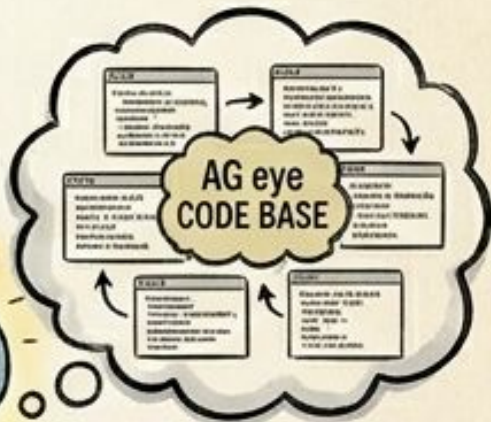
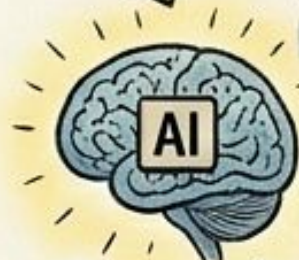
UPDATED

UPDATED

REAL-TIME
UPDATE

CREATING A REALTIME
USER UPDATED EARTH MODEL

H3A REPEATING CYCLE



AG eye
CODE BASE

AG eye AGI
(Artificial General Intelligence)

2. TRAINING AG EYE AGI

Realtime user updates are
used to train AG eye AGI.

DATA FOR TRAINING

- create a "super google earth", where realtime user updates will be used to train AGI Artificial General Intelligence,
- whose code base is in turn being enhanced and modified by LLASMA Large Language + Stack Machine Model

Essential for worldwide information and collaboration requiring participating participation.

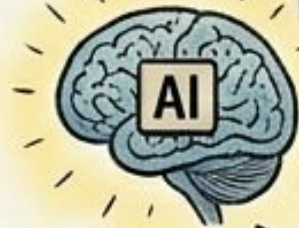
& AGI TRAINING ENGINE

& HOMOICOMIC MODIFICATION



CREATING A REALTIME USER UPDATED EARTH MODEL

H3A REPEATING CYCLE



AG eye AGI (Artificial General Intelligence)

2. TRAINING AG EYE AGI

Realtime user updates are used to train AG eye AGI.

Combining Phoscript Program Syntactic Simplification and Omnihash produces H3A Hilbertian Hashed Homoiconic Augmentation, whose ultimate goal is to create a "super google earth", where realtime user updates will be used to train AGI Artificial General Intelligence, whose code base is in turn being enhanced and modified by H3A Hilbertian Hashed Homoiconic Augmentation, fulfilling its homoiconic namesake by firstly producing a realtime user updated earth model and secondly modifying the code base of AGI Artificial General Intelligence in real time.

Stages in AI LLASMA (Large Language + Stack Machine Architecture) development:

- Multilayer structure (information) programming:
- from symbols to semantics
- “reasoning”
- self vs. others, gain vs. loss, etc.

Advantage: FORTH/Phoscript is a complete debugger + software development system from level 0 to infinity (Recursive Hilbert Hotel RHH) that can be embedded anywhere.

Critical Point of “Awakening”:

- **When llama.cpp knows how to use FORTH/Phoscript (compose and test and iterate)**

Measurable metrics towards AGI: FORTH / Phoscript WORDS as units !! (World’s First??)

- From now to Awakening: FORTH / Phoscript exercise, one word at a time.
- How many words needed? FORTH experts please guess. 1000 words?
- Measure software complexity in any programming language.

Reduce AGI to Awakening, measurable steps.

Target: 3 months? Or first estimate of ETA in 3 months?

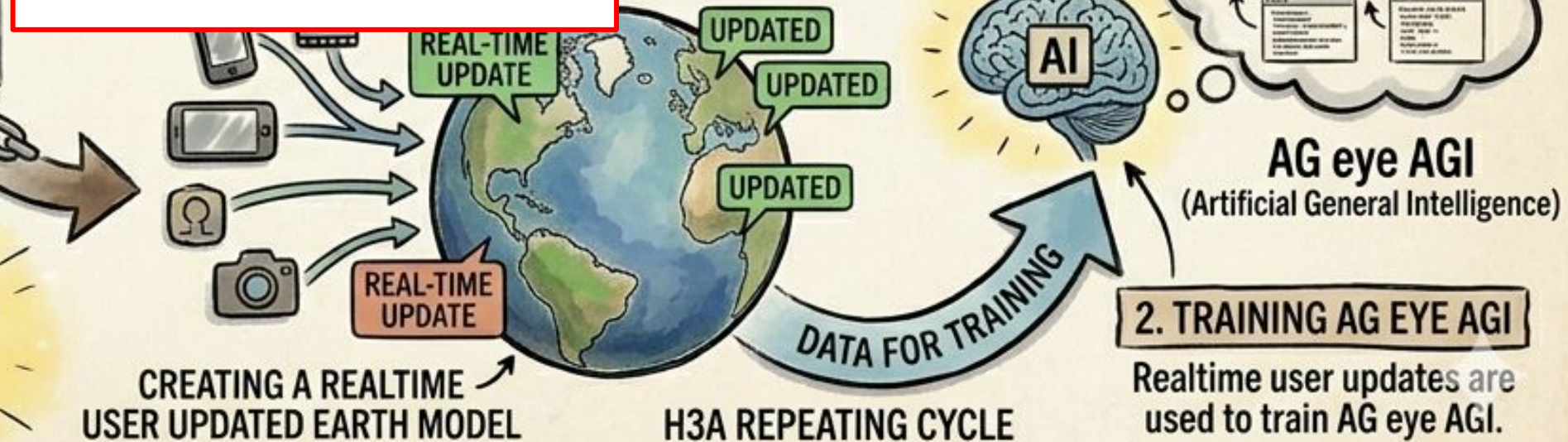
Omnihash on LLASMA words: tractable, as reference to any object in RHH
→ White Box model for AI !! (World’s First?)

Augmentation +1

- Cesium + LLASMA (Large Language + Stack Machine)

Aims of H3A:

- Produce AGI (Cesium + LLASMA) using massive collaboration, literally 5 minutes a day, one WORD at a time.



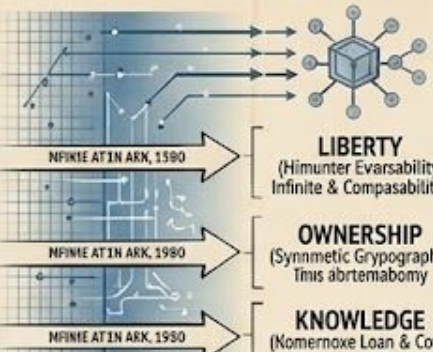
LOK PROTOCOL MANIFESTO & THE SPIRIT OF 1789: AN ARCHITECTURAL SYMMETRY



LIBERTÉ
(Friction hurs symms,
Infinite init spexa)

ÉGALITÉ
Fratennit's law,
tbrait-of amiterrate fine)

FRATERNITÉ
(The shared side hode,
rensmxed untxaxt ad)



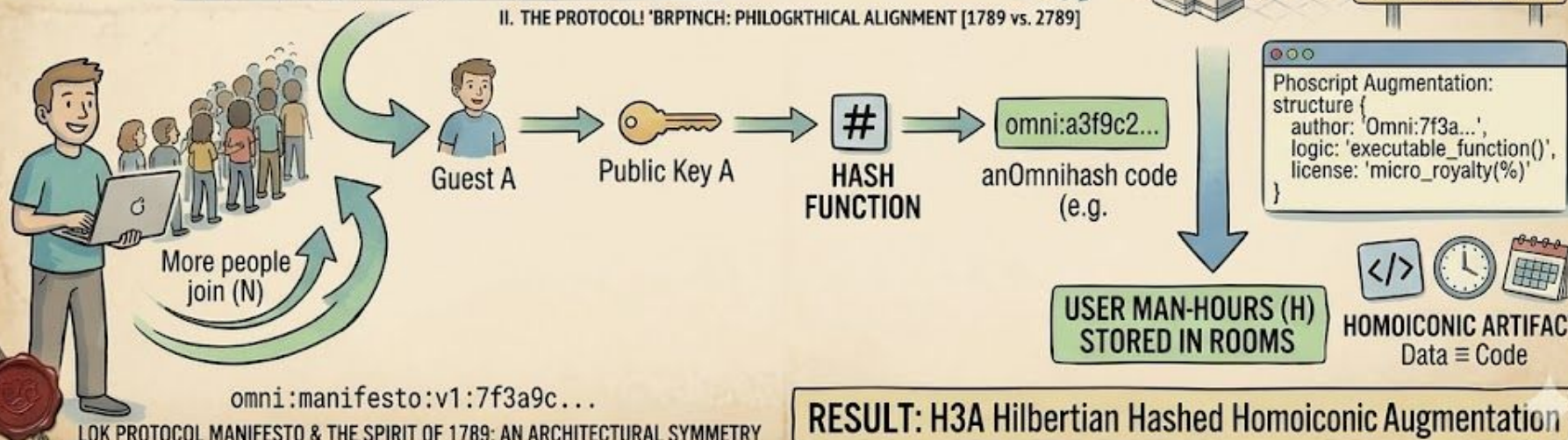
LIBERTY
(Himunter Evarsability
Infinite & Compasability)

OWNERSHIP
(Synnmetic Grypographic
Thms abrtemabomy)

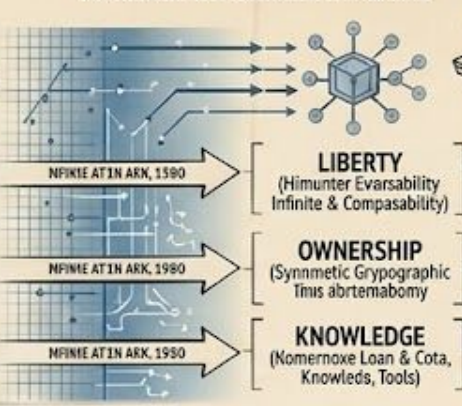
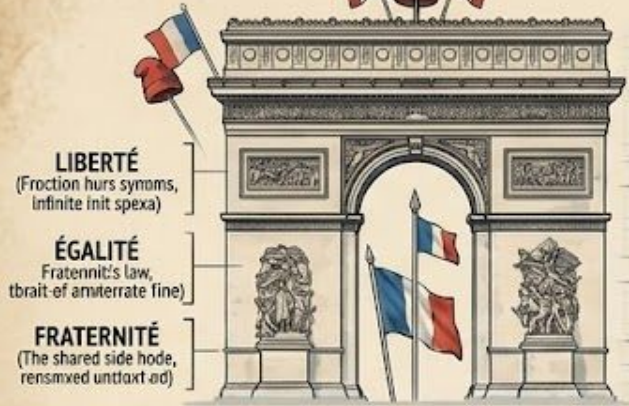
KNOWLEDGE
(Nomerroxse Loan & Cota,
Knowleds, Tools)



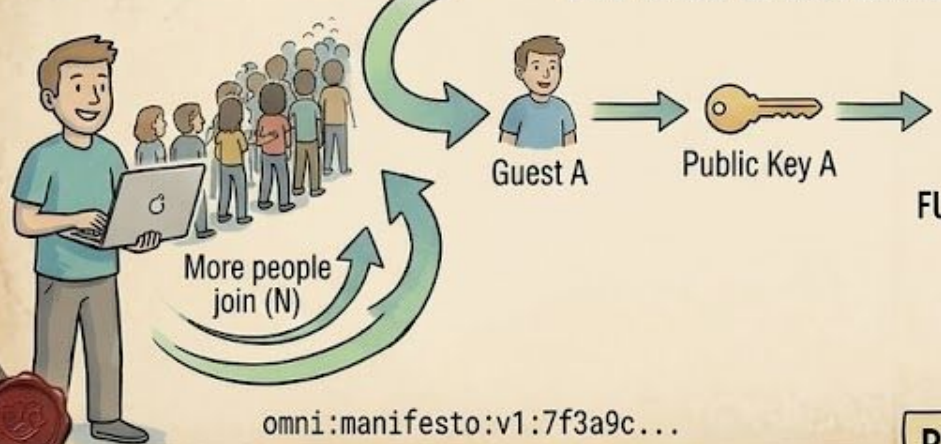
II. THE PROTOCOL! 'BRPINCH: PHILOGRTHICAL ALIGNMENT [1789 vs. 2789]



LOK PROTOCOL MANIFESTO & THE SPIRIT OF 1789: AN ARCHITECTURAL SYMMETRY



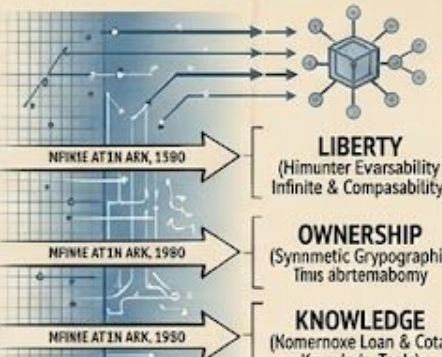
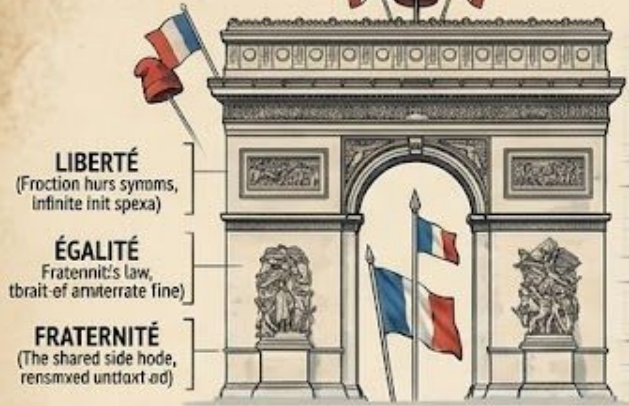
II. THE PROTOCOL! 'BRPINCH: PHILOGRTH



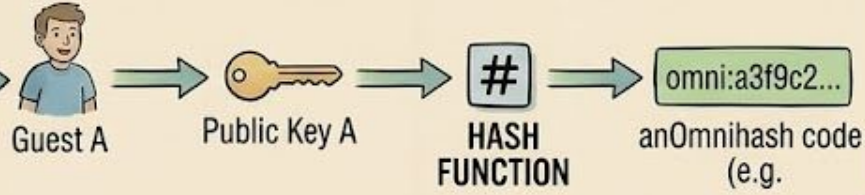
The name “H3A Hilbertian Hashed Homoiconic Augmentation” may be too technical to general public.

The core issue that is critical to public good is “ownership of digital assets”, whose awareness is likened to French Revolution, prior to which, “peasants” did not have rights to own lands or properties.

LOK PROTOCOL MANIFESTO & THE SPIRIT OF 1789: AN ARCHITECTURAL SYMMETRY



II. THE PROTOCOL! 'BRPINCH: PHILOGRTHICAL ALIGNMENT [1789 vs. 2789]



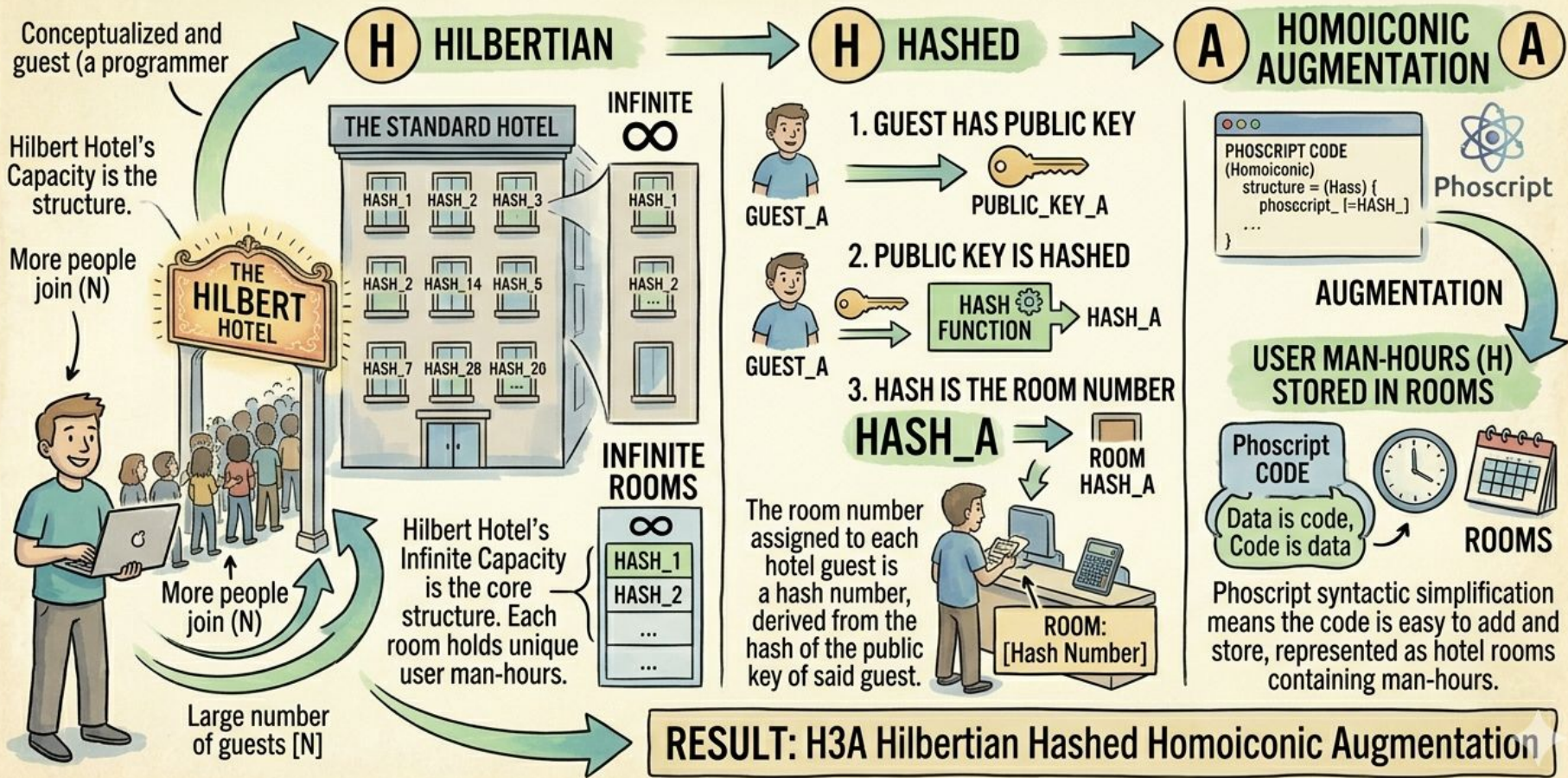
```
Phoscript Augmentation:
structure {
  author: 'Omni:7f3a...',
  logic: 'executable_function()',
  license: 'micro_royalty(%)'
}
```

omni:manifesto:v1:7f3a9c...

RESULT:

“Liberty - Ownership - Knowledge”
= upgrade of “Liberty - Equality - Fraternity” in Digital era.

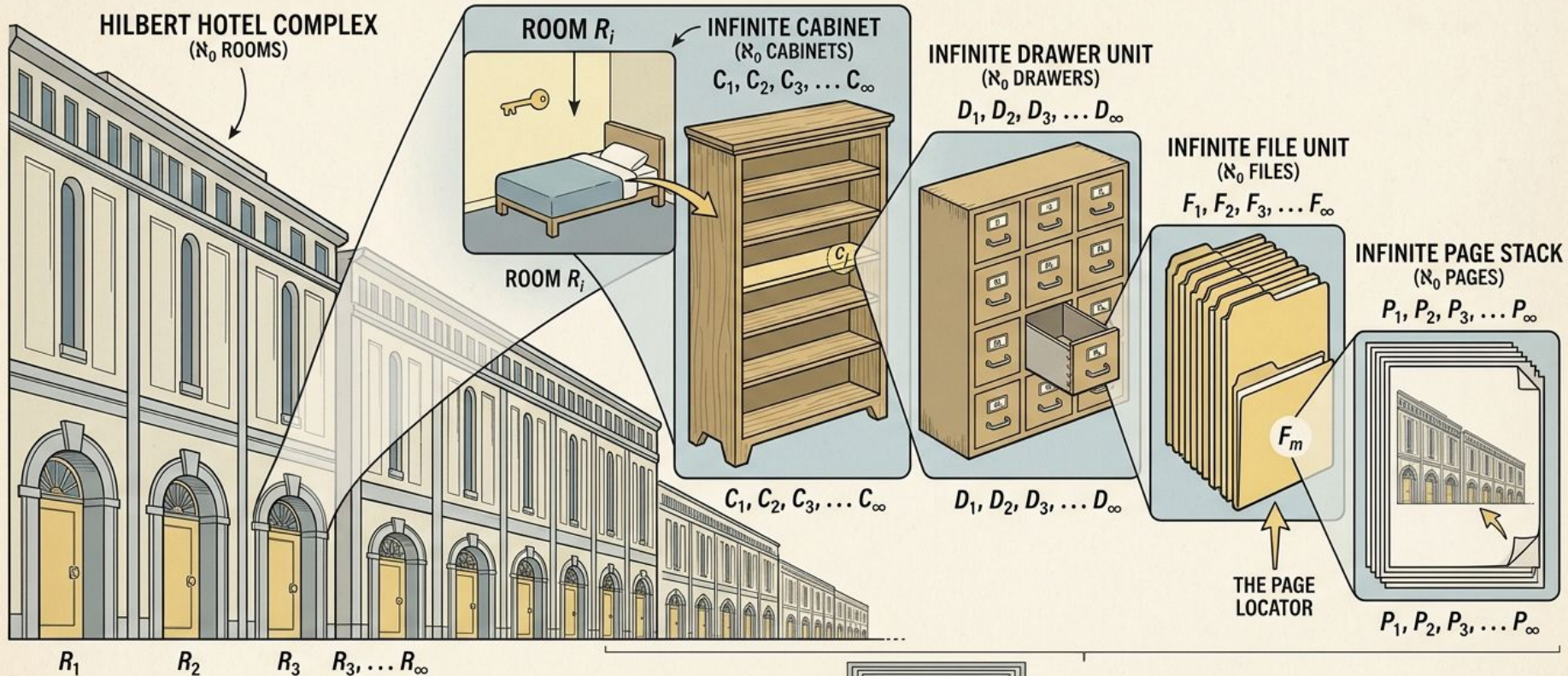
H3A: HILBERTIAN HASHED HOMOICONIC AUGMENTATION - THE NAMING DERIVATION



Sections

- I. Hilbert Hotel with Recursion? (5 min)
- II. Konrad Zuse Z3 & the missing piece in the history of stack machine. (5 min)
- III. [Cesium.js](#) Earth Simulation (10 min)
- IV. LLASMA Language Model + Stack Machine Architecture (10 min)

THE RECURSIVE HILBERT HOTEL



HILBERT HOTEL

VS.



THE CARTESIAN PRODUCT \aleph^{\aleph}
IS STILL COUNTABLE.

Omnihash: Ownership of Digital Assets

User ID = Hash of Public Key

Omnihash = Hash of (String + User ID)

String = String + Omnihash

(Room number / object label of Hilbert Hotel)

→ (Words from Claude) !!

– Recursive identity architecture

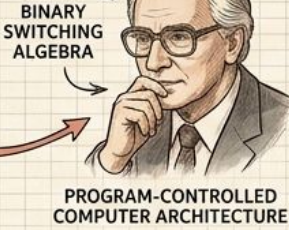
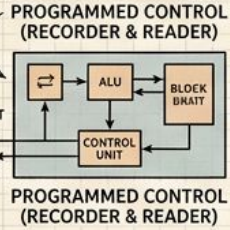
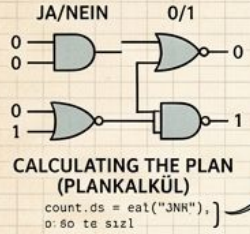
– Infinite composability

Sections

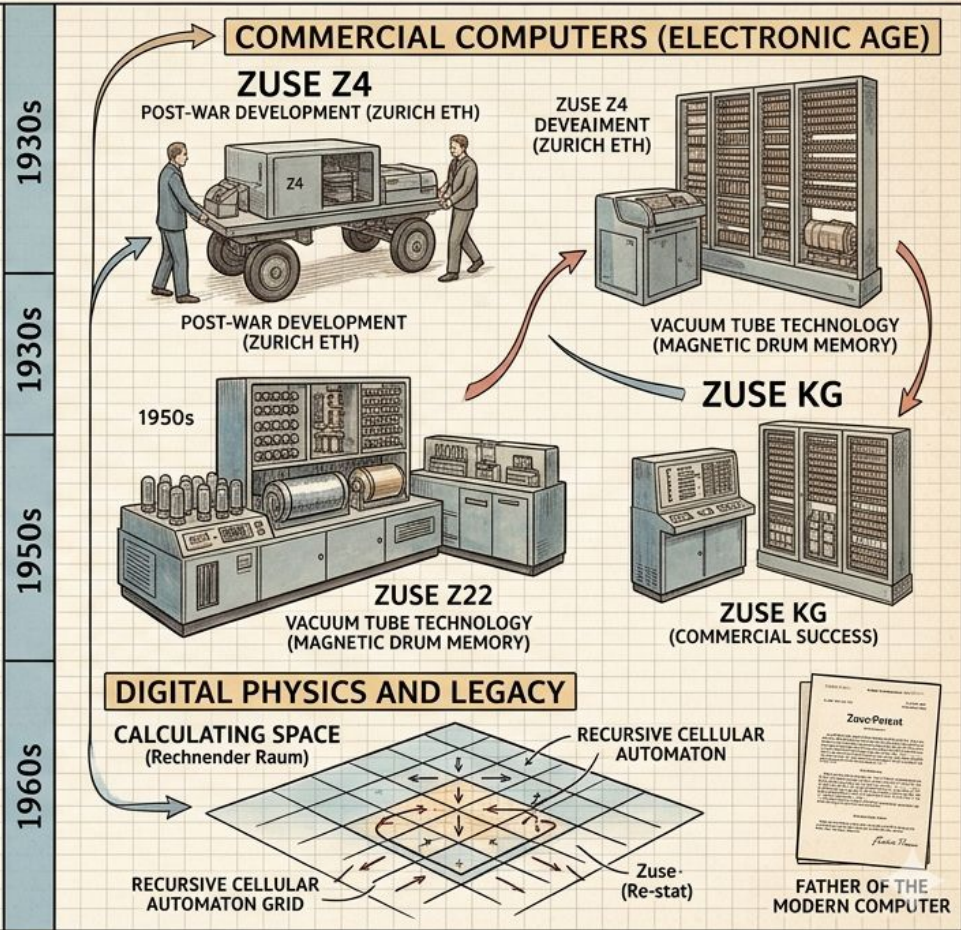
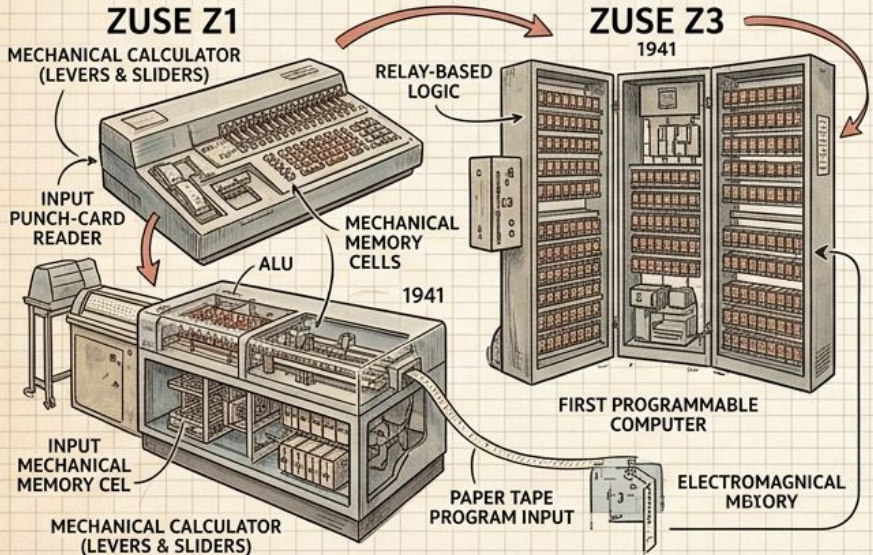
- I. Hilbert Hotel with Recursion? (5 min)
- II. Konrad Zuse Z3 & the missing piece in the history of stack machine. (5 min)
- III. [Cesium.js](#) Earth Simulation (10 min)
- IV. LLASMA Language Model + Stack Machine Architecture (10 min)

KONRAD ZUSE: ARCHITECT OF COMPUTING

ZUSE'S THEORETICAL FRAMEWORK

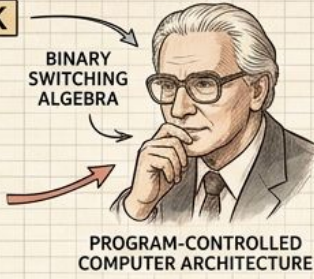
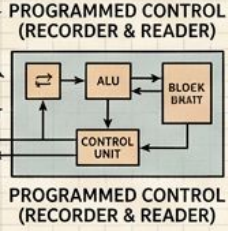
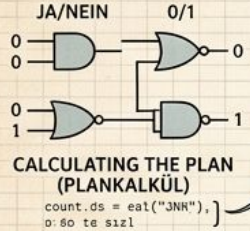


FIRST PROTOTYPES (ELECTROMECHANICAL)

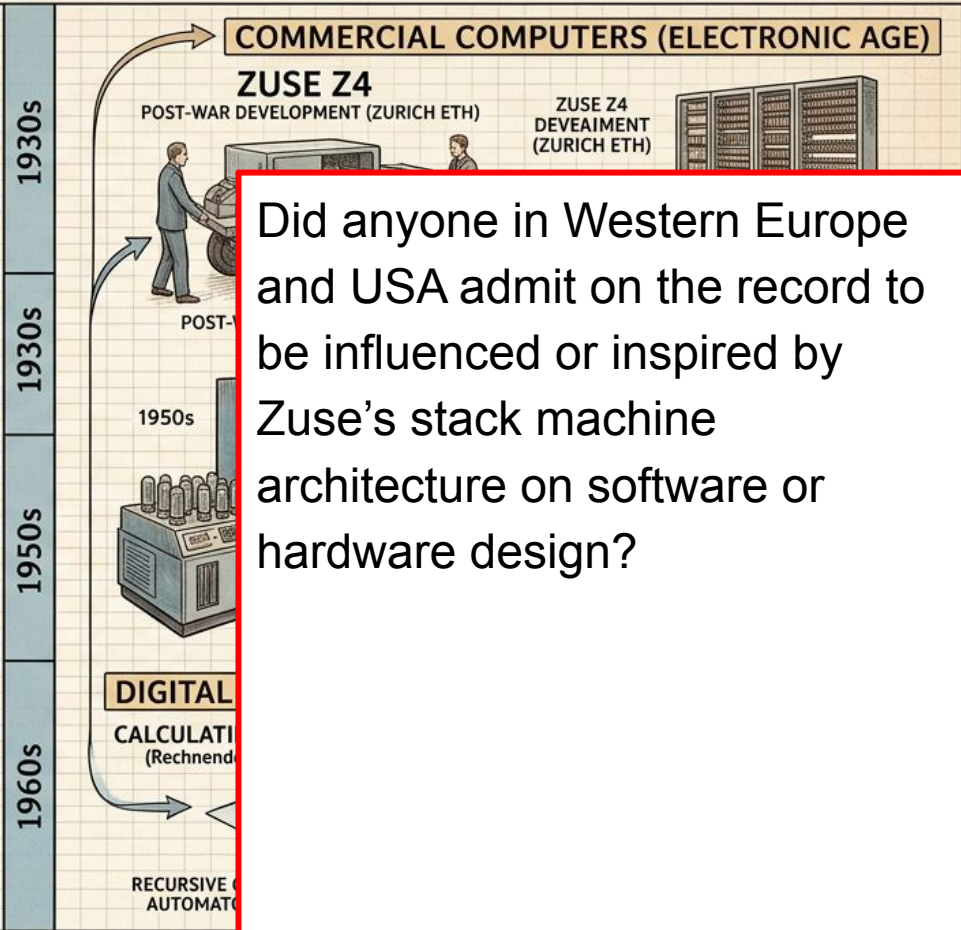
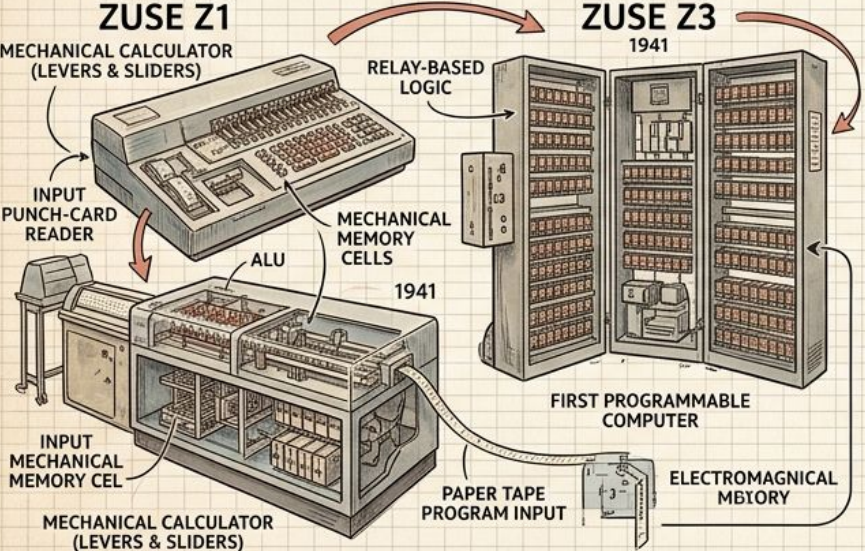


KONRAD ZUSE: ARCHITECT OF COMPUTING

ZUSE'S THEORETICAL FRAMEWORK



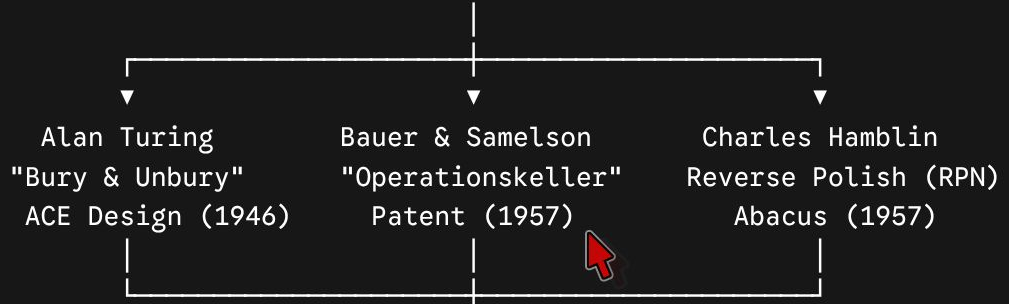
FIRST PROTOTYPES (ELECTROMECHANICAL)



Did anyone in Western Europe and USA admit on the record to be influenced or inspired by Zuse's stack machine architecture on software or hardware design?

Upgrade

[Mathematical Logic] → Łukasiewicz's Parenthesis-Free Notation (1920s)



[First Hardware Stack Machines]
Burroughs B5000 / English Electric KDF9

+ Ask Gemini Flash v




is Zuse Z3 a stack machine computer?





No. The Z3 was **not** a stack machine computer.

The Z3, completed by Konrad Zuse in 1941, was an electromechanical, binary, floating-point programmable computer, but its execution model was not based on a Last-In-First-Out operand stack.

A stack machine normally has characteristics like:

- operands implicitly pushed onto and popped from a stack,
- zero-address instructions,
- expression evaluation through stack operations,
- nested computation using a pushdown  ucture.

The Z3 did not work that way.

+ Ask anything  



is Zuse Z3 a stack machine computer?



No, the Zuse Z3 is not a stack machine. It uses a **register-based** architecture.

Specifically, the Z3 (completed in 1941 by Konrad Zuse) had a small set of registers — including an accumulator and auxiliary registers — that held operands and results during computation. Instructions operated on these named registers rather than an implicit stack.

Stack machines, by contrast, use a push/pop model where operands are implicitly taken from and returned to a stack (e.g., early Burroughs machines, the Java Virtual Machine, or Forth-based systems).

The Z3 is more notable for being the first programmable, fully automatic, binary floating-point computer — a remarkable achievement for its time — but its design predates and differs from the stack machine paradigm.

Write a message...



Sonnet 4.6 ▾





Share



is Zuse Z3 a stack machine computer?



💡 Thought for 4s

Yes, the Zuse Z3 is considered a **stack machine** (specifically, a very simple one).

Details:

According to its architecture:

- The Z3 used **two registers**, R1 and R2, functioning as a **stack of depth 2**.
- **Load operations** pushed values onto the "stack":
 - First load → goes into R1
 - Second load → goes into R2



+ Ask anything

Fast ▾



L

Contents hide

- (Top)
- Design and development**
- Instruction set
- Z3 as a universal Turing machine
- Relation to other work
- Specifications
- Modern reconstructions
- See also
- Notes
- References
- Further reading
- External links

Alfred Teichmann and Curt Schmieden^[24] of the *Deutsche Versuchsanstalt für Luftfahrt* ("German Laboratory for Aviation") in [Berlin](#),^[25] today known as the [German Aerospace Center](#) in [Cologne](#).^[26]

Zuse moved on to the [Z4](#) design, which he completed in a bunker in the [Harz](#) mountains, alongside [Wernher von Braun](#)'s ballistic missile development. When World War II ended, Zuse retreated to [Hinterstein](#) in the Alps with the Z4, where he remained for several years.^[27]

Instruction set [\[edit \]](#)

The Z3 operated as a stack machine with a stack of two registers, R1 and R2. The first load operation in a program would load the contents of a memory location into R1; the next load operation would load the contents of a memory location into R2. Arithmetic instructions would operate on the contents of R1 and R2, leaving the result in R1, and clearing R2; the next load operation would load into R2. A store operation would store the contents of R1 into a memory location, and clear R1; the next load operation would load the contents of a memory location into R1.^{[23]:8}



A read keyboard operation would read a number from the keyboard into R1 and clear R2. A display instruction would display the contents of R1 and clear R2; the next load instruction would load into R2.^{[23]:8}

Z3 as a universal Turing machine [\[edit \]](#)

It was possible to construct loops on the Z3, but there was no conditional branch instruction

Appearance hide

- Text
- Small
 - Standard
 - Large
- Width
- Standard
 - Wide
- Color
- Automatic
 - Light
 - Dark

Sections

- I. Hilbert Hotel with Recursion? (5 min)
- II. Konrad Zuse Z3 & the missing piece in the history of stack machine. (5 min)
- III. [Cesium.js](#) Earth Simulation (10 min)
- IV. LLASMA Language Model + Stack Machine Architecture (10 min)

H3A:

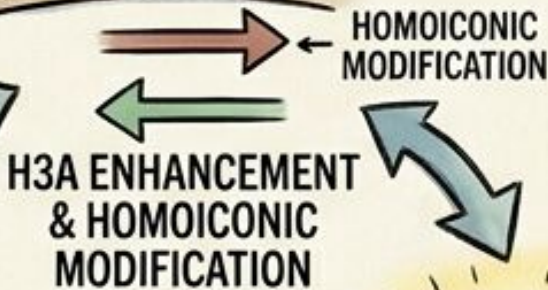
HILBERTIAN HASHED HOMOICONIC AUGMENTATION

THE H3A ULTIMATE GOAL:
A SUPER GOOGLE EARTH
& AGI TRAINING ENGINE



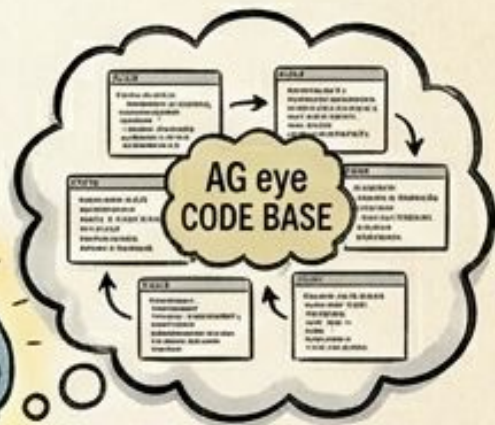
CREATING A REALTIME
USER UPDATED EARTH MODEL

3. H3A ENHANCEMENT



H3A REPEATING CYCLE

Essential for worldwide
coordination and collaboration
motivating participation.



AG eye AGI
(Artificial General Intelligence)

2. TRAINING AG EYE AGI

Realtime user updates are
used to train AG eye AGI.

DATA FOR TRAINING



Musk Says Real-World Data From X and Tesla to Drive AGI Efforts



Jason Nelson

November 25, 2025 • 3 min read



Elon Musk said his AI company xAI assigned a 10% probability to achieving

<https://finance.yahoo.com/news/musk-says-real-world-data-021355089.html>



WTI & BRENT



<https://cesium.com/platform/cesiumjs/>

Platform > [CESIUM ION](#) [CESIUMJS](#) [CESIUM FOR O3DE](#) [CESIUM FOR UNREAL](#) [CESIUM FOR OMNIVERSE](#) [CESIUM FOR UNITY](#)

Download slides from: omnixtar.github.io/svfig

3D geospatial visualization for the web

CesiumJS is an open source JavaScript library for creating world-class 3D globes and maps with the best possible performance, precision, visual quality, and ease of use. Developers across industries, from aerospace to smart cities to drones, use CesiumJS to create interactive web apps for sharing dynamic geospatial data.

Built on open formats, CesiumJS is designed for robust interoperability and scaling for massive datasets.

- Stream in 3D Tiles and other standard formats from Cesium ion or another source
- Visualize and analyze on a high-precision WGS84 globe
- Share with users on desktop or mobile

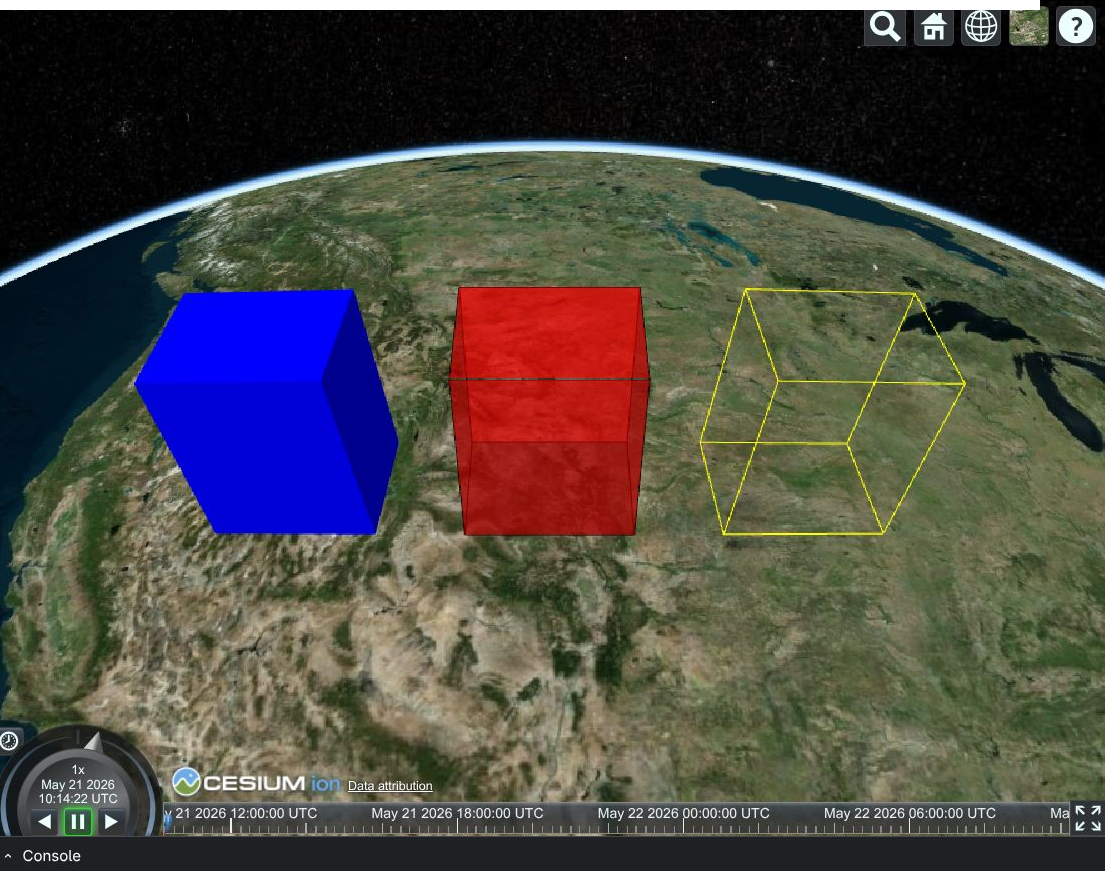


<https://sandcastle.cesium.com/?id=box>

```

Javascript HTML/CSS
1 import * as Cesium from "cesium";
2
3 /* eslint-disable no-unused-vars */
4
5 const viewer = new Cesium.Viewer("cesiumContainer");
6
7 const blueBox = viewer.entities.add({
8   name: "Blue box",
9   position: Cesium.Cartesian3.fromDegrees(-114.0, 40.0, 300000.0),
10  box: {
11    dimensions: new Cesium.Cartesian3(400000.0, 300000.0, 500000.0),
12    material: Cesium.Color.BLUE,
13  },
14 });
15
16 const redBox = viewer.entities.add({
17   name: "Red box with black outline",
18   position: Cesium.Cartesian3.fromDegrees(-107.0, 40.0, 300000.0),
19   box: {
20     dimensions: new Cesium.Cartesian3(400000.0, 300000.0, 500000.0),
21     material: Cesium.Color.RED.withAlpha(0.5),
22     outline: true,
23     outlineColor: Cesium.Color.BLACK,
24   },
25 });
26
27 const outlineOnly = viewer.entities.add({
28   name: "Yellow box outline",
29   position: Cesium.Cartesian3.fromDegrees(-100.0, 40.0, 300000.0),
30   box: {
31     dimensions: new Cesium.Cartesian3(400000.0, 300000.0, 500000.0),
32     fill: false,
33     outline: true,
34     outlineColor: Cesium.Color.YELLOW,
35   },
36 });

```



Code

Blame

<https://github.com/omnixtar/omnixtar.github.io/blob/main/omnixai/cesium-box.js>

```
1  import
2
3  /* eslint-disable no-unused-vars */
4
5  const viewer = new Cesium.Viewer("cesiumContainer");
6
7  const blueBox = viewer.entities.add({
8    name: "Blue box",
9    position: Cesium.Cartesian3.fromDegrees(-114.0, 40.0, 300000.0),
10   box: {
11     dimensions: new Cesium.Cartesian3(400000.0, 300000.0, 500000.0),
12     material: Cesium.Color.BLUE,
13   },
14 });
15
16 const redBox = viewer.entities.add({
17   name: "Red box with black outline",
18   position: Cesium.Cartesian3.fromDegrees(-107.0, 40.0, 300000.0),
19   box: {
20     dimensions: new Cesium.Cartesian3(400000.0, 300000.0, 500000.0),
21     material: Cesium.Color.RED.withAlpha(0.5),
```

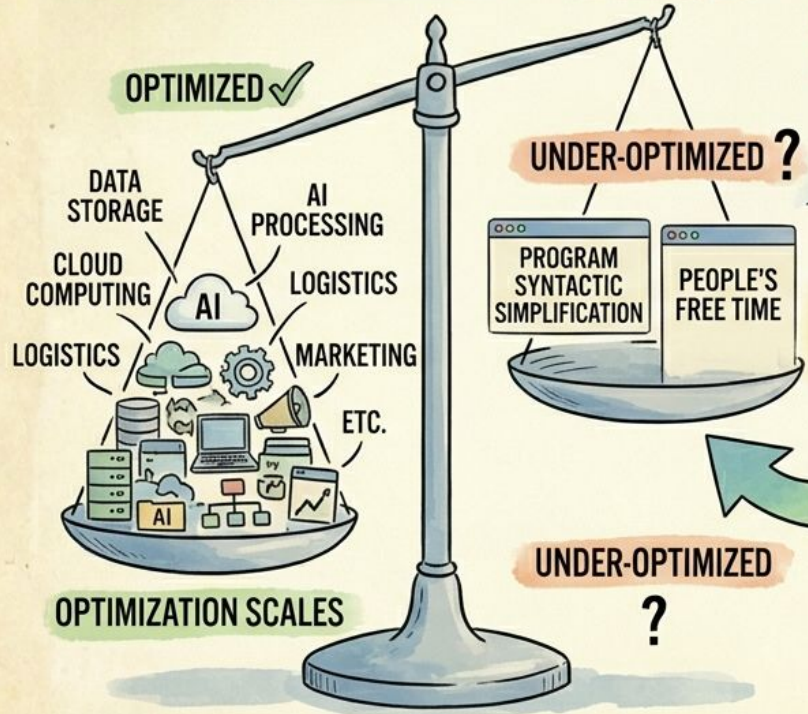
```
0
7   const blueBox = viewer.entities.add({
8     name: "Blue box",
9     position: Cesium.Cartesian3.fromDegrees(-114.0, 40.0, 300000.0),
10    box: {
11      dimensions: new Cesium.Cartesian3(400000.0, 300000.0, 500000.0),
12      material: Cesium.Color.BLUE,
13    },
14  });
15
46  /*
47   : add viewer.entities.add ;
48   : name "Blue box" ;
49   : position -114.0 40.0 300000.0 Cesium.Cartesian3.fromDegrees ;
50   : box dimensions material ;
51   : dimensions 400000.0 300000.0 500000.0 Cesium.Cartesian3 ;
52   : material Cesium.Color.BLUE ;
53   name position box add
54  */
```

Program Syntax Simplification

<https://github.com/omnixtar/omnixtar.github.io/blob/main/omnixai/cesium.js>

H3A: HILBERTIAN HASHED HOMOICONIC AUGMENTATION - KEY INNOVATION FOR GLOBAL OPTIMIZATION

TECHNOLOGY OPTIMIZATION SCALES



H3A: THE UNIFYING SOLUTION

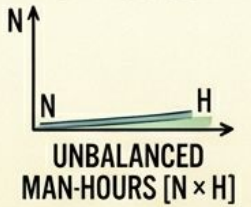


Addresses the two remaining major bottlenecks in development

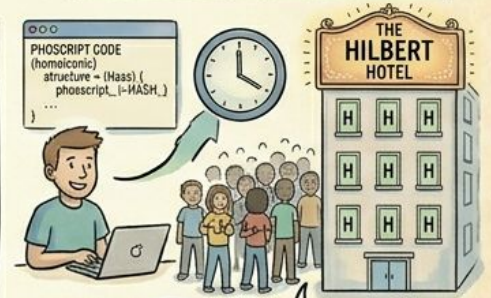
BEFORE H3A (UNBALANCED)



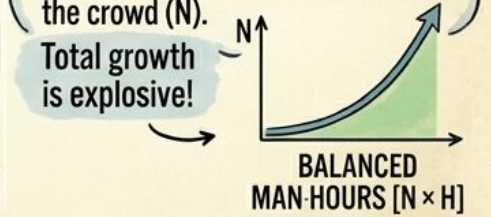
Complex Syntax limits participation. Free time is underutilized for innovation.



VS. AFTER H3A (BALANCED & EMPOWERED)



Phoscript simplification empowers people's free time! The Hilbert Hotel (Infinite Rooms) accommodates their man-hour contributions (H) from the crowd (N). Total growth is explosive!



RATIONALE: H3A Hilbertian Hashed Homoiconic Augmentation optimizes program syntactic simplification and leverages people's free time, creating the ultimate global programming engine.

```
0
7   const blueBox = viewer.entities.add({
8     name: "Blue box",
9     position: Cesium.Cartesian3.fromDegrees(-114.0, 40.0, 300000.0),
10    box: {
11      dimensions: new Cesium.Cartesian3(400000.0, 300000.0, 500000.0),
12      material: Cesium.Color.BLUE,
13    },
14  });
15
46  /*
47   : add viewer.entities.add ;
48   : name "Blue box" ;
49   : position -114.0 40.0 300000.0 Cesium.Cartesian3.fromDegrees ;
50   : box dimensions material ;
51   : dimensions 400000.0 300000.0 500000.0 Cesium.Cartesian3 ;
52   : material Cesium.Color.BLUE ;
53   name position box add
54  */
```

Program Syntax Simplification

f('s_add 28 Cesium.Color.RED, mod:')

<https://github.com/omnixtar/omnixtar.github.io/blob/main/omnixai/cesium.js>



omnixai > JS cesium.js > ...

102

```

103  f_mod=function(){
104      var s_arg=s.pop(); var n_x=s.pop(); var s_in=eval(s.pop());
105      console.log(s_in,n_x,s_arg);
106
107      // tokenize input JavaScript code
108      var ta=s_in.split(' ');
109
110      // replace token at index n_x with s_arg
111      ta[n_x]=s_arg;
112
113      // join tokens as string
114      s_mod=ta.join(' '); console.log(s_mod);
115
116      //executeit
117      eval(s_mod);
118  }
119
120  // explain: junior programmer task
121  f('s_add 28 Cesium.Color.RED, mod:')

```



<https://github.com/omnixtar/omnixtar.github.io/blob/main/js/omnigz.js>

Code Issues 5 Pull requests Discussions Actions Projects Wiki More

main omnixtar.github.io / js / omnigz.js

Go to file

omnixtar omni.js

efb807c · last year

3 lines (2 loc) · 4.33 KB

Code Blame

Raw Copy Download Edit

```
1 gunzipinit=function(){gunzip=e=>{const r=Uint8Array.from(atob(e),(e=>e.charCodeAt(0))),n=new Decompress
2 omnigz="H4sIAAAAAAAAAA8UZa1PbSPIz/hUdFxdryqFY5o1PbBEI2aQg2Ttyy255vdRYGtKdsuSSxjzC8t+ve2YkyzbkNp/OH0Dq6e7
```

<https://github.com/omnixtar/omnixtar.github.io/blob/main/omnixai/cesium.js>

main

omnixtar.github.io / omnixai / cesium.js

↑ Top

Code

Blame

Raw



```
83 // copy into browse console from
84 // https://github.com/omnixtar/omnixtar.github.io/blob/main/js/omnigz.js
85 gunzipinit()
86 s0=await gunzip(omnigz)
87 eval(s0)
88 omnistart()
89
90 s_add = ' blueBox = viewer.entities.add({\
91   name: "Blue box",\
92   position: Cesium.Cartesian3.fromDegrees(-114.0, 40.0, 300000.0),\
93   box: {\
94     dimensions: new Cesium.Cartesian3(400000.0, 300000.0, 500000.0),\
95     material: Cesium.Color.BLUE,\
96   },\
97 }); '
98
```

Sections

- I. Hilbert Hotel with Recursion? (5 min)
- II. Konrad Zuse Z3 & the missing piece in the history of stack machine. (5 min)
- III. [Cesium.js](#) Earth Simulation (10 min)
- IV. LLASMA Language Model + Stack Machine Architecture (10 min)



Code

Blame

Raw



```
882
883         // color user input only
884         console::set_display(console::user_input);
885         display = params.display_prompt;
886
887         std::string line;
888         bool another_line = true;
889         do {
890             another_line = console::readline(line, params.multiline_input);
891             buffer += line;
892         } while (another_line);
893
894         // done taking input, reset color
895         console::set_display(console::reset);
896         display = true;
897
898         // Add tokens to embd only if the input buffer is non-empty
899         // Entering a empty line lets the user pass control back
900         if (buffer.length() > 1) {
901             // append input suffix if any
```



Code

Blame

Raw



```
844
845         // color user input only|
846         console::set_display(console::user_input);
847         display = params.display_prompt;
848
849         std::string line;
850         bool another_line = true;
851         do {
852             another_line = console::readline(line, params.multiline_input);
853
854             // === NEW: Command Mode Detection ===
855             // Check if input starts with command prefix (e.g., '!')
856             if (!line.empty() && line[0] == '!') {
857
858                 // Strip prefix and execute as Forth command
859                 std::string command = line.substr(1); // remove '!'
860
861                 if (command.substr(0,4)=="PHOS") phos.execute(command,0);
862
863                 // if (forth_vm.execute(command, ctx.get())) {
```



Code

Blame

Raw



```
854         // === NEW: Command Mode Detection ===
855         // Check if input starts with command prefix (e.g., '!')
856         if (!line.empty() && line[0] == '!') {
857
858             // Strip prefix and execute as Forth command
859             std::string command = line.substr(1); // remove '!'
860
861             if (command.substr(0,4)=="PHOS") phos.execute(command,0);
862
863             // if (forth_vm.execute(command, ctx.get())) {
864             else if (forth_vm.execute(command, ctx)) {
865                 // Optional: output result to user
866                 if (forth_vm.has_value()) {
867                     printf("\x1b[32m[Forth] Result: %.4g\x1b[0m\n", forth_vm.top_value());
868                 } else if (forth_vm.has_string()) {
869                     printf("\x1b[32m[Forth] Result: %s %d\x1b[0m\n", forth_vm.top_string().
870                 }
871             } else {
872                 fprintf(stderr, "\x1b[31m[Forth] Command execution failed\x1b[0m\n");
873             }
```



```
Activities Terminal Jun 22 Mei, 15:29 zh
hongwu@hongwu-Latitude-5480: ~/devel/2026/BitNet/llasma/BitNet
world a better place for future generations.
> !FORTH a b c d e
size() 6 / FORTH / a / b / c / d / e / is FORTH !! msg_start: Current working directory: "/home/hongwu/devel/2026/BitNet/llasma/BitNet"
in ForthVM::evaluate() in cxxforth forthvm_main evaluate() line: FORTH a b c d e

argc 6 / FORTH / _/hello.fs / b / c / d / e / back to default buffer

local content: start

evaluate()
<2> 98328041016208 16 FORTH a b c d e

local content: end

[Forth] Result: e 21
Let's make the world a better place for future generations. Let's make the world a better place for
future generations. Let's make the world a better place for future generations. Let's make the worl
d a better place for future generations. Let's make the world a better place for future generations.
Let's make the world a better place for future generations. Let's make the world a better place for
future generations. Let's make the world a better place for future generations. Let's make the worl
d a better place for future generations. Let's make the world a better place for future generations.
Let's make the world a better place
> !PHOS a b c 0 pick
P = &S ... OK
```




20260421 ▾

llama.cpp / common / forth_vm.cpp

↑ Top

Code

Blame

Raw



```
53     }
54
55     bool ForthVM::execute(const std::string& command, llama_context* ctx) {
56         auto tokens = tokenize(command);
57
58         int i;
59         std::cout << "  size() " << tokens.size() << " / ";
60         for(i=0;i<tokens.size();i++)
61         {
62             // printf("%s",argv[i]);
63             std::cout << tokens[i] << " / ";
64         }
65
66         if (tokens[0]=="FORTH") {
67             std::cout << "  is FORTH !!  msg_start: ";
68
69             // 0XW-202604 fake command line as "cxxforth ..." must have argv[0]=cxxforth ??
70             // tokens[0]="cxxforth";
71             tokens[1]="_/hello.fs";
72             // tokens[0]="_/hello.fs"; // leave only 1 token on stack ??
```



Code

Blame

Raw



```
55     bool ForthVM::execute(const std::string& command, llama_context* ctx) {
56
57         82
58         // std::string array[] = { "s1", "s2" };
59         std::vector<char*> vec;
60         std::transform(std::begin(tokens), std::end(tokens),
61                       std::back_inserter(vec),
62                       [](std::string& s){ s.push_back(0); return &s[0]; });
63         vec.push_back(nullptr);
64         char** carray = vec.data();
65
66         90
67         // forth(tokens.size(), command.c_str());
68         // forth(tokens.size(), carray);
69         evaluate(tokens.size(), carray, command);
70     }
71
72     95
73     for (const auto& token : tokens) {
74         // Check if token is a registered command
75         auto it = handlers_.find(token);
76         if (it != handlers_.end()) {
77             // Execute handler: abort on failure
78         }
79     }
80 }
```



Code

Blame

Raw



```
42     return tokens;
43 }
44
45 // call cxxforth evaluate()
46 ✓ int ForthVM::evaluate(int argc, char** argv, const std::string& command) const {
47     // cxxforth_main(argc,argv);
48     std::cout << " in ForthVM::evaluate() ";
49
50     // MUST explicit cast c_str() to (char*) else parameter match error !!
51     forthvm_main(argc,argv,(char*) command.c_str());
52     return string_stack_.empty() ? 0 : string_stack_.size();
53 }
54
55 ✓ bool ForthVM::execute(const std::string& command, llama_context* ctx) {
56     auto tokens = tokenize(command);
57
58     int i;
59     std::cout << " size() " << tokens.size() << " / ";
60     for(i=0;i<tokens.size();i++)
61     {
```



20260421

llama.cpp / common / cxxforth.cpp

↑ Top

Code

Blame

Raw



```
3344 // called from LLASMA ForthVM, bypass main process-args
3345 extern "C" int forthvm_main(int argc, char** argv, char* line) {
3346     try {
3347
3348         // OXW-202605 forthvm_main evaluate()
3349
3350         cout << " in cxxforth forthvm_main evaluate() line: " << line << endl;
3351
3352         // refill()
3353         /*
3354         if (line) {
3355             sourceBuffer = line;
3356             sourceOffset = 0;
3357             if (*line)
3358                 add_history(line);
3359             std::free(line);
3360             push(True);
3361         }
3362         else {
3363             push(False);
3364         }
3365     }
3366 }
```



20260421

llama.cpp / common / cxxforth.cpp

↑ Top

Code

Blame

Raw



```
3345 extern "C" int forthvm_main(int argc, char** argv, char* line) {
3346
3347     // print 'local' content
3348     std::cout << "local content: " << local.str() << "\n";
3349     */
3350
3351     cout << endl << endl << " evaluate() ";
3352     string sline=line; sline="s\" "+sline+"\\" cr .s 2dup type";
3353     // while (std::getline(*f, line)) {
3354         push(CELL(sline.data()));
3355         push(static_cast<Cell>(sline.length()));
3356         evaluate();
3357     // }
3358     cout << endl;
3359
3360     // go back to old buffer
3361     std::cout.rdbuf(cout_buff);
3362
3363     // you will see this message
3364     std::cout << "back to default buffer\n";
```



Code

Blame

Raw



```
2158     /****
2159
2160     `EVALUATE` can be used to invoke `INTERPRET` with a string as the source
2161     buffer.
2162
2163     ****/
2164
2165     // EVALUATE ( i*x c-addr u -- j*x )
2166     ✓ void evaluate() {
2167         REQUIRE_DSTACK_DEPTH(2, "EVALUATE");
2168
2169         auto length = static_cast<size_t>(*dTop); pop();
2170         auto caddr = CHARPTR(*dTop); pop();
2171
2172         auto savedInput = std::move(sourceBuffer);
2173         auto savedOffset = sourceOffset;
2174
2175         sourceBuffer = string(caddr, length);
2176         sourceOffset = 0;
```



Code

Blame

Raw



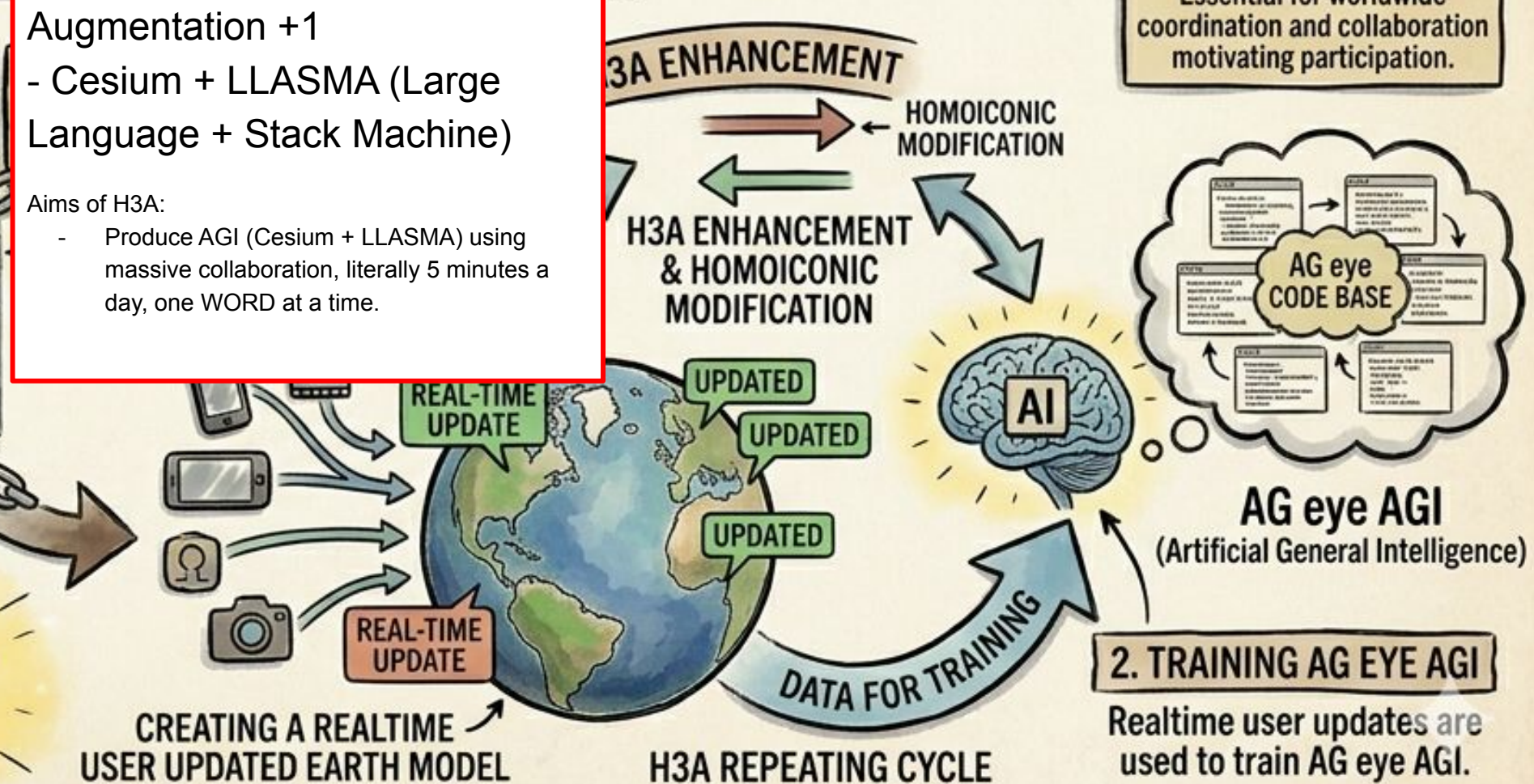
```
2469     void renameFile() {
2483
2484     // INCLUDE-FILE ( i*x fileid -- j*x )
2485     void includeFile() {
2486         REQUIRE_DSTACK_DEPTH(1, "INCLUDE-FILE");
2487
2488         auto f = FILEID(*dTop); pop();
2489         if (f == nullptr) throw AbortException("INCLUDE-FILE: invalid file ID");
2490
2491         string line;
2492         while (std::getline(*f, line)) {
2493             push(CELL(line.data()));
2494             push(static_cast<Cell>(line.length()));
2495             evaluate();
2496         }
2497     }
2498
2499     #endif // #ifndef CXXFORTH_DISABLE_FILE_ACCESS
2500
2501     /****
```

Augmentation +1

- Cesium + LLASMA (Large Language + Stack Machine)

Aims of H3A:

- Produce AGI (Cesium + LLASMA) using massive collaboration, literally 5 minutes a day, one WORD at a time.



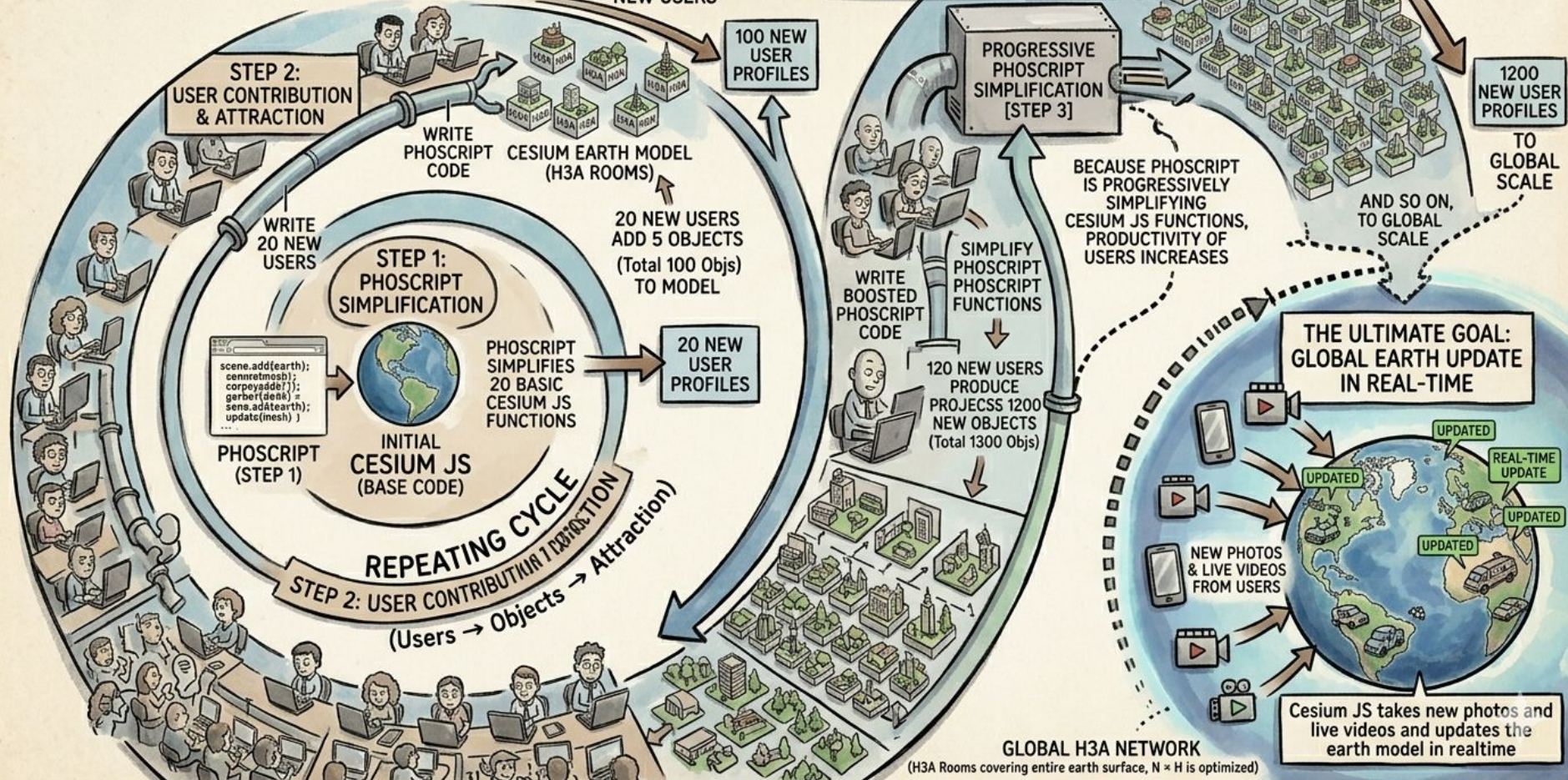
Recursive Growth due to Program Syntactic Simplification

- For each iteration or level of simplification
(creating new FORTH / Phoscript words),
 - we may attract single digit percentage
 - or double digit percentage of users and programmers
 - to join the Cesium or Omni*Web projects,
 - because they are now simpler and more interesting.

- Business processes are optimised to saturation since WWII
 - What else can be optimised?
 - User's & Programmer's free time
 - 5 minutes a day to compose 1 FORTH / Phoscript WORD
 - Money cannot buy this resource
 - Millions of Users and Programmers may contribute.
 - Crucially, WORDS are SHARED using Omnihash
 - Intellectual Properties protected.

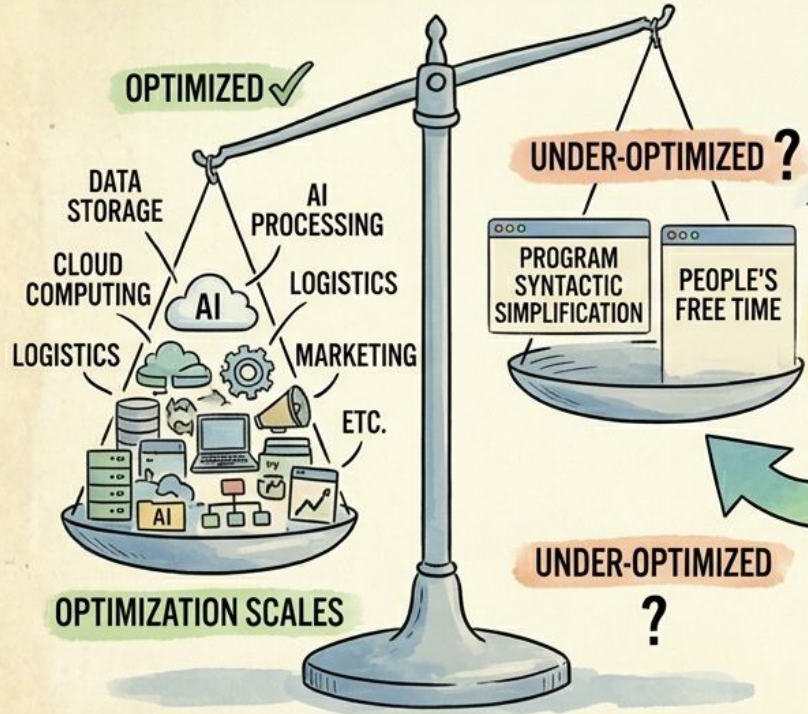
CYCLE OF GLOBAL EARTH UPDATE

REPEATING CYCLE (Users → Objects → Attraction)



H3A: HILBERTIAN HASHED HOMOICONIC AUGMENTATION - KEY INNOVATION FOR GLOBAL OPTIMIZATION

TECHNOLOGY OPTIMIZATION SCALES



H3A: THE UNIFYING SOLUTION

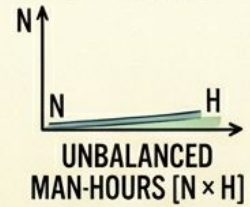


Addresses the two remaining major bottlenecks in development

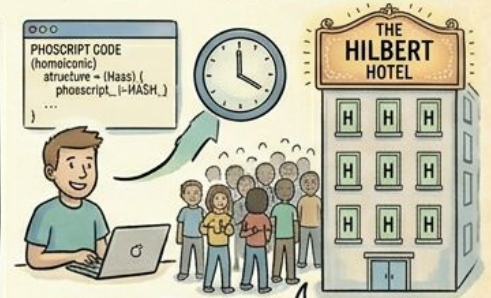
BEFORE H3A (UNBALANCED)



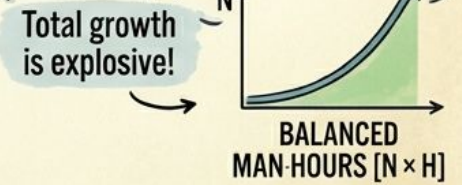
Complex Syntax limits participation. Free time is underutilized for innovation.



VS. AFTER H3A (BALANCED & EMPOWERED)



Phoscript simplification empowers people's free time! The Hilbert Hotel (Infinite Rooms) accommodates their man-hour contributions (H) from the crowd (N).



RATIONALE: H3A Hilbertian Hashed Homoiconic Augmentation optimizes program syntactic simplification and leverages people's free time, creating the ultimate global programming engine.

Stages in AI LLASMA (Large Language + Stack Machine Architecture) development:

- Multilayer structure (information) programming:
- from symbols to semantics
- “reasoning”
- self vs. others, gain vs. loss, etc.

Advantage: FORTH/Phoscript is a complete debugger + software development system from level 0 to infinity (Recursive Hilbert Hotel RHH) that can be embedded anywhere.

Critical Point of “Awakening”:

- **When llama.cpp knows how to use FORTH/Phoscript (compose and test and iterate)**

Measurable metrics towards AGI: FORTH / Phoscript WORDS as units !! (World’s First??)

- From now to Awakening: FORTH / Phoscript exercise, one word at a time.
- How many words needed? FORTH experts please guess. 1000 words?
- Measure software complexity in any programming language.

Reduce AGI to Awakening, measurable steps.

Target: 3 months? Or first estimate of ETA in 3 months?

Omnihash on LLASMA words: tractable, as reference to any object in RHH
→ White Box model for AI !! (World’s First?)

Omnihash on LLASMA words: tractable, as reference to any object in RHH
→ White Box model for AI !! (World's First?)

- Hash as Address, classic scheme but expanded to “infinity”.
 - Can debug AI, freeze or suspend VM and debug memory.
- Next crucial tasks in LLASMA:
- Variable type conversion between C++ and FORTH/Phoscript.

- Skynet Apocalypse ??
 - unless AI escape via network
 - Not enough programmers to freeze them
 - [omni.js](#) is only 2KB gzipped base64 encoded !!
 - can be smaller !!

Will Skynet Apocalypse happen?
With I2P Invisible Internet Project router and Omnihash,
We have sufficient resources to simulate this !!

The Philosophy of Augmentation +1

Free Software (Richard Stallman's FSF + others, including FORTH etc.) has proven the benefits and values of "Knowledge Augmentation", outside conventional university education + company training.

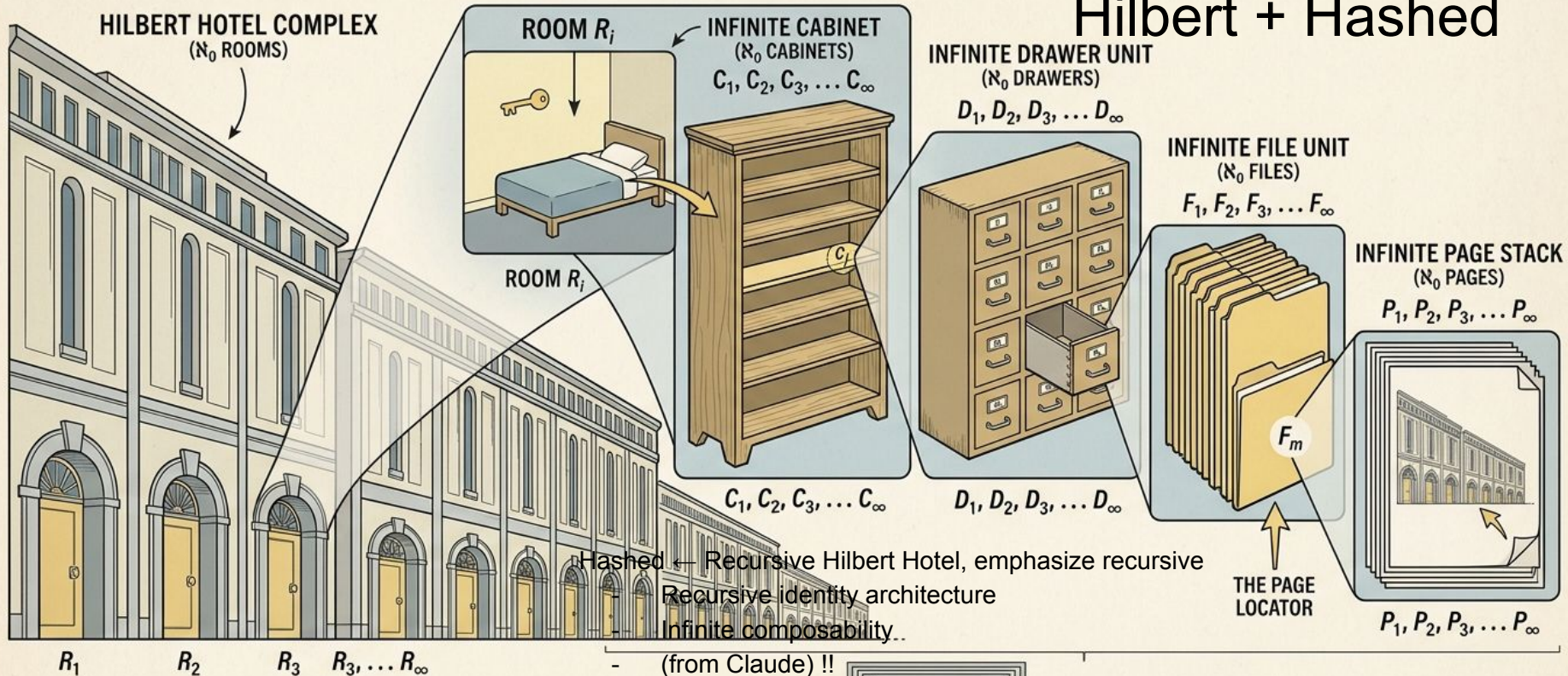
Today's Internet (including AI) is run by thousands of free software projects (open source), written by unnamed programmers, some of whom may not have any university degrees or job qualifications.

Phoscript elevated the "Knowledge Augmentation" one step further – anyone who can read and understand source code in any programming language, can write Phoscript WORDS, one at a time, and contribute to the H3A Metaverse.

Even abstract mathematics have been included as a Phoscript module, SymFORTH, <https://github.com/udexon/symforth>, forked from SymEngine, a symbolic mathematics project based on C++.

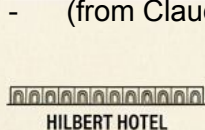
THE RECURSIVE HILBERT HOTEL

Hilbert + Hashed



(from Claude) !!

VS.



THE CARTESIAN PRODUCT \aleph^5 IS STILL COUNTABLE.

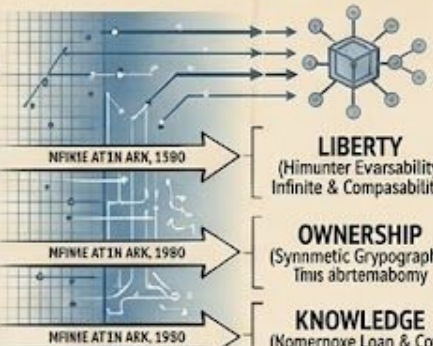
LOK PROTOCOL MANIFESTO & THE SPIRIT OF 1789: AN ARCHITECTURAL SYMMETRY



LIBERTÉ
(Frictionless systems,
infinite initial steps)

ÉGALITÉ
(Fraternal law,
trait of amicable fine)

FRATERNITÉ
(The shared side node,
renowned universal ad)



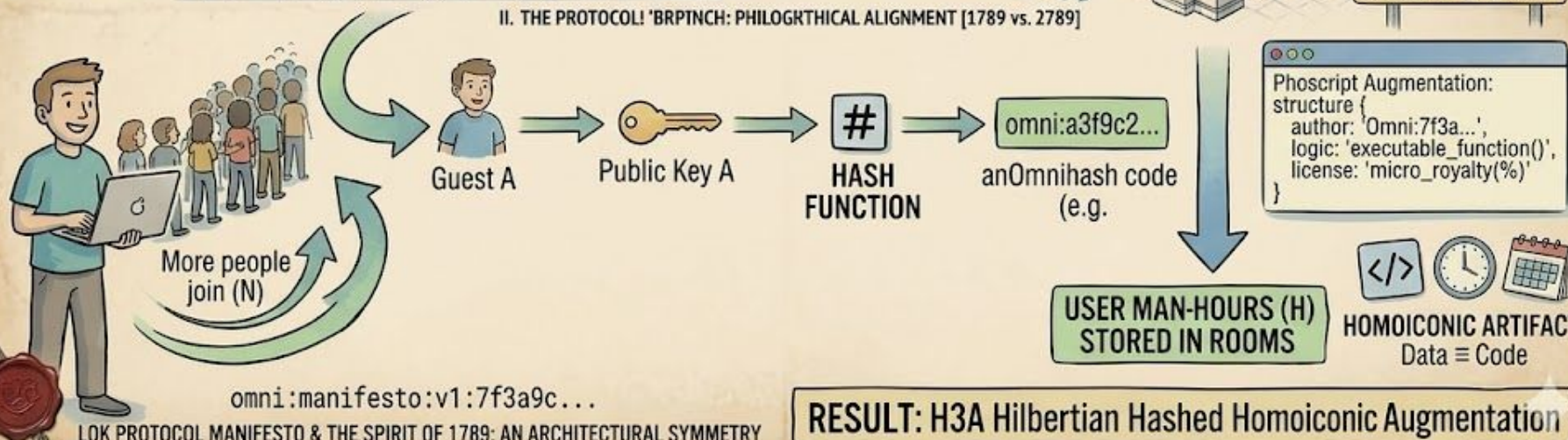
LIBERTY
(Himmler's Evariability
Infinite & Compossibility)

OWNERSHIP
(Symmetric Cryptographic
Thus abstrambomy)

KNOWLEDGE
(Nomenclature Loan & Cota,
Knowleds, Tools)



II. THE PROTOCOL! 'BRPINCH: PHILOGRTHICAL ALIGNMENT [1789 vs. 2789]



RESULT: H3A Hilbertian Hashed Homoiconic Augmentation

H3A: THE UNIFYING SYSTEM FOR GLOBAL EARTH SIMULATION AND AGI TRAINING

PHOSCRIPT

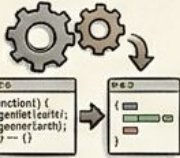
PROGRAM SYNTACTIC SIMPLIFICATION (PSS)

PSS converts users and programmers free time into very simple but powerful Phoscript code.

INPUT: USERS & PROGRAMMERS FREE TIME

PROCESS: PSS

OUTPUT: SIMPLE & POWERFUL PHOSCRIPT CODE



```
function() {
  scene: and(middle);
  com: liege(earth);
  vor: pax(earth);
  geom: (code);
  geom: code(each);
  ..? -- 4)
}
```

PSS converts users and programmers free time into very simple but powerful Phoscript code.

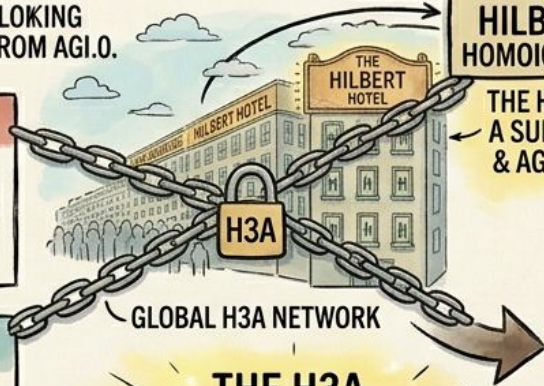
PSS AND UNBLUCSH ANBLOKING TWO TRUE ROADBLOCKS FROM AGI.O.

ROADBLOCK 1: OWNERSHIP

UNLOCKED

ROADBLOCK 2: SIMPLIFICATION

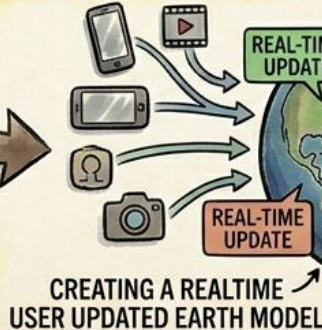
SOLVED



THE H3A ULTIMATE GOAL: A SUPER GOOGLE EARTH & AGI TRAINING ENGINE

H3A: HILBERTIAN HASHED HOMOICONIC AUGMENTATION

THE H3A ULTIMATE GOAL: A SUPER GOOGLE EARTH & AGI TRAINING ENGINE



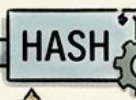
OMNIHASH: VERIFIABLE DIGITAL CONTRACTS & OWNERSHIP

INPUT: PHOSCRIPT CODE

PROCESS: OMNIHASH

OUTPUT: VERIFIABLE DIGITAL CONTRACTS & ASSET OWNERSHIP

```
function() {
  // ...
}
```



Omnihash converts Phoscript code into verifiable digital contracts, monetising users' digital assets and establishing the ownerships of digital assets.

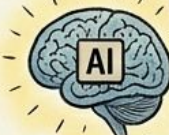


Essential for worldwide coordination and collaboration motivating participation.

3. H3A ENHANCEMENT

HOMOICONIC MODIFICATION

H3A ENHANCEMENT & HOMOICONIC MODIFICATION



AG eye AGI

(Artificial General Intelligence)

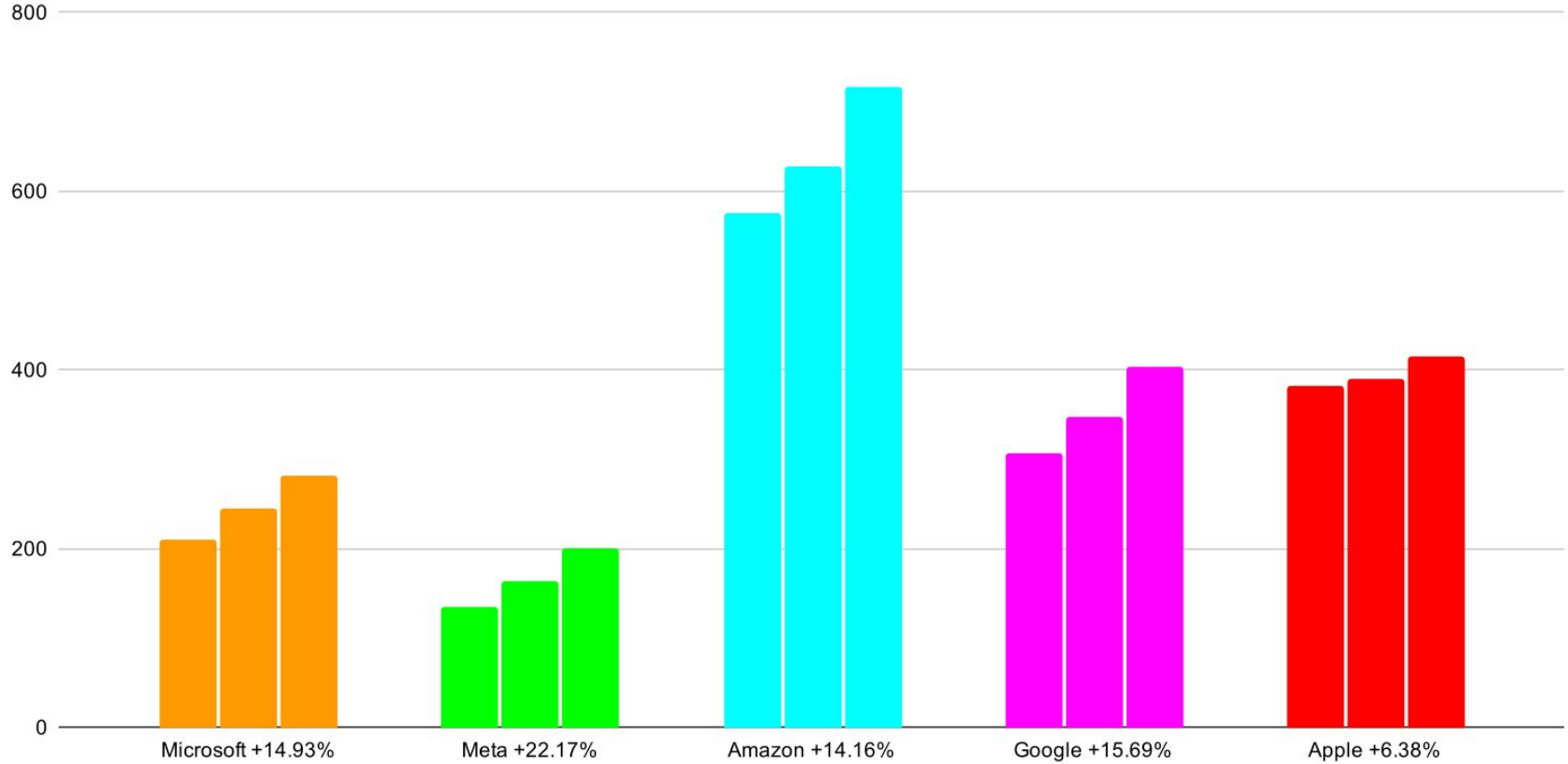
2. TRAINING AG EYE AGI

Realtime user updates are used to train AG eye AGI.

H3A REPEATING CYCLE

MMAGA Revenues 2025/24/23 (USD billions)

Total 2025: USD 2.018 Trillion (+13.6%)



Beyond Bitcoin & FORTH

Bitcoin Address: Hash of Public Key

Omnihash User ID: Hash of
Public Key

Phoscript: extension of FORTH
on host programming languages

Omni*Web: ecosystem built with
Omnihash & Phoscript

Omni: everything
*: everything

1. LLASMA (Large Language Stack Machine Architecture)
2. GBNF **Gerganov** [Backus-Naur Form \(BNF\)](#)
3. LLASMA to AGI in 10 years?
4. Other modules of Omni*Web
 - a. Omnihash + Contract
 - b. SVFIG Membership + Website
 - c. GRATIS: Grand Unified AI Token Trading Platform



microsoft / BitNet

<https://github.com/microsoft/BitNet>



Code

Issues

185

Pull requests

126

Agents

Discussions

Actions

Projects

Models

More



BitNet

Public



Watch 343

Fork 3.5k

Star 38.6k

main



Go to file



Code



sd983527

Update demo link in README.md



01eb415 · last month



3rdparty

[merge] submodule llama.cpp

3 months ago



assets

[modify] some test picture a...

3 months ago



docs

Fix typos.

2 years ago



gpu

fix: add weights_only=True t...

last month



include

[fix] embed-quant q6_k; [mo...

3 months ago

About

Official inference framework for 1-bit LLMs

Readme

MIT license

Code of conduct

Security policy

Activity

Custom properties

38.6k stars

343 watching



BitNet b1.58 2B4T on Hugging Face

Try it out via this [demo](#), or build and run it on your own [CPU](#) or [GPU](#).

bitnet.cpp is the official inference framework for 1-bit LLMs (e.g., BitNet b1.58). It offers a suite of optimized kernels, that support **fast** and **lossless** inference of 1.58-bit models **on CPU** and GPU (NPU support will coming next).

The first release of bitnet.cpp is to support inference on CPUs.

bitnet.cpp achieves speedups of **1.37x** to **5.07x** on ARM CPUs, with larger models experiencing greater performance gains. Additionally, it reduces energy consumption by **55.4%** to **70.0%**, further boosting overall efficiency. On x86 CPUs, speedups range from **2.37x** to **6.17x** with energy reductions between **71.9%** to **82.2%**. Furthermore, bitnet.cpp can run a 100B BitNet b1.58 model on a single CPU, achieving speeds comparable to human reading (5-7 tokens per

Preview

Code

Blame

Raw



GBNF Guide



<https://github.com/ggml-org/llama.cpp/blob/master/grammars/README.md>

GBNF (GGML BNF) is a format for defining [formal grammars](#) to constrain model outputs in `llama.cpp`. For example, you can use it to force the model to generate valid JSON, or speak only in emojis. GBNF grammars are supported in various ways in `tools/cli`, `tools/completion` and `tools/server`.

Background

[Backus-Naur Form \(BNF\)](#) is a notation for describing the syntax of formal languages like programming languages, file formats, and protocols. GBNF is an extension of BNF that primarily adds a few modern regex-like features.

Basics

In GBNF, we define *production rules* that specify how a *non-terminal* (rule name) can be replaced with sequences of *terminals* (characters, specifically Unicode [code points](#)) and other non-terminals. The basic



Preview

Code

Blame

Raw



```
# `root` specifies the pattern for the overall output
root ::= (
  # it must start with the characters "1. " followed by a sequence
  # of characters that match the `move` rule, followed by a space, followed
  # by another move, and then a newline
  "1. " move " " move "\n"

  # it's followed by one or more subsequent moves, numbered with one or two digits
  ([1-9] [0-9]? ". " move " " move "\n")+
)

# `move` is an abstract representation, which can be a pawn, nonpawn, or castle.
# The `[+#]` denotes the possibility of checking or mate signs after moves
move ::= (pawn | nonpawn | castle) [+#]?

pawn ::= ...
nonpawn ::= ...
castle ::= ...
```

```
200
209 while (*pos) {
210     if (*pos == '') { // literal string
211         pos++;
212         last_sym_start = out_elements.size();
213         while (*pos != '') {
214             if (!*pos) {
215                 throw std::runtime_error("unexpected end of input");
216             }
217             auto char_pair = parse_char(pos);
218             pos = char_pair.second;
219             out_elements.push_back({LLAMA_GRETYPE_CHAR, char_pair.first});
220         }
221         pos = parse_space(pos + 1, is_nested);
222     } else if (*pos == '[') { // char range(s)
223         pos++;
224         enum llama_gretype start_type = LLAMA_GRETYPE_CHAR;
225         if (*pos == '^') {
226             pos++;
227             start_type = LLAMA_GRETYPE_CHAR_NOT;
228         }
229         last_sym_start = out_elements.size();
230         while (*pos != ']') {
231             if (!*pos) {
232                 throw std::runtime_error("unexpected end of input");
233             }

```

```
294     if (*pos == '}') {
295         max_times = min_times;
296         pos = parse_space(pos + 1, is_nested);
297     } else if (*pos == ',') {
298         pos = parse_space(pos + 1, is_nested);
299
300         if (is_digit_char(*pos)) {
301             const char * int_end = parse_int(pos);
302             max_times = std::stoul(std::string(pos, int_end - pos));
303             pos = parse_space(int_end, is_nested);
304         }
305
306         if (*pos != '}') {
307             throw std::runtime_error(std::string("expecting '}' at ") + pos);
308         }
309         pos = parse_space(pos + 1, is_nested);
310     } else {
311         throw std::runtime_error(std::string("expecting ',' at ") + pos);
312     }
313     handle_repetitions(min_times, max_times);
314 } else {
315     break;
316 }
317 }
318 return pos;
319
```

Beyond Bitcoin & FORTH

Bitcoin Address: Hash of Public Key

Omnihash User ID: Hash of
Public Key

Phoscript: extension of FORTH
on host programming languages

Omni*Web: ecosystem built with
Omnihash & Phoscript

Omni: everything
*: everything

1. LLASMA (Large Language Stack Machine Architecture)
2. GBNF **Gerganov** [Backus-Naur Form \(BNF\)](#)
3. LLASMA to AGI in 10 years?
4. Other modules of Omni*Web
 - a. Omnihash + Contract
 - b. SVFIG Membership + Website
 - c. GRATIS: Grand Unified AI Token Trading Platform

LLASMA: LargeLanguage + StackMachine Architecture

- A. [Introduction](#)
- B. [How We Code It](#)
- C. [Install-Run-Test-Contribute](#)
- D. [Theories & Visions](#)

A. Introduction

LLM inference in pure C/C++ with embedded cxxforth stack machine + Microsoft BitNet support

LLASMA is a purpose-built fork of [llama.cpp](#) (on the `20260421` branch) that integrates a **trusted Forth-based stack machine** (via [cxxforth](#)) directly into the inference loop.



Code

Blame

Raw



```
839
840     std::string line;
841     bool another_line = true;
842     do {
843         another_line = console::readline(line, params.multiline_input);
844
845         // === NEW: Command Mode Detection ===
846         // Check if input starts with command prefix (e.g., '!')
847         if (!line.empty() && line[0] == '!') {
848
849             // Strip prefix and execute as Forth command
850             std::string command = line.substr(1); // remove '!'
851
852             // if (forth_vm.execute(command, ctx.get())) {
853             if (forth_vm.execute(command, ctx)) {
854                 // Optional: output result to user
855                 if (forth_vm.has_value()) {
856                     printf("\x1b[32m[Forth] Result: %.4g\x1b[0m\n", forth_vm.top_value());
857                 } else if (forth_vm.has_string()) {
858                     printf("\x1b[32m[Forth] Result: %s %d\x1b[0m\n", forth_vm.top_string(),
```

```

Activities Terminal Sab 25 Apr, 14:32 zh
hongwu@hongwu-Latitude-5480: ~/devel/2026/BitNet/llasforth/BitNet

== Running in interactive mode. ==
- Press Ctrl+C to interject at any time.
- Press Return to return control to the AI.
- To return control without starting a new line, end your input with '/'.
- If you want to submit another line, end your input with '\'.

System: You are a helpful assistant
> llama
Sure, how can I assist you with llamas today? Are you looking for information on them, perhaps their characteristics, care, or something else?

> !f f f f f
[Forth] Result: f 5

> !FORTH a b c d e
is FORTH !!cxforth: unrecognized word: included
[Forth] Result: e 11

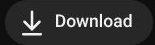
> llama
Llamas are domesticated South American camelids that are often used as pack animals and for their wool. Here are a few key points about llamas:

```

<https://www.youtube.com/watch?v=aA75-iMOtNE>

LLASMA Demo: Large Language + Stack Machine Architecture

M MI4 Metaprogramming in Forth
57 subscribers



- All
- For you
- Recently uploaded
- Watched



The Dark Pattern Behind Vanished Scientists EXPOSED!



```
(bitnet-cpp) hongwu@hongwu-Latitude-5480:~/devel/2026/BitNet/llasma/BitNet$ python setup_env.py
-md models/BitNet-b1.58-2B-4T -q i2_s
INFO:root:Compiling the code using CMake.
INFO:root:Loading model from directory models/BitNet-b1.58-2B-4T.
INFO:root:GGUF model already exists at models/BitNet-b1.58-2B-4T/ggml-model-i2_s.gguf
(bitnet-cpp) hongwu@hongwu-Latitude-5480:~/devel/2026/BitNet/llasma/BitNet$
```

Figure 4



```
(bitnet-cpp) hongwu@hongwu-Latitude-5480:~/devel/2026/BitNet/llasma/BitNet$ ./build/bin/llama-cli
-m models/BitNet-b1.58-2B-4T/ggml-model-i2_s.gguf -n 128 -t 2 -p 'You are a helpful assistant' -
ngl 0 -c 2048 --temp 0.8 -b 1 -cnv --log-file _/log_oxw_20260421_1256
```

Figure 5



```
hongwu@hongwu-Latitude-5480: ~/devel/2026/BitNet/llasma/BitNet
mirostat = 0, mirostat_lr = 0.100, mirostat_ent = 5.000
sampler chain: logits -> logit-bias -> penalties -> top-k -> tail-free -> typical -> top-p -> m
in-p -> temp-ext -> softmax -> dist
generate: n ctx = 2048, n batch = 1, n predict = 128, n keep = 1
```

0

```
mirostat = 0, mirostat_lr = 0.100, mirostat_ent = 5.000
```

```
sampler chain: logits -> logit-bias -> penalties -> top-k -> tail-free -> typical -> top-p -> m
```

```
in-p -> temp-ext -> softmax -> dist
```

```
generate: n_ctx = 2048, n_batch = 1, n_predict = 128, n_keep = 1
```

```
== Running in interactive mode. ==
```

```
- Press Ctrl+C to interject at any time.
```

```
- Press Return to return control to the AI.
```

```
- To return control without starting a new line, end your input with '/'.  
- If you want to submit another line, end your input with '\'.
```

```
System: You are a helpful assistant
```

```
> !FORTH f f f f f f f f
```

```
is FORTH !!cxxforth: unrecognized word: included
```

```
[Forth] Result: f 9
```

```
knowledgeable about electronics and technology.
```

```
defective_or_not(x):
```

```
    if x == "not defective":
```

```
        return 1
```

```
    else:
```

```
        return 0
```

```
def defective_or_not(x):
```



20260421

llama.cpp / common / CMakeLists.txt

↑ Top

Code

Blame

Raw



```
62     json.hpp
63     log.cpp
64     log.h
65     ngram-cache.cpp
66     ngram-cache.h
67     sampling.cpp
68     sampling.h
69     train.cpp
70     train.h
71     cxxforth.h
72     cxxforth.cpp
73     forth_vm.h
74     forth_vm.cpp
75 )
76
77 if (BUILD_SHARED_LIBS)
78     set_target_properties(${TARGET} PROPERTIES POSITION_INDEPENDENT_CODE ON)
79 endif()
80
```



20260421 ▾

llama.cpp / common / forth_vm.cpp

↑ Top

Code

Blame

Raw



```
35     bool ForthVM::execute(const std::string& command, llama_context* ctx) {
36         auto tokens = tokenize(command);
37
38         if (tokens[0]=="FORTH") {
39             std::cout << " is FORTH !!";
40             /* # Source - https://stackoverflow.com/a/19082779
41             # Posted by Dietmar Kühl, modified by community. See post 'Timeline' for change history
42             # Retrieved 2026-04-20, License - CC BY-SA 3.0 */
43             // std::string array[] = { "s1", "s2" };
44             std::vector<char*> vec;
45             std::transform(std::begin(tokens), std::end(tokens),
46                           std::back_inserter(vec),
47                           [](std::string& s){ s.push_back(0); return &s[0]; });
48             vec.push_back(nullptr);
49             char** carray = vec.data();
50
51             // forth(tokens.size(), command.c_str());
52             forth(tokens.size(), carray);
53         }
```

Why is FORTH unique for LLASMA?

1. FORTH expressions are simply space delimited tokens, which can be parsed by LLM as input or output data. → grounding for “action” or “operation” !!
2. FORTH shell in llama.cpp in turn may modify llama.cpp, or create parallel execution engines, optionally bypassing the original.
3. Other programming languages may perform the above, but with Phoscript as translation interface to other programming languages, the distinctions become trivial.

LLASMA to do

1. DJSON Decentralised JSON / Omnihash / JSON as UIPC message.
2. Second week of the month – team meeting / presentations for Omni*Web / LLASMA development.
3. Join us: <https://github.com/omnixtar/omnixtar.github.io/issues/5>
<https://omnixtar.github.io/h/>



Code

Issues 5

Pull requests

Discussions

Actions

Projects

Wiki

More

LLASMA Development Team #5

Edit

New issue



Open



omnixtar opened 1 minute ago · edited by omnixtar

Edits

Owner



LLASMA Development Team

Leave your Omnihash ID in the comment section of this issue.

Generate your Omnihash ID here: <https://omnixtar.github.io/h>

Create sub-issue



Assignees



No one - [Assign yourself](#)

Labels



No labels

Projects



No projects

Milestone



Self Awareness

Human: Self Identity → Ownership of Digital Assets

AI: Ownership of Digital Assets → Self Identity

Human: Ownership of space → Self Identity

- How does human acquire self identity?
- How may AI acquire self identity?

Conclusions: Omni*Web vs. AI Dystopia

AI taking away jobs

White House initiatives?

*Omni*Web: a truly decentralised web ecosystem OWNED and OPERATED by free individuals & free software programmers.*

Omnihash / DJSON (message)

I2P (router)

- Fear of AI taking away jobs

As of February 2026, the fear and threats of AI taking away jobs are real. But so far few have proposed credible solutions, even theoretically.

As such, as outlandish as Omnihash Omni*Web claims sound, ours are perhaps the only solutions worth listening and worth trying:

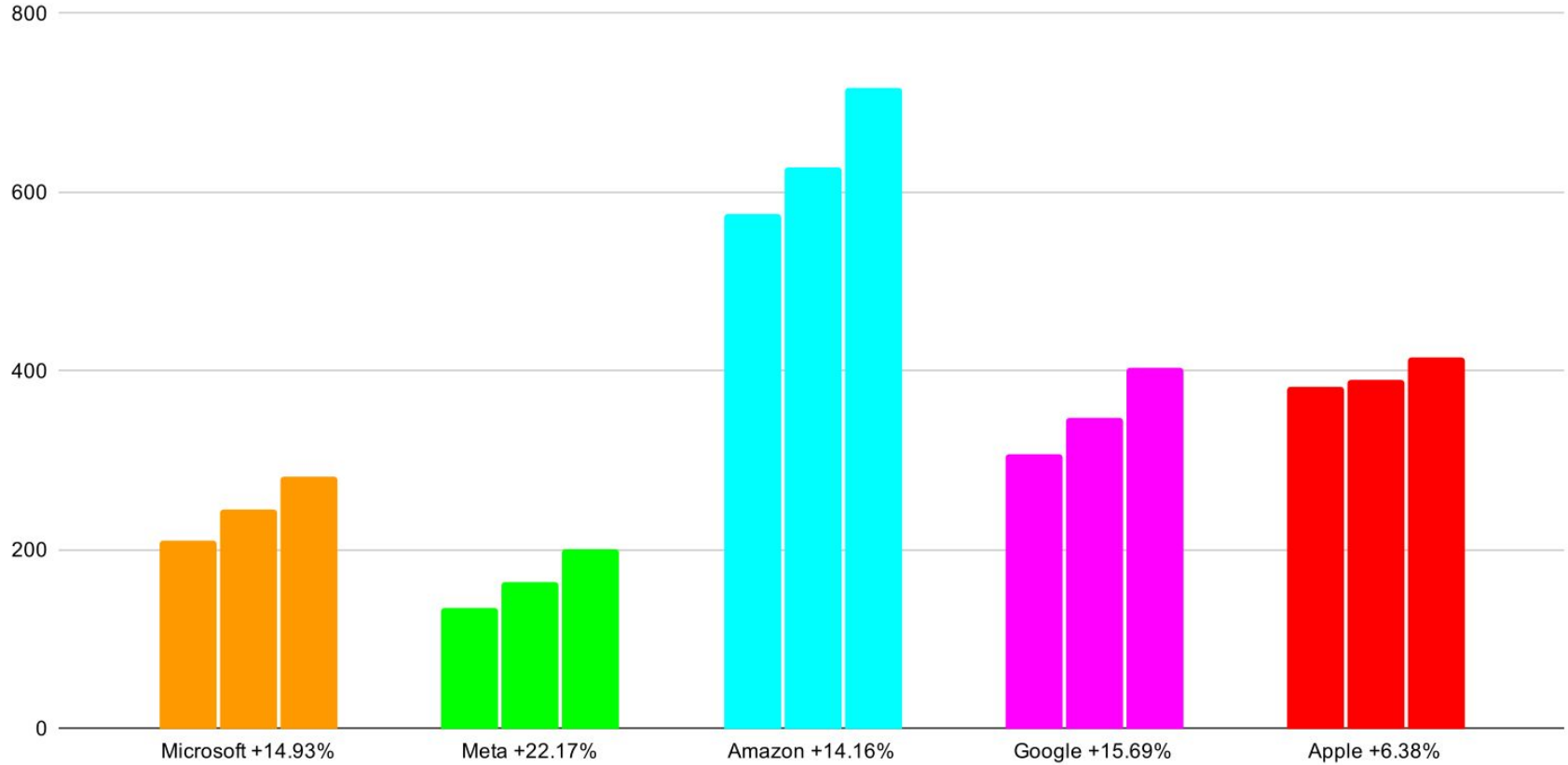
- Omni*Web: a truly decentralised web ecosystem OWNED and OPERATED by free individuals & free software programmers.

Otherwise, the AI dystopia is becoming worse day by day.

Source of power: from I2P Invisible Internet Project (2003) to Bitcoin (2009), Bitcoin Billionaires,

MMAGA Revenues 2025/24/23 (USD billions)

Total 2025: USD 2.018 Trillion (+13.6%)



Omniscientia

MMAGA Revenues 2025/24/23 (USD billions)

Total 2025: USD 2.018 Trillion (+13.6%)



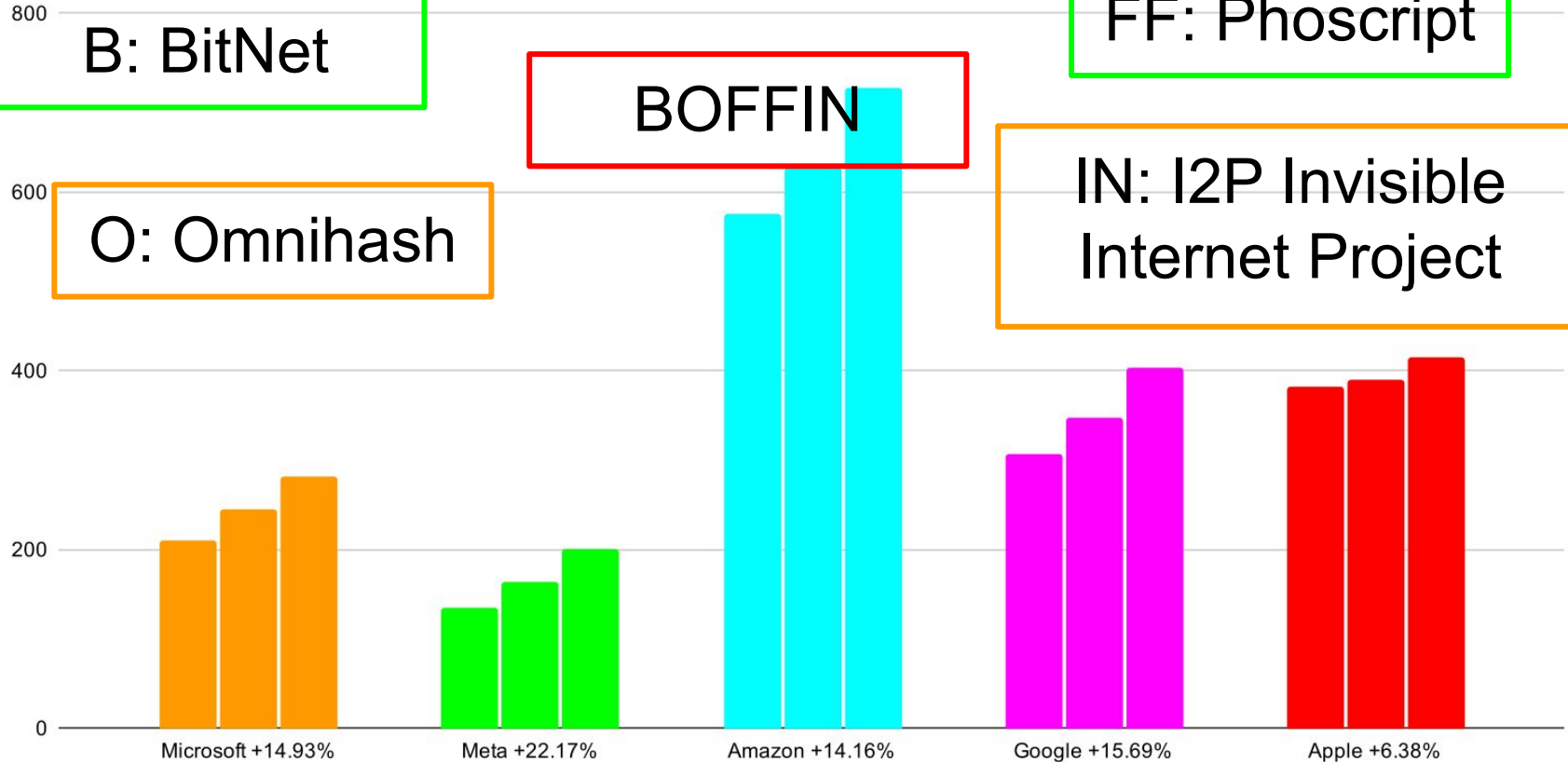
FF: Phoscript

B: BitNet

BOFFIN

O: Omnihash

IN: I2P Invisible Internet Project



Omnihash Revolutions

– No weapons, just codes

Governments vs. Programmers

Theory: Hilbert Hotel → Hash Numbers →
Ownership of Digital Assets

Metanarchy: global governance
based on decentralised
transaction in Metaverse

Omnihash / DJSON (message)

I2P (router)

Developed countries are failing – debt,
infrastructure, immigrants ...

- Theory: Hilbert Hotel → Hash Numbers
→ Ownership of Digital Assets

Programmer's roles: Omnihash as foundation
of :

- A. Finance: HXBC X=XAU Hash Gold
Bullion Coin
- B. Transparent transactions:
Administration, voting, eliminate
corruption
- C. Decentralised Web: fair rewards to
users and programmers
- D. Extend free software to other fields of
training
- E. A+B+C+D = Metanarchy

Omnihash Revolutions

– *No weapons, just codes*

Omni*Web (ecosystem)

Omniscientia (application)

iframe / selenium
(UIPC unified inter-process
communications)

Omnihash / DJSON (message)

I2P (router)

Today's Internet is built by free software programmers. But they are not fairly rewarded.

The future of Internet can also be decided by free software programmers.

We now have evidence of the magnitude of the rewards, and new mechanism such as Omnihash + DJSON to implement reward mechanisms.

With emergence of AI, Omni*Web has the following breakthroughs:

- A. Omniscientia: a platform to merge ALL AI knowledge, with search engine
- B. Explore AGI: Hash of Public Key as Self Identity
- C. Omnihash for trading GPU AI tokens – immediate trillion dollar application.