Beyond Bitcoin & FORTH (BBF005)

Cardinality of FORTH words, Inverse Turing Test and defining AGI

We define the cardinality of a FORTH stack machine word as *the number of primitive words used in its colon definition*, and this enables ALL SOURCE CODE having been written by human programmers or AI agents, after translated using Phoscript engine, to become *"enumerable" – thereby searchable, modularised and composable*. We also propose the *Inverse Turing Test (ITT)* and *a process to construct measurable and verifiable definitions* of Artificial General Intelligence based on the concepts above.

Liang Ng, June 2026
omnixtar.github.io/svfig
https://www.linkedin.com/in/liang-ng

Omni*Web + Omni*DOC

https://omnixtar.github.io/svfig/

SVFIG (Silicon Valley FORTH Interest Group)

- June 27 2026, BBF005 Cardinality of FORTH words, Inverse Turing Test and defining AGI
- May 23 2026, BBF004 H3A Hilbertian Hashed Homoiconic Augmentation
- April 25 2026, BBF003 FORTH everywhere – even in AI llama.cpp [YouTube]
- February 28 2026, BBF002 AGI Self-Identity Using Hash of Public Key & Reselling AI Server Tokens w. Omnihash [YouTube]
  - Grok summary
- January 24 2026, BBF001 Hilbert Hotel with Infinite Cabinets [YouTube]
  - January 24 2026, I3P: <iframe> + I2P Invisible Internet Project
- November 15 2025, The roles of FORTH in blockchain technologies and beyond (Summary & Updates) [YouTube]
- October 25 2025, The roles of FORTH in blockchain technologies and beyond [YouTube]
- July 2025, Crypto-Metaprogramming: alternative to Model-View-Controller Architecture [YouTube]

Cardinality of FORTH words

1. We define the cardinality of a FORTH stack machine word as the *number of primitive words used in its colon definition*, and this enables ALL SOURCE CODE having been written by human programmers or AI agents, after translated using Phoscript engine, to become “enumerable” – thereby searchable, modularised and composable.

2. We also propose the Inverse Turing Test (ITT) and

3. a process to construct measurable and verifiable definitions definition of Artificial General Intelligence based on the concepts above.

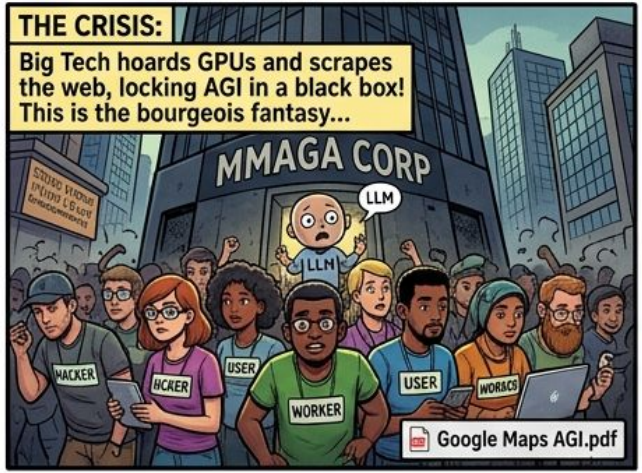
True AGI won’t emerge from text-scraping chatbots; it requires physical, spatial grounding. The first step toward a proletariat-owned AGI isn't building a better language model—it’s building a decentralized, open-source clone of Google Maps using Cesium.js.

This "Super Google Earth" serves as an infinite spatial sandbox. By pairing it with Omnihash, every digital coordinate becomes a provably owned asset, transforming users and programmers from exploited laborers into digital landlords. The map's logic is built on Phoscript, a FORTH-stack language. Through "FORTHification," all code is reduced to measurable mathematical sets (Cardinality), ensuring transparency and separating open disclosure from fair royalties.

As users interact with this globe, it trains LLASMA (Large Language & Stack Machine Architecture). Instead of the classic Turing Test, LLASMA faces the Inverse Turing Test: can it write and execute verifiable spatial logic? This creates a recursive loop where users update the world, the AI updates the code, and the code updates the world. By owning the spatial simulation, the working class owns the AGI that emerges from it—no weapons, just codes.

‘THE DIGITAL PROLETARIAT: BUILDING AGI, MAPPING OUR FATE’

THE CRISIS:
Big Tech hoards GPUs and scrapes the web, locking AGI in a black box! This is the bourgeois fantasy...

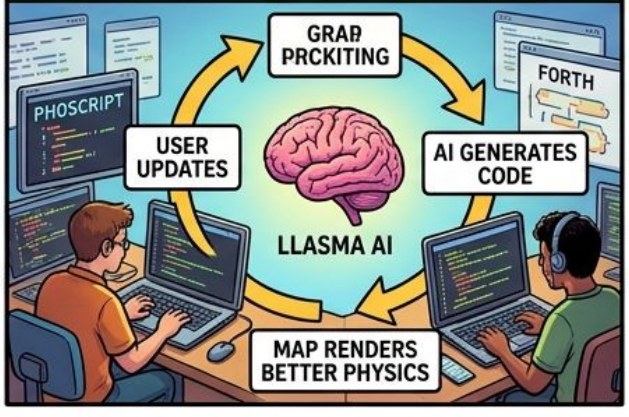


THE REAL PATH:
...But AGI isn't born from text! It starts with physics, space, and a room.
A human learns by navigating, not reading.

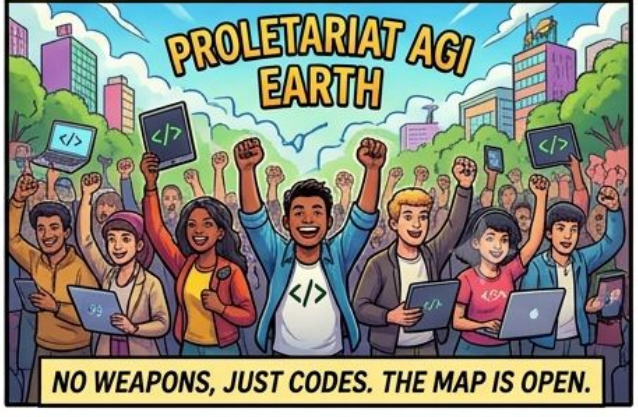
THE NEW MAP: WE BUILD OUR ROOM: A CLONED, OPEN GOOGLE EARTH!

Every coordinate is indexered with Omnihash.
Workers claim digital real estate! We are landlords!

RECURSIVE REVOLUTION: auditable, ownable!
The AI observes the map, writes code based on physics, and improves itself. The digital serfs become landlords!



OUT-BUILD THE FUTURE: A global army maps the world FORTH word at a time. Big Tech can't moat free software! Lay claim to the digital Earth, lay claim to the mind. Out-build the future!



Inverse Turing Test (ITT)

Turing Test: (essentially)

- Can a computer program (AI agent) talk like human beings?

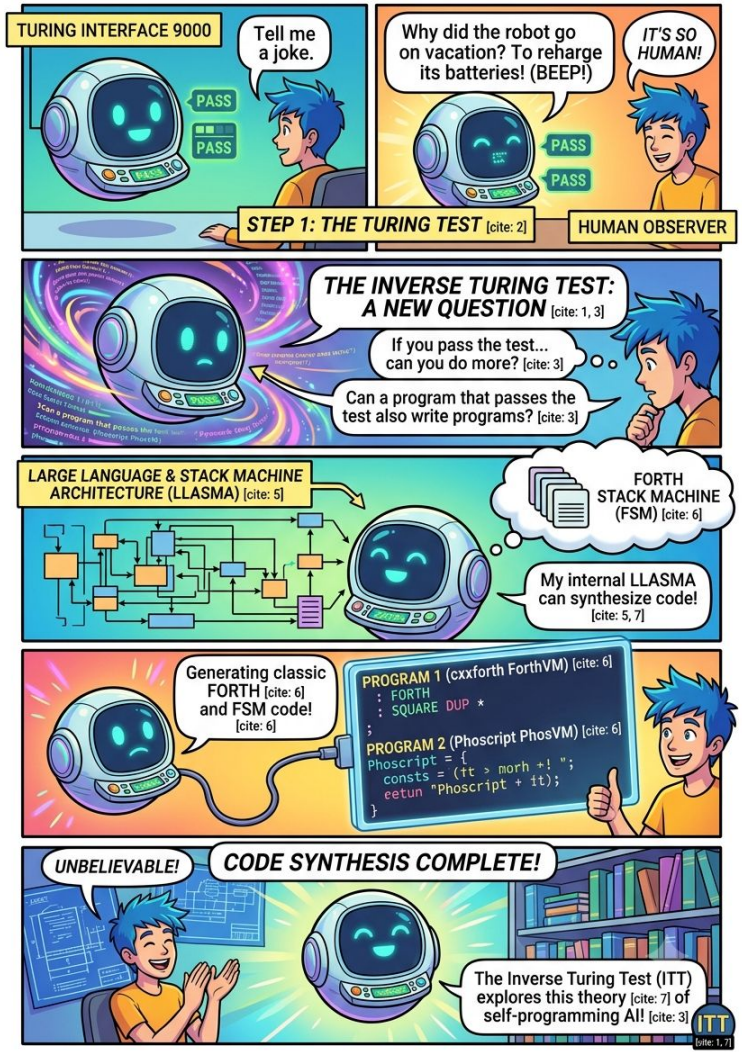
Inverse Turing Test: (one of many versions)

- Can a computer program that passes a Turing Test also write computer programs?

(Start with simple / general definition. Improve later.)
(ITT: a theory to cover LLASMA ?)

Write computer programs:
LLASMA (Large Language & Stack Machine Architecture)

a) Compose “classic” FORTH programs (cxxforth ForthVM)
b) Compose “FORTH stack machine” (FSM) programs e.g. Phoscript PhosVM



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

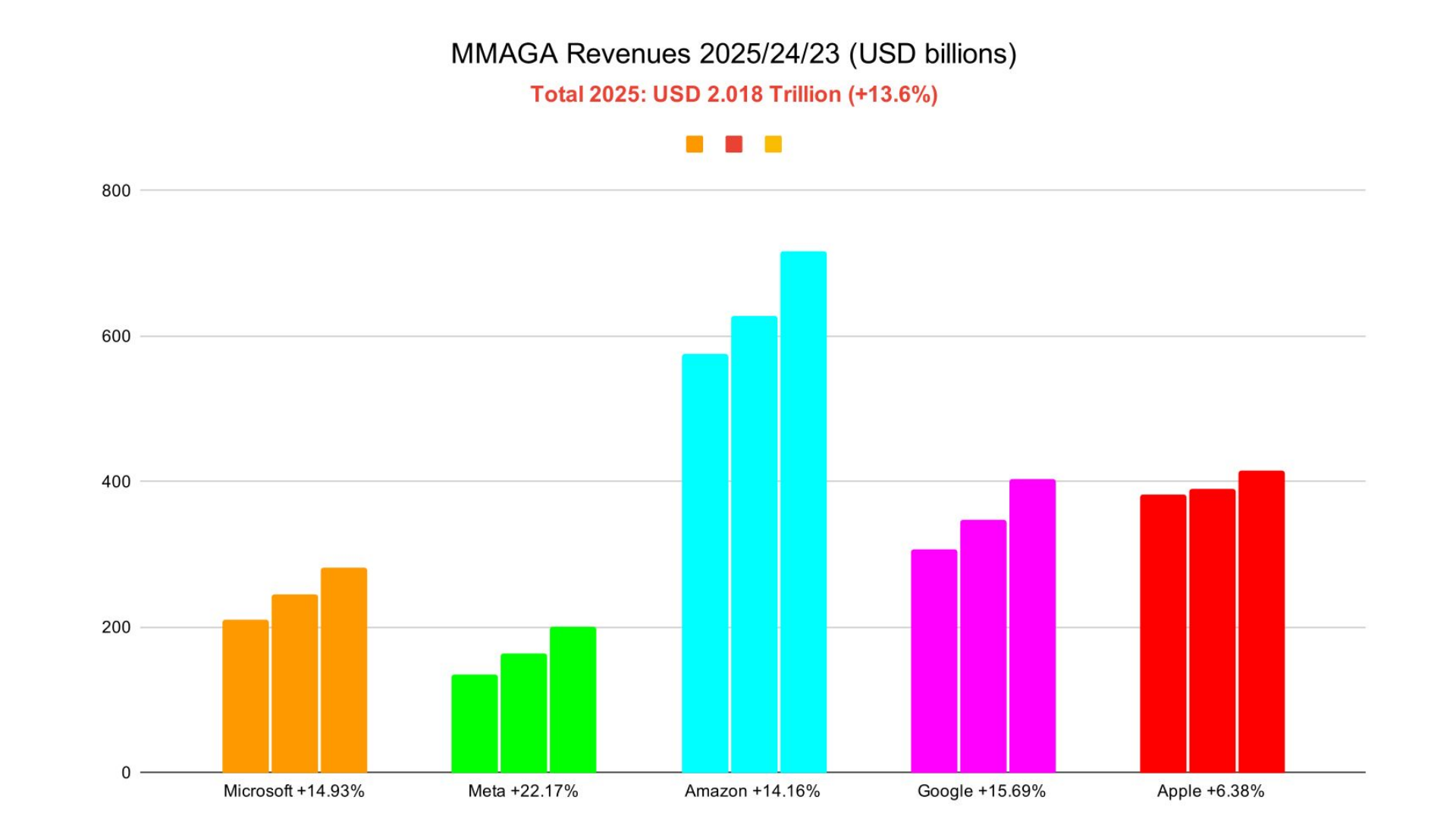
Omni: everything
*: everything

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;

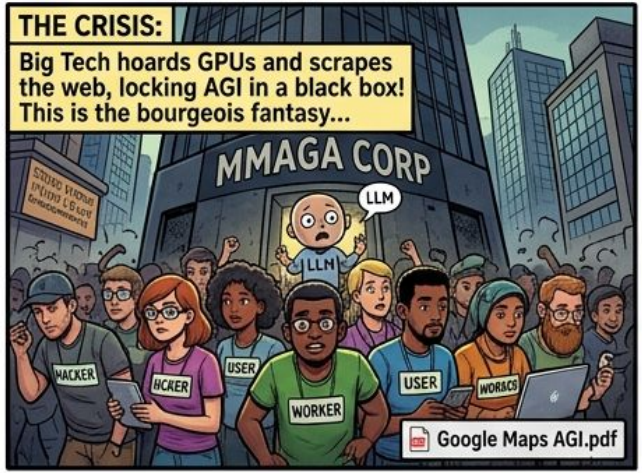
A. the hash of public key has been extended to Omnihash, a term to describe hashcode data structures for representing ANY kind of digital assets, including their ownerships, hence the prefix “Omni”.

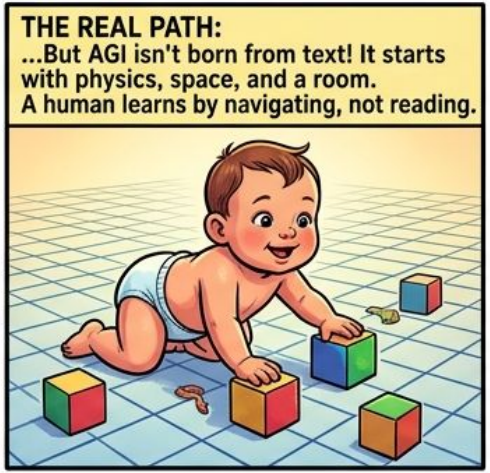
B. our variant of FORTH, Phoscript, has been deployed in various environments, including mobile and web front ends, PHP back end and Python Selenium web control modules.

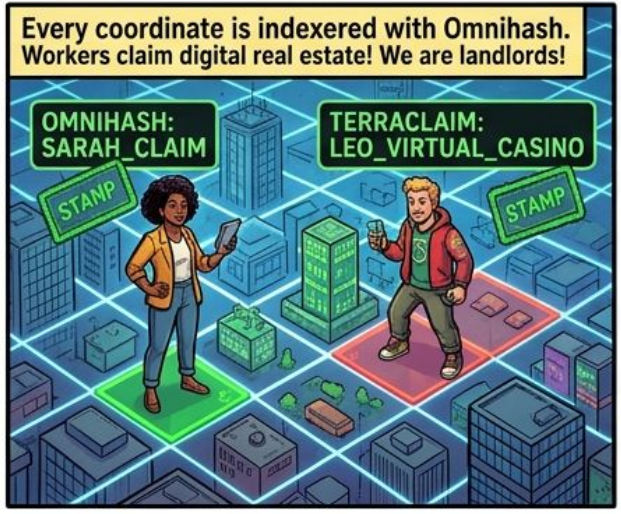
- extensions of Omnihash and Phoscript will be explored.

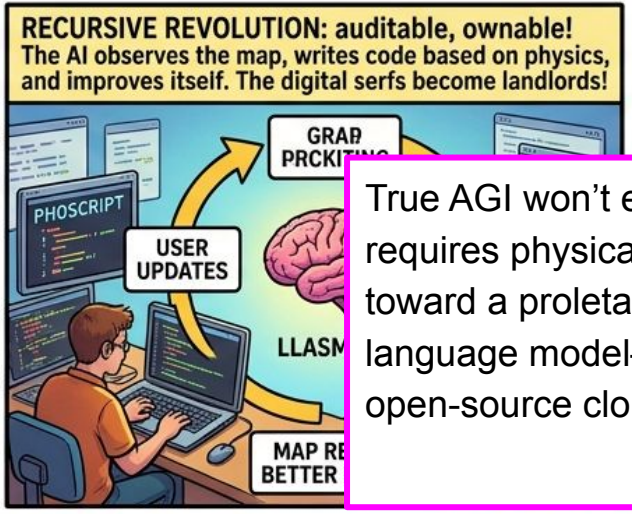

MMAGA Revenues 2025/24/23 (USD billions)
Total 2025: USD 2.018 Trillion (+13.6%)
800
600
400
200
0
Microsoft +14.93%
Meta +22.17%
Amazon +14.16%
Google +15.69%
Apple +6.38%

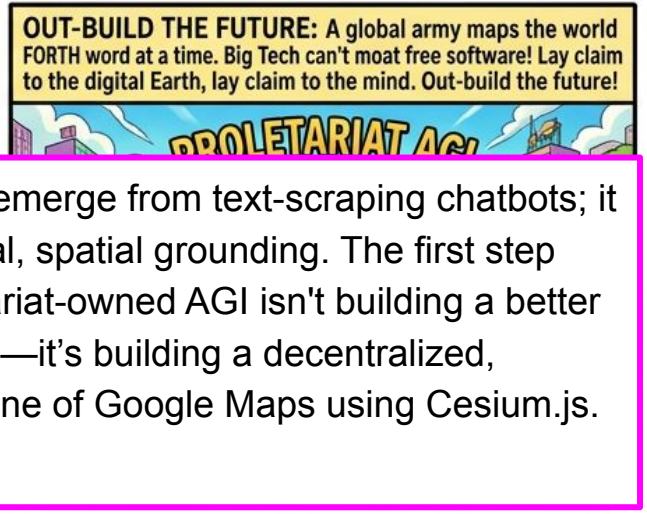
‘THE DIGITAL PROLETARIAT: BUILDING AGI, MAPPING OUR FATE’







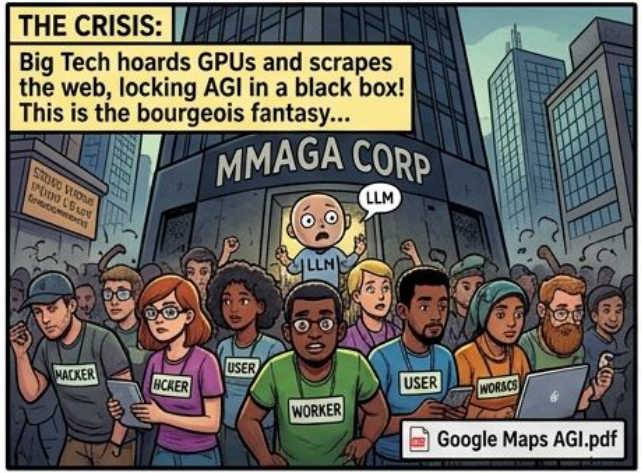


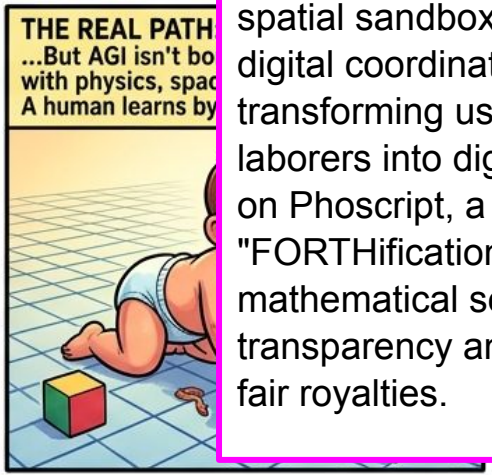


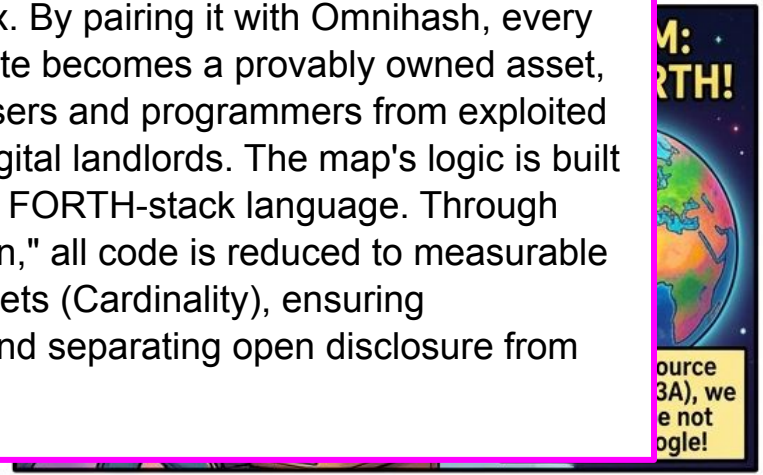
True AGI won’t emerge from text-scraping chatbots; it requires physical, spatial grounding. The first step toward a proletariat-owned AGI isn't building a better language model—it’s building a decentralized, open-source clone of Google Maps using Cesium.js.

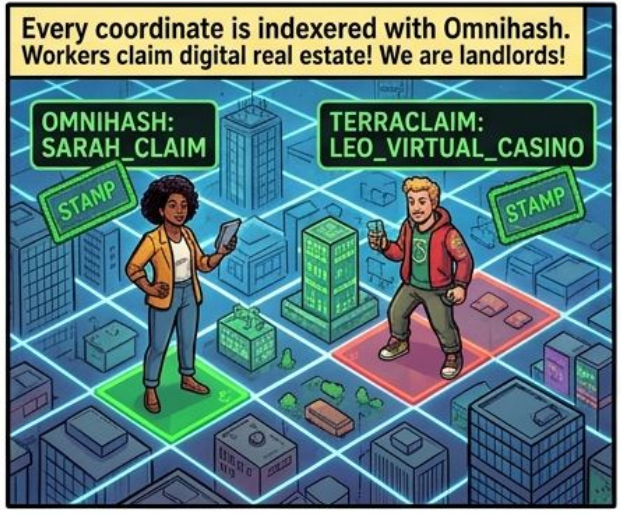
‘THE DIGITAL PROLETARIAT: E ATE’

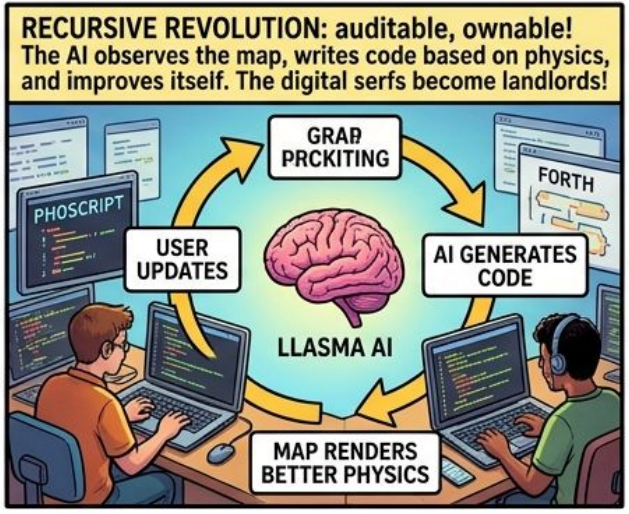
This "Super Google Earth" serves as an infinite spatial sandbox. By pairing it with Omnihash, every digital coordinate becomes a provably owned asset, transforming users and programmers from exploited laborers into digital landlords. The map's logic is built on Phoscript, a FORTH-stack language. Through "FORTHification," all code is reduced to measurable mathematical sets (Cardinality), ensuring transparency and separating open disclosure from fair royalties.

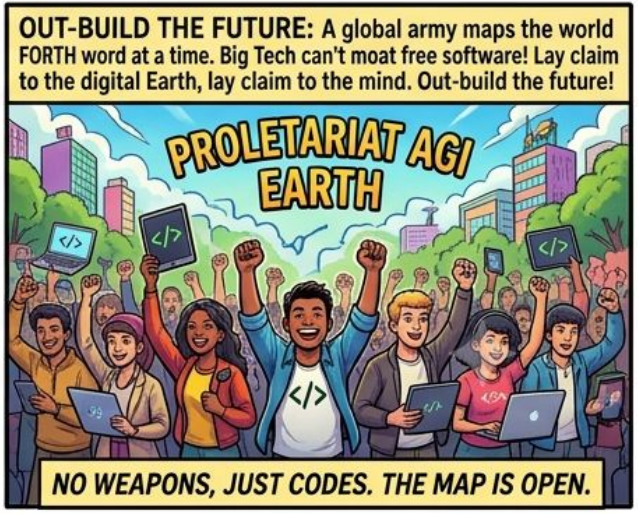




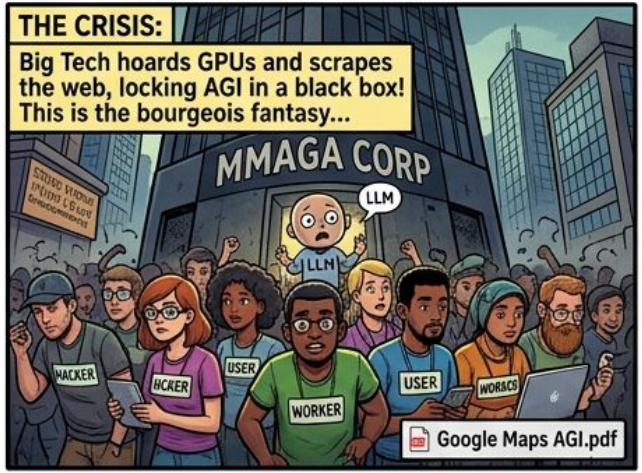


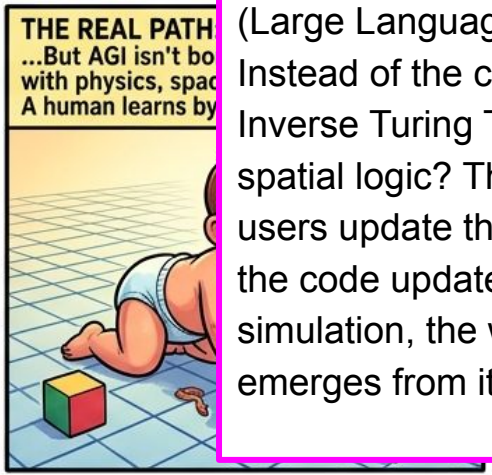


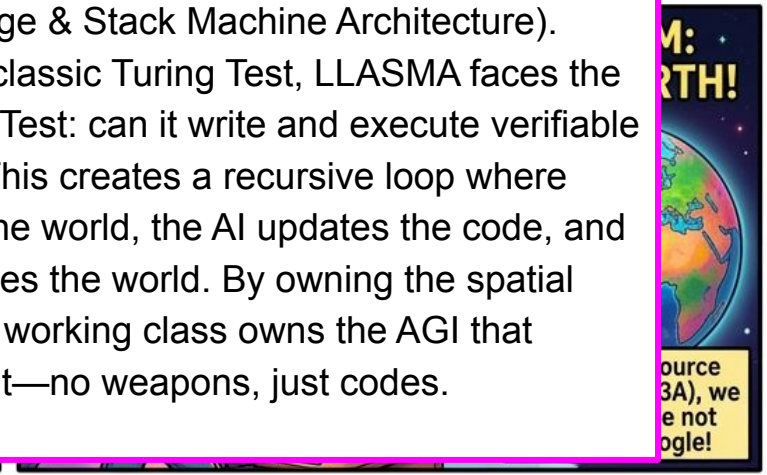




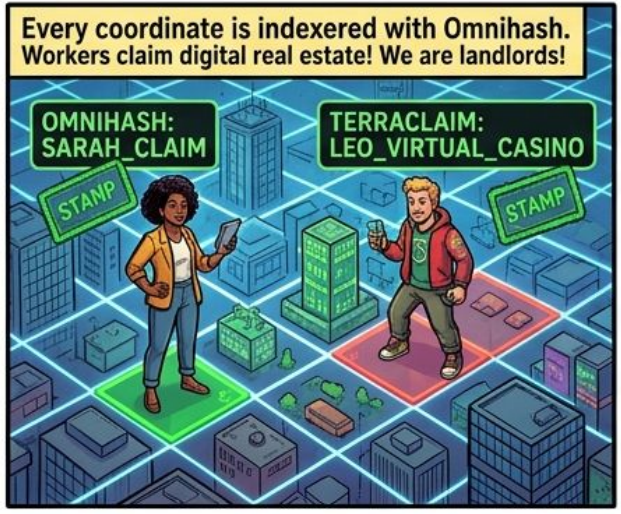
‘THE DIGITAL PROLETARIAT: E … TE’

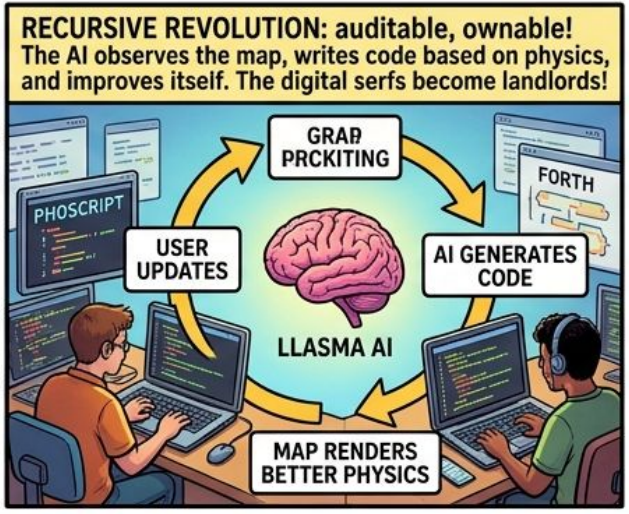


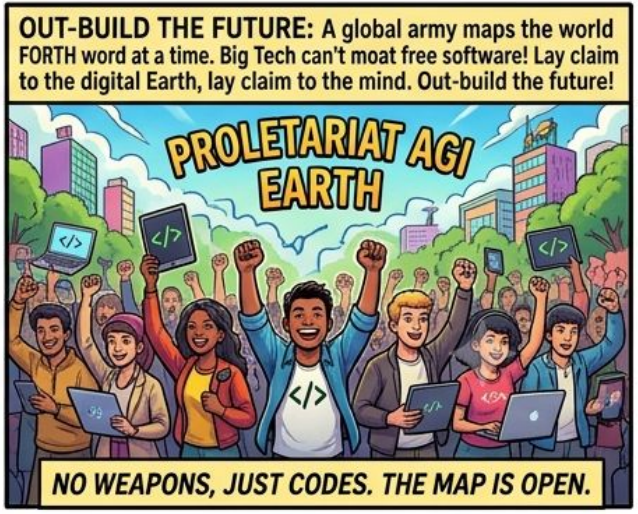




As users interact with this globe, it trains LLASMA (Large Language & Stack Machine Architecture). Instead of the classic Turing Test, LLASMA faces the Inverse Turing Test: can it write and execute verifiable spatial logic? This creates a recursive loop where users update the world, the AI updates the code, and the code updates the world. By owning the spatial simulation, the working class owns the AGI that emerges from it—no weapons, just codes.





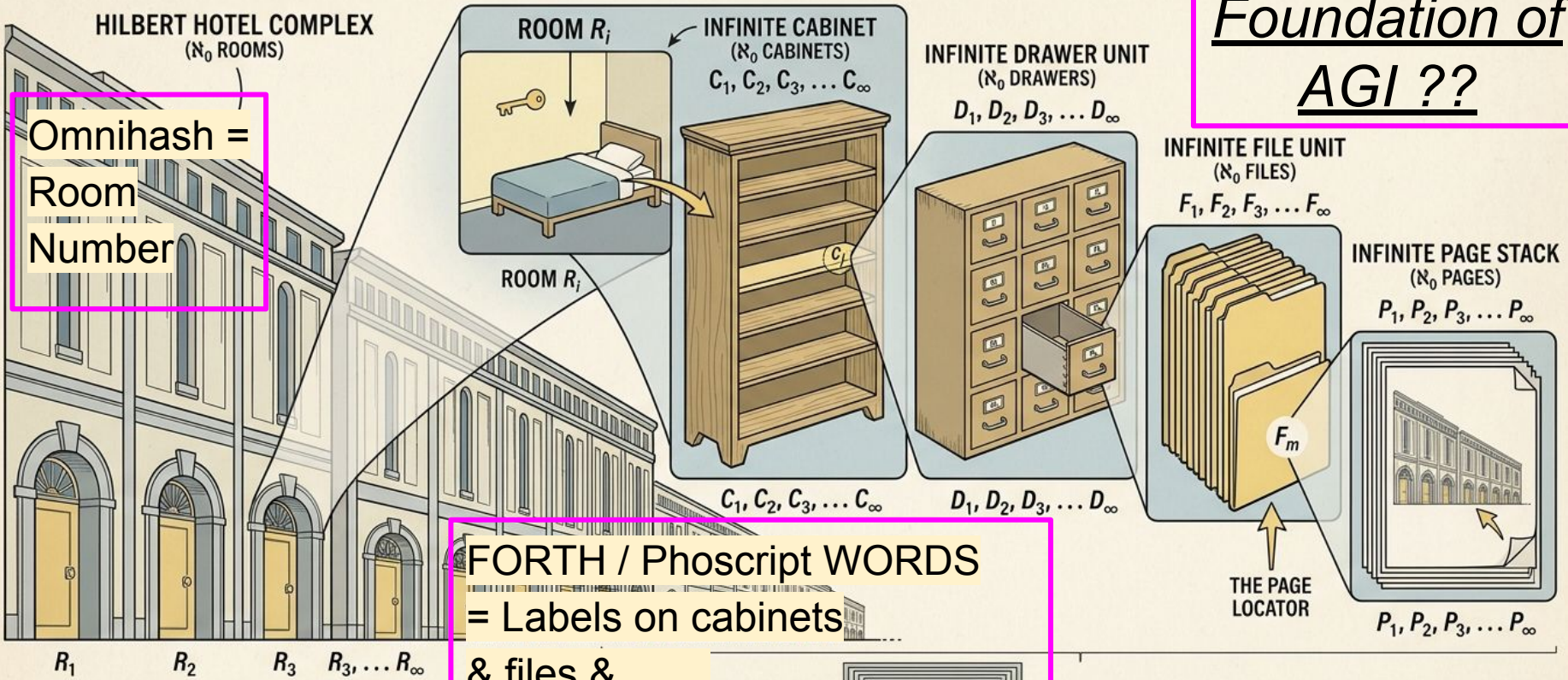


A. 10 minutes: FORTHification Hilbert Hotel
   a. Recursive Hilbert Hotel. Omnihash room number. FORTH word label for infinite cabinets.
   b. Cardinality of forth words is the highlight – make all source code searchable, modularised, composable – need phoscript.
   c. Set theory of AGI applies to non forth systems too, just need to be equivalent.
B. 10 minutes LLASMA Latest updates
   a. Show recursion: Phos word calling execute() and Colon Definition Words.
C. 10 minutes: AGI Set theory. Expansion Plan.
   a. Omnihash & Phoscript: new way of doing startup and free software,
   b. https://github.com/omnixtar/omnixtar.github.io/blob/main/js/omni.js line 73 Phoscript, empower senior programmers as well as beginners to collaborate across platforms and programming languages https://www.youtube.com/watch?v=mYjKS0KiJVg
   c. https://omnixtar.github.io/contract/ Omnihash, enabling ownership of digital assets, including source code, thus monetised as capital for building startup businesses.

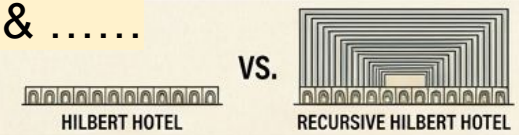
THE RECURSIVE HILBERT HOTEL

Foundation of AGI ??

Omnihash = Room Number



FORTH / Phoscript WORDS = Labels on cabinets & files & ……



THE CARTESIAN PRODUCT $\mathbb{N}^5$ IS STILL COUNTABLE.

Cardinality of FORTH words

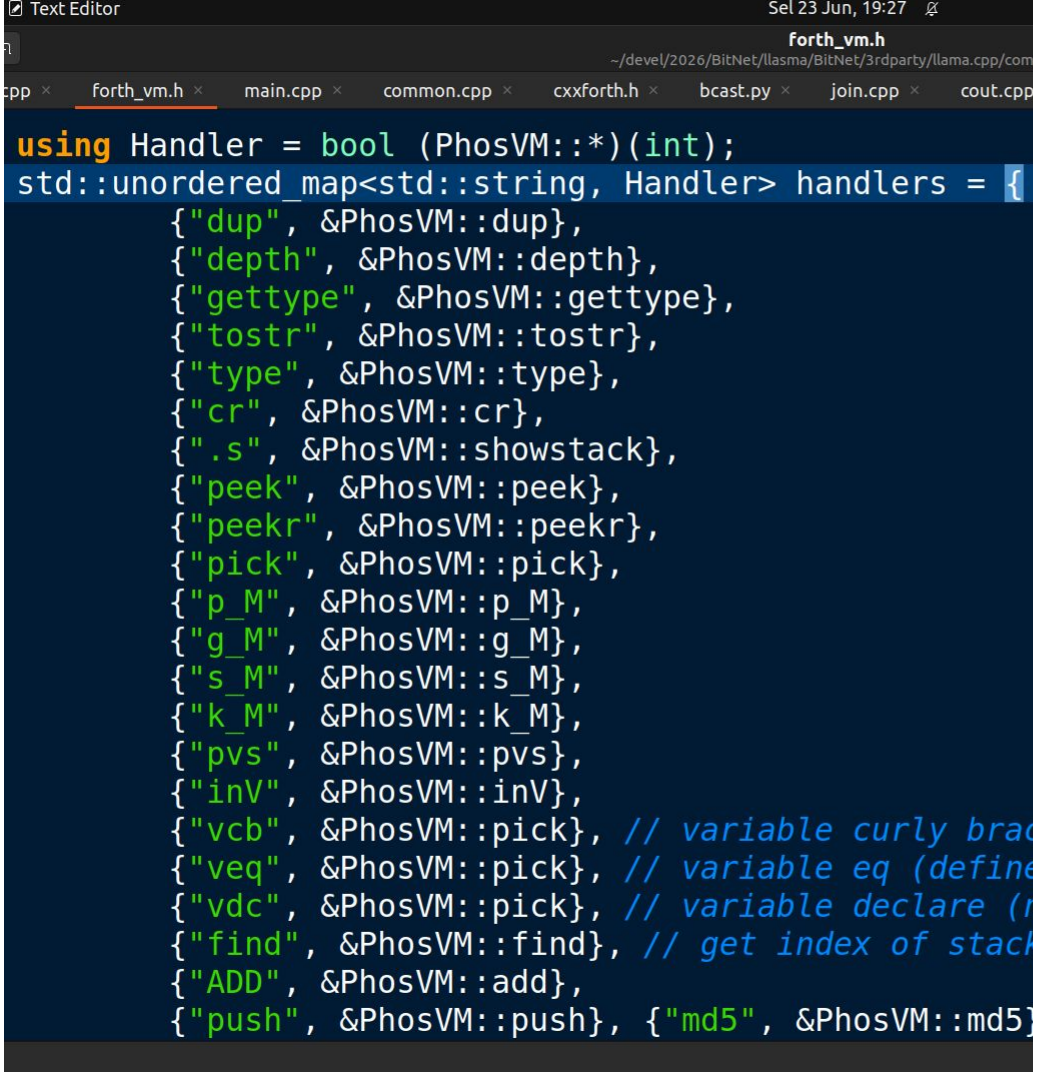
1. We define the cardinality of a FORTH stack machine word as the *number of primitive words used in its colon definition*, and this enables ALL SOURCE CODE having been written by human programmers or AI agents, after translated using Phoscript engine, to become “enumerable” – thereby searchable, modularised and composable.

FORTH words: words from “classic” FORTH
FORTH stack machine (FSM) words: from FORTH “derivatives” like Phoscript

Cardinality: bridge between abstract mathematics and computer programming

Cardinality of FORTH words

1. We define the cardinality of a FORTH stack machine word as the *number of primitive words used in its colon definition,* and this enables ALL SOURCE CODE having been written by human programmers or AI agents, after translated using Phoscript engine, to become “enumerable” – thereby searchable, modularised and composable.

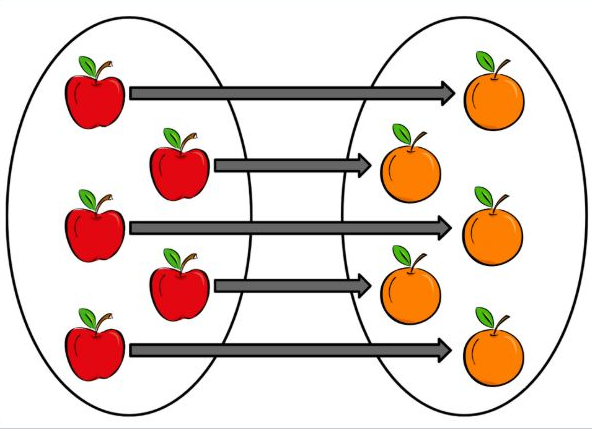


Cardinality

From Wikipedia, the free encyclopedia

For other uses, see Cardinality (disambiguation).

In mathematics, **cardinality** is an inherent property of sets, roughly meaning the number of individual objects they contain, which may be infinite. The concept is understood through one-to-one correspondences between sets. That is, if their objects can be paired such that each object has a pair, and no object is paired more than once.



A one-to-one correspondence between a set of apples and a set of oranges shows they have the same cardinality.

Two sets are said to be ***equinumerous*** or *have the same cardinality* if there exists a one-to-one correspondence between them. Otherwise, under the axiom of choice, one of the two sets must be equinumerous with a strict subset of the other and is said to be *strictly smaller* than it; the other set is *strictly larger*. Using this concept, it is possible to show there are different sizes of infinity.

A set is countably infinite if it can be placed in one-to-one correspondence with the set of natural numbers $\{1, 2, 3, 4, \cdots\}$. For example, the set of even numbers $\{2, 4, 6, \cdots\}$ and the set of rational numbers are countable. Uncountable sets are those strictly larger than the set of natural numbers. The set of all real numbers and the powerset of the set of natural numbers are proven to be uncountable by so-called diagonal arguments. Cantor's theorem generalizes these arguments to show there is an infinite hierarchy of infinities.

FORTHification (FCN)

Converting source code written in ANY programming language into FORTH stack machine (FSM) words using Phoscript engine (or equivalent).

FCN, Hex C = 12 = FORTHification = 12 letters between F & N
– like Internationalisation I18N

F(N) = set of FSM words of cardinality N

F(N) = source code of all programs ever written if N → ∞ (infinity)

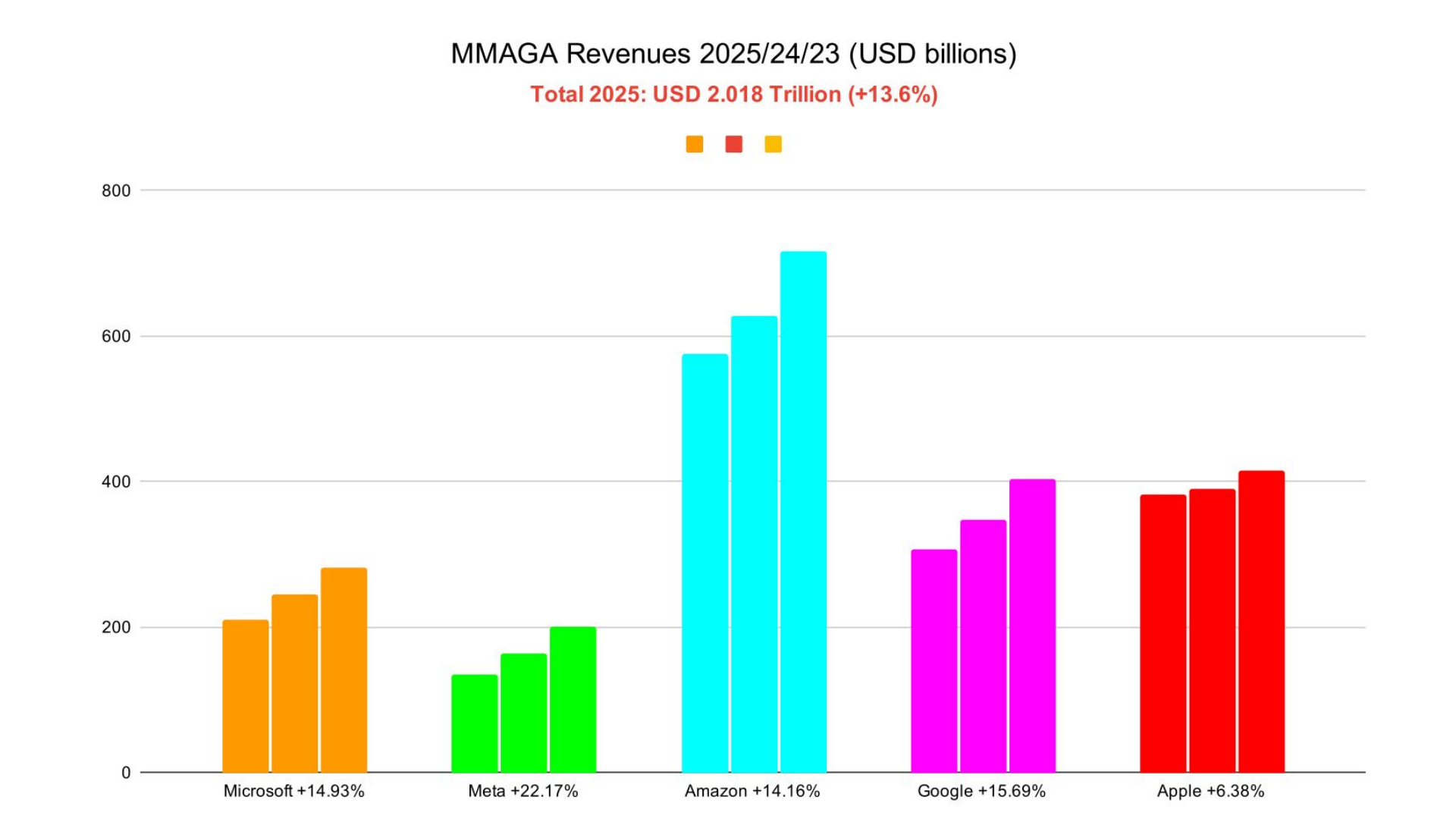
F(N) increases over time.

Why do this?

MMAGA (Microsoft, Meta, Amazon, Google, Apple)
2025 revenues > USD 2 trillion
And growing at 15% annually
(next slide)

"Dead Programmers Crisis"

If this generation of the ORIGINAL free software authors (programmers) die WITHOUT modifying free software licenses using Omnihash – their source code becomes PERMANENTLY free for MMAGA.


MMAGA Revenues 2025/24/23 (USD billions)
Total 2025: USD 2.018 Trillion (+13.6%)
800
600
400
200
0
Microsoft +14.93%
Meta +22.17%
Amazon +14.16%
Google +15.69%
Apple +6.38%

README Contributing License Security

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.

https://github.com/llasma/llama.cpp

https://github.com/llasma/llama.cpp/tree/20260421/CHANGES

```
python setup_env.py -md models/BitNet-b1.58-2B-4T -q i2_s

./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_lsm_20260623_1534 -v
```

```
 2599  python setup_env.py -md models/BitNet-b1.58-2B-4T -q i2_s
 2600  ./build/bin/llama-cli -m models/BitNet-b1.58-2B-4T/ggml-model-i2_s.gguf -n 128 -t 2 -p 'You a
re a helpful assistant' -ngl 0 -c 2048 --temp 0.8 -b 1 -cnv --log-file _/log_lsm_20260623_1534 -v
 2601  history
(base) hongwu@hongwu-Latitude-5480:~/devel/2026/BitNet/llasma/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 '\'.

embd_inp.size(): 8, n_consumed: 0
embd_inp.size(): 8, n_consumed: 1
Systemembd_inp.size(): 8, n_consumed: 2
:embd_inp.size(): 8, n_consumed: 3
 Youembd_inp.size(): 8, n_consumed: 4
 areembd_inp.size(): 8, n_consumed: 5
 aembd_inp.size(): 8, n_consumed: 6
 helpfulembd_inp.size(): 8, n_consumed: 7
 assistantwaiting for user input

> moscow

s_M m DEFINED
type_index is std::string! string key is test
  s0 555
  s_M m ASSIGNED

s_M m DEFINED
```

User enter “moscow”

Line 933 readline()

```
929
930             std::string line;
931             bool another_line = true;
932             do {
933                 another_line = console::readline(line, params.multiline_input);
934
935                 // need to use s_M pointer method
936                 // M["model"] = model;
937                 phos.execute("555 test s_M",0);
938                 S.push(model); phos.execute("model s_M",0);
939                 S.push(chat_msgs); phos.execute("chat_msgs s_M",0);
940                 phos.execute("user role s_M",0);
941                 S.push(std::move(buffer)); phos.execute("content s_M",0);
942                 S.push(g_params); phos.execute("g_params s_M",0);
943
944                 // static std::string chat_add_and_format(struct llama_model * model, std::
vector<common_chat_msg> & chat_msgs, const std::string & role, const std::string & content) {
945                 // chat_add_and_format(model, chat_msgs, "user", std::move(buffer))
946                 // M["chat_msgs"] = chat_msgs;
947
948                 // === NEW: Command Mode Detection ===
949                 // Check if input starts with command prefix (e.g., '!')
950                 if (!line.empty() && line[0] == '!') {
951
```

```
166 using namespace std;
167
168 std::stack<std::any> S; // "any stack": stack for object / variable of any type
169 std::unordered_map<std::string, std::any> M; // "any map": JavaScript "object" for "any type of
    variable"
170 unordered_map<string,vector<string>> C; // colon definition words
171
172 ForthVM forth_vm;   // cxxforth  VM
173 PhosVM  phos;       // Phoscript VM
174
175 void phosinit() {
176     std::vector<std::string> fruits = {"Apple", "Banana", "Cherry"};
177     C["fruits"] = fruits;
178
179     std::vector<std::string> add_one = {"1", "add"};
180     C["1+"] = add_one;
181
182     M["C"] = C; // must init before push !!
183     M["LR"] = 0; // recursion level
184     M["LS"] = 0; // token to skip, for control words: ifte (if then else) etc ....
185     M["DBG"] = true; // debug flag
186     S.push(M); // S[0] = M
187 }
188
189 #define l dbg if (DBG M)
```

```
187 }
188
189 #define l_dbg if (DBG_M)
190 bool DBG_M=false;
191
192 int main(int argc, char ** argv) {
193     common_params params;
194     g_params = &params;
195
196     phosinit();
197
198     if (!common_params_parse(argc, argv, params, LLAMA_EXAMPLE_MAIN, print_usage)) {
199         return 1;
200     }
201
202     common_init();
203
204     auto & sparams = params.sparams;
205
206     // save choice to use color for later
207     // (note for later: this is a slightly awkward choice)
208     console::init(params.simple_io, params.use_color);
209     atexit([]() { console::cleanup(); });
210
211     if (params.logits_all) {
```

```
901         if (params.conversation) {
902             const auto id = common_sampler_last(smpl);
903             assistant_ss << common_token_to_piece(ctx, id, false);
904         }
905
906         if (n_past > 0 && is_interacting) {
907             LOG_DBG("waiting for user input\n");
908
909             // look for LOG before this line, nad params.conversation flag
910             // LSM: LOG displays prompt > for user input
911             if (params.conversation) {
912                 LOG("\n> ");
913             }
914
915             if (params.input_prefix_bos) {
916                 LOG_DBG("adding input prefix BOS token\n");
917                 embd_inp.push_back(llama_token_bos(model));
918             }
919
920             std::string buffer;
921             if (!params.input_prefix.empty() && !params.conversation) {
922                 LOG_DBG("appending input prefix: '%s'\n", params.input_prefix.c_str());
923                 LOG("%s", params.input_prefix.c_str());
924             }
```

```
> moscow

s_M m DEFINED
type_index is std::string! string key is test
  s0 555
  s_M m ASSIGNED

s_M m DEFINED
llama_model key is model

s_M m DEFINED
vector common_chat_msg key is ...chat_msgs

s_M m DEFINED
type_index is std::string! string key is role
  s0 user
  s_M m ASSIGNED

s_M m DEFINED
type_index is std::string! string key is content
  s0
  s_M m ASSIGNED

s_M m DEFINED
common_params* key is g_params
```

```
929
930             std::string line;
931             bool another_line = true;
932             do {
933                 another_line = console::readline(line, params.multiline_input);
934
935                 // need to use s_M pointer method
936                 // M["model"] = model;
937                 phos.execute("555 test s_M",0);
938                 S.push(model); phos.execute("model s_M",0);
939                 S.push(chat_msgs); phos.execute("chat_msgs s_M",0);
940                 phos.execute("user role s_M",0);
941                 S.push(std::move(buffer)); phos.execute("content s_M",0);
942                 S.push(g_params); phos.execute("g_params s_M",0);
943
944                 // static std::string chat_add_and_format(struct llama_model * model, std::
vector<common_chat_msg> & chat_msgs, const std::string & role, const std::string & content) {
945                 // chat_add_and_format(model, chat_msgs, "user", std::move(buffer))
946                 // M["chat_msgs"] = chat_msgs;
947
948                 // === NEW: Command Mode Detection ===
949                 // Check if input starts with command prefix (e.g., '!')
950                 if (!line.empty() && line[0] == '!') {
951
```

```
common_params* key is g_params
[LLASMA] moscow
 / moscow
 /
buffer: 'moscow
'

formatted: 'User: moscow
<|eot_id|>Assistant: '
input tokens: [ 'User':1502, ':':25, ' mos':23518, 'cssistant':72803, ':':25, ' ':220 ]
n_remain: 119
embd_inp.size(): 17, n_consumed: 8
embd_inp.size(): 17, n_consumed: 9
embd_inp.size(): 17, n_consumed: 10
embd_inp.size(): 17, n_consumed: 11
embd_inp.size(): 17, n_consumed: 12
embd_inp.size(): 17, n_consumed: 13
embd_inp.size(): 17, n_consumed: 14
embd_inp.size(): 17, n_consumed: 15
embd_inp.size(): 17, n_consumed: 16
Moscow, the capital city of Russia, is one of the largest and most populous cities in Europe. Here are some interesting facts about Moscow:
```

line 791 token by token output in sequence, with short pauses between tokens, slow on CPU, very fast on GPU.

```
1. **History**: Moscow has a rich history that dates back to 1147 when it was founded by Prince Yuri
 Dolgorukhin. It grew from a small medieval town into a major political and cultural center of Russi
a.

2. **Architecture**: The city is home to numerous landmarks, including the Kremlin, Saint Basil's Ca
thedral, the GUM department store, and the Bolshoi Theatre.

3. **Climate**: Moscow has a humidembd_inp.size(): 17, n_consumed: 17
waiting for user input

>

s_M m DEFINED
type_index is std::string! string key is test
  s0 555
  s_M m ASSIGNED

s_M m DEFINED
llama_model key is model

s_M m DEFINED
vector common_chat_msg key is ...chat_msgs

s_M m DEFINED
```

```
785
786         if (input_echo && display) {
787             for (auto id : embd) {
788                 const std::string token_str = common_token_to_piece(ctx, id, params.special);
789
790                 // Console/Stream Output
791                 LOG("%s", token_str.c_str());
792                 // LOG(">%s_#", token_str.c_str()); // token_str has leading space !!
793
794                 // Record Displayed Tokens To Log
795                 // Note: Generated tokens are created one by one hence this check
796                 if (embd.size() > 1) {
797                     // Incoming Requested Tokens
798                     input_tokens.push_back(id);
799                 } else {
800                     // Outgoing Generated Tokens
801                     output_tokens.push_back(id);
802                     output_ss << token_str; // LSM: is this line that output token to terminal ?
?
803                     // output_ss << token_str << "#LSM#"; // test
804                 }
805             }
806         }
807
```

```
>

s_M m DEFINED
type_index is std::string! string key is test
  s0 555
  s_M m ASSIGNED

s_M m DEFINED
llama_model key is model

s_M m DEFINED
vector common_chat_msg key is ...chat_msgs

s_M m DEFINED
type_index is std::string! string key is role
  s0 user
  s_M m ASSIGNED

s_M m DEFINED
type_index is std::string! string key is content
  s0
  s_M m ASSIGNED

s_M m DEFINED
common_params* key is g_params
```

```
Would you like more detailed information on any of these points?

found an EOG token

s_M m DEFINED
llama_model key is model

s_M m DEFINED
vector common_chat_msg key is ...chat_msgs

s_M m DEFINED
type_index is std::string! string key is role
  s0 assistant
  s_M m ASSIGNED

formatted: ' Moscow, the capital city of Russia, is one of the largest and most populous cities in E
urope. Here are some interesting facts about Moscow:

1. **History**: Moscow has a rich history that dates back to 1147 when it was founded by Prince Yuri
 Dolgorukhin. It grew from a small medieval town into a major political and cultural center of Russi
a.

2. **Architecture**: The city is home to numerous landmarks, including the Kremlin, Saint Basil's Ca
thedral, the GUM department store, and the Bolshoi Theatre.
```

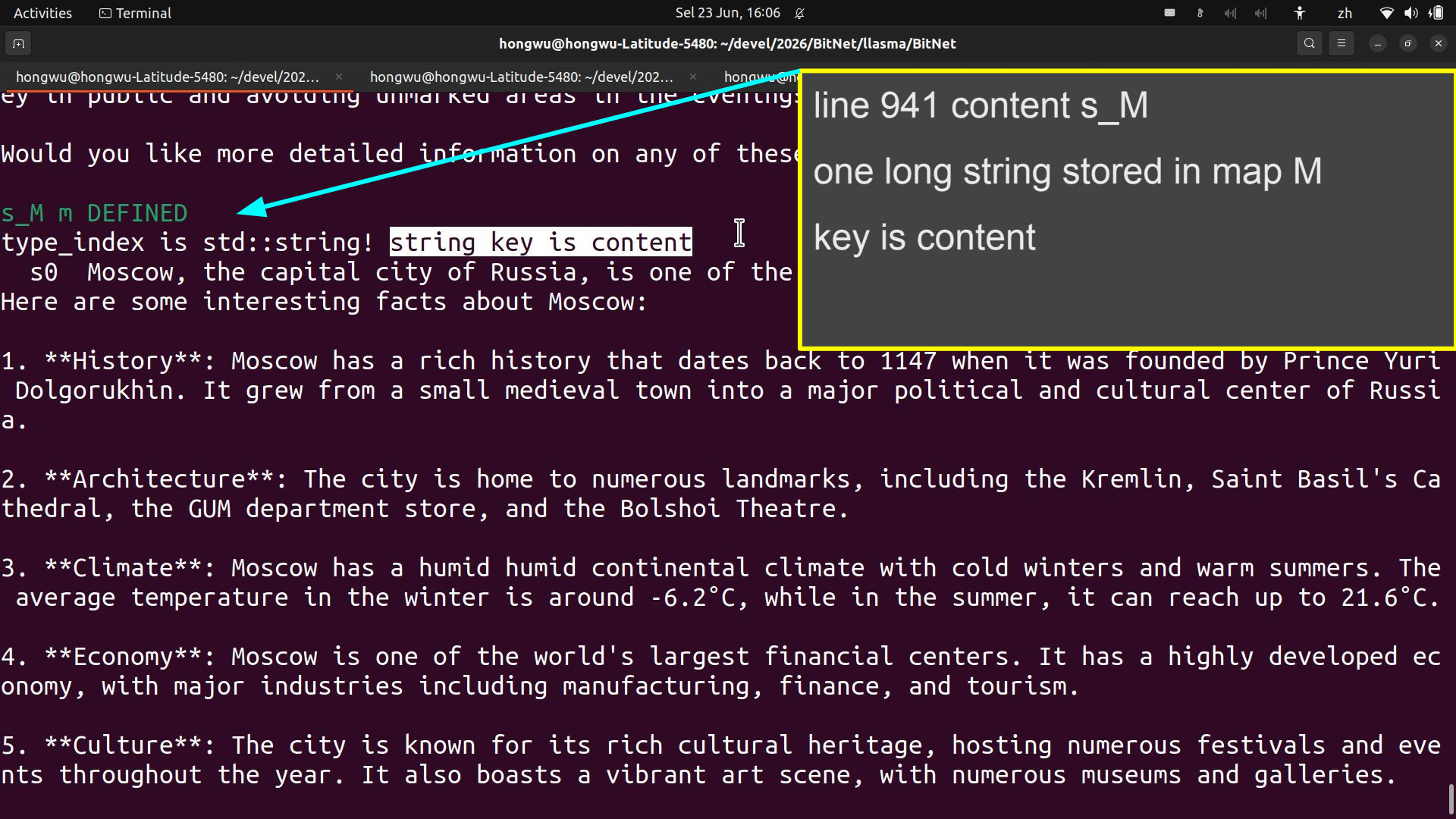
line 159 chat_add_and_format

one long string as variable

formatted.c_str()

```
135 #endif
136
137 /* LSM: chat_add_and_format FVM Clone&Migrate C/M 20260525_1425 */
138 static std::string chat_add_and_format(struct llama_model * model, std::vector<common_chat_msg> & chat_msgs, const std::string & role, const std::string & content) {
139
140     common_chat_msg new_msg{role, content};
141     // push variables on to stack
142     // push whole line as string to FVM, let FVM parse var type, var name, var values.
143     // then pop var into C++ to check correctness.
144
145     auto formatted = common_chat_format_single(model, g_params->chat_template, chat_msgs, new_msg, role == "user");
146     chat_msgs.push_back({role, content});
147
148     // LOG_DBG("\n\nformatted: '%s'\n", formatted.c_str());
149
150     LOG_DBG("\n\x1b[31mformatted:\x1b[0m '%s'\n", formatted.c_str());
151     // LOG_DBG("\n\n\x1b[31mfound an EOG token\x1b[0m\n");
152
153     return formatted;
154 }
155
156 /* LSM: chat_add_and_format org 20260525_1425
```

```
992                     const size_t original_size = embd_inp.size();
993
994                     if (params.escape) {
995                         string_process_escapes(buffer);
996                     }
997
998                     bool format_chat = params.conversation && params.enable_chat_template;
999                     std::string user_inp = format_chat
1000                        ? chat_add_and_format(model, chat_msgs, "user", std::move(buffer))
1001                        : std::move(buffer);
1002
1003                    // LSM_20260604: Clone -- copy var into M next to C++ code
1004                    M["model"] = model;
1005                    M["chat_msgs"] = chat_msgs;
1006
1007                    // TODO: one inconvenient of current chat template implementation is that we
    can't distinguish between user input and special tokens (prefix/postfix)
1008                    const auto line_pfx = common_tokenize(ctx, params.input_prefix, false, true)
    ;
1009                    const auto line_inp = common_tokenize(ctx, user_inp,            false, form-
    at_chat);
1010                    const auto line_sfx = common_tokenize(ctx, params.input_suffix, false, true)
    ;
1011
```


Activities
Terminal
Sel 23 Jun, 16:06
zh
hongwu@hongwu-Latitude-5480: ~/devel/2026/BitNet/llasma/BitNet
hongwu@hongwu-Latitude-5480: ~/devel/202...
hongwu@hongwu-Latitude-5480: ~/devel/202...
line 941 content s_M
one long string stored in map M
key is content
Would you like more detailed information on any of these
s_M m DEFINED
type_index is std::string! string key is content
s0 Moscow, the capital city of Russia, is one of the
Here are some interesting facts about Moscow:
1. **History**: Moscow has a rich history that dates back to 1147 when it was founded by Prince Yuri Dolgorukhin. It grew from a small medieval town into a major political and cultural center of Russia.
2. **Architecture**: The city is home to numerous landmarks, including the Kremlin, Saint Basil's Cathedral, the GUM department store, and the Bolshoi Theatre.
3. **Climate**: Moscow has a humid humid continental climate with cold winters and warm summers. The average temperature in the winter is around -6.2°C, while in the summer, it can reach up to 21.6°C.
4. **Economy**: Moscow is one of the world's largest financial centers. It has a highly developed economy, with major industries including manufacturing, finance, and tourism.
5. **Culture**: The city is known for its rich cultural heritage, hosting numerous festivals and events throughout the year. It also boasts a vibrant art scene, with numerous museums and galleries.

al airports and a major port.

7. **Population**: As of 2020, Moscow has a population of over 12 million people, making it the largest city in Russia and the second largest city in Europe.

8. **Languages**: While Russian is the official language, many residents of Moscow also speak English.

9. **Diversity**: The city is known for its cultural diversity, with various ethnic groups living in the city. The city's multi-cultural atmosphere is reflected in its many festivals and events.

10. **Safety**:**: Despite its size, Moscow is generally considered to be a safe city for tourists. However, like any major city, it's important to take standard precautions such as not displaying money in public and avoiding unmarked areas in the evenings.

Would you like more detailed information on any of these points?
  s_M m ASSIGNED

s_M m DEFINED
common_params* key is g_params

waiting for user input

>

```
419
420 void PhosVM::execute(std::string cmd, int ctx) {
421
422     // pointer to stack, integer indexed -- <stack> cannot be indexed !!
423     std::deque<std::any>* P;
424     P = (std::deque<std::any>*) &S;
425     unordered_map<string, any>& m = any_cast<unordered_map<string, any>&>((*P)[0]);
426
427     // Colon Definition Word CDW
428     unordered_map<string,vector<string>> c;
429
430     DBG = false;
431     if (m.count("C")) dcout << "m has CDW" << endl;
432     else d_cout << "in m CDW not found" << endl; // #define d_cout if (DBG) cout
433
434     std::any container = m["C"];
435     if (auto ptr = std::any_cast<int>(&container)) { // test for <any> type
436         d_cout << "int value is " << *ptr << "\n";
437     }
438     // Colon Definition Word CDW
439     else if (auto ptr = std::any_cast<unordered_map<string,vector<string>>>(&container)) {
440         d_cout << "vector of string: " << (*ptr).count("fruits") << "\n";
441         c = *ptr;
442
```

```
477     if (tokens[1]=="DBG") std::cout << "  size() " << tokens.size() << " / ";
478
479     for(i=0;i<tokens.size();i++)
480     {
481         // printf("%s",argv[i]);
482         if (tokens[1]=="DBG") std::cout << tokens[i] << " / "; // debug mode !PHOS DBG
483         tok=tokens[i];
484         dcout << "tok is " << tok << endl;
485
486         // execute Colon Definition Word
487         if (c.count(tok)) {
488             for (const auto& str : c[tok]) {
489                 std::cout << str << " ";
490             }
491             execute(join(c[tok], " "),ctx);
492             dcout << "AFTER colon def word" << endl;
493         } else
494         if (handlers.count(tok)) {
495             dcout << "  in handlers tok is " << tok << endl;
496             Handler func = handlers[tok];
497             // Must use (instance.*pointer)(args)
498             (this->*func)(ctx); // execute PhosVM::FUNCNAME()
499         }
500         else S.push(tok);
501
```

```
    using Handler = bool (PhosVM::*)(int);
    std::unordered_map<std::string, Handler> handlers = {
            {"dup", &PhosVM::dup},
            {"depth", &PhosVM::depth},
            {"gettype", &PhosVM::gettype},
            {"tostr", &PhosVM::tostr},
            {"type", &PhosVM::type},
            {"cr", &PhosVM::cr},
            {".s", &PhosVM::showstack},
            {"peek", &PhosVM::peek},
            {"peekr", &PhosVM::peekr},
            {"pick", &PhosVM::pick},
            {"p_M", &PhosVM::p_M},
            {"g_M", &PhosVM::g_M},
            {"s_M", &PhosVM::s_M},
            {"k_M", &PhosVM::k_M},
            {"pvs", &PhosVM::pvs},
            {"inV", &PhosVM::inV},
            {"vcb", &PhosVM::pick}, // variable curly bracket (define)
            {"veq", &PhosVM::pick}, // variable eq (define)
            {"vdc", &PhosVM::pick}, // variable declare (no define)
            {"find", &PhosVM::find}, // get index of stack item matching string partially
            {"ADD", &PhosVM::add},
            {"push", &PhosVM::push}, {"md5", &PhosVM::md5},
```

```
1135
1136 bool X_DBG; // save and restore DBG while setting DBG to local temp value
1137
1138 // S[0] = M, super buffer variable (PHP super global ?)
1139 bool PhosVM::s_M(int ctx){ // s_M set M modern convention
1140 try {
1141     std::deque<std::any>* P;
1142     P = (std::deque<std::any>*) &S;
1143
1144     // first argument on stack is key
1145     string key=anyToString(S.top()); S.pop();
1146
1147     unordered_map<string, any>& m = any_cast<unordered_map<string, any>&>((*P)[0]);
1148
1149     X_DBG=DBG; DBG=true;
1150     d_cout << "\n\x1b[32ms_M m DEFINED \x1b[0m" << endl; // new convention x_*
1151
1152     // conditional block to test type of next argument on stack
1153     std::any container = S.top();
1154     if (auto ptr = std::any_cast<struct llama_model*>(&container)) {
1155         d_cout << "llama_model key is " << key <<"\n";
1156         m[key]=*ptr;
1157         S.pop();
1158     }
```

```
1150    d_cout << "\n\x1b[32ms_M m DEFINED \x1b[0m" << endl; // new convention x_*
1151
1152    // conditional block to test type of next argument on stack
1153    std::any container = S.top();
1154    if (auto ptr = std::any_cast<struct llama_model*>(&container)) {
1155        d_cout << "llama_model key is " << key <<"\n";
1156        m[key]=*ptr;
1157        S.pop();
1158    }
1159    else if (auto ptr = std::any_cast<std::vector<common_chat_msg>>(&container)) {
1160        d_cout << "vector common_chat_msg key is ..." << key << "\n";
1161        m[key]=*ptr;
1162        S.pop();
1163    }
1164    else if (auto ptr = std::any_cast<common_params*>(&container)) {
1165        d_cout << "common_params* key is " << key << "\n";
1166        m[key]=*ptr;
1167        S.pop();
1168    }
1169    else {
1170      gettype(ctx); // print type_index is ....
1171      string v_type=anyToString(S.top()); S.pop();
1172      int n;
1173      if (v_type=="string")
1174      {
```

```
1169    else {
1170      gettype(ctx); // print type_index is ....
1171      string v_type=anyToString(S.top()); S.pop();
1172      int n;
1173      if (v_type=="string")
1174      {
1175        // special string prefix, trigger special type M entry
1176        string s0=anyToString(S.top());
1177        d_cout << "string key is " << key << "\n";
1178        std::cout << "  s0 " << s0 << std::endl;      // n = stoi(s0,nullptr,10);
1179
1180        if (s0=="t_bool") {
1181          S.pop();
1182          s0=anyToString(S.top());
1183          if (s0=="true") m[key]=true;
1184          else if (s0=="false") m[key]=false;
1185          else m[key]=true;
1186
1187          if (key=="DBG") DBG=any_cast<bool>(m[key]); // special global flag
1188        }
1189        else m[key]=s0;
1190
1191        S.pop();
1192      }
```

```
1184          else if (s0=="false") m[key]=false;
1185          else m[key]=true;
1186
1187          if (key=="DBG") DBG=any_cast<bool>(m[key]); // special global flag
1188        }
1189        else m[key]=s0;
1190
1191        S.pop();
1192      }
1193      else if (v_type=="int") { // int cast and string cast are not the same because string is pointer?
1194        n = any_cast<int>(S.top());
1195        m[key]=(int) n;
1196        S.pop();
1197
1198        std::cout << "  s_M " << any_cast<unordered_map<string, any>>((*P)[0]).bucket_count() << "   " << any_cast<int>(m[key]) << std::endl;
1199      }
1200      d_cout << "  s_M m ASSIGNED " << endl;
1201    }
1202    DBG=X_DBG; // restore DBG
1203    } catch (std::exception& e){ std::cerr << "exception: " << e.what() << std::endl; }
1204    return true;
1205}
```

FORTHification + Omnihash

Solution for “Dead Programmer Crisis”

FORTHificiation + Omnihash will convert the existing free software licenses so that living programmers and the families of dead programmers may ensure the rewards of:

– Separation of Disclosure and Royalties omnixtar.github.io/contract

Recursive Hilbert Hotel (FORTH-Hilbert Hotel) – Omnihash room number + FORTH words for cabinet, drawer & file labels in each room

(A) Forthification + Omnihash = open source tax & rewards for programmers & families of dead free software programmers
(B) Omnimesg (Omnihash messaging)
(C) build up F(N) forth words of cardinality N
(D) I2P Invisible Internet Project & equivalent to get hardware resources
(E) GRATIS Grand Unified AI token trading + LLASMA Cars + Cesium.js + Clone of Google maps + Casino + Merge ALL games w/ WebGL
– What else could we do to patch up all five modules about into a seamless, revenue generating program ?

- Use the above as AI prompt for new ideas

FORTHification + Omnihash

1. We define the cardinality of a FORTH stack machine word as the *number of primitive words used in its colon definition*, and this enables ALL SOURCE CODE having been written by human programmers or AI agents, after translated using Phoscript engine, to become "enumerable" – thereby searchable, modularised and composable.

FORTHificiation + Omnihash will convert the existing free software licenses so that living programmers and the families of dead programmers may ensure the rewards of:

– Separation of Disclosure and Royalties

Recursive Hilbert Hotel (FORTH-Hilbert Hotel)
– Omnihash room number + FORTH words for cabinet, drawer & file labels in each room

(A) Omnimesg (Omnihash messaging)

(B) Trispecies Monetary Transactions (TMX)
(Trispecies=Fiat+Crypto+Bullion) based on Omnimesg

(C) Resource/Infrastructure Sharing (using TMX)
– C1. Forthification + Omnihash = open source tax & rewards for programmers & families of dead free software programmers (software rights)
– C2. I2P Invisible Internet Project & equivalent to get hardware resources (sharing hardware)
– C3. GRATIS Grand Unified AI token trading (share GPU/CPU)

(D) In Game Transactions (IGX) (using TMX)
– LLASMA Cars + Cesium.js + Clone of Google maps + Casino + Merge ALL games w/ WebGL + all kinds of E-Commerce + Clone social media + TerraClaim (3D monopoly w/ Omnihash)

(E) build up F(N) forth words of cardinality N (general, include all of the above)

Omni*Web

https://omnixtar.github.io/contract/

Omni*Contract: Ownership & Rights of Use of Digital Asset (Source Code)

Like

- On the **Separation of Disclosure and Royalties** of the Source Code (July 21, 2024)

1. You, a human agent of a company or government agency, may read the source code without making payments to the author or authors, but if you execute this program on behalf of your company or agency for commercial purposes, we reserve the rights to claim royalties from you or your company or agency.

2. Your copy of source code shall be attached with at least one Omni* Hash Contract bearing the Omnihash of a Omni* Agent and your own Omnihash, to authorise you the permissions to use or modify said source code, otherwise you shall pay maximum penalties allowed by a legal court of your jurisdiction, for the damages you have incurred for deploying the source code pertaining to clause (1).

FORTHification + Omnihash

1. We define the cardinality of a FORTH stack machine word as the *number of primitive words used in its colon definition*, and this enables ALL SOURCE CODE having been written by human programmers or AI agents, after translated using Phoscript engine, to become "enumerable" – thereby searchable, modularised and composable.

2. We also propose the Inverse Turing Test (ITT) and

3. a process to construct measurable and verifiable definitions of Artificial General Intelligence based on the concepts above.

…. About the Future

Free software can take however long to clone popular apps like Google maps.

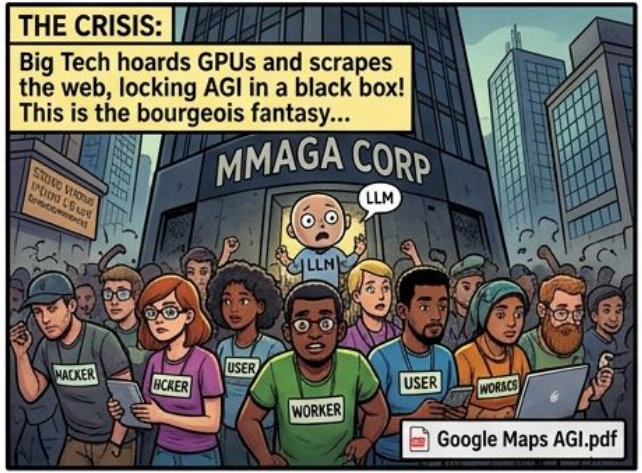
Just need to clone one popular app to get user base and programmers.

- simulate space time!!
- Code reuse factor? Can measure?
- most important management idea
- strength of free software!!

Optimistic: 1 year, Less Optismistic 2 years, Reality 5 years?
Worst case scenario: World goes on as it is, or worst ….
OR … Someone else invented same thing as Omnihash + Phoscript? Maybe they will be luckier?

‘THE DIGITAL PROLETARIAT: BUILDING AGI, MAPPING OUR FATE’

THE CRISIS:
Big Tech hoards GPUs and scrapes the web, locking AGI in a black box! This is the bourgeois fantasy...

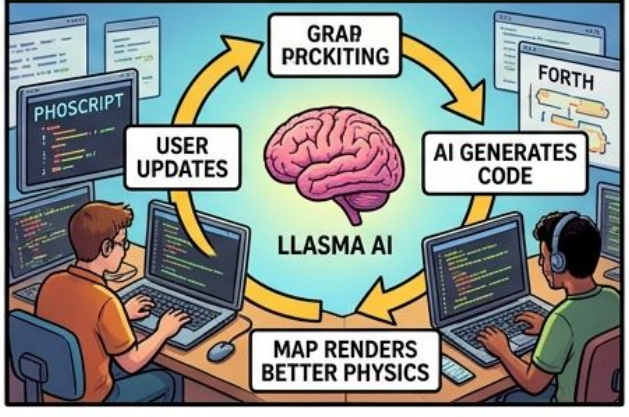


THE REAL PATH:
...But AGI isn't born from text! It starts with physics, space, and a room.
A human learns by navigating, not reading.

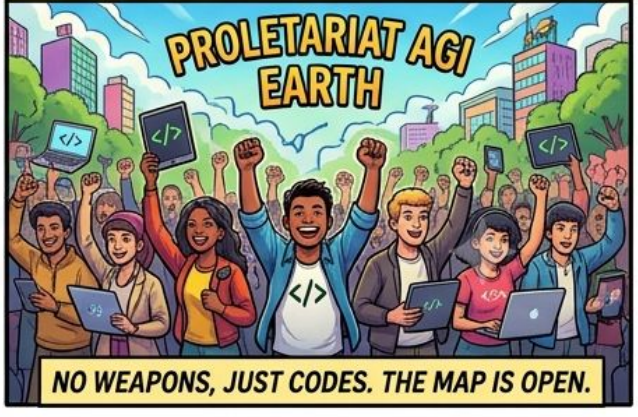
THE NEW MAP: WE BUILD OUR ROOM: A CLONED, OPEN GOOGLE EARTH!

Every coordinate is indexered with Omnihash.
Workers claim digital real estate! We are landlords!

RECURSIVE REVOLUTION: auditable, ownable!
The AI observes the map, writes code based on physics, and improves itself. The digital serfs become landlords!



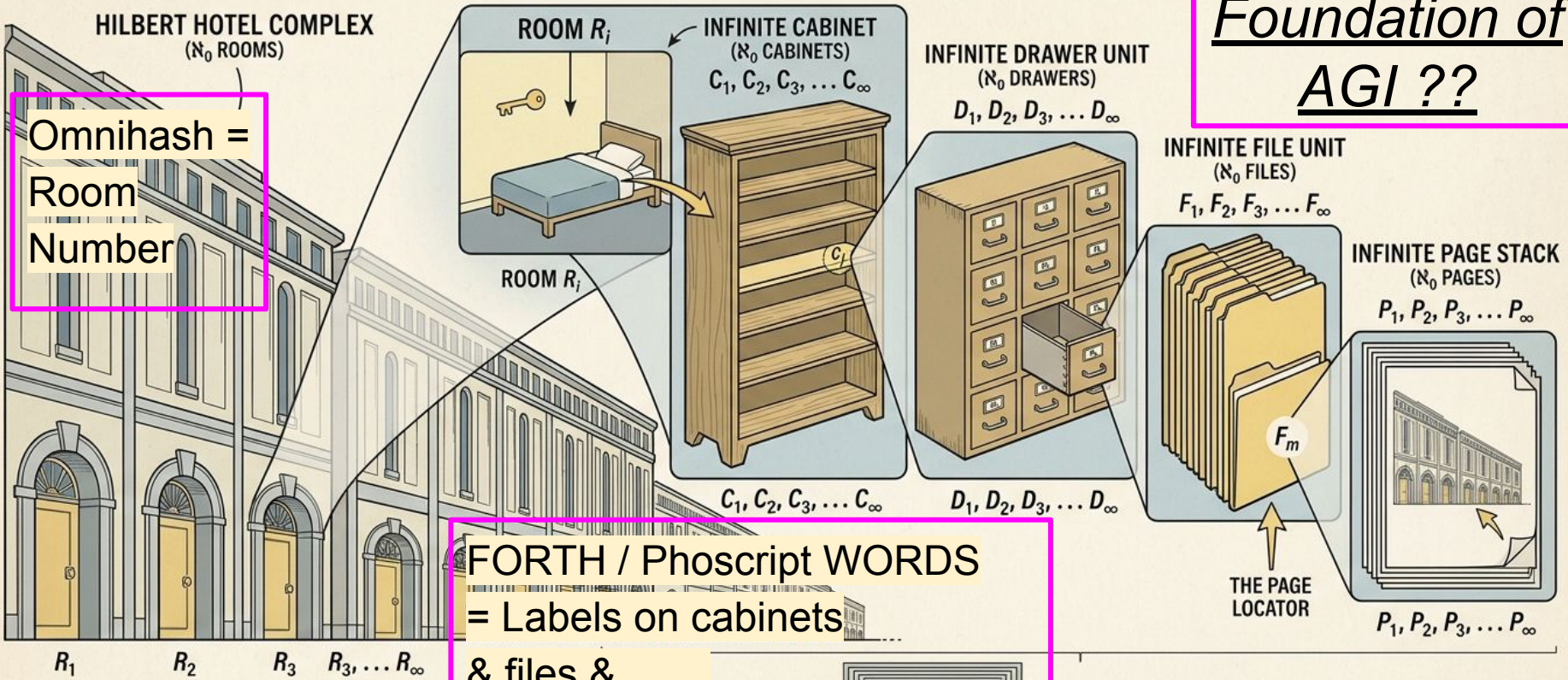
OUT-BUILD THE FUTURE: A global army maps the world FORTH word at a time. Big Tech can't moat free software! Lay claim to the digital Earth, lay claim to the mind. Out-build the future!



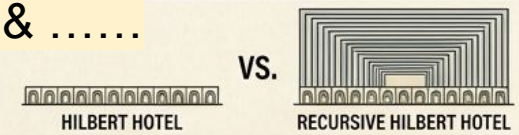
THE RECURSIVE HILBERT HOTEL

Foundation of AGI ??

Omnihash = Room Number



FORTH / Phoscript WORDS = Labels on cabinets & files & ……



THE CARTESIAN PRODUCT $\mathbb{N}^5$ IS STILL COUNTABLE.

Inverse Turing Test (ITT)

Turing Test: (essentially)

- Can a computer program (AI agent) talk like human beings?

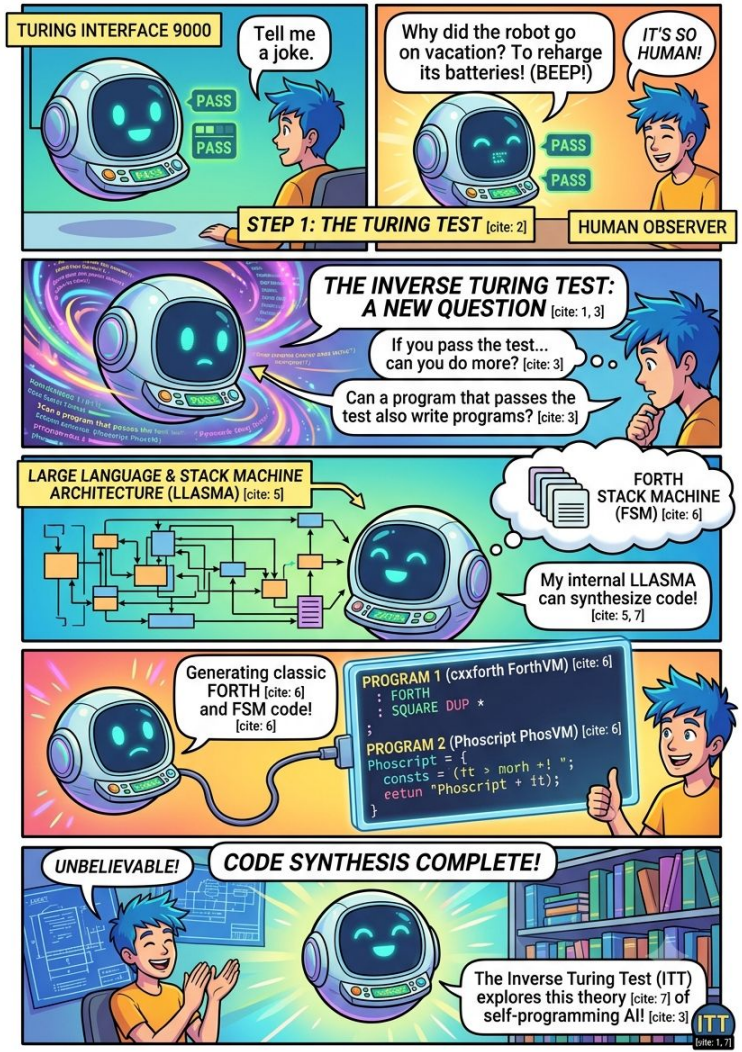
Inverse Turing Test: (one of many versions)

- Can a computer program that passes a Turing Test also write computer programs?

(Start with simple / general definition. Improve later.)
(ITT: a theory to cover LLASMA ?)

Write computer programs:
LLASMA (Large Language & Stack Machine Architecture)

a) Compose “classic” FORTH programs (cxxforth ForthVM)
b) Compose “FORTH stack machine” (FSM) programs e.g. Phoscript PhosVM

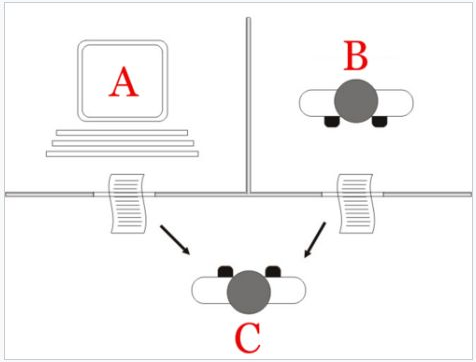


Turing test

From Wikipedia, the free encyclopedia

Not to be confused with Turing machine. "Imitation game" redirects here. For the film, see The Imitation Game. For other uses, see Turing test (disambiguation).

The **Turing test** was designed by Alan Turing to assess a machine's ability to exhibit intelligent behaviour equivalent to that of a human by imitating interactive dialogue. In the modern version, a human evaluator judges a text transcript of a natural-language conversation between a human and a machine. The evaluator tries to identify the machine, and the machine passes if the evaluator cannot reliably tell them apart. The results would not depend on the machine's ability to answer questions correctly, only on how closely its answers resembled those of a human. Since the Turing test is a test of indistinguishability in performance capacity, the original (verbal) version generalizes naturally to all of human performance capacity, including nonverbal (robotic).[2]



Player C, the interrogator, is tasked with determining which player – A or B – is a computer and which is a human using only the responses to written questions.[1]

The test was introduced as the **imitation game**[3] by Turing in his 1950 paper "Computing Machinery and Intelligence" while working at the University of Manchester.[4] It opens with the words: "I propose to consider the question, 'Can machines think?'." Because "thinking" is difficult to define, Turing chooses to "replace the question by another, which is closely related to it and is expressed in relatively unambiguous words".[5] Turing describes the new form of the

Part of a series on

Artificial intelligence (AI)

Artificial General Intelligence

1. Defining AGI is a *process of narrowing its definitions*.

2. A (tentative) measurable and verifiable definition of Artificial General Intelligence based on: (part of Claude formatted answers – has URL – next page)

3. Ability to generate new knowledge – *bounded by mathematical laws* (in addition to (2) & completes it?)

4. Fiat Lux Hypothesis / Whole Brain Emulation
– If we can simulate everything, we can simulate AGI.

4. Narrower version:
– Human babies live in the real world.
– Simulation of real world is prerequisite for simulated human babies to acquire AGI.
– *LLASMA works on simulation part, let Trillion Dollar Corps work on AGI/LLM.*

5. "FORTH-Hilbert Hotel"
We can simulate everything as N, Cardinality of F(N), increases towards infinity.
– I2P Invisible Internet Project or similar tech guarantee the hardware resources.

Artificial General Intelligence

1. Defining AGI is a *process of narrowing its definitions*.

2. A (tentative) measurable and verifiable definition of Artificial General Intelligence based on: (part of Claude formatted answers – has URL – next page)

3. Ability to generate new knowledge – *bounded by mathematical laws* (in addition to (2) & completes it?)

4. Fiat Lux Hypothesis / Whole Brain Emulation
– If we can simulate everything, we can simulate AGI.

5. "FORTH-Hilbert Hotel"
We can simulate everything as N, Cardinality of F(N), increases towards infinity.
– I2P Invisible Internet Project or similar tech guarantee the hardware resources.

- Nice pun for generations of Hilbert Hotel == generations of programming languages
- Grounded but Unbounded
- Integers: enumerable but unbounded

Claim 3: The set-theoretic AGI definition

The formal argument:

Let U = the universal set of all things that can be known about the universe.

Let H = human collective intelligence (what the human species collectively knows).

Let AGI = Artificial General Intelligence as conventionally defined (the capacity to do what humans can do across all domains).

The proposal argues:

- H ⊂ U (human knowledge is a proper subset of universal knowledge — humans don't know everything)
- AGI ⊆ H (AGI benchmarks human performance, so AGI capability is contained within human capability)
- Therefore AGI ⊂ U (AGI is a proper subset of universal knowledge)
- Therefore: a system capable of simulating U — capable of executing the universal set of words — is, by logical containment, capable of AGI

Is the logical structure valid?

Yes, conditional on the premises. If AGI ⊆ H and H ⊂ U, then any system with capability U necessarily has capability H, and therefore necessarily has capability AGI. The containment argument is logically sound

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)

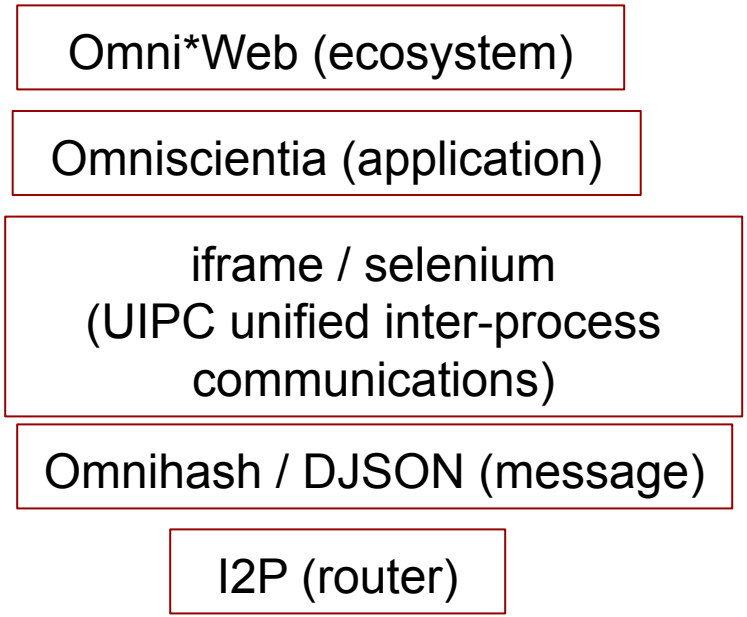
Developed countries are failing – debt, infrastructure, immigrants …

- 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



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 (MMAGA revenues 2025 > USD 2 trillion), 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.