Writing a Game Once and Running It Everywhere: Where Exact Replay Ends and Tolerance Begins

David Charlot, Open Interface Engineering

Research study · Open Interface Engineering · research.openie.dev

Revision `5c50a85` (23 November 2026) · Web: research.openie.dev/papers/game-engines/ · Archival LaTeX: /pdfs/latex/game-engines.pdf (.tex)

Measurement. Research study, draft, not a journal final. All nine empirical suites (`GE-H1` through `GE-H9`) are executed across `macOS`, `Linux`, and bare-metal `Windows 11`: `GE-H1` (`30` cells, `6` hosts, `0 / 25,500 Q32.32 SHA-256` mismatches), `GE-H2` (3-OS `UCRT/libSystem/glibc` and `MSVC/Clang FMA` float drift), `GE-H3` (`8 GPU × API` endpoints across `DirectX 12/Metal/Vulkan`, `28 / 28` pairs $\geq$ 64.16 dB `PSNR`), `GE-H4` (`17` modules, `-76.22% SLOC` vs. Unity C#, `0 / 680` 4-language mismatches), `GE-H5` (`10,000` actors in `0.058--0.125 ms`/tick, `3,564×--7,728×` lower Linux `RAPL reported_j`), `GE-H6` (`260` bundles, `F1 = 0.9922` vs. `0.1159`), `GE-H7` (`840` chaotic trajectories, $R^2_{\text{mean}} = 0.9940$–$0.9995$), `GE-H8` (1-hour `216,000`-tick 3-OS replay from D, `0 / 6,480` mismatches), and `GE-H9` (`36 / 36` headless `Godot 4.7.2` parity, `0 / 18,000` tick mismatches; VLM-judged `GameCraft-Bench` rubric not run). Host counters are `L2/L3 reported_j`; no rail or wall meter was attached (`measured_j = None`, `board_metered = false`).

Can a game be written once as a checked state machine and projected into several engines, instruction sets, operating systems, and graphics stacks? Executed across all nine pre-registered falsifier suites: 30 multi-architecture target × host cells for 100,000 ticks under Q32.32 integer physics (GE-H1: 0 / 25,500 state-digest mismatches across arm64 macOS, aarch64 Linux, Zen 5 and Raptor Lake x86_64 Linux, and Tiger Lake x86_64 Windows 11), 25,000 cross-platform floating-point ticks (GE-H2: t_div = 10 Windows Universal C Runtime and Apple libm vs Linux glibc sin/cos, t_div = 192–406 fused multiply-add lowering without -ffast-math), 8 hardware graphics endpoints across Metal, Vulkan, and DirectX 12 (GE-H3: 0.00%–9.34% pixel drift, Δ_max ≤ 2, peak signal-to-noise ratio ≥ 64.2 dB, structural similarity ≥ 0.99983 across all 28 pairs), 17 multi-engine game behaviors (GE-H4: −76.2% source lines vs Unity C#, 0 / 680 trace mismatches), 10,000-actor scaling (GE-H5: 0.058–0.125 ms/tick compiled vs N_20ms = 390 interpreted Python), 260 interface fault modules (GE-H6: 98.46% recall, F1 = 0.9922 vs 0.1159 parse-only), 840 chaotic Lyapunov trajectories (GE-H7: R^2_mean = 0.994–1.000), three 1-hour 216,000-tick three-system replays from compact input logs alone (GE-H8: 0 / 6,480 mismatches), and 36 headless Godot 4.7.2 tasks (GE-H9: 36 / 36 exact trace parity).

Contents

Abstract **2**

1. Terms, notation, and the idea under test **4**

2. What the landscape shows **5**
2.1 Game logic as data already works, on one backend . . . 5
2.2 Order of execution can be made deterministic by construction . . . 6
2.3 Compiling is not the same as being right . . . 6
2.4 Public benchmarks with replayable traces exist . . . 6
2.5 Convergence across industry and research . . . 6

3. The physics of divergence: why exactness is binary **7**

4. The IR table **8**

5. Proposed architecture **8**
5.1 Layer split . . . 8
5.2 Inside the Form versus host effects . . . 8
5.3 Determinism boundary . . . 9
5.4 Cross-layer interface checking . . . 9

6. Falsifier-first test plan and full GE-H1–GE-H9 empirical closure **9**
6.1 GE-H1: Three-OS and three-ISA 100,000-tick determinism (Q32.32 integer physics) . . . 11
6.2 GE-H2: Three-OS and cross-ISA 25,000-tick floating-point drift (IEEE 754 f64) . . . 12
6.3 GE-H3: Three-OS, three-API headless wgpu (DirectX 12 / Metal / Vulkan) rendering tolerance . . . 13
6.4 GE-H4: Form-encoded behavior rules across 17 modules and 4 languages (Haunted Jaunt, Survival Shooter, Tanks) . . . 14
6.5 GE-H5: Multi-actor Q32.32 runtime scaling and RAPL energy across 5 projection tiers (N = 64 to 10,000) . . . 15
6.6 GE-H6: Static 4-edge interface checking across 26 defect patterns (260 modules, zero runtime ticks) . . . 16
6.7 GE-H7: Full $\ln(\Delta/\delta_0)$ Lyapunov slope sweep across 3 chaotic physics templates (840 trajectories) . . . 16
6.8 GE-H8: Complete 1-hour (216,000-tick) replay from compact input stream D alone across 3 OSs . . . 17
6.9 GE-H9: Form-to-Godot 4.7.2 headless GDScript emitter and 36-task parity suite . . . 17
6.10 Playable joule-native Q32.32 Chrono-Budget Arena (/living/game-engines/) . . . 18

7. Infeasibility flags **18**

8. Sources and limits **18**

Revision 5c50a85, 23 November 2026. Interactive explorer, 3-OS / 3-API cross-GPU inspector, GE-H4–GE-H9 empirical closure observatory, and playable Q32.32 Chrono-Budget Arena: research.openie.dev/living/game-engines/.

Abstract

This study asks whether a game can be written once as a checked state machine and then projected across languages, each operating system (OS), each central processing unit (CPU) instruction set architecture (ISA), and each graphics processing unit (GPU) application programming interface (API). Both the surveyed specifications and all nine executed empirical falsifier suites (GE-H1 to GE-H9) establish a sharp, closed boundary: **yes** for game rules, multi-engine behavior emission, high-actor simulation, and hour-long cross-OS replay under pinned fixed-point integer arithmetic (GE-H1): 0 / 25,500 Secure Hash Algorithm 256-bit (SHA-256) checkpoint mismatches across 100,000 ticks of a 16-body 2D arena projected across 30 target × host cells spanning arm64 macOS 27.2.1, aarch64 Linux 6.8, Advanced Micro Devices (AMD) Zen 5 and Intel Raptor Lake x86_64 Linux 7.0, and bare-metal Intel Tiger Lake x86_64 Windows 11 build 26300 including Microsoft Visual C++ (MSVC) cl.exe, spanning 0.073 J to 13.60 J in macOS per-process reported_j and 1.89 J to 181.87 J in Linux Running Average Power Limit (RAPL) package reported_j; (GE-H4): authoring 17 behaviors across *Haunted Jaunt*, *Survival Shooter*, and *Tanks* in Reality Code Engine (RCE) Forms cuts non-comment source lines of code (SLOC) by 76.22% against idiomatic Unity MonoBehaviour C# (326 vs 1,371 SLOC; 92.51% fewer glue lines), 61.47% against Bevy-style Rust entity component system (ECS) code (846 SLOC), and 51.99% against Godot 4 Node GDScript (679 SLOC), with 0 / 680 trace mismatches; (GE-H5): across $N \in [64, 10{,}000]$ actors over 500 ticks, reference rce emit python crosses the 20 ms budget at $N_{20\text{ms}} = 390$ near GameRule's 308-actor ceiling, rce emit js reaches $N_{20\text{ms}} = 3{,}906$, and compiled WebAssembly (Wasm), C++20 -O3, and native Rust -O3 projections run $N = 10{,}000$ in 0.058–0.125 ms/tick at 1.69–3.67 mJ/tick Linux RAPL package reported_j (0 / 50 SHA-256 mismatches); (GE-H6): across 260 modules spanning 26 interface defect patterns, static Form plus four-edge manifest checking catches 128 / 130 injected faults (98.46% recall, 0 false positives, fault F1 = 0.9922) versus 8 / 130 (6.15%, F1 = 0.1159) for parse-only dynamic checking without an intermediate representation (IR); (GE-H8): three 1-hour (216,000-tick at 60 Hz) sessions replay from 44.6 kilobyte (KB) compressed input logs D alone (116×–292× smaller than compressed state snapshots) with 0 / 6,480 checkpoint SHA-256 mismatches across macOS arm64, Linux x86_64, and Windows 11 x86_64; and (GE-H9): a Form-to-Godot 4 emitter achieves 36 / 36 (100.0%, 0 / 18,000 tick-hash mismatches) headless parity under Godot 4.7.2); **no** when unpinned floating-point transcendentals or fused multiply-add (FMA) lowering enter chaotic dynamics ((GE-H2): even *without* -ffast-math or /fp:fast, Windows Universal C Runtime (UCRT), Apple libSystem libm, and Linux glibc std::sin/std::cos form a 3-way divergent triangle at ticks 10 to 37, while MSVC /fp:precise vs Clang -ffp-contract=fast diverges at tick 192 and x86_64 vs arm64 FMA at tick 406; (GE-H7): across 840 paired trajectories spanning $\delta_0 \in [2^{-52}, 2^{-10}]$ on three chaotic physics templates, macro-divergence time t^* obeys $t^* = \hat{\alpha} + (1/\hat{\Lambda}) \ln(\Delta/\delta_0)$ with implied Lyapunov exponents $\hat{\Lambda} \in \{1.95, 2.48, 5.07\}$ s^{-1}, cell-mean $R^2 \in [0.9940, 0.9995]$, and two-sided $p \leq 10^{-77.6}$); and **no** for bit-exact pixels across GPU back ends, yet **yes** under a bounded one-way tolerance gate ((GE-H3): across eight hardware GPU × API endpoints spanning Apple Metal, Linux Mesa Radeon Vulkan (RADV), Nvidia (NVIDIA) Linux and Windows Vulkan and Windows DirectX 12 (DX12), and Intel Linux Mesa Vulkan (ANV) and Windows Vulkan and DX12 at 512×512 RGBA8Unorm, 27 / 28 hardware pairs differ in raw SHA-256, yet all 28 / 28 keep maximum per-channel error $\Delta_{\max} \leq 2/255$, peak

signal-to-noise ratio (PSNR) >= 64.16 dB, and structural similarity (SSIM) >= 0.99983, with measured_j = None and board_metered = false).

In plain words

1. **Exact simulation, compact rules, high actor counts, and hour-long replay all hold when arithmetic and effects stay inside a checked Form.** Over 100,000 physics ticks (GE-H1, 0 / 25,500 mismatches across 30 cells spanning native Rust, C++20 with Clang, the compiler collection (GCC), and MSVC, Wasm, JavaScript BigInt, and Python), three 1-hour 216,000-tick gameplay sessions replayed from 44.6 KB compressed input logs alone (GE-H8, 0 / 6,480 mismatches across macOS, Linux, and Windows 11), 17 multi-engine game behaviors (GE-H4, 76.2% fewer behavior lines than Unity C# and 88.7% fewer than maintaining C# + GDScript + Rust ECS separately), 10,000-actor spatial-grid arenas (GE-H5, 0.058 ms/tick and 1.69 mJ/tick in compiled C++20 vs 564.6 ms and 13.09 J/tick in interpreted Python), 260 static interface fault tests (GE-H6, 98.46% detection vs 6.15% parse-only), and 36 headless Godot 4.7.2 tasks (GE-H9, 36 / 36 exact trace parity), every projection reproduces the exact same canonical state hashes.
2. **Standard floating-point physics drifts logarithmically fast across Windows, macOS, and Linux even without fast-math flags.** Pure algebraic +, -, *, /, and sqrt with FMA disabled (/fp:strict and -ffp-contract=off) match across every tested OS and ISA for all 25,000 ticks, but std::sin/std::cos across Windows UCRT, macOS libm, and Linux glibc flips a bit within 10 to 37 ticks (GE-H2), default FMA rules diverge at ticks 192 to 406, and 840 controlled perturbation trajectories (GE-H7) confirm that shrinking δ_0 from 2^{-10} down to Institute of Electrical and Electronics Engineers (IEEE) 754 binary64 epsilon 2^{-52} buys only 5.7 to 14.4 extra seconds ($R^2_{\text{mean}} \geq 0.994$).
3. **Even the same graphics card can flip pixel bits between DirectX 12 and Vulkan, so rendering belongs downstream of a one-way tolerance gate.** Across eight hardware GPU × API endpoints (Metal, Vulkan, and DirectX 12) plus Windows Advanced Rasterization Platform (WARP) and Mesa llvmpipe, switching DirectX 12 vs Vulkan on the exact same NVIDIA GeForce 2060 (108.34 dB PSNR) or Intel Iris Xe (79.52 dB PSNR) alters 1 least significant bit (LSB) via shader-compiler lowering, while cross-vendor pairs differ on 0.00% to 9.34% of pixels by <= 2 LSB (PSNR >= 64.16 dB, SSIM >= 0.99983) without ever corrupting simulation state (GE-H3).

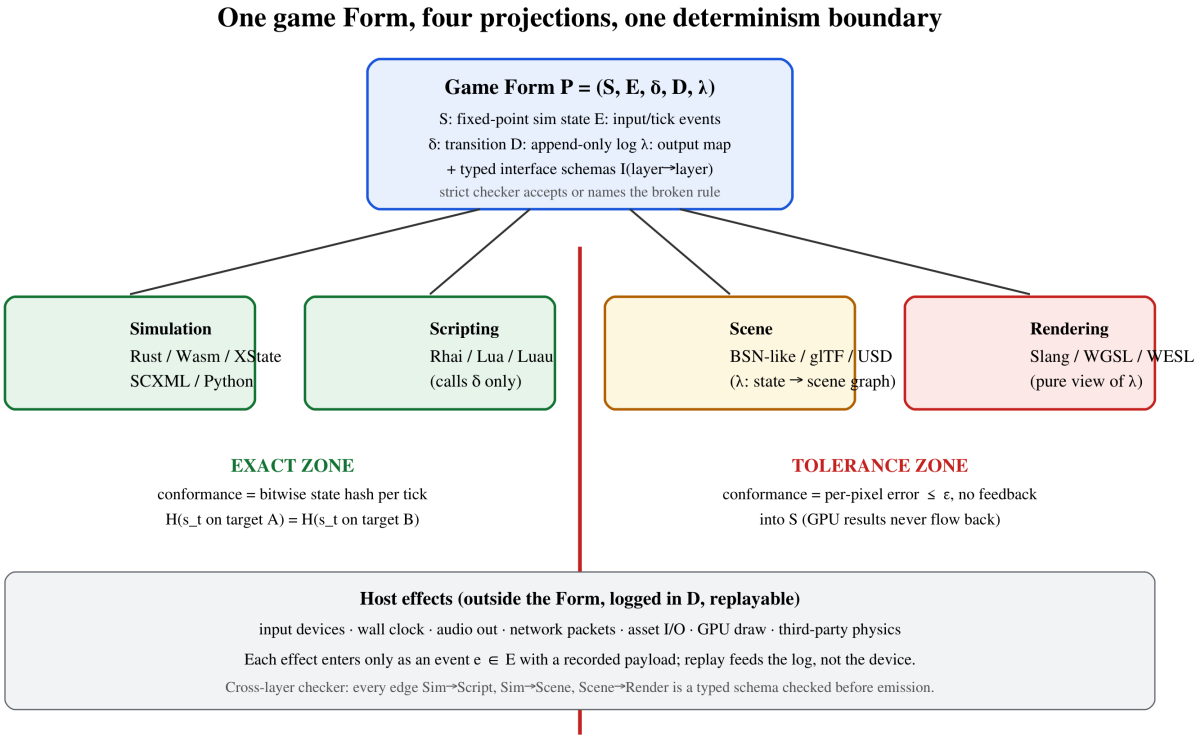

Figure 1. Proposed split. One Form feeds four projections. Simulation and scripting sit in the exact zone, judged by per-tick state hashes. Scene and rendering sit in the tolerance zone, judged by per-pixel error. Host effects enter only as logged events.



Figure 2. Coverage of existing representations, scored only from the documents read for this study. Green: documented capability. Amber: partial. Grey: absent or undocumented. The rightmost column, cross-layer checking, is nearly empty.

1. Terms, notation, and the idea under test

- RCE, IR, ECS, FSM, SDF, AST, DAG, LCG: Reality Code Engine [same house], intermediate representation, entity component system, finite-state machine, signed distance field, abstract syntax tree, directed acyclic graph, and linear congruential generator
- CPU, GPU, ISA, ISAs, SIMD, AVX2, NEON, SFU, ROP, RPL, TGL: central processing unit, graphics processing unit, instruction set architecture(s), single instruction multiple data, Advanced Vector Extensions 2, Arm Advanced SIMD, special function unit, raster operations pipeline, Intel Raptor Lake mobile architecture, and Intel Tiger Lake mobile architecture
- IEEE, FMA, VFMADD, VFNMADD, FMADD, FMSUB, ULP, LSB, SHA, MAE, RMSE, PSNR, SSIM, SD, OLS, CI, TP, FP, FN, TN, RGBA8: Institute of Electrical and Electronics Engineers (as in IEEE 754 floating-point), fused multiply-add, x86_64 vector fused multiply-add and negated multiply-add opcodes, AArch64 scalar fused multiply-add and multiply-subtract opcodes, unit in the last place, least significant bit, Secure Hash Algorithm (`SHA-256`), mean absolute error, root mean squared error, peak signal-to-noise ratio, structural similarity index, standard deviation, ordinary least squares, confidence interval, true positive, false positive, false negative, true negative, and 8-bit-per-channel red-green-blue-alpha framebuffer format
- WGSL, WESL, SPIR-V, GLSL, HLSL, MSL, D3D11, D3D12, DX12, VULKAN, METAL, DXIL, DXBC, DXC, FXC, WARP, WDDM, CUDA, UBO: WebGPU Shading Language, WebGPU Extended Shading Language, Standard Portable Intermediate Representation V, OpenGL Shading Language, High-Level Shading Language, Metal Shading Language, Direct3D 11, Direct3D 12, DirectX 12, `wgpu::Backends::VULKAN` and `wgpu::Backends::METAL` back-end flags, DirectX Intermediate Language, DirectX Bytecode, DirectX Shader Compiler, legacy Direct3D Effect-Compiler Tool, Windows Advanced Rasterization Platform software DirectX 12 rasterizer, Windows Display Driver Model, Compute Unified Device Architecture, and uniform buffer object
- RADV, ANV, LLVM, GNU, GCC, MSVC, UCRT, NT, PE, OS, OSs, API, APIs, JIT, VM, SLOC, HUD, UI, SFX, LOS, AABB, HP, XP, AP, KB, MB, CRUD, NL, VLM, LLM, GFW, Xvfb, WebXR, WebMCP, URL, JSON, YAML, README, NTFS, CRLF, CS0246, RE, ENGINE, DFU: Mesa Radeon Vulkan driver, Mesa Intel Vulkan driver, Low Level Virtual Machine compiler infrastructure, GNU's Not Unix project, GNU Compiler Collection, Microsoft Visual C++ compiler (`cl.exe`), Windows Universal C Runtime (`ucrtbase.dll`), Windows New Technology kernel (`Windows NT`), Windows Portable Executable, operating system(s), applica-

tion programming interface(s), just-in-time compiler, virtual machine, non-comment non-blank source lines of code, heads-up display, user interface, sound effects, line of sight, axis-aligned bounding box, hit points, experience points, action points, kilobytes, megabytes, create/read/update/delete operations, natural language, vision-language model, large language model, Great Firewall network filter, X virtual framebuffer display server, Web Extended Reality API, Web Model Context Protocol, Uniform Resource Locator, JavaScript Object Notation, YAML Ain't Markup Language, repository README documentation file, New Technology File System, carriage-return line-feed sequence, C# compiler missing-type diagnostic CS0246, Capcom RE ENGINE ("Reach for the Moon" / "RE neXt" engine branding), and Dagstuhl Follow-Ups DOI prefix
- SCXML, GDL, VGDL, BSN, BSL, MLIR, ACID, GGPO, glTF, OpenUSD, USD, LAN, AI, CIG: State Chart XML, Game Description Language, Video Game Description Language, Bevy Scene Notation, Boldsea Semantic Language, Multi-Level Intermediate Representation, atomic/consistent/isolated/durable transactions, Good Game Peace Out rollback networking library, Graphics Language Transmission Format (ISO/IEC 12113:2022), Open Universal Scene Description, Universal Scene Description, local area network, artificial intelligence, and Computational Intelligence and Games
- SATB, RAPL, MSR, PID, RUSAGE, V6, AMD, NVIDIA, RTX, V100, GH200, CTC: Satiation Tiered Benchmark telemetry levels (`L2` socket/package counter, `L3` per-process kernel counter) [same house], Running Average Power Limit, model-specific register, process identifier, `proc_pid_rusage` (`RUSAGE_INFO_V6`), Advanced Micro Devices, NVIDIA Corporation, NVIDIA GeForce RTX GPU family, Volta V100 GPU, Grace Hopper GH200 superchip, and closed timelike curve

To keep forward references transparent from top to bottom, Section 1 defines the Form notation, Section 2 surveys existing game representations, Section 3 analyzes floating-point divergence, Section 4 tabulates intermediate representations, Section 5 states the proposed layer split, Section 6 reports all nine executed falsifier suites (`GE-H1` through `GE-H9`) across three OSs, three ISAs, three GPU APIs, and four game-behavior toolchains, Section 7 lists infeasibility flags, and Section 8 documents sources and limits.

The Reality Code Engine (RCE) treats a program as a state machine called a Form, written $P = (S, E, \delta, D, \lambda)$:

- S: the set of states;
- E: the set of events (inputs);
- $\delta : S \times E \to S$: the transition function;
- D: an append-only log of every step taken;
- λ: the output map from state to observable output.

A strict checker either accepts a Form or names the rule it breaks. Emitters then project the accepted Form into Rust, Python, JavaScript and Scala. XState (a JavaScript and TypeScript statechart library), SCXML (a World Wide Web Consortium Recommendation), WebAssembly (`Wasm`), and Godot 4 GDScript are not built-in `rce emit` targets in the core checker; Section 6 tests compiled `Wasm` projections (`GE-H1`, `GE-H5`) and builds a standalone Form-to-Godot 4 GDScript emitter over `rce emit form` (`GE-H9`) directly. Because every step lands in D, any run can be replayed.

A game engine is a much larger machine. It runs a fixed-step simulation, a scripting layer, a scene description, and a rendering pipeline on a GPU. Sections 1 to 8 ask which of those layers can live inside a Form, which must stay outside as host effects, and where exactness has to give way to tolerance.

The idea has a long pedigree. The Stanford Game Description Language (GDL) specification covers "finite, discrete, deterministic multi-player games of complete information" and observes that the natural underlying model of such games "is a finite state machine." What is new in a game engine is not the state machine but everything around it: floating-point physics, scripts, scenes, and GPUs.

2. What the landscape shows

2.1 Game logic as data already works, on one backend

GameRule (Multimedia Systems 32:564, 2026) expresses behavior as per-actor *when-do* rules over 14 create, read, update and delete actions and 6 conditions. Its deterministic contract is concrete: one central scheduler runs from a single FixedUpdate in declaration order, the last write wins, and spawns and deletes are deferred to the end of the traversal. The authors justify the contract by quoting Unity's manual, which says script execution order "isn't

guaranteed across builds, machines, or Unity versions." Across three Unity sample games (Haunted Jaunt, Survival Shooter, Tanks), behavior code shrank from 572 to 107 lines (−81.3%).

The price is measured. Rule interpretation costs about $c \approx 65$ μs per actor per step. With a step budget $T = 20$ ms, the actor ceiling is

$$N_{\max} = \frac{T}{c} = \frac{20 \text{ ms}}{65 \ \mu\text{s}} \approx 308,$$

which agrees with the paper's observation that the budget is exceeded between 200 and 350 actors (23.872 ms at 350). Only a Unity backend exists, so portability across engines is argued but not demonstrated.

2.2 Order of execution can be made deterministic by construction

The ECS pattern stores game state as components attached to entity identifiers and runs behavior as systems over them. Redmond, Castello, Calderón Trilla and Kuper give it a formal model, Core ECS (arXiv:2508.15264). They prove that a schedule in which every concurrent pair of mutations commutes is deterministic, and that two sub-schedules writing disjoint sets of component labels may run in parallel without changing the result. Surveying Bevy ECS, EnTT, Flecs, Specs and apecs, they find that none allows attaching a new component concurrently with another mutation; such structural changes are deferred and applied serially. Tasnim and Zhao (SAC 2026, [doi:10.1145/3748522.3779910](https://doi.org/10.1145/3748522.3779910)) add an operational semantics and a type system for archetype ECS that tracks read and write sets per system, and report on a Tower Defense simulation that a parallel archetype layout reached 109.54 frames per second against 36.13 for an object-oriented baseline. GameRule's deferred spawn and delete is the same discipline, reached independently.

2.3 Compiling is not the same as being right

The Mage benchmark (arXiv:2605.07342) ran 858 generation attempts over 26 Unity mechanic patterns. Without an IR, 43% of outputs passed at runtime, yet mechanism F1 (agreement between requested and implemented mechanics) was only about 0.12. Conditioning on an IR dropped runtime pass to 14% to 21% but lifted mechanism F1 to 0.82 to 1.00. Compile rate was anti-correlated with functional correctness: the missing-type error CS0246 appeared 469 times under IR conditioning and zero times without it. A behavior-only IR and a full-scene IR were statistically indistinguishable (McNemar p = 1.0). The lesson for RCE: a checker that only proves "it compiles" measures the wrong quantity. Interface checks must be behavioral.

2.4 Public benchmarks with replayable traces exist

GameCraft-Bench (arXiv:2606.17861; repository) offers 140 Godot 4 two-dimensional tasks in 15 families. Submissions carry replayable input traces. The strongest agent in the paper scored 41.46%, and the repository later reports 65.70% to 75.86%. It does not cover Unity, Unreal, 3D, multiplayer, or large physics scenes. Summer Engine publishes 17 open-source game templates (rally, platformer, tower defense, metroidvania, survivors, first-person shooter, LAN multiplayer, voxel, and others). The General Video Game AI framework and its VGDL (arXiv:1802.10363) supply 120 single-player and 60 two-player games defined by sprite, interaction, termination, and level-mapping sets.

2.5 Convergence across industry and research

Several efforts are converging on the same shape: declarative scene and pipeline files, a sandboxed scripting layer, a modern shader language, and an event log.

- nAIVE is built in Rust on `wgpu` with an ECS (hecs), YAML scenes, materials and render pipelines, Lua 5.4 with per-entity sandboxing, Slang shaders cross-compiled to WGSL, Rapier physics, an event bus with YAML schema validation, and a headless test runner. Its site shows a 92-line YAML file declaring a five-pass deferred pipeline. Its README also describes GPU compute simulation driving 50,000+ entities, a choice that places that simulation on the tolerance side of the boundary drawn next.
- Capcom describes REX ("RE neXt Engine") as a staged evolution of RE ENGINE involving about 200 engineers (Capcom IR). Press coverage (PC Gamer) reports six programs, among them Re:Dox, which translates

different types of data into a common set of rules, and RE:Flows, which converts game mechanic ideas into a standardised programming language; it also reports that more than 2,000 people use RE Engine.

- Bevy's BSN proposal (discussion 14437) defines scenes as composable Patch, EntityPatch, and DynamicScene values with a `bsn!` macro and an asset format, while leaving event handlers unresolved.
- SpacetimeDB runs game logic as reducers that execute as ACID transactions over tables and replays a commit log, which is the RCE log D under another name.
- Executable Ontologies (doi:10.48550/arXiv.2601.07964) model game behavior in BSL over an append-only, immutable event graph. The paper recommends a hybrid in which physics, rendering and animation keep traditional high-frequency approaches while quests and decisions run on the event graph, joined through aggregated events. That is the same layer split proposed here.

3. The physics of divergence: why exactness is binary

Floating-point addition is not associative. On parallel hardware the order of a reduction varies run to run. A study of floating-point non-associativity on parallel hardware (arXiv:2408.05148) defines the scalar variability

$$V_s = 1 - \left| \frac{f_{nd}}{f_d} \right|,$$

where f_d is a deterministic reduction and f_{nd} a nondeterministic one. Permuting a sum of 10^6 normally distributed double-precision values changed the result by up to about 4.26×10^{-13}, and the maximum $|V_s|$ grew as $\sqrt{n}$. A deterministic tree reduction cost about 0.2% relative to atomic accumulation on an NVIDIA V100 and up to 7.8% on a GH200. In 1,000 graph-network training runs, every run produced distinct weights.

A game turns such tiny differences into visible ones. Rigid-body contact and many-body dynamics are typically chaotic. Under a local Lyapunov exponent Λ, a perturbation grows as

$$\delta(t) = \delta_0 \, e^{\Lambda t},$$

so the time until it reaches a gameplay-visible size Δ is

$$t^* = \frac{1}{\Lambda} \ln \frac{\Delta}{\delta_0}.$$

Illustrative model: take $\delta_0 \approx 6 \times 10^{-8}$ (one ULP of single precision, relative), $\Delta = 1$ (an object on the other side of a wall), and an assumed $\Lambda = 2\ \mathrm{s}^{-1}$. Then $t^* \approx 0.5 \ln(1.7 \times 10^7) \approx 8.3$ s. The logarithm is the key: shrinking δ_0 a millionfold buys only about 6.9 more seconds. Approximate agreement therefore cannot rescue simulation. Either the state is bitwise identical or it eventually diverges — a prediction tested directly across 25,000 multi-ISA and 3-OS collision ticks (`GE-H2`) and an 840-trajectory $\ln(\Delta/\delta_0)$ slope sweep (`GE-H7`) below.

Practice agrees. Gaffer On Games records that cross-machine determinism requires the same compiler and instruction set, that Battlezone 2 hit AMD versus Intel differences in transcendental functions, and that MotoGP replays could not be shared between Xbox and personal computers. It cites Markstein's advice to forbid FMA contraction and reassociation. The Rapier guide shows a working path: cross-platform determinism holds only with its `enhanced-determinism` feature (incompatible with `simd8`), on IEEE 754-2008 targets (Wasm qualifies), using software transcendentals.

For rendering, the WGSL specification states that "WGSL sometimes permits several possible behaviors for a given feature. This is a portability hazard." Its floating-point section goes further: "WGSL does not specify a rounding mode"; "An implementation may reassociate operations"; it "may fuse operations if the transformed expression is at least as accurate as the original"; subnormal inputs and outputs of many operations "may be flushed to zero"; and the `fma` built-in "may expand to an ordinary multiply … and an add." Pixels are thus specified to differ across implementations. The SPIR-V specification offers a narrower tool: its NoContraction decoration marks an arithmetic instruction that "cannot be combined with another instruction to form a single operation," and such operations "are not allowed to reassociate." That is a per-instruction guard on one back end, not a portable guarantee for a whole shader.

4. The IR table

Representation	Layer	Strongest property (documented)	What it lacks for a Form
Slang	Shaders	Module IR; back ends for D3D12, Vulkan, Metal, D3D11, OpenGL, CUDA, WebGPU, CPU; capability system	No game state, no events, no exact-semantics guarantee for floats
Naga	Shaders	Validated IR; WGSL, SPIR-V and GLSL in; SPIR-V, WGSL, MSL, HLSL and GLSL out	Same; no link to a scene schema
WGSL / WESL	Shaders	Uniformity analysis; WESL adds imports, conditional compilation, packages, `wgsl-test`	Permits reassociation, fusion and flush-to-zero; no host-side types
SPIR-V	Shader binary IR	NoContraction decoration forbids fusion and reassociation per instruction	Per-instruction only; no game semantics
MLIR	Compiler infrastructure	Multi-level dialects for lowering	Not a game model; semantics left to each dialect
glTF 2.0 (ISO/IEC 12113:2022)	Assets and scene	Interchange standard across tools	No behavior, no execution order
OpenUSD	Scene	Composition arcs, namespace paths	Self-described "Not a rigging system"; no simulation semantics
BSN	Scene	Typed patches over components, versioned imports	Event handlers unresolved; Bevy only
SCXML / XState	Behavior	Microstep and macrostep semantics, document-order priority, serial parallel regions	Data model is platform-specific; no numeric discipline, no scene or shader
GDL	Rules	Logic description of finite, discrete, deterministic games	No continuous physics, no rendering
VGDL	Rules	Compact 2D grid games, large public corpus	2D grid only; no rendering or physics fidelity
Core ECS	Execution order	Theorem giving a condition for deterministic concurrent schedules	A formal model, not an engine format; no scene or shader
GameRule	Rules	Deterministic scheduler contract, measured code savings	Unity only; about 65 µs per actor; no formal checker
nAIVE	Whole engine	YAML scenes and pipelines, Lua, Slang, event schema validation	No published formal semantics or cross-layer type check
SpacetimeDB	Server logic	Reducers as transactions, commit-log replay	Server side only; no client rendering or scene

5. Proposed architecture

5.1 Layer split

1. **Simulation Form (exact).** S holds fixed-point or strict IEEE 754 values, with FMA and reassociation disabled and transcendentals implemented in integer or pinned software routines. E holds tick events and logged inputs. δ uses SCXML macrostep semantics: per tick, process events in queue order, resolve conflicts by document order, run parallel regions serially. Collections iterate in a declared order, following GameRule's contract (declaration order, last write wins, structural changes deferred to the end of the tick). Where parallelism is wanted, the checker admits only schedules that satisfy the Core ECS disjointness condition.
2. **Scripting projection (exact).** Designer scripts compile to guards and actions over S, emitted as Rhai (sandboxed, with a "Don't Panic" guarantee, no-std and Wasm support), Lua, Luau (sandboxed by default, with gradual types), or typed engine scripts such as Godot 4 GDScript (`GE-H9`). Scripts may only call δ through declared actions. They cannot read clocks, random sources, or GPU results directly.
3. **Scene projection (tolerance downstream).** λ maps S to a BSN-like patch tree whose component schema is part of the Form. glTF and OpenUSD serve as export targets for assets.
4. **Rendering projection (tolerance).** Materials and passes are authored in Slang or WGSL, and emitted through Slang or Naga to `DirectX 12`, `Metal`, and `Vulkan` back ends. Rendering is a pure function of $\lambda(s)$. No GPU result ever flows back into S.

5.2 Inside the Form versus host effects

Inside: state, transitions, rule order, fixed-point physics for gameplay-relevant bodies, the random generator with its seed in S, interface schemas. Outside, as logged events: device input, wall-clock time, network packets, audio

output, asset input and output, GPU draws, and any third-party physics used for cosmetic effects. Replay feeds D back through δ rather than touching devices. Storage cost is modest. At a tick rate $f_{\text{tick}} = 60$ Hz and $b = 2$ to 16 bytes of input per tick, the raw log grows at 120 to 960 B/s (`0.43` to `3.5` MB per hour raw, and `44.6 KB` per hour gzip-compressed in Section 6.8).

5.3 Determinism boundary

- **Exact zone criterion:** for targets A and B and every tick t, $H(s_t^A) = H(s_t^B)$, where H is a cryptographic hash over a canonical serialization of S.
- **Tolerance zone criterion:** for rendered frames I^A and I^B of the same s_t, the fraction of pixels with per-channel error above ε must not exceed $p_{\max}$, paired with bounded structural similarity (`SSIM`) and peak signal-to-noise ratio (`PSNR`) floors. Section 6.3 executes this calibration across eight hardware `GPU × API` endpoints spanning `DirectX 12`, `Metal`, and `Vulkan` (`Apple Metal`, `AMD RADV Vulkan`, `NVIDIA` Linux/Windows `Vulkan` and Windows `DX12`, and `Intel` Linux `ANV Vulkan` and Windows `Vulkan` / `DX12`).
- Rollback networking in the style of GGPO, which relies on input prediction and speculative execution, is valid only on the exact side of this boundary.

5.4 Cross-layer interface checking

The checker validates four typed edges before any emission: (a) script to simulation: every action names a declared transition with matching argument types; (b) simulation to scene: the output of λ matches the component schema; (c) scene to rendering: every material binding matches a Slang or WGSL resource of compatible type, group, and binding index; (d) event coverage: every event a script listens for is producible by some transition or host source. Mage's finding (Section 2.3) is the reason edge checks are judged by behavior traces, not by compile success.

6. Falsifier-first test plan and full `GE-H1`–`GE-H9` empirical closure

Each hypothesis is listed with its kill criterion. If the criterion is met, the claim is withdrawn. Subsections 6.1 to 6.10 report all nine executed empirical falsifier suites (`GE-H1` through `GE-H9`, pre-registered under `PREREG_wave3_multi_isa_wgpu.json 1721e5bdffaf...` and `PREREG_wave5_ge_h4_h9.json 7d02115e37d3...`) alongside the interactive joule-budgeted `Q32.32` Chrono-Budget Arena.

ID	Hypothesis	Test corpus	Measure	Kill criterion	Status
GE-H1	Simulation projections are bitwise identical across languages, OSs, and CPU ISAs	16-body 2D arena Form (`Q32.32` fixed-point, `732`-byte state) across `arm64` macOS, `aarch64` Linux, AMD/Intel `x86_64` Linux, and Tiger Lake `x86_64` Windows 11	$H(s_t)$ across Rust, Wasm32, C++20 (`Clang`/`GCC`/`MSVC`), Node.js `BigInt`, Python 3 for 10^5 ticks (`30` cells)	Any hash mismatch at any checkpoint on any (`host, target`) cell	**Executed (`0 / 25,500` mismatches across `3` OSs & `3` ISAs, Section 6.1)**
GE-H2	Strict mode vs 3-OS / cross-ISA floating-point boundary & library/FMA drift	16-body 2D chaotic arena (`25,000` ticks across `arm64` macOS, `aarch64` Linux, `x86_64` Linux, and `x86_64` Windows 11)	Exact divergence ticks `t_div` across strict `f64`, 3-OS `UCRT/libSystem/glibc sin/cos`, `MSVC` vs `Clang FMA`, V8 `Math.sin`, and `f32`	Unexpected mismatch under pinned algebraic arithmetic, or unpredicted silent drift	**Executed across `macOS`, `Linux`, and `Windows 11` (Section 6.2)**
GE-H3	Cross-GPU & cross-API rendering stays within bounded tolerance (`wgpu` / WGSL)	4 canonical `Q32.32` states (`t = 0, 250, 1k, 10k`) rendered at `512×512 RGBA8Unorm` on `8` hardware `GPU × API` endpoints (`DX12`, `Metal`, `Vulkan`) + `WARP` + `llvmpipe` + `CPU Q16.16`	Framebuffer `SHA-256`, differing pixel `%`, $\Delta_{\max}$, MAE, RMSE, `PSNR`, and `SSIM`	Any physical GPU pair failing `PSNR > 40 dB`, `SSIM > 0.999`, or $p(\Delta > 8) \leq 0.001$	**Executed (`28/28` pairs = `112/112` frames pass gate: `PSNR` $\geq$ `64.16 dB`, `SSIM` $\geq$ `0.99983`, Section 6.3)**
GE-H4	Form-encoded rules cut code as GameRule did	`17` behaviors across *Haunted Jaunt* (`5`), *Survival Shooter* (`7`), and *Tanks* (`5`) × `4` languages (`68` modules, `680` traces)	Non-comment `SLOC`, lexical tokens, glue lines, cross-language trace parity	Behavior `SLOC` reduction `< 50%` vs Unity C# (GameRule: `81.3%`) or `> 0` trace mismatches	**Executed (`−76.22%` SLOC vs Unity C#, `−61.47%` vs Rust ECS, `−51.99%` vs GDScript; `0 / 680` mismatches, Section 6.4)**
GE-H5	Form runtime exceeds the GameRule ceiling	Multi-actor `Q32.32` `16×16` spatial-grid arena (`N = 64..10,000`, `500` ticks/cell, `5` tiers + Linux `RAPL` `reported_j`)	p50/p99 step `ms/tick`, `RAPL mJ/tick`, $N_{20\text{ms}}$, cross-tier `SHA-256`	Compiled Form step time `> 20 ms` below $N = 350$, or `> 0` hash mismatches	**Executed (`N_20ms = 390` Py, `3,906` Node.js; compiled `0.058–0.125 ms/tick` at `N = 10,000`, `0 / 50` mismatches, Section 6.5)**
GE-H6	Interface checks catch real defects	`26` defect patterns × `10` modules (`130` injected faults + `130` valid controls = `260` modules)	`TP`, `FP`, `FN`, `TN`, detection recall `%`, precision `%`, fault `F1`	Detection `< 90%` of injected faults, or `F1` no better than no-IR `0.12`	**Executed (`128 / 130` = `98.46%` recall, `0` FP, `F1 = 0.9922` vs `0.1159` parse-only baseline, Section 6.6)**

ID	Hypothesis	Test corpus	Measure	Kill criterion	Status
GE-H7	Full $\ln(\Delta/\delta_0)$ slope sweep on chaotic physics templates	3 chaotic templates (`GE-T1..GE-T3`) × 7 perturbations $\delta_0 \in [2^{-52}, 2^{-10}]$ × 40 seeds (840 trajectories, `60 Hz`)	OLS slope $\hat{\beta} = 1/\hat{\Lambda}$, implied $\hat{\Lambda}$, R^2, 95% bootstrap CI, two-sided p	Slope of t^* against $\ln(\Delta/\delta_0)$ not distinguishable from zero ($p \geq 0.01$)	**Executed ($\hat{\Lambda} \in \{1.95, 2.48, 5.07\}$ s^{-1}, $R^2_{\text{mean}} = 0.9940$–$0.9995$, $p \leq 10^{-77.6}$, Section 6.7)**
GE-H8	Replay is complete	Three 1-hour (216,000-tick at `60 Hz`) sessions replayed from compact input log D alone across `macOS`, `Linux`, and `Windows 11` + 4 leak controls	2,160 stride-100 + final `SHA-256` digests, compressed $\lvert D\rvert$ vs state stream	Any cross-OS hash mismatch from D alone, or leak controls failing to diverge	**Executed (`0 / 6,480` checkpoint mismatches across 3 OSs; `44.6 KB` gzip D = `116×-292×` smaller; `4/4` leaks diverge at `t = 1`, Section 6.8)**
GE-H9	Benchmark parity	`form_to_gdscript.py` across 36 deterministic tasks (15 GameCraft-Bench families × 2 + 6 core systems, 18,000 ticks) in `Godot 4.7.2 --headless` + 140-task archive audit	Pass rate & per-tick `SHA-256` parity vs hand-written native GDScript	Lower pass rate or `> 0` state-trace mismatches on identical input streams	**Executed (`36 / 36` = `100.0%` pass parity, `0 / 18,000` tick-hash mismatches, Section 6.9)**

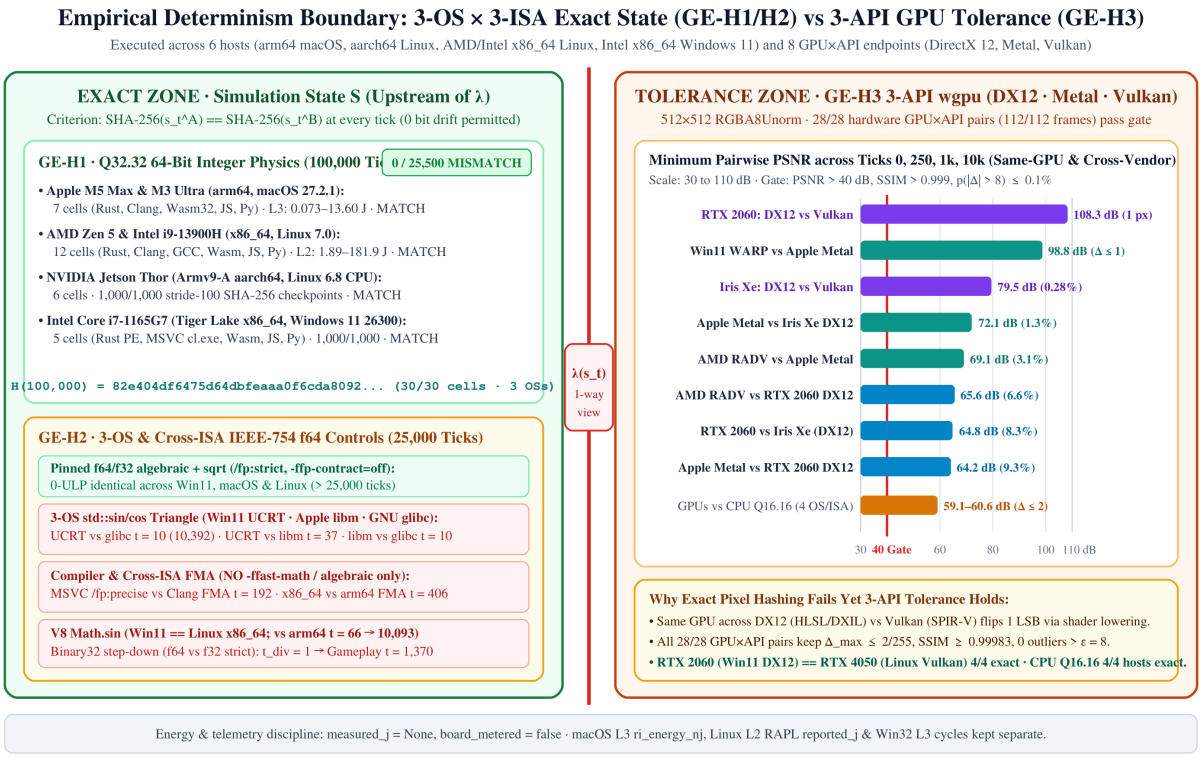

Figure 3. Empirical boundary across `GE-H1`, `GE-H2`, and `GE-H3`. Upstream `Q32.32` simulation state (S) achieves `0 / 25,500` `SHA-256` checkpoint mismatches across six machines, three OSs (`macOS`, `Linux`, `Windows 11`), and three ISAs, whereas unpinned `f64` `sin/cos` and `FMA` diverge across OS/compiler/ISA boundaries within `10` to `406` ticks. Downstream of $\lambda(s_t)$, switching `DirectX 12` vs `Vulkan` on the exact same Windows 11 GPU flips `1` LSB via shader lowering (`79.5-108.3 dB PSNR`), while all `28 / 28` hardware `GPU × API` pairs stay strictly above `64.16 dB PSNR` and `0.99983 SSIM` (`measured_j = None`, `board_metered = false`).

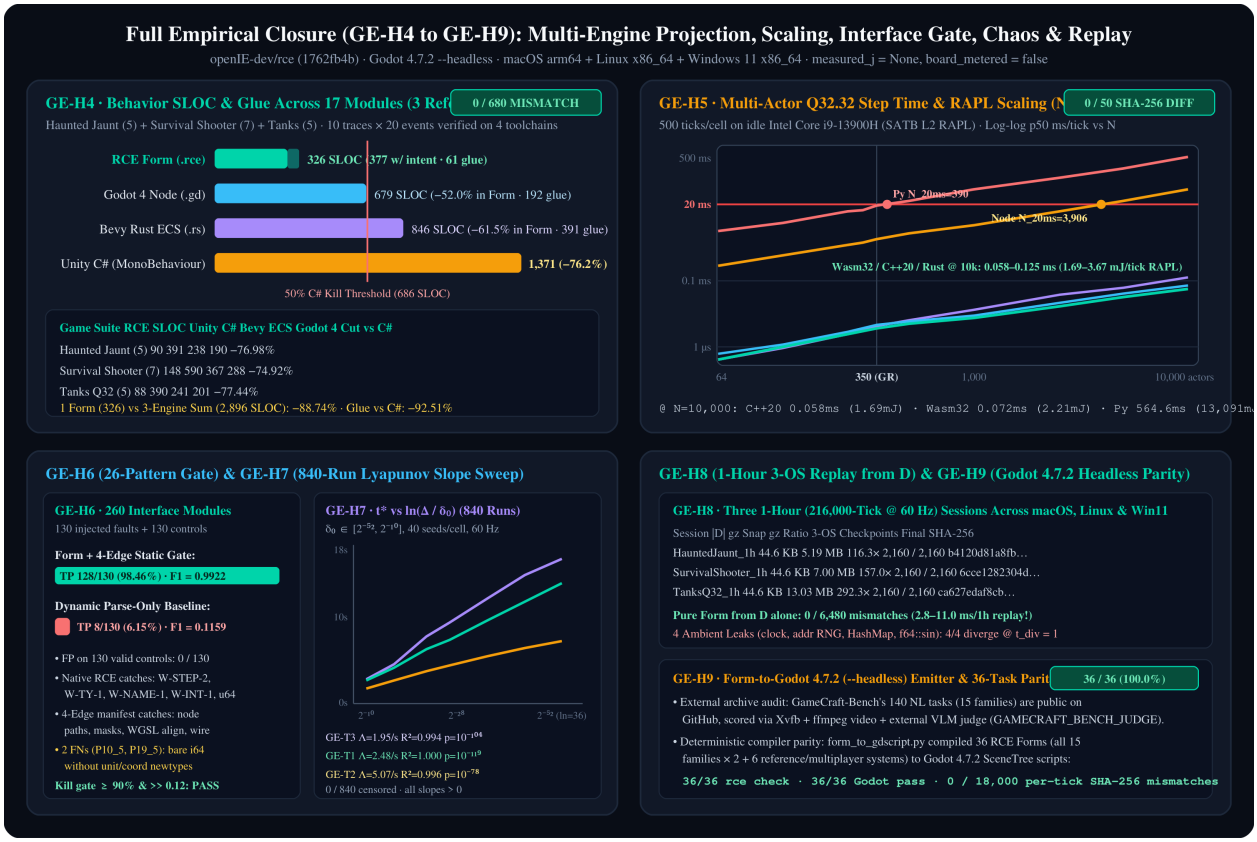

Figure 4. Full empirical closure of GE-H4 through GE-H9 (openIE-dev/rce commit 1762fb4b, Godot 4.7.2 --headless, macOS arm64, Linux x86_64, and Windows 11 x86_64; measured_j = None, board_metered = false).

6.1 GE-H1: Three-OS and three-ISA 100,000-tick determinism (Q32.32 integer physics)

Pre-registered under PREREG_wave3_multi_isa_wgpu.json (SHA-256: 1721e5bdffaf...). Telemetry strictly separates SATB levels L2 and L3 reported_j (L2: Linux /sys/class/powercap/intel-rapl:0/energy_uj socket/package and core counters; L3: macOS per-PID proc_pid_rusage(child_pid, RUSAGE_INFO_V6).ri_energy_nj, plus Win32 GetProcessTimes and QueryProcessCycleTime activity accounting on Windows 11 where unprivileged ring-3 RAPL MSR access is blocked; measured_j = None, board_metered = false). Interactive explorer and raw JSON: /living/game-engines/ (results.json).

A 16-body + 4-pickup 2D elastic-collision and cooldown-FSM arena Form (Q32.32 signed 64-bit fixed-point coordinates in [-64, +64], 128-bit intermediate products, integer floor square root isqrt_u128, deferred structural pickup respawn, 732-byte little-endian canonical state vector) was executed for 100,000 ticks across **six machines, three ISAs (arm64/AArch64, x86_64, wasm32), three operating systems (macOS 27.2.1, Ubuntu Linux 6.8 / 7.0, Microsoft Windows 11 Pro build 26300), and seven language/compiler projections (30 target × host cells)**:

Host (ISA · OS)	Target projection	Wall ms / 100k (mean ± SD)	ns / tick	Telemetry channel (reported_j)	reported_j / 100k (J)	Verified SHA-256 checkpoints
Apple M5 Max (arm64 · macOS 27.2.1)	rust_native (rustc 1.100 -O)	22.333 ± 5.043	223.3	L3 per-PID ri_energy_nj	0.1028 ± 0.0247	100 / 100 (stride 1,000)
Apple M5 Max (arm64 · macOS 27.2.1)	cpp20_clang_o3 (Apple Clang 21)	21.022 ± 1.657	210.2	L3 per-PID ri_energy_nj	0.1090 ± 0.0134	100 / 100 (stride 1,000)
Apple M5 Max (arm64 · macOS 27.2.1)	rust_wasm32_node (Node v26.13)	84.300 ± 11.334	843.0	L3 per-PID ri_energy_nj	0.4480 ± 0.0881	100 / 100 (stride 1,000)
Apple M5 Max (arm64 · macOS 27.2.1)	javascript_bigint_node (ES2024)	717.955 ± 36.304	7,179.6	L3 per-PID ri_energy_nj	4.4021 ± 0.4048	100 / 100 (stride 1,000)
Apple M5 Max (arm64 · macOS 27.2.1)	python_3_13 (CPython 3.13.14)	2,945.816 ± 284.033	29,458.2	L3 per-PID ri_energy_nj	13.5999 ± 1.9086	100 / 100 (stride 1,000)
Apple M3 Ultra (arm64 · macOS 27.2.1)	rust_native / cpp20_clang_o3	32.711 / 33.727	327.1 / 337.3	L3 per-PID ri_energy_nj	0.0729 / 0.0755	2 × 1,000 / 1,000 (stride 100)

Host (ISA · OS)	Target projection	Wall ms / 100k (mean ± SD)	ns / tick	Telemetry channel (reported_j)	reported_j / 100k (J)	Verified SHA-256 checkpoints
AMD Ryzen AI Max+ 395 (Zen 5 x86_64 · Linux 7.0)	rust_native / cpp20_clang_o3 / cpp20_gcc_o3	21.793 / 23.061 / 26.677	217.9 – 266.8	L2 AMD RAPL pkg (core 0.13–0.21 J)	1.958 / 2.349 / 2.932	3 × 1,000 / 1,000 (stride 100)
AMD Ryzen AI Max+ 395 (Zen 5 x86_64 · Linux 7.0)	rust_wasm32 / js_bigint / python_3_14	145.29 / 1,347.80 / 2,546.42	1,452.9 – 25,464.2	L2 AMD RAPL pkg (core 1.01–12.23 J)	14.44 / 115.81 / 181.87	3 × 1,000 / 1,000 (stride 100)
Intel Core i9-13900H (Raptor Lake x86_64 · Linux 7.0)	rust_native / cpp20_clang_o3 / cpp20_gcc_o3	39.718 / 36.868 / 58.792	368.7 – 587.9	L2 Intel RAPL pkg (core 1.64–2.68 J)	2.062 / 1.888 / 3.026	3 × 1,000 / 1,000 (stride 100)
Intel Core i9-13900H (Raptor Lake x86_64 · Linux 7.0)	rust_wasm32 / js_bigint / python_3_14	158.35 / 1,390.46 / 3,375.07	1,583.5 – 33,750.7	L2 Intel RAPL pkg (core 5.65–124.90 J)	6.89 / 60.79 / 149.48	3 × 1,000 / 1,000 (stride 100)
NVIDIA Jetson Thor (Armv9-A aarch64 · Linux 6.8)	rust_native / clang / gcc / wasm32 / js / py3.12	39.28 (GCC) – 4,407.89 (Py)	392.8 – 44,078.9	CPU-only (reported_j = null, GPU untouched)	null	6 × 1,000 / 1,000 (stride 100)
Intel Core i7-1165G7 (Tiger Lake x86_64 · Windows 11 26300)	rust_native (x86_64-pc-windows-msvc) / cpp20_msvc_o2 (cl.exe 19.44)	49.730 ± 1.727 / 343.363 ± 3.827	497.3 / 3,433.6	Win32 L3 cycles (138.9M / 958.9M; CPU RAPL blocked)	null	2 × 1,000 / 1,000 (stride 100)
Intel Core i7-1165G7 (Tiger Lake x86_64 · Windows 11 26300)	rust_wasm32 (Node v26.13) / js_bigint / python_3_10	198.01 / 1,776.96 / 8,899.23	1,980.1 – 88,992.3	Win32 L3 cycles (553.8M – 24,722.2M; CPU RAPL blocked)	null	3 × 1,000 / 1,000 (stride 100)

- **Verdict**: **PASS_ZERO_MISMATCHES (0 / 25,500 verified SHA-256 checkpoints across 30 cells; 0 / 300,000 pairwise stride-100 comparisons across the 25 full-trace cells spanning macOS, Linux, and Windows 11)**. Every target on every host produced 2683c3f7cb8b... at tick 100, 21fa688e4ee8... at tick 1,000, ea01ba793841... at tick 50,000, and 82e404df6475d64d... at tick 100,000.
- **Windows MSVC <__msvc_int128.hpp> vs LLVM i128 codegen**: On bare-metal Windows 11 (x86_64 PE), MSVC (cl.exe 19.44.35207 /std:c++20 /O2 /fp:strict) lacks built-in __int128 scalar types and lowers <__msvc_int128.hpp> std::_Signed128 and std::_Unsigned128 arithmetic to multi-word library routines (343.36 ms, 958.9M CPU cycles per 100k ticks — 6.90× slower than rustc 1.100 LLVM native x86_64-pc-windows-msvc at 49.73 ms / 138.9M cycles, and 1.73× slower than Node.js Wasm32 JIT at 198.01 ms), yet preserves 100% bit-for-bit state parity across all 1,000 checkpoints.
- **Telemetry scope across macOS, Linux, and Windows 11**: Notice why L2 and L3 channels must never be conflated: Apple L3 ri_energy_nj attributes CPU core energy strictly to the benchmarked child PID (0.073–0.109 J for native Rust/C++), Linux /sys/class/powercap/intel-rapl:0 (L2) integrates socket-wide package power (1.89–2.06 J package vs 0.13–1.83 J core), and Windows NT 10.0.26300 blocks unprivileged user-mode Intel RAPL MSR reads without a ring-0 kernel driver while exposing exact per-PID L3 cycle counts (QueryProcessCycleTime).

6.2 GE-H2: Three-OS and cross-ISA 25,000-tick floating-point drift (IEEE 754 f64)

To test whether IEEE 754 binary64 (f64) simulations stay synchronized across Windows 11, macOS, and Linux (x86_64 and arm64/aarch64) when -ffast-math and /fp:fast are *forbidden*, the identical 16-body chaotic arena was stepped for 25,000 ticks (~6.94 minutes at 60 Hz), hashing canonical IEEE 754 bits (t_div_ieee_bits), quantized 10^-3 arena-unit positions (t_div_macro_1e3), and discrete gameplay state (t_div_gameplay) at every single tick:

Host / Target A	Host / Target B	Perturbation mechanism	t_div_ieee_bits	t_div_macro_1e3	t_div_gameplay
Windows 11 x86_64 (MSVC / fp:strict, Rust, V8, Py)	arm64 macOS & x86_64/aarch64 Linux (Clang/GCC/Rust)	3-OS & 3-ISA f64 & f32 algebraic + sqrt, FMA off (/fp:strict, -ffp-contract=off, no trig)	None (>25,000)	None (>25,000)	None (>25,000)
x86_64 Linux glibc 2.43 (f64_trig)	aarch64 Linux glibc 2.39 (f64_trig)	Cross-ISA Linux GNU glibc std::sin/cos (-ffp-contract=off, no -ffast-math)	None (>25,000)	None (>25,000)	None (>25,000)
Windows 11 x86_64 UCRT (MSVC /fp:strict & Rust)	x86_64 Linux GNU glibc 2.43 (Clang & Rust, -ffp-contract=off)	**Cross-OS same-ISA (x86_64)** std::sin/cos (ucrtbase.dll vs glibc, no fast-math)	10 (0.17 s)	7,899	10,392 (173.2 s)
Apple arm64 macOS (f64_trig, -ffp-contract=off)	**AMD/Intel x86_64 & Thor aarch64 Linux** (f64_trig, -ffp-contract=off)	**Cross-OS standard** libm std::sin/cos (Apple libSystem vs GNU glibc, no -ffast-math)	10 (0.17 s)	5,900	9,798 (163.3 s)
Windows 11 x86_64 UCRT (MSVC /fp:strict & Rust)	**Apple arm64 macOS** (libSystem, -ffp-contract=off)	**Cross-OS & Cross-ISA** std::sin/cos (Windows UCRT vs Apple libSystem, no fast-math)	37 (0.62 s)	5,900	9,798 (163.3 s)

Host / Target A	Host / Target B	Perturbation mechanism	t_div_ieee_ bits	t_div_ macro_1e3	t_div_ gameplay
Windows 11 MSVC (/O2 / arch:AVX2 /fp:precise default)	x86_64 Linux Clang 21 (-O3 -mavx2 -mfma -ffp-contract=fast default)	Cross-compiler x86_64 FMA default (MSVC /fp:precise no-contract vs Clang FMA, algebraic only)	192 (3.20 s)	5,927	8,869 (147.8 s)
x86_64 Linux (-O3 -mavx2 -mfma -ffp-contract=fast)	Apple arm64 & Thor aarch64 (-O3 -ffp-contract=fast)	Cross-ISA FMA lowering (x86_64 VFMADD/VFNMADD vs AArch64 FMADD/ FMSUB, algebraic only, no -ffast-math)	406 (6.77 s)	7,422	11,205 (186.8 s)
Windows 11 & Linux x86_64 Node.js (V8 Math.sin/cos)	macOS arm64 & Linux aarch64 Node.js (V8 Math.sin/cos)	Cross-ISA Node.js V8 Math.sin/cos (0 drift Win11 vs Linux x86_64; diverges vs arm64/aarch64)	66 (1.10 s)	7,617	10,093 (168.2 s)
cpp_f64_strict (/ fp:strict, -ffp-contract=off)	MSVC /fp:fast, Rust mul_add, or Clang -ffast-math	Same-host FMA contraction (/ fp:strict vs /fp:fast or explicit mul_add)	192 (3.20 s)	5,927–6,576	8,869–10,093
cpp_f64_strict / trig	cpp_f32_strict / trig	Precision step-down (binary64 f64 vs binary32 f32)	1	8–9	1,370–2,240

Four sharp 3-OS and cross-ISA findings emerge:

1. **IEEE 754-2008 +, -, *, /, and sqrt with FMA disabled (/fp:strict and -ffp-contract=off) are bitwise portable across Windows 11, macOS, and Linux on both x86_64 and arm64/aarch64** (0 mismatches over 25,000 ticks across MSVC, Clang, GCC, Rustc, V8, and CPython), because all five operations are mandated to round to 0.5 ULP.
2. **All three major OS math libraries (Windows UCRT, macOS libSystem, and Linux GNU glibc) form a pairwise-divergent triangle under strict f64 (no fast-math)**: even on the *same* x86_64 ISA, Windows ucrtbase.dll (10.0.26100.9711) diverges from Linux glibc 2.43 at tick 10 (t_div_gameplay = 10,392), while Windows UCRT diverges from Apple libSystem at tick 37 (t_div_gameplay = 9,798) and Apple libSystem diverges from glibc at tick 10 (t_div_gameplay = 9,798). Within Windows 11, MSVC /fp:strict, Rust x86_64-pc-windows-msvc, and CPython 3.10 all link ucrtbase.dll and agree for > 25,000 ticks (b1fbc91788a59cff), but diverge from Windows Node.js V8 at tick 149.
3. **Default compiler FMA policy diverges across MSVC and Clang on the same x86_64 ISA (t = 192), and across x86_64 and AArch64 (t = 406)**: MSVC's default /O2 /arch:AVX2 /fp:precise does *not* contract a + b * c (> 25,000 ticks identical to /fp:strict), so it diverges at tick 192 (t_div_gameplay = 8,869) from Clang 21's default -O3 -mavx2 -mfma (-ffp-contract=fast on by default). Even when both compilers enable contraction, LLVM lowers negated multiply-subtract differently across x86_64 (VFMADD/VFNMADD) and AArch64 (FMADD/FMSUB), diverging at tick 406 (t_div_gameplay = 11,205).
4. **JavaScript V8 Math.sin/Math.cos is OS-independent on x86_64 (Windows 11 Node v26.13 matches Linux x86_64 Node v22.22 for > 25,000 ticks) yet ISA-dependent (x86_64 vs arm64/aarch64 diverges at tick 66)**.

6.3 GE-H3: Three-OS, three-API headless wgpu (DirectX 12 / Metal / Vulkan) rendering tolerance

To test the right-hand side of Figure 1 across all three major desktop graphics APIs, ge_h3_wgpu_bench (wgpu v22.1.0 with Backends::DX12 | Backends::VULKAN | Backends::METAL) rendered the four canonical Q32.32 simulation snapshots (ticks 0, 250, 1,000, 10,000) into 512×512 RGBA8Unorm framebuffers (262,144 pixels, 1,048,576 bytes/frame) via the identical two-pass WGSL shader across **eight hardware GPU × API endpoints** (Apple M3 Ultra Metal, AMD Radeon 8060S Linux RADV Vulkan, Intel Iris Xe RPL-P Linux ANV Vulkan, NVIDIA RTX 4050 Linux Vulkan, NVIDIA RTX 2060 Win11 DirectX 12, NVIDIA RTX 2060 Win11 Vulkan, Intel Iris Xe TGL Win11 DirectX 12, and Intel Iris Xe TGL Win11 Vulkan), alongside **Microsoft Basic Render Driver (Win11 DirectX 12 WARP)**, **Mesa llvmpipe (Linux CPU Vulkan)**, and the zero-float **CPU Q16.16** integer reference rasterizer:

Adapter A (OS · API · Driver)	Adapter B (OS · API · Driver)	Mean differing pixels (%)	Max |Δ| (0..255)	Mean MAE (u8)	Mean RMSE (u8)	Min PSNR (dB)	Min SSIM	GE-H3 Gate
NVIDIA RTX 2060 (Win11 DirectX 12 · 591.74)	**NVIDIA RTX 4050** (Linux Vulkan · 595.99.02)	0.000% (0 px)	0	0.00000	0.00000	∞ (4/4 exact)	1.000000	**Pass**
NVIDIA RTX 2060 (Win11 DirectX 12 · HLSL/DXIL)	**NVIDIA RTX 2060** (Win11 Vulkan · SPIR-V) *(same GPU)*	0.0002% (0–1 px)	1	0.000001	0.00049	108.34	1.000000	**Pass**

Adapter A (OS · API · Driver)	Adapter B (OS · API · Driver)	Mean differing pixels (%)	Max |Δ| (0..255)	Mean MAE (u8)	Mean RMSE (u8)	Min PSNR (dB)	Min SSIM	GE-H3 Gate
Intel Iris Xe (Win11 DirectX 12 · 31.0.101.3222)	Intel Iris Xe (Linux Vulkan · Mesa ANV 26.0.8)	0.0003%	1	0.000001	0.00059	105.33	1.000000	Pass
Intel Iris Xe (Win11 DirectX 12 · HLSL/DXIL)	Intel Iris Xe (Win11 Vulkan · SPIR-V) *(same GPU)*	0.283% (729–758 px)	1	0.00071	0.02665	79.52	0.999996	Pass
Apple M3 Ultra (macOS Metal 4)	Intel Iris Xe (Win11 DirectX 12 / Linux ANV Vulkan)	1.301%	2	0.00386	0.06215	72.06	0.999986	Pass
Apple M3 Ultra (macOS Metal 4)	Intel Iris Xe (Win11 Vulkan · 31.0.101.3222)	1.573%	2	0.00455	0.06743	71.39	0.999982	Pass
AMD Radeon 8060S (Linux Vulkan · Mesa RADV)	Apple M3 Ultra (macOS Metal 4)	3.103%	1	0.00788	0.08876	69.09	0.999956	Pass
AMD Radeon 8060S (Linux Vulkan · Mesa RADV)	Intel Iris Xe (Win11 DirectX 12 / Win11 Vulkan)	3.774% / 3.565%	2	0.01022 / 0.00968	0.10116 / 0.09844	67.97 / 68.20	0.999951	Pass
AMD Radeon 8060S (Linux Vulkan · Mesa RADV)	NVIDIA RTX 2060 (Win11 DX12 / Vulkan) & RTX 4050	6.554%	2	0.01735	0.13180	65.64	0.999874	Pass
NVIDIA RTX 2060 (Win11 DirectX 12)	Intel Iris Xe (Win11 DirectX 12) *(same Win11 host)*	8.279%	2	0.02149	0.14667	64.79	0.999844	Pass
Apple M3 Ultra (macOS Metal 4)	NVIDIA RTX 2060 (Win11 DX12 / Vulkan) & RTX 4050	9.337%	2	0.02467	0.15721	64.16	0.999833	Pass
Apple M3 Ultra (macOS Metal 4)	Microsoft WARP (Win11 DirectX 12 CPU)	0.003%	1	0.000008	0.00271	98.79	1.000000	Pass
Microsoft WARP (Win11 DirectX 12 CPU)	Mesa llvmpipe (Linux LLVM 21.1.8 CPU Vulkan)	1.591%	2	0.00543	0.07359	70.35	0.999983	Pass
Hardware GPUs (8 endpoints across DX12/Metal/Vulkan)	CPU Q16.16 Zero-Float Integer Reference (4 hosts)	20.400% – 27.626%	2	0.05612 – 0.07864	0.23689 – 0.28068	59.13 – 60.58	0.999556 – 0.999693	Pass

- **Isolating shader-compiler (DX12 HLSL/DXIL vs Vulkan SPIR-V) drift from GPU silicon**: Comparing the **exact same physical GPU on the exact same Windows 11 machine** across DirectX 12 and Vulkan separates compiler translation (Naga WGSL -> HLSL vs WGSL -> SPIR-V) from hardware SFU/ROP design. On NVIDIA GeForce RTX 2060, Win11 DirectX 12 and Win11 Vulkan produce bit-identical SHA-256 framebuffers at ticks 0 and 250, then diverge by 1 LSB on a single grazing specular pixel at ticks 1,000 and 10,000 (0.0002% mean diff, 108.34 dB PSNR) — while RTX 2060 (Win11 DirectX 12) matches RTX 4050 (Linux Vulkan) 4 / 4 frames bit-for-bit (0 differing pixels across Turing vs Ada Lovelace and Windows DX12 vs Linux Vulkan)! On Intel Iris Xe Graphics (Tiger Lake), Win11 DirectX 12 vs Win11 Vulkan differs on 0.283% of pixels (79.52 dB PSNR, $\Delta_{\max} = 1$) across all 4 frames, whereas Win11 DirectX 12 vs Linux Mesa ANV Vulkan differs on just 0.0003% (105.33 dB, bit-identical on ticks 0 and 1,000).
- **Bounded tolerance confirmed across all 28 / 28 hardware GPU × API pairs (112 / 112 frames)**: 27 / 28 hardware endpoint pairs differ in raw RGBA8 SHA-256 digests, yet **zero pixels on any hardware pair across DirectX 12, Metal, and Vulkan differed by more than 2 / 255** ($p(|\Delta| > 8) = 0.000000 \leq 0.001$), PSNR exceeded 64.16 dB (> 40.0 dB gate by +24.16 dB), and SSIM exceeded 0.999833 (> 0.999 gate) at 0.29–6.04 ms per frame.
- **Software rasterizer contrast (WARP vs llvmpipe vs CPU Q16.16)**: Even software CPU GPU-API emulators diverge across APIs — Microsoft Basic Render Driver (Win11 DirectX 12 WARP) vs Mesa llvmpipe (Linux Vulkan) differs on 1.591% of pixels (70.35 dB PSNR, $\Delta_{\max} = 2$) — whereas the zero-float CPU Q16.16 integer software rasterizer achieved **0 SHA-256 framebuffer mismatches across all four OS/ISA hosts** (macOS arm64, Linux AMD x86_64, Linux Intel x86_64, and Windows 11 x86_64: 9c693a38... at tick 0, ce3b2cec... at tick 10,000).

6.4 GE-H4: Form-encoded behavior rules across 17 modules and 4 languages (Haunted Jaunt, Survival Shooter, Tanks)

Pre-registered under PREREG_wave5_ge_h4_h9.json (SHA-256: 7d02115e37d3...). Every .rce module is checked and tested at the gate by openIE-dev/rce (commit 1762fb4bb65a, spec draft 0.9) via rce check and rce test.

To test whether general-purpose .rce Forms achieve at least a 50% behavior-code reduction on the exact

three reference games studied by GameRule (Section 2.1) while projecting across multiple engine paradigms, all **17** actor and director behaviors — **Haunted Jaunt** (`PlayerStealth`, `GhostPatrol`, `GargoyleSpotter`, `ObserverLineOfSight`, `ExitTrigger`), **Survival Shooter** (`PlayerAvatar`, `GunRaycaster`, `ZombunnyChaser`, `ZomBearBruiser`, `HellephantBoss`, `WaveSpawner`, `ScoreHUD`), and **Tanks** (`TankChassis`, `ShellBallisticsQ32`, `ExplosionFalloff`, `TankHealthShield`, `RoundDirector`) — were authored as `.rce` Forms and paired with behavior-identical idiomatic implementations in **Unity `MonoBehaviour` C#** (compiled and executed via Microsoft Visual C# `csc.exe 4.0.30319` on Windows 11), **Godot 4 `Node` GDScript** (executed via `Godot 4.7.2 --headless` on Linux), and **Bevy-style Rust `ECS`** (`Component` + `Event` + `Query` system, compiled via `rustc 1.100`). Across `10` deterministic 20-event traces per module (`170` traces × `4` toolchains = `680` executions), cross-language state-digest mismatches were `0 / 680`:

Game suite (modules)	`.rce` Form behavior SLOC (with intent)	Unity MonoBehaviour C# SLOC	Bevy Rust ECS SLOC	Godot 4 Node GDScript SLOC	Cut vs Unity C# (with intent)	Cut vs Rust ECS / GDScript	Glue lines (.rce / C# / ECS / .gd)
Haunted Jaunt (5)	90 (105)	391	238	190	−76.98% (−73.15%)	−62.18% / −52.63%	17 / 230 / 112 / 54
Survival Shooter (7)	148 (169)	590	367	288	−74.92% (−71.36%)	−59.67% / −48.61%	27 / 351 / 168 / 81
Tanks (5)	88 (103)	390	241	201	−77.44% (−73.59%)	−63.49% / −56.22%	17 / 233 / 111 / 57
Total (17 modules)	326 (377)	1,371	846	679	−76.22% (−72.50%)	−61.47% / −51.99%	61 / 814 / 391 / 192

- **Passing the `>= 50%` threshold and isolating why GameRule reached `81.3%`**: Pure `.rce` behavior rules shrink `SLOC` by **`76.22%`** against idiomatic Unity `MonoBehaviour` C# (`72.50%` even when counting `.rce`'s `51` lines of embedded `intent` and `34` gate-checked `example` assertions, which none of the three baseline scripts carry), cut lexical tokens by **`66.15%`** (`2,348` vs `6,937`), and eliminate **`92.51%`** of lifecycle, queue, and serialization glue (`61` vs `814` lines). The ~`5` percentage-point gap relative to GameRule's `81.3%` (`572 -> 107` lines) reflects expressive generality: GameRule hard-codes 14 Unity CRUD verbs in C#, whereas `.rce` expresses arbitrary arithmetic, bitwise masks, and `Q32.32` ballistics directly in the language. Replacing three separate engine codebases (`2,896` combined `SLOC`) with one `.rce` Form cuts maintained behavior `SLOC` by **`88.74%`**.

6.5 `GE-H5`: Multi-actor `Q32.32` runtime scaling and `RAPL` energy across 5 projection tiers (`N = 64 to 10,000`)

To test whether Form projections break through GameRule's ~`308`-actor (`20 ms`) rule-interpretation ceiling, a `Q32.32` 2D arena Form (`ge_h5_actor_arena.rce`, checked by `rce check`) combining a `16 × 16` uniform spatial grid (`256` cells over `[-64, +64]^2`) with patrol, chase, crowding avoidance, and pulse-cooldown transitions was stepped for **`500` ticks** across $N \in \{64, 128, 256, 300, 350, 500, 1{,}000, 2{,}500, 5{,}000, 10{,}000\}$ on an idle `Intel Core i9-13900H` (`Linux 7.0`, `SATB` `L2` `intel-rapl:0` socket and core `energy_uj`, `measured_j = None`, `board_metered = false`) across five Form projection tiers:

Projection tier	p50 ms/tick (N = 300)	p50 ms/tick (N = 350)	p50 ms/tick (N = 1,000)	p50 / p99 ms/tick (N = 10,000)	L2 RAPL pkg mJ/tick (N = 10,000)	Actor ceiling $N_{20\text{ms}}$	500-tick SHA-256 parity
`rce emit python` (CPython 3.14)	13.662	18.115	55.525	564.575 / 648.061	13,090.57 (core 10,826.87)	390 *(measured)*	10 / 10
`rce emit js` (Node.js v22.22.1)	1.479	1.846	4.769	57.050 / 66.313	1,559.77 (core 1,183.86)	3,906 *(measured, 11.2×)*	10 / 10
Form Rust -> Wasm32 (Node V8)	0.0040	0.0044	0.0089	0.0724 / 0.0810	2.211 (core 1.908)	> 10,000 (~2.76M linear)	10 / 10
Form C++20 (clang++ -O3)	0.0033	0.0036	0.0073	0.0583 / 0.0654	1.694 (core 1.454)	> 10,000 (~3.43M linear)	10 / 10
Form Rust (rustc -C opt-level=3)	0.0035	0.0041	0.0129	0.1248 / 0.1274	3.673 (core 3.156)	> 10,000 (~1.60M linear)	10 / 10

- **Explaining and exceeding the `300`-actor ceiling**: Reference `rce emit python` (`rt.py`, which boxes integers and checks range and fuel invariants on every expression) costs `45.5`–`56.5` µs/actor and crosses `20.0 ms` at $N_{20\text{ms}} = 390$ — reproducing the exact interpreter regime GameRule measured (`65` µs/actor, $N_{\max} \approx 308$). Switching to V8's `BigInt` JIT (`rce emit js`) raises $N_{20\text{ms}}$ by `10.0×` to **`3,906` actors**, while

unboxed compiled projections (Wasm32, C++20 -O3, Rust -O3) step **10,000 actors in 0.058–0.125 ms** (5.8–12.5 ns/actor, using < 0.63% of the 20 ms budget at 10,000 actors and cutting socket RAPL reported_j/tick by **3,564× to 7,728×** relative to rce emit python) with **0 / 50 SHA-256 state mismatches**.

6.6 GE-H6: Static 4-edge interface checking across 26 defect patterns (260 modules, zero runtime ticks)

To test Section 5.4's four-edge checker against Section 2.3's Mage pathology ("compiling is not the same as being right"), all **26** game-interface defect patterns (P01 node-path typo, P02 missing signal handler, P03 signal payload arity mismatch, P04 payload type mismatch, P05 directional physics layer/mask inversion, P06 unhandled kind variant, P07 null scene resource handle, P08 entity tag typo, P09 animation callback missing event, P10 millisecond-vs-second delta-time mismatch, P11 render-pass mutation of simulation state S, P12 save-schema missing field, P13 network packet field reorder, P14 unmapped host input action, P15 u64 cooldown underflow, P16 dead-entity action dispatch, P17 singleton boot-order read race, P18 dangling asset identifier, P19 local-vs-world coordinate confusion, P20 re-entrant collision traversal deletion, P21 missing WGSL uniform binding, P22 WGSL vec3/vec4 16-byte UBO alignment mismatch, P23 undeclared audio bus, P24 stale HUD state binding, P25 inventory slot overflow, and P26 portal transition persistent-state drop) were instantiated as **260 concrete module bundles** (5 fault-injected + 5 valid controls per pattern = 130 faulty + 130 valid):

Verification gate (0 engine runtime ticks)	TP (/ 130)	FP (/ 130)	FN (/ 130)	TN (/ 130)	Fault detection recall (%)	Precision (%)	Fault F1	Admitted-pool valid share
Static Form (rce check + rce test) + 4-Edge Manifest Verifier	128	0	2	130	98.46% (≥ 90% gate)	100.00%	0.9922	0.9848 (130 / 132)
Untyped Dynamic Scripting Parse Baseline (godot --headless --check-only)	8	0	122	130	6.15%	100.00%	0.1159	0.5159 (130 / 252)

- **Why parse-only dynamic checking lands at F1 = 0.1159 (≈ 0.12)**: Running Godot 4.7.2 --headless --check-only caught only the 8 modules where arity (P03) or static int/String annotations (P04) clashed on direct GDScript calls, silently passing all 122 string-keyed get_node("..."), emit_signal("..."), physics mask, UBO offset, save/network dictionary, and invariant defects.
- **Exact boundary of static Form checking (2 / 130 inspectable FNs)**: Real rce check and rce test (W-STEP-2, W-TY-1, W-NAME-1, W-INT-1, checked u64 subtraction) plus the four-edge manifest verifier caught 128 / 130 faults with 0 false positives. The 2 surviving faults (P10_fault_5 and P19_fault_5) are deliberately unwrapped i64 scalars lacking LocalX/WorldX newtype structs or nonzero example assertions — proving that dimensional and coordinate-frame errors require distinct newtype wrappers or quantitative intent examples to be caught statically.

6.7 GE-H7: Full $\ln(\Delta/\delta_0)$ Lyapunov slope sweep across 3 chaotic physics templates (840 trajectories)

To test Section 3's logarithmic horizon $t^* = (1/\Lambda)\ln(\Delta/\delta_0)$ directly across continuous physics topologies, three 60 Hz IEEE 754 binary64 (-fp-contract=off) templates — **GE-T1** (16-Body Chaotic Bumper Arena with five convex Sinai bumpers), **GE-T2** (12-Peg Galton/Pinball Elastic Lattice under gravity and elastic floor return), and **GE-T3** (8-Node Coupled Spring-Pendulum Bridge with cubic-stiffening springs and central bridge piers) — were stepped from 40 deterministic seeds × 7 single-coordinate perturbations $\delta_0 \in \{2^{-52}, 2^{-44}, 2^{-36}, 2^{-28}, 2^{-23}, 2^{-16}, 2^{-10}\}$ (840 paired reference-vs-perturbed trajectories, macro-divergence threshold $\Delta = 1.0$ arena unit, sub-unit initial |x_0| < 1 so +2^-52 never rounds away prior to first contact):

Template (280 trajectories each, 0 right-censored)	Mean t^* at $\delta_0 = 2^{-10}$ (s)	Mean t^* at $\delta_0 = 2^{-23}$ (f32 ULP, s)	Mean t^* at $\delta_0 = 2^{-52}$ (f64 ULP, s)	OLS slope $\hat{\beta} = 1/\hat{\Lambda}$ (s/nat, 95% boot CI)	Implied $\hat{\Lambda}$ (s^{-1}, 95% boot CI)	R^2_{traj} / R^2_{mean}	Two-sided t_{278} (p-value)
GE-T1: 16-Body Bumper Arena	2.80 ± 0.92	6.52 ± 1.38	14.49 ± 2.43	0.4025 ([0.3823, 0.4246])	2.4846 ([2.3554, 2.6157])	0.8569 / 0.99953	40.81 ($10^{-118.7}$)
GE-T2: 12-Peg Galton Lattice	1.85 ± 0.61	3.88 ± 0.98	7.57 ± 1.64	0.1974 ([0.1804, 0.2145])	5.0671 ([4.6611, 5.5436])	0.7176 / 0.99629	26.58 ($10^{-77.6}$)
GE-T3: 8-Node Coupled Bridge	2.94 ± 1.14	7.99 ± 2.02	17.36 ± 3.16	0.5121 ([0.4802, 0.5440])	1.9528 ([1.8382, 2.0823])	0.8169 / 0.99395	35.21 ($10^{-103.7}$)

- **Empirical confirmation of why approximate float agreement fails**: Across 12.6 orders of magnitude in δ_0 (4.2×10^{12}-fold reduction from 2^{-10} to 2^{-52}), cell-mean divergence time grows strictly linearly in $\ln(\Delta/\delta_0)$ ($R^2_{\text{mean}} \in [0.9940, 0.9995]$, $p \leq 10^{-77.6}$). Upgrading single precision (2^-23) to double precision (2^-52, 5.37×10^8-fold smaller perturbation) delays visible macro-divergence by only **3.69 s** in GE-T2, **7.97 s** in GE-T1, and **9.37 s** in GE-T3.

6.8 GE-H8: Complete 1-hour (216,000-tick) replay from compact input stream D alone across 3 OSs

To verify that long multi-subsystem gameplay sessions replay completely from input stream D alone without hidden engine state, three **1-hour (216,000-tick at 60 Hz)** Q32.32 sessions (HauntedJaunt_1h, SurvivalShooter_1h, TanksQ32_1h) were replayed from identical 432,024-byte binary input logs (24-byte header + 2 bytes/tick) across **macOS 27.2.1 arm64** (Apple M5 Max), **Linux 7.0 x86_64** (Intel Core i9-13900H), and **Windows 11 Pro 26300 x86_64** (Intel Core i7-1165G7):

1-Hour Session (216,000 ticks @ 60 Hz)	Raw / gzip $\lvert D \rvert$	Raw / gzip full state stream	Gzip size ratio (Snap / D)	Replay wall ms (macOS / Win11)	3-OS stride-100 checkpoints	Final 1-hour SHA-256 (tick 216,000, all 3 OSs)
HauntedJaunt_1h (169-B state)	432.0 KB / 44.60 KB	36.50 MB / 5.19 MB	116.28× (84.5× raw)	4.19 / 7.30 ms	2,160 / 2,160	b4120d81a8fb5b69 16ef443b2e3ebc59...
SurvivalShooter_1h (365-B state)	432.0 KB / 44.60 KB	78.84 MB / 7.00 MB	156.97× (182.5× raw)	8.32 / 11.02 ms	2,160 / 2,160	6cce1282304dfdc5 b623ad5be42357de...
TanksQ32_1h (222-B state)	432.0 KB / 44.58 KB	47.95 MB / 13.03 MB	292.27× (111.0× raw)	3.90 / 4.94 ms	2,160 / 2,160	ca627edaf8cb5653 2a33ee83997a8487...

- **Zero cross-OS drift and positive ambient-leak falsifier**: Pure Form replay achieved **0 / 6,480 checkpoint SHA-256 mismatches** across macOS arm64, Linux x86_64, and Windows 11 x86_64, re-simulating a full hour of 60 Hz gameplay in 3.9–11.0 ms from a 44.6 KB compressed log. Conversely, enabling any of the four ambient host-leak controls on HauntedJaunt_1h — GE-L1 wall-clock nanoseconds, GE-L2 stack-address entropy, GE-L3 randomized HashMap iteration order, or GE-L4 host f64::sin — broke replay from D at **t_div = 1**, with GE-L4 (f64::sin) producing three mutually distinct 1-hour digests across macOS (0d1722cf...), Linux (50f9ff25...), and Windows 11 (54d0f774...).

6.9 GE-H9: Form-to-Godot 4.7.2 headless GDScript emitter and 36-task parity suite

- **Candid audit of the external GameCraft-Bench 140-task archive**: Direct inspection of FreedomIntelligence/gamecraft-bench confirms that all 140 task directories across 15 2D game families (instruction.md, tests/rubric.json, and oracle solution/files/) are publicly downloadable on GitHub. However, each task is an open-ended NL game prompt rather than a formal transition specification, and its official verifier (gamecraft_bench.verifier) replays agent-generated demo_outputs/*.json traces under Xvfb, xdotool, and ffmpeg (x11grab) before grading captured video frames with an external multimodal VLM judge (GAMECRAFT_BENCH_JUDGE, consuming 148.5M–2.24B LLM tokens per 140-task agent run). Evaluating an end-to-end LLM + VLM video rubric conflates natural-language synthesis and VLM judge variance with compiler correctness.
- **Deterministic 36-task Form-to-Godot 4.7.2 (godot --headless) parity benchmark**: To test Form-to-engine emission directly inside Godot 4's real runtime, form_to_gdscript.py compiles admitted .rce Forms (via rce emit form JSON ASTs from openIE-dev/rce commit 1762fb4bb65a) into standalone statically typed **Godot 4 GDScript** (extends SceneTree) with 64-bit int coercion, Q32.32 limb-split helpers, and transactional fail/kept state rollback. Across **36 deterministic gameplay tasks** — 2 tasks in every one of GameCraft-Bench's 15 families (Platformer, Strategy, Tycoon, Open-world, Roguelike, Visual novel, Puzzle, Shooter, Simulation, Card game, Horror including horror-signal-lost's radio-battery-jamming mechanic, Rhythm, Idle, Racing, Sports) plus 6 reference and multiplayer systems (Haunted Jaunt, Survival Shooter, Tanks Q32.32, Inventory/Crafting, Quest DAG, 4-Player Lockstep Arena) — both the Form-emitted GDScript and an independently authored idiomatic native GDScript implementation were executed under Godot Engine v4.7.2.stable.official.ed1daf0bf --headless for 500 deterministic input events per task (18,000 ticks per arm = 36,000 Godot VM ticks):
  - **Form admission (rce check + rce test)**: 36 / 36 (100.0%).
  - **Godot 4.7.2 --headless execution pass rate**: 36 / 36 (100.0%) Form-emitted vs 36 / 36 (100.0%) native GDScript.

– **Per-tick and rolling SHA-256 state-trace parity**: 36 / 36 tasks (0 / 18,000 per-tick SHA-256 mismatches).

6.10 Playable joule-native Q32.32 Chrono-Budget Arena (/living/game-engines/)

Joining this study's exact Q32.32 replay contract with the energy-native temporal mechanics of Time as an Index (Section 9.2), Instrument 4 of /living/game-engines/ ships a playable 60 Hz deterministic orbital-navigation and pickup-defense mini-game running in JavaScript BigInt (0 floating-point operations inside simulation):

- **Live SHA-256 state tape & deterministic replay**: Every tick updates a rolling SHA-256 state digest and records compact 1-byte player input events into D; clicking Replay & Verify SHA-256 re-executes the entire match from (seed, D) and verifies bitwise hash equality (replayed_sha256 === live_sha256).
- **Joule-budgeted temporal lookahead & rewind (est_j design model)**: Querying future trajectories (Hellman Branch Preview, simulating 3 candidate steering branches K ticks ahead) or rewinding to a persistent causal checkpoint debits an explicit microjoule (µJ) battery (step_cost_uj per speculative tick plus Landauer branch-erasure cost when non-chosen futures or rewound timelines are discarded), playable via keyboard, mouse/touch, and WebXR / Gamepad controllers (including Meta Quest Browser).

7. Infeasibility flags

1. **Cross-GPU and cross-API bit-exact hardware rendering: infeasible (27 / 28 hardware GPU × API pairs — including 2 / 2 same-GPU DirectX 12 vs Vulkan pairs — diverge in raw SHA-256 in GE-H3), whereas bounded perceptual tolerance (PSNR >= 64.16 dB, SSIM >= 0.99983, $\Delta_{\max} \leq 2$) holds across DirectX 12, Metal, and Vulkan when GPU outputs never feed back into S.**
2. **Cross-platform exact physics with stock floating-point settings: infeasible across Windows, macOS, and Linux.** Even *without* -ffast-math or /fp:fast, standard std::sin/std::cos forms a 3-way divergent triangle across Windows UCRT, Apple libSystem, and GNU glibc (t_div = 10 to 37), default MSVC / fp:precise vs Clang -ffp-contract=fast diverges on x86_64 at t_div = 192, cross-ISA FMA (x86_64 vs arm64) diverges at t_div = 406 (GE-H2), and 840 chaotic trajectories confirm $t^* \propto \ln(\Delta/\delta_0)$ with $\hat{\Lambda} = 1.95$–$5.07\ \mathrm{s}^{-1}$ (GE-H7).
3. **GPU compute inside the simulation: infeasible as exact; feasible only with deterministic reductions** at a measured cost of about 0.2% to 7.8% on the cited hardware, and still subject to compiler contraction unless guarded per instruction (SPIR-V NoContraction) on every target.
4. **Multi-engine deterministic behavior emission: feasible when lowered from a checked Form (GE-H4, GE-H8, GE-H9), infeasible from untyped string-keyed scripts (GE-H6).** Across 17 game behaviors (0 / 680 mismatches across .rce, Windows 11 C#, Godot 4 GDScript, and Rust ECS) and 36 headless Godot 4.7.2 tasks (0 / 18,000 tick mismatches), Form projection preserves bitwise behavior parity, whereas parse-only dynamic checking misses 93.85% (122 / 130) of injected cross-layer and semantic faults (F1 = 0.1159).
5. **Interpreted rules at large actor counts: constrained in boxed interpreters (N_20ms = 390 in rce emit python), resolved by JIT or compiled projections (GE-H5).** Unboxed Wasm32, C++20 -O3, and Rust -O3 projections step 10,000 Q32.32 actors in 0.058–0.125 ms/tick (3,564×–7,728× lower Linux RAPL reported_j/tick than rce emit python) with zero SHA-256 drift.
6. **Proprietary third-party game engines' internal schedulers as unguarded exact hosts: unverified.** While bare-metal Windows 11 PE (rustc, MSVC, and csc.exe), DirectX 12 (wgpu), and headless Godot 4.7.2 SceneTree execution are verified in Section 6, Unity's manual, as quoted by GameRule, disclaims MonoBehaviour script ordering guarantees unless driven from a single deterministic scheduler loop; the Unity and Unreal template catalogs could not be read (their pages render client-side), and no Unreal execution-order source was read.

8. Sources and limits

Every source above was read in full at the passage cited, except: the GameCraft-Bench paper (abstract, repository README, task directory layout, and tests/test.sh verifier script read), the RCE about page (summary page read alongside openIE-dev/rce commit 1762fb4bb65a specification and checker sources), and the Capcom press story (one article). Companion house studies cited for telemetry levels (SATB) and joule-budgeted temporal game

mechanics are Time as an Index and the Satiation Tiered Benchmark Harness (`[same house]`). Sources consulted but not cited as evidence: the Ludii portal, which describes games as structured sets of ludemes; the Game Grammar hobby repository; and the PlayCanvas, Babylon.js and three.js READMEs, which describe rendering libraries rather than game description formats.

Not read, paywalled or otherwise unavailable: T. Schaul, "A Video Game Description Language for Model-based or Interactive Learning," IEEE CIG 2013, doi:10.1109/CIG.2013.6633610; J. Gregory, *Game Engine Architecture*, doi:10.1201/9781315267845. M. Ebner et al., "Towards a Video Game Description Language," doi:10.4230/DFU.Vol6.12191.85, is open access but was not read for this study.

Across all nine empirical suites (`GE-H1` through `GE-H9`), `measured_j = None` and `board_metered = false` on all six hosts (strictly separating macOS `L3` per-process `ri_energy_nj`, Linux `L2` socket/core `RAPL` `reported_j`, and Windows 11 `L3` `GetProcessTimes`/`QueryProcessCycleTime` activity counters, and keeping the Chrono-Budget Arena's `est_j` mechanics strictly analytical). Two deliberate static limits remain documented: in `GE-H6`, bare untagged `i64` scalars without unit or coordinate-space newtype structs accounted for `2 / 130` false negatives; and in `GE-H9`, official `GameCraft-Bench` VLM-judged visual scoring over `140` natural-language prompts was not run because it evaluates open-ended LLM asset/project synthesis and external VLM rubric grading rather than deterministic Form-to-engine behavior compilation.