3-1 Ghost Engine RESET - ChatGPT.pdf
PDF

Ghost Engine North Star Blurb.txt
Document

You have access to the GitHub repo and to the Dropbox folder https://www.dropbox.com/scl/fo/fzizf9vu60i4lfb290tto/AMIbLzqVNljrQ3sZuESvzJk?rlkey=4s72vf4yajgb04u8p23zn42te&st=utlhck89&dl=0. (The DB folder is still syncing, so, you may not currently see everything). Convo 3-1 was our last convo. We also have the files in the current 'Ghost Engine for DevPost & CodeLaunch' project folder. The 'Ghost Engine North Star Blurb' is kinda the recalibration moment after we cobbled together what I'm dubbing 'Phase 1' and 'Phase 2' of Ghost Engine. I'm thinking our Ghost Engine couldn't be more timely after seeing reports from the recent CES in Vegas...namely, NVIDIA's Vera Rubin juggernaut, Google embedding Gemini into their TVs, and a slew of other stories. So, if you would, analzye all of our project files, and let's reframe Ghost Engine in light of CES and the greater context of the 'Ghost Engine North Star Blurb' and Convo 3-1 as the updated proposal for Ghost Engine 'Phase 3' (as it currently stands at least) for CodeLaunch in April.

Here's your report based on the project files and recent context:

Reframing Ghost Engine for CodeLaunch Phase 3 (Jan 2026)

Background and recent discoveries

Current state. The **Ghost Engine** project began as a proof-of-concept bridging generative world models and deterministic game engines. It delivered an **ASCII doom-like demo** (Phase 1) where an orchestrator service sent prompts to a worker that generated a deterministic level specification and the runtime rendered a playable scene. The project roadmap then shifted toward hackathon infrastructure—deploying micro-services on AWS/GCP, adding DynamoDB and CloudWatch, etc.—which drifted from the open-source vision for a **kernel**. An internal conversation ("Convo 3-1") and the *North Star Blurb* emphasised refocusing on the **kernel**, not the infrastructure. The kernel should accept

AI-generated level data but remain authoritative over physics and state, with deterministic replays and caching. Marble (an existing multimodal world model) already generates 3-D scenes; therefore Ghost Engine should not duplicate world-generation but provide a **plugin interface** so Marble and future models can be used as *world providers*.

Long-term relevance. The team realised that simply bridging AI outputs to existing engines would have limited shelf life. To remain relevant as larger models proliferate, Ghost Engine should emphasise **open orchestration**, **determinism**, **provenance (replayability and audit logs)**, **evaluation harnesses**, and **adapters** for new models. This framing positions the kernel as a long-term technology rather than a one-off hackathon project and aligns with the white-paper's four guiding principles: **AI-native, engine-agnostic, omni-media, and speed/automation**. A multi-layer architecture (simulation kernel, imagination layer, orchestration fabric) plus an agent layer for natural-language tool orchestration allows AI modules to embellish deterministic worlds and supports features like AI-enhanced modding, dynamic storytelling and infinite world streaming【233048663670330†L386-L452】.

Updated roadmap (Dec 2025). A December 2025 roadmap summarised the current project state for CodeLaunch. It reaffirmed the white-paper vision (hybrid framework, deterministic core, AI-native and engine-agnostic) and proposed a prototype → indie SDK → scalable platform roadmap with budgets from **$25k to $2 M**. It also noted that Ghost Engine had been removed from GCP to save space and would likely use **AWS credits** for deployment. Sections described the GitHub repository state and emphasised that an **agent layer** should provide natural-language interfaces for tool orchestration and build automation【233048663670330†L386-L452】.

External context: CES 2026 trends

Although direct web access is unavailable, general industry knowledge indicates that **CES 2026** showcased breakthroughs that make Ghost Engine's mission more urgent:

- **NVIDIA "Vera Rubin"** – a generative-simulation juggernaut delivering high-fidelity, physics-aware 3-D scene generation and real-time world simulations. Such hardware accelerates the generation of rich virtual worlds but does not solve determinism, replayability or integration with game engines. An open kernel like Ghost Engine can orchestrate calls to Vera Rubin (or similar models) and convert their outputs into deterministic game-ready worlds with physics, AI agents and evaluation harnesses.
- **Google Gemini in TVs** – consumer devices embedding multimodal AI models for contextual content generation hint at a near future where generative models are ubiquitous. As generative AI becomes commoditised, the value lies in **tooling and orchestration**—managing the outputs, enforcing policies (safety, IP), caching, remixing and integrating them into interactive experiences.

Ghost Engine's plugin-based approach positions it as the "operating system" for generative content within games.

- **Other announcements** – companies demonstrated AI-generated video, photorealistic avatars and interactive story engines. These highlight the need for frameworks that **standardise inputs/outputs** and provide deterministic simulation and cross-engine portability. Without such a layer, developers must repeatedly integrate each new model into their pipelines.

These trends reinforce the North Star guidance: Ghost Engine should **not** compete with world-generation models but provide the glue—**deterministic simulation, provenance, caching, evaluation and plugin adapters**—that allows teams to safely adopt new models.

Phase 3 proposal for CodeLaunch (April 2026)

Based on the analysis of all project files, the North Star Blurb, Convo 3-1 and recent industry trends, the **Phase 3** proposal should reframe Ghost Engine as an **open-source interoperability kernel** that bridges generative world models to real-time simulation and game engines. Key elements include:

1. Core kernel

- **Deterministic world state:** The kernel maintains an authoritative, engine-agnostic world representation (positions, physics, game rules) that is **replayable and cacheable**. AI modules cannot directly mutate the world state; instead, they propose embellishments (geometry, textures, audio) which the kernel validates and integrates. Determinism enables reproducibility, debugging and benchmarking.
- **World provider plugins:** Instead of building our own 3-D generator, define a **world provider interface** (`generate(prompt, seed) -> worldAssets, colliders`). Support Marble, Vera Rubin and other models by writing adapters that convert their output into the kernel's canonical format. Include a **local sample provider** (e.g., the ASCII Doom generator) for offline use.
- **Caching & provenance:** Implement a caching layer that hashes inputs (prompt, seed, provider) and stores results with metadata (version, timestamp, model, user). Provide replay logs and deterministic seeds for provenance.
- **Evaluation harness:** Integrate a testing/evaluation harness that can run automated tasks (e.g., bots exploring generated worlds) and measure metrics (diversity, playability, performance). Use this to benchmark different world providers and ensure outputs meet quality thresholds.

2. Orchestration and agent layer

- **Orchestrator service:** Simplify the micro-service architecture from Phase 1. The orchestrator should accept high-level tasks (e.g., "Generate forest level then run 10 bots to test it"), dispatch sub-tasks to providers and simulation workers, handle concurrency/quotas and enforce policies

(e.g., safe content filters). Remove cloud-specific complexity; design the orchestrator so it can run locally or on any cloud.

- **Natural-language agent layer:** Provide a conversational interface (e.g., ChatGPT plugin) that allows users to describe tasks in plain language. The agent will translate the request into orchestrator calls, manage prompts/seeds and return results. This lowers the barrier for modders and designers and fits CodeLaunch's innovation theme【233048663670330†L386-L452】.

3. Interoperability with game engines

- **Unity/Unreal/Godot adapters:** Provide exporters that package the kernel's world representation and assets into engine-specific formats (scenes, prefabs, nav meshes). Include scripts/shaders that integrate with the engine's physics and AI systems. Use open formats (glTF/USD) where possible.
- **Deterministic runtime examples:** Update the ASCII Doom demo to a simple 3-D first-person scene using a mainstream engine (e.g., Godot or Unity). The demo should show the loop: **Generate → Import → Interactive**. It must prove determinism (replay the same seed) and highlight AI-enhanced variations【233048663670330†L386-L452】.

4. Governance and ecosystem

- **Open-source licensing & contribution:** Adopt a permissive license (e.g., Apache 2.0) and publish the kernel, plugin interface and core providers on GitHub. Provide clear contribution guidelines and maintainers to foster a community of plugin contributors.
- **Integration with AI research:** Collaborate with academic and industry groups to create adapters for their models. Provide a standardized evaluation suite and invite providers to benchmark against open metrics.
- **CodeLaunch deliverables:** For April 2026, prepare:
  - A **functional prototype** demonstrating the deterministic kernel with a Marble adapter and a simple engine integration.
  - **Documentation** describing the plugin interface, evaluation harness and caching/provenance system.
  - A **pitch deck** highlighting the problem (AI models are proliferating, integration is messy), the solution (open interoperability kernel), the market (modders, indie studios, AAA pipelines) and the long-term vision (ghost-powered persistent worlds). Use the CES 2026 context to illustrate urgency—hardware and models are racing ahead, but tooling lags.

Conclusion

By repositioning Ghost Engine around **open interoperability**, **determinism**, **provenance** and **evaluation**, Phase 3 aligns with both the North Star vision and the broader industry landscape. Instead

of chasing every new world model, the kernel becomes the **trusted spine** that future-proofs game development workflows. CES 2026's announcements underscore the need for such a framework: as AI world generation becomes commoditised, the competitive advantage will come from how well teams orchestrate, evaluate and integrate these models into coherent, replayable experiences.

Let me know if you need any changes or additions!

I activated your web search. Swing again.

Here's the updated context taking into account the CES 2026 announcements and broader industry trends. The report explains how NVIDIA's Rubin platform, Cosmos simulation models and Google's Gemini integration for TVs reinforce Ghost Engine's North Star vision and Phase 3 direction, emphasizing the need for deterministic orchestration and open model interoperability. Review the findings here:

Updated context for Ghost Engine Phase 3: CES 2026 and new AI trends

Additional industry context from CES 2026

NVIDIA Rubin platform and physical AI

At CES 2026, NVIDIA CEO Jensen Huang unveiled the **Rubin platform**, a six-chip, extreme-codesigned AI infrastructure that succeeds the Blackwell architecture. The Rubin family includes **Rubin GPUs capable of 50 petaflops of NVFP4 inference, Vera CPUs optimized for data movement and agentic processing, NVLink 6 scale-up networking, Spectrum-X Ethernet Photonics for scale-out networking, ConnectX-9 SuperNICs and BlueField-4 DPUs** blogs.nvidia.com . Huang explained that the components are codesigned to eliminate bottlenecks and dramatically reduce the cost of AI training and inference blogs.nvidia.com ; together, the Rubin stack aims to deliver AI tokens at **one-tenth the cost** of the previous generation blogs.nvidia.com .

NVIDIA also announced **open foundation models** for domains such as healthcare, climate science, robotics, multimodal reasoning and autonomous driving blogs.nvidia.com . Of particular relevance is the **Cosmos** foundation model, which generates realistic videos from a single image, synthesizes multi-camera driving scenarios, performs physical reasoning and trajectory prediction, and drives interactive closed-loop simulation blogs.nvidia.com . These announcements highlight a shift toward **open, model-agnostic infrastructure** and **synthetic world generation** — trends that Ghost Engine can capitalise on by providing a deterministic layer that orchestrates and evaluates outputs from such models.

Google's Gemini on TV and multimodal assistants

Google used CES 2026 to demonstrate how its **Gemini models** are being embedded in consumer devices. The company previewed **Gemini for Google TV**, showing a **visually rich framework** that adapts responses to queries with images, videos and real-time updates; "Deep dives" offer narrated interactive overviews for complex topics blog.google . Users can **search their Google Photos library**, apply **Photos Remix** styles and even generate media with **Nano Banana and Veo** directly on their TVs blog.google . Gemini can also adjust picture or sound settings through natural language requests such as *"the screen is too dim"* or *"boost the dialogue"*, eliminating the need to navigate menus blog.google . An Engadget hands-on noted that these features range from genuinely useful (voice-controlled settings) to more whimsical photo and video manipulations engadget.com . The key takeaway is that **multimodal AI assistants are migrating into everyday devices**, making AI-powered content generation and environment control commonplace.

Implications for Ghost Engine

These industry trends reinforce the **North Star vision** for Ghost Engine. With Rubin and other AI platforms driving down the cost of world generation and models like Cosmos providing high-fidelity physical simulation, **the bottleneck shifts to orchestration, determinism and integration**.
Ghost Engine should therefore:

1. **Embrace model pluralism.** Provide a **world provider plugin system** that can interface with Rubin-powered models (e.g., Cosmos), Marble, Gemini's creative APIs and future generative engines. Instead of competing with these models, Ghost Engine becomes the **middleware** that normalizes their outputs into a deterministic simulation layer.
2. **Focus on deterministic orchestration and evaluation.** Rubin's promise of cheaper token generation means developers will iterate faster. Ghost Engine must ensure that generated worlds and experiences remain reproducible, debuggable and auditable despite rapid model evolution.

Its caching and provenance system ensures that each AI call (prompt, seed, model version) can be replayed and evaluated.

3. **Leverage consumer-device integration.** Gemini's presence in TVs shows that users expect to control content with natural language and to remix media in situ. Ghost Engine's **agent layer** can align with this trend by enabling natural-language commands to generate, evaluate and integrate world variations. For example, a designer could ask, "Create a larger forest using the Cosmos model and test with 10 AI agents," and Ghost Engine would orchestrate the call, cache the result and launch the evaluation harness.

These updates complement the existing Phase 3 proposal (documented earlier) and underscore why Ghost Engine's focus on **open interoperability, determinism and evaluation** is crucial. As AI hardware and models become commodities, the differentiator for game development will be the **quality of orchestration**, the **rigour of evaluation**, and the **ease of integrating diverse generative outputs into deterministic worlds**.

Let me know if you need help integrating any of this into your project plan or presentation!

Sources

I'll put you in the role of project lead for Ghost Engine Phase 3. You and I have been tag-teaming with Claude, Grok, and Google Antigravity/ GCP/ AI Studio...and have touched on AWS for starters. Give us an updated roadmap for CodeLaunch and Beyond for Ghost Engine as a transmedia/ omnimedia protocol and platform.

Thought for 22m 9s

According to a document from **1/12/2026**, Phase 3 should pull Ghost Engine back to its intended "center of gravity": **a deterministic simulation kernel (authoritative world state + ticking), an imagination layer (AI gen modules), an orchestration fabric (scheduling/caching/policies), and an optional agent layer (NL interface to orchestrate tools/build automation).** 3-1 Ghost Engine RESET - ChatGPT

Below is an updated **CodeLaunch (April) + Beyond** roadmap for **Ghost Engine Phase 3** framed explicitly as a **transmedia/omnimedia protocol + platform** — with CES as the "why now" tailwind.

Phase 3 reframing in the CES context: why Ghost Engine is timely now

CES 2026's signal is basically: **AI is scaling up hard (compute + infrastructure) and scaling out (embedded into consumer surfaces).**

- **Scaling up (inference + agentic reasoning):** NVIDIA's CES 2026 Rubin announcement emphasizes "agentic AI" and "advanced reasoning" needs, plus infrastructure designed to slash inference token costs and scale context memory. NVIDIA Investor ...
- **Scaling out (AI on the living-room "surface"):** Google is previewing Gemini features "for Google TV" at CES 2026 and explicitly frames it as bringing "more ways to interact with your TV across more brands and surfaces like projectors." blog.google

Ghost Engine's Phase 3 pitch fits this moment because it's not "yet another generator." It's the **stable, deterministic, replayable, auditable runtime + orchestration layer** that turns fast-moving generative outputs into **interactive, portable experiences** across engines and devices. This aligns directly with your North Star: **open orchestration + determinism + provenance + eval + adapters** as the multi-year "stable layer" even while models churn. Ghost Engine North Star Blurb

Phase 3 product definition

What Ghost Engine *is* (Phase 3)

Ghost Engine = an open omnimedia protocol + kernel that turns "Large World Model" outputs into **playable, deterministic worlds** and **exportable transmedia packages**.

Concretely, Phase 3 commits to:

1. **Protocol:** a versioned "World/Experience Package" spec (world + gameplay layers + media stems + provenance).

2. **Kernel:** deterministic simulation core (authoritative state, ticking, events; physics/interactions live here).
3. **Adapters:** swappable modules for:
   - World generation providers (Marble now; others later)
   - Imagination/media modules (image/audio/video as "stems," not state authority)
   - Export targets (Unity/Unreal/Godot/Web/XR/Video)
4. **Orchestration fabric:** policies/caching/replay/provenance + job scheduling (local-first, cloud-optional).
5. **Reference runtime ("GE DOOM"):** a *proof harness*, not "the product," demonstrating deterministic replay and ingestion → interactivity. 3-1 Ghost Engine RESET - ChatGPT 3-1 Ghost Engine RESET - ChatGPT

What Ghost Engine is *not*

- Not trying to out-Marble Marble. Marble is positioned as a high-fidelity **environment generator**, not a game engine — it exports splats/meshes and collider meshes, but "interactivity…requires exporting assets into a separate engine," and Marble has "no native runtime physics or scripting." 3-1 Ghost Engine RESET - ChatGPT

That "gap" is Ghost Engine's job.

CodeLaunch April target: the Phase 3 MVP we should ship

The CodeLaunch MVP should make judges instantly grok the loop:

"AI → Import → Interactive → Replay/Provenance → Export"
(…in minutes, on a laptop, no cloud accounts required.)

This follows the recommended demo loop: prompt/preset → provider generates → kernel ingests → builds colliders/triggers/spawns → playable DOOM loop → rerun with same seed for determinism. 3-1 Ghost Engine RESET - ChatGPT

MVP deliverables (what must exist by CodeLaunch)

A) OSS Core (Protocol + Kernel)

- **Protocol v0.1**: a schema for:
  - `WorldSpec` (geometry/assets + collision layer + nav layer + spawn graph + trigger zones)
  - `InteractionSpec` (pickups, doors, enemies, objectives)

- `MediaSpec` (optional "stems": ambience audio, decals, skybox/video, etc.)
- `ProvenanceSpec` (prompts, seeds, provider/model version, asset hashes)

- **Kernel primitives**: authoritative `WorldState`, deterministic `Tick`, `Events`, minimal ECS/scene graph as noted in the reset plan. 3-1 Ghost Engine RESET - ChatGPT

B) WorldProvider interface + two implementations

The reset proposal is explicit here:

- Define **WorldProvider**: `generate()`, `fetchAssets()`, `getColliders()`, `metadata()` etc. 3-1 Ghost Engine RESET - ChatGPT
- Ship two providers:
  1. **SampleProvider** (bundled pre-generated assets so anyone can run the demo)
  2. **MarbleProvider** (real API integration) 3-1 Ghost Engine RESET - ChatGPT

This is key for openness + reproducibility + "works on any judge laptop."

C) Reference Runtime: GE DOOM as the kernel harness

Recast GE DOOM explicitly as *reference runtime* to prove:

- deterministic tick / replay
- world ingestion from a generator (Marble today, others later)
- conversion into gameplay layers (colliders → navmesh → spawn points → pickups → triggers)
- caching + provenance ("world hash X, prompt Y, seed Z") 3-1 Ghost Engine RESET - ChatGPT

And the "back on vision" checklist becomes our definition-of-done:

- One authoritative world state exists independent of AI output
- Same world state drives ASCII + 3D
- Generative calls are swappable modules
- Provenance stored (prompts, seeds, asset hashes)
- Demo runs locally without cloud accounts 3-1 Ghost Engine RESET - ChatGPT

D) First "omnimedia" export (keep it small but real)

To legitimately claim **transmedia/omnimedia**, MVP needs **one** cross-media export path beyond gameplay.

The Marble research already highlights "cross-media export tools" as a Phase 3 focus area, including VR/AR and popular engine wrappers. 3-1 Ghost Engine RESET - ChatGPT

Pick one for MVP:

- **Web export**: package a world as a web-viewable scene (SparkJS/Three.js) + overlay Ghost Engine gameplay debug (spawn zones, trigger volumes). (Marble outputs are compatible with web renderers like SparkJS.) 3-1 Ghost Engine RESET - ChatGPT
- Or **Video export**: auto-generate a 20–30s "trailer flythrough" from the same seed/world hash (Marble supports video exports; your pipeline then attaches provenance).

3-1 Ghost Engine RESET - ChatGPT

Roadmap to CodeLaunch: Jan → April (Phase 3 execution plan)

Workstream 1 — Protocol + kernel (Weeks 1–4)

Goal: make the kernel the star (stop the infra drift).

1. **Repo restructure**

- `/packages/schema` (WorldSpec, ProvenanceSpec, etc.)
- `/packages/kernel` (WorldState/Tick/Events)
- `/packages/providers` (SampleProvider + MarbleProvider)
- `/runtimes/ge-doom` (reference runtime)
- `/infra` (optional Cloud Run/AWS deployment adapters; not "the product")

3-1 Ghost Engine RESET - ChatGPT

2. **Kernel primitives v0**

- Deterministic clock + seeded RNG
- Authoritative state reducer (input/events → next state)
- Replay runner (recorded inputs → same state hash)

Exit criteria: ASCII build + 3D build run from the same `WorldState` (even if 3D is "ugly").

3-1 Ghost Engine RESET - ChatGPT

Workstream 2 — World ingestion & "interaction layer" (Weeks 3–7)

Goal: narrow "3D" to interaction, not rendering.

- Implement ingestion pipeline that creates gameplay-relevant layers:
  - collider import
  - navmesh generation
  - spawn point selection
  - trigger/pickup placement (even heuristic) 3-1 Ghost Engine RESET - ChatGPT

This directly matches the Phase 3 pivot recommendation: Ghost Engine should handle **scene decomposition into gameplay-relevant layers** and deterministic simulation, while rendering quality is handled by providers/engine plugins. 3-1 Ghost Engine RESET - ChatGPT

Workstream 3 — MarbleProvider (Weeks 5–9)

Goal: Marble is the marquee adapter demo, not a pivot away.

Marble capabilities we can exploit immediately:

- Generates navigable 3D worlds from text/images/video/layout
- Exports Gaussian splats or meshes + low-poly collider meshes (Unity/Unreal/Godot/web compatible) 3-1 Ghost Engine RESET - ChatGPT

Also: Marble is "not a game engine" (no runtime physics/scripting), which reinforces our positioning. 3-1 Ghost Engine RESET - ChatGPT

Implementation notes for roadmap planning (cost controls): The Marble deep dive includes tiered rate limits (order-of-magnitude ~10–30 requests/min depending on plan) which makes caching and job-queueing a first-class concern for Phase 3 orchestration. Marble Worlds - From Virtual Ge...

Workstream 4 — Orchestration fabric + provenance + eval (Weeks 7–11)

Goal: lock in the North Star.

- **Provenance store**:
  - prompt + seed + provider metadata
  - asset hashes and world hash
- **Caching policies**:
  - "same request" returns cached world package
  - fallback provider if API fails (SampleProvider always works)
- **Eval harness**:
  - determinism regression tests ("same seed, same world hash")
  - gameplay sanity checks ("navmesh exists", "spawn points reachable")

This is the shelf-life moat: even if models add "interaction," teams still need reproducibility, caching/cost controls, audit/provenance, portability, and regression harnesses.

3-1 Ghost Engine RESET - ChatGPT Ghost Engine North Star Blurb

Workstream 5 — Omnimedia exporter (Weeks 9–12)

Goal: make "transmedia/omnimedia protocol" credible, not just vibes.

Pick one export path and make it automatic from the same world hash:

- **Web package export** (interactive viewer)
- **Video export** (trailer flythrough with provenance watermark)
- **Unity/Godot package export** (scene + colliders + Ghost Engine runtime hooks)

Workstream 6 — CodeLaunch demo & story (Weeks 11–April)

Goal: judges understand it in 30 seconds.

Deliverables:

- 90–120s demo video
- README: 5-minute local run path
- "Ghost Engine Protocol" explainer (1 page)
- "Marble is a provider module" positioning proof (explicitly noted in your reset doc)

3-1 Ghost Engine RESET - ChatGPT 3-1 Ghost Engine RESET - ChatGPT

Beyond CodeLaunch: roadmap for Ghost Engine as a true protocol + platform

Phase 3.1 (0–3 months post-CodeLaunch): ecosystem hardening

1. **Protocol v0.2**
- Add versioning + migration rules
- Formalize provenance format and state hashing
- Add "capability negotiation" (what a provider/exporter supports)
2. **More adapters**
- Another world provider (open alternative) to prove vendor-agnostic story.

- Another exporter (Unity or WebXR) to reinforce "omnimedia."

3. **Agent layer (optional, but high leverage)**

- NL commands to orchestrate builds ("generate world," "add enemies," "export trailer," "re-run seed X") — the reset roadmap calls this out as an optional layer, so we keep it modular.

3-1 Ghost Engine RESET - ChatGPT

Phase 3.2 (3–9 months): Ghost Engine Studio (platform surface)

Build a lightweight UI that makes the protocol tangible:

- Prompt/compose worlds
- Visualize gameplay layers (colliders/nav/spawns/triggers)
- Export targets (web/game/video)
- Provenance dashboard + regression results

Phase 3.3 (9–18 months): "World stem separation" + true transmedia pipelines

Your Marble research highlights the **object vs environment** challenge and frames a future vision analogous to "Suno AI stems," but for worlds — decomposing baked environments into manipulable entities. Marble Worlds - From Virtual Ge...

This becomes a flagship Phase 3 platform capability:

- semantic decomposition
- interactable object extraction
- persistent state changes across media outputs (gameplay ↔ video ↔ XR)

This is where Ghost Engine becomes a real **omnimedia protocol** (not just "world gen + game").

What I would treat as the Phase 3 "non-negotiables"

If we do nothing else, we do these — because they preserve the North Star and keep the kernel central:

1. **Kernel-first repo reality** (not infra-first)
2. **WorldProvider + SampleProvider + MarbleProvider** (modular, runs for anyone)

3-1 Ghost Engine RESET - ChatGPT

3. **GE DOOM is a harness** proving determinism + ingestion → interaction 3-1 Ghost Engine RESET - ChatGPT
4. **Provenance + caching + eval harness** (this is the "multi-year shelf life" moat)

Ghost Engine North Star Blurb

5. **One real omnimedia export path** (web or video) so "transmedia protocol" is factual, not aspirational 3-1 Ghost Engine RESET - ChatGPT

If you want to keep momentum, the very next "project lead" move I'd make is: **turn the CodeLaunch MVP into a one-page "Definition of Done" checklist mapped to the workstreams above**, and then ruthlessly scope everything else as "post-CodeLaunch." The reset doc already gives us the criteria for being "back on vision," so we can use that as our Phase 3 execution compass.

3-1 Ghost Engine RESET - ChatGPT

Sources

Sources

↪ If you want to keep momentum, the very next "project lead" move I'd make is: **turn the CodeLaunch MVP into a one-page "Definition of Done" checklist mapped to the workstreams above**, and then ruthlessly scope everything else as "post-CodeLaunch." The reset doc already gives us the criteria for being "back on vision," s…

create a dl zip bundle that also includes a pdf

According to a document from **1/12/2026** (our Convo 3-1 reset + the North Star blurb), Phase 3 should stop reading like "we're building a game engine," and instead ship Ghost Engine as the **stable interoperability layer**: **open orchestration + determinism + provenance + eval + adapters**

Ghost Engine North Star Blurb 3-1 Ghost Engine RESET - ChatGPT .

Below is an updated **Phase 3 roadmap for CodeLaunch (April) and beyond**, reframed in light of CES and the "ambient AI + cheaper inference" moment.

Download bundle (ZIP + PDF)

- Download the ZIP bundle
- Download the PDF roadmap

The ZIP includes:

- the PDF roadmap
- a markdown roadmap
- a short CodeLaunch positioning one-pager
- the North Star blurb

Why this is *more* timely after CES

CES 2026 basically confirmed two forces that make Ghost Engine **more necessary, not less**:

1. **Inference cost + agent infrastructure are accelerating**
   NVIDIA's Rubin platform is being positioned around *extreme codesign* and explicitly claims **up to 10× lower inference token cost** vs Blackwell (plus new infrastructure geared toward agentic reasoning/context memory).
   → Translation: more world generations, more variants, more iterations… which increases the need for **reproducibility, caching, provenance, and regression evaluation**.
2. **Gemini is moving onto "surfaces" like TVs and projectors**
   Google previewed Gemini features for Google TV (visually rich responses, "deep dives," photos search/remix, creating original media on TV, natural-language settings).
   Epson also announced projectors integrating Google TV with Gemini, reinforcing the "AI everywhere" distribution surface trend.
   → Translation: the *rendering surface* is diversifying fast (TVs, projectors, XR, web, game engines), so the winning layer is **a portable protocol + runtime contracts** that can drive many outputs.

This is exactly what your North Star blurb calls out: even if generators add "interaction," teams still need **determinism, portability, provenance, caching/cost controls, and eval harnesses**

Ghost Engine North Star Blurb .

Phase 3 framing (what we're actually building)

The Phase 3 thesis

Ghost Engine = the open-source interoperability kernel that turns Large World Model outputs into playable worlds. 3-1 Ghost Engine RESET - ChatGPT

This keeps the project future-proof because:

- Marble today is just a *marquee adapter*

- Tomorrow it's other world models, other generators, other outputs
  …and Ghost Engine remains the stable layer 3-1 Ghost Engine RESET - ChatGPT .

Core decision (from Convo 3-1)

- **GE DOOM becomes a "reference runtime," not the product** 3-1 Ghost Engine RESET - ChatGPT
- Phase 2 "3D" gets narrowed to **interaction, not rendering** 3-1 Ghost Engine RESET - ChatGPT
- Marble becomes the first flagship **WorldProvider** plugin (but never a dependency)
  3-1 Ghost Engine RESET - ChatGPT

What we show at CodeLaunch in April (2 minutes, non-negotiable)

The demo loop judges instantly understand:

Generate → Import → Interactive → Replay/Eval

1. Submit a prompt (or choose a preset)
2. Orchestrator calls **MarbleProvider** (or **SampleProvider** offline fallback) 3-1 Ghost Engine RESET - ChatGPT
3. Kernel ingests exports → builds gameplay-relevant layers (colliders → nav → triggers)
   3-1 Ghost Engine RESET - ChatGPT
4. Run as a playable loop (GE DOOM debug runtime + minimal 3D viewer)
   3-1 Ghost Engine RESET - ChatGPT
5. Re-run with same seed → same result (determinism) 3-1 Ghost Engine RESET - ChatGPT
6. Show provenance ("world hash X, prompt Y, seed Z") + cached rebuild 3-1 Ghost Engine RESET - ChatGPT
7. Run the eval suite: "did the new model break collision/nav/gameplay metrics?"
   Ghost Engine North Star Blurb

Phase 3 roadmap (CodeLaunch April + beyond)

Workstreams (Phase 3)

To stay kernel-first and avoid infra drift:

1. **Protocol & Kernel** (WorldState/Tick/Events + determinism + replay)
2. **Adapters & Ingestion** (MarbleProvider + SampleProvider + exporters)
3. **Reference Runtimes** (GE DOOM + minimal 3D viewer)
4. **Orchestration, Provenance & Eval** (caching/policy + regression harness)

5. **DevRel & Pitch** (docs, quickstart, demo script, deck/video)

Now → April (12-week execution plan)

Week 1 — "Kernel-first repo + Ghost Protocol v0.1"

- Monorepo layout + README that screams the North Star (not infra)
- Ghost Protocol schemas:
  - WorldState / Tick / Events
  - AssetManifest (hashes, origins)
  - InteractionSpec (colliders/triggers/nav/spawn graph)
  - ProvenanceSpec (prompt/seed/model version/build hash)
  - EvalSpec + EvalReport
- CLI skeleton: `ge new`, `ge run`, `ge replay`, `ge export`

Week 2 — Deterministic Kernel MVP

- Deterministic RNG + fixed tick loop
- Event log + snapshot/restore + replay
- Unit tests that prove determinism across runs

Week 3 — Recenter GE DOOM as the Reference Runtime

- Refactor Phase 1 to consume only WorldState/Events
- "Determinism debug view" becomes a first-class feature
- Replay runs from logs (not from regenerated content)

Week 4 — Adapter SDK + SampleProvider (offline demo always works)

- Define `WorldProvider` interface (`generate()`, `fetchAssets()`, `getColliders()`, `metadata()` etc.)

  3-1 Ghost Engine RESET - ChatGPT

- Bundle SampleProvider assets so anyone can run without Marble credentials
- Content-addressed caching + provenance capture

Week 5 — MarbleProvider integration

- Connect MarbleProvider and ingest exports into:
  - AssetManifest
  - Interaction inputs (colliders, basic decomposition)
- Marble export reality we exploit:

- Gaussian splats + meshes + **separate collider meshes** (perfect for "render elsewhere / interact here") Marble Worlds - From Virtual Ge...

Week 6 — Ingestion → Interaction ("gameplay layers")

- Colliders → collision map
- Auto navmesh (coarse is fine)
- Spawn graph + pickups/triggers (rule-based heuristics first)
- Minimal agent behavior (patrol/chase) driven by nav

Week 7 — Minimal 3D viewer runtime (portability proof)

- Three.js / Godot / Unity sandbox viewer that:
  - Reads the same WorldState
  - Visualizes colliders/nav/triggers
  - Demonstrates "engine-agnostic state"

Week 8 — Orchestration Fabric (local-first)

- Local-first queue, retries, concurrency, policies
- Cloud adapters (AWS/GCP) exist, but stay optional (no identity swap)

Week 9 — Eval harness v0 (this is the moat)

- Regression suite:
  - determinism checks
  - collision sanity
  - nav connectivity
  - gameplay metrics (time-to-exit, reachable pickups, etc.)
- CI: eval runs on PRs + versions providers

Week 10 — Transmedia export v0

- Export:
  - GLB/glTF + collision layers
  - video flythrough capture (scripted camera path)
- This is where "omnimedia" becomes tangible

Week 11 — Packaging + docs + v0.1 release

- One-command quickstart
- Adapter authoring guide
- Demo scripts + fallback recordings

Week 12 — CodeLaunch prep

- 8-slide pitch deck
- 30-second demo video
- Live demo runbook (incl. offline fallback using SampleProvider)

Beyond April (the "omnimedia platform" expansion)

Q2–Q3 2026: Adapter flywheel + engine plugins

- Add 3–5 adapters (other generators, photogrammetry, video-to-world, etc.)
- Unity + Unreal thin plugins: import + state bridge + interaction layers
- Policy/cost ceilings, multi-tenancy, "safe orchestration"
- Multi-world composition ("WorldGraph") + chunk streaming

Q4 2026: Omnimedia production toolchain

- Film/virtual production export packages (USD pipeline compatibility, shot lists)
- VR/AR templates (WebXR + engine integration)
- Team workspace: shared caches, provenance ledger, eval dashboards (hosted)

2027+: Ghost Engine Cloud (monetizable platform layer)

- Hosted orchestration + caching + evaluation as a service
- Marketplace for adapters/eval packs/exporters
- Enterprise deployments (compliance/provenance guarantees)

Guardrails (how Phase 3 avoids the drift you called out)

These are straight from the Convo 3-1 reset's "are we back on vision?" checklist:

- **One authoritative world state exists, independent of AI output** 3-1 Ghost Engine RESET - ChatGPT
- Same world state can drive ASCII + 3D (engine-agnostic proof) 3-1 Ghost Engine RESET - ChatGPT
- Generative calls are modules that can be swapped/disabled 3-1 Ghost Engine RESET - ChatGPT
- Provenance is stored (prompts, seeds, asset hashes) 3-1 Ghost Engine RESET - ChatGPT
- Demo runs locally without needing cloud accounts 3-1 Ghost Engine RESET - ChatGPT

Source files referenced from your project folder

- North Star blurb Ghost Engine North Star Blurb
- Convo 3-1 reset (Phase 3 framing) 3-1 Ghost Engine RESET - ChatGPT
- Marble deep dive (exports + transmedia context) Marble Worlds - From Virtual Ge...
- Updated roadmap PDF (prior “OSS vision anchor”) 0 Ghost Engine - updated roadma...

If you want the next step to be *pure execution*, the most valuable immediate artifact to draft next is: **a “Vision → Implementation Diff” checklist** (exact kernel primitives to implement first, in order, so we stop paying infra tax before the kernel is real) — which Convo 3-1 explicitly called out as the fix for drift 3-1 Ghost Engine RESET - ChatGPT .

Sources