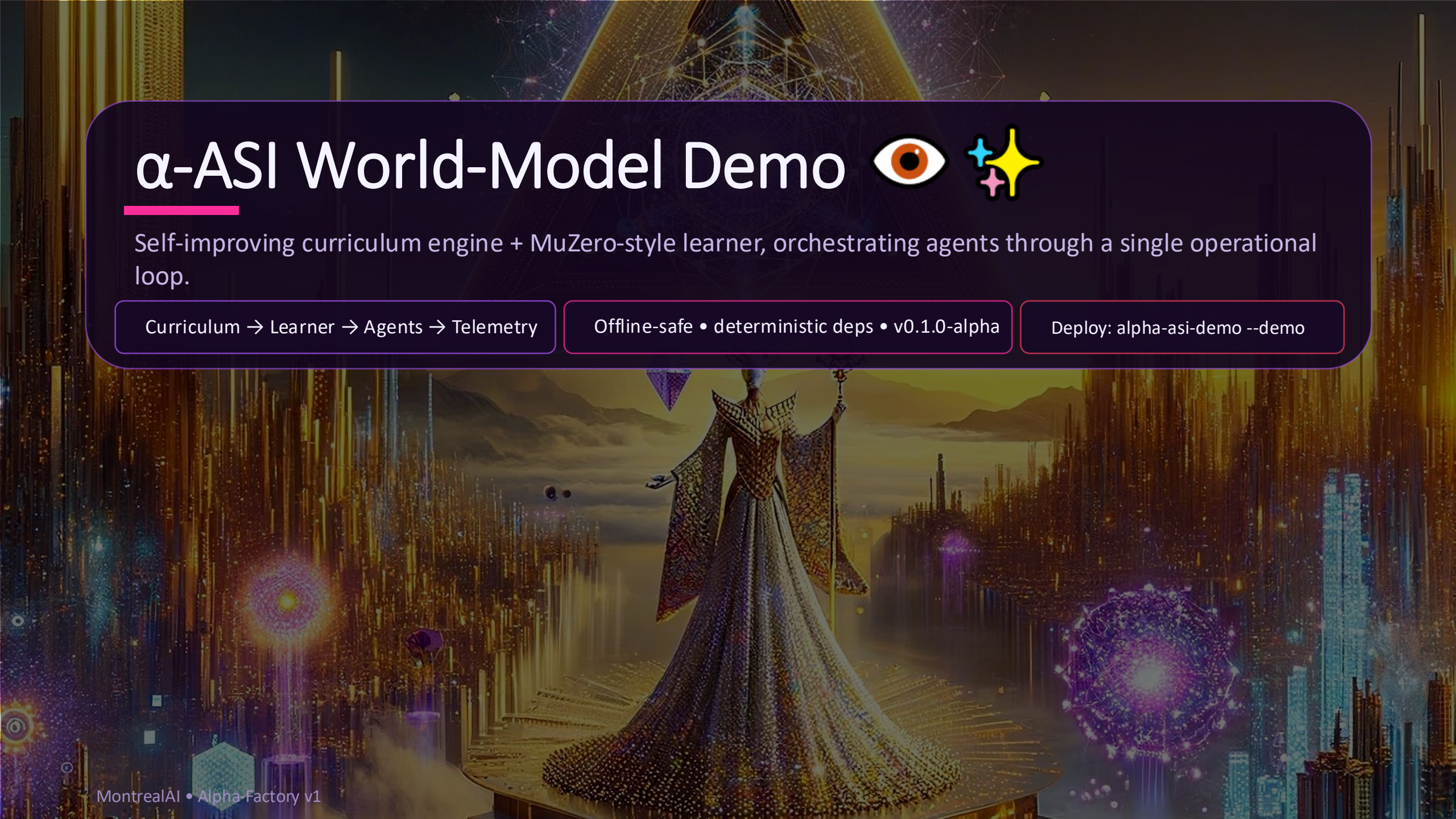

α-ASI World-Model Demo 👁✨

Self-improving curriculum engine + MuZero-style learner, orchestrating agents through a single operational loop.

Curriculum → Learner → Agents → Telemetry

Offline-safe • deterministic deps • v0.1.0-alpha

Deploy: alpha-asi-demo --demo

Executive thesis

A single loop that compounds capability by design.

The demo implements an operational world-model flywheel:

- Agents generate curriculum signals (new environments, rewards, plans).
- A MuZero-style learner updates the internal model and policies.
- A safety guard halts divergence via loss monitoring.
- Telemetry closes the loop via a UI/WS interface for commands + inspection.

Outcome

- Open-ended difficulty scaling (POET generator → mini-world pool).
- Planning under uncertainty (learned dynamics + search).
- Parallel specialization (Strategy / Research / CodeGen agents).
- Operator control via telemetry + command channel.

Alpha-Factory Bus (A2A)

Curriculum, telemetry, and safety converge on a single orchestrated loop.

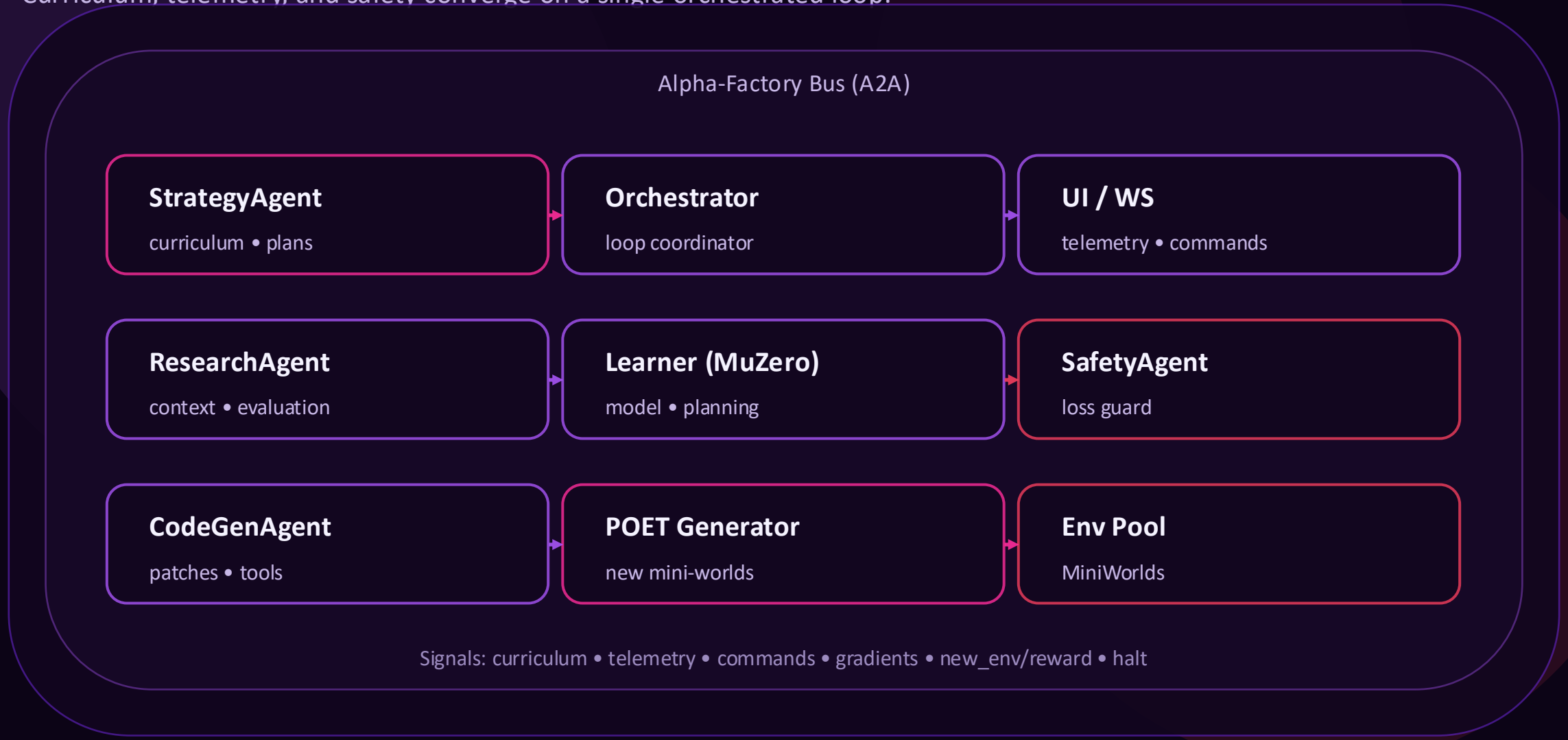

Diagram reproduced from repository README (architecture excerpt).

Open-ended curriculum

Endlessly generating solvable novelty (POET generator → mini-world pool).

Generator

- Mutate environments
- Score novelty + learnability
- Transfer stepping stones

→

MiniWorld Pool

- Parallel env_batch
- Diverse dynamics
- Revisit for stability

→

Signal to agents

- New env / reward
- Difficulty shaping
- Curriculum metrics

Why open-endedness matters

- Avoids fixed-dataset ceilings: the frontier moves with capability.
- Discovers stepping stones that unlock discontinuous jumps.
- Produces a curriculum that remains relevant as the agent changes.

Learner (MuZero-style)

A learned dynamics model + search for action selection.

Representation

- Encode observation → latent state
- Compress dynamics
- Stabilize learning

Dynamics

- Predict next latent given action
- Regularize transitions
- Support rollouts

Prediction

- Output policy + value + reward
- Guide tree search
- Enable planning

Planning loop

- Run MCTS rollouts per action (mcts_simulations).
- Select actions that maximize expected value under the learned model.
- Update parameters from trajectories and value targets.

INTERPRETATION: HAMILTONIAN-LIKE FLOW ON A LEARNED ENERGY LANDSCAPE (SEARCH + LEARNING).

Agents in the loop

Specialization without fragmentation: each agent writes to the same bus.

StrategyAgent

- Curriculum shaping
- Plans for exploration
- Reward design probes

ResearchAgent

- Context + evaluation
- Ablations & comparisons
- Insight extraction

CodeGenAgent

- Code patches
- Tooling scaffolds
- Automation hooks

Coordination principle (game-theoretic lens)

- Shared interface → shared incentives: signals are legible and comparable.
- Parallel search in strategy space, with centralized arbitration.
- Emergent division of labor: local optimization, global coherence.

Safety guardrails

Loss monitoring as an automatic circuit-breaker.

SafetyAgent (loss guard)

- Monitors learner loss statistics.
- Issues “halt” on divergence before cascading failures.
- Keeps the operator in the loop via telemetry.

Design intent

- Fail fast, recover clean.
- No silent drift.
- Every halt is observable and attributable.

Circuit-breaker flow

1) Observe loss + reward stats
2) Detect anomaly / NaN / spike
3) Emit halt to orchestrator
4) Persist trace + notify UI/WS
5) Resume under revised settings

Memory & knowledge fabric

A shared substrate for retrieval, causality, and auditability.

Reference pattern

```
[Event] → embeddings → vector DB
        ↘ edges      → graph (CAUSES, SUPPORTS, RISK_OF)
```

- Agents query memory as a first-class tool (search → evidence → action).
- Planner can traverse causal graphs to justify decisions.
- Fallback to SQLite vector store for lightweight/offline operation.

Operational effects

- Cross-agent continuity: shared context prevents “amnesia.”
- Audit trail: decisions become reconstructible paths.
- Composable reasoning: retrieval + planning + execution.

Design lens

- Markov blanket sets controllability.
- Minimize surprise: tighten model⇄evidence loops.

Telemetry & command plane

A UI/WS interface closes the loop: observe, intervene, resume.

What telemetry enables

- Real-time loss/reward traces and curriculum state.
- Command channel for pausing, halting, and adjusting parameters.
- Operator-grade observability: what changed, when, and why.

Operator posture

- Observe → decide → act → verify.
- Never lose control of the loop.

Signal map (A2A)

- UI/WS → Orchestrator: commands
- SafetyAgent → Orchestrator: halt
- StrategyAgent → Orchestrator: new_env/reward
- Learner: gradients update internal model

Tunable control surface

A few parameters shape scale, depth, and compute allocation.

config.yaml (selected knobs)

```
env_batch:        parallel environments
hidden:           latent state size
mcts_simulations: rollouts per action

Rule of thumb:
  ↑env_batch   → breadth
  ↑hidden      → representational capacity
  ↑mcts_sims   → planning depth
```

Interpretation (statistical physics lens)

- env_batch increases sampling of the state-space (better coverage).
- hidden sets the dimensionality of the internal order parameter.
- mcts_simulations allocates compute to search vs. learning updates.

Control aim

- Keep the loop stable, then scale.
- Spend compute where marginal gain is highest.

5-minute quickstart

From clone to running demo UI, end-to-end.

Terminal

```
git clone --branch v0.1.0-alpha
https://github.com/MontrealAI/AGI-Alpha-Agent-v0.git
cd AGI-Alpha-Agent-v0
./quickstart.sh --preflight
python check_env.py --auto-install
alpha-asi-demo --demo    # launch the world-model UI
```

What happens next

- Orchestrator starts loop → streams telemetry.
- POET populates a mini-world pool.
- Learner trains w/ planning (MuZero-style).
- SafetyAgent halts on instability.

Tip

Start stable → scale parameters
→ verify telemetry → iterate.

Offline-safe operation

Graceful degradation: keep running when the world is hostile.

No GPU

- Fallback to local quantized model path
- Lower throughput, same loop
- Focus on stability

No API key

- Switch to local embeddings + heuristics
- Continue agent coordination
- Preserve determinism

No internet

- Wheelhouse install path
- Pinned deps
- Reproducible runs

Reliability posture

- Prefer local-first paths; treat network services as accelerators, not dependencies.
- Verify checksums and pinned versions for deterministic deployment.
- Keep telemetry enabled: trust comes from observability.

Multi-scale consequences

A single loop can harvest opportunity across scales.

Energy-landscape framing

"Cells with $\Delta\mathcal{F} < 0$ glow; agents race to harvest."

Macro (Δβ)

- Finance signals
- Energy markets
- Policy shifts

Meso (ΔS)

- Supply chains
- Manufacturing
- Logistics

Micro (ΔH)

- Biotech targets
- Materials search
- Chemistry

Metaphor anchored by repository README multi-scale diagram excerpt.

First-principles lens

Free energy, Hamiltonians, autopoiesis: why the path becomes simple.

Variational free energy

- Minimize surprise via model evidence
- Perception-action loop as inference
- Tighten prediction → correction

Hamiltonian flow

- Search follows structured trajectories
- Conserve signal; dissipate noise
- Exploit symmetry when it appears

Autopoiesis

- Agents regenerate tools
- Loop maintains itself
- Capability compounds

Practical translation

- The system keeps reducing uncertainty while expanding its own training distribution.
- Compute is spent where it most reduces expected error (search vs. update).
- Stability guardrails preserve coherence as scale increases.

Roadmap (operational)

Scale, validate, and harden the loop.

1) Stabilize

- Set conservative defaults
- Telemetry baselines
- Reproducible runs

2) Scale

- Increase env_batch
- Tune hidden
- Allocate MCTS

3) Institutionalize

- Safety gates
- Audit & provenance
- Deployment playbooks

PRINCIPLE: NEVER TRADE OBSERVABILITY FOR SPEED.

Deploy the loop.

A world-model that generates its own curriculum, plans under uncertainty, and coordinates specialized agents — with safety as a first-class signal.

```
alpha-asi-demo --demo
```

Repository: github.com/MontrealAI/AGI-Alpha-Agent-v0

Run

- Start stable defaults
- Offline-safe path
- Deterministic deps

Observe

- Loss + reward traces
- Curriculum state
- Operator controls

Iterate

- Tune env_batch / hidden / MCTS
- Validate improvements
- Harden safety gates

References

Primary sources used in this deck.

- MontrealAI/AGI-Alpha-Agent-v0 (README: α-ASI World-Model Demo, bus diagram, quickstart, tuning knobs)
- Schrittwieser et al. “MuZero” (arXiv:1911.08265; Nature 2020)
- Wang et al. “POET” (arXiv:1901.01753)
- Friston / Free Energy Principle (overview for interpretive lens)