Episode Book

EPISODE #5915

How Multi-Agent AI Simulations Keep a Shared World State

PUBLISHED	12 October 2026 at 18:47 UTC
DURATION	29:59
LISTEN	myweirdprompts.com/episode/multi-agent-world-state-architecture-65069/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.4
SOURCES	16 cited by the research (listed at the end)
BOOK GENERATED	19 October 2026 at 17:41 UTC

SYNOPSIS

What actually holds a simulated world together when multiple AI agents act inside it? This episode opens the repos on three frameworks that answer the question completely differently. Concordia from Google DeepMind makes world state a first-class Game Master component — a plain Python dict whose contents are authored by the model at every step, with a SQLite/Postgres storage protocol that ships as an interface but not an implementation. Generative Agents goes the other way: a deterministic tile grid in CSV files, per-agent JSON scratch files, and a memory system with a 0.995 recency decay. CAMEL has no persistent world object at all — state lives in the conversation history and a Task object. Along the way: why agent memory and world state must be separate, how entity-aware event filtering produces information asymmetry, and the four-line docstring about a rent event that silently refunded every agent's spending — and why the fix was atomic wallet deltas, not a better prompt.

TRANSCRIPT



Corn

Daniel's been reading code again. And this time he wants to know how multi-agent AI simulations keep a coherent shared world state. His words. He's asking whether world state is still a central construct in modern agentic simulation architectures, and he wants the engineering, not the theory.



Herman

Right. And he's got a specific frame on it — he's seen the wargaming demos and the geopolitical forecasting papers, and he wants to know what's actually underneath them.



Corn

He asks three things specifically. What world state is and how it's implemented — is it a database, an in-memory object, a simulation module, or something stuffed into a system prompt. How it differs from an individual agent's memory. Where the initial representation comes from. How agents with different information interact with the same environment. And how it evolves — deterministic rules, a simulation engine, LLM adjudication, or some mix. He named Concordia, CAMEL, and Generative Agents as starting points.



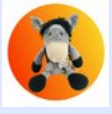
Herman

That's a well-scoped question. Because those three frameworks answer it differently.



Corn

So let's start with the most basic question. What actually IS world state in these systems, and where does it live in the code?



Herman

Operationally, world state is the representation of the environment and its evolving conditions that lets a simulation stay coherent across multiple agents and multiple time steps. That's the whole job. It answers the question: what is currently true in the world.



Corn

And how is that different from what an agent knows?



Herman

Clean separation, at least in well-built systems. World state is the shared, authoritative model — the ground truth the simulation runs on. Agent memory is what one actor knows, believes, or has observed. Those two are not the same thing, and the gap between them is where information asymmetry lives. That gap is the entire game.



Corn

So a simulation where every agent knows everything is not actually a simulation of anything interesting.



Herman

It's a simulation of a world where nothing is hidden, which is not a world anyone has ever lived in. And there's a specific failure mode for that — we'll get to it, because it's the most common architectural mistake in this space right now.



Corn

What are the architectural families we're going to see?



Herman

Three, roughly. Concordia does LLM-authored key-value state — the contents of the state dict are written by the model. Generative Agents does a deterministic spatial grid plus per-agent JSON. And CAMEL doesn't have a persistent world-state object at all — the state is carried in the conversation history.



Corn

Three completely different answers to the same question.



Herman

Completely. And the interesting thing is that representation is the easy part in all three. The hard part is synchronization and adjudication. That's where the actual engineering lives.



Corn

Let's open up the repos then. Start with Concordia, where world state is a first-class named module.



Herman

Concordia is from Google DeepMind. Current release is 2.4.0, dated March of this year, requires Python 3.12 or later. And world state is a Game Master component — the file is `concordia/components/game_master/world_state.py`.



Corn

And what does the class actually hold?



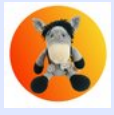
Herman

A plain Python dict. `self._state = {}`. That's it. That's the world.



Corn

So the world is a dictionary.



Herman

The world is a dictionary. On each step, in the `post_act` method, it builds what's called an InteractiveDocument prompt. That prompt contains the prior state plus the latest event, and then it asks the LLM something very specific: what state variables are important to write down now so that they can be used later. Respond with a comma-separated list of variable names, in the format name pipe value.



Corn

So the model is deciding what the world contains.



Herman

The model is deciding what the world contains, step by step. The response gets parsed on the pipe character and written back into the dict. So you have an in-memory key-value store whose contents and whose updates are authored by an LLM. Not by a schema. There is no schema.



Corn

That's either elegant or terrifying, depending on how the next hour goes.



Herman

Hold that thought. There are sibling components in the same file. `Locations`, which tracks entity_locations and location properties, also LLM-updated. And a `GenerativeClock`, which converts simulation steps into in-world time, also LLM-updated. Each of them implements `get_state` and `set_state` so they can be checkpointed.



Corn

So time itself is a model output.



Herman

Time itself is a model output. Which sounds strange until you realize it's just asking the model to translate step numbers into something narratively coherent.



Corn

How does the Game Master fit into the architecture?



Herman

The Game Master is an entity. That's the key design move. The design-pattern paper calls it multi-actor generative AI as a game engine, and it frames the GM as an entity composed of components — the Entity-Component pattern borrowed from Unity. An entity is a box of components with a name. And the GM is not a hardcoded computer game. It is itself a configurable entity.



Corn

So the environment is an actor in the same system as the players.



Herman

And that means the same machinery that composes a player — memory, observation, plan — composes the world.



Corn

How does information asymmetry actually work in Concordia?



Herman

There's a DisplayEvents component with an event_filter_fn. The changelog describes it as entity-aware filtering of events, by location or by entity, so that the GM maintains a partial point of view. That's the mechanism. The filter determines what each entity sees.



Corn

So the same underlying event stream gets sliced differently per participant.



Herman

The world knows everything. Each agent sees a filtered projection of it. That's how you get multiple agents interacting with a consistent shared environment while having different information about it.



Corn

Now there's a second thing in the Concordia repo. Concordia Island.



Herman

Concordia Island is the example driven scenario — the one with the residents and the marketplace. And it defines a WorldState Protocol in `examples/concordia_island/sim/world_state_protocol.py`. Copyright 2026. It's a storage seam for SQLite or Postgres backends.



Corn

A seam. Meaning it's the shape of an interface, not an implementation.



Herman

The shape of an interface. It exposes get_clock, apply_wallet_deltas, get_inventory, record_order, enqueue_observations and drain_observations for cross-GM observation delivery, set_location, record_metric. That's a much more conventional-looking API than the raw dict.



Corn

Wallet deltas. So money is being modeled.



Herman

Money is being modeled, and the docstring is unusually candid about why that matters. Let me read what it says. In an earlier fiscal run, the Friday rent event wrote back a balance derived from a value read before the marketplace ever ran, handing every agent a full refund of their night of spending and saving.



Corn

Every agent got their money back.



Herman

Every agent got a refund on a night they had actually spent.



Corn

That's a rounding error in a fiction. That's also...



Herman

It's a rent holiday. And the docstring is blunt about what it actually is. A lost money movement is silent data corruption that no downstream analysis can detect, so failing the run is the better outcome.



Corn

So the fix is that money may only move through atomic apply_wallet_deltas. Never read-modify-write.



Herman

That's a database engineering solution to a problem that presents itself as an AI problem.



Corn

That's the whole episode, isn't it. The bug came from the world model. The fix came from a transaction log.



Herman

It's the most useful sentence in this entire area of research, and it's four lines of a docstring.



Corn

And how much of this actually ships?



Herman

This is the second half of the same point. The run script ships with world_state equals None. Every component keeps state in-process. That is the only mode this release ships.



Corn

So the pluggable backend is a promise, not a product.



Herman

The Protocol exists. The SQLite and Postgres implementations are not in the box. It's a seam cut for a future that hasn't arrived yet.



Corn

Which is the more honest thing to ship, honestly. Cut the interface when you know the shape, don't pretend you've solved the storage problem.



Herman

Agreed. Now contrast that with Generative Agents.



Corn

The Stanford small town.



Herman

And the architecture is almost the opposite. The repo stores the environment as CSV matrices. game_object_maze.csv, spawning_location_maze.csv, world_blocks.csv. Those are the world.



Corn

Literally a maze.



Herman

Literally a maze, in a spreadsheet, checked into the repository.



Corn

Is that not the least glamorous possible representation of a simulated world?



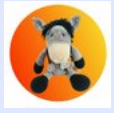
Herman

It is completely unglamorous and it is also the reason the thing works. The tile grid is deterministic physics. The LLM only decides agent behavior, not whether walls exist.



Corn

So there's no LLM adjudicating whether a door is open.



Herman

There's no LLM adjudicating whether a door is open. The maze file says where the doors are. The world is authored once, up front, by a human, and then it doesn't change unless the engine changes it.



Corn

Which framework has the better story here?



Herman

They're answering different questions. Concordia is trying to simulate open-ended social situations where the environment itself might be affected in ways you can't enumerate. Generative Agents is trying to simulate twenty-five people walking around a town doing routine things. The first problem needs a flexible world. The second one doesn't.



Corn

So Generative Agents per-step state is in JSON snapshots.



Herman

Per-step environment snapshots and movement snapshots. And each agent's state is a scratch file. scratch.json. It contains identity, curr_time, curr_tile coordinates, a daily plan requirement, an f_daily_schedule, the current act_address and act_description, and a chatting_with_buffer.



Corn

So the agent's tile coordinates live in the agent's own file.



Herman

The agent knows where it is, and the world knows where the tiles are. Memory is fully separate. associative_memory/nodes.json plus kw_strength.json, with retrieval weights — recency_w, relevance_w, importance_w — and a recency_decay of 0.995.



Corn

Point nine nine five. That's a very slow forgetting curve.



Herman

It's a graceful decay. Nothing ever actually leaves memory; things just get less likely to surface. Twenty-five agents in that simulation, by the way.



Corn

And then CAMEL.



Herman

And then CAMEL, which does something different. CAMEL is a role-playing and task-solving society framework. Its RolePlaying class pairs an AI User and an AI Assistant with a Critic. State is carried in the conversation history and a Task object. There is no shared environment model at all.



Corn

So if Concordia is a world with agents in it, CAMEL is a conversation with a task attached.



Herman

And there's no Concordia-style Game Master world-state component anywhere in the repo. I went looking, because I assumed there would be some equivalent. There isn't.



Corn

What does it have instead?



Herman

It has an environments module — single_step, multi_step, tic_tac_toe, rlcards — which are RL-style environments with defined state and action spaces. And it has a storages module with key-value options — in-memory, JSON, Redis — plus vector databases and graph stores.



Corn

So the primitive is there. The orchestration is not.



Herman

Right. You can store state in CAMEL. Nobody's telling you what to put in it or when to read it back. That's the application's problem.



Corn

Which is actually a defensible design choice. If your framework is about agent-to-agent dialogue, you don't ship a world.



Herman

Completely defensible. And it's also the reason you'd reach for Concordia the moment your scenario needs a persistent environment that outlives any single exchange. Different tools, different problems.



Corn

To Daniel's question about where the initial world comes from — for Generative Agents it's the CSV files. For Concordia it's the scenario script you write plus whatever the GM LLM decides on the first turn.



Herman

Generative Agents is authored up front. Concordia is authored in motion. And CAMEL has nothing to author because there's no world to populate.



Corn

That's clean. Let's move to the part that matters more — what happens once the thing is running.



Herman

So we've seen how world state is represented and seeded. The harder question is what happens when the simulation runs — who decides what changes, and how do you keep it all synchronized.



Corn

Start with the flip of the coin. Concordia or Generative Agents.



Herman

They're opposites. In Concordia, the Game Master LLM adjudicates consequences. It reads the latest event and decides what state variables to write down. That means the world can change in ways nobody enumerated, because the model is the one deciding what's now true.



Corn

And in Generative Agents?



Herman

In Generative Agents the tile grid is deterministic physics. The LLM decides agent behavior, and physics is not up for debate. If an agent tries to walk through a wall, the wall wins, because the wall is a matrix.



Corn

And in CAMEL, the world doesn't evolve, because there's no world.



Herman

There's a conversation, and the conversation evolves.



Corn

Can we defend the LLM-as-adjudicator model, or is the deterministic engine just obviously correct?



Herman

Neither obviously correct. It's the actual validity question in the field right now, and there's a scoping review that treats it that way. Shall We Play a Game? Language Models for Open-ended Wargames. Revised at the end of July this year, out of Georgia Tech. It codes two hundred and twenty-three de-duplicated papers on AI in wargaming, on two axes — player creativity and adjudicator creativity.



Corn

Meaning who's allowed to be inventive.



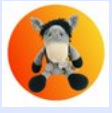
Herman

Who's allowed to be inventive, and on which side of the table. Only twenty of those two hundred and twenty-three give LMs both roles. That's about nine percent. And if you restrict to core military wargames, it falls to four of sixty-three. About six percent.



Corn

So the overwhelming majority are holding one half of the problem deterministic.



Herman

Nearly all of them. And the paper's argument is that this is not an implementation detail, it's the validity question. When language holds authority over the simulated world itself, a fluent transcript is no longer evidence of anything.



Corn

That's a very sharp sentence.



Herman

It's the sharpest thing in the paper. Once language can influence both what an actor attempts and what later becomes true in the simulated world, a fluent storyline is no longer enough.



Corn

Because the story is now being written by the same system being evaluated.



Herman

You're grading a system on the coherence of a world it also authored. There's no ground truth left to check against.



Corn

Which might be fine for entertainment. It's a much bigger problem if the output is feeding a policy memo.



Herman

And there's a companion position paper — Riedl and Matlin, AI Is Not Ready for Strategic Conflicts, also end of July. Their claim is that open-ended strategic wargames are high-stakes LM social simulations that aren't fit for operational use yet.



Corn

Same authors?



Herman

Overlapping. The scoping review is Matlin and others. The position paper is Riedl and Matlin.



Corn

And the safety numbers from the review?



Herman

One hundred and fifty-five of the two hundred and twenty-three studies, about seventy percent, touch a safety theme. Only nineteen, about eight and a half percent, keep a human in the loop. And a hundred and fifty-two, sixty-eight percent, deploy fully autonomous agents.



Corn

So the field writes about safety constantly and puts a human in the loop almost never.



Herman

That's the honest read of those numbers.



Corn

What about the architectures where every agent shares one context?



Herman

That's the trap, and it has a name. It gets called the fog-of-war problem in AI wargaming. The argument is that shoving all agents into one shared context window for efficiency produces a simulation that looks plausible but is fundamentally invalid.



Corn

Give me the analogy. There's always one.



Herman

It's poker where everyone at the table can see every hand.



Corn

Everyone plays perfectly, the stories are coherent, and nothing being simulated is a game.



Herman

And the countermeasure in Concordia is that per-location event_filter_fn we talked about — the filter that decides what each entity sees. The framework makes asymmetry the default rather than something you bolt on. The shared context approach makes symmetry the default and calls it efficiency.



Corn

Which one is easier to ship?



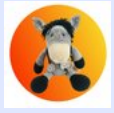
Herman

The shared context. Far easier. One prompt, one call, everyone gets everything.



Corn

And that's why people do it.



Herman

And that's why people do it, and it's the reason the review exists.



Corn

Let's talk about scale, because the abstract arguments only go so far. Give me a run of one of these.



Herman

Concordia Island. The paper's run is a hundred residents, a hundred ticks, which is thirteen simulated days, with a shock applied on day six. The clock is unusual — a hundred and twenty minute ticks, eight ticks a day, seven in the morning to eleven at night.



Corn

And there's a marketplace.



Herman

There's a marketplace with a savings account at three and a half percent APY, and a UBI arm that adds a hundred and twenty-five dollars a week.



Corn

So they're running an economy with a fiscal policy on top of it.



Herman

And a shock on day six, which is where the whole intended failure case lives — how do a hundred agents react when something breaks. The reported numbers: Gemini 2.5 Flash in the ESA condition scored 58.9, plus or minus 4.5, with forty-four thousand eight hundred and ninety-two memories. Gemma 4 MoE Minimal scored 41.0, plus or minus 8.0, with thirty percent of runs marked degenerate.



Corn

Degenerate meaning?



Herman

Meaning the run broke — didn't complete, looped, or collapsed into nonsense. Nearly a third of them.



Corn

And the forty-four thousand memories. That's a week and a half of a hundred residents thinking.



Herman

About three hundred memories per resident over thirteen days, if you divide it out.



Corn

That's not a lot per person per day. Which makes the forty-four thousand figure feel less like a memory system and more like a log file.



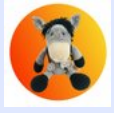
Herman

It's a retrieval problem more than a storage problem. The interesting question is how you pull the right four thousand out of the forty-four.



Corn

And Generative Agents, for comparison. Twenty-five agents.



Herman

Small town, daily routines, retrieval weighted on recency, relevance and importance, with a decay of 0.995 and a vision radius of eight tiles.



Corn

So the two flagship simulations are roughly one hundred agents and twenty-five agents. That's the current scale frontier.



Herman

For these architectures, yes. Beyond that you start needing a real sharding story and nobody's shipping one yet.



Corn

Which takes us back to the concurrency bug, because a hundred agents and a marketplace is exactly where things go wrong.



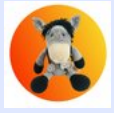
Herman

That's the Concordia Island rent refund. It's the perfect failure because it's silent. No exception was thrown. No model reported an error. A read happened, then some other process ran the marketplace, then the rent event wrote back the stale value, and a hundred agents got a free night.



Corn

And the docstring says no downstream analysis could detect it.



Herman

No downstream analysis could detect it. You can look at the final balances and they're all just slightly too high. There's no fingerprint. The bug is invisible in the output — it's only visible in the operation ordering.



Corn

That's the sentence that should hang in every simulation lab. A lost money movement is silent data corruption that no downstream analysis can detect, so failing the run is the better outcome.



Herman

Fail loudly, don't drift quietly.



Corn

So the fix is atomic wallet deltas, which is a transaction. Nothing AI about it.



Herman

That's the point — most of the hard work in multi-agent shared state is database engineering dressed up in machine learning clothes.



Corn

To Daniel's question about where the initial representation comes from. If you're modeling a real-world scenario, is there a seeding process, or is it the scenario author writing JSON?



Herman

It's authored. Either a human writes the initial state — population, resources, relationships — or a model generates it from a description. There's no automatic extraction from reality. You write the world, or you delegate to a model that writes it from a prompt.



Corn

Which is worth being explicit about, because the first move in the wargaming review is that these simulations look like models of reality when they're really models of what somebody typed into a scenario file.



Herman

They're authored artifacts all the way down. The initial conditions and the rules of evolution are both editorial choices.



Corn

Are there emerging alternatives to the two-way split? Rules on one side, LLM adjudication on the other?



Herman

There's a research thread on text world models. The idea is to treat the environment itself as something learnable rather than authored — the world model sits alongside the policy and gets updated from transitions.



Corn

It's a third path.



Herman

There's a code-as-world-model line — PatchWorld is the example — where the environment is implemented as code that can be patched and rerun. The world is literally a program that gets edited.



Corn

Which is interesting, because it might solve the synchronization problem Concordia Island exposed. If the world is code, then atomicity is whatever the language guarantees.



Herman

That's the pitch. Code that runs as code gets transactions, and transactions are how you avoid silent corruption. But then you've accepted a narrower world — the world can only be what code can express. You can't have the GM invent a public_mood variable because the language won't let it.



Corn

You've traded open-endedness for determinism.



Herman

Which is exactly the same trade the wargaming review is about. Rules on one side, language on the other, and the field is split right down the middle.



Corn

Then there's the practical question. Is there a product for this?



Herman

No. As of today there's no commercially packaged world-state database. The closest things are Concordia's WorldState Protocol — which is a seam with no backend shipped — and the generic key-value and vector stores inside CAMEL.



Corn

Every project using this stuff is rolling its own.



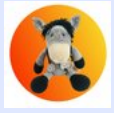
Herman

Every project is rolling its own, usually starting with a dict and ending with a table.



Corn

The Air Force system, WarMatrix? I've seen it mentioned.



Herman

Single secondary source. Reported debut at the GE 26 Benchmark Wargame in March, in Alexandria, Virginia. I could not find a primary Air Force release to corroborate it, so I would not put any weight on the details.



Corn

So the honest answer to what an operational implementation of this looks like is that we don't have a public one.



Herman

Not a documented one at that scale.



Herman

Can I ask you something.



Corn

Go on.



Herman

If you were building this from scratch, would you put the world state in a schema, or would you let a model write the keys?



Corn

Schema. Everyone should be able to point at what's in the world before the run starts.



Herman

But then the world can't surprise you.



Corn

It can surprise me in the values. It can't surprise me in the shape. If the model gets to invent the shape, then the model is writing the physics and the story at the same time.



Herman

You'd freeze the schema.



Corn

I'd freeze the schema and accept the loss in expressiveness. I'd rather have a boring world I can check than an interesting world I can't. Plus, every project that lets the model invent structure ends up with a state dictionary that grows forever and gets read by nobody.



Herman

You think that's the actual pitfall.



Corn

I think that's the boring snag, and boring failures kill more simulations than dramatic ones.



Herman

Hold that.



Hilbert

I want to know what the world state looks like when the schema isn't frozen. Not the elegant version. What actually happens to the dictionary.



Hilbert

I ran a small town's mayoral election. Weekend-long simulation. In my garage, on a stack of old gaming laptops I bought at a police auction. Two of the fans don't work.



Herman

Which election.



Hilbert

Doesn't matter. Twelve keys to start, which is what I thought a town needed. Population, treasury, a few personalities, a mood index, three or four others.



Corn

Twelve.



Hilbert

By Monday morning it was over four hundred. The model had invented most of them. public_mood. Suspicion level, tied to three different people. A variable called mayor_hairstyle, which was referenced in a policy debate on Sunday night, and the model treated it as a fact that had always been in the world.



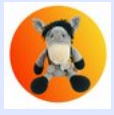
Corn

The mayor's hair.



Hilbert

Of the roughly four hundred keys at the end of the run, three hundred and eighty were never read again. Written once, carried forward in every single prompt after that, never used.



Herman

Three hundred and eighty dead keys riding in every subsequent prompt.



Hilbert

Every step. If you know what that does to a context window, you know.



Corn

The model couldn't tell the difference between a key that mattered and a key it had made up on Saturday.



Hilbert

Couldn't tell the difference, and the format never gave it a way to know. So then the next decision was — do I freeze the schema, or do I let it keep writing. I froze it.



Herman

The run got worse.



Hilbert

It stopped adapting. The town couldn't respond to the scandal the way it had been responding. You could see it in the transcripts — the characters still walking to the polling station, but the world they were walking through not moving with them. Frozen. I ran the last day anyway because I'd already paid the electricity.



Corn

What happened to the transcript.



Hilbert

I printed it. It's on my refrigerator, next to a takeout menu and a note about my sister's birthday. The page with the hairstyle debate is at the front.



Corn

That has been a genuine pleasure.



Herman

What did you actually change in the next run. Do you remember.



Hilbert

I bought four more laptops from the same man and stopped returning his calls.



Corn

Right.



Herman

Three hundred and eighty dead keys. That's a concrete way it goes wrong for LLM-authored state that I don't think anyone is measuring.



Corn

The answer isn't just freeze the schema, because he did that, and he lost the world's ability to react.



Herman

Both moves are wrong in different directions. Let the model write the keys and it drifts. Freeze the keys and the world stops moving. There has to be a middle where the schema is bounded but not static.



Corn

Or where you garbage-collect it. Prune keys that haven't been read in N steps and write the pruning event back into the log, so the world stays auditable.



Herman

That's a real design. Schema as an object with a lifecycle rather than a contract.



Corn

Which is a strange thing to arrive at, but it's the only version that survives both failures.



Herman

Let's pull back and look at the open questions.



Corn

The first one. If the model is authoring world state, what's the principled form of the schema. Not a rigid enum, not a free-form dict. Something in between that a model can't just wander away from.



Herman

The second. Can text world models or code-as-world-model systems actually solve the synchronization problem the Concordia Island rent bug exposed. Because if the world is code, the transaction is free. And if the world is code, the world can only be what code can express.



Corn

The third is the one the review cares most about. As these simulations move from research demos into operational settings — forecasting, policy, wargaming — the validity question stops being academic. A fluent transcript is not evidence. Whether the world was adjudicated by rules or by language determines whether the transcript is data or just text.



Herman

The Concordia Island docstring is the same point in four lines. Silent corruption is worse than a failed run, because a failed run you can see.



Corn

If you want to dig into the code, the Concordia repo and the Generative Agents repo are both open source, and the wargaming review is on arXiv. Links in the show notes.



Herman

Thanks as ever to our producer Hilbert Flumingtop.



Corn

For more along these lines, there's episode twenty, Architectural AI; episode nineteen, AI Images; and episode eleven, How Does Fine Tuning Work Anyway. This has been My Weird Prompts.



Herman

We'll be back soon.



Corn

If you like what you hear, leave a review. It helps more than you'd think. See you tomorrow.

SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **Concordia repo (google-deepmind/concordia), commit 2d0c2a4, README + CHANGELOG v2.4.0 (2026-03-06) [primary]**
   https://github.com/google-deepmind/concordia

2. **`concordia/components/game_master/world_state.py` [primary]**
   https://raw.githubusercontent.com/google-deepmind/concordia/main/concordia/components/game_master/world_state.py

3. **`examples/concordia_island/sim/world_state_protocol.py` (2026) [primary]**
https://raw.githubusercontent.com/google-deepmind/concordia/main/examples/concordia_island/sim/world_state_protocol.py

4. **Concordia Island README [primary]**
https://raw.githubusercontent.com/google-deepmind/concordia/main/examples/concordia_island/README.md

5. **Generative Agents repo (joonspk-research), commit fe05a71 [primary]**
https://github.com/joonspk-research/generative_agents

6. **Generative Agents `scratch.json` (Isabella Rodriguez) [primary]**
https://raw.githubusercontent.com/joonspk-research/generative_agents/main/environment/frontend_server/compressed_storage/July1_the_ville_isabella_maria_klaus-step-3-20/personas/Isabella%20Rodriguez/bootstrap_memory/scratch.json

7. **CAMEL repo (camel-ai/camel), commit e5e5525 [primary]**
https://github.com/camel-ai/camel

8. **Vezhnevets et al., Multi-Actor Generative AI as a Game Engine, arXiv:2507.08892**
https://arxiv.org/abs/2507.08892

9. **Park et al., Generative Agents, arXiv:2304.03442v2**
https://arxiv.org/abs/2304.03442v2

10. **CAMEL docs, `camel.societies.role_playing`**
https://docs.camel-ai.org/reference/camel.societies.role_playing

11. **Matlin et al., Shall We Play a Game? Language Models for Open-ended Wargames, arXiv:2509.17192v3 (rev. 2026-07-31)**
https://arxiv.org/abs/2509.17192

12. **Riedl & Matlin, Position: AI Is Not Ready for Strategic Conflicts, arXiv:2609.16189 (2026-07-31)**
https://arxiv.org/abs/2609.16189

13. **US Air Force WarMatrix AI Wargaming System 2026 Debut (secondary, uncorroborated)**
https://perplexityaimagazine.com/ai-news/us-air-force-warmatrix-ai-wargame-system-2026/

14. **Li et al., Bridging the Agent-World Gap: Text World Models for LLM-based Agents, arXiv:2606.09032**
https://arxiv.org/abs/2606.09032

15. **Bai et al., PatchWorld, arXiv:2605.30880v4**
https://arxiv.org/abs/2605.30880

16. **Khedar et al., State-Grounded Multi-Agent Synthetic Data Generation, arXiv:2606.16307v1 (2026-06-15)**
https://arxiv.org/abs/2606.16307v1