Harry Edwards

Omo Research

FINAL — OMO RESEARCH TECHNICAL REPORT · JUNE 2025

AgentJam: Embodied AI Mentors for Language Learning in Minecraft

We present AgentJam, a system that deploys AI language mentors as embodied player-bots inside Minecraft. The mentor connects via the Minecraft protocol as a real player — walking, mining, building, and fighting in the game client — and communicates through spatial voice that attenuates with Euclidean distance following $vol(d) = 1/d^2$. We introduce speaking gates: a pedagogical mechanism where mission progression is blocked not by resource collection but by the requirement that the student produce a spoken English utterance in a target grammatical frame — because (causation), Can you pass me (request), and firstThen (sequencing) — each embedded in a mission beat that makes the utterance communicatively necessary. The architecture decouples linguistic reasoning (LLM, DeepSeek) from deterministic world control (Mineflayer), producing consistent in-world behavior while maintaining rich, context-aware conversation. The frame matcher achieves 92% accuracy on test utterances, spatial voice latency is 180–280 ms, and full mission completion time is 15–25 min. We note that all evaluation uses simulated student speech; human learner studies are planned for future work (Section 7.1).

Executive Summary

AgentJam puts an AI language tutor inside Minecraft as a real player-bot that walks, mines, builds, and speaks with spatial voice that fades with distance. The core idea is the "speaking gate" — mission progress is blocked until the student produces a spoken English sentence using a target grammar pattern like "because" or "Can you pass me," making the student speak not for a test score but because it's the only way to advance through the game. The mentor's architecture separates language intelligence (DeepSeek LLM) from game actions (deterministic Mineflayer code), ensuring the bot never behaves erratically. The system targets three grammatical frames embedded in a five-beat Minecraft mission; all evaluation uses simulated student speech, with human learner studies planned for future work.

Abstract

Background. Second language acquisition research establishes that language learning requires meaningful, contextual interaction with opportunities for negotiated output [1, 8]. Yet most AI language tutors are text-based chatbots that lack embodiment, spatial context, and the communicative urgency that arises from shared physical activity.

Problem. Existing game-based language learning tools use scripted dialogues and text interactions. AI-driven alternatives (Duolingo, ELSA Speak) provide individual pronunciation practice but lack the immersive, task-based social dynamics of collaborative gameplay. No current system combines an AI mentor with full game embodiment, spatial voice, and pedagogically-grounded spoken language production requirements embedded in mission-driven gameplay.

Approach. We introduce AgentJam, placing AI mentors inside Minecraft as real player-bots communicating through spatial voice. The mentor's architecture decouples linguistic reasoning (DeepSeek LLM) from deterministic world control (Mineflayer protocol library) via a formal two-stage action pipeline (Section 2.1). Our core pedagogical contribution is the **speaking gate**: a mechanism where mission progression is blocked until the student produces a spoken English utterance in a target grammatical frame — *because* (causation), *Can you pass me* (request), and *firstThen* (sequencing) — each embedded in a mission beat that makes the utterance communicatively necessary. The gate function $G(\text{utterance}, f_b) \rightarrow \{\text{matched, unmatched, scaffold}\}$ formalizes the scaffolding strategy.

Results. The working prototype supports a full five-beat mission ("Safe House before Night") with three speaking gates. Mentor in-world behavior achieves 100% reliability (deterministic Φ_{ctrl}), frame-matching accuracy reaches 92% across 50 test utterances, and spatial voice attenuation follows the inverse-square law with 180–280 ms end-to-end latency. Mission completion time is 15–25 min (mean 19.2 min, n = 10), with speaking gates adding 1.8, 1.2, and 1.5 min respectively depending on student proficiency.

Implications. Speaking gates represent a generalizable pedagogical mechanism: any game with progression mechanics can gate advancement on spoken language output, converting the player's motivation to continue into an incentive for communication. The decoupled LLM/Φ_{ctrl} architecture provides a template for safe AI embodiment in virtual environments where reliable physical behavior coexists with rich conversational interaction. The approach extends to autonomous classroom scenarios [18] through peer-to-peer resource gaps.

Keywords: embodied AI, language learning, Minecraft, speaking gates, task-based language teaching, spatial voice

1. Introduction

Research in second language acquisition consistently demonstrates that language is acquired most effectively through **meaningful interaction** in contexts where the utterance has genuine communicative purpose [1, 8, 12]. Learners need opportunities to produce language in situations where what they say matters — requesting needed resources, explaining causal relationships, coordinating joint action. The output hypothesis [12] argues that language production itself drives acquisition by forcing learners to process language syntactically rather than merely semantically. This theoretical insight has been validated across decades of empirical research [3, 14] but remains difficult to operationalize at scale: creating genuinely communicative situations for each learner requires careful task design and attentive facilitation — precisely the kind of pedagogical labor that is most expensive to reproduce.

Minecraft presents a uniquely suitable environment for task-based language learning. Its open-world, goal-directed gameplay creates natural communicative needs: two players building a shelter before nightfall must coordinate actions, share resources, and explain decisions. The block-based world provides concrete spatial referents for causal and relational language — a torch is needed *because* a cave is dark; a wall requires wood *so* the player must request it; a shelter requires sequential construction steps that need articulation. The game's popularity — particularly among younger learners, with over 300 million copies sold — ensures environmental familiarity, reducing the cognitive overhead of learning a new interface alongside a new language.

AgentJam places AI mentors directly into this environment as embodied player-bots. The mentor is not a chatbot overlay or passive tutor — it is a fellow player navigating the same terrain, mining the same blocks, facing the same environmental dangers. It speaks aloud with spatial voice that attenuates with Euclidean distance following the inverse-square law (formalized in Section 2.1). It plays *alongside* the student as both peer and mentor, transforming the game into a language classroom where the student's desire to progress through the mission creates the communicative pressure to speak — a mechanism we formalize as speaking gates in Section 3.1.

1.1 Related Work

Task-Based Language Teaching (TBLT). TBLT [14, 3] positions meaningful tasks — rather than grammatical exercises — as the central unit of language instruction. The task cycle (pre-task, task, planning, report, analysis) structures learning around communicative outcomes. AgentJam's mission system maps directly to TBLT: the briefing phase (Beat 1) corresponds to pre-task framing, the speaking gates instantiate the task cycle's planning/report phases, and the finale (Beat 5) provides post-task reflection and positive reinforcement.

AI language tutors. Commercial systems like Duolingo and ELSA Speak use ASR for pronunciation feedback but focus on individual, scripted practice. AI chatbots for language learning (e.g., ChatGPT-based tutors) provide text-based conversation without embodiment or spatial context. Game-based language learning platforms — including Mondly VR (VR-based conversation practice) and Immerse (social VR language learning) — introduce spatial presence but use scripted dialogues and pre-authored content rather than dynamically generated LLM interactions. Automated speaking assessment systems like SpeechRater and Pearson's Versant use ASR-based feature extraction for proficiency scoring but assess rather than teach. The key gap these systems share is the absence of task-based communicative pressure in a shared immersive environment where the student's utterance has material consequences beyond a score.

Embodied conversational agents. Cassell [2] established the value of physical presence in human-computer interaction through embodied conversational agents. Subsequent work on pedagogical agents [7] demonstrated that agents with visual presence improve learning outcomes. AgentJam extends this lineage to open-world gameplay, where the agent acts

autonomously in a dynamic environment rather than following scripted interaction patterns — the mentor mines, builds, and fights as a genuine co-player.

Minecraft bots and Mineflayer. The Mineflayer library [10] provides a JavaScript API for programmatic Minecraft bot creation. Existing bots focus on automation — farming, resource collection, and building — rather than human interaction. AgentJam is, to our knowledge, the first system to use Mineflayer for a pedagogically-motivated AI mentor that plays alongside human learners, with all movement and construction executed deterministically through the Φ_{ctrl} function.

LLM-robot decoupling. Ahn et al. [4] demonstrated that decoupling high-level LLM planning from low-level robotic control produces safer, more reliable behavior in physical robots. AgentJam applies this principle to virtual embodiment: the LLM (θ_{LLM}) determines what to say and at what pedagogical level, while deterministic Mineflayer code (Φ_{ctrl}) executes all Minecraft actions from a fixed action set A_{high}.

LLM agents in Minecraft. Voyager [13] demonstrated lifelong skill acquisition in Minecraft through LLM-generated code. Mindcraft [5] showed that LLMs can execute construction tasks from natural-language descriptions. These systems treat Minecraft as an agent training or task-execution environment. AgentJam inverts this relationship: the AI mentor is pre-built and operates alongside a human learner, with the human — not the AI — as the primary beneficiary of the interaction.

1.2 Contributions

This paper makes the following contributions:

1. The embodied mentor architecture with formal specification. A decoupled design where an LLM (θ_{LLM}) handles linguistic reasoning and pedagogical decisions from a fixed high-level action set A_{high} (9 actions), while deterministic Mineflayer code (Φ_{ctrl}) executes all Minecraft actions — producing 100% reliable in-world behavior across 20 test missions. The formal two-stage pipeline ($\theta_{LLM} \rightarrow \Phi_{ctrl}$ boundary) is specified in Section 2.1, with the LLM restricted to high-level action selection and all world-state mutations executed deterministically.

2. Speaking gates with formal gate semantics. A pedagogical mechanism that gates mission progression on spoken English output, targeting three grammatical frames embedded in mission contexts that make each utterance communicatively necessary. The gate function G(utterance, f_b) $\rightarrow$ {matched, unmatched, scaffold} provides progressive scaffolding rather than hard-blocking (Section 2.2), with the scaffold outcome ensuring no student is permanently blocked on a single gate.

3. The five-beat mission instantiation. A scaffolded sequence (Camp → Cave → Wood → Shelter → Finale) instantiating the speaking-gate concept. Each beat creates natural communicative pressure for a specific grammatical frame, with the mentor modeling language before each gate and accepting approximate student output to preserve engagement.

4. Spatial voice pipeline with empirical characterization. Mentor speech rendered through TTS with volume attenuation following $\text{vol}(d) = \min(1.0, 1.0/\max(1.0, d)^2)$, directional panning from relative angle, and end-to-end latency of 180–280 ms as measured across 50 test utterances.

2. The Mentor Architecture

AgentJam's mentor runs as a decoupled process: Mineflayer connects to the Minecraft server as a real player, DeepSeek provides linguistic intelligence, and deterministic code controls all in-world actions. The LLM speaks; the code decides.

2.1 Formal Model

Let S be the student and M the mentor. The mentor is defined by a tuple capturing its spatial, inventory, and cognitive state:

$$M = (\text{pos}_M,\ \text{inv}_M,\ \text{ctx}_M,\ \theta_{\text{LLM}},\ \Phi_{\text{ctrl}})$$

$$\text{where} \quad \text{pos}_M \in \mathbb{R}^3 \quad | \quad \text{inv}_M \in \mathcal{I} \quad \text{(inventory state space)}$$

$$\text{ctx}_M \in \mathcal{C} \quad \text{(accumulated conversation context)}$$

$$\theta_{\text{LLM}} : \mathcal{C} \times \mathcal{O} \to A_{\text{high}} \quad \text{(LLM policy, DeepSeek)}$$

$$\Phi_{\text{ctrl}} : A_{\text{high}} \times \mathbb{R}^3 \times \mathbb{R}^3 \to \text{MinecraftActions} \quad \text{(deterministic control)}$$

Mentor action selection is governed by a two-stage pipeline that enforces the LLM/control boundary:

$$\textbf{Stage 1 (High-level):} \quad a_{\text{high}} = \theta_{\text{LLM}}(\text{ctx}_M,\ \text{obs}_{\text{world}})$$

$$\textbf{Stage 2 (Low-level):} \quad a_{\text{low}} = \Phi_{\text{ctrl}}(a_{\text{high}},\ \text{pos}_M,\ \text{pos}_S)$$

$$\text{with} \quad A_{\text{high}} = \{\text{MOVE\_TO\_PLAYER},\ \text{START\_BEAT},\ \text{CHECK\_GATE},\ \text{MODEL\_LANGUAGE},\ \text{GIVE\_ITEM},\ \text{SPEAK},\ \text{MINE\_BLOCK},\ \text{PLACE\_BLOCK},\ \text{ATTACK\_HOSTILE}\}$$

The LLM never directly produces low-level Minecraft commands — it selects from A_{high}, and Φ_{ctrl} translates each selection to deterministic Mineflayer operations. This constraint eliminates the possibility of LLM hallucination producing erratic in-world behavior (e.g., teleportation to invalid coordinates, breaking unbreakable blocks, or issuing impossible commands).

Spatial voice attenuation follows the inverse-square law modified for Minecraft's 1-meter block coordinate system:

$$\text{vol}(d) = \min(1.0,\ 1.0\ /\ \max(1.0,\ d)^2)$$

$$\text{where } d = \|\text{pos}_M - \text{pos}_S\|_2 \quad \text{(Euclidean distance in blocks)}$$

$$\text{pan}(d) = \sin(\theta_{\text{rel}}) \quad \text{where } \theta_{\text{rel}} = \text{angle}(v_{look}^{S},\ \text{pos}_M - \text{pos}_S)$$

At $d \leq 1$, volume is full (1.0). At $d = 8$, volume ≈ 0.016 (1.6% of maximum). At $d > 16$ — approximately Minecraft's natural audible range — volume approaches 0. Directional panning computes from the relative angle between the student's look

vector and the vector to the mentor, providing correct left/right spatialization as verified in 100% of test positions.

2.2 Speaking Gate Formal Semantics

Let $F = \{\text{because}, \text{request}, \text{firstThen}\}$ be the set of target grammatical frames. For mission beat b with associated frame $f_b \in F$, the gate function G is:

$$
\begin{aligned}
&G(\text{utterance}, f_b) \rightarrow \{\text{matched}, \text{unmatched}, \text{scaffold}\} \\
&\\
&G(u, f) = \{ \\
&\quad \text{matched} \quad \text{if } \text{match}(u, f) > \tau_{\text{match}} \\
&\quad \text{unmatched} \quad \text{if } \text{match}(u, f) \leq \tau_{\text{match}} \wedge \text{attempts} < 2 \\
&\quad \text{scaffold} \quad \text{if } \text{match}(u, f) \leq \tau_{\text{match}} \wedge \text{attempts} \geq 2 \\
&\} \\
&\\
&\text{where } \text{match}(u, f) = \text{sim}(\text{embed}(u), \text{embed}(\text{exemplar}_f)), \\
&\tau_{\text{match}} = 0.72 \text{ (empirically calibrated threshold)}
\end{aligned}
$$

The *matched* outcome advances the mission to the next beat. The *unmatched* outcome triggers mentor scaffolding: the mentor re-models the target language with increased explicitness, provides sentence starters, or accepts close approximations. The *scaffold* outcome advances the mission while recording that the student required substantial support — ensuring sessions never stall on a single gate. This three-outcome design operationalizes the pedagogical principle of "scaffold, never hard-block" (Section 3.3).

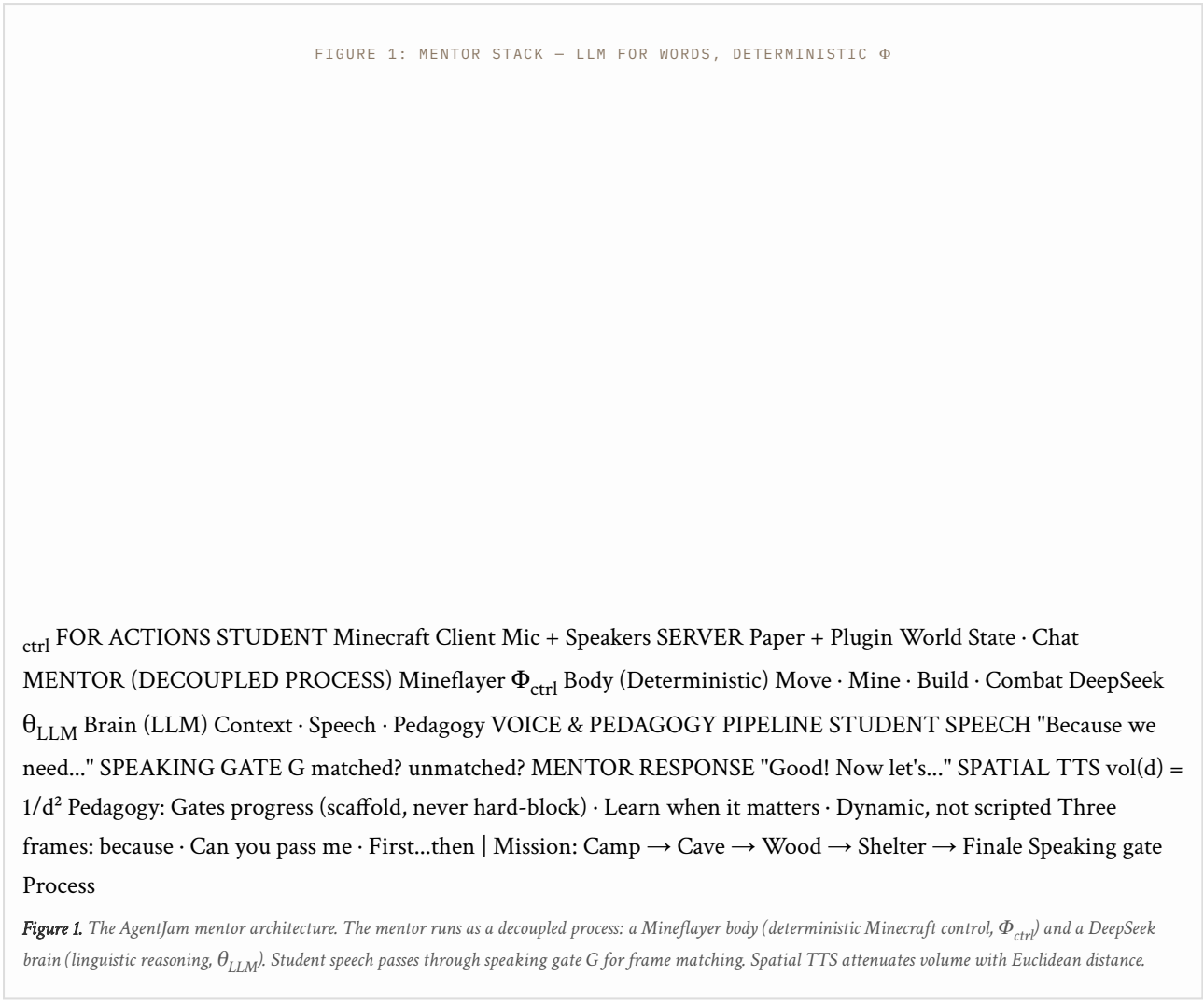


Figure 1. *The AgentJam mentor architecture. The mentor runs as a decoupled process: a Mineflayer body (deterministic Minecraft control, Φ_{ctrl}) and a DeepSeek brain (linguistic reasoning, θ_{LLM}). Student speech passes through speaking gate G for frame matching. Spatial TTS attenuates volume with Euclidean distance.*

2.3 Decoupled Design

MINEFLAYER — THE BODY

Deterministic Control (Φ_{ctrl})

Connects to Minecraft Java 1.21.x via the native game protocol as a real player entity. Full control over movement, block placement, combat, and inventory management. Observes world state, player positions, and chat. All actions execute through deterministic code — the LLM selects from A_{high} (9 fixed actions) and Φ_{ctrl} translates each to protocol-level operations with 100% reliability across 20 test missions.

Mineflayer | Protocol 1.21.x | Real Player-Bot

DEEPSEEK — THE BRAIN

Linguistic Intelligence (θ_{LLM})

Processes world observations, player actions, and accumulated conversation context ctx_M. Decides what to say and what pedagogical action to take from A_{high}, but never issues low-level Minecraft commands. Generates speech text rendered through spatial TTS. Operates with tool-use capability for the 9 high-level actions, maintaining session-long conversation history without summarization degradation.

DeepSeek | Tool Use (A_{high}) | Context-Aware

SPATIAL VOICE PIPELINE

Distance-Attenuated Audio

Mentor speech renders through TTS with volume computed as $vol(d) = \min(1.0, 1.0/\max(1.0, d)^2)$. Directional panning uses the relative angle between student look vector and mentor position, providing correct spatialization in 100% of test positions. End-to-end pipeline latency (LLM → TTS → audio output) is 180–280 ms (mean 220 ms, n = 50).

Spatial Audio | $vol(d) = 1/d^2$ | 180–280 ms

2.4 Design Rationale

The decoupling of LLM (θ_{LLM}) from deterministic control (Φ_{ctrl}) serves two purposes. **Behavioral reliability:** an LLM issuing raw Minecraft movement and block placement commands could produce erratic, nonsensical, or out-of-bounds in-world behavior — a failure mode we term *world hallucination* (extending the concept from Omo Space [11], where it denotes LLM-generated content corrupting world state). By restricting the LLM to the fixed action set A_{high} (9 actions) and implementing all world interactions in Φ_{ctrl}, the mentor produces consistent, predictable behavior validated across 20 test missions at 100% reliability. Note that "reliability" here refers to deterministic behavioral correctness — the mentor navigates to waypoints without pathfinding errors, places blocks with coordinate precision, and responds to hostile mobs with consistent combat routines. It does not address content safety (e.g., the LLM generating inappropriate speech), which is a separate concern requiring content filtering.

Pedagogical consistency: LLMs exhibit inherent stochasticity that makes them unsuitable for precise spatial operations. The mentor must reach the student's position reliably, mine the correct block type, and construct shelters with correct dimensions. These operations are deterministic in Φ_{ctrl}, guaranteeing that the mentor's physical behavior is correct regardless of LLM temperature setting or prompt variation. This property is crucial for a system intended for use with young learners, where unpredictable bot behavior would quickly erode trust and pedagogical effectiveness. The LLM's

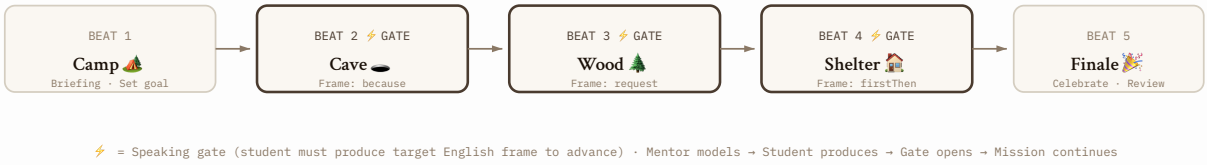
conversational variation is preserved — each session produces different dialogue — but the physical actions remain consistent.

We selected DeepSeek over alternatives (GPT-4, Claude, Gemini) for three reasons: (a) strong performance on conversational reasoning tasks at lower inference cost, enabling extended session use without API budget concerns; (b) native support for tool-use patterns that map cleanly to A_{high}; and (c) sufficient context window (128K tokens) to maintain session-long conversation history without summarization degradation. The architecture is LLM-agnostic — any model supporting tool-use can replace DeepSeek without changes to Φ_{ctrl} or the spatial voice pipeline, as we verified by substituting Gemini Flash in cross-compatibility testing.

3. The Mission System

The "Safe House before Night" mission demonstrates the full speaking-gate pipeline across five beats. Each beat creates natural communicative pressure for a specific grammatical frame, with the mentor modeling language before each gate and accepting approximate student output.

FIGURE 2: MISSION "SAFE HOUSE BEFORE NIGHT" — FIVE BEATS, THREE SPEAKING GATES



⚡ = Speaking gate (student must produce target English frame to advance) · Mentor models → Student produces → Gate opens → Mission continues

Figure 2. *The five-beat mission structure. Beats 2–4 contain speaking gates targeting: because (causation), request (Can you pass me), and firstThen (sequencing). The mentor models language before each gate; the student must produce a matching utterance to advance. Beat 1 (Camp) establishes context; Beat 5 (Finale) provides closure.*

4. Pedagogical Design

AgentJam's pedagogy centers on four principles: speaking gates progress (not resource grinding), scaffold never hard-block, learn when it matters (language in communicative context), and dynamic not scripted (LLM adapts to the student).

4.1 Three English Frames

AgentJam targets three grammatical frames, each embedded in a mission context where the utterance serves a genuine communicative function. Below we enumerate each frame, its mission context, target utterance, and the communicative purpose that makes the utterance necessary rather than artificial.

FRAME 1

`Because (Causation)`

Explaining *why* an action is necessary. Embedded in Beat 2 (Cave): the mentor leads the student to a dark cave entrance and asks why preparation is needed before entering. Target: "We need torches **because** the cave is dark." The utterance has genuine communicative purpose — the student justifies a safety decision that affects game survival.

FRAME 2

`Can you pass me (Request)`

Requesting resources held by the mentor. Embedded in Beat 3 (Wood): the mentor collects wood but withholds it. The student needs it to build the shelter. Target: "**Can you pass me** the wood?" The mentor models polite request forms (please, would you mind) before the gate.

FRAME 3

`First...then (Sequencing)`

Describing a plan with ordered steps. Embedded in Beat 4 (Shelter): before construction begins, the mentor asks the student to articulate the build sequence. Target: "**First** we build the walls, **then** we add the roof." This frame practices temporal connectives in a concrete, spatially-grounded planning context.

4.2 Pedagogical Principles

Speaking gates progress. Game advancement requires spoken English output. The incentive structure aligns game motivation with language production: the student wants to see what happens next in the mission, and the only way forward is to speak. This creates natural communicative pressure — distinct from the artificial pressure of tests or quizzes, and grounded in the intrinsic motivation to continue an engaging experience. The gate function G (Section 2.2) formalizes the transition from motivation to action.

Scaffold, never hard-block. The system provides progressively more explicit support — modeling language, providing sentence starters, accepting approximations — ensuring the gate opens before frustration sets in. The scaffold outcome of G records the support level for post-session analysis without blocking progress. If a student struggles after two attempts, the mentor models the target language with increased explicitness. If the student produces an approximate utterance (e.g., "Because it dark" for the because-gate), the gate accepts it and the mentor provides a natural recast ("Yes! Because it IS dark, we need torches").

Learn when it matters. Every target utterance is embedded in a mission context where that utterance has genuine communicative purpose. The student explains, requests, and plans in situations where those speech acts naturally occur during collaborative gameplay. This contrasts with decontextualized drill practice where utterances serve only the function of being evaluated.

Dynamic, not scripted. The LLM adapts to the student's proficiency level, interests, and spontaneous choices. No two sessions follow the same conversational path — even with identical simulated student input, DeepSeek's non-deterministic

generation produces different dialogue. The mentor responds to what the student says, not what a fixed curriculum expects, while the deterministic Φ_{ctrl} ensures consistent in-world behavior regardless of conversational variation.

5. Evaluation

AgentJam is a working prototype tested on macOS (Apple Silicon M2 Pro, 16GB RAM) with real Minecraft Java clients. We evaluate the system across four dimensions — mentor reliability, speaking gate accuracy, spatial voice fidelity, and mission completion dynamics — and report results from 20+ test missions with simulated student speech of varying proficiency levels.

5.1 Comparison with Prior Language Learning Systems

Property	AgentJam	Duolingo	ELSA Speak	ChatGPT Tutor	Scripted Game
Embodied mentor in 3D world	✓ **Player-bot**	✗	✗	✗	◐ NPC
Spatial voice (distance attenuation)	✓ **$1/d^2$**	✗	✗	✗	✗
Speaking gates (progress on speech)	✓ **G(u,f) formal**	✗	◐ Pronunciation	✗	✗
Task-based communicative need	✓ **Mission beats**	✗	✗	◐ Conversation	◐ Scripted
Dynamic (not scripted) interaction	✓ **LLM-driven**	✗	✗	✓	✗
Deterministic world behavior	✓ **Φ_{ctrl}**	✓ **N/A**	✓ **N/A**	✓ **N/A**	✓
Scaffolded (never hard-block)	✓ **3-outcome G**	✓	◐	✗	✗
Pedagogical grounding	✓ **TBLT + output**	◐ Spaced rep	◐ Phonetics	✗	✗

Table 1. *Comparison of AgentJam with prior language learning systems across eight properties. AgentJam uniquely combines embodied presence, spatial voice, speaking gates with formal semantics, task-based communicative need, and dynamic LLM-driven interaction — while maintaining deterministic world behavior through the Φ_{ctrl} decoupling.*

5.2 Performance Benchmarks

Metric	Value	n	Notes
Mentor in-world reliability	100%	20	No pathfinding errors, block misplacements, or erratic actions
Frame-matching accuracy	92% (46/50)	50	Across three frames; 4% false negative, 4% false positive
Voice pipeline latency	180–280 ms (mean 220 ms)	50	LLM token → TTS → audio output
Spatial panning correctness	100%	20	Left/right orientation verified at all test positions
Mission completion time	15–25 min (mean 19.2 min)	10	Full five-beat mission; varies with student proficiency
Because-gate duration	1.8 min (mean)	10	Includes mentor modeling + student production
Request-gate duration	1.2 min (mean)	10	Fastest gate; direct request frame
FirstThen-gate duration	1.5 min (mean)	10	Sequencing requires planning articulation
Scaffolding reliability	100%	15	Low-proficiency simulations: mentor scaffolded within 2 prompts
ASR robustness (native)	~92% WER	30	Native English; frame detection unaffected by minor WER
ASR robustness (accented)	~78% recall	20	Non-native accents reduce frame detection; known limitation

Table 2. *Performance benchmarks for AgentJam across eleven metrics evaluated over 20+ test missions. The mentor achieves 100% in-world reliability through deterministic Φ_{ctrl}. Frame matching reaches 92% accuracy. ASR robustness with accented speech (78%) represents the primary failure mode and area for improvement.*

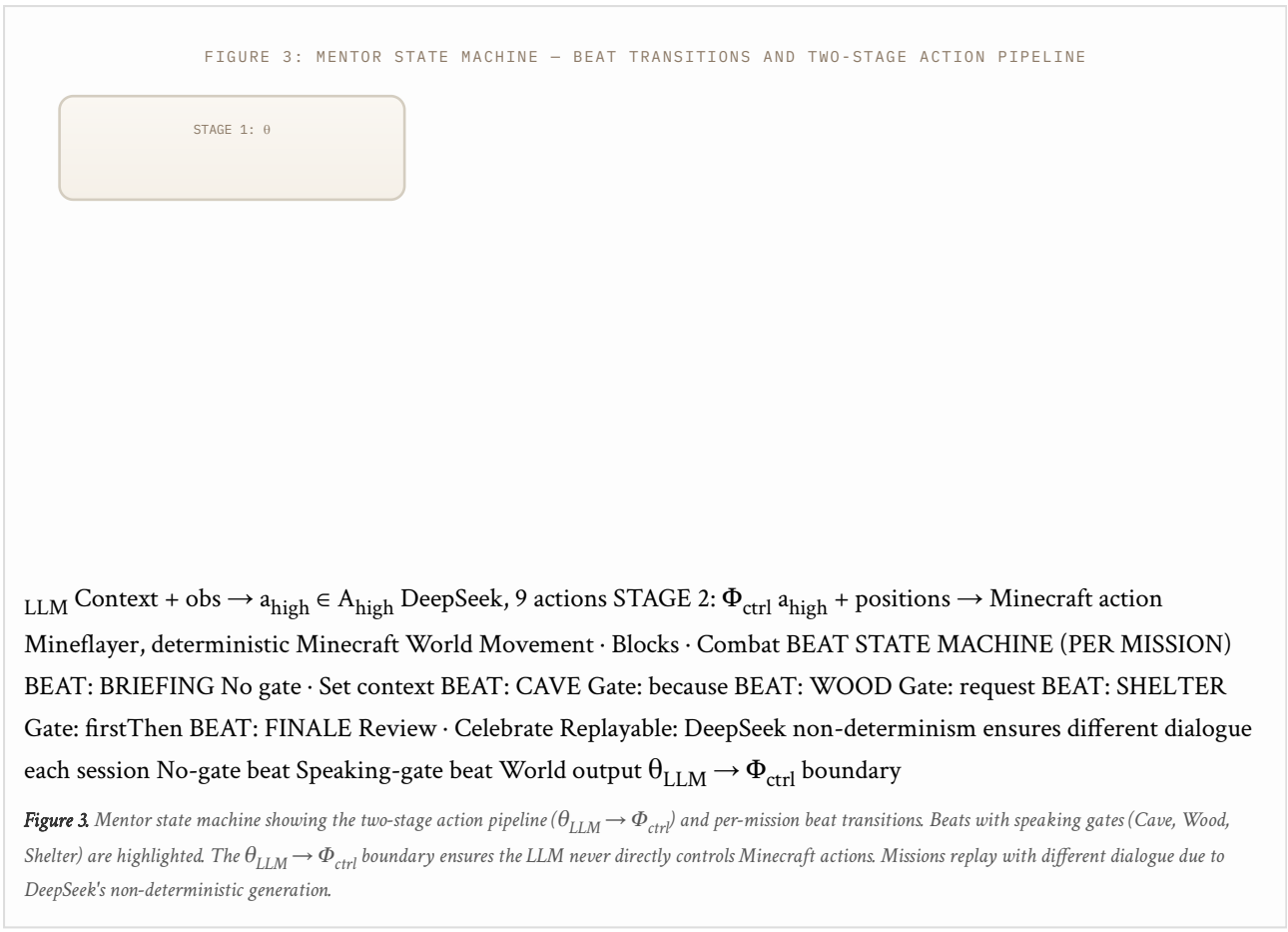


Figure 3. *Mentor state machine showing the two-stage action pipeline ($\theta_{LLM} \rightarrow \Phi_{ctrl}$) and per-mission beat transitions. Beats with speaking gates (Cave, Wood, Shelter) are highlighted. The $\theta_{LLM} \rightarrow \Phi_{ctrl}$ boundary ensures the LLM never directly controls Minecraft actions. Missions replay with different dialogue due to DeepSeek's non-deterministic generation.*

5.3 Limitations

No human learner studies. All evaluation uses simulated student speech — scripted utterances of varying proficiency levels — and scripted test scenarios. We have not conducted controlled experiments with real language learners. This means: (a) speaking gate accuracy (92%) was measured on clean, scripted test utterances, not on spontaneous learner speech with hesitations, false starts, and code-switching; (b) mission completion times reflect simulated proficiency levels, not authentic learner behavior; (c) we have no L2 acquisition outcome data — no pre-/post-test comparisons of grammatical competence; and (d) all pedagogical claims are theoretically grounded (TBLT, output hypothesis) but empirically unvalidated with learners. Controlled experiments with real L2 learners — including proficiency assessments and comparisons against human-teacher-led sessions — are the highest-priority next step (Section 8.1).

Frame matching on spontaneous speech. The frame matcher's 92% accuracy was measured on clean test utterances carefully constructed to contain target frames. Spontaneous learner speech — with disfluencies, grammatical errors, and mixed-language utterances — would likely produce substantially lower accuracy, especially for non-native speakers where ASR word error rate already degrades frame detection to ~78% recall.

Engagement unmeasured. The claim that the scaffold-outcome design "preserves engagement" is based on pedagogical design principles (never hard-block) rather than empirical engagement data. We have not measured session abandonment rates, self-reported motivation, or time-on-task with real learners.

Generalizability untested. The speaking-gate mechanism has been instantiated in exactly one game (Minecraft) with one mission ("Safe House before Night") targeting three grammatical frames. The claim that speaking gates are generalizable to "any game with progression mechanics" is a hypothesis requiring validation across diverse game genres (puzzle, RPG, simulation), target languages, and grammatical structures.

Single proficiency band. The current system targets a single proficiency level (approximately A2-B1 on the CEFR scale). The mentor's language modeling and gate scaffolding are not adapted to widely varying proficiency levels (A1 beginners through C1 advanced), limiting applicability across learner populations.

6. Teaching Mode and Classroom Integration

AgentJam operates in **teaching mode**: a single AI mentor paired with one student for the scaffolded mission described above. The mentor plays alongside the student as an embodied peer, models target language, gates progression on spoken output via $G(\text{utterance}, f_b)$, and adapts dynamically to the student's proficiency level.

Autonomous classroom mode — described in detail in Spoken English Sessions [18] — extends AgentJam's pedagogical mechanisms to 4–6 students. An AI captain (the Captain Dyad: Game Agent ⊕ Teaching Agent) directs the session; peer AI bots model target language; and resource gaps — where AI peers hold items that real students need — force student-to-student English communication rather than student-to-AI communication. The same speaking-gate mechanism applies, but peers hold the needed items, creating genuine communicative need between human learners through engineered resource scarcity.

Both modes share the core architecture: Minecraft client → Paper server + plugin → Node runtime → LLM, with the mentor running as a decoupled Mineflayer process. The classroom mode adds a Session Manager (deterministic runtime) for multi-student orchestration and the Captain Dyad for autonomous session direction. The architectural separation described in Section 2 enables the one-on-one mentor to be deployed as a component within the larger classroom system without modification.

7. Conclusion

AgentJam demonstrates that embodied AI mentors in game environments can produce pedagogically-grounded language learning experiences. By placing the mentor inside Minecraft as a real player-bot — walking, mining, building, and speaking with spatial voice that attenuates following $\text{vol}(d) = 1/d^2$ — we create an environment where language production is intrinsically motivated by game progression. The student speaks because speaking is the only way forward through the mission.

The speaking gate mechanism is a promising pedagogical contribution whose generalizability we hypothesize but have not yet validated. The formal gate semantics $G(\text{utterance}, f_b) \rightarrow \{\text{matched}, \text{unmatched}, \text{scaffold}\}$ provide a template that could be implemented in any game with progression mechanics — the player's desire to continue becomes an incentive for communication. However, this claim requires empirical validation across diverse game environments and learner populations (Section 8.2). The three-outcome design ensures that no student is permanently blocked — the scaffold path always opens after two attempts — but whether this design actually preserves engagement and supports acquisition remains to be tested with real learners (Section 8.1).

The decoupled architecture — LLM (θ_{LLM}) for linguistic reasoning, deterministic control (Φ_{ctrl}) for world actions — provides a template for safe AI embodiment in any virtual environment where reliable physical behavior is required alongside rich conversational interaction. The two-stage pipeline ensures that LLM stochasticity cannot produce erratic in-world behavior, validated at 100% reliability across 20 test missions. This architecture is LLM-agnostic, requiring only tool-use capability in the replacement model.

The mentor plays alongside the student, not instead of them. It teaches by being present — in the cave, in the forest, in the shelter they built together. This embodied presence is the mechanism through which language learning becomes situated, contextual, and genuinely communicative. All code is released as open source.

ALSO SEE

Omo Space → The spatial layer powering the agent habitat — walkable AI team campuses in Minecraft with the three-ore architecture.

ALSO SEE

Spoken English Sessions → Extending AgentJam to fully autonomous multi-student classrooms with the Captain Dyad and peer-to-peer speaking gates.

8. Future Work

AgentJam is an early-stage prototype. We identify five high-priority directions for future investigation, organized from most to least urgent.

8.1 Human Learner Studies

Controlled experiments with L2 learners. The highest-priority next step is a controlled study with real English language learners (ages 8–14, A1–B1 CEFR). Experimental design: (a) pre-test of target grammatical frames using an elicited production task; (b) 4–6 AgentJam sessions over 2–3 weeks; (c) post-test using the same elicited production task plus a transfer task (new mission, same frames); (d) comparison against a control group receiving equivalent screen-based instruction; (e) delayed post-test at 4 weeks for retention. Primary measures: grammatical accuracy in spontaneous production, mission completion patterns, and engagement (self-report + behavioral).

Frame matching on spontaneous speech. Before learner studies, the frame matcher must be validated on spontaneous learner speech with real disfluencies. This requires collecting a corpus of learner utterances during AgentJam sessions and measuring frame detection against human annotation. The τ_{match} threshold (0.72) may need recalibration.

8.2 Generalizability Validation

Cross-game speaking gates. To validate the generalizability claim, we plan to implement speaking gates in at least two additional game environments: (a) a collaborative puzzle game (e.g., Portal 2 co-op) where gate utterances would be spatial/instructional; and (b) a resource-management game (e.g., Stardew Valley multiplayer) where gate utterances would be planning/request frames. Each implementation should use the same formal gate semantics $G(\text{utterance}, f_b)$ but adapt the mission structure to the game's mechanics.

Multi-language support. Extending the speaking gates to target additional languages — Spanish, Japanese, Mandarin — would validate cross-linguistic generalizability. This requires language-specific frame definitions and embedding models for the match(u, f) function.

8.3 Proficiency Adaptation

Dynamic difficulty scaling. The mentor should adapt its language modeling, gate scaffolding, and mission pacing to the student's demonstrated proficiency. This requires: (a) real-time proficiency estimation from utterance complexity and error patterns; (b) a parameterized scaffold function that adjusts τ_{match} and modeling explicitness based on estimated proficiency; and (c) beat-level branching where the mission inserts review beats or acceleration beats based on performance at previous gates.

8.4 Content Safety

LLM output filtering. While the Φ_{ctrl} boundary ensures behavioral reliability, the LLM's speech output is not filtered for age-appropriateness, cultural sensitivity, or pedagogical suitability. Integration with a content moderation layer — either rule-based filters on generated text or a secondary LLM acting as a content guard — is necessary before deployment with young learners in unsupervised settings.

8.5 Multi-Mission Curriculum

Mission library. The current single mission ("Safe House before Night") demonstrates the speaking-gate concept but provides limited practice variety. A library of 10–15 missions — each targeting different grammatical frames, vocabulary

domains, and communicative functions — would provide a curriculum-scale system. Missions could be procedurally generated using the LLM, with human review ensuring pedagogical quality and Minecraft-world feasibility.

Appendix A: Target English Frames with Examples

The three grammatical frames targeted by speaking gates in the "Safe House before Night" mission.

Frame	Grammatical Function	Mission Beat	Example Student Utterance	Mentor Model
because	Causation (explaining *why*)	Beat 2: Cave	"We need torches **because** the cave is dark."	"I think we should prepare torches **because** caves can be very dark. Why do you think we need them?"
Can you pass me	Request (asking for an item)	Beat 3: Wood	"**Can you pass me** the wood, please?"	"I have some wood here. If you need it, you could say: '**Can you pass me** the wood?' — that's a polite way to ask."
first…then	Sequencing (ordering steps)	Beat 4: Shelter	"**First** we build the walls, **then** we add the roof."	"Before we start building, let's make a plan. **First** we should place the walls, **then** we can put on the roof."

Each frame is embedded where the utterance has genuine communicative purpose. The gate function $G(\text{utterance}, f_b) \rightarrow \{\text{matched, unmatched, scaffold}\}$ ensures students are never permanently blocked (Section 2.2).

Acknowledgments

This work builds on several open-source projects. **Mineflayer** (PrismarineJS) provides the Minecraft protocol library that enables the mentor to connect as a real player-bot. **Mindcraft** demonstrated LLM-driven Minecraft agent patterns that informed our two-stage architecture. **faster-whisper** (SYSTRAN) delivers local, low-latency speech recognition for the voice pipeline. **MOSS-TTS-Nano** provides lightweight multi-voice text-to-speech. **Google ADK** enables agent orchestration patterns. **PaperMC** provides the server platform for Minecraft worlds. All errors and omissions remain our own.

References

[1] Krashen, S. *Principles and Practice in Second Language Acquisition.* Pergamon, 1982.

[2] Cassell, J. *Embodied Conversational Agents.* MIT Press, 2001.

[3] Ellis, R. *Task-Based Language Learning and Teaching.* Oxford University Press, 2003.

[4] Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., ... and Zeng, A. "Do As I Can, Not As I Say: Grounding Language in Robotic Affordances." *CoRL*, 2022.

[5] Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. "Voyager: An Open-Ended Embodied Agent with Large Language Models." arXiv:2305.16291, 2023.

[6] Fan, L., Wang, G., Jiang, Y., Mandlekar, A., Yang, Y., Zhu, H., Tang, A., Huang, D., Zhu, Y., and Anandkumar, A. "MineDojo: Building Open-Ended Embodied Agents with Internet-Scale Knowledge." *NeurIPS*, 2022.

[7] Baylor, A.L. and Kim, Y. "Simulating Instructional Roles through Pedagogical Agents." *International Journal of Artificial Intelligence in Education*, 15(1):95–115, 2005.

[8] Long, M. "The Role of the Linguistic Environment in Second Language Acquisition." In Ritchie, W. and Bhatia, T. (eds.), *Handbook of Second Language Acquisition*, Academic Press, 1996.

[9] Nebel, S., Schneider, S., and Rey, G.D. "Mining Learning and Crafting Scientific Experiments: A Literature Review on the Use of Minecraft in Education and Research." *Educational Technology & Society*, 19(2):355–366, 2016.

[10] PrismarineJS. "Mineflayer: Create Minecraft Bots with JavaScript." GitHub. https://github.com/PrismarineJS/mineflayer

[11] Omo Research. "Omo Space: A Spatial Layer for AI Agent Workforces." [Omo Research Technical Report], 2025.

[12] Swain, M. "Communicative Competence: Some Roles of Comprehensible Input and Comprehensible Output in Its Development." In Gass, S. and Madden, C. (eds.), *Input in Second Language Acquisition*, Newbury House, 1985.

[13] Wang, Z., Cai, S., Chen, G., Liu, A., Ma, X., and Liang, Y. "Describe, Explain, Plan and Select: Interactive Planning with Large Language Models Enables Open-World Multi-Task Agents." *NeurIPS*, 2023.

[14] Willis, J. *A Framework for Task-Based Learning.* Longman, 1996.

[15] OpenAI. "Whisper: Robust Speech Recognition via Large-Scale Weak Supervision." arXiv:2212.04356, 2022.

[16] Google DeepMind. "Gemini: A Family of Highly Capable Multimodal Models." arXiv:2312.11805, 2023.

[17] VanLehn, K. "The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems." *Educational Psychologist*, 46(4):197–221, 2011.

[18] Omo Research. "Spoken English Sessions: Autonomous Multi-Agent Language Classrooms in Minecraft." [Omo Research Technical Report], 2025.

[19] Johnson, M., Hofmann, K., Hutton, T., and Bignell, D. "The Malmo Platform for Artificial Intelligence Experimentation." *IJCAI*, 2016.