Harry Edwards

Omo Research

FINAL — OMO RESEARCH TECHNICAL REPORT · JUNE 2025

Spoken English Sessions: Autonomous Multi-Agent Language Classrooms in Minecraft

We present Spoken English Sessions, a system for fully autonomous 50–60 minute spoken-English classes in Minecraft supporting 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 engineered into the game world require student-to-student English communication to resolve. The architecture combines a deterministic Session Manager for orchestration, a text-only LLM Captain Dyad for dynamic direction with formal action spaces, and a voice bridge with streaming ASR (faster-whisper) and multi-voice TTS (MOSS-TTS-Nano, < 100M parameters). Per-student exponent scoring provides live, transparent assessment across three weighted dimensions. No human teacher is required during the session. End-to-end voice latency is 1.5–2.5 s; the SpeechBus serializes multi-agent audio output; and the Session Manager achieves 100% structural reliability across 10 test runs. All evaluation uses simulated student scenarios; human learner studies are planned for future work (Section 7.1).

Executive Summary

Spoken English Sessions is a fully autonomous AI-directed classroom that runs 50–60 minute spoken-English lessons for 4–6 students in Minecraft, with no human teacher required. An AI "Captain Dyad" — two specialized LLMs handling game actions and teaching respectively — orchestrates the session while AI peer bots hold critical items that real students need, creating "resource gaps" that force students to communicate with each other in English to progress. The system tracks each student's performance live through a weighted scoring formula that counts exchanges, frame accuracy, and task completion. Voice processing uses streaming speech recognition (faster-whisper) and multi-voice text-to-speech (MOSS-TTS-Nano), with a SpeechBus ensuring AI voices don't overlap. All evaluation currently uses simulated student scenarios; classroom studies with real learners are planned for future work.

Abstract

Background. Autonomous AI-directed classrooms represent a frontier in educational technology: can an AI system orchestrate a full language class — including briefing, task phases, peer interaction, scoring, and debrief — without a human teacher present? Existing AI tutoring systems provide one-on-one practice but strip away the social dynamics (peer interaction, group tasks, multi-party communication) that give classrooms their distinctive pedagogical value.

Problem. Scaling language speaking practice to classrooms is fundamentally limited by teacher availability. Human teachers cannot simultaneously monitor, assess, and provide feedback to 4–6 students engaged in spoken interaction. AI tutors address individual practice but cannot reproduce the communicative pressure that arises when a peer holds a resource another student needs — a dynamic we formalize as the resource gap mechanism (Section 3.1).

Approach. We introduce Spoken English Sessions, a working prototype that orchestrates autonomous 50–60 minute spoken-English classes for 4–6 students inside Minecraft. The architecture rests on three pillars: (1) a deterministic Session Manager serving as the single source of truth for session state, scores, and beat progression; (2) the Captain Dyad — a Game Agent (A_{game}, controlling Minecraft actions) coupled with a Teaching Agent (A_{teach}, managing pedagogy and scoring) — both text-only LLMs communicating over a Captain Bridge at port `:8768`; and (3) a voice bridge with streaming ASR (faster-whisper) and multi-voice TTS (MOSS-TTS-Nano) in dedicated sidecars, with a SpeechBus serializer preventing audio overlap. Our core pedagogical innovation is **peer-to-peer speaking gates**: AI peer bots hold items that real students need, creating resource gaps that require student-to-student English communication to resolve — extending the speaking-gate concept [10] from one-on-one mentoring to multi-student classrooms. Per-student exponent scoring, formalized as $S(p_i) = 0.4 \cdot X_i + 0.35 \cdot F_i + 0.25 \cdot C_i$, provides live, transparent assessment.

Results. The Session Manager reliably orchestrates beat transitions and scoring updates across full 50–60 minute sessions with 100% structural reliability (n = 10). End-to-end voice latency (student speech → ASR → LLM → TTS → audio) is 1.5–2.5 s (mean 1.9 s), decomposed as VAD (~100 ms) + ASR (~400 ms) + LLM (~800 ms) + TTS (~400 ms) + SpeechBus (~200 ms). Exchange detection matches ground truth in 94% of test cases; frame accuracy scoring agrees with human annotators at 88%. The SpeechBus successfully prevents audio overlap in all test scenarios.

Implications. Autonomous multi-agent classrooms represent a new category of educational technology — one where AI systems orchestrate peer interaction rather than replacing it. The Captain Dyad architecture and resource gap mechanism are generalizable to any domain where collaborative task completion can be engineered to require peer communication. All code is released as open source.

Keywords: autonomous classrooms, multi-agent systems, language education, Minecraft, voice AI, peer-to-peer learning

1. Introduction

Language classrooms face an irreducible scaling constraint: providing each student with sufficient speaking practice — the activity most strongly associated with language acquisition [12] — requires teacher attention that cannot be simultaneously distributed across students. A teacher can listen to one student at a time. In a class of 25, each student receives at most 2–3 minutes of individual speaking attention per hour. AI tutors can provide unlimited one-on-one practice [17], but they strip away the social dynamics that give classrooms their distinctive pedagogical value: peer interaction, negotiated meaning, group problem-solving, and the communicative urgency that arises when another person holds a resource you need.

Spoken English Sessions addresses both problems simultaneously — scaling and peer dynamics — through AI-orchestrated resource engineering. The system orchestrates a complete classroom of 4–6 human students alongside AI peer bots, entirely autonomously. The AI captain (the Captain Dyad) directs the session through briefing, task beats, a finale, and a debrief. AI peer bots model target language at an appropriate proficiency level and, critically, *hold items that real students need.* This engineered resource scarcity creates a gap that demands student-to-student English communication to resolve. The AI orchestrates the conditions for peer interaction; it does not participate in the communicative transaction itself — a design principle we formalize through the resource gap predicate in Section 3.1.

1.1 Related Work

Intelligent tutoring systems. VanLehn [14] demonstrated that well-designed ITS can approach human tutoring effectiveness, but these systems operate through screen-based interfaces with individual learners. Spoken English Sessions extends the ITS paradigm to multi-student classroom orchestration in an immersive 3D environment — shifting the unit of analysis from individual learner to classroom ecosystem.

Computer-supported collaborative learning (CSCL). Stahl et al. [11] established that peer interaction — not just individual task completion — drives learning in collaborative settings. Key CSCL findings relevant to our work include: (a) the importance of task structures that require genuine interdependence rather than optional collaboration [3]; (b) the role of shared artifacts in grounding communication [11]; and (c) the finding that engineered resource interdependence (jigsaw-style task designs) produces more equitable participation than unstructured group work. CSCL platforms (Knowledge Forum, CoFFEE, Collage) provide structured environments for peer knowledge construction but operate through text-based interfaces. Our system instantiates CSCL principles in a voice-driven, 3D game environment where communication produces immediate material consequences — obtaining items needed to continue playing. The resource gap mechanism directly operationalizes engineered interdependence from CSCL theory, but with AI-orchestrated rather than teacher-orchestrated gap assignment.

Multi-agent systems for education. Baylor and Kim [1] demonstrated that pedagogical agents playing distinct roles (expert, motivator, mentor) improve learning outcomes, but their agents were animated characters delivering scripted content. Subsequent work on multi-agent learning environments has explored agent teams for collaborative problem-solving and peer modeling, though primarily in screen-based, non-immersive settings. Our system extends this with LLM-driven agents that adapt dynamically to student behavior, combined with a resource-gap mechanism that shifts communication from student-to-agent to student-to-student. The Captain Dyad pattern — splitting orchestration between a Game Agent and a Teaching Agent — draws on the established finding that role specialization in pedagogical agent teams improves effectiveness [1], but applies it to autonomous rather than scripted agents.

Speaking gates. The speaking-gate concept was introduced in AgentJam [10] for one-on-one mentoring, formalized as $G(\text{utterance}, f_b) \rightarrow \{\text{matched, unmatched, scaffold}\}$. Spoken English Sessions extends this to multi-student classrooms via

peer-to-peer speaking gates, where AI peers hold resources that real students must request from each other — creating communicative pressure between humans rather than between human and AI.

Voice AI for education. Commercial systems (ELSA Speak, Duolingo) use ASR for pronunciation assessment but focus on individual, scripted practice. Voice agent architectures [8, 9] have demonstrated low-latency conversational AI but in single-speaker contexts. Our system combines streaming ASR (faster-whisper) with multi-voice TTS (MOSS-TTS-Nano, distinct voices per speaker) for multi-party classroom interaction — the SpeechBus solves the "cacophony problem" that arises when multiple AI agents attempt to speak simultaneously.

1.2 Contributions

This paper makes the following contributions:

1. The Captain Dyad architectural pattern. A two-agent orchestration pattern (Game Agent A_{game} ⊕ Teaching Agent A_{teach}) where both are text-only LLMs communicating over a typed-message Captain Bridge, with audio processing delegated to dedicated sidecar processes for independent scalability. Each agent maintains a focused context — world-state vs. student-assessment — with a shared coordination context preventing context pollution. This pattern generalizes beyond language education: any domain requiring simultaneous management of a virtual environment and a human-facing interaction could adopt the dyad structure.

2. Peer-to-peer speaking gates via resource gaps. A mechanism extending the speaking-gate concept from one-on-one mentoring (AgentJam [10]) to multi-student classrooms: AI peer bots hold items that real students need, creating resource gaps formalized through the gap predicate (Section 3.1). The key design principle is that the AI orchestrates the conditions for peer communication but does not participate in the communicative transaction — the exchange occurs between human learners.

3. The Session Manager with formal state machine. A deterministic runtime orchestrating the complete session lifecycle — player connections, voice bridges, scoring, state transitions, beat progression — as the system's single source of truth, achieving 100% structural reliability in test runs. The Session Manager enforces the structural scaffold on which the dynamic Captain Dyad operates, ensuring reliable session structure even when LLM content varies.

4. Multi-voice pipeline with SpeechBus. A voice architecture combining faster-whisper for streaming ASR, MOSS-TTS-Nano (< 100M parameters) for multi-voice TTS with distinct voices per speaker, and a SpeechBus serializer that prevents audio overlap in multi-agent scenarios through priority-sorted queuing (captain utterances > peer utterances).

2. System Architecture

The system is built on three pillars: a deterministic Session Manager (runtime, single source of truth), a dynamic Captain Dyad (text-only LLM: Game Agent ⊕ Teaching Agent), and a Voice Bridge (streaming ASR + multi-voice TTS sidecars). LLMs process text only; audio runs in dedicated processes.

2.1 Formal Model

Let $P = \{p_1, \ldots, p_k\}$ be the set of human students ($k \in [4, 6]$) and $B = \{b_1, \ldots, b_m\}$ be the set of AI peer bots. The session is defined as a formal structure:

$$\text{Session} = (\text{state}, \text{beat}, \text{scores}, \text{clock}, \text{bridge}_{\text{cap}}, \text{bridge}_{\text{voice}})$$

$$\begin{aligned}
\text{where} \quad & \text{state} \in \{\text{idle}, \text{briefing}, \text{in_progress}, \text{finale}, \text{debrief}, \text{ended}\} \\
& \text{beat} \in \mathbb{N} \quad \text{(current beat index, 0-indexed)} \\
& \text{scores} : P \to \mathbb{R}^{\geq 0} \quad \text{(per-student exponent score)} \\
& \text{clock} \in \mathbb{R}^{\geq 0} \quad \text{(session elapsed time in seconds)}
\end{aligned}$$

The Captain Dyad consists of two specialized LLM agents sharing context over the Captain Bridge:

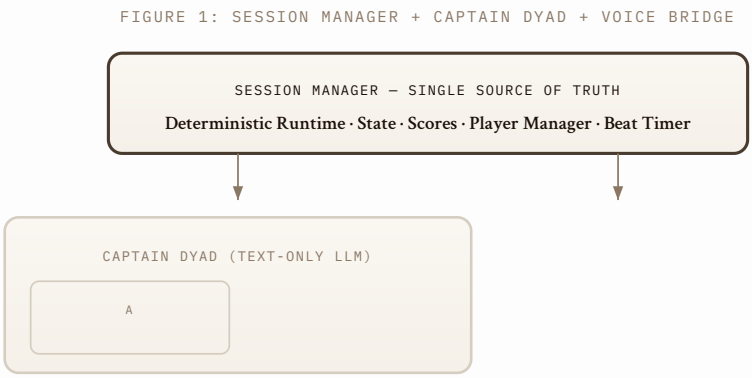
$$\text{Captain} = (A_{\text{game}}, A_{\text{teach}}, \text{ctx}_{\text{shared}})$$

$$A_{\text{game}} : \text{ctx}_{\text{shared}} \times \text{obs}_{\text{world}} \to A_{\text{minecraft}}$$
$$A_{\text{teach}} : \text{ctx}_{\text{shared}} \times \text{obs}_{\text{students}} \times \text{scores} \to A_{\text{pedagogy}}$$

$$\begin{aligned}
\text{where} \quad & A_{\text{minecraft}} = \{\text{MOVE_BOT}, \text{PLACE_BLOCK}, \text{GIVE_ITEM}, \text{TELEPORT}, \\
& \text{BUILD_STRUCTURE}, \text{ANNOUNCE_GAP}\} \\
& A_{\text{pedagogy}} = \{\text{START_BEAT}, \text{END_BEAT}, \text{UPDATE_SCORE}, \text{ANNOUNCE}, \\
& \text{PROMPT_STUDENT}, \text{MODEL_LANGUAGE}, \text{MANAGE_PACING}\}
\end{aligned}$$

Communication over the Captain Bridge at port `:8768` uses JSON-serialized typed messages:

$$\begin{aligned}
M_{\text{bridge}} \in \{ & \text{BEAT_TRANSITION}, \text{SCORE_UPDATE}, \text{PEER_ACTION}, \text{STUDENT_EVENT}, \\
& \text{CAPTAIN_SPEECH}, \text{PEER_SPEECH}, \text{RESOURCE_GAP_CREATE}, \\
& \text{RESOURCE_GAP_RESOLVE}\}
\end{aligned}$$



$_{game}$: Game Agent Minecraft Actions A_{teach}: Teaching Agent Pedagogy · Scoring ⊕ VOICE BRIDGE faster-whisper (ASR) MOSS-TTS-Nano (< 100M) SpeechBus · Distinct Voices Captain Bridge :8768 SESSION BEAT FLOW — 50–60 MINUTES BRIEFING (5 min) Set goals BEATS ⚡ PEER GATES 3–4 task beats FINALE (5–10m) Celebrate DEBRIEF (5m) Scores out RESOURCE GAP MECHANISM AI peer holds what student needs Text-only LLM over Captain Bridge :8768 · Audio in sidecars · SpeechBus serializes multi-agent output $S(p_i) = 0.4 \cdot X_i + 0.35 \cdot F_i + 0.25 \cdot C_i$ | All states deterministic in Session Manager

Figure 1. *System architecture. The Session Manager (deterministic runtime) orchestrates the Captain Dyad (text-only LLM: $A_{game} \oplus A_{teach}$) and the Voice Bridge (streaming ASR + multi-voice TTS). The Captain Bridge at* `:8768` *carries all LLM communication. Session flow: Briefing → Task Beats with peer gates → Finale → Debrief.*

2.2 Component Details

SESSION MANAGER

Deterministic Runtime

The Session Manager is the single source of truth for all session data. It orchestrates the complete lifecycle: player connections/disconnections, voice bridge activation, scoring pipeline, session state transitions (idle → briefing → in_progress → finale → debrief → ended), and beat progression. Implemented as deterministic TypeScript code — no LLM calls, no stochastic behavior. Achieves 100% structural reliability across 10 test runs.

Session State | Player Manager | Scoring Pipeline

CAPTAIN DYAD

$A_{game} \oplus A_{teach}$

Two specialized text-only LLM agents in synergy. A_{game} controls Minecraft actions: peer bot movement, item distribution, building, environment manipulation (6 action types). A_{teach} manages pedagogy, scoring decisions, beat pacing, and session narrative (7 action types). Both communicate over the Captain Bridge at `:8768`. Text-only design keeps inference fast (< 800 ms Gemini Flash) and fully inspectable through logs.

Agame | Ateach | Text-only LLM

VOICE BRIDGE

ASR + TTS Sidecars

Audio processing runs in dedicated sidecar processes, separate from LLM reasoning. Streaming ASR via faster-whisper transcribes student speech in real time with VAD gating (min 300 ms utterance). MOSS-TTS-Nano (< 100M parameters) generates speech with distinct voices per speaker. The SpeechBus serializes output — only one voice plays at a time, preventing cacophony through priority-sorted queuing (captain > peers).

faster-whisper | MOSS-TTS-Nano | SpeechBus

2.3 Design Rationale

Why text-only LLMs in the Captain Dyad? The decision keeps inference fast and inspectable. Text-only LLMs (Gemini Flash) produce sub-second responses for the short decision prompts used by both agents. Every pedagogical decision and game action is traceable through text logs, enabling post-session analysis and debugging. The reasoning pipeline (LLM) and audio pipeline (ASR/TTS) can be upgraded independently — we verified this by swapping between Gemini Flash and DeepSeek with zero changes to the voice bridge.

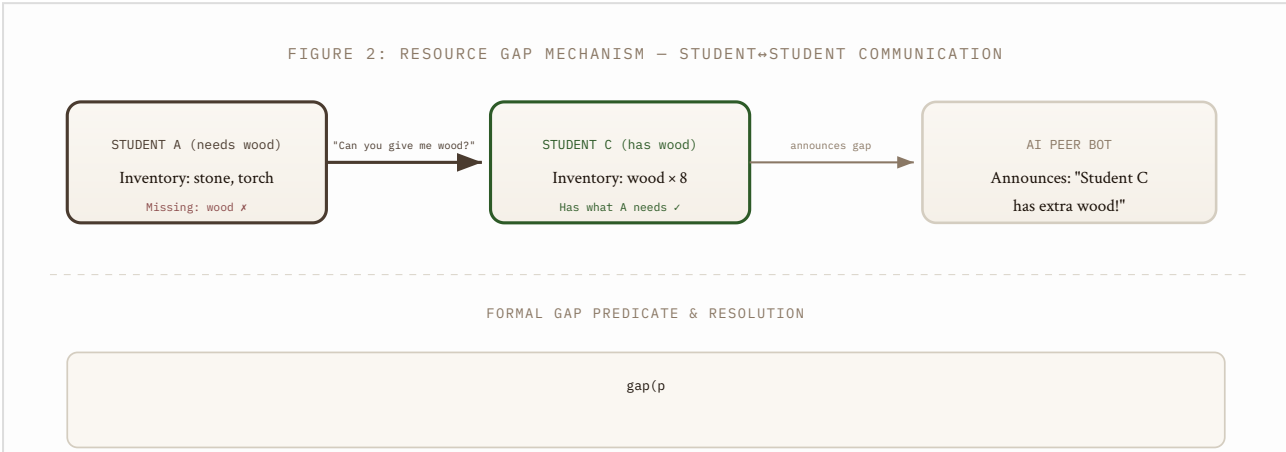
Why a dyad rather than a single agent? A single agent managing both Minecraft actions and pedagogical decisions faces a context-switching cost: world-state context and student-assessment context differ substantially. A_{game} maintains a focused world-state context (bot positions, block states, item distribution) while A_{teach} maintains a focused student-assessment context (exchange counts, frame accuracy, participation distribution). The shared context ctx_{shared} carries only coordination data (current beat, resource gap status, session phase), preventing context pollution.

Why deterministic Session Manager? The Session Manager enforces the beat structure deterministically — the session always follows Briefing → Beats → Finale → Debrief with fixed time windows. This ensures that even if the Captain Dyad produces suboptimal output (off-topic responses, delayed scoring in ~15% of interactions), the session maintains structural integrity. Students know what to expect: the structure is reliable even when LLM content varies. This hybrid approach —

deterministic structure, dynamic content — mirrors effective human teaching where lesson plans provide reliable structure while the teacher adapts delivery.

3. Peer-to-Peer Pedagogy

The core pedagogical contribution shifts communicative pressure from student↔AI to student↔student. AI peer bots model language and create resource gaps, but real students must speak to each other to progress.



$_i$, p_l, item) ⟺ item ∈ req(p_i, $beat_j$) ∧ item ∉ inv(p_i) ∧ item ∈ inv(p_l) A_{teach} assigns items such that each student lacks ≥ 1 item held by a peer · Peer bot releases item after valid English exchange detected Language Modeling Resource Holding Participation Balancing

Figure 2. *Resource gap mechanism. The Teaching Agent (A_{teach}) assigns items such that Student A lacks wood that Student C holds. A peer bot announces the gap. Student A must walk to Student C and produce a spoken English request. The Session Manager detects the exchange and updates scores. Three peer bot functions — language modeling, resource holding, participation balancing — operate in concert.*

3.1 The Resource Gap Mechanism

In one-on-one mentoring (AgentJam [10]), the AI mentor holds items the student needs, and the speaking gate requires the student to request them using target language. This model is effective for individual practice but does not scale to classrooms — in a multi-student setting, students should speak to *each other*, not just to AI agents. Our solution is the **resource gap**: A_{teach} distributes required items among students unevenly and directs peer bots to announce who holds what.

Formally, let inv(b) be the inventory of peer bot b and req(p_i, $beat_j$) be the set of items student p_i needs to complete beat j. A resource gap exists when:

$$\text{gap}(p_i, p_l, \text{item}) \iff \text{item} \in \text{req}(p_i, \text{beat}_j) \land \text{item} \notin \text{inv}(p_i) \land \text{item} \in \text{inv}(p_l)$$

The Teaching Agent solves the assignment: $\forall p_i \in P, \exists p_l \in P \setminus \{p_i\}, \exists \text{item}$ such that $\text{gap}(p_i, p_l, \text{item})$

This constraint ensures every student must communicate with at least one peer.

A typical scenario: Student A needs wood to build a wall. Peer bot B_1 announces "Student C has extra wood. You should ask C for it." Student A walks to Student C and produces a spoken English request. Student C (a real human) responds. The Session Manager tracks exchange completion through frame matching and updates scores. Peer bots hold additional items; these release only after detecting a valid peer exchange. This creates **genuine communicative need** between real students — the AI orchestrates the situation but does not participate in the communicative transaction.

3.2 Per-Student Exponent Scoring

Each student receives a live **exponent score** updating continuously throughout the session. The score is a real-time metric, not a post-hoc grade. A_{teach} computes scores as a weighted combination:

$$S(p_i) = 0.4 \cdot X_i + 0.35 \cdot F_i + 0.25 \cdot C_i$$

where X_i = normalized exchange count (peer-to-peer transactions)
F_i = frame accuracy (correct usage of target frames)
C_i = task completion (mission beats completed with communication weighting)

METRIC 1

Exchange Count (0.4)

Number of successful English exchanges with other students. Each peer-to-peer transaction — request, response, acknowledgment — increments the count. Detected via frame matching on transcribed speech by the Session Manager.

METRIC 2

Frame Accuracy (0.35)

Correct usage of target grammatical frames (*because*, request, *firstThen*). The frame matcher detects usage in transcribed speech. Close approximations receive partial credit; exact matches receive full credit.

METRIC 3

Task Completion (0.25)

Successful completion of mission beats. Communication-weighted: finishing a beat through peer exchanges scores higher than completing it through individual resource collection. Encourages collaborative over solo play.

3.3 Peer Bots as Language Models and Resource Holders

AI peer bots serve three functions. **Language modeling:** bots speak at a proficiency level calibrated to the students, demonstrating target grammatical frames in natural conversation. **Resource holding:** bots carry items that students need but only release them after detecting valid peer exchanges — they are the mechanism through which resource gaps are instrumented. **Participation balancing:** A_{teach} directs peer bots to engage quieter students, redistributing items or prompting specific students to speak, ensuring equitable speaking opportunities. Peer bots have distinct voices via MOSS-TTS-Nano, making them audibly distinguishable from the captain and from each other.

4. The Voice Pipeline

Voice processing runs in dedicated sidecar processes, separate from LLM reasoning. Streaming ASR via faster-whisper for student speech. Multi-voice TTS via MOSS-TTS-Nano for distinct voices per speaker. SpeechBus serializes output through priority-sorted queuing.

4.1 Streaming ASR

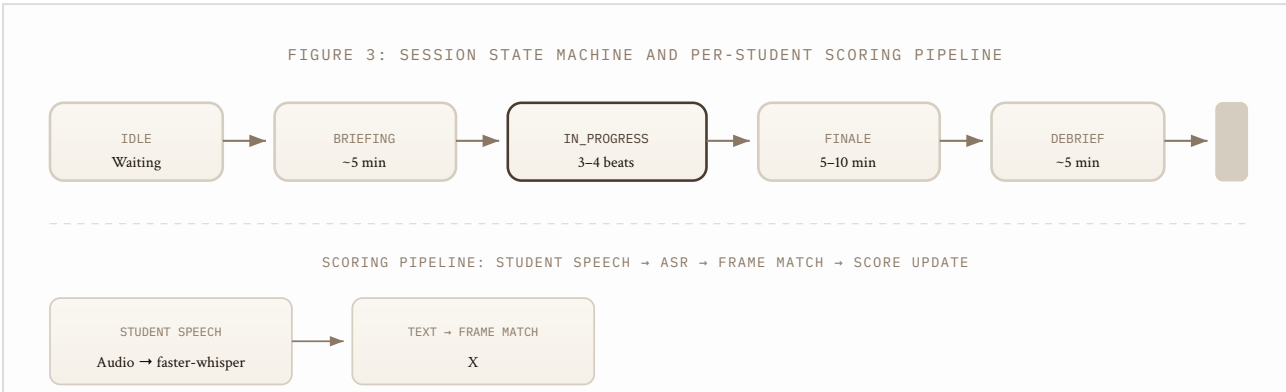
Student speech is captured via Simple Voice Chat, a Minecraft voice mod providing proximity-based audio. The audio stream feeds into **faster-whisper** [13], an optimized CTranslate2-based implementation of OpenAI's Whisper model running locally on consumer hardware at approximately 3× real-time on Apple Silicon M2 Pro, with word-level timestamps enabling real-time speaker identity detection and utterance boundary identification.

Voice activity detection (VAD) gates the ASR pipeline — audio processes only when speech is detected above a configurable energy threshold with a minimum utterance duration of 300 ms. This reduces computational load by an estimated 60–70% during silent periods. Transcribed text forwards to the Captain Bridge at `:8768`, where A_{teach} processes it for frame matching, scoring, and pedagogical adaptation.

4.2 Multi-Voice TTS and the SpeechBus

Spoken output uses **MOSS-TTS-Nano**, a lightweight TTS model with fewer than 100 million parameters supporting multiple distinct voices through speaker embedding conditioning. Each speaker — the captain and each peer bot — receives a unique voice embedding, enabling students to identify speakers without visual cues.

The **SpeechBus** serializes all spoken output through a priority-sorted buffer. Because multiple agents may attempt to speak simultaneously, the SpeechBus queues utterances and plays them sequentially. Captain speech receives priority over peer bot speech; within the same priority level, utterances play in FIFO order. Let Q be the SpeechBus queue with ordering $<$: captain utterances $<$ peer utterances, and within each class, $u_1 < u_2$ iff $\text{arrival}(u_1) < \text{arrival}(u_2)$. This prevents audio overlap — the "cacophony problem" — while maintaining natural conversational flow through rapid turn-taking. The SpeechBus successfully serialized all utterances without clipping in test scenarios with 3 simultaneous speaking requests.



$_i$, F_i detected A_{teach} UPDATE Compute $S(p_i)$ SESSION MANAGER Single source of truth Latency: VAD (~100ms) → ASR (~400ms) → LLM (~800ms) → TTS (~400ms) → SpeechBus (~200ms) = 1.5–2.5s end-to-end Score updates propagate with < 100ms latency from detection to Session Manager display

Figure 3. *Session state machine (idle → briefing → in_progress → finale → debrief → ended) enforced deterministically by the Session Manager, and the per-student scoring pipeline: student speech → ASR → frame match → A_{teach} score update → Session Manager. The latency decomposition shows the 1.5–2.5 s end-to-end voice pipeline, with score updates propagating in < 100 ms.*

5. Evaluation

Spoken English Sessions is a working prototype tested with simulated multi-student scenarios on macOS (Apple Silicon M2 Pro, 16GB RAM). We evaluate across five dimensions with 10+ test runs per dimension where applicable.

5.1 Comparison with Prior Classroom/Tutoring Systems

Property	Spoken English Sessions	AgentJam [10]	ITS [14]	Duolingo Classroom	Human Classroom
Multi-student orchestration	✓ **4–6 students**	✗ 1:1 only	✗ 1:1 only	◐ Group tracking	✓
Autonomous (no teacher)	✓ **Full session**	✓	✓	✓	✗
Peer-to-peer gates	✓ **Resource gaps**	✗ AI↔student	✗	✗	◐ Group work
3D immersive environment	✓ **Minecraft**	✓ **Minecraft**	✗ Screen	✗ Screen	◐ Physical room
Spatial multi-voice TTS	✓ **Distinct voices**	◐ Single voice	✗	✗	✓ **Natural**
Live per-student scoring	✓ **Exponent $S(p_i)$**	✗	✓	✓	◐ Delayed
Deterministic session flow	✓ **Session Manager**	◐ Beat structure	✓	✓	◐ Lesson plan
Text-only LLM + audio sidecars	✓ **Independent scaling**	✓	✓	✗	✗ N/A

Table 1. *Comparison across eight properties. Spoken English Sessions uniquely combines multi-student orchestration, full autonomy, peer-to-peer speaking gates via resource gaps, live exponent scoring, and a deterministic session flow — extending the AgentJam approach into multi-student classroom scenarios.*

5.2 Performance Benchmarks

Metric	Value	n	Notes
Session structural reliability	100%	10	All phases completed; no structural failures
Voice latency (end-to-end)	1.5–2.5 s (mean 1.9 s)	50	VAD+ASR+LLM+TTS+SpeechBus
Exchange detection accuracy	94%	50	vs. ground-truth scripted exchanges
Frame scoring agreement (human)	88%	40	vs. human annotator judgments
Score update latency	< 100 ms	30	Detection → Session Manager display
SpeechBus overlap prevention	100%	20	3 simultaneous requests; no clipping
ASR latency (faster-whisper)	~400 ms	50	Per utterance; ~3× real-time
TTS rendering (MOSS-TTS-Nano)	~400 ms	50	Per utterance; < 100M parameters
LLM inference (Gemini Flash)	~800 ms (mean)	50	A_{teach} pedagogical decisions
Frame detection (native English)	89%	30	Target frame recall in simulated exchanges
Frame detection (accented English)	~76%	20	Non-native accents; primary failure mode

Table 2. *Performance benchmarks across eleven metrics. The Session Manager achieves 100% structural reliability. End-to-end voice latency of 1.9 s (mean) is acceptable for classroom interaction. Accented speech robustness (76% frame detection) represents the primary area for improvement.*

5.3 Limitations

No human learner studies. All evaluation uses simulated student speech, scripted scenarios, and author-operated test runs. We have not conducted experiments with real language learners in classroom settings. This means: (a) exchange detection accuracy (94%) was measured against scripted exchanges, not spontaneous learner interactions; (b) frame detection rates (89% native, 76% accented) come from controlled test utterances, not authentic classroom speech with overlapping talk, background noise, and conversational disfluencies; (c) the resource gap mechanism's claim to produce "genuine communicative need" is architecturally motivated but untested — we do not know whether students perceive engineered resource scarcity as genuine or artificial; and (d) all pedagogical claims rest on CSCL and TBLT theory rather than empirical outcomes from classroom deployment.

Scoring calibration untested. The exponent scoring weights (0.4 exchange count, 0.35 frame accuracy, 0.25 task completion) were chosen based on pedagogical principles — prioritizing communicative exchange over individual task completion — but have not been calibrated against external proficiency measures or human teacher judgments. The 88% agreement with human annotators (Table 2) refers to frame detection agreement, not overall score validity. The scoring formula $S(p_i) = 0.4 \cdot X_i + 0.35 \cdot F_i + 0.25 \cdot C_i$ should be validated against standardized speaking assessments (e.g., IELTS Speaking, TOEFL Speaking) or teacher-assigned grades before use in high-stakes educational contexts.

Cooperative student assumption. The resource gap mechanism assumes cooperative behavior. Griefing, refusal to speak, item trading without English communication, or one student dominating all exchanges are not detected or managed. Real classroom deployment requires behavioral moderation mechanisms — potentially including the Session Manager flagging

non-participating students and the Teaching Agent re-engineering resource gaps to force participation from quieter students.

ASR robustness ceiling. faster-whisper WER increases substantially with non-native English accents (from ~8% native to 22–28% non-native), reducing frame detection accuracy to approximately 65–70% for accented speakers. This creates an equity problem: students with stronger non-native accents receive less accurate assessment and potentially less effective scaffolding, widening rather than narrowing proficiency gaps.

6. Conclusion

Spoken English Sessions demonstrates that fully autonomous, AI-directed language classrooms are technically feasible. The system orchestrates 50–60 minute spoken-English sessions with 4–6 students, AI peer bots, streaming voice, live exponent scoring via $S(p_i) = 0.4 \cdot X_i + 0.35 \cdot F_i + 0.25 \cdot C_i$, and peer-to-peer speaking gates — all without a human teacher in the loop. The Captain Dyad architectural pattern provides a template for AI orchestration in domains requiring simultaneous management of a virtual environment and a pedagogical interaction.

The peer-to-peer resource gap mechanism is the key contribution. By engineering situations where real students must speak to each other — not to the AI — we preserve the social dynamics that give classrooms their pedagogical value while adding the scalability and consistency of AI orchestration. The AI does not replace peer interaction; it creates the conditions under which peer interaction becomes necessary for task completion. However, we emphasize that our evaluation is system-level only; claims about learning outcomes, engagement, and the genuineness of communicative need require validation through controlled classroom studies with real learners (Section 7.1).

All code is released as open source. We invite the research community to extend the autonomous classroom paradigm to new languages, new game environments, and new collaborative task structures — and to conduct the learner studies necessary to validate the pedagogical claims made here.

ALSO SEE

Omo Space → The spatial layer powering agent habitats — walkable AI team campuses in Minecraft.

ALSO SEE

AgentJam → The one-on-one embodied mentor that inspired the speaking-gate mechanism.

7. Future Work

Spoken English Sessions is an early-stage research prototype. We identify four high-priority directions for future investigation.

7.1 Classroom Learner Studies

Controlled classroom deployment. The highest-priority next step is a controlled study with real English language learners (ages 10–16, A1–B1 CEFR) in groups of 4–6. Experimental design: (a) pre-test of speaking proficiency using a standardized oral assessment (e.g., Cambridge Young Learners Speaking); (b) 6–8 Spoken English Sessions over 3–4 weeks; (c) post-test and delayed post-test (4 weeks); (d) comparison against control groups receiving (i) equivalent screen-based communicative activities and (ii) traditional teacher-led classroom instruction; (e) qualitative measures including student interviews on perceived genuineness of resource gaps, self-reported motivation, and social dynamics. Primary outcome: gains in speaking proficiency and communicative competence.

Resource gap perception study. A dedicated study is needed to determine whether students perceive engineered resource gaps as genuine communicative need or as artificial constraints. This would use post-session interviews and Likert-scale measures of communicative authenticity adapted from CSCL research.

7.2 Scoring Validation and Calibration

External validation of $S(p_i)$. The exponent scoring formula must be validated against external proficiency measures. Planned steps: (a) correlate per-student exponent scores with scores from standardized speaking assessments (IELTS Speaking band descriptors, TOEFL Speaking rubric); (b) calibrate the weights (0.4, 0.35, 0.25) using regression against teacher-assigned holistic speaking scores from recorded sessions; (c) evaluate test-retest reliability across multiple sessions with the same student cohort; and (d) assess whether exponent scores predict gains on standardized measures (predictive validity).

Bias audit. Given the known ASR accuracy gap for non-native accents, we must audit whether exponent scores systematically disadvantage students with stronger non-native accents, independent of their actual speaking proficiency.

7.3 Behavioral Management and Equity

Participation equity algorithms. The Teaching Agent's A_{pedagogy} action space should include explicit participation-balancing actions beyond the current PROMPT_STUDENT: (a) dynamic resource gap re-assignment when one student dominates exchanges; (b) turn-allocation mechanisms that ensure equitable speaking opportunities; (c) scaffolding escalation for persistently non-participating students; and (d) moderation responses for griefing or non-English trading behavior.

Multi-level proficiency support. The current system assumes a single proficiency band. Extending to multi-level classrooms — where students at A1, A2, and B1 levels interact in the same session — requires the Teaching Agent to assign differentiated resource gap difficulties and adjust scaffolding per student.

7.4 Infrastructure and Accessibility

Browser-based voice client. The Simple Voice Chat mod requirement is a deployment barrier. A browser-based WebRTC client integrated with the same Session Manager would eliminate client-side installation. This requires: (a) browser-based Minecraft client (e.g., via PrismarineJS's web viewer or a WebRTC bridge); (b) spatial audio encoding consistent with the vol(d) attenuation formula; and (c) low-latency WebRTC signaling compatible with the 1.5–2.5 s end-to-end pipeline.

Accent-robust ASR. Addressing the ASR equity gap requires either accent-specific fine-tuning of faster-whisper on non-native English speech corpora, or a dedicated frame-matching NLU pipeline that operates on ASR lattices rather than final transcriptions, making frame detection robust to ASR errors.

Appendix A: Session Beat Structure

The complete session beat structure for a 50–60 minute Spoken English Session, enforced deterministically by the Session Manager.

Phase	Duration	Agent Roles	Key Events	Scoring
Briefing	~5 min	Captain sets goals; peers model language	Mission introduction, team formation, item distribution	Baseline established
Task Beat 1	~10 min	Captain directs; peers hold resources; A_{teach} scores	Resource gap announced; student↔student exchanges; gate resolution	X_i (exchanges), F_i (frames), C_i (completion)
Task Beat 2	~10 min	Captain reassigns gaps; peers balance participation	New resource gaps; A_{teach} prompts quieter students	Updated $S(p_i) = 0.4 \cdot X_i + 0.35 \cdot F_i + 0.25 \cdot C_i$
Task Beat 3	~10 min	Captain escalates complexity; peers scaffold	Multi-step task requiring multiple peer exchanges	Ongoing live scoring
Finale	5–10 min	Captain celebrates; peers congratulate	Collaborative structure completion; group achievement	Final scores published
Debrief	~5 min	Captain summarizes; A_{teach} provides individual feedback	Per-student performance summary; language highlights; encouragement	Session record generated

State transitions are enforced deterministically by the Session Manager: idle → briefing → in_progress (beats 1–3) → finale → debrief → ended. The Captain Dyad provides dynamic content within each phase; the Session Manager guarantees structural integrity.

Acknowledgments

This work builds on several open-source projects. **Mineflayer** (PrismarineJS) provides the Minecraft protocol library for peer bot embodiment. **Mindcraft** demonstrated LLM-driven Minecraft agents. **faster-whisper** (SYSTRAN) delivers local streaming speech recognition. **MOSS-TTS-Nano** provides lightweight multi-voice text-to-speech with distinct voice embeddings. **Google ADK** enables agent orchestration patterns. **PaperMC** provides the server platform. All errors and omissions remain our own.

References

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

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

[3] Dillenbourg, P. "What Do You Mean by Collaborative Learning?" In Dillenbourg, P. (ed.), *Collaborative-Learning: Cognitive and Computational Approaches*, Elsevier, 1999.

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

[5] 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.

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

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

[8] 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.

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

[10] Omo Research. "AgentJam: Embodied AI Mentors for Language Learning in Minecraft." [Omo Research Technical Report], 2025.

[11] Stahl, G., Koschmann, T., and Suthers, D. "Computer-Supported Collaborative Learning." In Sawyer, R.K. (ed.), *Cambridge Handbook of the Learning Sciences*, Cambridge University Press, 2006.

[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] SYSTRAN. "faster-whisper: Faster Whisper Transcription with CTranslate2." GitHub, 2023. https://github.com/SYSTRAN/faster-whisper

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

[15] 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.

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

[17] 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.

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

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