MI-World: Multi-Inhabitant World Models with 3D Evolution

Anonymous authors
Paper under double-blind review



Abstract

Interactive world models simulate future environments in response to user actions. Current approaches rely primarily on video generation, where visual impacts often appear correct while the underlying physical state remains unchanged. This disconnect becomes disruptive when multiple characters share the scene. A removed block can reappear once a camera turns away, character health can remain static despite visible hits, and off-screen non-player characters leave no lasting traces for players who arrive later. We present MI-World, an interactive world model that couples multi-character simulation with an evolving 3D voxel map and structured entity records. Scene completion synthesizes unobserved terrain without overwriting established coordinates, preserving earlier structures during exploration. Player and NPC controls update voxel occupancy and character attributes through a shared transition model, enabling autonomous NPCs to modify the environment even while outside any active viewpoint. Perspective video streams are then rendered directly from the updated canonical state. We collect a dataset of 45,572 synchronized multi-character trajectories annotated with explicit interaction events. MI-World achieves a cross-view consistency FVD of 240.7, compared with 280.0 for γ-World and 443.1 for Solaris. Under recorded controls, it recovers reference block edits and attacks at F1 scores of 0.886 and 0.851, respectively. The demo is at `http://miworld-anonymous.github.io/`.

1 Introduction

Advances in world modeling have improved visual fidelity, enabled longer rollouts, and enhanced action control (Bruce et al., 2024; Valevski et al., 2025; He et al., 2025; Wang et al., 2026b; Sun et al., 2026). These capabilities allow users to navigate and interact with generated environments over successive steps, motivating research on interactive exploration, multiplayer interaction, and consistency over extended horizons (Savva et al., 2026; Wu et al., 2026; Liu et al., 2026).

Recent work moves toward stateful interaction through persistent world-state representations (Mo et al., 2026), shared-state dynamics with view-conditioned rendering (Zhao et al., 2026; Cai et al.,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

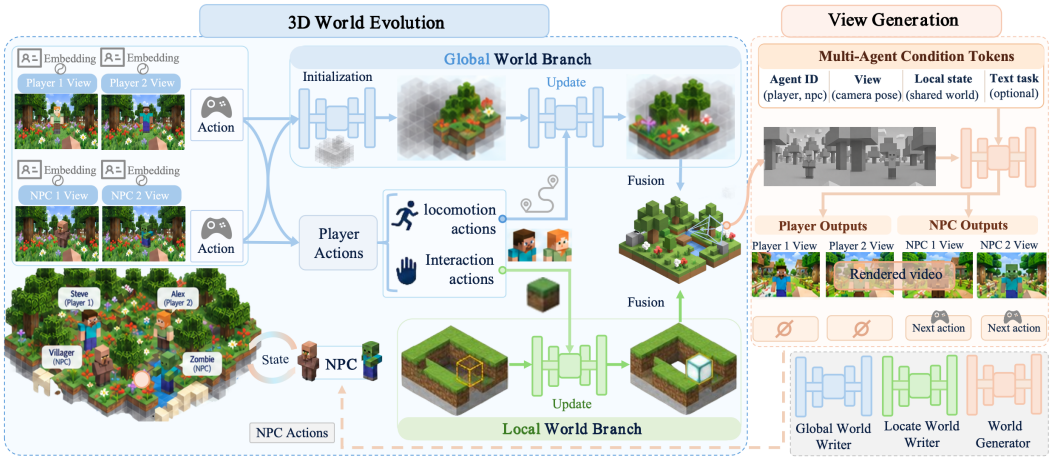


Figure 1: MI-WORLD splits 3D world evolution into global scene completion and local action-driven updates. Human players and autonomous NPCs share one persistent canonical state, which is used to render viewpoint-consistent video and to close the NPC perception-action loop.

2026), editable external memories (Po et al., 2026), and state-aware modeling of game mechanics (Lin et al., 2026). We focus on explorable and editable multi-character worlds, where exploration reveals new content and successive actions can modify both the environment and character states. In this setting, an interaction initiated by one character must remain consequential for others, including those who encounter its outcome only later. To make this requirement explicit, we distinguish two kinds of action. *Locomotion actions* change a character's pose while leaving the environment and other characters unchanged. *Interaction actions* modify the environment or update character attributes. An interaction is *persistent* when its outcome becomes part of the shared state used for subsequent observations and interactions. Exploration and interaction impose complementary requirements on state evolution. When a character enters unobserved territory, generative completion must synthesize new geometry without overwriting established structures or earlier edits. Preserving known content during completion must not, however, prevent subsequent actions from intentionally changing it. Similarly, an interaction that changes an entity attribute must update its stored value rather than merely produce a transient visual effect. Later interactions by any character must then use the updated value, so that their outcomes reflect the shared interaction history. Autonomous NPCs extend this requirement beyond active viewpoints: they must be able to perceive and act on the same evolving world even when no player is observing their actions. Their off-screen modifications must therefore remain available to characters that encounter the affected region later.

As illustrated in Figure 1, MI-WORLD addresses these requirements with an explicit 3D voxel map and an entity table that together form a persistent canonical state shared by all characters. Actions are mapped to structured events that update this state, from which synchronized perspective views are rendered. Scene completion fills unobserved regions while preserving established voxels, whereas action-driven transitions modify voxel occupancy and entity attributes. Autonomous NPCs use the same action interface and transition dynamics as human players, allowing their off-screen actions to contribute to the same evolving world state. Experiments show that MI-WORLD improves cross-view consistency over the evaluated baselines and accurately recovers reference block edits and attacks under recorded controls. Our contributions are as follows:

- We introduce MI-WORLD, a stateful multi-inhabitant world model that formalizes persistent interaction through updates to a shared 3D voxel and entity representation, decoupling spatial exploration, action-driven discrete state transitions, and state-conditioned visual rendering.
- We integrate autonomous NPCs into the shared state-update loop, enabling closed-loop decision making and off-screen environmental modifications under unified transition dynamics.
- We present a synchronized dataset of 45,572 multi-character trajectories (approximately 15.48M observation steps) with explicit ground-truth event attributions, providing a benchmark for persistent multi-agent world modeling.

2 Related Work

Multi-agent interactive world models. Action-conditioned video models handle several agents by coordinating control streams and matching views across cameras. Early systems align two players with identity-aware conditioning and shared encoders (Savva et al., 2026; Wu et al., 2026). Later architectures support larger groups of interacting agents (Liu et al., 2026; Hu et al., 2026). In particular, γ-World uses permutation-symmetric agent encoding and sparse cross-agent communication. However, visual cross-attention leaves the underlying environment implicit, so interaction outcomes can degrade or disappear when viewpoints change. Shared-state approaches explicitly model world evolution. WorldWeaver updates shared world-state registers after each generated chunk (Mo et al., 2026). MASS advances an authoritative state from joint actions, while Khora updates dynamic entity states before independently rendering their views (Cai et al., 2026; Zhao et al., 2026). MultiGen supports editable multiplayer worlds through external scene memory (Po et al., 2026). MI-World focuses on persistent voxel edits and character interactions: it combines completion of unknown regions with action-driven updates to stored geometry and character attributes, and places autonomous NPCs within the same state-update loop.

Persistent memory and state updates. Long-horizon consistency requires environmental states to remain intact after the immediate visual context window has passed. Retrieval models index past frames, surfels, or 3D point-cloud patches by camera pose and geometric coordinates, fetching historical context upon return visits (Xiao et al., 2025; Li et al., 2025; Wu et al., 2025; Ren et al., 2025; Yu et al., 2026). Retrieval supports return visits during passive navigation, but it does not execute physical modifications. Methods that record modifications maintain evolving 3D representations (Garcin et al., 2026), interaction tokens (Xiong et al., 2026), explicit game-mechanic variables (Lin et al., 2026), or executable transition code (Huang et al., 2026; Chen et al., 2026). MI-World explicitly separates geometric expansion during exploration from discrete, action-driven state changes. Because exploratory completion preserves established coordinates, deliberate modifications remain stored in the shared map and can be queried from any participant viewpoint.

Game agents and NPC behavior modeling. Embodied agent research in virtual environments focuses primarily on policies for navigation and task execution. Video PreTraining (Baker et al., 2022), Voyager (Wang et al., 2024), STEVE-1 (Lifshitz et al., 2023), and TeamCraft (Long et al., 2024) learn goal-directed control and collaborative behavior inside an existing game engine that computes all environment transitions. In generative simulators, NPC behavior is learned from visual dynamics (Agarwal et al., 2026) or steered by high-level prompts (Wang et al., 2026a). WorldMind (Deng et al., 2026) closes the loop by inferring a compact state from generated observations and planning subsequent actions. In MI-World, NPCs operate within the same action space as human players and apply updates to the shared 3D state through the same transition model, ensuring off-screen actions take persistent effect even outside active player viewpoints.

3 Method

As shown in Figure 2, MI-World maintains a persistent world shared by human-controlled characters and autonomous NPCs. At step t, its state is

$$S_t = (V_t, K_t, C_t, L_t), \tag{1}$$

where V_t stores voxel categories, K_t marks coordinates with stored values, C_t records character poses and attributes under persistent identities, and L_t logs applied interactions. Stored air is marked as known, so removing a block does not make its location unknown again.

Two writers update this state. The *Global World Writer* ($\mathcal{W}_{\mathrm{G}}$) completes requested regions where $K_t = 0$ without overwriting stored values. The *Local World Writer* ($\mathcal{W}_{\mathrm{L}}$) takes a joint action chunk $A_t = \{a^i_{t:t+T-1}\}_{i=1}^N$ containing player controls and NPC actions,

$$\begin{aligned} (\hat{Q}_{t+1:t+T}, \hat{E}_t) &= \mathcal{W}_{\mathrm{L}}(S_t, A_t; \mathcal{H}_t), \\ S_{t+1:t+T} &= \mathrm{Apply}(S_t, \hat{Q}_{t+1:t+T}, \hat{E}_t). \end{aligned} \tag{2}$$

Here, $\mathcal{H}_t$ contains completed observations and history. $\hat{Q}$ predicts camera poses (position and orientation), while $\hat{E}_t$ predicts block edits and attacks. The deterministic operator Apply executes these predictions in temporal order and records their effects in the shared state.

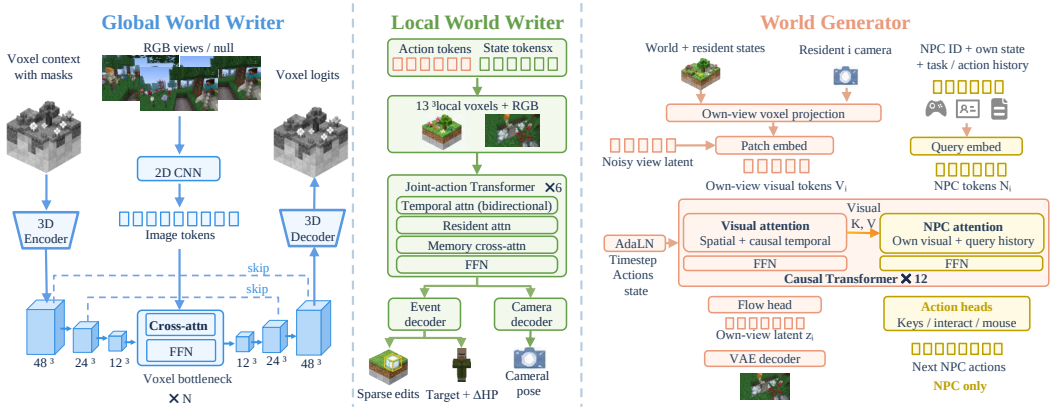


Figure 2: Overview of the MI-WORLD framework. Three decoupled modules operate on a shared 3D representation to execute global voxel expansion, action-driven physical state transitions, and view-consistent visual generation.

Both writers contribute updates to the same shared world state. The *World Generator* renders player views from the state sequence incorporating both writers' updates and predicts the next NPC action chunk from completed rendering features. NPC actions enter the same shared-state update as human controls, including when NPCs are outside the players' views. The updated state conditions subsequent chunks, retaining interaction outcomes for later observations and actions.

3.1 GLOBAL WORLD WRITER: PERSISTENT WORLD EXPANSION

The Global World Writer ($\mathcal{W}_{\mathrm{G}}$) expands the shared voxel map by completing requested unknown regions while preserving stored geometry and interaction outcomes. Let Ω_{req} denote the union of the agent-centered voxel windows selected for completion. We define the write mask

$$M_t(x) = \mathbb{I}[x \in \Omega_{\mathrm{req}} \wedge K_t(x) = 0]. \tag{3}$$

Inputs include masked voxel context, known and request masks, and optional RGB views I_t. Initialization is the fully masked case: $K_0(x) = 0$ and $M_0(x) = 1$ throughout the initial requested region. Given only two RGB views $I_0 = \{I_0^1, I_0^2\}$, the same network generates the initial shared voxel map without any known voxel context. During subsequent exploration, overlapping stored geometry conditions completion of newly requested unknown regions, with or without images.

The network encodes voxel context into multiscale 3D features. We encode the RGB views into spatial latent tokens using the encoder of PERSIST's pretrained 2D VAE (Garcin et al., 2026). The voxel bottleneck reads these tokens through cross-attention followed by feed-forward layers. The decoder combines bottleneck features with encoder features through skip connections and produces voxel-category logits:

$$\begin{aligned} \ell_t &= \mathcal{W}_{\mathrm{G}}(\mathrm{Mask}(V_t, K_t), K_t, M_t, I_t)\,, \\ \hat{V}_t(x) &= \arg\max_{c \in \mathcal{V}} \ell_t(x, c). \end{aligned} \tag{4}$$

Here, Mask replaces unknown entries with a dedicated unknown token, and $\mathcal{V}$ denotes the voxel-category vocabulary. Encoding and training details are given in Appendix A.

We write the predicted category where $M_t(x) = 1$ and retain the stored value elsewhere. Let V_t^+ denote the map after this completion:

$$V_t^+(x) = \begin{cases} \hat{V}_t(x), & M_t(x) = 1, \\ V_t(x), & M_t(x) = 0, \end{cases} \tag{5}$$

We then mark the completed voxels as known, $K_t^+(x) = K_t(x) \vee M_t(x)$. Completion is committed to the shared state through an in-place update:

$$S_t \leftarrow (V_t^+, K_t^+, C_t, L_t).$$

This update preserves character records and interaction logs. The input S_t in Equation 2 denotes the latest committed shared state at the start of the local update, including any global completion committed before that call. All characters read this shared state.

3.2 Local World Writer: Action-Driven World Updates

The Local World Writer ($\mathcal{W}_{\mathrm{L}}$) predicts camera trajectories and interaction effects under joint controls over T steps. For each character, we encode its boundary state and action sequence as tokens indexed by character and time. A 13^3 character-centered voxel window, nearby character states, and latent features of the completed view provide scene context at the chunk boundary.

A six-block joint-action Transformer combines these inputs. Each block applies bidirectional temporal attention within the action chunk, cross-character attention, cross-attention to the scene context, and a feed-forward layer. Temporal attention relates actions across steps, while cross-character attention accounts for the other participants' controls. Two decoders read the resulting shared features. The *Camera Decoder* predicts each character's camera pose (position and orientation) at every step.

The *Event Decoder* predicts sparse block edits and attacks. For each acting character, learned queries predict an event count and the time, operation, and target of each event. We represent an event as

$$e = (k, s, g, \tau, \varphi), \tag{6}$$

where k is the step within the chunk, s the acting character, g the target, τ the operation, and φ its effect. Removal and placement select voxel targets, while attacks select character targets. A shared payload network combines event and target features to predict the new block category or signed health change Δh.

3.3 World Generator: Views and NPC Actions

The World Generator produces each character's view and the next NPC controls from the updated shared state. For character i, a projection Π_i maps voxel geometry and character states into its camera (Garcin et al., 2026; Zhao et al., 2026). We patch-embed the projected features together with noisy image latents to form that character's visual tokens. The visual branch uses a 12-block causal Transformer with spatial attention and temporal attention that is causal across chunks. Adaptive layer normalization conditions the blocks on the flow timestep, actions, and state. A flow head predicts the rectified-flow velocity to generate view latents in the space of PERSIST's frozen 2D VAE:

$$z^i_{t+1:t+T} \sim \mathcal{R}\big(\cdot \mid \Pi_i(S_{t+1:t+T}), A_t, H^i_t\big)\,, \tag{7}$$

where H^i_t is the completed visual history of character i. All characters share the renderer weights and maintain their own visual histories. Completed own-view latent features provide visual context for the next Local World Writer call. The VAE decoder converts player latents into RGB; NPCs use latent features directly for state updates and action prediction.

For NPC j, query tokens encode its identity, own state, task, and executed actions. NPC attention reads its own completed visual features as keys and values and attends causally to its query history. This connection is one-way: visual tokens do not attend to NPC tokens. Action heads read the final NPC token to predict keyboard controls, interactions, and mouse motion for the next chunk:

$$a^j_{t+T:t+2T-1} \sim \mathcal{D}(\cdot \mid F^j_{t+T}, h^j_{t+T}, y^j), \tag{8}$$

where F^j_{t+T} contains the completed block's visual features, h^j_{t+T} its state/action history, and y^j its task instruction when provided.

3.4 Training

We train the modules using synchronized voxels, character states, actions, events, and RGB frames. Both the pretrained 3D and 2D VAEs are from PERSIST (Garcin et al., 2026). The Global World Writer learns masked voxel completion with its 3D VAE fixed. The Local World Writer jointly optimizes camera-pose regression and event prediction, with both losses updating its shared backbone. Within the World Generator, the renderer learns video generation through flow matching with the 2D VAE frozen. The NPC policy is trained by behavior cloning on completed rendering features, keeping the renderer fixed. Objectives and training settings are summarized in Appendix A.

Table 1: Quantitative comparison of multiplayer video generation on the same selected evaluation set across interaction categories (FVD / FID, lower is better; bold denotes best).

Method	Movement	Grounding	Memory	Building	Consistency
Frame concatenation	556.9/65.0	528.3/63.2	450.6/69.8	551.8/87.3	576.0/123.2
Solaris	311.1/36.3	301.9/36.1	333.8/51.7	448.6/71.0	443.1/94.8
γ-World	191.5/21.2	199.3/24.0	184.1/24.8	264.5/32.1	280.0/46.9
MI-WORLD (ours)	**151.3/19.8**	**173.5/18.6**	**166.9/17.3**	**228.4/27.0**	**240.7/38.2**

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We collect synchronized multi-character trajectories in Luanti/VoxeLibre through Craftium, building on Solaris's activity coordination and PERSIST's structured-state recording (Malagón et al., 2025; Savva et al., 2026; Garcin et al., 2026). Automated expert policies control all characters during data collection. Each trajectory contains two to four characters and records their RGB views, actions, voxel observations, character states, and attributed block-edit and damage events. The dataset contains 45,572 trajectories and approximately 15.48M synchronized time points, with deterministic training, validation, and test splits. The 14 collection scenarios cover construction and resource manipulation, coordinated navigation, and combat. Task specifications assign character roles and action sequences, while event annotations identify the acting character, target, and applied effect. Each synchronized time point includes the participating characters' views of the same world, so the observation-step count is separate from the total number of individual RGB frames. Appendix B describes collection, filtering, and task-instruction annotation.

We evaluate multiplayer video generation and structured interaction prediction under recorded joint actions, including recorded NPC actions. Our video evaluation follows the task design of Solaris (Savva et al., 2026). Video generation is compared with frame concatenation, Solaris, and γ-World on the same selected evaluation trajectories containing two to four characters, using FVD and FID. The 45,572 trajectories describe the full collection; evaluation uses selected trajectories rather than the entire collection. Interaction prediction starts from recorded states and measures block-edit and attack recovery, together with damage prediction error, against held-out labels. We also present qualitative world-expansion rollouts over 600 steps. The event benchmark measures transition prediction under supplied controls; the NPC policy described in Section 3.3 instead predicts its own controls during autonomous execution.

4.2 COMPARISON WITH MULTIPLAYER WORLD MODELS

We compare MI-WORLD with frame concatenation, Solaris (Savva et al., 2026), and γ-World (Liu et al., 2026) on the same selected evaluation set. Frame concatenation joins the synchronized views before video generation and keeps no explicit shared state. Solaris uses joint cross-player attention and learned player identities, while γ-World coordinates participants through permutation-symmetric agent encoding and sparse cross-agent communication. Table 1 reports FVD and FID across movement, grounding, memory, building, and cross-view consistency. These scores assess the distributions of generated clips and frames within each category. Building and memory sequences add geometry changes and return visits to the camera motion present in navigation. We therefore examine whether the visual-quality advantage extends to settings where earlier interactions affect later observations. The consistency category evaluates videos with multiple viewpoints; direct event correctness is measured separately in Section 4.3. As shown in Table 1, MI-WORLD attains the lowest FVD and FID across five categories. On movement tasks, MI-WORLD reaches an FVD of 151.3 and an FID of 19.8, compared with γ-World (191.5 FVD and 21.2 FID), Solaris (311.1 FVD and 36.3 FID), and frame concatenation (556.9 FVD and 65.0 FID). The advantage extends beyond navigation to sequences involving construction, revisiting, and multiple viewpoints.

On building tasks, MI-WORLD achieves an FVD of 228.4, compared with 264.5 for γ-World and 448.6 for Solaris. Block placements and excavations change the geometry that subsequent views must depict. MI-WORLD conditions each viewpoint on the same updated voxel map, providing a common

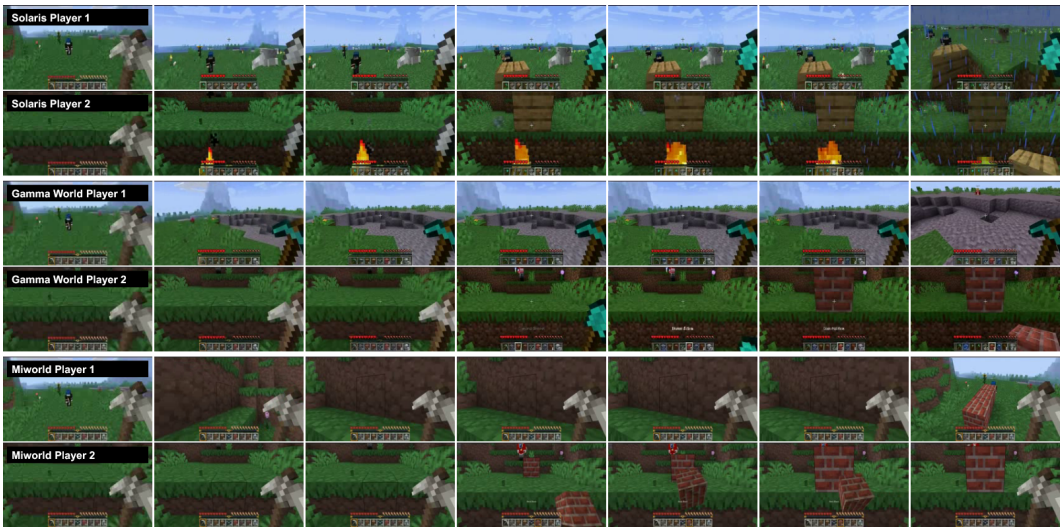


Figure 3: Qualitative comparison of multiplayer video generation. Synchronized viewpoints of two interacting players generated by Solaris, γ-World, and MI-WORLD. MI-WORLD maintains consistent spatial modifications across perspectives.

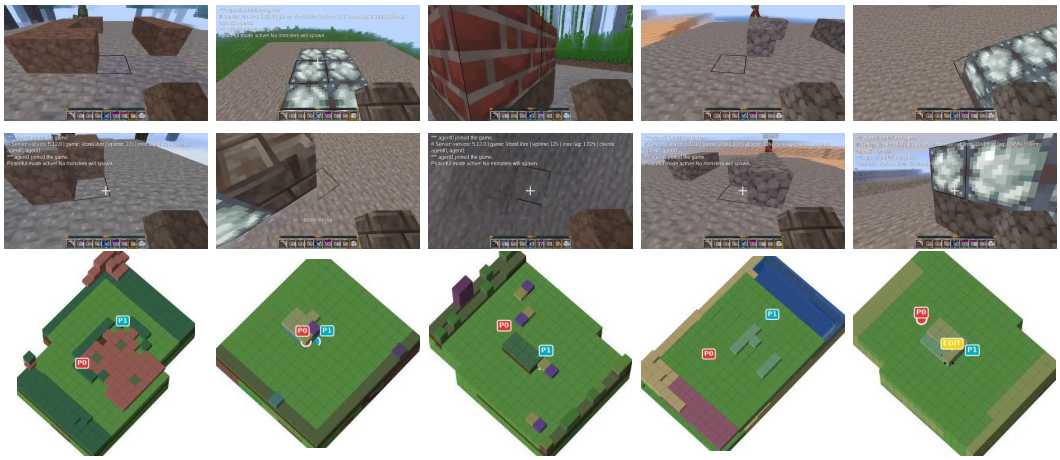


Figure 4: Local 3D state updates during block editing. Each column shows first-person observations from two viewpoints (top two rows) and the corresponding local voxel state (bottom).

geometric reference for these changes. Figure 3 presents synchronized viewpoints for inspecting whether visible edits agree across cameras. In the cross-view consistency category, MI-WORLD also achieves lower FVD and FID (240.7 / 38.2) than γ-World (280.0 / 46.9) and Solaris (443.1 / 94.8).

On memory tasks, where characters return to previously explored or modified regions, MI-WORLD obtains an FVD of 166.9 and an FID of 17.3, compared with 184.1 / 24.8 for γ-World and 333.8 / 51.7 for Solaris. The shared map supplies stored geometry upon return, even when the corresponding frames have left a character's recent visual history. This yields persistent geometric conditioning despite potential appearance errors. While FVD and FID assess visual quality, the subsequent evaluation directly verifies whether predicted interactions align with recorded targets and effects.

4.3 BLOCK EDITING AND CHARACTER INTERACTIONS

We evaluate the Local World Writer on held-out eight-step windows with recorded initial states and joint controls, including NPC actions. Future states and events are withheld, and both interaction and no-event windows are included. Events are matched one-to-one by acting character, target, and

Table 2: Structured interaction prediction on held-out action sequences. Damage MAE is computed over matched attacks with valid damage labels.

Interaction	GT / predicted events	Precision ↑	Recall ↑	F1 ↑	Damage MAE ↓	Matched damage labels
Block edits	1,200 / 1,170	0.897	0.875	0.886	-	-
Attacks	480 / 465	0.865	0.838	0.851	0.64	402

Table 3: State accuracy over continuous rollouts under recorded joint actions. Edited-Voxel Accuracy measures category agreement at reference-edited coordinates, while Health MAE measures character health error at each checkpoint.

Rollout length	Edited-Voxel Accuracy ↑	Health MAE ↓
8 steps	91.86%	0.141
32 steps	90.80%	0.299
64 steps	82.27%	0.468

simulation step. Block edits additionally require the correct operation and resulting category; attacks require positive predicted damage. We report precision, recall, F1, and Damage MAE over matched attacks with valid labels. Detailed matching and aggregation rules are provided in Appendix C.

Block editing. As shown in Table 2, MI-WORLD recovers 1,050 of 1,200 reference edits among 1,170 predictions, achieving an F1 of 0.886. Exact matching checks whether controls produce the intended changes at the correct coordinates and simulation steps, including the resulting block categories. Figure 4 illustrates local voxel updates during editing.

Character interactions. MI-WORLD recovers 402 of 480 reference attacks among 465 predictions, achieving an F1 of 0.851. Damage MAE is 0.64 health units on matched attacks, compared with a default character health of 20. Since missed attacks are excluded from this conditional error, it should be interpreted alongside attack precision and recall.

Continuous state rollouts. We further execute consecutive action chunks under recorded joint controls, providing reference states and observations only at initialization. Predicted states and generated visual context condition subsequent chunks without reference-state resets. The same eligible trajectories are evaluated at 8, 32, and 64 steps. Edited-Voxel Accuracy measures category agreement at reference-edited coordinates against their current state, including air after removal. Health MAE measures health error across characters with valid labels, including the effects of missed attacks, false attacks, and accumulated damage errors.

Table 3 shows a small decrease in Edited-Voxel Accuracy from 91.86% at eight steps to 90.80% at 32 steps, followed by a larger decline to 82.27% at 64 steps. Health MAE increases from 0.141 to 0.468, indicating growing state error over longer rollouts. These results assess transition prediction under supplied controls; autonomous NPC task completion requires a separate policy evaluation.

Persistence and prediction accuracy. MI-WORLD retains interaction outcomes in the shared state used by subsequent updates and observations. Once an edit is applied, its voxel value remains stored until a later interaction changes it. This includes air after removal: the known mask prevents exploratory completion from filling the location again. Prediction accuracy measures whether these stored outcomes agree with the reference trajectory. An incorrect edit or damage estimate can therefore persist and influence later predictions, contributing to the accumulated errors in Table 3. The two evaluation protocols examine complementary aspects of this process: event matching measures the recovery of individual interactions, while continuous rollouts measure the resulting state after successive updates. Together, they assess both the immediate effects of supplied controls and how prediction errors affect the world inherited by subsequent interactions.

4.4 LONG-HORIZON WORLD EXPANSION AND AGENT ROLLOUTS

Figure 5 shows spatial expansion and multi-agent rollouts over 600 simulation steps. As agents navigate into unexplored regions, the Global World Writer synthesizes new terrain while leaving established coordinates intact. The rendered first-person perspectives reflect this underlying state,

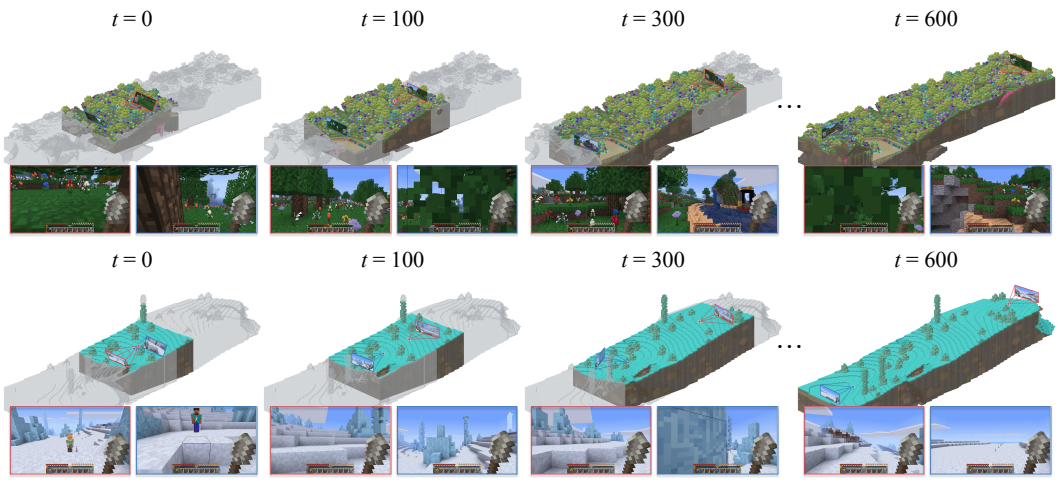


Figure 5: Long-horizon global map expansion and agent rollouts. Isometric 3D voxel terrain expanding over 600 simulation steps ($t = 0, 100, 300, 600$), with synchronized first-person viewpoints for two agents (red and blue markers) and co-present NPCs.



Text-guided NPC behavior: Building a solid 2×2×2 stone-brick cube



Action-driven character interaction: Defeating a zombie

Figure 6: Qualitative examples of text-guided NPC behavior and action-driven character interactions. Top: successive block placements form the target structure, shown from changing first-person viewpoints. Bottom: the zombie exhibits a red hit response and disappears upon defeat. Frames in each row progress from left to right.

maintaining identity and geometric consistency when agents interact with co-present NPCs over extended horizons. Figure 6 complements the world-expansion sequence with examples of text-conditioned NPC behavior and a player attacking a zombie.

5 Conclusion

In this work, we introduced MI-World, an interactive world model that grounds multi-character simulation in a shared 3D voxel map and structured entity records. By separating scene completion, action-driven state updates, and view generation, MI-World preserves established structures during exploration and retains environmental edits and character-state changes for subsequent observations and interactions. All character views are rendered from the updated canonical state. Autonomous NPCs use the same transition interface as human players, allowing their off-screen actions to modify the shared world that other participants encounter later. Experiments show improved cross-view consistency over the evaluated baselines and accurate recovery of reference block edits and attacks under recorded controls. We also contribute a dataset of 45,572 synchronized multi-character trajectories with fine-grained interaction annotations, providing a resource for studying how action outcomes persist across characters, viewpoints, and time.

AI USE STATEMENT

Generative AI tools were used to assist with English-language editing and literature organization. The authors are responsible for verifying all claims, citations, technical details, and the final manuscript.

REFERENCES

Anmol Agarwal, Pranay Meshram, Sumer Singh, Saurav Suman, Andrew Lapp, Shahbuland Matiana, Louis Castricato, and Spencer Frazier. COMBAT: Conditional world models for behavioral agent training. *arXiv preprint arXiv:2603.00825*, 2026.

Bowen Baker, Ilge Akkaya, Peter Zhokov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (VPT): Learning to act by watching unlabeled online videos. In *NeurIPS*, 2022.

Jake Bruce, Michael D. Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Maria Elisabeth Bechtle, Feryal Behbahani, Stephanie C. Y. Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In *ICML*, 2024.

Ziqi Cai, Siqi Yang, Yimu Wang, Zixian Gao, Yunheng Liu, Shuchen Weng, Erwin Wu, Kaipeng Zhang, and Boxin Shi. MASS: Multiplayer world models with authoritative shared state. *arXiv preprint arXiv:2608.06257*, 2026.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion Forcing: Next-token prediction meets full-sequence diffusion. In *NeurIPS*, 2024.

Yiwen Chen, Guosheng Lin, and Chi Zhang. Code world model: Coding agent as world brain. *arXiv preprint arXiv:2608.25927*, 2026.

Zhiyang Deng, Boran Zhang, Danze Chen, and Yeying Jin. WorldMind: Decoupled game world model for state-aware NPC behavior. *arXiv preprint arXiv:2608.21439*, 2026.

Samuel Garcin, Thomas Walker, Steven McDonagh, Tim Pearce, Hakan Bilen, Tianyu He, Kaixin Wang, and Jiang Bian. Beyond pixel histories: World models with persistent 3d state. In *ICML*, 2026.

Xianglong He, Chunli Peng, Zexiang Liu, Boyang Wang, Yifan Zhang, Qi Cui, Fei Kang, Biao Jiang, Mengyin An, Yangyang Ren, Baixin Xu, Hao-Xiang Guo, Kaixiong Gong, Size Wu, Wei Li, Xuchen Song, Yang Liu, Yangguang Li, and Yahui Zhou. Matrix-Game 2.0: An open-source real-time and streaming interactive world model. *arXiv preprint arXiv:2508.13009*, 2025.

Anthony Hu, Václav Volhejn, Adrien Ramanana Rahary, Chris Mulder, Aditya Makkar, Alyx Liao, Amélie Royer, Manu Orsini, Adam Jelley, Eloi Alonso, Florian Laurent, Fredrik Norén, James Swingos, Jan Hünermann, Kent Rollins, Lucas Hosseini, Matthieu Le Cauchois, Maxim Peter, Pim de Witte, Tim Brown, Vincent Micheli, Moritz Böhle, Gabriel de Marmiesse, Viktoriia Sharmanska, Lucia Specia, Michael Black, and Patrick Pérez. Multiplayer interactive world models with representation autoencoders. *arXiv preprint arXiv:2607.05352*, 2026.

Zheng-Hui Huang, Guixu Lin, Jiacheng Lin, Yi-Chuan Huang, Ruihan Yu, Muyao Niu, Siqi Yang, Yu-Lun Liu, Yung-Yu Chuang, Kaipeng Zhang, and Zhixiang Wang. Programmable world model. *arXiv preprint arXiv:2609.10540*, 2026.

Runjia Li, Philip Torr, Andrea Vedaldi, and Tomas Jakab. VMem: Consistent interactive video scene generation with surfel-indexed view memory. In *ICCV*, 2025.

Shalev Lifshitz, Keiran Paster, Harris Chan, Jimmy Ba, and Sheila McIlraith. STEVE-1: A generative model for text-to-behavior in Minecraft. In *NeurIPS*, 2023.

Zijun Lin, Zeqing Wang, Cheston Tan, Bihan Wen, and Yeying Jin. StatePlay: State-aware game world models for mechanics-consistent generation. *arXiv preprint arXiv:2607.26754*, 2026.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *ICLR*, 2023.

Fangfu Liu, Kai He, Tianchang Shen, Tianshi Cao, Sanja Fidler, Yueqi Duan, Jun Gao, Igor Gilitschenski, Zian Wang, and Xuanchi Ren. Gamma-World: Generative multi-agent world modeling beyond two players. *arXiv preprint arXiv:2605.28816*, 2026.

Qian Long, Zhi Li, Ran Gong, Ying Nian Wu, Demetri Terzopoulos, and Xiaofeng Gao. TeamCraft: A benchmark for multi-modal multi-agent systems in Minecraft. *arXiv preprint arXiv:2412.05255*, 2024.

Mikel Malagón, Josu Ceberio, and Jose A. Lozano. Craftium: Bridging flexibility and efficiency for rich 3D single- and multi-agent environments. In *ICML*, pages 42765–42794, 2025.

Sicheng Mo, Yuheng Li, Ziyang Leng, Krishna Kumar Singh, and Bolei Zhou. Streaming multi-agent autoregressive diffusion model with world state registers. *arXiv preprint arXiv:2607.21594*, 2026.

Ryan Po, David Junhao Zhang, Amir Hertz, Gordon Wetzstein, Neal Wadhwa, and Nataniel Ruiz. MultiGen: Level-design for editable multiplayer worlds in diffusion game engines. *arXiv preprint arXiv:2603.06679*, 2026.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

Xuanchi Ren, Tianchang Shen, Jiahui Huang, Huan Ling, Yifan Lu, Merlin Nimier-David, Thomas Müller, Alexander Keller, Sanja Fidler, and Jun Gao. GEN3C: 3D-informed world-consistent video generation with precise camera control. In *CVPR*, 2025.

Georgy Savva, Oscar Michel, Daohan Lu, Suppakit Waiwitlikhit, Timothy Meehan, Dhairya Mishra, Srivats Poddar, Jack Lu, and Saining Xie. Solaris: Building a multiplayer video world model in minecraft. *arXiv preprint arXiv:2602.22208*, 2026.

Wenqiang Sun, Haiyu Zhang, Haoyuan Wang, Junta Wu, Zehan Wang, Zhenwei Wang, Yunhong Wang, Jun Zhang, Tengfei Wang, and Chunchao Guo. WorldPlay: Towards long-term geometric consistency for real-time interactive world modeling. In *ICML*, 2026.

Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines. In *ICLR*, 2025.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *TMLR*, 2024.

Zeqing Wang, Danze Chen, Zhaohu Xing, Zizhao Tong, Yinhan Zhang, Xingyi Yang, and Yeying Jin. ReactiveGWM: Steering NPC in reactive game world models. *arXiv preprint arXiv:2605.15256*, 2026a.

Zile Wang, Zexiang Liu, Jiaxing Li, Kaichen Huang, Baixin Xu, Fei Kang, Mengyin An, Peiyu Wang, Biao Jiang, Yichen Wei, Yidan Xietian, Jiangbo Pei, Liang Hu, Boyi Jiang, Hua Xue, Zidong Wang, Haofeng Sun, Wei Li, Wanli Ouyang, Xianglong He, Yang Liu, Yangguang Li, and Yahui Zhou. Matrix-Game 3.0: Real-time and streaming interactive world model with long-horizon memory. *arXiv preprint arXiv:2604.08995*, 2026b.

Haoyu Wu, Jiwen Yu, Yingtian Zou, and Xihui Liu. MultiWorld: Scalable multi-agent multi-view video world models. *arXiv preprint arXiv:2604.18564*, 2026.

Tong Wu, Shuai Yang, Ryan Po, Yinghao Xu, Ziwei Liu, Dahua Lin, and Gordon Wetzstein. Video world models with long-term spatial memory. In *NeurIPS*, 2025.

Zeqi Xiao, Yushi Lan, Yifan Zhou, Wenqi Ouyang, Shuai Yang, Yanhong Zeng, and Xingang Pan. WorldMem: Long-term consistent world simulation with memory. In *NeurIPS*, 2025.

Zhexiao Xiong, Yizhi Song, Hao Kang, Qing Yan, Liming Jiang, Jenson Yang, Zhoujie Fu, Stathi Fotiadis, Angtian Wang, Zichuan Liu, Bo Liu, Yiding Yang, Xin Lu, and Nathan Jacobs. ActWorld: From explorable to interactive world model via action-aware memory. In *NeurIPS*, 2026.

Wei Yu, Runjia Qian, Yumeng Li, Liquan Wang, Songheng Yin, Sri Siddarth Chakaravarthy P, Dennis Anthony, Yang Ye, Yidi Li, Weiwei Wan, and Animesh Garg. MosaicMem: Hybrid spatial memory for controllable video world models. *arXiv preprint arXiv:2603.17117*, 2026.

Renjie Zhao, Yuxiang Wu, Mingyu Zhang, Jiaxin Li, Sisi Li, He Li, Yimin Sheng, Tianxi Tan, Zhenkai Zhang, Jiao Liang, Jianyi Zhu, and Yong-Lu Li. Population-scalable multi-agent world modeling. *arXiv preprint arXiv:2608.08600*, 2026.

A Model Configuration and Training

Table A.1 summarizes the three modules introduced in Section 3 and their input/output settings. The two writers are trained on synchronized state, action, and event records. Within the World Generator, the renderer is trained first and then frozen during NPC behavior cloning. Losses exclude invalid labels from both their numerators and normalization counts.

Table A.1: Module configurations of MI-World. Visual generation and NPC action prediction are two branches of the World Generator.

Module	Configuration
Global World Writer	48^3 voxel tiles; masked voxel context and optional RGB conditioning; voxel-category output
Local World Writer	Six joint-action Transformer blocks; 13^3 character-centered voxel windows; eight-step joint controls; Camera Decoder and Event Decoder
World Generator	Visual generation: 12 causal Transformer blocks; one context frame and 64 future frames per training sequence; eight frames per generated block. NPC action prediction: eight completed own-view frames; one query per frame; eight-step action prediction; 10 binary controls, 10 hotbar classes, and 17 mouse-motion bins per axis

A.1 Global World Writer

The voxel encoder and decoder use PERSIST's pretrained 3D VAE and remain fixed during completion training (Garcin et al., 2026). RGB conditions are encoded with its pretrained 2D VAE. Voxel features read the image tokens through bottleneck cross-attention before decoding voxel-category logits.

Let $r(x)$ indicate a reliable target label and $m(x) = M_t(x)r(x)$ select supervised completion locations. The voxel objective is

$$\mathcal{L}_{\text{voxel}} = \frac{\sum_x m(x)\,\text{CE}(\ell_t(x), V^\star(x))}{\sum_x m(x)}, \tag{9}$$

where $V^\star$ is the reference voxel category. Samples with no valid completion targets do not contribute to this term. Known context voxels condition the network but do not contribute to the completion loss. The write mask in Equation 3 also restricts inference to unknown coordinates, preserving previously stored geometry and edits.

A.2 Local World Writer

Each character contributes its boundary state, latent features of its completed view, and eight-step control sequence. Local voxel tokens encode categories, known masks, and relative coordinates. The joint Transformer uses bidirectional attention within the supplied action chunk and attention across characters. Past state/action tokens can provide additional context for the Event Decoder; padded characters and invalid history entries are masked out.

The shared backbone is optimized by

$$\mathcal{L}_{\text{local}} = \mathcal{L}_{\text{camera}} + \lambda_{\text{event}}\mathcal{L}_{\text{event}}. \tag{10}$$

The Camera Decoder objective combines Smooth-L_1 position regression with an orientation loss respecting angular periodicity, averaged over active characters with valid component labels. The Event Decoder objective comprises

$$\mathcal{L}_{\text{event}} = \mathcal{L}_{\text{count}} + \mathcal{L}_{\text{time}} + \mathcal{L}_{\text{operation}} + \mathcal{L}_{\text{target}} + \mathcal{L}_{\text{block}} + \mathcal{L}_{\text{damage}}. \tag{11}$$

Count, time, operation, target, and block category use cross-entropy; signed health changes use Smooth-L_1. Reference events are assigned to query slots in temporal order. Count supervision includes valid no-event windows; slot losses use active reference events, and payload losses use valid labels for the corresponding operation. Character windows with incomplete event annotations or

more events than the query capacity are excluded from event supervision. Their valid camera labels remain usable.

Both objectives update the joint-action Transformer. The Camera Decoder predicts $\hat{Q}_{t+1:t+T}$, and the Event Decoder predicts $\hat{E}_t$; their effects are applied to the shared state as described in Appendix A.4.

A.3 World Generator

Visual generation. The visual branch $\mathcal{R}$ learns conditional flow matching with PERSIST's 2D VAE frozen (Lipman et al., 2023). For a clean latent x_1, Gaussian noise x_0, and flow time $\lambda \sim \mathcal{U}[0,1]$, the interpolation and target velocity are

$$x_\lambda = (1-\lambda)x_0 + \lambda x_1, \qquad u_\lambda = x_1 - x_0. \tag{12}$$

With conditions κ comprising projected shared state, actions, and completed visual history, the objective is

$$\mathcal{L}_{\mathrm{FM}}^{\mathrm{rgb}} = \mathbb{E}\Big[\mathrm{mean}_{\text{valid future latents}}\left(v_\psi(x_\lambda, \lambda; \kappa) - u_\lambda\right)^2\Big]. \tag{13}$$

Training uses causal diffusion forcing over blocks of eight frames (Chen et al., 2024). Attention is joint within a block and causal across blocks. At inference, the velocity field is integrated from $\lambda = 0$ to 1, and completed blocks are added to each character's visual history H_t^i. The flow loss weights valid latent elements uniformly.

NPC action prediction. The action branch $\mathcal{D}$ reads the NPC's completed visual features F_{t+T}^j and combines them with its state/action history h_{t+T}^j and optional task instruction y^j. Its visual context comprises eight completed own-view frames. Visual tokens do not attend to policy queries. The final query predicts the next eight actions while the renderer remains fixed.

For prediction step $k \in \{0, \ldots, 7\}$, let

$$\ell_k^{\mathrm{act}} = \mathrm{BCE}_{\mathrm{keys},k} + \mathrm{CE}_{\mathrm{hotbar},k} + \mathrm{CE}_{\text{mouse-x},k} + \mathrm{CE}_{\text{mouse-y},k}\,. \tag{14}$$

The key loss is averaged over the 10 binary controls. Hotbar classes represent no selection and slots 1–9; mouse targets select the nearest bin for each axis. With valid-step mask v_k and horizon weight $w_k = 8 - k$, behavior cloning minimizes

$$\mathcal{L}_{\mathrm{BC}} = \mathbb{E}_{\mathrm{demo}}\left[\frac{\sum_{k=0}^{7} v_k w_k \ell_k^{\mathrm{act}}}{\sum_{k=0}^{7} v_k w_k}\right]. \tag{15}$$

Samples with no valid target actions are excluded. Task instructions are encoded by a frozen T5-v1.1-base encoder (Raffel et al., 2020), with a maximum length of 64 tokens. Their annotation is described in Appendix B.3.

A.4 Shared-State Updates and Rollouts

Interaction targets. A local window keeps its integer center b_t^s fixed throughout the eight-step call. An index $\ell \in \{0, \ldots, 12\}^3$ maps to world coordinate $g = b_t^s - (6,6,6) + \ell$. Character targets use the persistent identities in C_t. These targets give the events in Equation 6 a shared spatial and temporal reference across character views.

Applying predicted effects. The operator Apply in Equation 2 executes camera-pose predictions and interaction events in temporal order. Removal writes air, placement writes the predicted block category, and attacks change the target character's health by $\Delta\mathrm{hp}$. The resulting states form $S_{t+1:t+T}$, with each state retaining the updated voxel map, character records, and accumulated interaction log. Stored air remains known, so subsequent global completion preserves a removed block's location. Completion cannot revise a stored value; an incorrect completion therefore persists until an interaction changes it.

Generating the next chunk. The World Generator conditions each character's view on the corresponding updated state and its own completed visual history. Its NPC action branch predicts the next action chunk from completed rendering features. These actions join the player controls in the next A_t. Completed own-view latent features also condition the next Local World Writer call. Characters share renderer weights while maintaining separate visual histories, and only player view latents are decoded to RGB.

B Dataset

B.1 Collection and Recorded Modalities

Trajectories are collected in Luanti/VoxeLibre through Craftium (Malagón et al., 2025). Each shared world contains two to four characters controlled by automated expert policies. Player and NPC denote character types; neither implies human demonstration data. Task templates specify roles, behavior sequences, and completion conditions.

At each step, the collector applies the joint action $u_t^{1:N}$ and records $(S_{t+1}, O_{t+1}^{1:N}, E_{t\to t+1})$. State and RGB observations belong to time points; actions and events belong to the intervals between them. Thus, a trajectory with H joint actions contains $H + 1$ synchronized observation time points, including initialization.

Records include RGB, actions, camera parameters, local voxels, character pose, health, inventory, and lossless instance masks. Block-edit events identify the source, step, world coordinate, operation, and resulting category; damage events identify the source, target character, step, and effect. Voxel observations cover local neighborhoods rather than a full export of the world.

Filtering. Trajectory filtering checks temporal alignment, video decodability and frame counts, task completion, and valid event source/target records. Missing field labels are masked separately for each training objective and evaluation metric.

B.2 Scale, Task Coverage, and Splits

The collection contains 45,572 trajectories, 15,478,109 synchronized observation time points, and 40,929,257 individual RGB frames. The corresponding joint-action count is 15,432,537. In particular, the reported 15.48M counts synchronized time points rather than the sum of frames across cameras. Table B.1 gives the fixed data splits, and Table B.2 lists all 14 activity categories.

Table B.1: Dataset splits. Observation counts are synchronized across the participating characters.

Split	Trajectories	Observation time points	Individual RGB frames
Training	40,403	13,679,582	36,219,668
Validation	2,576	890,755	2,334,475
Test	2,593	907,772	2,375,114
Total	45,572	15,478,109	40,929,257

Table B.2: Trajectory counts by collection activity.

Group	Activity	Trajectories
Building and resources	Build structure	14,045
	Collector	1,050
	Mine together	1,050
	Place and mine	6,248
	Build house	1,800
	Tower bridge	1,050
	Build tower	1,050
Movement	Walk and look	1,730
	Straight-line walk	1,666
	Orbit	1,785
	Chase	1,649
	Walk and look away	1,676
Combat and mixed tasks	Player versus environment	8,487
	Player versus player	2,286

There are 26,233 two-character, 12,915 three-character, and 6,424 four-character trajectories. Episode identifiers and episode seeds are disjoint across the three splits. Task templates can recur across splits,

so these splits evaluate held-out episodes within the collection's task distribution rather than unseen task families. Training and evaluation windows inherit the split of their parent trajectory.

B.3 Instruction Annotation

For text-conditioned tasks, instructions are instantiated from the collection specification: operation, shape, dimensions, material, and ordered substeps. For example, a 4×2 sea-lantern floor is described as "Lay a 4 by 2 floor using sea lanterns." The same specification defines the expert controller's target geometry and action sequence.

An instruction is stored with its trajectory and remains constant across the associated training windows. The task condition y^j in Equation 8 associates an instruction with NPC j. The current dataset supplies a shared trajectory-level description, which is used for the participating NPCs. Text is masked when language conditioning is absent. Whitespace-normalized English sentences are deduplicated and encoded once for caching.

A policy sample consists of a trajectory, target character, and action start t. Completed history ends at t, and supervision covers $a^j_{t:t+7}$. Language-conditioned samples require a valid nonempty instruction. Action masks exclude dead characters, invalid health records, and termination crossings.

C Evaluation Protocols

C.1 Structured Event Prediction

Each eight-step window starts from its recorded state and observation context and supplies the recorded joint controls, including NPC actions. Future states and events are withheld. This protocol resets the initial state for each window; autonomous policy execution is a separate setting.

Reference events are selected by annotation validity before any truncation to model query capacity. Valid events beyond the capacity remain in the recall denominator. Unknown annotations are not negative labels, while valid no-event windows are retained to count false positives.

Predictions are decoded normally and mapped to world voxel coordinates or persistent character identities. Within each window, maximum-cardinality one-to-one matching uses the criteria in Table C.1. Duplicates receive at most one match. Ties are resolved by event order, without using damage error to select favorable pairs.

Table C.1: Exact matching criteria for structured events.

Event	Required agreement
Block edit	Acting character, step, world voxel coordinate, operation, and resulting block category; air represents removal
Attack	Attacker, target character, and step; the prediction must apply positive damage

Let TP, FP, and FN denote matched events, unmatched predictions, and unmatched references. Counts are pooled across windows before computing

$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \qquad R = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \qquad F_1 = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}. \tag{16}$$

Damage MAE averages $|\hat{d} - d|$ only over matched attacks with reliable damage labels. Predicted damage is the applied health loss, in the same units as the reference. Ambiguous attribution and unrelated health changes are excluded. Its valid match count is reported separately; missed attacks affect recall but do not enter this conditional MAE. Undefined ratios are reported as n/a.

C.2 Continuous State Rollouts

The protocol in Section 4.3 supplies recorded state and observations only at initialization. Predicted state and generated observations then condition successive chunks under recorded joint controls. The same eligible trajectories are used at 8, 32, and 64 steps, without intermediate reference-state resets.

At rollout step $r \in \{8, 32, 64\}$, let U_r contain unique coordinates affected by valid reference edits up to r whose reference category is available at r. Let J_r contain characters with valid reference health. Pooling these sets over trajectories, the state metrics are

$$\begin{aligned} \mathrm{EVA}(r) &= \frac{1}{|U_r|} \sum_{x \in U_r} \mathbb{I}[\hat{V}_r(x) = V_r^\star(x)], \\ \mathrm{HealthMAE}(r) &= \frac{1}{|J_r|} \sum_{j \in J_r} |\hat{h}_r^j - h_r^{j,\star}|. \end{aligned} \tag{17}$$

Coordinates and identities are indexed by trajectory when pooling. Repeated edits are scored against the current reference state; predicted unknown voxels are incorrect. Valid terminal health remains included. Empty evaluation sets yield n/a. Edited-Voxel Accuracy covers reference-edited coordinates and does not measure erroneous writes elsewhere.

C.3 Additional Multi-Player Visualizations

Figures 7 and 8 present additional first-person sequences in two-player and three-player settings, respectively. These examples illustrate two complementary aspects of multi-player interaction: how visible structures evolve over time and how different players observe the scene from their respective viewpoints. Within each episode, reading horizontally reveals changes in an individual player's view, while comparing rows highlights differences in viewing direction, distance, and occlusion.

Two-player sequences. Figure 7 shows six episodes featuring different block structures and surrounding environments. The two player views often reveal different parts of the evolving construction. For example, Episode 4 shows brick arrangements becoming more extended across the displayed frames, while Episode 5 presents a wooden structure from viewpoints that emphasize its upper surfaces and underside, respectively. These examples highlight that changes in image content arise from both structural modifications and shifts in viewpoint. Overlapping scene elements provide useful references for relating the two views, even when the structure occupies different image regions or is only partially visible.

Three-player sequences. Figure 8 extends the visualization to six three-player episodes. The examples include stone structures in Episodes 1 and 3, brick structures in Episodes 2 and 4, glass construction in Episode 5, and wooden construction in Episode 6. Player 0 and Player 1 frequently provide close-up views of the structures, whereas Player 2 often shows more of the surrounding environment and, in some frames, nearby players. This difference in visibility is important when interpreting multi-player sequences: a structural change that is prominent in one view may be distant, occluded, or outside the field of view in another. The additional perspective therefore complements the close-up views by exposing more of the spatial context in which the interaction takes place.

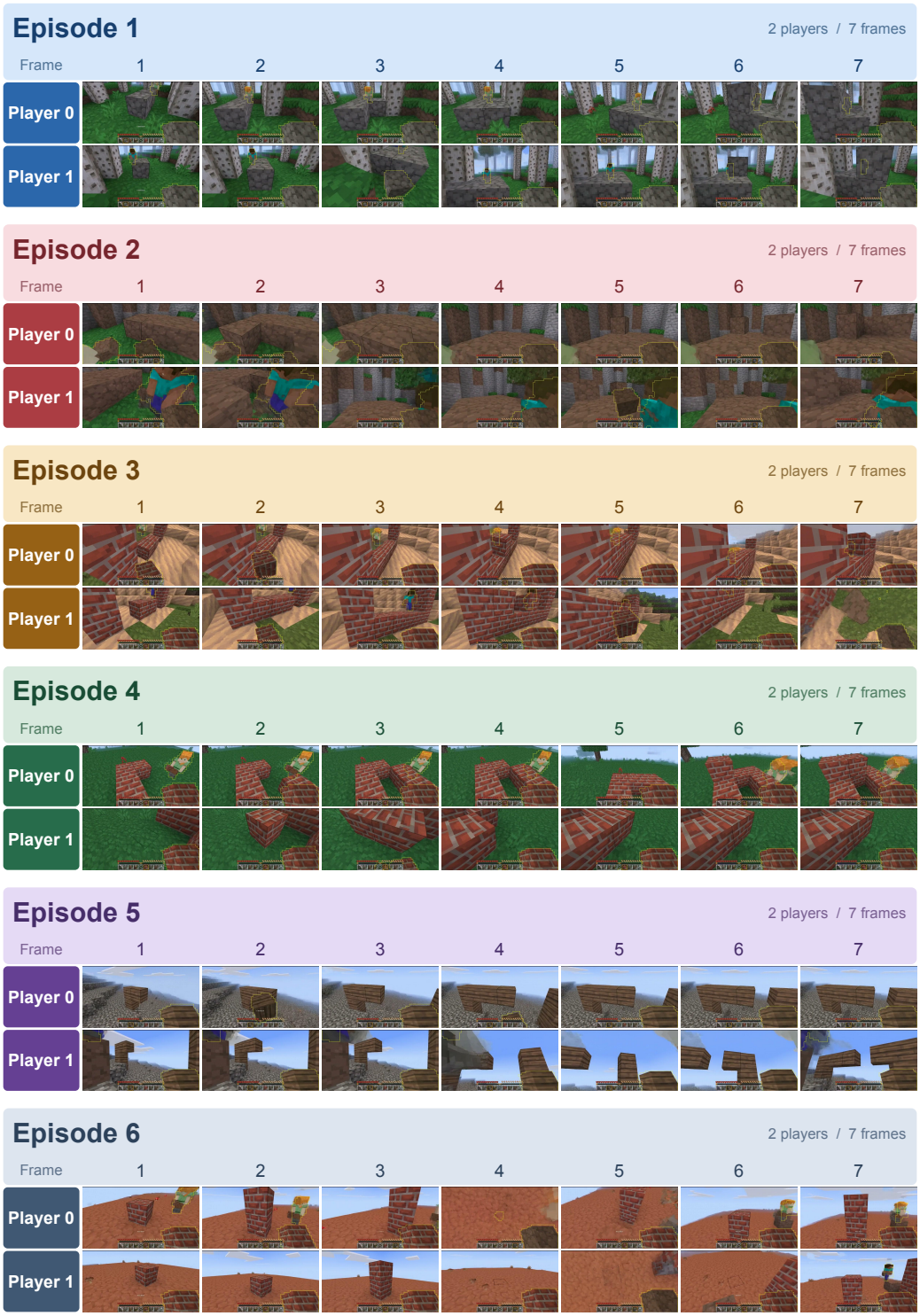


Figure 7: Additional two-player visualizations. Six episodes are shown, with each color-coded group corresponding to one episode. Within each episode, the two rows show the first-person views of Player 0 and Player 1, and the seven columns display selected frames in temporal order from left to right. The sequences illustrate evolving block structures from different viewpoints, including changes in visible geometry, viewing direction, and occlusion.

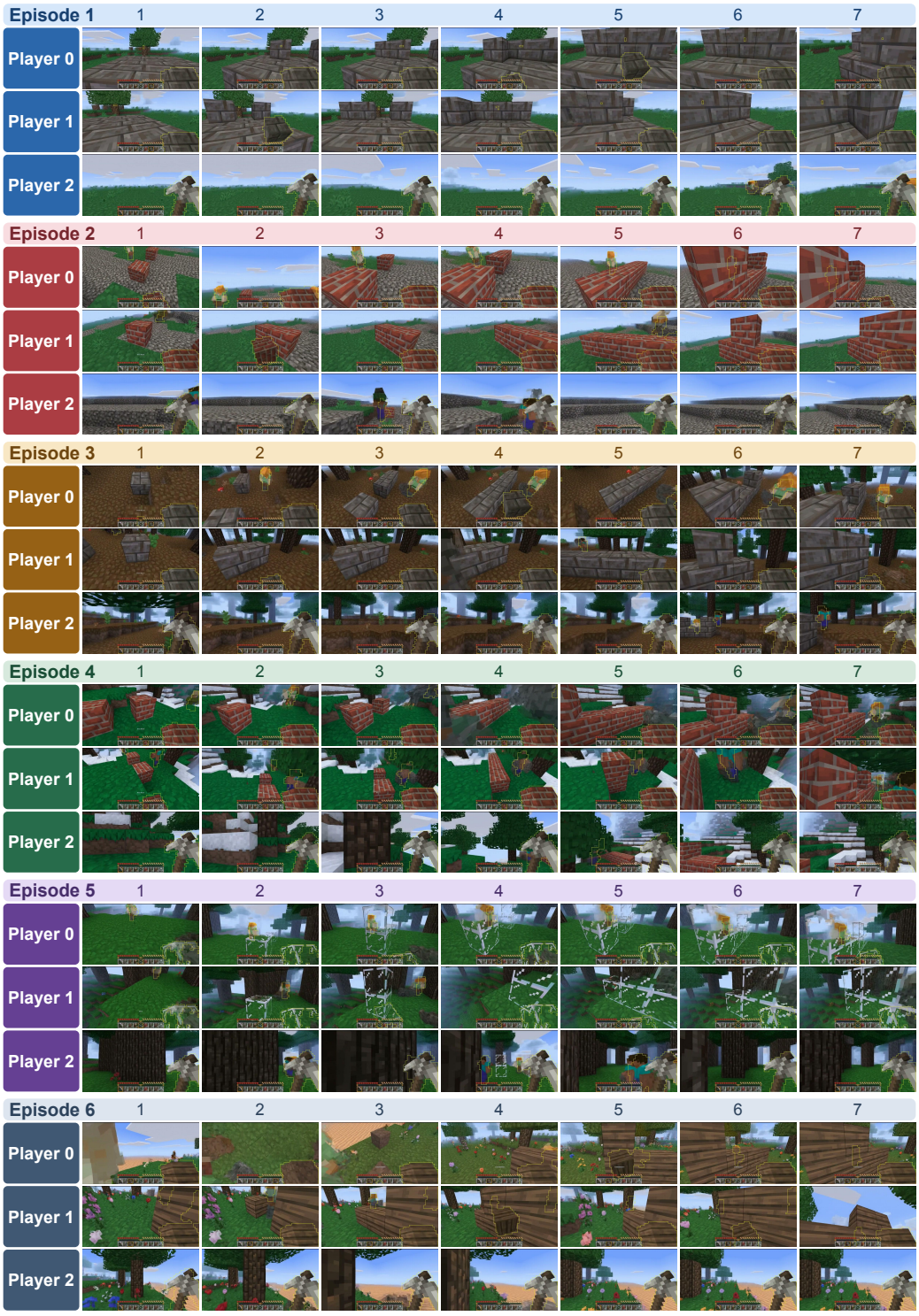


Figure 8: Additional three-player visualizations. Six episodes are shown, with each color-coded group corresponding to one episode. Within each episode, the rows show the first-person views of Player 0, Player 1, and Player 2 from top to bottom, and the seven columns display selected frames in temporal order from left to right. The examples cover stone, brick, glass, and wooden structures. The three views reveal differences in the visibility of the evolving structures, nearby players, and surrounding environment.