

# Reinforcement Learning Inside a Closed Game Platform: Combat NPCs Trained and Evaluated Entirely Within Roblox

Micah Zhou<sup>1</sup>, Weipeng Nie<sup>2</sup>

*Received August 14, 2026*

*Accepted September 27, 2026*

*Electronic access October 15, 2026*

Roblox is one of the largest game platforms. Creator code runs as sandboxed Luau with no GPU access and no first-party reinforcement-learning (RL) workflow, and external trainers require rate-limited HTTP bridges that creators build themselves. We are not aware of a previously reported study of deep-RL combat non-player characters (NPCs) trained and evaluated entirely inside Roblox and scored at the encounter level. The task is blocked line-of-sight combat: the NPC must route around cover, regain sight, enter sword range and attack. We compare two deep Q-network parameterizations of one masked six-by-two movement-attack action space, a flat-joint model and a factored dual-head model, trained by two-agent concurrent independent learning on the CPU inside Roblox Studio. Each architecture was trained six times from independent seeds, checkpointed at 30, 45, 60, 90 and 120 min, and evaluated on frozen encounter fixtures. The training run, not the pair episode, is the unit of analysis. Recovery varies enormously across runs of the same architecture: at 120 min six flat-joint runs span 0.0% to 79.3%. Neither architecture separates from the other at any checkpoint (exact permutation  $p \geq 0.56$ ). The 60-to-120-min collapse does not recur within either arm (paired  $p \geq 0.84$ ) or as an architecture-by-duration interaction (permutation  $p = 0.89$ ). A single training run therefore cannot support an architecture claim in this system. The durable contribution is the encounter-level protocol (pair episodes, start-condition labels, an active-attack success rule and a clean/excluded ledger), which exposes behavioral collapse that aggregate hit counts hide.

**Keywords:** Reinforcement learning, game AI, Roblox, deep Q-network, action factorization, evaluation methodology, NPC behavior

## Introduction

The standard research stacks for game-AI reinforcement learning (RL) are open environments with an external trainer: the Arcade Learning Environment<sup>1</sup>, ViZDoom<sup>2</sup>, Unity ML-Agents<sup>3</sup> and Godot<sup>4</sup> all expose the game to a Python training process, typically with GPU acceleration and parallel environments<sup>5,6</sup>. Commercial engines follow the same pattern when RL is used at all<sup>7-9</sup>, and practitioners report that building such pipelines is the dominant cost of using learned agents in production<sup>10</sup>.

Roblox, one of the largest game platforms in the world, sits outside this pattern. Creator code executes as sandboxed Luau inside the engine, scripts have no GPU access, and public Creator documentation describes no first-party workflow for training non-player character (NPC) policies with RL. An external trainer is nevertheless possible: HttpService lets a game server or Studio plugin call outside services, including a process on the same machine, within 500 requests per minute per server<sup>11</sup>, and plugins such as Rojo use this chan-

nel routinely<sup>12</sup>. A recent course report took that route for obstacle-course agents, with an external proximal policy optimization trainer behind a local Flask bridge and quantized weights hot-loaded back into Luau for inference<sup>13</sup>. The route is asynchronous, rate-limited and creator-built. The opposite choice, learning entirely inside the engine on the CPU, is also established in the community: pure-Luau neural-network libraries<sup>14</sup>, the DataPredict library with a self-learning sword-fighting example<sup>15</sup>, and hobbyist in-engine deep Q-network (DQN) projects for Snake<sup>16</sup> and sword-fighting self-play<sup>17,18</sup>, none of which reports an evaluation protocol or quantitative results. Roblox's own published agent research trains externally by behavior cloning<sup>19</sup>.

This paper takes the in-engine route as one implementation choice under Roblox's constraints. We are not aware of a previously reported study that trains and evaluates deep-RL combat NPCs entirely inside Roblox under a controlled protocol. The contribution of this paper is that study and its evaluation protocol, not a priority claim over the platform.

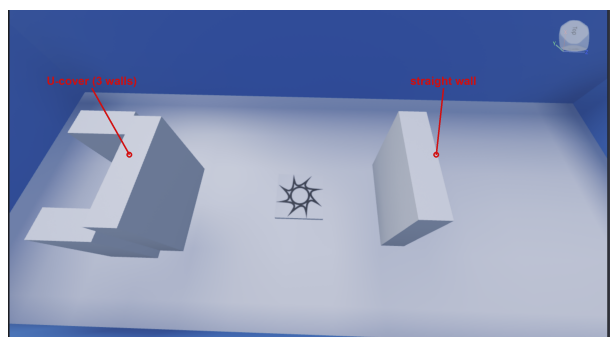
Four bodies of work frame the study. First, value-based deep RL: both policies are DQN agents in the lineage of Mnih et al.<sup>20</sup>, whose stabilizers, experience replay<sup>21</sup> and a target

<sup>1</sup> Dublin High School, Dublin, California, USA

<sup>2</sup> King's College London, London, United Kingdom

network, were extended by double Q-learning<sup>22</sup>, prioritized replay<sup>23</sup>, dueling decomposition<sup>24</sup> and Rainbow<sup>25</sup>. We reuse only the dueling decomposition and use a very small replay buffer and no target network, a configuration the literature on replay capacity<sup>26</sup> and on bootstrapped function approximation<sup>27</sup> identifies as fragile. Second, action factorization: the factored dual-head model is related to action branching<sup>28,29</sup> and parameterized-action methods<sup>30</sup>, which exist for large combinatorial action spaces. Ours is fully enumerable, so we study the factored dual-head model as an inductive bias rather than for scalability. Third, two-agent concurrent learning, whose non-stationarity interacts badly with replay<sup>31</sup> and in which mutual adaptation alone can shift behavior qualitatively<sup>32,33</sup>. Fourth, evaluation methodology: single training runs cannot support algorithm comparisons, and uncertainty must come from replication across runs<sup>34–39</sup>, while work on game agents argues that aggregate scores hide the behaviors designers care about<sup>40–43</sup>.

Deep RL for game NPCs is often presented as a scoring problem: train an agent, report a score, compare it to a baseline. In a game-engine combat loop, that framing can miss the behavior designers and players care about: a sword-fighting NPC must respect line of sight (LOS), move through geometry, avoid standing still and convert approach into attack. This case study describes a Roblox 1v1 sword-fighting project that exposed the problem. Figure 1 shows the arena, with a U-shaped cover and a separate straight-wall occluder. The blocked-start results below pool pair episodes from both occluders unless stated otherwise.



**Figure. 1 The 1v1 arena in Roblox Studio, with a three-walled U-shaped cover and a separate straight-wall occluder.** At a blocked start, a cover wall lies on the line of sight between the two NPCs, which must route around it to reacquire sight and attack.

When LOS is blocked at the start of a fight, a useful policy must move around cover, regain LOS, enter sword range, choose an attack and land a hit. This is a delayed-credit problem and an evaluation-design problem at once. A raw hit total over a fixed window mixes the first encounter with later deaths and respawns, and hides whether the agent that moved around

cover actually attacked or merely walked into range and was hit.

The lessons below are organized around one evaluation principle: learned NPC behavior should be scored one shared encounter at a time, a unit we call a pair episode, rather than through aggregate hit windows. The question becomes whether either NPC can recover from a blocked start and produce real combat. Answering it exposed, in a single training run, a factored dual-head model that recovered often at 60 min and degraded after continued training, a change a window-level hit count would have obscured. Six replicated runs per architecture later reproduced neither as a systematic effect.

## Methods

### Game environment

The setting is a 1v1 sword-fighting scene in Roblox Studio. The arena is a walled rectangle of roughly  $60 \times 62$  studs (the stud is Roblox's unit of length) containing the two solid, collidable cover structures shown in Figure 1. Each NPC carries a short-range sword. A swing that contacts the opponent costs it 10 health out of 100. At zero health, an NPC dies and respawns 20–30 studs from its opponent. There is no human player: combat timing, hit detection and health follow ordinary Roblox rules.

### AI system

Training and evaluation both run inside the Roblox game loop, in the Luau environment used to debug NPCs. The two NPCs in a run carry fixed identities, NPC-1 and NPC-2, each with its own network. NPC-1 is the side whose recovery is reported as the primary figure throughout, a convention fixed before any comparison was computed.

Each NPC receives the 18-dimensional observation of Table 1.

The comparison concerns action representation. Both models are DQN policies<sup>20</sup> over the identical executable action space in Table 2: six movement choices crossed with two attack choices, giving twelve joint actions (m, a). The flat-joint model emits one Q-value per joint action. The factored dual-head model emits a shared state value, six movement advantages and two attack advantages, and reconstructs each joint-action value additively, following the value/advantage decomposition of dueling networks<sup>24</sup>. The two networks are close in size, with 1,052 and 1,001 trainable parameters, so the comparison changes how Q-values are parameterized over actions, not the model scale.

A distance-zone mask restricts which of the twelve pairs are executable in a given state. Beyond 8 studs only ReEngage with NoAttack is allowed; between 4.8 and 8 studs only

**Table 1 The 18-dimensional observation vector.** Distances are in studs before normalization. Rows covering more than one index group together features that share a definition, range and normalization. *Relative* features depend on the opponent's position or the NPC's own heading. *Absolute* features depend only on the NPC or the arena.

| #     | Feature                               | Meaning                                                                                                    | Range and normalization                            | Absolute / relative    |
|-------|---------------------------------------|------------------------------------------------------------------------------------------------------------|----------------------------------------------------|------------------------|
| 1     | Distance                              | Horizontal (XZ) distance to the opponent                                                                   | [0, 1]; $\div 62$ (arena span), clipped            | Relative               |
| 2     | Health loss                           | 1 – own health / maximum health                                                                            | [0, 1]; ratio                                      | Absolute               |
| 3     | Facing                                | Dot product of own heading with the direction to the opponent                                              | [–1, 1]; unit vectors                              | Relative               |
| 4–5   | Direction x, z                        | Components of the unit XZ vector toward the opponent                                                       | [–1, 1]; unit vector                               | Relative (world axes)  |
| 6     | Under fire                            | Hit within the last 1.5 s                                                                                  | {0, 1}; binary                                     | Absolute               |
| 7     | LOS blocked                           | Ray to the opponent intersects scene geometry                                                              | {0, 1}; binary                                     | Relative               |
| 8     | Opponent health loss                  | 1 – opponent health / maximum health                                                                       | [0, 1]; ratio                                      | Absolute               |
| 9     | LOS-blocked duration                  | Consecutive frames with LOS blocked                                                                        | [0, 1]; $\div 60$ frames ( $\approx 1$ s), clipped | Relative               |
| 10    | Stuck                                 | Moved $< 0.1$ studs per frame for $\geq 10$ consecutive frames                                             | {0, 1}; binary                                     | Absolute               |
| 11–12 | In U-cover, near straight wall        | Inside the U-shaped cover; within the straight-wall occluder's region                                      | {0, 1}; binary                                     | Absolute               |
| 13–14 | Distance to U-cover, to straight wall | Distance to each cover center                                                                              | [0, 1]; $\div 30$ , clipped                        | Absolute               |
| 15–18 | Clearance left, right, front, back    | Distance to the nearest obstacle within 4 studs in each direction relative to the NPC's heading; 1 = clear | [0, 1]; $\div 4$                                   | Relative (own heading) |

MaintainRange with NoAttack; in both zones the two strafes with NoAttack become available while the NPC is stuck (observation 10) or a stuck-override window is open: if ReEngage beyond 8 studs closes the distance by less than 1 stud in 1.0 s, the strafes are allowed for 0.6 s, after which the test can fire again. Within 4.8 studs every movement except ReEngage is allowed, Hold with NoAttack always is, AttackIntent only with clear LOS at  $2.8 \leq d \leq 4.6$  and never with Retreat, and Retreat not beyond 12 studs (15 when health  $\leq 35\%$ ). The mask is identical for both models and applies to the chosen action and, in the per-decision updates, to the maximum over next-state actions in the bootstrap target, where the mask function reads the stored next-state observation together with the NPC's stuck-override state at the time of the update, so the override window can also change which next-state actions enter the maximum. Heading is controlled separately by a yaw controller that keeps the NPC facing its opponent.

Over the twelve joint actions  $(m, a)$ , the flat-joint model outputs a value per pair directly,  $Q_{\text{flat}}(s, m, a) = f_{\theta}(s)_{(m, a)}$ , while

the factored dual-head model shares one state value and adds centered movement and attack advantages:

$$Q_{\text{fac}}(s, m, a) = V(s) + (A_{\text{move}}(s, m) - \bar{A}_{\text{move}}) + (A_{\text{atk}}(s, a) - \bar{A}_{\text{atk}}),$$

where  $\bar{A}_{\text{move}}$  and  $\bar{A}_{\text{atk}}$  are the means over the six movement and two attack advantages. Both models score the same twelve joint actions. Only the parameterization differs. Supplementary Figure S1 diagrams the factored dual-head model.

The factored dual-head model is related to action-branching DQN<sup>28,29</sup>, which exists because compositional NPC decisions make flat joint action spaces grow multiplicatively. Here, the  $6 \times 2$  space is small enough to enumerate and the mask restricts it further, so we claim no scalability result. The matched setting instead asks how factoring movement and attack Q-values changes blocked-start recovery and failure modes when scalability pressure is removed.

The reward is the sum of the components in Table 3, accumulated over the frames of each 0.20 s decision.

**Table 2 The eight branch actions that compose the twelve joint actions.** A joint action is a pair (movement, attack). Movement is executed as one move-to target per decision, while attack intent is checked every frame. Here,  $d$  is the horizontal distance to the opponent in studs.

| Branch   | Action                   | Execution                                                                                   |
|----------|--------------------------|---------------------------------------------------------------------------------------------|
| Movement | Hold                     | Stay at the current position                                                                |
| Movement | ReEngage                 | Move to the point 7.5 studs from the opponent along the line between them                   |
| Movement | MaintainRange            | If $d$ is outside $[3.35, 4.45]$ , move to the point 4.0 studs from the opponent; else hold |
| Movement | Retreat                  | If $d < 10$ , move to the point 10 studs from the opponent; else hold                       |
| Movement | StrafeLeft / StrafeRight | Move 3 studs left or right, perpendicular to the direction to the opponent                  |
| Attack   | NoAttack                 | Never swing                                                                                 |
| Attack   | AttackIntent             | Swing whenever LOS is clear, $2.8 \leq d \leq 4.8$ , and $\geq 0.25$ s since the last swing |

The approach term has the potential-based form of Ng et al.<sup>44</sup>, a scaled difference  $0.99 \phi_t - \phi_{t-1}$  of a distance-and-LOS potential between consecutive frames. No component rewards regaining LOS directly. The blocked-start chain (route around cover, regain LOS, close to sword range, attack) is rewarded only through the approach-shaping term, passive penalty and delayed hit and kill rewards. Dying carries no explicit penalty beyond the accumulated damage penalties and the absence of bootstrapping at the terminal transition.

We do not compare against a hand-scripted NPC: scripted baselines have no standard strength, and the question here is representation and evaluation, not learned-versus-scripted combat outcomes.

### Training configuration

Both architectures were trained with an identical configuration. Only the output head differs. Each NPC has its own network, replay buffer and learning-rate schedule and learns only from its own transitions while its opponent learns concurrently (two-agent concurrent independent learning: no shared policy, no pool of frozen opponents). The learner uses one-step Q-learning with a multilayer perceptron. It uses a minimal form of experience replay: a ring buffer of 35 transitions per NPC from which 16 transitions (all of them while the buffer holds fewer) are drawn uniformly with replacement at each update and applied one at a time as separate gradient steps, with no target network: the bootstrap target  $r + \gamma \max_{a'} Q(s', a')$  is computed with the online network. There is no double Q-learning and no prioritized replay. These are therefore minimal DQN-style learners rather than implementations of the full DQN recipe<sup>20</sup>.

### Network

Each network maps the 18-dimensional observation through three hidden layers of 16 units to 12 joint-action values (flat-joint) or one state value plus six movement and two attack ad-

vantages (factored dual-head), giving 1,052 and 1,001 trainable parameters. The hyperbolic tangent is applied to every non-input layer including the output, so each network output lies in  $(-1, 1)$ : the twelve joint-action values of the flat-joint model directly, and for the factored dual-head model the state value and the eight advantages, whose centered sum can range over  $(-11/3, 11/3)$  because the centered movement term reaches  $\pm 5/3$  and the centered attack term  $\pm 1$ . Weights and biases are initialized uniformly in  $[-1.5, 1.5]$  from a per-NPC seed derived from the training seed (seed +  $1,000 \times \text{NPC index} + 1$ ). A second derived seed (+ 2) drives that NPC's spawn placement and exploration draws, and replay sampling uses the run's shared generator seeded by the training seed. Spawn timing and physics are not seeded, so a run is still not repeatable.

### Decision cycle

The simulation runs at the engine frame rate of about 60 Hz. Each NPC recomputes its observation every frame but selects a new joint action every 0.20 s (5 Hz), continuing to execute the last action in between while the per-frame rewards of Table 3 accumulate into that decision. Action selection is  $\epsilon$ -greedy with a constant  $\epsilon = 0.3$  during training, exploring uniformly over the currently allowed joint actions. Evaluation uses  $\epsilon = 0$  and the greedy action.

### Update

One update per decision: the new transition  $(s, a, R, s')$  is appended to the buffer, 16 transitions are sampled, and for each the temporal-difference error  $\delta = R + \gamma \max_{a'} Q(s', a') - Q(s, a)$  is clipped to  $[-10, 10]$  (Huber-type, threshold 10) and back-propagated by plain stochastic gradient descent without momentum, in draw order and without averaging across the batch, with every weight gradient clipped element-wise to  $[-1, 1]$ . The discount is  $\gamma = 0.99$  per decision step, an effective horizon of about 20 s. The learning rate starts at  $10^{-6}$  and is multiplied by 1.00015 per simulation frame to

**Table 3 Reward components.** Per-frame components are evaluated at every simulation frame (about 60 per second) and summed into the reward of the current decision. The two shaping terms are applied during training only. “Combat event” means a hit landed or taken by this NPC.

| Component              | Trigger                                                                                  | Value                                                                                                                                                                                          | Rate      |
|------------------------|------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|
| Hit landed             | Own sword damages the opponent                                                           | $+0.4 \times \text{damage}$ ( $= +4.0$ per 10-damage slash)                                                                                                                                    | Per hit   |
| Damage taken           | Hit by the opponent                                                                      | $-0.4 \times \text{damage}$ ( $= -4.0$ )                                                                                                                                                       | Per hit   |
| Exposed when hit       | Hit while not inside cover                                                               | $-0.05$                                                                                                                                                                                        | Per hit   |
| Kill                   | Opponent’s health reaches zero from own hit                                              | $+70$                                                                                                                                                                                          | Per kill  |
| Swing cost             | Sword swung                                                                              | $-0.05$                                                                                                                                                                                        | Per swing |
| Passive penalty        | $t > 12$ s since the last combat event (until the 90 s stalemate cutoff)                 | $-\min(1.5, 0.08(t - 12))$ per frame; at most $-400$ per training episode                                                                                                                      | Per frame |
| Cover bonus            | Inside cover and $< 5$ s since the last combat event                                     | $+0.02$                                                                                                                                                                                        | Per frame |
| Shaping, training only | Under fire: within 1.5 s of being hit, outside cover. Approach: $8 \leq d \leq 20$ studs | Under fire: $-0.02$ for Hold, $+0.01$ for Retreat or either strafe. Approach: $0.12(0.99 \phi_t - \phi_{t-1})$ , $\phi = (20 - d)/12 \times (1 \text{ if LOS clear, } 0.7 \text{ if blocked})$ | Per frame |

a ceiling of  $3 \times 10^{-4}$ , reached after about 10.6 min. It is not stored in the checkpoint and restarts from  $10^{-6}$  when training is continued. For the factored dual-head model the gradient of the reconstructed joint value is 1 for the value output,  $1[m' = m] - 1/6$  for movement advantage  $m'$  and  $1[a' = a] - 1/2$  for attack advantage  $a'$ , the centered dueling form.

#### Training episodes and terminal states

A training episode is the lifetime of one NPC: it begins at a random spawn 20–30 studs from its opponent and ends when the NPC dies or when 90 s pass without either NPC landing a hit. In both cases, the final transition is learned with the reward alone as target, so the stalemate cutoff is treated as terminal. An opponent’s death does not end the training episode. The NPC receives the kill reward and continues against the respawned opponent.

#### Transitions from engine events

Observations are read from the engine every frame: positions, a ray cast between the NPCs for LOS, four short ray casts for nearby geometry, and both NPCs’ health. Combat events come from attributes set by the sword script on contact (10 damage per hit; the swing rule is in Table 2). A transition closes at the next decision boundary with the accumulated reward and observation at that boundary. Pseudocode is provided in Supplementary Algorithm S1.

#### Horizon, checkpoints and seeds

The training horizon is in-engine wall-clock time, not a fixed number of updates. The original study trained each architecture about 60 min from random initialization and continued training from each checkpoint for about 60 min more. The replication trains each run continuously for 120 min. Weights are saved every 300 s and a checkpoint is the most recent save. Each replicated run also logs its cumulative learner calls every 60 s (each applying up to 16 gradient steps) and its environment steps and completed training episodes, as accumulated at episode ends. The values recorded at each checkpoint are in Supplementary Table S8. The original 60-min runs used training seed 60100 for both architectures and the 120-min factored continuation was clock-seeded. The configuration file for the flat-joint continuation was not committed and its raw log not retained, so its seed is unrecorded. Because spawning is asynchronous and the physics is not bit-reproducible, a fixed seed does not make a run repeatable, which is why the replicated matrix (Results) uses six runs per architecture, seeds 70001 to 70006. A diagnostic mode that forces lateral movement while LOS is blocked was disabled in every run reported here.

#### Availability

The training loop is given as pseudocode in Supplementary Algorithm S1 and the training configuration is described above. The evaluation ledgers and parsed pair-episode records of the replicated runs, the parsed pair-level ledger of the four original checkpoints, the per-run training counters of Supplementary Table S8, and the preregistrations, results, ledgers and

---

weight snapshots of the configuration screen described under Limitations are available from the corresponding author on reasonable request.

### Evaluation protocol

Three units are used throughout. An *evaluation window* is one bounded 600 s run of frozen policies (no learning,  $\epsilon = 0$ , greedy actions) under one evaluation seed. Inside it, the NPCs fight repeatedly, respawning after each death, so a raw hit count over the window sums across many separate fights. A *fixture* is a fixed pair of spawn positions for NPC-1 and NPC-2. The bank holds 24 fixtures, nominally 20 blocked (11 behind the straight wall, 9 behind the U-shaped cover) and 4 clear, replayed in a fixed cyclic order, one fixture per pair episode. The labels are nominal, spawn coordinates are clamped by the engine, and every start condition is classified by the LOS ray at the instant the pair episode begins. Every window of the replicated, fixed-opponent and legacy re-evaluations (Results) replays this bank under evaluation seeds 61000, 61001 and 61002 (one per window). The seed sets only the starting point in the cycle. The windows of the four original checkpoints predate the bank and respawned each NPC independently. A *pair episode* is one shared encounter: it begins when both NPCs are alive at a fixture and ends when either dies or when 90 s pass without a hit by either NPC (the stalemate cutoff). In both cases, both respawn at the next fixture and a new pair episode begins. How a pair episode ended (death, cutoff, or still open at window end) is recorded separately from whether either NPC recovered in it. Every recovery rate in this paper is computed over the pair episodes of one run. The conventional combat metrics reported in Results have their own denominators, and across-run statistics are then computed over runs. A *training episode*, used only in the training description above, is the lifetime of one NPC during training.

For each pair episode, we record its start condition at the instant it begins. A *blocked start* is a pair episode at whose start neither NPC has LOS to the other, because a cover wall lies on the line between them. A *clear start* is one in which LOS is clear. Supplementary Figure S2 shows a blocked start and a recovered engagement for each of the two occluders.

In a blocked start, an NPC *recovers* if it completes the whole chain: moves around the cover, regains LOS, enters sword range and lands a hit. Operationally, a recovery is recorded when the parser finds, within the pair episode, clear LOS, entry into sword range and at least one own hit. Supplementary Table S3 reports how far along the chain each pair episode got. Recovery is reported per side, and NPC-1 recovery is the primary figure throughout. Two further figures are retained for continuity: *any-side recovery*, the share of blocked-start pair episodes in which at least one NPC recovers, and the stricter *both-side recovery*, in which both do. In self-play, both NPCs

run the same checkpoint of one training run, each with its own network: the two networks of a run share configuration, seed and training course but not weights, so an any-side rate counts a recovery by either of the run's two networks and is systematically higher than either side's own rate. It is also not comparable with the fixed-opponent condition, where the two sides come from different training runs. We fixed this idiom after the first architecture's replicated data were complete and before any comparison between architectures. The sampling target for window counts had been set on the any-side figure and was not revised.

When a blocked-start pair episode is not a recovery, we record why. The main navigation failure is *range-without-LOS*: the NPC reaches sword-range distance while still occluded and never reacquires LOS. The second is *navigation-without-attack*: it reaches clear LOS and range but records no swing of its own before the pair episode ends. The remaining no-recovery, stall, near-cover and swing-without-hit cases form an *other* class. The split says whether a dropped rate means the agent stopped getting around cover or stopped swinging, two problems with different fixes.

Not every pair episode enters the headline counts. A pair episode is *excluded* if it comes from an invalid evaluation session (a launch of the evaluation harness that failed to build; a completed session produces one window), from a short window (60 or 300 s, from early testing), from an overrun window (far past 600 s), from a restart-contaminated session (overlapping a Studio restart), or, where the rule was applied, if it is a flagged respawn-boundary artifact with duration under 1 s. All remaining pair episodes are *clean*. A pair episode still open when its window ends is not excluded: it stays in the denominator and is scored by what was observed before the cutoff, so an open pair episode can already count as a recovery. The number excluded in each category is reported with the original trajectory in Results, and excluded pair episodes remain in the retained ledger.

In addition to self-play, in which both NPCs run the checkpoint under test, a fixed-opponent condition pairs each run against a different, independently trained run of the same architecture at the same checkpoint, so that recovery of the evaluated side does not depend on a co-trained opponent.

### Data analysis

The training run, not the pair episode, is the unit of analysis for any comparison between architectures or checkpoints. Pair episodes pooled within one run estimate that run's rate. They do not estimate between-run variation, and no interval or test is computed as if they were independent<sup>34,37</sup>. For the original single-run trajectory, we report counts and rates only. For the replicated runs, we report each run's NPC-1 recovery rate (per-run counts and the any-side and both-side figures are in

Supplementary Table S7), the across-run mean and median, and a 95% bootstrap percentile interval resampling runs. With six runs, the resample space ( $6^6 = 46,656$ ) is enumerated exhaustively, so the intervals carry no Monte Carlo error, with endpoints taken at  $\lfloor 0.025B \rfloor$  and  $\lceil 0.975B \rceil - 1$  without interpolation. Comparisons are reported as mean differences in percentage points, using exact tests matched to their design.

## Results

### The original observed trajectory

Table 4 reports the clean blocked-start pair episodes of the original four checkpoints, one training run per architecture. These rows are one observed trajectory, not a benchmark: each architecture is a single run and the 60-min rows use overlapping but not identical evaluation seeds, so the 51/102 (50.0%) against 36/107 (33.6%) gap in NPC-1 recovery is descriptive. Per-side counts were recovered by joining the clean ledger to the per-seed pair records, with all 410 pair episodes joined without loss.

The 60-min factored dual-head row shows why the three figures are reported separately: its any-side rate of 70/102 (68.6%) exceeds its NPC-1 rate of 51/102 (50.0%) because 19 additional pair episodes were completed by NPC-2 alone, and the same cell is the most one-sided in the conventional combat metrics reported below, where NPC-2 landed 616 of its 620 swings and NPC-1 83 of 86.

In this trajectory, the 60-min factored dual-head checkpoint recovered more often on either figure, and after continued training to 120 min, its NPC-1 recovery fell from 51/102 (50.0%) to 9/100 (9.0%), a drop of 41.0 percentage points. Under the any-side convention of the original report, those figures were 68.6% and 13.0%, a drop of 55.6 points. Reporting one side reduces the magnitude without changing the direction and affects only the factored dual-head rows. Supplementary Figure S3 (drawn in the any-side idiom of the original report) shows where the chain stopped: for the factored dual-head model, the dominant 120-min failure class is

range-without-LOS, rising from 14 to 55 pair episodes, with navigation-without-attack rising from 4 to 22. The flat-joint rows did not show the same drop, but they are single runs and not seed-matched, so this is descriptive only. Table 5 gives the exclusion accounting.

The exclusion fraction differs strongly between checkpoints, almost entirely because one 60-min flat-joint window ran for more than eight hours and was excluded in full (489 blocked starts). Two inconsistencies of the original ledger are disclosed rather than repaired: the sub-second respawn-boundary rule was applied only to the 120-min factored dual-head checkpoint (if applied uniformly, it gives denominators of 95, 99, 97 and 100 and any-side recovery of 37.9%, 70.7%, 35.1% and 13.0%, without changing the direction of any comparison), and the short-window filter raises the 60-min flat-joint any-side rate from 31.1% to the 33.6% reported. The pair counts of the two excluded restart-contaminated sessions cannot be recovered.

Across the 180 evaluation windows of the replicated matrix, the parser produced 2,838 pair episodes, of which 1,244 were clear starts by the LOS ray (749 from nominally clear fixtures and 495 from nominally blocked fixtures whose start LOS was in fact clear) and therefore out of scope, leaving 1,594 blocked starts. None was excluded: by construction, the fixture bank removes the respawn-boundary fragments excluded by hand above, and 27 evaluation windows that aborted during collection were re-run in full rather than partially included (Supplementary Table S1).

### Replicated training runs

The replication trains each architecture six times from independent seeds, checkpointing each run at 30, 45, 60, 90 and 120 min, so a training course is a within-run trajectory. Every checkpoint is evaluated on the same frozen fixture sequence. The unit of analysis is the training run, whose pooled pair episodes give one observation. NPC-1's recovery is reported (per-run any-side and both-side figures are in Supplementary Table S7). Table 6 reports the flat-joint arm and Table 7 the

**Table 4 Blocked-start recovery of the original four checkpoints (one training run per architecture).** Rows pool clean blocked-start pair episodes from several evaluation windows and are descriptive. They are not seed-matched between architectures. Counts are pair episodes. NPC-1 recovery counts the pair episodes in which NPC-1 completed the chain; any-side, those in which at least one NPC did; both-side, those in which both did.

| Checkpoint                 | Clean blocked-start pair episodes | NPC-1 recovery | Any-side recovery | Both-side recovery |
|----------------------------|-----------------------------------|----------------|-------------------|--------------------|
| 60-min flat-joint          | 107                               | 36/107 (33.6%) | 36/107 (33.6%)    | 30/107 (28.0%)     |
| 60-min factored dual-head  | 102                               | 51/102 (50.0%) | 70/102 (68.6%)    | 51/102 (50.0%)     |
| 120-min flat-joint         | 101                               | 34/101 (33.7%) | 34/101 (33.7%)    | 30/101 (29.7%)     |
| 120-min factored dual-head | 100                               | 9/100 (9.0%)   | 13/100 (13.0%)    | 7/100 (7.0%)       |

**Table 5 Accounting of pair episodes and exclusions for the original four checkpoints.** Recomputed from the parsed pair-level ledger. The raw server logs of these runs were not retained. Clear-start pair episodes are out of scope rather than excluded, and pair episodes still open at window end stay in the clean denominator, scored by what was observed before the cutoff. Supplementary Table S1 gives the same accounting for the replicated runs.

|                                              | <b>60-min flat-joint</b>                  | <b>60-min factored dual-head</b> | <b>120-min flat-joint</b>           | <b>120-min factored dual-head</b> |
|----------------------------------------------|-------------------------------------------|----------------------------------|-------------------------------------|-----------------------------------|
| Evaluation windows parsed                    | 22                                        | 8                                | 11 (+2 excluded sessions, unparsed) | 23                                |
| Pair episodes parsed                         | 788                                       | 133                              | 131                                 | 188                               |
| Clear-start pair episodes (out of scope)     | 177                                       | 31                               | 30                                  | 66                                |
| Blocked-start pair episodes before exclusion | 611                                       | 102                              | 101                                 | 122                               |
| Excluded: invalid session (failed build)     | 1 session (0 pair episodes)               | 0                                | 0                                   | 0                                 |
| Excluded: short windows (60 s / 300 s)       | 6 windows (15 blocked starts)             | 0                                | 0                                   | 0                                 |
| Excluded: overrun window (ran 8 h 14 min)    | 1 window (489 blocked starts)             | 0                                | 0                                   | 0                                 |
| Excluded: restart-contaminated sessions      | 0                                         | 0                                | 2 sessions (count unrecoverable)    | 0                                 |
| Excluded: flagged sub-second rows (< 1 s)    | 0 (rule not applied; 12 such rows remain) | 0 (not applied; 3 remain)        | 0 (not applied; 4 remain)           | 25 rows (22 blocked starts)       |
| Clean blocked-start pair episodes (Table 4)  | 107                                       | 102                              | 101                                 | 100                               |
| Open at window end (kept in denominator)     | 9                                         | 6                                | 11                                  | 15                                |
| Excluded fraction of blocked starts          | 504/611 (82.5%)                           | 0/102 (0%)                       | 0/101 (+2 sessions uncounted)       | 22/122 (18.0%)                    |

factored dual-head arm. The two arms are independent samples of six runs and are never pooled.

Two features of Table 6 (Figure 2A) matter more than the mean row. At every checkpoint, the six runs span at least 34 percentage points and, at three of the five, more than 60. At 120 min, one run recovers in 79.3% of its blocked starts and another in none. No run is consistently the strongest and runs cross each other repeatedly. Against that spread, the mean trajectory is shallow, from 51.0% at 30 min to 62.4% at 90 and 55.8% at 120, about 11 points, while individual runs move by 40 or 50 points between adjacent checkpoints. A single run at a single checkpoint is therefore drawn from a distribution about as wide as the difference reported in the original comparison. The replicated arm neither confirms nor refutes the rows of Table 4, but shows that their sampling error was never estimated and is large.

Failure composition is more stable than failure frequency

(Figure 2B). Pooled within each checkpoint of the flat-joint arm, range-without-LOS leads at all five, accounting for 13% to 36% of all blocked-start pair episodes and 49% to 76% of the failures. Navigation-without-attack, stalling near cover and swing-without-hit share the remainder. Composition is pooled over pair episodes and is descriptive only. Run 70005 at 120 min is the extreme case of the same pattern rather than a different one: 12 of its 12 clean blocked-start pair episodes end in range-without-LOS, with a median encounter length of 90.0 s against 13.2 s at its own 90-min checkpoint. It swings and hits when it has LOS. What it has lost by 120 min is specifically the step that regains sight.

The factored dual-head arm reproduces the shape of the flat-joint arm rather than the trajectory of Table 4: its mean rises from 42.8% at 30 min to 63.4% at 90 and falls to 50.4% at 120, again about 20 points against per-run swings several times larger, and three of its thirty cells are at 0.0% (run 70004

**Table 6 Blocked-start recovery of the flat-joint architecture across six independent training runs.** Each cell pools three 600 s evaluation windows on the frozen fixture sequence. Rates are NPC-1 recoveries over clean blocked-start pair episodes. Intervals are the 95% exhaustive bootstrap percentile intervals over runs defined under Data analysis.

| Training seed           | 30 min       | 45 min       | 60 min       | 90 min       | 120 min      |
|-------------------------|--------------|--------------|--------------|--------------|--------------|
| 70001                   | 45.0%        | 64.7%        | 73.0%        | 98.2%        | 72.2%        |
| 70002                   | 31.8%        | 30.8%        | 23.8%        | 68.8%        | 76.6%        |
| 70003                   | 72.7%        | 47.4%        | 84.0%        | 53.8%        | 42.9%        |
| 70004                   | 72.7%        | 55.2%        | 24.1%        | 33.3%        | 63.6%        |
| 70005                   | 22.7%        | 57.9%        | 50.0%        | 46.7%        | 0.0%         |
| 70006                   | 61.1%        | 65.0%        | 93.0%        | 73.8%        | 79.3%        |
| <b>Mean across runs</b> | <b>51.0%</b> | <b>53.5%</b> | <b>58.0%</b> | <b>62.4%</b> | <b>55.8%</b> |
| Median across runs      | 53.1%        | 56.5%        | 61.5%        | 61.3%        | 67.9%        |
| 95% interval over runs  | [35.6, 66.2] | [43.3, 62.0] | [36.4, 79.5] | [46.4, 79.4] | [31.8, 74.6] |
| Range across runs       | 50.0 pp      | 34.2 pp      | 69.2 pp      | 64.8 pp      | 79.3 pp      |

at 30 and 120 min, run 70001 at 45), while the same runs recover in 41% to 79% of their blocked starts at the adjacent checkpoint. The 60-to-120-min collapse of the original run does recur here, but as an event of individual runs: runs 70004, 70006 and 70003 fall over that interval, from 71.4% to 0.0%, 63.6% to 38.9% and 42.1% to 17.6%, while runs 70001, 70002 and 70005 rise by 20 to 57 points. Failure composition differs from the flat-joint arm at the earliest checkpoint only: at 30 min, 46% of this arm's pair episodes (87 of 191) end in navigation-without-attack against 5% for the flat-joint arm, and from 45 min onward range-without-LOS leads in both, at 16% to 28% of pair episodes here and 13% to 36% there.

#### Comparisons within and between runs

Three questions follow, each answered with an exact test matched to its design: whether continued training changes recovery within a run, whether architecture changes it at a

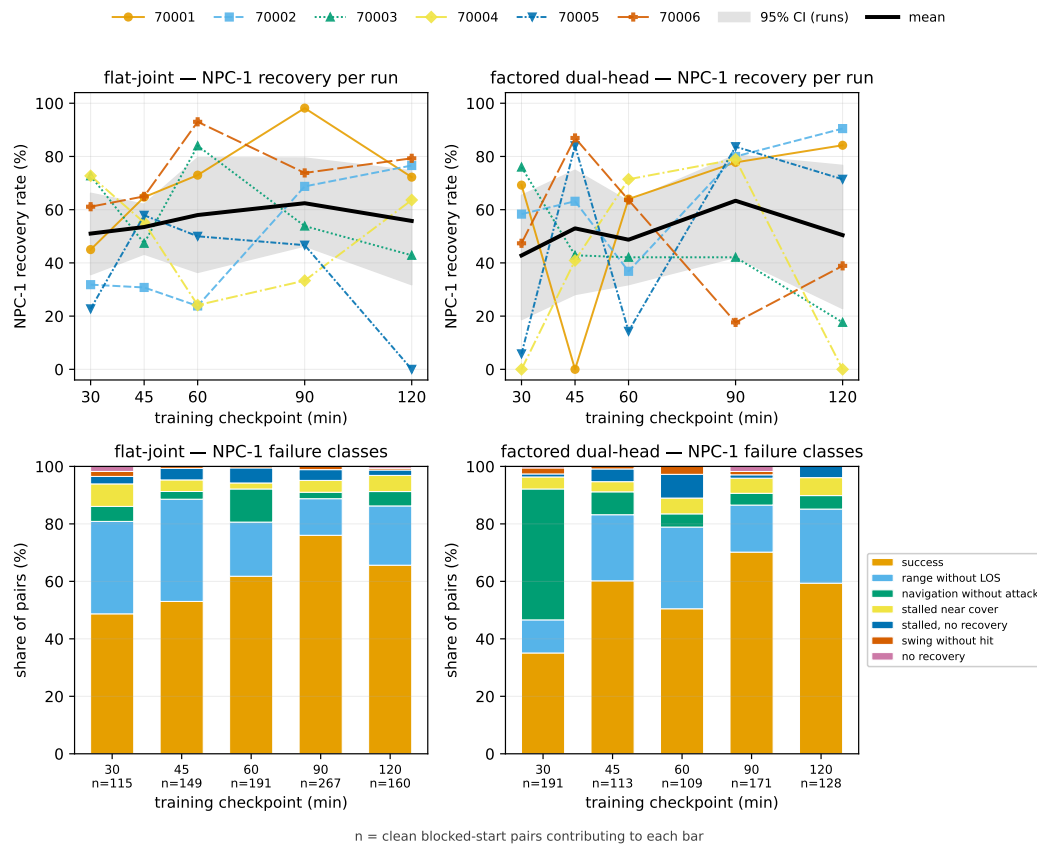
given checkpoint, and whether the change with training differs between architectures, the form of the original observation. Within-run changes use a paired sign-flip test over the six differences. The architecture contrast uses an exact label permutation over the twelve independent runs. The smallest attainable two-sided p is  $2/64 = 0.031$  for the paired test and  $2/924 = 0.002$  for the permutation test. Six seeds per arm were chosen so the paired test could reach the conventional level, which five cannot.

None of the three questions receives a positive answer. Within runs, the 60-to-120-min change in NPC-1 recovery averages  $-2.2$  percentage points for the flat-joint arm (per-run changes of  $-50.0$ ,  $-41.1$ ,  $-13.7$ ,  $-0.8$ ,  $+39.5$  and  $+52.8$  points; sign-flip  $p = 0.84$ ) and  $+1.7$  points for the factored dual-head arm ( $-71.4$ ,  $-24.7$ ,  $-24.5$ ,  $+20.2$ ,  $+53.6$  and  $+57.1$ ;  $p = 0.94$ ). No adjacent-checkpoint transition in either arm reaches  $p < 0.37$ , and the end-to-end 30-to-120-min change is  $+4.8$  points for the flat-joint arm ( $p = 0.72$ ) and  $+7.7$  for the

**Table 7 Blocked-start recovery of the factored dual-head architecture across six independent training runs.** Same protocol, denominators and interval convention as Table 6. Rates are NPC-1 recoveries over clean blocked-start pair episodes, three 600 s windows per cell.

| Training seed           | 30 min       | 45 min       | 60 min       | 90 min       | 120 min      |
|-------------------------|--------------|--------------|--------------|--------------|--------------|
| 70001                   | 69.2%        | 0.0%         | 64.0%        | 77.8%        | 84.2%        |
| 70002                   | 58.3%        | 63.2%        | 36.8%        | 80.0%        | 90.5%        |
| 70003                   | 76.0%        | 42.9%        | 42.1%        | 42.1%        | 17.6%        |
| 70004                   | 0.0%         | 40.9%        | 71.4%        | 78.9%        | 0.0%         |
| 70005                   | 5.8%         | 84.0%        | 14.3%        | 83.7%        | 71.4%        |
| 70006                   | 47.4%        | 87.0%        | 63.6%        | 17.6%        | 38.9%        |
| <b>Mean across runs</b> | <b>42.8%</b> | <b>53.0%</b> | <b>48.7%</b> | <b>63.4%</b> | <b>50.4%</b> |
| Median across runs      | 52.9%        | 53.0%        | 52.9%        | 78.4%        | 55.2%        |
| 95% interval over runs  | [18.7, 65.3] | [28.1, 74.8] | [31.8, 63.1] | [42.1, 80.5] | [22.9, 76.6] |
| Range across runs       | 76.0 pp      | 87.0 pp      | 57.1 pp      | 66.1 pp      | 90.5 pp      |

## Recovery from a blocked start: per-run trajectories and where the chain stops



**Figure. 2 Blocked-start recovery along the training course, and the composition of the failures.** A) NPC-1 recovery of each of the six independent training runs of both architectures at the 30-, 45-, 60-, 90- and 120-min checkpoints, with the across-run mean and its exact bootstrap interval. B) Failure-class composition per checkpoint, as a share of clean blocked-start pair episodes.

factored dual-head arm ( $p = 0.69$ ). Between architectures, the difference in mean recovery (factored dual-head minus flat-joint) is  $-8.2, -0.5, -9.3, +0.9$  and  $-5.3$  percentage points at the five checkpoints, with permutation  $p$ -values of  $0.60, 0.98, 0.56, 0.95$  and  $0.80$ . At no checkpoint is either architecture separable from the other. The interaction that would correspond to the original observation, a 60-to-120-min change that differs between architectures, is estimated at  $+3.9$  points with a permutation  $p$  of  $0.89$  over the twelve within-run changes.

These are not null results in the sense of small effects precisely estimated: the within-run changes have standard deviations of about 40 percentage points in the flat-joint arm and 50 in the factored dual-head arm (41.7 and 50.7 with the  $n - 1$  divisor), and the exhaustive bootstrap intervals for the differences of run means (Supplementary Table S4) are correspondingly wide. The architecture difference at 120 min is  $-5.3$  points with a 95% interval of  $[-39.6, +30.3]$ . The 60-to-120-min within-run change is  $-2.2$  points  $[-32.1, +28.4]$  for

the flat-joint arm and  $+1.7$  points  $[-34.9, +36.8]$  for the factored dual-head arm. The interaction is  $+3.9$  points  $[-44.5, +51.0]$ . What the tests and intervals establish is that the data are consistent with no systematic effect of training duration or architecture and that the runs disagree widely. An interval that crosses zero is not evidence of equivalence. The picture that replaces the original trajectory is instability of run-level outcomes: factored dual-head runs that collapse to zero recovery at 30 or 45 min (0 of 10 pair episodes in each case) recover by the next checkpoint, one run of each architecture ends the course at zero at 120 min (0 of 12 and 0 of 13) with no later checkpoint to observe, and a single run at a single checkpoint carries less information about the architecture than about the run.

### Legacy checkpoints under the new protocol

The four original checkpoints were re-evaluated on the fixture bank with the same harness as the replicated arms, so a gap

---

against the figure originally reported reflects the protocol and evaluation sampling rather than a change in the checkpoint's capability. The gaps are large and run in both directions (Table 8).

Two cautions attach to this table. The original windows ran 620.6 s against 598.8 to 600.9 s (mean 599.7) for the twelve bridge windows here, and the fixture bank changes how many encounters a window yields, so only the rates should be read across, not the counts. The spread across the three evaluation seeds in each cell (57.1–71.4%, 50.0–62.5%, 0.0–33.3%, 0.0–0.0%) is evaluation noise for one training run, not a run-level interval, and must not be read beside those of Tables 6 and 7. Under the revision protocol, both legacy runs also recover less at 120 min than at 60 min (65.0% to 11.1% and 55.6% to 0.0%). Each is one training run scored over 10 to 27 pair episodes, so these are two further single-run trajectories of the kind seen among the replicated runs, not independent evidence of a systematic decline.

### Fixed-opponent evaluation

Self-play recovery describes the two networks of one training run against each other. Table 9 reports the same checkpoints against a different, independently trained run of the same architecture at the same checkpoint, arranged as a ring over the six runs (70001 against 70002, and so on to 70006 against 70001), so that each evaluated run meets one foreign opponent and each run serves once as the evaluated side and once as the opponent. Each pairing uses three 600 s windows, with recovery reported for the run named first.

Against a foreign opponent, the means move by 2 to 12 points relative to self-play, and every cross-play mean falls inside the self-play interval of the same cell. This is a descriptive observation, not a test that the two conditions share a level. The spread across pairings is also wide: two runs of the same architecture, at the same checkpoint, recover in 88.6% and 13.9% of their blocked starts against their respective opponents (flat-joint run 70005 against 70006, and run 70004 against 70005, at 60 min), so a fixed-opponent evaluation built on a single reference policy would itself have been a single draw<sup>45</sup>. The fixed-opponent condition therefore adds no precision: recovery cannot be separated from the pairing, and both conditions show wide dispersion whose sources the design cannot decompose. Table 10 adds the conventional combat metrics.

At the 120-min factored dual-head checkpoint, the hit rate still looks healthy (24/29, 82.8%) while the volume of combat has collapsed: 29 swings across 100 blocked-start pair episodes, 76 of 85 ended pair episodes closing with no kill at the stalemate cutoff, and NPC-1 recovery of 9/100 (9.0%). A dashboard tracking hit rate alone would have missed it.

The same metrics for the replicated runs are in Supplemen-

tary Table S2 (where no-kill also includes pair episodes still open at the window cutoff), pooled over the six runs of each architecture and descriptive only. The hit rate there is high and nearly flat across all ten cells, 84.0% to 95.5%, while pooled recovery moves between 35.1% and 76.0% and median encounter length between 10.6 and 90.0 s.

The chain shows where recovery is lost (Supplementary Table S3). Reaching sword range is almost never the obstacle (86.2% to 94.2% of blocked-start pair episodes in every cell). What separates the cells is regaining LOS (57.7% to 82.7%) and swinging (37.2% to 77.2%). The 30-min factored dual-head checkpoint is the clearest case: 82.7% regain LOS and 94.2% reach range, the highest of any cell, yet only 37.2% ever swing. In every cell, the share landing a hit is within 3 points of the share swinging, so the failures are failures to regain sight or to attack, not of the attack itself.

## Discussion

### Lessons for game-AI practice

Three lessons concern the protocol. The first is to make the pair episode the scoring unit. The second is to record each pair episode's start condition and failure class, which separates blocked-start recovery from clear-start combat and failure to route from failure to attack. The third is to keep headline and excluded evidence separate, so that excluded evidence stays auditable without entering the headline table.

The fourth lesson is to re-test continued checkpoints across independent runs before replacing an earlier model: across six runs per architecture, neither the original collapse nor the architecture difference recurred as a systematic effect, and the between-run variation is wide enough to produce either result from a single run.

### Limitations

Several limitations bound what the study can show. First, the action space is small and masked: the twelve joint actions are nominal, and the distance-zone mask leaves one executable movement beyond 8 studs and one between 4.8 and 8, so most real choices occur inside sword range and part of the route around cover is produced by the engine's move-to behavior and the stuck-override rule rather than by the learned policy. The factored dual-head model was not tested on any larger compositional space, and nothing here predicts its behavior in the regime action branching was designed for<sup>28</sup>. Second, the networks apply the hyperbolic tangent to their output layer, so every network output is bounded in (−1, 1) and the factored dual-head model's reconstructed joint values in (−11/3, 11/3), while a single decision's reward can reach ±4 for a hit, +70 for a kill or below −10 under the passive penalty, and

**Table 8 The four original checkpoints re-evaluated under the revision protocol.** Each cell is one training run scored over three 600 s windows on the frozen fixture sequence, and the unit here is the evaluation seed rather than the training run. The last two columns give the Original protocol figures of Table 4 (NPC-1 recomputed from the original ledger, and any-side as originally reported). Compare NPC-1 with NPC-1 and any-side with any-side, since the two idioms differ for the factored dual-head rows.

| Checkpoint                 | Clean blocked-start pair episodes | NPC-1 recovery | Any-side | Both-side | Original protocol: NPC-1 (Table 4) | Original protocol: any-side (Table 4) |
|----------------------------|-----------------------------------|----------------|----------|-----------|------------------------------------|---------------------------------------|
| 60-min flat-joint          | 20                                | 13/20 (65.0%)  | 65.0%    | 65.0%     | 33.6%                              | 33.6%                                 |
| 60-min factored dual-head  | 27                                | 15/27 (55.6%)  | 59.3%    | 55.6%     | 50.0%                              | 68.6%                                 |
| 120-min flat-joint         | 18                                | 2/18 (11.1%)   | 11.1%    | 11.1%     | 33.7%                              | 33.7%                                 |
| 120-min factored dual-head | 10                                | 0/10 (0.0%)    | 0.0%     | 0.0%      | 9.0%                               | 13.0%                                 |

**Table 9 Blocked-start recovery of NPC-1 against a different training run of the same architecture (fixed-opponent condition).** Six seed pairings of three windows each per row. Mean, median, range and interval are over the six pairings, and the self-play mean is from Tables 6 and 7. The pairings reuse the six training runs, each run appearing once as the evaluated side and once as the opponent, so they are not six independent replicates. The interval is the exact enumerated bootstrap over pairings, given as a descriptive summary of their spread, and is not interchangeable with the run-level intervals of Tables 6 and 7.

| Architecture       | Checkpoint | Pairings | Clean blocked-start pairs | Mean  | Median | Range          | 95% interval | Self-play mean |
|--------------------|------------|----------|---------------------------|-------|--------|----------------|--------------|----------------|
| Flat-joint         | 60 min     | 6        | 158                       | 55.5% | 57.4%  | 13.9% to 88.6% | [35.8, 73.1] | 58.0%          |
| Flat-joint         | 120 min    | 6        | 126                       | 43.9% | 55.4%  | 7.1% to 70.8%  | [24.0, 63.2] | 55.8%          |
| Factored dual-head | 60 min     | 6        | 168                       | 43.9% | 46.0%  | 5.0% to 87.1%  | [20.8, 67.1] | 48.7%          |
| Factored dual-head | 120 min    | 6        | 129                       | 46.9% | 42.0%  | 5.6% to 84.6%  | [26.6, 67.7] | 50.4%          |

**Table 10 Conventional combat metrics beside the encounter-level metric, over the clean blocked-start pair episodes of Table 4.** Recomputed from the parsed ledger of the original runs. Damage equals 10 × hits. Duration is censored at the stalemate cutoff (90 s without a hit) and at window end. “Ended” pair episodes are those closed by a death (a win for the other side) or by the stalemate cutoff (“no kill”). Pair episodes still open at window end are not in the win split.

| Checkpoint                 | Swings | Hits (rate) | Wins NPC-1 / NPC-2 / no kill | Median duration (s) | NPC-1 recovery | Any-side recovery |
|----------------------------|--------|-------------|------------------------------|---------------------|----------------|-------------------|
| 60-min flat-joint          | 426    | 343 (80.5%) | 25 / 27 / 46                 | 90.0                | 36/107 (33.6%) | 36/107 (33.6%)    |
| 60-min factored dual-head  | 706    | 699 (99.0%) | 26 / 61 / 9                  | 10.3                | 51/102 (50.0%) | 70/102 (68.6%)    |
| 120-min flat-joint         | 464    | 445 (95.9%) | 25 / 30 / 35                 | 90.0                | 34/101 (33.7%) | 34/101 (33.7%)    |
| 120-min factored dual-head | 29     | 24 (82.8%)  | 7 / 2 / 76                   | 90.0                | 9/100 (9.0%)   | 13/100 (13.0%)    |

the temporal-difference error is clipped to  $\pm 10$  before back-propagation. The networks therefore learn a ranking of actions rather than a calibrated return, and this, with the 35-transition buffer, the absence of a target network and the constant exploration rate, is a generic source of instability<sup>26,27</sup> that we cannot separate from any property of the two output heads. A preregistered screen of this fragility (Supplementary Table S6) trained 42 further flat-joint runs: six variants of the configuration at six runs each (a hard-updated target network; a 5,000-transition buffer with 32-sample updates; a linear output layer; hit and damage rewards scaled to one quarter; and the larger buffer with weight decay at two strengths) plus a six-run replication of the larger-buffer variant, each run evaluated at 60 and 120 min on the same fixtures. No screened candidate setting met all three preregistered criteria: between-run dispersion below the original configuration's; no 60-to-120-min decline by the screen's rule (a one-sided sign-flip  $p \geq 0.05$  on the mean change and a median change of at least  $-5$  points); and final weights within the screen's bound of 1.5 times the largest absolute weight of the original runs. The larger-buffer variant passed the non-decline rule in both cohorts, but its median 120-min recovery in replication was below the original's on the screen's windows (49.4% against 65.5%) and two of its twelve runs exceeded the weight bound. The original configuration was retained because the screen selected no replacement, not because it was shown to be stable. The screen does not locate the source of the instability among these settings. Third, all results come from one  $60 \times 62$  stud arena with one U-shaped cover and one straight-wall occluder. Transfer to other layouts, cover positions or opponent placements is untested. A descriptive split of the replicated runs by the occluder that blocked NPC-1 at the start (Supplementary Table S5) puts run-mean recovery behind the straight wall above that behind the U-shaped cover in eight of ten architecture-by-checkpoint cells (at 120 min, 65.5% against 43.7% flat-joint, 59.9% against 39.5% factored dual-head). The two occluders differ in shape, position, start distance and fixture composition at once, so the split describes this arena and supports no claim about occluder shape or other layouts. Fourth, each NPC trains against an opponent that is itself learning, so behavior is the product of two co-adapting policies<sup>31,32</sup>. The fixed-opponent condition of Table 9 bounds rather than removes this: means move by 2 to 12 percentage points, but the spread across the six ring pairings, each of which changes both the evaluated run and its opponent, is wide, as is the spread across runs in self-play. With one foreign opponent per run, the design cannot separate the evaluated run's contribution from the opponent's, so recovery cannot be attributed to the policy under test alone. Fifth, the training horizon is wall-clock rather than a fixed number of updates, so a policy that becomes passive collects fewer transitions and fewer updates per minute, confounding longer training with more up-

dates. Each run-level rate also pools only three windows, 10 to 109 clean blocked-start pair episodes, so evaluation sampling within a run and differences between runs are not separated by the design. Sixth, the reward was designed for engagement and contains no explicit LOS-recovery term. The evaluation records where each blocked-start pair episode stopped along the chain (Supplementary Table S3) but does not decompose the reward: the training logs hold per-life totals only, with no per-decision breakdown and no link to pair episodes. The evidence is therefore behavioral: it shows that the chain is traversed under this reward, not that the reward's components produce it. A reward-component analysis and a shaping ablation are left to future work. Also left to future work are a structural ablation of the factored dual-head model (value stream, separate branches, additive reconstruction), variation of weapon range, movement speed, health and arena size, a comparison with a classic dueling DQN or a lightweight policy-gradient learner, and comparisons at matched compute budgets. The configuration screen above tried a target network and a larger buffer but is not such a comparison. Finally, the raw server logs of the original runs were not retained, so their update counts cannot be recovered and the seed of the 120-min flat-joint continuation is unrecorded. The replicated experiments record both for every run.

### Impact and significance

The platform choice is part of the study's significance. The value of this small DQN-style setup is not that it replaces mature stacks<sup>3</sup> or the creator-built external route<sup>13</sup> but that it shows a lower-overhead path for creators without a GPU or a lab: discrete CPU-friendly policies inspectable with ordinary Studio logs and pair-episode labels.

Two parts of the work should be distinguished. The evaluation protocol is platform-agnostic: pair-episode segmentation on death and respawn boundaries, realized start-condition labels, an active-attack success rule, a failure taxonomy separating failure to get around cover from failure to attack, an explicit clean/excluded ledger, and re-testing of continued checkpoints across runs. Each needs only positions, a LOS query and combat events exposed to a script, which Unity, Unreal, Godot and Minecraft mods all provide. The training envelope is Roblox-specific: CPU-only Luau under a per-frame budget, Actor parallelism without shared memory<sup>46</sup>, HttpService as the only external channel at 500 requests per minute per server<sup>11</sup>, and DataStore as the checkpoint store. Both routes share one sampling ceiling: the simulation runs in real time at about 60 Hz with no time acceleration, a place cannot be replicated cheaply into headless instances, and an external bridge is capped by the request limit above, so neither route collects experience faster than the game plays. On a platform with a first-party trainer<sup>3,4,8</sup>, the same protocol would sit on

top of an external trainer.

## Conclusion

### Protocol contribution

The transferable artifact is a compact evaluation protocol for LOS-dependent NPC combat: pair-episode scoring, blocked-start labels, an active-attack success rule, a clean/excluded ledger, and failure-class labels separating failure to get around cover from failure to attack. None depends on the DQN variant used.

### Empirical findings

In the original single-run trajectory, the factored dual-head model recovered from blocked starts more often than the flat-joint model at 60 min and collapsed by 120 min, mainly by failing to get around the cover. Across six independent runs per architecture, neither the drop nor the advantage recurred as a systematic effect (exact sign-flip  $p = 0.84$  and  $0.94$ ; permutation  $p$  from  $0.56$  to  $0.98$ ), while the between-run range at a fixed checkpoint was 34 to 90 percentage points, with individual runs collapsing to zero recovery and, except at the final checkpoint, recovering again. The original observation stands as the case that motivated the protocol, not as evidence that either parameterization is superior or that continued training degrades one of them. We do not claim that one DQN parameterization is better in general.

## Acknowledgments

The authors thank Jack Zhou for research mentorship and computing infrastructure. M.Z. designed and ran the experiments, wrote the evaluation protocol and analysis code, and drafted the manuscript; W.N. contributed to the earlier conference versions of this work and to the design and interpretation of the replicated experiments. This research received no external funding, and the authors declare no competing interests. AI-assisted tools (Anthropic's Claude) were used for writing and formatting assistance; the experiments, data, and conclusions are the authors' own.

## References

- 1 M. G. Bellemare, Y. Naddaf, J. Veness, M. Bowling. The Arcade Learning Environment: an evaluation platform for general agents. *Journal of Artificial Intelligence Research*. Vol. 47, pg. 253–279, 2013, <https://doi.org/10.1613/jair.3912>.
- 2 M. Kempka, M. Wydmuch, G. Runc, J. Toczek, W. Jaśkowski. ViZDoom: a Doom-based AI research platform for visual reinforcement learning. 2016 IEEE Conference on Computational Intelligence and Games (CIG). pg. 1–8, 2016, <https://doi.org/10.1109/CIG.2016.7860433>.
- 3 A. Juliani, V.-P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar, D. Lange. Unity: a general platform for intelligent agents. arXiv preprint, <https://arxiv.org/abs/1809.02627>, 2020.
- 4 M. Ranaweera, Q. H. Mahmoud. Deep reinforcement learning with Godot game engine. *Electronics*. Vol. 13, 985, 2024, <https://doi.org/10.3390/electronics13050985>.
- 5 N. Justesen, P. Bontrager, J. Togelius, S. Risi. Deep learning for video game playing. *IEEE Transactions on Games*. Vol. 12, pg. 1–20, 2020, <https://doi.org/10.1109/TG.2019.2896986>.
- 6 S. Risi, M. Preuss. From chess and Atari to StarCraft and beyond: how game AI is driving the world of AI. *KI – Künstliche Intelligenz*. Vol. 34, pg. 7–17, 2020, <https://doi.org/10.1007/s13218-020-00647-w>.
- 7 E. Alonso, M. Peter, D. Goumard, J. Romoff. Deep reinforcement learning for navigation in AAA video games. *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)*. pg. 2133–2139, 2021, <https://doi.org/10.24963/ijcai.2021/294>.
- 8 T. Liu, A. Cann, I. Colbert, M. Saeedi. Combining reinforcement learning and behavior trees for NPCs in video games with AMD Schola. arXiv preprint, <https://arxiv.org/abs/2510.14154>, 2025.
- 9 J. Gillberg, J. Bergdahl, A. Sestini, A. Eakins, L. Gisslén. Technical challenges of deploying reinforcement learning agents for game testing in AAA games. 2023 IEEE Conference on Games (CoG). pg. 1–8, 2023, <https://doi.org/10.1109/CoG57401.2023.10333194>.
- 10 M. Jacob, S. Devlin, K. Hofmann. “It’s unwieldy and it takes a lot of time” — challenges and opportunities for creating agents in commercial games. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. Vol. 16, pg. 88–94, 2020, <https://doi.org/10.1609/aiide.v16i1.7415>.
- 11 Roblox Corporation. In-game HTTP requests and HttpService. Roblox Creator Hub documentation, <https://create.roblox.com/docs/cloud-services/http-service> and <https://create.roblox.com/docs/reference/engine/classes/HttpService>, 2026.
- 12 Rojo contributors. Rojo documentation: sync details. <https://rojo.space/docs/v6/sync-details/>, 2026.
- 13 A. Whitedeer, A. Li, C. Sang. Parallel deep-RL agents for Roblox obstacle-course navigation: from single-course memorization to generalizing across procedurally-composed courses. Stanford CS224R course project report, Stanford University, [https://cs224r.stanford.edu/projects/cs224r\\_final\\_projects.html](https://cs224r.stanford.edu/projects/cs224r_final_projects.html), 2026.
- 14 Kironite. Neural network library 2.0. Roblox Developer Forum, <https://devforum.roblox.com/t/neural-network-library-20/869557>, 2020.
- 15 A. H. Aiman. DataPredict: Lua-based machine learning, deep learning and reinforcement learning library (for Roblox and pure Lua), with the Self-Learning Sword-Fighting AIs example. GitHub and Roblox Developer Forum, <https://github.com/AqamCreates/DataPredict>, 2023.
- 16 ImNotKevPlayz. Deep reinforcement learning Snake AI | N-step dueling double DQN | improves over time. Roblox Developer Forum, <https://devforum.roblox.com/t/deep-reinforcement-learning-snake-ai-n-step-dueling-double-dqn-improves-over-time/2541442>, 2023.
- 17 ImNotKevPlayz. Sword fighting AIs, recommendations for reward functions? / devlog. Roblox Developer Forum, <https://devforum.roblox.com/t/sword-fighting-ais-recommendations-for-reward-functions-devlog/2096226>, 2023.
- 18 Eternity.Devs. AI based for 1v1 sword fighting. Roblox Developer Forum, <https://devforum.roblox.com/t/ai-based-for-1v1-sword-fighting/4608738>, 2026.
- 19 Y. Yue, C. Green, S. Hunt, I. Salia, W. Shi, J. J. Hunt. Pixels to play: a foundation model for 3D gameplay. 2025 IEEE Conference on Games (CoG). pg. 1–4, 2025, <https://doi.org/10.1109/CoG64752>

- .2025.11114171.
- 20 V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis. Human-level control through deep reinforcement learning. *Nature*. Vol. 518, pg. 529–533, 2015, <https://doi.org/10.1038/nature14236>.
  - 21 L.-J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*. Vol. 8, pg. 293–321, 1992, <https://doi.org/10.1007/BF00992699>.
  - 22 H. van Hasselt, A. Guez, D. Silver. Deep reinforcement learning with double Q-learning. *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30, pg. 2094–2100, 2016, <https://doi.org/10.1609/aaai.v30i1.10295>.
  - 23 T. Schaul, J. Quan, I. Antonoglou, D. Silver. Prioritized experience replay. *International Conference on Learning Representations (ICLR)*. 2016, <https://arxiv.org/abs/1511.05952>.
  - 24 Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, N. de Freitas. Dueling network architectures for deep reinforcement learning. *Proceedings of the 33rd International Conference on Machine Learning (PMLR)*. Vol. 48, pg. 1995–2003, 2016, <https://proceedings.mlr.press/v48/wangf16.html>.
  - 25 M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, D. Silver. Rainbow: combining improvements in deep reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32, pg. 3215–3222, 2018, <https://doi.org/10.1609/aaai.v32i1.11796>.
  - 26 W. Fedus, P. Ramachandran, R. Agarwal, Y. Bengio, H. Larochelle, M. Rowland, W. Dabney. Revisiting fundamentals of experience replay. *Proceedings of the 37th International Conference on Machine Learning (PMLR)*. Vol. 119, pg. 3061–3071, 2020, <https://proceedings.mlr.press/v119/fedus20a.html>.
  - 27 H. van Hasselt, Y. Doron, F. Strub, M. Hessel, N. Sonnerat, J. Modayil. Deep reinforcement learning and the deadly triad. *NeurIPS 2018 Deep Reinforcement Learning Workshop*. 2018, <https://arxiv.org/abs/1812.02648>.
  - 28 A. Tavakoli, F. Pardo, P. Kormushev. Action branching architectures for deep reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32, pg. 4131–4138, 2018, <https://doi.org/10.1609/aaai.v32i1.11798>.
  - 29 A. Tavakoli, M. Fatemi, P. Kormushev. Learning to represent action values as a hypergraph on the action vertices. *International Conference on Learning Representations (ICLR)*. 2021, [https://openreview.net/forum?id=Xv\\_s64FiXTv](https://openreview.net/forum?id=Xv_s64FiXTv).
  - 30 Z. Fan, R. Su, W. Zhang, Y. Yu. Hybrid actor-critic reinforcement learning in parameterized action space. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)*. pg. 2279–2285, 2019, <https://doi.org/10.24963/ijcai.2019/316>.
  - 31 J. N. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. S. Torr, P. Kohli, S. Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. *Proceedings of the 34th International Conference on Machine Learning (PMLR)*. Vol. 70, pg. 1146–1155, 2017, <https://proceedings.mlr.press/v70/foerster17b.html>.
  - 32 T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, I. Mordatch. Emergent complexity via multi-agent competition. *International Conference on Learning Representations (ICLR)*. 2018, <https://arxiv.org/abs/1710.03748>.
  - 33 B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, I. Mordatch. Emergent tool use from multi-agent autocurricula. *International Conference on Learning Representations (ICLR)*. 2020, <https://arxiv.org/abs/1909.07528>.
  - 34 P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, D. Meger. Deep reinforcement learning that matters. *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32, pg. 3207–3214, 2018, <https://doi.org/10.1609/aaai.v32i1.11694>.
  - 35 C. Colas, O. Sigaud, P.-Y. Oudeyer. How many random seeds? Statistical power analysis in deep reinforcement learning experiments. *arXiv preprint*, <https://arxiv.org/abs/1806.08295>, 2018.
  - 36 R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, M. G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*. Vol. 34, pg. 29304–29320, 2021, <https://proceedings.neurips.cc/paper/2021/hash/f514cec81cb148559cf475e7426eed5e-Abstract.html>.
  - 37 A. Patterson, S. Neumann, M. White, A. White. Empirical design in reinforcement learning. *Journal of Machine Learning Research*. Vol. 25, no. 318, pg. 1–63, 2024, <https://jmlr.org/papers/v25/23-0183.html>.
  - 38 S. M. Jordan, Y. Chandak, D. Cohen, M. Zhang, P. S. Thomas. Evaluating the performance of reinforcement learning algorithms. *Proceedings of the 37th International Conference on Machine Learning (PMLR)*. Vol. 119, pg. 4962–4973, 2020, <https://proceedings.mlr.press/v119/jordan20a.html>.
  - 39 R. Gorsane, O. Mahjoub, R. J. de Kock, R. Dubb, S. Singh, A. Pretorius. Towards a standardised performance evaluation protocol for cooperative MARL. *Advances in Neural Information Processing Systems*. Vol. 35, pg. 5510–5521, 2022, <https://doi.org/10.52202/068431-0398>.
  - 40 Y. Zhao, I. Borovikov, F. de Mesentier Silva, A. Beirami, J. Rupert, C. Somers, J. Harder, J. Kolen, J. Pinto, R. Pourabghasem, J. Pestrak, H. Chaput, M. Sardari, L. Lin, S. Narravula, N. Aghdaie, K. Zaman. Winning is not everything: enhancing game development with intelligent agents. *IEEE Transactions on Games*. Vol. 12, pg. 199–212, 2020, <https://doi.org/10.1109/TG.2020.2990865>.
  - 41 A. Sestini, L. Gisslén, J. Bergdahl, K. Tollmar, A. D. Bagdanov. Automated gameplay testing and validation with curiosity-conditioned proximal trajectories. *IEEE Transactions on Games*. Vol. 16, pg. 113–126, 2024, <https://doi.org/10.1109/TG.2022.3226910>.
  - 42 I. Oh, S. Rho, S. Moon, S. Son, H. Lee, J. Chung. Creating pro-level AI for a real-time fighting game using deep reinforcement learning. *IEEE Transactions on Games*. Vol. 14, pg. 212–220, 2022, <https://doi.org/10.1109/TG.2021.3049539>.
  - 43 S. Devlin, R. Georgescu, I. Momennejad, J. Rzepecki, E. Zuniga, G. Costello, G. Leroy, A. Shaw, K. Hofmann. Navigation Turing Test (NTT): learning to evaluate human-like navigation. *Proceedings of the 38th International Conference on Machine Learning (PMLR)*. Vol. 139, pg. 2644–2653, 2021, <https://proceedings.mlr.press/v139/devlin21a.html>.
  - 44 A. Y. Ng, D. Harada, S. Russell. Policy invariance under reward transformations: theory and application to reward shaping. *Proceedings of the Sixteenth International Conference on Machine Learning (ICML)*. pg. 278–287, 1999, <https://dl.acm.org/doi/10.5555/645528.657613>.
  - 45 D. Balduzzi, K. Tuyls, J. Pérolat, T. Graepel. Re-evaluating evaluation. *Advances in Neural Information Processing Systems*. Vol. 31, pg. 3272–3283, 2018, <https://papers.nips.cc/paper/7588-re-evaluating-evaluation>.
  - 46 Roblox Corporation. Parallel Lua. Roblox Creator Hub documentation, <https://create.roblox.com/docs/scripting/luau/parallel-luau>, 2026.

## Supplementary Information

The online version contains supplementary material available at <https://nhsjs.com/?p=57564>