ROME WASN'T BUILT IN ONE TURN: REPLAY-GROUNDED REWARD SIGNALS FOR LONG-HORIZON FREECIV AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Training language-model agents to play long-horizon strategy games is hard because deterministic feedback is sparse and often arrives only at the end of a game. In-game scores can offer dense signals, but they do not reliably predict future outcomes: in 8–68% of position pairs across eight strategy games, the higher-scoring position leads to the worse outcome. We term this phenomenon *scoreboard traps*. In Freeciv, an open-source turn-based empire-building game where several players manage units, cities, and diplomacy over hundreds of turns, we look into better reward signals for gameplay agents. To facilitate this, we build CIVHARNESS, an open system that saves current game states, branches on different decisions, and replays the game to future turns to provide measurable long-horizon feedback from saved positions. We use CIVHARNESS to train CIVTELESCOPE, an LLM Judge that predicts which one of two game positions will yield the better outcome. On scoreboard-trap pairs, CIVTELESCOPE is 1.3–2.0 times more accurate than off-the-shelf language models, and it generalizes to new game starts, later turns, and unseen rules. We then train a Freeciv gameplay agent with CIVTELESCOPE as a dense reward, and call this recipe CIVMARSHMALLOW. When used in the middle stage of the game where scoreboard traps are common, CIVMARSHMALLOW reaches an average training phase gain of 8.94 points, 77% more than the best baseline, the end-of-game reward. Over the full 120-turn game, CIVMARSHMALLOW uses a hybrid reward that starts from the scoreboard or the end-of-game reward and shifts to CIVTELESCOPE as the game advances; this hybrid reward scores highest among the rewards we compare.

1 INTRODUCTION

Large Language Model (LLM) agents have made substantial progress in decision-making in settings that require exploration and planning, like everyday web use (Zhou et al., 2024; Feng et al., 2025), mobile device control (Bai et al., 2025), and strategy game playing (Schultz et al., 2025; Belle et al., 2025; Jin et al., 2026). In strategy games, agents must learn to plan carefully, manage resources, and make tactical decisions. Reinforcement learning (RL) with game-based feedback offers a way to train these LLM agents (Zhou et al., 2024; Wang et al., 2025; Liu et al., 2026). However, strategy games often involve *long horizons*, with the consequences of decisions unfolding over many turns. Agents must therefore learn from sparse, delayed feedback at the end of the game and connect these eventual outcomes to intermediate decisions.

One readily available source of dense feedback for gameplay agents is the current in-game score, shown in the game's *scoreboard* (Mnih et al., 2015; Küttler et al., 2020; Klissarov et al., 2024). However, this score may not reflect the long-term value of a current game position: the benefits of strategic investments often emerge later, while changing game conditions and other players' actions can undermine an apparent lead. We call a pair of positions a *scoreboard trap* when the position with the higher current score compared to the other leads to a worse future outcome (See Figure 1(a)). Across eight strategy games with varying gameplay complexity, we demonstrate that such traps affect 8–68% of position pairs (Figure 2).

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.

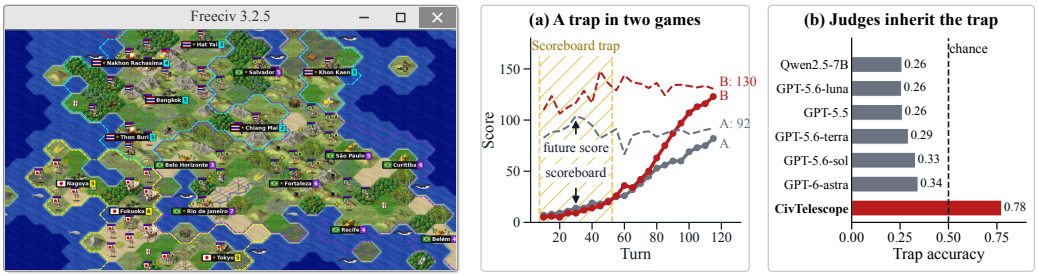


Figure 1: **Freeciv and the scoreboard trap.** (a) Current scoreboard score (solid) and future score at turn 120 (dashed) in Freeciv games A (gray) and B (red). A leads the scoreboard until turn 50, while B leads the future score during the trap (yellow). (b) Frontier LLMs fail at identifying scoreboard traps given a pair of two positions.

To study how to provide better reward signal for strategic gameplay agents, we focus our experiments on Freeciv, an open-source turn-based empire-building game (Freeciv developers, 2026). We introduce CIVHARNESS (Section 4), a reusable harness for agentic reinforcement learning that saves game states, branches on alternative decisions, and replays their continuations to provide training feedback. With CIVHARNESS, we find that frontier LLMs rarely detect scoreboard traps: when asked which of two positions leads to the better future outcome, five frontier models reach only 26%–34% accuracy on held-out trap pairs, below chance (Figure 1(b)). We instead train CIVTELESCOPE (Section 5.2), a pairwise language-model judge, on future outcomes averaged over replays from each position, which reaches 77.5% accuracy on the same pairs.

We use CIVTELESCOPE to provide dense rewards for training a Freeciv gameplay agent (Section 5.3), with the final game score as the training objective, and call this recipe CIVMARSHMALLOW[1]. Training proceeds in four phases, with 20, 40, 80, and 120 turns remaining, allowing us to compare CIVMARSHMALLOW with sparse end-of-game and dense scoreboard rewards at each phase. With 80 turns remaining, where scoreboard traps are common and CIVTELESCOPE can already rank positions, CIVMARSHMALLOW improves the phase gain by 77% over the best baseline (the end-of-game reward). For the full 120-turn game, CIVMARSHMALLOW shifts from the scoreboard or the end-of-game outcome to CIVTELESCOPE as the game advances, which keeps the early-game signal stable; this hybrid reward scores highest among the rewards we compare.

Our contributions are:

1. **Scoreboard traps.** We show that scoreboard traps are common across eight strategy games and that frontier LLMs fall below chance on them in Freeciv (Section 3).
2. **CIVHARNESS.** We release an open-source harness for agentic RL in Freeciv that saves, branches, and replays game states, so any feedback signal can be checked against actual future outcomes (Section 4).
3. **CIVTELESCOPE.** We train a pairwise judge on replay outcomes that generalizes to new starts, later turns, and an unseen ruleset; trained on a frontier model's preferences instead, the same judge stays below chance (Sections 5.2 and 6.2).
4. **CIVMARSHMALLOW.** We train Freeciv agents in phases with CIVTELESCOPE as a dense reward, alone and combined with scoreboard or end-of-game rewards over the full game (Sections 5.3, 6.3 and 6.4).

2 RELATED WORK

Turn-based strategy-game agents. CivRealm provides a Freeciv environment for learning and language-based agents (Qi et al., 2024); SAGA and Vox Deorum add planning structure (Jin et al., 2026; Chen et al., 2025), and Civilization benchmarks report sparse terminal feedback over hundreds of turns (Chen et al., 2026; Andrews et al., 2026). These environments start each run from a new game

[1]As in the marshmallow test of delayed gratification (Mischel et al., 1972), CIVTELESCOPE rewards a position for an outcome that arrives many turns later, even when the position gives up score now.

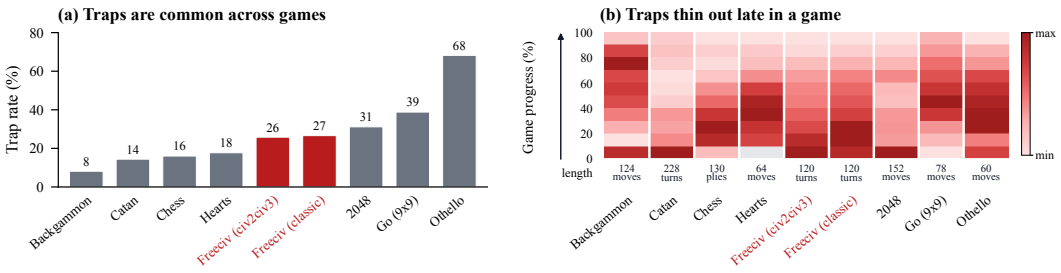


Figure 2: **Scoreboard traps.** (a) Trap rate across eight strategy games. (b) Trap rate by game progress; darker red is a higher rate, scaled within each game.

or a designed scenario, whereas CIVHARNESS branches and replays any saved position, so policies and rewards can be compared against future outcomes from shared positions. Unlike the value estimator of Chen et al. (2026) which learned from played games to evaluate agents, CIVTELESCOPE learns from replayed branches of the same position and serves as a training reward.

Model feedback and proxy rewards. Language- and vision-model feedback can be distilled into rewards (Klissarov et al., 2024; Wang et al., 2024c; Cao et al., 2024; Xie et al., 2024), but it depends on the model recognizing progress; optimizing a proxy can diverge from the intended objective (Gao et al., 2022; Razin et al., 2025), and game benchmarks document gaps between visible scores and progress (Küttler et al., 2020; Paglieri et al., 2025). We measure these gaps against replayed outcomes and show that replay labels, unlike preference labels, do not inherit the labeler's trap errors.

Rollout-grounded feedback and curricula. Process supervision and value estimation use step labels, Monte-Carlo rollouts, or progress measures (Lightman et al., 2024; Wang et al., 2024b; Yuan et al., 2025; Setlur et al., 2025; Kazemnejad et al., 2025), and agent methods learn rewards or values from interaction (Xia et al., 2025; Bai et al., 2025; Zhou et al., 2024; Feng et al., 2025). We instead train a pairwise judge from rollout outcomes in a multiplayer strategy game and calibrate it with reference positions of known value. Following reverse curricula (Florensa et al., 2017; Salimans & Chen, 2018), we increase the remaining turns from phase to phase and compare reward signals within each phase.

3 BACKGROUND

Freeciv. Freeciv (Freeciv developers, 2026) is an open-source turn-based strategy game modeled on Sid Meier's Civilization[2]. Each player leads a civilization from its first settlers: it founds and grows cities, researches technologies, builds units and wonders, and negotiates or fights with the other players. In our games, the agent controls one of four players on a 1000-tile map; the other three are Freeciv's built-in AI players at a fixed *bot difficulty* (Section 4). The agent sees only what its units and cities have explored. Freeciv keeps a running score for every player, computed from quantities such as citizens, technologies, wonders, and culture, and shows it on the in-game scoreboard; we call the agent's score at the end of the game its *terminal score*. The games follow one of two rulesets, CIV2CIV3 (the Freeciv 3 default, used in our main experiments) or CLASSIC (Appendix A.1).

Scoreboard traps across strategy games. Strategy games often suffer from *scoreboard traps*: when the higher current position can lead to a worse future outcome. To measure the traps, we build pairs of positions from recorded games and count the fraction of pairs in which the scoreboard order opposes the order of their future outcomes. Scoreboard traps occur in 8–68% of decidable position pairs across eight games (Figure 2(a)), including 26% in Freeciv under CIV2CIV3 and 27% under CLASSIC (Appendix Table 5). Trap rates also vary with game progress (Figure 2(b)): in CIV2CIV3, the rate falls from 55% in the first tenth of the game to 8% in the last (Appendix A.2). In Freeciv, the scoreboard also becomes a worse predictor as games get longer: from 60-turn to 80-turn games, its

[2] https://civilization.2k.com/

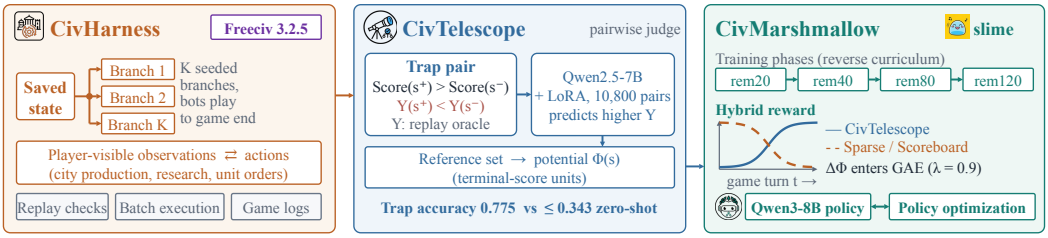


Figure 3: **Overview of CIVHARNESS, CIVTELESCOPE, and CIVMARSHMALLOW.** CIVHARNESS branches saved Freeciv states and exchanges player-visible observations and actions with the agent. CIVTELESCOPE, a pairwise judge trained on replay-labeled trap pairs, gives a potential $\Phi(s)$ through a reference set. CIVMARSHMALLOW trains the policy in four phases with CIVTELESCOPE reward, and with a hybrid reward for full games.

correlation with the terminal score falls from 0.81 to 0.12 and the trap rate rises from 30% to 54% (Appendix Table 9).

Off-the-shelf LLM judges. On held-out Freeciv trap pairs, all six LLMs, from frontier closed models to a small open model, score below chance (0.26–0.34), even though they reach 0.77–0.82 on non-trap pairs (Figure 1(b); Appendix Table 11). Warning them in the prompt that visible scores can mislead does not help: the frontier models still rank only 0.32–0.38 of trap pairs correctly. The failure also appears in simpler games: four of five frontier models rank below chance in Chess, and two of five in Othello (Appendix Table 6).

4 DESIGNING AND BUILDING CIVHARNESS

To facilitate our training and evaluation, we introduce CIVHARNESS, a reusable harness for agentic reinforcement learning in Freeciv. From a saved state, CIVHARNESS runs K deterministically seeded branches on the real Freeciv game engine, either to the end of the game or for a chosen number of turns, and records their observations, actions, and outcomes; replay checks verify that the same save, seed, and action trace reproduce the same state.

A gameplaying agent interacts through configurable observations and actions, including city production, research, and unit orders executed by the game engine. The agent sees only its own player's view of the game, not the full game state. We also speed gameplay rollouts by implementing batch execution, which manages independent games, retries on failed gameplay runs, and records completed work so collection can resume.

CIVHARNESS records the agent's ordinary rollouts, and restoring a saved position lets it collect sibling rollouts that take different decisions from the same state, for group-based training and for comparing reward signals from shared starts. Game logs report observations, issued actions, branch seeds, and bot difficulty. Researchers using CIVHARNESS can therefore reuse position banks, replace a policy or reward, and reproduce the resulting comparisons with the same replay checks.

Figure 3 shows how CIVHARNESS connects to slime, the reinforcement-learning training engine used in our experiments. The rollout interface exchanges player-visible observations and actions between the policy and the game engine, converts CIVTELESCOPE's pairwise preferences into state values with a reference set for judge-based rewards (Equation (2)), and passes the resulting samples to slime, which updates the policy. Appendix B.1 provides further details on the harness design.

5 TRAINING A FREECIV GAMEPLAYING AGENT

In this section, we describe using CIVHARNESS to first create *trap pairs* to train an LLM judge, CIVTELESCOPE, that can escape the scoreboard trap. We then detail CIVMARSHMALLOW, our recipe for training a Freeciv gameplay agent with CIVTELESCOPE rewards, used alone or combined with other rewards over the course of a game.

5.1 REPLAY ORACLE AND TRAP PAIRS

We define a *replay oracle* as a mechanism that takes the current game state and uses CIVHARNESS to observe future game states, possible till the end of the game. For a game state s, the replay oracle creates K branches forward from s, all the way to the end of the game with Freeciv's built-in bots at a fixed bot difficulty. We define $Y(s)$ as the player's mean terminal score over these branches from s. For positions taken from policy rollouts, $Y(s)$ is the realized terminal score of that rollout. When the agent's policy π plays the player instead of the bots, we write $Y_\pi(s)$ for its mean terminal score from s.

We use the replay oracle to create *trap pairs* that explicitly capture the scoreboard trap, as demonstrated in Section 3. A pair (s^+, s^-) is a trap pair when

$$\text{SCOREBOARD}(s^+) > \text{SCOREBOARD}(s^-), \qquad Y(s^+) < Y(s^-). \tag{1}$$

We control the starting position, replay length, branch count, and bot difficulty when generating branches. Since the terminal score is not the *true* terminal score but rather future replays, we only pick pairs where the replay oracle's terminal scores differ more than twice their pooled error. This is done so that only significantly different states are considered. Pairs of rollouts that share the same start and are compared at the same turn show the trap at equal or higher rates (Appendix A.4).

5.2 USING CIVHARNESS TO BUILD CIVTELESCOPE

CIVTELESCOPE is a pairwise language-model-based judge, fine-tuned from Qwen2.5-7B. It predicts which position has the higher terminal score under the replay oracle. The input of CIVTELESCOPE is a pair of game positions, each rendered as text from the evaluated player's point of view: every player's score together with that player's gold, technologies, cities, and units. The output is the position expected to reach the higher terminal score; because the answer can depend on which position is listed first, we query both orders and average the two answers. The training labels are the terminal scores of two kinds of games: branches played to the end by Freeciv's bots from saved positions, which give $Y(s)$, and the agent's own rollouts. A training pair holds two positions from different games at most ten turns apart whose labels differ by more than twice their pooled standard error, and we fine-tune the model with LoRA (rank 16) on 10,800 such pairs. Section 5.3 turns its pairwise preferences into a reward.

5.3 CIVMARSHMALLOW: TRAINING RECIPE FOR FREECIV GAMEPLAY AGENTS

We now provide the full training recipe, CIVMARSHMALLOW, for training a Freeciv gameplaying agent. The recipe has three parts: a phased training schedule that moves from late-game to full-game starts, the CIVTELESCOPE reward, which turns the judge's preferences into a dense reward, and, for full games, a hybrid reward that combines CIVTELESCOPE with a sparse or scoreboard reward. We train the baseline rewards, SPARSE and SCOREBOARD, with the same schedule, so that every reward is compared under the same conditions.

Phased training approach. Every gameplay agent, whether trained with CIVMARSHMALLOW or with a baseline reward, is trained on late-game positions first and on earlier positions later. We denote the number of game turns remaining to play by `rem`, and the maximum number of turns in our gameplay is 120. The gameplay agent is trained sequentially in four *training phases*, `rem20`, `rem40`, `rem80`, and `rem120`, each with more turns remaining than the one before. Each training phase starts from the policy checkpoint of the preceding phase and a fixed position bank , a set of saved game states from which training games start (turn 100, 80, 40, and 1). Training thus follows a reverse curriculum, which starts near the end of a task and moves the start earlier as training proceeds (Florensa et al., 2017; Salimans & Chen, 2018).

Rewards. We compare two baseline rewards, SPARSE and SCOREBOARD, with the CIVTELESCOPE reward of CIVMARSHMALLOW; all three use the same rollout budget. SPARSE rewards use only the terminal score, which is only available at the end of gameplay. Its advantage normalizes the terminal score across the rollouts from the same starting position, as in GRPO (Shao et al., 2024). SCOREBOARD rewards use the current scoreboard value from Freeciv as a dense reward signal. In order to use CIVTELESCOPE as a reward, we convert its pairwise preferences into a scalar potential

with a *reference set* $\mathcal{B} = \{r_j\}_{j=1}^{M}$: positions r_j from nearby game turns whose terminal scores $Y(r_j)$ are known. Let $p_\theta(s \succ r)$ be the judge's probability that s leads to the better outcome. The potential is the reference score at the quantile given by the mean win probability,

$$\Phi_{\mathcal{B}}(s) = Q_{\mathcal{B}}\Big(\tfrac{1}{M}\textstyle\sum_{j=1}^{M} p_\theta(s \succ r_j)\Big), \tag{2}$$

where $Q_{\mathcal{B}}(w)$ linearly interpolates the sorted $Y(r_j)$ at quantile w. The judge supplies the ranking and the reference set supplies the scale, so Φ has the units of R, the realized terminal score of a rollout. A fixed reference set saturates as the policy improves, so the set evolves with the policy. Every N policy updates we form a candidate $\tilde{\mathcal{B}}$ from the most recent training rollouts, taking positions whose terminal scores are evenly spaced in rank and keeping the strongest bot-played positions, and accept it only if its potential still ranks held-out recent rollouts:

$$\mathcal{B}^{(n)} = \begin{cases} \tilde{\mathcal{B}}^{(n)} & \text{if } N \mid n \text{ and } \rho\big(\Phi_{\tilde{\mathcal{B}}^{(n)}}, R\big) \geq \rho_{\min}, \\ \mathcal{B}^{(n-1)} & \text{otherwise,} \end{cases} \tag{3}$$

where ρ is the Spearman correlation between the potential and the terminal score on those rollouts. Appendix C.1 gives M, N, $\rho_{\min}$, and the construction for each game turn.

For a state sequence $s_0, \ldots, s_T$, potential differences enter generalized advantage estimation (Schulman et al., 2018; Ng et al., 1999). With $\gamma = 1$ and $\lambda < 1$,

$$A_t = \sum_{k=t}^{T-1} \lambda^{k-t}\big(\Phi(s_{k+1}) - \Phi(s_k)\big) + \lambda^{T-t}\big(R - \Phi(s_T)\big). \tag{4}$$

Using $\lambda < 1$ retains information from intermediate potential differences, while the observed terminal score remains the final target. With $\lambda = 1$, the differences cancel and the estimator reduces to $A_t = R - \Phi(s_t)$, in which the potential serves only as a baseline (Schulman et al., 2018). Section 6.5 ablates this choice with a variant that uses CIVTELESCOPE only on the final position of each game.

Combining rewards to produce a hybrid reward for training. For full games, to keep the early-game signal stable, CIVMARSHMALLOW combines the CIVTELESCOPE reward with a baseline reward, using the baseline early in the game and CIVTELESCOPE later (Figure 3 right, Section 6.4); we call these the hybrid rewards, SCOREBOARD→CIVTELESCOPE and SPARSE→CIVTELESCOPE. At `rem120`, a hybrid reward increases the weight of CIVTELESCOPE as each game advances. The weight at game turn u is $w(u) = (1 + \exp(-(u - \tau)/\beta))^{-1}$, with transition turn τ and width $\beta > 0$; u_t denotes the game turn of s_t. Each rollout uses this schedule independently of the training phases.

SCOREBOARD→CIVTELESCOPE combines potentials on a common scale. We map the current score's rank among the reference positions to their terminal-reward scale, giving a calibrated scoreboard potential $\Phi_{\text{sb,cal}}$. We then use

$$\Phi_{\text{hybrid}}(s_t) = (1 - w(u_t))\Phi_{\text{sb,cal}}(s_t) + w(u_t)\Phi_{\text{judge}}(s_t) \tag{5}$$

in Equation 4. SPARSE→CIVTELESCOPE combines the group-normalized terminal advantage A^{sparse} with the judge advantage A_t^{judge} from Equation 4 as

$$A_t^{\text{hybrid}} = (1 - w(u_t))A^{\text{sparse}} + w(u_t)A_t^{\text{judge}}. \tag{6}$$

The reverse order, CIVTELESCOPE→SPARSE, replaces $w(u_t)$ with $1 - w(u_t)$. SCOREBOARD→CIVTELESCOPE→SPARSE uses SCOREBOARD early in the game, CIVTELESCOPE in the middle, and SPARSE from turn 80 (parameters in Appendix C.2).

6 EXPERIMENTS

6.1 SETUP

Model and Training Parameters. The gameplay agent is Qwen3-8B (Yang et al., 2025) with thinking disabled, and CIVTELESCOPE is fine-tuned from Qwen2.5-7B-Instruct (Qwen et al., 2025). All reward families update every parameter of the agent with the same clipped policy objective, without KL or entropy terms, and differ only in the advantage (Section 5.3). Each training phase runs 12 iterations, and each iteration plays eight rollouts from each of eight starting positions drawn from the phase's position bank. We use Adam with a constant learning rate of 10^{-6}, clipping ratios of 0.2 below and 0.28 above, sampling temperature 1.0, and $\lambda = 0.9$ in Equation 4.

Evaluation. We construct the evaluation sets from 45 held-out games that are unseen during training: each training phase `remX` has its own evaluation set $\mathrm{E}^{\texttt{remX}}$, the 45 positions at the turn at which that phase begins, each played to the end of the game. A gameplay agent checkpoint plays on E, the evaluation set of the current phase, before and after each training phase (from the $\text{phase}_{\text{start}}$ to the $\text{phase}_{\text{end}}$ checkpoint), eight games per position. Its *terminal score* $S = \frac{1}{|\mathrm{E}|}\sum_{p\in\mathrm{E}} Y_\pi(p)$ averages $Y_\pi(p)$, the agent's mean terminal score over the eight games from position p (Section 5.1). The *phase gain* $G = S_{\text{phase}_{\text{end}}} - S_{\text{phase}_{\text{start}}}$ is the change in S over a training phase, and g_p is that change at position p alone. The *paired win rate* $W_{a,b} = \frac{1}{|\mathrm{E}|}\sum_{p\in\mathrm{E}} \mathbb{1}[g_p^a > g_p^b]$ is the fraction of positions on which method a gains more than method b. In this work, we primarily care about G. CIVTELESCOPE quality is measured separately by accuracy on held-out trap pairs. In order to mitigate position bias (Zheng et al., 2023; Wang et al., 2024a), we query the judge twice on every pair, once with each position listed first, and average the two results.

6.2 CIVTELESCOPE RESULTS

CIVTELESCOPE vs other judges. Table 1 compares CIVTELESCOPE with the scoreboard, zero-shot open and frontier models, a Qwen2.5-7B-Instruct judge distilled from the preferences of GPT-5.6-sol (OpenAI, 2026), and a regression value head on the same backbone, as in gameplay RL (Silver et al., 2016; 2018), on held-out games and on an unseen ruleset (trained on civ2civ3, tested on classic). On held-out games, CIVTELESCOPE ranks 77.5% of trap pairs correctly, where its own zero-shot backbone and five frontier models rank 26–34% (Figure 1b). It is also the most accurate on non-trap pairs, 0.878 against 0.77–0.82, so it does not simply reverse the score order (Appendix Table 11). A regression value head trained on the same positions with replay outcomes as targets is at chance on held-out trap pairs (0.520, against 0.688 for CIVTELESCOPE on the same pairs; Appendix D.3); on the unseen ruleset the two are not distinguishable on trap pairs (0.622 and 0.592, overlapping intervals), and CIVTELESCOPE stays ahead on non-trap pairs (0.741 against 0.703). Out of domain with unseen rules, CIVTELESCOPE reaches 0.592 on classic trap pairs, against 0.241 for its backbone and 0.20–0.49 for five frontier models.

Table 1: Trap accuracy on held-out games (in-domain) and an unseen ruleset; chance is 0.5. ↑: higher is better. †No significant difference between the two marked values.

Method	Held-out games ↑	Unseen ruleset ↑
Baselines		
SCOREBOARD	0	0
7B backbone, zero-shot	0.260	0.241
GPT-5.6-luna, zero-shot	0.258	0.203
GPT-5.5, zero-shot	0.262	0.285
GPT-5.6-terra, zero-shot	0.293	0.268
GPT-5.6-sol, zero-shot	0.330	0.250
GPT-6-astra, zero-shot	0.343	0.488
Preference-labeled judge (7B)	0.391	0.415
Regression value head (7B)	0.520	**0.622**†
Replay-grounded judge		
CIVTELESCOPE	**0.775**	**0.592**†

6.3 CIVMARSHMALLOW VS. BASELINE REWARDS ACROSS TRAINING PHASES

Table 2 reports the held-out phase gain G of the two baseline rewards and of CIVMARSHMALLOW with the CIVTELESCOPE reward in each training phase. CIVTELESCOPE alone helps most in the middle game: at `rem80` it gains 8.94 points, against 5.04 for SPARSE and 0.83 for SCOREBOARD, and its paired win rate W is 81% against sparse and 96% against scoreboard, with both differences positive in every seed (Table 2). SCOREBOARD is a good proxy for the terminal score towards the end of the game (`rem20`) or right at the beginning (`rem120`), whereas SPARSE rewards are good towards the end of the game as well, at `rem40`.

We note that improvements from CIVTELESCOPE are most prominent where scoreboard traps are common (Appendix Table 12). In the policy's own 120-turn training games, rollouts from the same start are scoreboard traps 18–22% of the time over turns 30–69, where `rem80` begins, whereas `rem40` and `rem20` begin after the rate has fallen to 10% and 7% (Appendix Table 10). CIVTELESCOPE thus gives the best reward where traps are common, which motivates the hybrid rewards of CIVMARSHMALLOW for full games: a baseline reward early in the game and CIVTELESCOPE later.

Table 2: Held-out phase gain G (↑: higher is better) in each training phase: mean over three training seeds, with per-seed values in parentheses; each value averages the 45 positions of the evaluation set. Bold marks the best mean in each phase.

Reward	rem20	rem40	rem80	rem120
Baselines				
SPARSE	0.53 (1.1/0.9/−0.4)	**7.11** (7.2/7.0/7.1)	5.04 (4.7/3.0/7.4)	2.87 (4.3/4.6/−0.3)
SCOREBOARD	**2.72** (1.4/3.3/3.5)	3.68 (7.0/1.9/2.2)	0.83 (−0.1/1.4/1.1)	4.87 (5.8/6.3/2.5)
CIVMARSHMALLOW (Ours)				
CIVTELESCOPE	0.62 (1.9/−0.3/0.3)	3.61 (7.4/3.9/−0.4)	**8.94** (5.4/9.9/11.6)	2.77 (5.1/2.6/0.6)
SCOREBOARD→CIVTELESCOPE	–	–	–	**5.21** (6.2/5.5/3.9)
SPARSE→CIVTELESCOPE	–	–	–	5.15 (7.2/5.2/3.1)

Table 3: Terminal score S and gameplay statistics on the 45 held-out full-game positions (120 turns, eight games per position); trained policies are evaluated after the final phase (rem120). *Eliminated*: % of games in which the agent's player is destroyed; *invalid actions*: % of decisions rejected by the interface. ↑/↓: higher/lower is better.

Method	Terminal score S ↑	Eliminated (%) ↓	Invalid actions (%) ↓	Calls per decision
Prompting agents				
Base model (Qwen3-8B)	7.38	30.0	6.0	1.00
Direct prompting	7.62	26.2	5.7	1.00
BaseLang (Qi et al., 2024)	11.65	12.4	16.2	1.00
Mastaba (Qi et al., 2024)	11.30	16.2	24.7	4.00
SAGA (Jin et al., 2026)	13.23	8.8	41.6	1.10
Reflexion (Shinn et al., 2023)	10.22	15.1	36.7	1.01
RL baselines				
SPARSE	23.21	4.2	0.3	1.00
SCOREBOARD	23.29	**3.5**	0.3	1.00
CIVMARSHMALLOW (Ours)				
CIVTELESCOPE	21.30	9.7	**0.1**	1.00
SCOREBOARD→CIVTELESCOPE	22.37	6.0	0.5	1.00
SPARSE→CIVTELESCOPE	**23.35**	4.0	0.4	1.00

6.4 CIVMARSHMALLOW WITH HYBRID REWARDS FOR FULL GAMES

Hybrid rewards gain the most over the full game (rem120): SCOREBOARD→CIVTELESCOPE gains 5.21 points and SPARSE→CIVTELESCOPE 5.15, against 4.87 for SCOREBOARD, 2.87 for SPARSE, and 2.77 for CIVTELESCOPE alone (Table 2). Only pure CIVTELESCOPE starts from the same checkpoint as the hybrids, and both hybrids beat it in all three seeds, by 2.4 points on average.

Early in the game the score tracks growth, while CIVTELESCOPE cannot yet tell positions apart: in turns 10–29 its potential has a rank correlation of 0.09–0.10 with the terminal score, against 0.34–0.36 for the scoreboard (Appendix Table 12). From turn 30 on its correlation rises to 0.41–0.60 while traps remain common (Appendix Table 10), so the hybrid rewards hand over to CIVTELESCOPE at turn 35.

CIVMARSHMALLOW vs. prompting agents. Table 3 compares the terminal score S of the trained policies with five prompting-based agents re-implemented for our interface (Appendix D.6); all agents use Qwen3-8B (Yang et al., 2025) and the same action interface. Every trained policy outscores every prompting agent: the strongest CIVMARSHMALLOW policy, trained with the hybrid reward SPARSE→CIVTELESCOPE, reaches 23.4 against 13.2 for the best scaffold, and it is eliminated in 4% of games against 9–26%. The trained policies also almost never propose an illegal action that is not allowed in the gameplay (0.1–0.5% of decisions), where the scaffolds reach 41.6%.

Table 4: Ablations. Each block changes one component and keeps the rest fixed. Trap accuracy is evaluated on the 600 held-out trap pairs. ↑: higher is better.

Component	Variant	Metric	Result
Label source	Replay outcomes (ours)	Trap accuracy ↑	0.610
	GPT-5.6-sol preferences		0.391
Judge input	Full state (ours)	Trap accuracy ↑	0.778
	Score numbers masked		0.687
	Score numbers only		0.113
Hybrid order	SCOREBOARD→CIVTELESCOPE (ours)	Phase gain G at `rem120` ↑	5.21
	SPARSE→CIVTELESCOPE (ours)		5.15
	SCOREBOARD→CIVTELESCOPE→SPARSE		2.84
	CIVTELESCOPE→SPARSE		1.76

6.5 ABLATIONS

Label source. Replacing the replay oracle's rewards with a frontier model's preferences brings the trap back. We train two judges that are identical except for their labels: one learns from replay outcomes, the other from GPT-5.6-sol's preferences on the same pairs, as in Motif (Klissarov et al., 2024). The preference-distilled judge reaches 0.391 on held-out trap pairs, near its labeler's own 0.330, whereas the replay-labeled judge reaches 0.610 (Table 4).

Inputs provided to CIVTELESCOPE. As mentioned in Section 5.2, CIVTELESCOPE's input is a text description of a position: every player's score together with the evaluated player's gold, technologies, cities, and units. With every score number masked, trap accuracy is 0.687, where the zero-shot backbone reaches 0.275 on the full input; with only the score numbers left and without textual description, it falls to 0.113 while non-trap accuracy stays at 0.842 (Table 4). A score-only judge is right on non-trap pairs and wrong on trap pairs, which is the scoreboard heuristic itself; the rest of the position is what lets CIVTELESCOPE escape it. Therefore, CIVTELESCOPE does use the visible scores but does not reduce to them.

Using CIVTELESCOPE as a reward. Applying CIVTELESCOPE only to the final position of a game, instead of at every stage of the game, removes its benefit: at `rem40` the training score of that variant, which is SPARSE with the terminal score replaced by $\Phi(s_T)$, does not improve (-0.13 points per iteration, two-seed mean), where CIVTELESCOPE at every decision improves by 0.44 (Appendix D.8).

Ordering of hybrid rewards in CIVMARSHMALLOW. Reversing the order of the rewards within a game removes the hybrid rewards' gain. The hybrid rewards of Section 5.3, SCOREBOARD→CIVTELESCOPE and SPARSE→CIVTELESCOPE, gain 5.21 and 5.15 points at `rem120`. CIVTELESCOPE→SPARSE gains 1.76, and SCOREBOARD→CIVTELESCOPE→SPARSE, which returns to SPARSE at turn 80, gains 2.84 (Table 4). CIVTELESCOPE should therefore take over after the early game and keep its weight to the end.

7 CONCLUSION

We show that current game scores frequently disagree with eventual outcomes, creating *scoreboard traps* that are also difficult for off-the-shelf LLM judges to recognize. We introduce CIVHARNESS, which branches and replays saved game states to obtain grounded long-horizon feedback, and use these replay outcomes to train CIVTELESCOPE, a judge that more reliably distinguishes positions by their future value. We then use CIVTELESCOPE as a dense reward within CIVMARSHMALLOW, a phased training recipe for Freeciv agents. We show that replay-grounded rewards are especially useful in the middle of the game and CIVTELESCOPE can already rank positions, while combining them with simpler signals yields stronger full-game training. More broadly, our findings suggest that for long-horizon agents, useful reward signals need not come directly from what looks good now: grounding feedback in realized future outcomes can provide a better basis for learning gameplay.

AI USE

In this work, we used generative AI tools, primarily Codex and Claude Code, to implement methods, execute experiments, implement analyses selected by the authors, and parse experimental output logs. We have not used generative AI tools to develop research ideas or hypotheses or to formulate mathematical claims or proofs; the intellectual ideas and choice of analyses originated with the authors. Additionally, we used generative AI tools to create and edit code and improve the paper's readability. We did not use generative AI tools for literature searches. The use of language-model agents within gameplay simulations is part of the experimental methodology described in the paper. The authors manually verified all written text and citations and checked all experiments for correctness. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Austin Tudor David Andrews, Liam Wilkinson, Jamie Heagerty, Harry Coppock, Jakob Nicolaus Foerster, and Rui Ponte Costa. Civbench: A long-horizon benchmark for tool-mediated agents in civilization vi, 2026. URL https://arxiv.org/abs/2609.02459.

Hao Bai, Yifei Zhou, Li Erran Li, Sergey Levine, and Aviral Kumar. Digi-q: Learning VLM q-value functions for training device-control agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=CjfQssZtAb.

Nikolas Belle, Dakota Barnes, Alfonso Amayuelas, Ivan Bercovich, Xin Eric Wang, and William Wang. Agents of change: Self-evolving LLM agents for strategic planning. *arXiv preprint arXiv:2506.04651*, 2025. URL https://arxiv.org/abs/2506.04651.

Meng Cao, Lei Shu, Lei Yu, Yun Zhu, Nevan Wichers, Yinxiao Liu, and Lei Meng. Enhancing reinforcement learning with dense rewards from language model critic. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9119–9138, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.515. URL https://aclanthology.org/2024.emnlp-main.515/.

John Chen, Sihan Cheng, Can Gurkan, Ryan Lay, and Moez Salahuddin. Vox deorum: A hybrid llm architecture for 4x / grand strategy game ai – lessons from civilization v, 2025. URL https://arxiv.org/abs/2512.18564.

John Chen, Sihan Cheng, Can Gurkan, and Mingyi Lin. Civbench: Progress-based evaluation for LLMs' strategic decision-making in civilization v. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=8eM14jMVDh.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=QXEhBMNrCW.

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg (eds.), *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pp. 482–495. PMLR, 13–15 Nov 2017. URL https://proceedings.mlr.press/v78/florensa17a.html.

Freeciv developers. Freeciv 3.2.5. Open-source software, 2026. URL https://github.com/freeciv/freeciv/tree/R3_2_5. GNU General Public License, version 2 or later (GPL-2.0-or-later).

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization, 2022. URL https://arxiv.org/abs/2210.10760.

Tianyu Jin, Shuo Chen, Yida Wang, Liuyu Xiang, Yingzhuo Liu, Yexin Li, Peipei Li, and Zhaofeng He. Saga: Scene-aware, goal-evolving agents for long-horizon strategy game planning, 2026. URL https://arxiv.org/abs/2606.29932.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=Myx2kJFzAn.

Martin Klissarov, Pierluca D'Oro, Shagun Sodhani, Roberta Raileanu, Pierre-Luc Bacon, Pascal Vincent, Amy Zhang, and Mikael Henaff. Motif: Intrinsic motivation from artificial intelligence feedback. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=tmBKIecDE9.

Heinrich Küttler, Nantas Nardelli, Alexander Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The nethack learning environment. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 7671–7684. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/569ff987c643b4bedf504efda8f786c2-Paper.pdf.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Bo Liu, Simon Yu, Zichen Liu, Leon Guertler, Penghui Qi, Daniel Balcells, Mickel Liu, Cheston Tan, Weiyan Shi, Min Lin, Wee Sun Lee, and Natasha Jaques. SPIRAL: Self-play on zero-sum games incentivizes reasoning via multi-agent multi-turn reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=7Yayy5fNLg.

Walter Mischel, Ebbe B. Ebbesen, and Antonette R. Zeiss. Cognitive and attentional mechanisms in delay of gratification. *Journal of Personality and Social Psychology*, 21(2):204–218, 1972. doi: 10.1037/h0032198.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. ISSN 1476-4687. doi: 10.1038/nature14236. URL https://doi.org/10.1038/nature14236.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pp. 278–287, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1558606122.

OpenAI. GPT-5.6 preview system card. https://deploymentsafety.openai.com/gpt-5-6-preview, 2026.

Davide Paglieri, Bartłomiej Cupiał, Samuel Coward, Ulyana Piterbarg, Maciej Wolczyk, Akbir Khan, Eduardo Pignatelli, Łukasz Kuciński, Lerrel Pinto, Rob Fergus, Jakob Nicolaus Foerster, Jack Parker-Holder, and Tim Rocktäschel. BALROG: Benchmarking agentic LLM and VLM reasoning on games. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=fp6t3F669F.

Siyuan Qi, Shuo Chen, Yexin Li, Xiangyu Kong, Junqi Wang, Bangcheng Yang, Pring Wong, Yifan Zhong, Xiaoyuan Zhang, Zhaowei Zhang, Nian Liu, Yaodong Yang, and Song-Chun Zhu. Civrealm: A learning and reasoning odyssey in civilization for decision-making agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=UBVNwD3hPN.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin

Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Noam Razin, Zixuan Wang, Hubert Strauss, Stanley Wei, Jason D. Lee, and Sanjeev Arora. What makes a reward model a good teacher? an optimization perspective. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=7pufO0SJAC.

Tim Salimans and Richard Chen. Learning montezuma's revenge from a single demonstration, 2018. URL https://arxiv.org/abs/1812.03381.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2018. URL https://arxiv.org/abs/1506.02438.

John Schultz, Jakub Adamek, Matej Jusup, Marc Lanctot, Michael Kaisers, Sarah Perrin, Daniel Hennes, Jeremy Shar, Cannada A. Lewis, Anian Ruoss, Tom Zahavy, Petar Veličković, Laurel Prince, Satinder Singh, Eric Malmi, and Nenad Tomasev. Mastering board games by external and internal planning with language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=KKwBo3u3IW.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for LLM reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=A6Y7AqlzLW.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. ISSN 1476-4687. doi: 10.1038/nature16961. URL https://doi.org/10.1038/nature16961.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018. doi: 10.1126/science.aar6404. URL https://www.science.org/doi/abs/10.1126/science.aar6404.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9440–9450, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.511. URL https://aclanthology.org/2024.acl-long.511/.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of*

the Association for Computational Linguistics (Volume 1: Long Papers), pp. 9426–9439, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Yufei Wang, Zhanyi Sun, Jesse Zhang, Zhou Xian, Erdem Biyik, David Held, and Zackory Erickson. RL-VLM-f: Reinforcement learning from vision language foundation model feedback. In *Forty-first International Conference on Machine Learning*, 2024c. URL https://openreview.net/forum?id=YSoMmNWZZx.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning, 2025. URL https://arxiv.org/abs/2504.20073.

Yu Xia, Jingru Fan, Weize Chen, Siyu Yan, Xin Cong, Zhong Zhang, Yaxi Lu, Yankai Lin, Zhiyuan Liu, and Maosong Sun. AgentRM: Enhancing agent generalization with reward modeling. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19277–19290, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.945. URL https://aclanthology.org/2025.acl-long.945/.

Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. Text2reward: Reward shaping with language models for reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=tUM39YTRxH.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. Free process rewards without process labels. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 73511–73525. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/yuan25c.html.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=uccHPGDlao.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. ArCHer: Training language model agents via hierarchical multi-turn RL. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62178–62209. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/zhou24t.html.

APPENDIX CONTENTS

A Background **15**

A.1 Freeciv rulesets 15

A.2 Cross-game trap probe 15

A.3 Game-length curve 16

A.4 Within-game trap rate 16

B CIVHARNESS **17**

B.1 Harness acceptance tests 17

C Training a Freeciv gameplaying agent **18**

C.1 Judge training and calibration 18

C.2 Hybrid schedules at `rem120` 18

D Experiments **19**

D.1 Aggregation 19

D.2 Judge evaluation and zero-shot judges 19

D.3 Regression value head protocol 19

D.4 Ruleset transfer protocol 19

D.5 Offline window diagnostic 20

D.6 Prompting-scaffold panel 20

D.7 Label-source ablation protocol 20

D.8 Additional mechanism controls 21

Table 5: The future outcome behind each Freeciv measurement. Every row uses the evaluated player's terminal score: K branches are played from the position to the configured final turn by the built-in bots at the *normal* bot difficulty and averaged, or a policy rollout's own terminal score is taken. The judge's held-out set is that of Tables 11 and 1 (in-domain column). Cross-game pairs join positions of different games at most ten turns apart; a pair is decidable when its outcome gap exceeds twice the pooled standard error of the branch means, or, for single rollouts, is at least six points.

Measurement	Positions	Played out by	K	Game length	Decidable pairs
Fig. 2, CIV2CIV3; Fig. 1a	40 games, turns 10–115	bots	8	120 turns	cross-game, $>$ 2 SE
Fig. 2, CLASSIC	40 games, turns 10–115	bots	8	120 turns	cross-game, $>$ 2 SE
Ruleset transfer (Table 1)	same games, turns 10–65	bots	8	120 turns	cross-game, $>$ 2 SE
Table 9	10 games per length	bots	8	60, 70, 80 turns	cross-game, $>$ 2 SE
Judge held-out set	bot games; rollouts	bots; the rollout	8; 1	70 turns	cross-game, $>$ 2 SE; $\geq$ 6
Table 10	training rollouts	the rollout	1	120 turns	same start and turn, $\geq$ 6

A Background

A.1 Freeciv rulesets

A Freeciv ruleset is a set of data files that defines the units, buildings, technologies, governments, terrain, and game parameters, so the same server plays different games under different rulesets. We use two rulesets shipped with Freeciv 3.2.5. CIV2CIV3, the default ruleset of the Freeciv 3 server, keeps the rules of Civilization II as its reference and moves the game toward Civilization III, aiming at slower and more linear city development; CLASSIC is the older standard ruleset from which it departs. Relative to CLASSIC, CIV2CIV3 removes rapture growth, so celebrating cities no longer grow every turn; makes Settlers cost two citizens instead of one; adds food upkeep for most units; enables plague in cities larger than four; makes a technology cheaper when a nation the player has an embassy with already knows it, and charges a per-city research upkeep late in the game; and requires at least three tiles between cities. Scoreboard traps occur under both rulesets, in 26% of decidable position pairs under CIV2CIV3 and 27% under CLASSIC (Figure 2(a)). The main experiments, the judge's training data, and its held-out evaluation use CIV2CIV3; CLASSIC is used only to test the judge on unseen rules (Appendix D.4).

A.2 Cross-game trap probe

Every Freeciv trap rate and trap pair in the paper uses the terminal target of Section 5.1, the mean terminal score of the evaluated player; Table 5 lists the continuation policy, branch count, game length, and pair rule behind each.

The chess bank holds 541 positions from 80 played games, seven snapshots per game, each valued by 20 stochastic Stockfish replays to the end with the mean result for the side to move as the oracle and material balance in pawns as the visible proxy. The Othello bank holds 420 positions from 70 games, six snapshots per game, each valued by 24 stochastic replays with the final disc differential as the oracle and the current disc differential as the proxy. Pairs are built exactly as for Freeciv: across games, bounded move gap, decidable when the oracle gap exceeds twice the pooled standard error. The zero-shot evaluation uses 300 trap and 300 non-trap pairs per game, both presentation orders, using the Freeciv prompt with only the game name changed. Calls that time out are excluded rather than scored. Table 6 reports the per-judge results.

Five further banks (Table 7) apply the same pipeline to the OpenSpiel implementations of backgammon, 9×9 Go, Hearts, and 2048, and to the Catanatron implementation of Settlers of Catan. Positions are snapshots of self-play by a deliberately weak stochastic engine, Monte Carlo tree search with 30 simulations of one random rollout each and 10% uniformly random moves, which plays the role of the skill-level noise of Stockfish in chess. 2048 uses a one-ply greedy policy with the same random fraction and Hearts uniform random play, because tree search with terminal rewards applies neither to a single-player game with intermediate rewards nor to an imperfect-information game. Each position is replayed $K{=}100$ times by the same engine ($K{=}48$ for Catan, where every snapshot yields four positions, one per seat, sharing the same continuations). The visible proxy is the number the game itself displays: the pip-count lead in backgammon, the naive area count with komi in Go, penalty

Table 6: Zero-shot accuracy of the five API judges on chess and Othello by pair slice. Each game uses 300 trap and 300 non-trap pairs in both presentation orders, with the Freeciv prompt and only the game name changed; chance is 0.5.

	Chess		Othello	
Judge	Non-trap	Trap	Non-trap	Trap
GPT-5.4-mini	0.938	0.157	0.847	0.188
DeepSeek-V3.2	0.862	0.315	0.815	0.263
GPT-5.2	0.892	0.352	0.785	0.637
Claude Sonnet 5	0.883	0.485	0.797	0.522
GPT-5.6-sol	0.677	0.598	0.492	0.813

Table 7: Trap prevalence in every cross-game bank. Positions are snapshots of self-play games between copies of the named engine (80 chess, 70 Othello, 40 backgammon, 59 Go, 60 Hearts, 60 2048, and 40 Catan games); the oracle replays each position K times to the end with the same engine and takes the mean outcome. Pairs are built as in Freeciv: across games, bounded move gap, decidable when the oracle gap exceeds twice its pooled standard error; the trap rate is taken over decidable pairs whose visible proxy is not tied. Zero-shot judges are evaluated on the chess and Othello banks only (Table 6).

Game (genre)	Visible proxy	Engine	Positions	K	Pairs	Trap rate
Chess (board)	material balance	Stockfish, skill-level noise	541	20	34,024	16.0%
Othello (board)	disc differential	depth-2 search + temperature	420	24	18,334	68.2%
Backgammon (board, dice)	pip-count lead	MCTS-30 + 10% random	187	100	6,587	8.1%
Go (9x9) (board)	area count with komi	MCTS-30 + 10% random	367	100	6,182	38.8%
Hearts (card)	penalty so far vs. table	uniform random	300	100	14,543	17.7%
2048 (single-player puzzle)	score	one-ply greedy + 10% random	381	100	12,042	31.2%
Catan (4-player economic)	public victory points	victory-point greedy bots	952	48	79,737	14.4%

points relative to the table in Hearts, the running score in 2048, and public victory points in Catan, whose rules hide victory-point cards from the opponents so that the oracle, the same player's final victory points, includes what the proxy cannot see. Where the game has a natural terminal quantity of the same kind, the oracle is that quantity (area margin, penalty, score, victory points); in backgammon it is the mean result. Every bank, pair set, and engine setting is reproducible from the repository. For the progress axis of Figure 2b a pair is placed at the mean position of its two snapshots divided by the typical length of a complete game under the same engine (last row of the figure), and bins with fewer than 30 pairs are left blank. Table 8 lists the rate in each of the ten progress bins. The Freeciv rows use the first two banks of Table 5, built with one protocol: 40 games of 120 turns per ruleset, labeled every five turns from turn 10 to 115, 40 new CIV2CIV3 games (57,846 decidable pairs with untied scores) and the 40 CLASSIC games of the ruleset transfer (55,873 such pairs).

A.3 GAME-LENGTH CURVE

For each game length in Table 9 we generated ten fresh small-map games from disjoint seed pools and labeled every fifth turn with the replay oracle at $K{=}8$ branches at the *normal* bot difficulty. Per turn bucket we correlate the current scoreboard and the oracle value with the actual terminal score of the real game. At 80 turns the oracle's own early-bucket correlation is 0.10–0.39, so early positions in long games need more branches or labels built phase by phase; the training runs use reference sets rebuilt for the phase being trained for this reason.

A.4 WITHIN-GAME TRAP RATE

The cross-game pair rule of Section 5.1 keeps the turn number from leaking into the comparison, but it raises the question of whether the trap merely reflects incomparable maps. We therefore measure the trap rate on pairs of policy rollouts that share the same start, that is the same map, seed, opponents, and save, and are compared at the same turn, so that the two positions differ only in what the focal policy did. The rollouts are the archived training games of 16 arms from the scoreboard, judge, and

Table 8: Trap rate (%) in each progress bin of Figure 2b. Progress is the mean position of a pair divided by the typical length of a complete game; a dash marks a bin with fewer than 30 pairs: no points are scored while cards are passed in Hearts, and the Freeciv banks cover turns 10–115 of 120-turn games (Table 5).

Progress (%)	0–10	10–20	20–30	30–40	40–50	50–60	60–70	70–80	80–90	90–100
Backgammon	10	5	6	8	9	10	9	11	9	6
Catan	25	16	15	16	12	11	11	12	13	12
Chess	6	19	21	17	12	9	5	3	4	2
Hearts	–	18	20	23	22	18	12	8	7	4
Freeciv (civ2civ3)	55	49	40	35	29	28	24	18	10	8
Freeciv (classic)	41	46	46	35	31	28	24	17	11	7
2048	73	37	38	38	29	30	41	32	24	19
Go (9x9)	0	13	21	45	58	46	37	30	21	15
Othello	61	50	72	72	69	65	59	47	43	33

Table 9: The scoreboard predicts the terminal score less well in longer games. For ten games at each game length, we correlate the current scoreboard and the replay-oracle value (K=8 branches) with the game's terminal score at every fifth turn, and average the correlations over turns. "Turns won" counts the sampled turns at which the oracle value correlates at least as well as the scoreboard; the trap rate is the share of decidable position pairs whose scoreboard order disagrees with their terminal-score order.

Game length	Scoreboard corr.	Oracle corr.	Turns won by oracle	Trap rate
60 turns	0.81	0.98	7 / 10	30%
70 turns	0.53	0.95	11 / 12	31%
80 turns	0.12	0.58	12 / 14	54%

hybrid families in every training phase and seed, forming 640 arm-by-start clusters, with games that end in elimination excluded. A pair counts when the realized terminal scores differ by at least six points, the rule used for the judge's policy-rollout training pairs; a trap is a pair whose leader on the visible score finishes behind, and pairs with tied visible scores are dropped. Cross-start pairs from the same rollouts at the same turn give the comparison. Table 10 reports both rates by turn bucket. Same-start pairs are traps 18–23% of the time through turn 69 and 7–10% afterwards, and from turn 30 on they are traps two to ten times more often than cross-start pairs, because start luck no longer explains the score gap. The trap is therefore not an artifact of cross-game pairing.

B CivHarness

B.1 Harness acceptance tests

The harness acceptance suite checks replica identity, deterministic re-seeded branching, a known-order positive control, branch-level noise floors, turn-matched readouts, faithful client actions, restricted observation/action spaces, and complete game replay. A branch is excluded from an analysis if its provenance digest or restored opponent configuration fails validation.

Replaying the same save, branch seed, and action trace must reproduce the same state digest. An `EnvSpec` separates the raw engine state from the observation and action space exposed to an agent, allowing selected economic, technology, diplomacy, unit, and terrain fields to be exposed without access to the raw save. Game logs include the observation digest and restored bot difficulty so that a stochastic language-model run remains replayable.

Table 10: Within-game trap rate on the training rollouts. Pairs of rollouts from the same start, compared at the same turn, whose realized terminal scores differ by at least six points; a trap is a pair whose leader on the visible score finishes behind, with tied visible scores dropped. Cross-start pairs are drawn from the same rollouts at the same turn under the same rule. Intervals are cluster bootstraps over 640 arm-by-start clusters.

Turn bucket	Same-start trap rate	95% interval	Same-start pairs	Cross-start trap rate (pairs)
10–29	0.233	[0.173, 0.304]	7,255	0.298 (95,108)
30–49	0.179	[0.152, 0.208]	31,384	0.113 (216,768)
50–69	0.219	[0.199, 0.241]	104,653	0.037 (327,427)
70–89	0.102	[0.091, 0.114]	119,881	0.019 (348,151)
90–119	0.071	[0.063, 0.079]	389,279	0.010 (742,316)

C Training a Freeciv gameplaying agent

C.1 Judge training and calibration

The judge is evaluated on 600 non-trap and 600 trap pairs from games and rollouts excluded from training. For each sampled game turn of the phase being trained, the reference set of Equation 2 is drawn from eight positions with known terminal scores: six from policy rollouts, evenly spaced in rank by terminal score, and the two highest-scoring bot-played positions. During training $\mathcal{B}$ holds four of the eight, chosen to span the score range (M=4), and the judge is queried in one presentation order. In the JUDGE and hybrid arms a candidate set is rebuilt every N=4 policy updates from 40 recent training rollouts and validated on the 40 rollouts that follow them. The candidate replaces the set in use only if the Spearman correlation between its potential and the terminal score is at least $\rho_{\min}$=0.2 both at turn 40 and at the last decision (Equation 3); the set never changes within a policy update. The potential is sensitive to the score range of the reference set. In the `rem120` phase of seed 44, replacing only the reference set, from one whose four scores ranged from 7 to 33 to one ranging from 2 to 36, turned the training-score slope from -0.06 to $+0.06$ points per iteration; on seed 43 the slope was positive with both sets (0.10 and 0.16).

On the terminal-score label set, full-parameter SFT memorizes the training set and lands at chance on held-out pairs while dismantling the base model's score-reading prior; LoRA with rank 16 on the same data, labels, and prompt generalizes, and it is stable across three training seeds. Adding more pairs beyond about seven thousand does not raise held-out accuracy.

C.2 Hybrid schedules at REM120

SCOREBOARD→CIVTELESCOPE, SPARSE→CIVTELESCOPE, and CIVTELESCOPE→SPARSE (Section 5.3) use transition turn $\tau = 35$ and width $\beta = 5$, so the judge weight is $w(u) = (1 + \exp(-(u - 35)/5))^{-1}$. SCOREBOARD→CIVTELESCOPE→SPARSE first blends the SCOREBOARD and CIVTELESCOPE potentials with $w(u_t)$, then shifts from the resulting dense advantage to the sparse advantage using a second sigmoid centered at turn 80 with scale 5.

Tables 2 and 4 list the `rem120` phase gain of every schedule. Every hybrid and the pure judge arm start from the judge $\text{phase}_{\text{end}}^{\texttt{rem80}}$ checkpoint of their seed; the pure sparse and scoreboard arms start from their own $\text{phase}_{\text{end}}^{\texttt{rem80}}$ checkpoints. CIVTELESCOPE→SPARSE was replicated in two seeds, the others in three. All eleven hybrid $\text{phase}_{\text{end}}^{\texttt{rem120}}$ checkpoints have also been rolled out on the `rem80` evaluation set $\text{E}^{\texttt{rem80}}$: SCOREBOARD→CIVTELESCOPE scores 58.4, 57.0, and 56.7 for seeds 43, 44, and 45; SPARSE→CIVTELESCOPE 58.2, 59.3, and 51.9; CIVTELESCOPE→SPARSE 58.7 and 54.4 for seeds 43 and 44; and SCOREBOARD→CIVTELESCOPE→SPARSE 59.6, 59.0, and 51.8. The base model scores 40.7 on this set and the single-signal $\text{phase}_{\text{end}}^{\texttt{rem80}}$ checkpoints 52.4–60.2. The spread across seeds follows the $\text{phase}_{\text{start}}^{\texttt{rem120}}$ checkpoint: seed 45's judge $\text{phase}_{\text{end}}^{\texttt{rem80}}$ checkpoint, from which its hybrids start, scores 52.4 on this set, and its three hybrids score 51.8–56.7.

Table 11: Trap accuracy on the judge's held-out set: 300 trap and 300 non-trap pairs from games and rollouts excluded from training, both presentation orders, content-level. Chance is 0.5. *Warned* repeats the trap pairs with a sentence in the prompt saying that the visible score can mislead and that the future outcome decides. The last column is how many times more accurate the judge is than the model on the same trap pairs, with a 95% paired bootstrap interval clustered over source games.

Judge	Non-trap pairs	Trap pairs	Warned	Times more accurate
Qwen2.5-7B-Instruct (zero-shot backbone)	0.766	0.260	—	1.98 [1.22, 3.27]
GPT-5.6-luna	0.792	0.258	0.317	2.00 [1.27, 3.25]
GPT-5.5	0.805	0.262	0.347	1.96 [1.25, 3.10]
GPT-5.6-terra	0.823	0.293	0.355	1.64 [1.03, 2.64]
GPT-5.6-sol	0.802	0.330	0.378	1.35 [0.87, 2.13]
GPT-6-astra	0.818	0.343	0.378	1.26 [0.76, 2.03]
Replay-grounded judge (same 7B base)	**0.878**	**0.775**	—	—

D EXPERIMENTS

D.1 AGGREGATION

For each starting position, the eight games are averaged before computing a phase gain, and a game in which the agent's player is eliminated counts as a score of zero. Paired win rates compare these start-level gains.

D.2 JUDGE EVALUATION AND ZERO-SHOT JUDGES

Off-the-shelf models are scored on a seeded subsample of 300 trap and 300 non-trap pairs of the judge's held-out set, with the same fog renderings and prompt as the judge; intervals are cluster bootstraps over source games, and ratios are paired over the same pairs (Table 11). The judge's accuracies on the full 600-pair slices are 0.778 and 0.866. Every pair is presented in both orders and accuracy is content-level.

D.3 REGRESSION VALUE HEAD PROTOCOL

The head shares CIVTELESCOPE's base, adapter, and recipe (Qwen2.5-7B, LoRA rank 16, two epochs, one seed) and replaces the pairwise objective with a scalar output trained to predict a position's replay outcome, the mean terminal score of its $K{=}8$ branches, under a Huber loss on standardized targets. Per-position replay outcomes are available for 2,900 positions of the judge's training bank; the head is trained on the 2,583 of them from games outside the held-out set and read after the two epochs, without snapshot selection. On a pair it picks the position with the higher predicted score. It is scored on the 200 held-out trap pairs and 197 non-trap pairs whose positions carry a per-position label, and CIVTELESCOPE and the zero-shot backbone are re-read on the identical pairs from their existing answers: the head scores 0.520 $[0.317, 0.688]$ on trap pairs and 0.548 on non-trap pairs, CIVTELESCOPE 0.688 and 0.777, and the backbone 0.219 and 0.612. The head fits its training positions (Pearson $r = 0.77$ with the replay outcome) but not held-out positions ($r = 0.16$). On the classic pairs of Appendix D.4, 1,800 trap and 600 non-trap, the head scores 0.622 $[0.481, 0.750]$ and 0.703, CIVTELESCOPE 0.592 $[0.515, 0.672]$ and 0.741, and the backbone 0.241 and 0.818. The head has no presentation order and gives one pick per pair, while CIVTELESCOPE keeps its content-level score over both orders. No classic game appears among the head's training games, and its predictions correlate weakly with the turn ($r = 0.15$) and with the visible score ($r = 0.24$).

D.4 RULESET TRANSFER PROTOCOL

The classic bank holds 40 games generated under the classic ruleset with the judge's own protocol: positions every five turns from turn 10 to 65 of 120-turn games, 480 positions in all, each labeled by $K{=}8$ branches replayed to the end of the game at the *normal* bot difficulty and rendered from the

Table 12: Offline window diagnostic on `rem120` rollouts. Per turn bucket, the median absolute error and the Spearman correlation of each potential against the realized terminal score, for two training seeds (a / b). The judge's error is below the scoreboard's from the first bucket, but its rank correlation only exceeds 0.4 from the 30–49 bucket onward, and the scoreboard becomes an identity in the last bucket.

	Replay-grounded judge		Scoreboard	
Turn bucket	Median error	Spearman ρ	Median error	Spearman ρ
10–29	7.8 / 9.6	0.09 / 0.10	15.0 / 16.0	0.34 / 0.36
30–49	4.8 / 6.2	0.42 / 0.41	12.0 / 12.0	0.64 / 0.65
50–69	3.9 / 5.1	0.60 / 0.54	8.0 / 8.0	0.78 / 0.78
70–89	3.6 / 3.9	0.64 / 0.65	6.0 / 6.0	0.86 / 0.87
90–109	2.9 / 3.0	0.74 / 0.76	3.0 / 3.0	0.92 / 0.92
110–129	2.7 / 3.2	0.77 / 0.79	0.0 / 0.0	0.98 / 0.98

focal player's view. Pairs follow the training rule (cross-game, $|\Delta\text{turn}| \leq 10$, decidable at two pooled standard errors), giving 26,819 decidable pairs of which 39.1% are traps; the evaluation set is a seeded subsample of 1,800 trap and 600 non-trap pairs, and five frontier models are scored on 200 trap and 100 non-trap pairs of it. The readouts were frozen before judging: the paired trap-accuracy difference against the zero-shot backbone, $+0.351$ $[0.248, 0.454]$ over 10,000 bootstrap draws clustered by game, and against the strongest zero-shot judge on the API subsample, with the strongest re-taken inside each draw, $+0.31$ $[0.152, 0.421]$ over the backbone and GPT-5.6-sol as frozen and $+0.07$ $[-0.08, 0.21]$ once the four further frontier models are included. The non-trap difference is -0.077 $[-0.188, 0.018]$.

D.5 Offline window diagnostic

Equation 4 consumes differences of Φ between successive decisions, so a potential helps in proportion to how well it *ranks* states, not to how small its pointwise error is. Table 12 buckets the decisions of the `rem120` rollouts of two seeds by turn and reports, per bucket, the median absolute error and the Spearman correlation of the judge potential and of the scoreboard with the realized terminal score. The judge's pointwise error is half the scoreboard's from the first bucket, yet its rank correlation is only 0.09 there; it reaches 0.54–0.60 at turns 50–69, within the `rem80` phase that begins at turn 40. The scoreboard's rank correlation exceeds the judge's in every bucket, because the quantities that make up the current score persist into the terminal score, and reaches 0.98 in the last bucket; the judge's explained variance beyond the scoreboard is largest at `rem80`, 24–26% against 2–11% in the other phases.

D.6 Prompting-scaffold panel

The five prompting-based agents in Table 3 are direct prompting; BaseLang, which writes a chain of reasoning before the action, and Mastaba, a hierarchical design with one summarizing call over several advisor calls, both from CivRealm (Qi et al., 2024); the scene-graph planner SAGA (Jin et al., 2026); and the self-reflection loop Reflexion (Shinn et al., 2023). We re-implement the four published designs on our menu interface with our own prompts. Of the 1,800 prompting games, 1,684 completed and the rest failed on infrastructure. All use the same Qwen3-8B base, the same menu interface, and the same opponents as the trained policies, on the `rem120` evaluation set $\mathrm{E}^{\texttt{rem120}}$. The trained policies are the checkpoints produced by the `rem120` phase, the last of the four training phases, on that base. Prompt cost dominates the per-call cost, so the trained policy and the scaffolds with about one call per decision cost the same; only the hierarchical advisor scaffold, with four calls per decision, is materially more expensive.

D.7 Label-source ablation protocol

The two judges are trained on the same 3,592 pairs, a seeded subset of the judge's training pairs, with the judge's recipe (Qwen2.5-7B, LoRA rank 16, two epochs, one seed); one arm keeps the replay

label of each pair, the other replaces it with GPT-5.6-sol's answer on that pair in one presentation order. The labeler agrees with the replay outcome on 0.678 of the pairs overall and on 0.489 of the trap pairs. Both arms are read at the training amount of the full judge (its 4,800 steps scaled to the subset) on the judge's held-out set, 600 trap and 600 non-trap pairs from games excluded from training. At that point the replay arm scores 0.610 $[0.481, 0.677]$ on trap pairs and 0.859 on non-trap pairs; the preference arm scores 0.391 $[0.239, 0.482]$ and 0.763. At each arm's best trap-accuracy snapshot the gap is the same, 0.692 against 0.444, and the two intervals are disjoint. On the classic trap pairs of Appendix D.4 these snapshots score 0.531 and 0.415, against 0.243 for the backbone.

D.8 Additional mechanism controls

In the terminal-judge control the judge's potential of the final position replaces the terminal score, and everything else follows SPARSE. Its readout is the improvement in training score per iteration, estimated for each starting position over the twelve iterations of a training phase and averaged over positions. On seeds 43 and 44 the control changes by -0.19 and -0.07 points per iteration and the judge applied at every decision by $+0.65$ and $+0.23$; SPARSE improves by 1.13 and 0.80 and SCOREBOARD by 0.80 and 0.19. The control starts `rem40` from the base model, whereas the three signal families continue from their $\text{phase}_{\text{end}}^{\texttt{rem20}}$ checkpoints, and it uses a fixed reference set. At `rem80` its training score recovers from its weaker $\text{phase}_{\text{end}}^{\texttt{rem40}}$ checkpoint but still ends below SPARSE on both seeds.