December 10, 2025

Multi-Agent Reinforcement Learning for Large-Scale MMO Combat

Coordination, Social Network Analysis, and Emergent Strategy in an MMO Boss Encounter Environment

By Jason Fu

Introduction

In our environment, a single autonomous Boss agent fights a coordinated party of four player-role agents in an MMO-inspired boss encounter. At the start of each episode, all four party agents spawn together on one side of a square arena, facing the Boss, who spawns at the center. The party can specialize into one of four classes, Tank, Healer, Melee DPS, and Ranged DPS, each with distinct damage, survivability, and utility properties. Class selection is self learned, where party agents choose their class through a dedicated action branch and remain locked into that role for the duration of the episode.

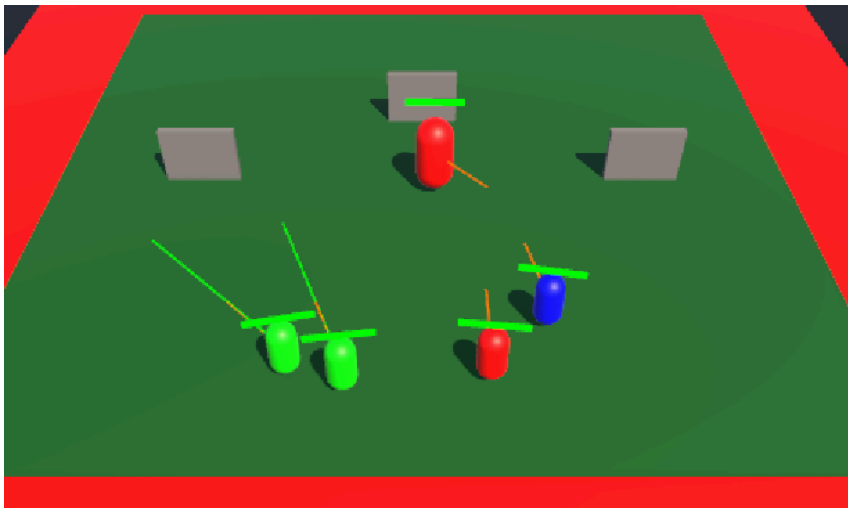


Figure A. Basic arena with a Boss agent (red capsule in center), four party agents (two green, one red, and one blue capsule facing the Boss agent), three walls around the Boss agent, and a lava moat surrounding the arena.

The arena contains several interactive and hazardous features. Three movable walls spawn near the Boss and can be picked up and repositioned by the Boss agent, allowing it to reshape the battlefield by creating cover, blocking paths, or forcing the party toward dangerous zones, as can be seen in Figure A. The arena floor is surrounded by lava that deals continuous damage and applies a lingering burn effect, and beyond the lava lies a void that causes instant death. These hazards encourage agents to learn meaningful navigation and positioning rather than static or purely greedy behaviors.

Both the Boss and party members act in discrete action spaces that control movement, rotation, attacks, and class or role specific abilities. Party agents can move forward or backward, rotate, attack, and depending on their selected class, can either heal nearby allies (Healer) or generate a large burst of threat to draw aggro (Tank). The Boss can move, rotate, attack for high burst damage, and interact with walls by picking them up or placing them elsewhere in the arena. All agents receive high-dimensional observations composed of self-state features (position, velocity, health, class, threat, cooldowns), LIDAR-style range readings over 30 rays, and compact embeddings of visible entities (other agents and walls) within a frontal vision cone.

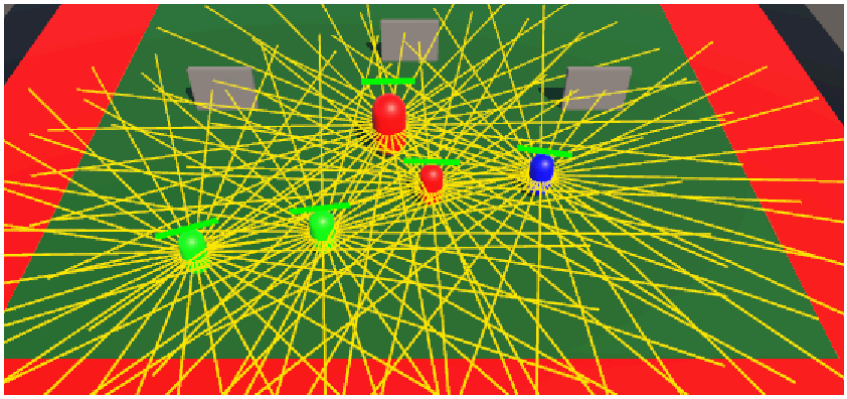


Figure B. Vision rays provide each agent with line-of-sight information about nearby entities and obstacles.

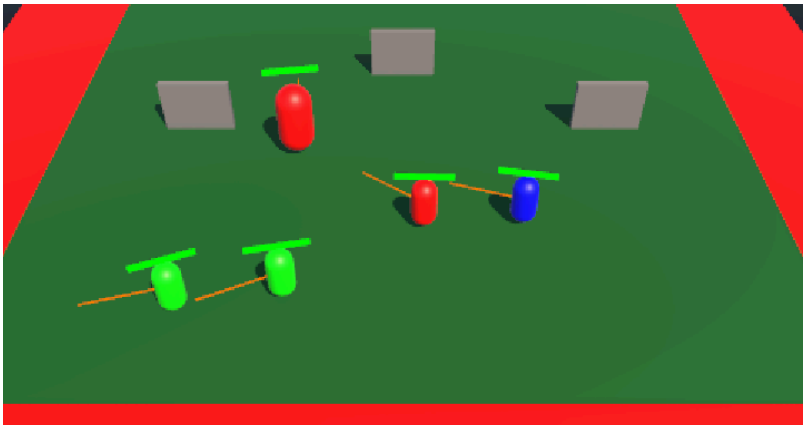


Figure C. Attack-range rays indicate when enemies fall within a valid offensive range for the agent.

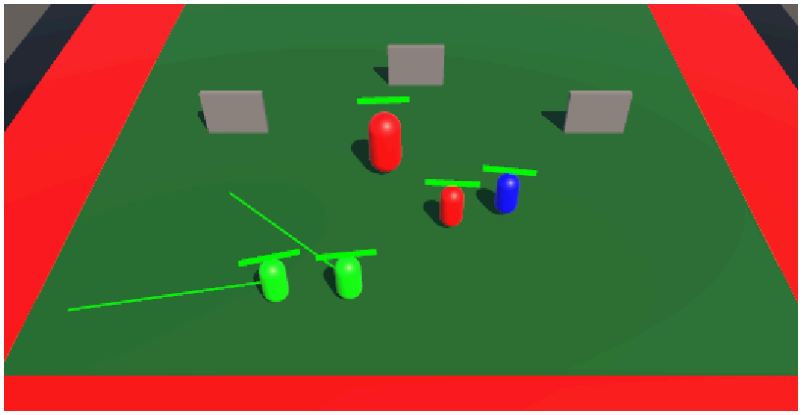


Figure D. The Healer uses a dedicated green healing ray to detect allies within healing range.

The reward structure is purely sparse and team based. Episodes end when either the Boss's health is reduced to zero (party win) or all party members are dead (Boss win), in which case the winning side receives a reward of +1 and the losing side -1. No intermediate rewards are provided for dealing damage, healing, avoiding hazards, or using abilities. All behaviors must be discovered as a way to maximize win probability over time. Episodes may also end on a time limit, in which case the outcome is treated as a loss for both sides or according to a predefined tie-breaking rule. Between episodes, all agents and walls are reset to their initial spawn positions, and health, threat, and class selections are cleared.

This design intentionally minimizes hand-crafted shaping and does not explicitly reward standard MMO conventions such as tanking, healer protection, kiting, or coordinated burst damage. The only supervision comes from the win/loss signal at the end of each episode. As a result, behaviors like stable tank aggro, healer positioning behind cover, opportunistic melee dives, ranged kiting, and Boss tool use with movable walls must emerge from the interaction between the environment dynamics and adversarial multi-agent learning, rather than being directly encoded into the reward function.

Autocurricula and emergent behavior

As the Boss and party agents train against each other in the boss-fight environment, we observe a sequence of increasingly sophisticated behaviors on both sides. Each new behavior by one side creates a new learning pressure on the other, forming an implicit curriculum without any manual scheduling. There are no explicit incentives for agents to manipulate walls, respect class fantasy, avoid lava, or manage aggro. The only supervision provided is the sparse win/loss signal at the end of each episode. The emergent strategies we describe below arise purely from this self-generated autocurriculum induced by multi-agent competition and the simple dynamics of the encounter.

We first describe the progression of behaviors that emerges in our default arena, which consists of one Boss, four party members, three movable walls near the Boss, and lava and void hazards surrounding the main platform.

Phase 0 - Random movement

At the beginning of training, both the Boss and party agents move and act at random. Party members frequently walk into lava or stand idle in front of the Boss, and the Boss attacks without regard for range, cooldowns, or target selection. Class selection, where it occurs, is effectively uniform noise over the four available roles.

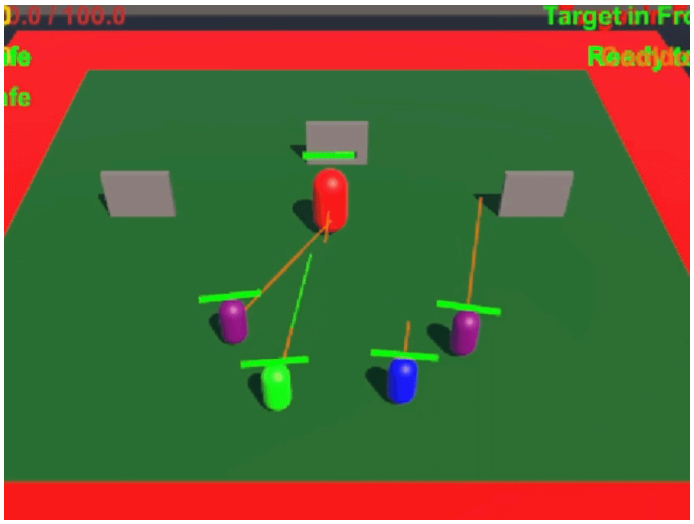


Figure E. Agents move at random and are currently untrained, resulting in expected behavior this early in the training.

Phase 1 – Naive damage racing.

The first consistent behavior to emerge is simple damage racing. Party members begin to approach the Boss more reliably and repeatedly attack in range, while the Boss learns to orient toward the nearest target and attack on cooldown. At this point, classes are still used poorly: melee and ranged agents stand in similar positions, healers rarely use their healing action effectively, and the Tank's threat tools are underutilized.

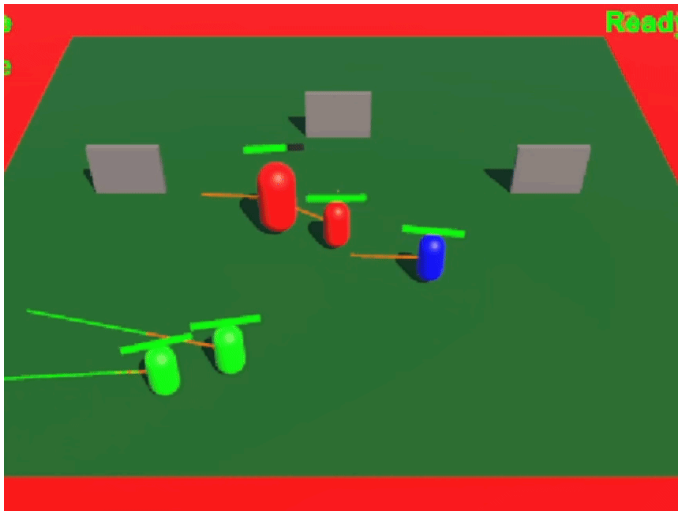


Figure F. The party agents begin to approach the boss and even kill it to score a point for their team.

Phase 2 – Basic role emergence and aggro anchoring.

As training progresses, the Melee DPS class starts to dominate class selection among party members that spend a lot of time close to the Boss. These agents learn to remain in melee range and repeatedly attack, incidentally generating enough threat to keep the Boss's focus. The Boss, in turn, increasingly tracks the highest-threat target, effectively anchoring on a front-line agent, although positioning is still largely suboptimal.

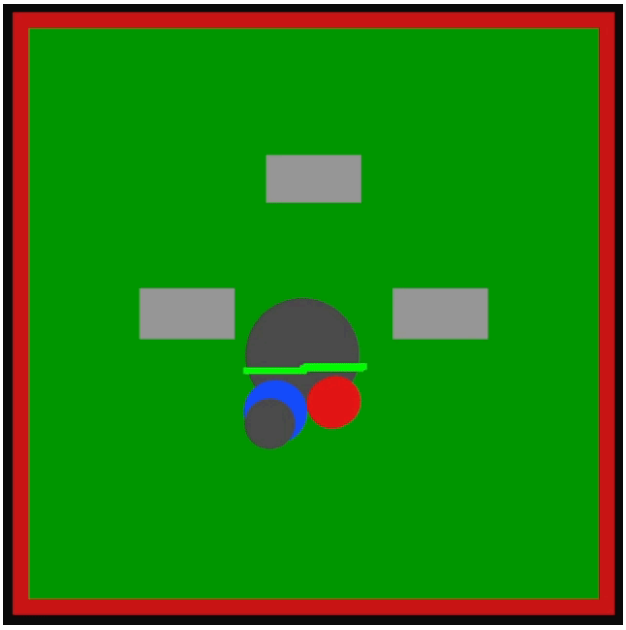


Figure G. **This is a recreation of the episode due to GPU failure.** The agents learn to rush the boss and kill it immediately with the tank taunting. The Melee DPS blocks the Boss's attack on the Tank.

Phase 3 – Wall-aware tactics and hazard-aware navigation.

Later in training, both sides begin to exploit the interactive elements of the arena. The Boss learns to pick the walls up and reposition them defensively, creating temporary cover or using them to play a kind of "ring-around-the-rosie" with the party. By orbiting around a carried or newly placed wall, the Boss intermittently breaks line-of-sight and forces the melee agents to chase it around obstacles. Party members, in turn, learn to reroute around these improvised barriers, maintain attack lines through openings, and stay coordinated as the Boss repeatedly slips behind cover. These exchanges produce longer, more dynamic episodes in which movement, pathing, and line-of-sight management play a noticeable role in the flow of combat.

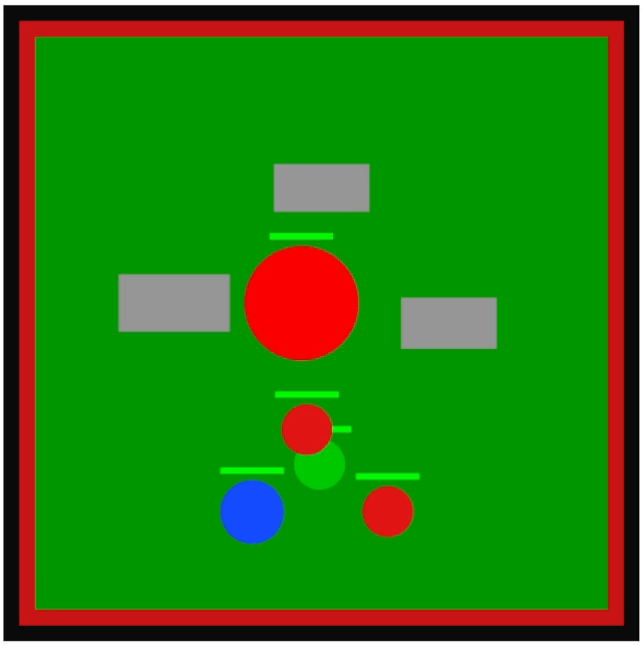


Figure H. **This is a recreation of the episode due to GPU failure.** The boss learns to pick up a wall and defend itself from attacks. Agents become confused as they cannot see the Boss anymore and aimlessly run around in response.

Throughout these phases, Social Network Analysis over recorded episodes reveals a shift from diffuse, symmetric interaction graphs to more structured networks: Tanks and Melee DPS tend to emerge as central nodes in damage and taunting graphs, while the Boss's interaction network becomes increasingly focused on a small subset of high-threat agents.

Although we do not directly compare against intrinsic motivation baselines in this work, our results support the view that multi-agent competition can induce rich, human-interpretable behavior without explicit exploration bonuses or handcrafted shaping. In our setup, agents are not rewarded for moving walls, staying out of lava, or protecting high-value teammates; nonetheless, we observe behaviors that resemble MMO-like best practices: Tanks drawing and holding aggro, Healers clustering behind cover, ranged agents kiting near the edge of their attack range, and Boss wall usage that meaningfully distorts the flow of combat.

These behaviors arise because each adaptation by one side reshapes the learning landscape for the other. When the Boss learns to focus high-threat targets, party members are pressured to discover aggro management strategies. When party members survive longer and deal more sustained damage, the Boss is pressured to learn better wall usage and hazard exploitation. This back-and-forth generates a natural curriculum over strategies and counterstrategies, without the need to design explicit auxiliary tasks or reward schedules.

Surprising behaviors

As in many complex physics-based environments, agents occasionally discover strategies that exploit implementation details rather than embodying the intended game design. One notable example occurred early in training, when the Boss learned to use walls as an unintended movement tool. By picking up a wall and then rapidly placing or "throwing" it downward while standing on or near it, the Boss could generate a net upward force greater than the combined effect of gravity on the Boss and the wall, effectively using the wall as a makeshift launch platform. This resulted in the Boss "flying" out of the intended play area or reaching otherwise inaccessible positions.

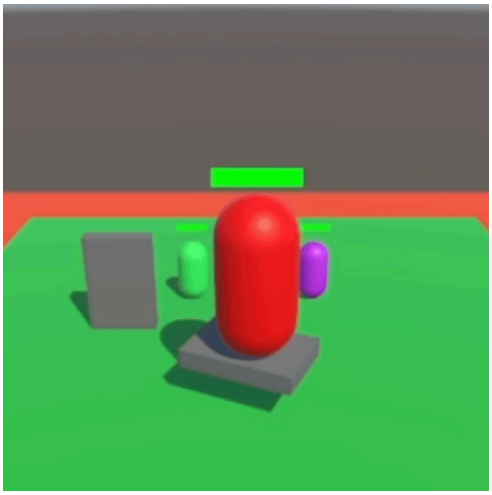


Figure I. The Boss discovers an unintended mobility exploit: repeatedly placing a wall downward while standing near it generates a net upward force, allowing the Boss to launch itself into the air and leave the intended play area.

A second unexpected behavior appeared when the Boss interacted with walls placed directly behind it. In this case, the pickup logic attempted to ensure that any grabbed wall would end up in front of the Boss, snapping the wall from its current position to a target position near the Boss's forward direction. When the Boss tried to pick up a wall that was very close behind it, this repositioning created a large instantaneous displacement. Combined with the physics system resolving overlapping colliders, the result was an explosive impulse that effectively "slingshotted" the Boss out of the arena. In some episodes, the Boss reached unrealistically high speeds or was launched across the map, not because it had learned a deliberate mobility trick, but because the underlying wall pickup implementation created a catapult-like effect.

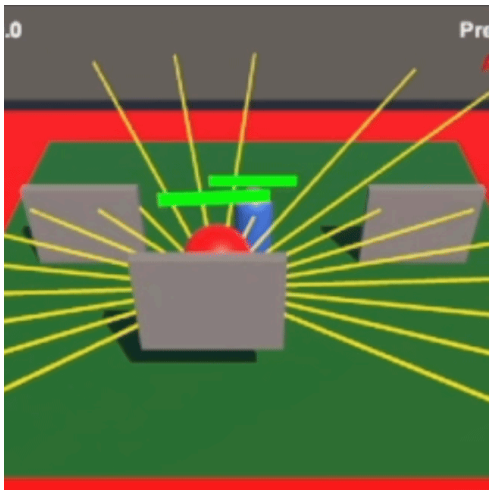


Figure J. When attempting to pick up a wall located directly behind it, the Boss triggers a corrective repositioning that snaps the wall forward, producing a physics impulse strong enough to fling the Boss across or entirely out of the arena.

While entertaining to observe, such behaviors highlight the importance of carefully specifying physics constraints, collider interactions, and out-of-bounds penalties in multi-agent environments. In our case, addressing the exploit required adjusting force application, adding stricter checks on wall placement, and reinforcing negative incentives for leaving the arena or spending prolonged time in invalid states. These kinds of failure modes serve as valuable stress tests for both the environment and the learning algorithms.

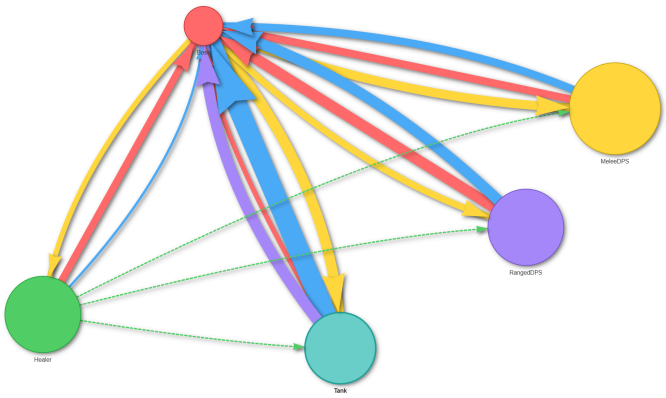
Social Network Analysis of Combat Interactions

To move beyond episode-level win rates and individual trajectories, I treated the boss fight as a role-level interaction network and analyzed it using Social Network Analysis (SNA). For each recorded episode, every combat action (attack, heal, or threat-related ability) contributes to a directed, weighted edge between the source role (who acted) and the target role (who was affected). Aggregating tens of thousands of these interactions yields a compact five-node graph (Boss, Tank, Healer, MeleeDPS, RangedDPS) where edge weight encodes interaction frequency or intensity, such as Boss causing Tank damage or Healer causing MeleeDPS healing, and node size approximates weighted degree centrality, representing how involved a role is in the total combat traffic.

Edges are visually differentiated by interaction type: yellow for Boss damage, red for party damage, blue for threat generation, purple for taunt, and green for healing. Node size additionally reflects class-selection frequency over training, allowing the visualization to simultaneously capture both behavioral patterns and role preference trends. Together, these design choices give a bird’s-eye view of the “social structure” of combat. Instead of asking only which episode won, we can see which roles consistently interact, who supports whom, and how these patterns evolve as learning progresses.

Early Phase (0–10,000 episodes): Diffuse, Symmetric Network

In the first ~10,000 episodes, policies are close to random. Class selection is roughly uniform, agents frequently wander into lava, and the Boss attacks whatever happens to be in front of it. The SNA graph in this regime is nearly symmetric and diffuse as shown in Graph A:

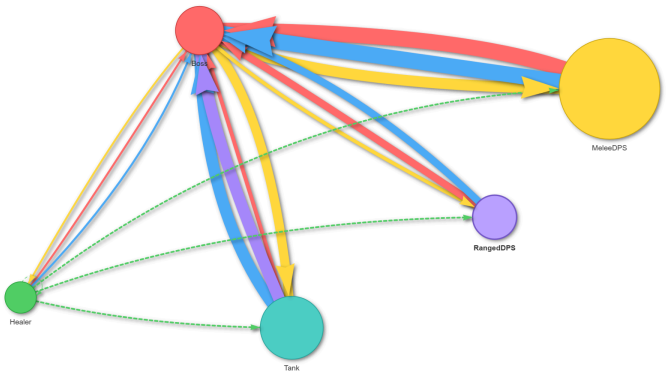


Graph A. All roles interact at similar frequencies, producing uniformly sized nodes and evenly weighted edges. The network is diffuse and symmetric, reflecting largely random behavior with no clear front line, focus target, or coordinated strategy.

Outgoing party-damage edges to the Boss from Tank, MeleeDPS, and RangedDPS have similar weights, reflecting that all three roles attack with comparable frequency. Boss damage edges are spread across all four party roles; no single role consistently anchors aggro. Healing edges from the Healer are weak and thin, often comparable in magnitude to incidental actions by other roles. Node sizes are similar, indicating low differentiation in centrality-no clear "front line" or "back line" has emerged yet. In short, the early network is close to what you would expect if roles were randomly permuted each episode: everyone hits everything a little bit, and no role structurally stands out.

Late Phase (20,000+ episodes): Melee-Rush Strategy Becomes Dominant

In the final portion of training, the SNA graph becomes highly skewed and structured, revealing the dominant strategy the agents converged to. As shown in Graph B:



Graph B. Tank and MeleeDPS emerge as dominant roles, shown by their enlarged nodes and thick edges to the Boss. Healing and threat flows concentrate on the melee cluster, illustrating the learned melee-rush strategy, while RangedDPS remains peripheral.

In the late-training network, the MeleeDPS node dominates the graph, appearing far larger than all other roles. The thick red and blue arrows flowing between MeleeDPS and the Boss indicate that the majority of both outgoing party damage and incoming boss damage centers on this role. This reflects a learned strategy in which MeleeDPS becomes the primary source of offensive pressure and simultaneously absorbs a significant amount of retaliation from the Boss.

The Tank node is the second largest, positioned directly beneath the Boss with substantial red (damage) and blue (threat) arrows directed upward into the Boss. The strong yellow Boss to Tank edge shows that the Boss frequently targets the Tank, anchoring much of its attention on the frontline engagement.

The Healer node, though smaller, connects broadly across the team through multiple thin green healing edges, most notably toward the Tank and MeleeDPS. These edges form a wide healing "fan," indicating that the Healer contributes support across the melee core during late-stage combat, but with less emphasis on the more peripheral RangedDPS.

RangedDPS becomes a noticeably smaller, more isolated node. Its red edges toward the Boss remain present but thinner, and it receives relatively little interaction compared to the melee roles, reflecting its reduced involvement in the final learned strategy.

Overall, the network collapses into a melee-centric combat structure: a dense triad of Boss to MeleeDPS to Tank and back to Boss, continually exchanging damage, threat, and healing support. The Healer serves as a stabilizing support role feeding into this melee cluster, while RangedDPS remains structurally peripheral. This topology mirrors the behavioral convergence observed in training, where the agents default to a straightforward but effective melee-rush strategy and commit the Tank to hold attention, pour damage through MeleeDPS, and rely on the Healer to keep the front line alive.

Interpretation and Strategic Implications

The SNA results show that even with sparse rewards and limited training time, the agents did not remain in a random interaction regime. Instead, they discovered the simplest viable team strategy supported by the environment: focus damage into a single high-output role (MeleeDPS), anchor aggro on a durable front-liner (Tank), and route healing primarily into the roles that share that frontline.

However, the same graphs also reveal what didn't emerge. There is no strong signature of kiting (which would appear as strong Boss to Tank damage with long episodes and relatively modest MeleeDPS to Boss edges spread over time). There is little evidence of sophisticated healer triage. Healing edges cluster around all class cores rather than dynamically shifting across the team.

RangedDPS never becomes strategically central, suggesting that the agents did not discover positioning-heavy or range-control strategies. Taken together, the SNA analysis supports two key conclusions.

First, it confirms that the learned policies are not arbitrary: combat traffic organizes into a coherent, melee-centric pattern that reflects an understandable boss-rush strategy. Second, it highlights the limitations of the current dataset and training budget: the interaction graph stabilizes around a simple damage race rather than more nuanced MMO patterns (e.g., phases, positional mechanics, or role-swapping), suggesting that larger-scale training or richer reward structures would be required to push beyond this local optimum.

This combination of structural insight and failure-mode visibility is exactly why SNA is useful in this setting. It turns millions of low-level combat events into a small number of interpretable patterns that summarize what the agents actually learned, and what they did not.

What I Wished I Could Have Seen

Because of the limited training horizon and relatively small dataset, the agents converged on the simplest workable solution rather than discovering richer or more creative MMO-style tactics. Several emergent behaviors never appeared, but would have been especially interesting to observe purely as outcomes of the reinforcement learning process.

1. Full Ranged Kiting and Distributed Threat Management

One behavior I hoped for was a party that intentionally gravitates toward four ranged classes and treats the entire encounter as a kiting problem.

Here is how I would imagine the episode to play out. All four ranged agents would maintain a rotating orbit around the Boss. Threat would be shared across multiple agents instead of collapsing onto a single Tank. The Healer would dynamically allocate healing to the current "aggro holder" before the group collectively shifts the threat again.

In SNA terms, this would produce a balanced, four-node DPS cluster with similar outgoing damage edges and evenly distributed Boss to DPS interactions, which is very different from the melee-centric triad that ultimately emerged.

2. Boss Exploiting Walls to Force Environmental Eliminations

Another behavior I wished had emerged was a Boss that discovers how to weaponize the environment, not accidentally, but as a consistent high-level strategy. The boss would use walls to herd party members toward the lava ring, then create choke points to cut off escape routes or physically push agents off safe tiles to secure fast eliminations.

If this had developed naturally through RL, a SNA graph with environment nodes would show bursts of Boss to DPS interactions concentrated near hazard zones, alongside sudden shifts in the Boss to Tank ratio that indicate opportunistic targeting decisions rather than routine melee engagements.

Time-constrained development and scope

This project was intentionally executed under a strict 30-day constraint: designing the environment, implementing the Boss and party agents, integrating ML-Agents, building the data analysis pipeline, and drafting this paper all had to fit within a one-month window. Treating the work as a "30-day research sprint" forced a series of trade-offs in scope and depth. For example, we prioritized a single, well-instrumented boss encounter over a broader set of different bosses or maps, we focused on sparse rewards and self-play instead of experimenting with multiple shaping schemes, and we limited ourselves to a single PPO-based training setup rather than exploring a large hyperparameter or algorithmic sweep.

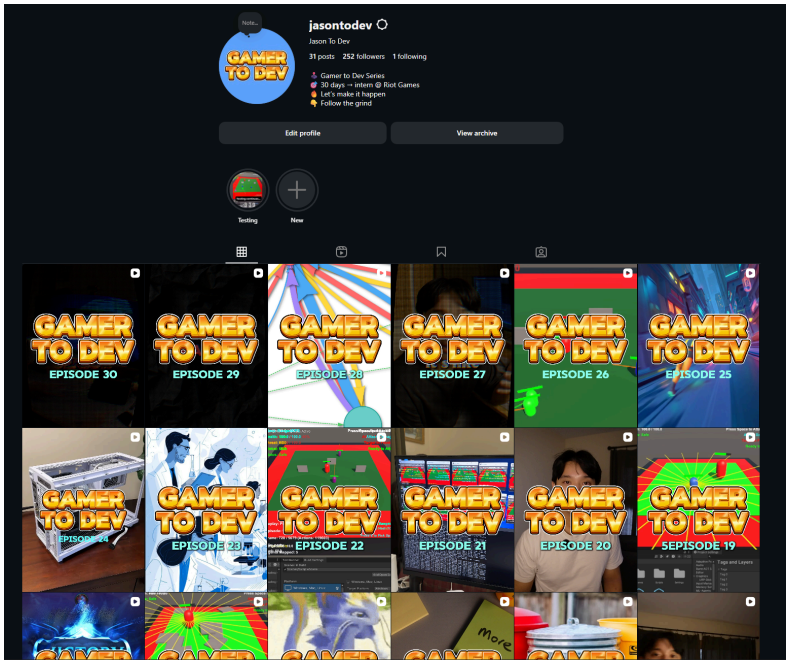


Figure K. A visual record of the author's daily updates on Instagram, documenting the evolution of the project as it developed from concept to a complete multi-agent RL experiment.

This constraint is an important context for interpreting the results. The goal of the project was not to exhaustively solve MMO boss combat with multi-agent RL, but to demonstrate that, within a realistic, time-bounded internship-style timeline, it is possible to build a functioning research environment, train agents to exhibit non-trivial emergent strategies, and analyze those strategies using tools such as Social Network Analysis. In that sense, the 30-day limit serves both as a practical boundary condition and as part of the contribution: it mirrors the kind of rapid prototyping and applied research cadence that a game AI research team might expect from an intern or early-stage project.