

A Framework for Genre-Aware Player Adaptation in LLM-Driven Game AI

Predrag Peharda, David Slavik

University of Zagreb Faculty of Organization and Informatics

Artificial Intelligence Laboratory

Pavlinka 2, 42000 Varaždin, Croatia

pehardap@gmail.com, dslavik@foi.hr

Abstract. *Large language models make it inexpensive to add a "strategic" adaptation layer to game AI: a component that reads how a player behaves and re-shapes an agent's strategy between encounters. This paper argues that the value of such adaptation is genre-contingent. It undermines genres that reward mastering a stable, learnable system, and benefits genres that reward a responsive or living world. The main problem is therefore calibration, not capability: more adaptive is not always better. In this paper we propose a conceptual framework for deciding when such adaptation is appropriate.*

Keywords. game AI, large language models, player modeling, player adaptation, game design, framework

1 Introduction

AI and games share a long history, beginning with programs built to play games as a test of whether a machine could perform tasks that appeared to require intelligence, and progressing through Checkers, Chess, and Go to modern video games (Yannakakis & Togelius, 2025, pp. 4–16).

Over time, the role of AI in games expanded beyond playing them, to generating content and to adapting the game experience to the individual playing it. Early on-line games collected player data during play and tuned themselves from it, later titles responded to a player's emotional state, and recent systems delegate the memories and behavior of non-player characters to LLMs (Yannakakis & Togelius, 2025, pp. 4–16). Adapting a game to the player has become a common practice rather than an exceptional one, and much of this work proceeds on the assumption that tuning a game more closely to its player produces a better experience.

LLMs change the cost of building such adaptation. An LLM can take an aggregated description of how a player has behaved and re-shape an agent's strategy through a prompt alone, without a trained model or a manually built difficulty controller. This makes it inexpensive to add a strategic adaptation layer to an agent that already exists. Once a capability becomes

inexpensive to build, the relevant question is no longer whether it can be built, but whether it should be applied, and in which contexts.

In this paper, we argue that the value of adapting to the player is genre-contingent. Some genres are built around mastering a fixed system: a competitive opponent is satisfying precisely because the player can study it, learn its patterns, and eventually overcome it, and an agent that quietly reshapes itself to match the player removes that source of satisfaction (Cowley & Charles, 2016). Other genres are built around a world that feels alive and responds to the player, such as a narrative RPG or a companion character, and in these the same adaptation is what makes the experience compelling. A single mechanism therefore produces opposite results depending on what the game is designed to offer.

Our contribution is a conceptual framework for making that determination. A three-tier architecture confines the LLM to a strategic role and keeps it out of the real-time control loop. The framework is conceptual, and in this paper it is not empirically validated.

Section 2 reviews player modeling and DDA, LLMs in games, and the planning limits that motivate keeping the model off the real-time loop. Section 3 presents the framework, Section 4 an illustrative example in which a boss enemy and a companion run on the same mechanism to opposite effect, Section 5 discusses what the framework does and does not claim, and Section 6 concludes and outlines future work.

2 Background and Related Work

2.1 Player Modeling and Dynamic Difficulty Adjustment

Player modeling is one of the established uses of AI in games, alongside playing games and generating content. It refers to the detection, prediction, and expression of the cognitive, affective, and behavioral characteristics that a person exhibits while playing (Yannakakis & Togelius, 2025, pp. 315–316).

A player model, in this sense, is a representation of how an individual interacts with a game and responds

to it, learned from observations of their play. Such a model can be put to use at different stages. It may be applied during the analysis of play, during the design process, or during play itself, and it is this last case that the present paper concerns. Yannakakis and Togelius term it adaptation: the game reads a model of the player and modifies itself in response while the player is still playing (Yannakakis & Togelius, 2025, pp. 320–321). The use cases they group under adaptation include dynamic game balancing, matchmaking, and personalized content, as well as experiences tailored through non-player characters (Yannakakis & Togelius, 2025, p. 322).

The most extensively studied form of this kind of adaptation is dynamic difficulty adjustment (DDA), which modifies a game’s features and behavior in real time according to the player’s skill in order to keep the player engaged. Approaches range from heuristic adjustments based on game state to methods that respond to the player’s measured affective state. Reviews of this work are broadly positive, reporting improvements in engagement and related measures, while also noting that player preferences remain incompletely understood and that further study is needed (Zohaib, 2018). The assumption underlying much of this work is that a closer fit between the game and the player produces a better experience.

A common thread across this work is that effective adaptation depends on a model of the player. The player model is what supplies the constraints the adaptive system acts on, rather than the system adjusting in an unprincipled way. This view frames adaptation as the application of a player model rather than as a standalone mechanism (Cowley & Charles, 2016).

2.2 LLMs in Games

The use of LLMs in games has grown quickly, and recent surveys have begun to map the roles such models can play. Gallotta et al. (2024) catalogue the applications of LLMs in and for games and identify the distinct roles a model can take, principally as a player that makes decisions, as a non-player character that controls dialogue and behavior, and as a generator of content such as levels or whole games. Much of the surveyed work concentrates on these roles. LLMs have proven well suited to generating natural-language content, to driving the speech and behavior of characters, and to playing games whose states and actions can be expressed as text (Gallotta et al., 2024).

A parallel line of work studies LLMs specifically as agents that play games. Hu et al. (2024) survey this area and describe such agents in terms of three components: memory, reasoning, and input and output, organizing the literature across six game genres, including adventure, communication, competition, cooperation, simulation, and crafting and exploration. The emphasis throughout is on the model as a participant that per-

ceives a game and acts within it, rather than as a system that reshapes the game around the person playing it (Hu et al., 2024).

What receives far less attention in this literature is the use of an LLM to model and respond to the player. Gallotta et al. (2024) note this directly, identifying player modeling as one of the most promising but least explored directions for future work on LLMs in games. The present paper takes up that direction. Rather than using an LLM to play a game or to author its content, we consider an LLM that reads a model of the player and adjusts the behavior of a game agent in response, and we ask under what conditions doing so is appropriate.

2.3 Planning Limitations of LLMs

A growing body of work questions how far LLMs can be trusted to reason and plan on their own. Kambhampati (2024) argues that, despite their fluency, LLMs do not perform reliable planning in the way a dedicated planner does; what looks like reasoning is better understood as approximate retrieval over patterns seen in training, and the models are poor at verifying or correcting their own plans. The practical consequence is that an LLM cannot be assumed to produce a sound plan, and it cannot be relied upon to notice when its own plan is unsound (Kambhampati, 2024).

This does not mean LLMs have no place in tasks that involve planning. Kambhampati et al. (2024) propose what they call LLM-Modulo frameworks, in which the model generates candidate solutions but an external component checks them against explicit constraints, accepting only those that pass. In this arrangement the LLM contributes ideas while a separate, verifiable system holds responsibility for correctness. The division of labor is deliberate: the model proposes, and the surrounding system disposes (Kambhampati et al., 2024).

These limitations bear directly on where an LLM can sit within a game. A game’s moment-to-moment control loop runs on a strict timing budget and demands consistent, correct behavior many times per second, which is precisely the regime in which an unverified and slow generator is least appropriate. The framework proposed in the next section takes this constraint as a starting point. It keeps the LLM out of the real-time loop and confines it to a slower role in which it proposes adjustments, while authored, verifiable game logic retains control over what the agent actually does.

3 The Framework

The framework proposed has three components. The first is an architecture that places the language model within a game agent without letting it compromise the agent’s real-time behavior. The second is a heuristic for deciding whether adaptation suits a given game at all. The third is a mechanism for controlling how much

adaptation is allowed once it has been deemed appropriate.

3.1 A Three-Tier Architecture

We organize a game agent into three tiers, separated by the timescale on which they operate and by who or what is responsible for their decisions.

1. The reflex tier governs behavior at the scale of milliseconds: animation, collision and hitbox resolution, target tracking, and the immediate execution of an action once it has been chosen. Decisions at this tier must be fast and consistent, and they must be correct every time. For the reasons set out in Section 2.3, an LLM has no place in this tier. It is too slow to meet the timing budget of a real-time loop, and its output is too variable to guarantee the consistency this tier requires.
2. The tactical tier governs behavior at the scale of seconds: choosing which action or short sequence of actions to perform in the current situation. This tier draws on a fixed set of designer-authored strategies, which we refer to as the agent's tactical vocabulary. Each strategy is written, tested, and balanced by the game's designers, exactly as in a conventional game agent. The tactical tier is where the agent's actual conduct is decided, and it remains under authored control at all times.
3. The strategic tier governs behavior between encounters, on the slowest timescale of the three. The language model operates only at this tier. The model receives an aggregated player profile, a summary of how the player has behaved over prior encounters, and uses it to re-weight the tactical vocabulary: raising the priority of some authored strategies and lowering that of others for the next encounter. The model does not act in real time, and it does not act directly on the game. It changes only the relative emphasis among strategies that the designers have already authored.

This separation is the central safety property of the framework. The language model never invents an action, never executes one, and never participates in the moment-to-moment control of the agent. It proposes a re-weighting of an authored vocabulary, and the tactical and reflex tiers retain full responsibility for what the agent actually does. This arrangement follows the division of labor described in Section 2.3: the model only makes suggestions, while authored and tested game logic determines what the agent does. In the worst case, an unsound suggestion can do no more than shift emphasis among strategies that are all individually safe, since every available action has been authored and tested in advance.

3.2 The Stable-Model versus Living-World Spectrum

The architecture of Section 3.1 describes where adaptation happens and how it is constrained, but not whether it should happen at all. That decision depends on the game. We propose a single heuristic for making it: ask whether the genre rewards the player for building a stable mental model of a fixed system, or for responding to a world that is unpredictable and alive. The first kind of game we call stable-model; the second, living-world. Where a game falls on this spectrum predicts whether strategic adaptation will help or harm it.

At the stable-model end, the player's satisfaction comes from studying a system that holds still long enough to be learned. A "souls-like" boss enemy is the clearest case. The term refers to the punishing, pattern-based combat popularized by the *Dark Souls* series (FromSoftware), in which an enemy's attacks and timings form a fixed challenge that the player is expected to learn through repeated attempts. Mastery is the experience the game offers. A competitive opponent is similar, in that fairness depends on both sides facing rules that do not silently change. For games of this kind, strategic adaptation is corrosive. If the agent quietly re-weights its strategies in response to the player, the system the player is trying to learn becomes a moving target, and the act of mastering it loses its meaning. The very mechanism described in Section 3.1 works against this kind of genre.

At the other end of the spectrum, the living-world case, the player's satisfaction comes from the sense that the world notices them and responds. A companion character that adjusts to the player's habits, or a narrative agent that shifts its behavior as it comes to know the player, is more compelling for doing so. Here the same strategic re-weighting is not a flaw but the point. Adaptation makes the world feel responsive and particular to this player, which is exactly what that kind of genre promises.

This division has a precedent in existing taxonomies of player modeling. Yannakakis and Togelius (2025) distinguish adaptation aimed at player behavior, such as dynamic balancing and matchmaking, from adaptation aimed at player experience, such as an experience tailored through non-player characters. The behavior-oriented cases tend toward our stable-model pole, where what is adjusted is the challenge a player is trying to master, while the experience-oriented cases tend toward our living-world pole, where what is adjusted is how the world feels. Our contribution is not to redraw this distinction but to read it normatively: to treat a game's position on the spectrum as a prediction of whether adaptation should be applied at all.

Most games are not purely of one kind, and a single game may contain both a system meant to be mastered and a world meant to feel alive. For example a "souls-like" game with a companion character has a stable-model combat system alongside a living-world

relationship, and the right decision may differ for each. The spectrum is therefore best applied not to a whole game but to the particular system an agent governs, and to the source of satisfaction that system is meant to provide.

3.3 The Governor

Deciding that a game sits toward the living-world end of the spectrum does not settle how much adaptation it should receive. A companion that re-invents itself after every conversation is as unconvincing as one that never changes at all. The spectrum tells a designer whether to adapt; it does not tell them how far. For that we introduce a governor: a small set of controls that bound the strategic tier's influence over the agent. The governor has two "knobs".

The first is the cap. The strategic tier re-weights the tactical vocabulary, but the cap places a limit on how much weight any single strategy may receive. With a low cap, no authored strategy can come to dominate the agent's behavior, so the agent continues to draw on its full set of actions even as the emphasis shifts. With a high cap, the strategic tier is free to concentrate the agent's behavior around a few strategies that suit the player. The cap therefore controls how strongly adaptation is allowed to reshape the agent in any one direction.

The second is the cadence. The strategic tier acts between encounters, but the cadence sets how often it is permitted to act: after every encounter, after several, or only at designer-chosen moments. A slow cadence means the agent changes rarely, leaving the player a stable target across many encounters before any adjustment occurs. A fast cadence means the agent changes often, responding quickly to how the player is behaving. The cadence therefore controls how readily the player perceives the agent as changing at all.

Together, the two "knobs" convert the spectrum's qualitative judgment into a quantitative setting. A stable-model system is governed tightly: a low cap, so the agent never reshapes itself into a different opponent, and a slow cadence, so what little adaptation occurs does not disturb a player. The limiting case is a cap and cadence set so low that adaptation is switched off entirely, which is the correct setting for a game that must not read its player at all. A living-world system is governed loosely: a higher cap and a faster cadence, so the world can respond visibly and often. Most systems fall between these settings, and the governor is what lets a designer place them there rather than choosing only between adaptation and its absence.

This is the sense in which the problem is calibration rather than capability. The capability, an LLM re-weighting an authored vocabulary, is the same across every game. What differs is how the governor is set, and setting it well is the act of matching adaptation to what a particular system is for.

4 Illustrative Example

To show how the framework operates, we walk the same mechanism through two cases. In both, a game agent is built on the three tiers of Section 3.1, and in both, the strategic tier re-weights an authored tactical vocabulary from a player profile. The cases differ only in where their games sit on the spectrum of Section 3.2, and therefore in how the governor of Section 3.3 is set. The mechanism is identical; the calibration is not.

4.1 A Boss in a Stable-Model Game

Consider a boss enemy in a game whose challenge is meant to be studied and overcome. The agent's tactical vocabulary is a set of authored strategies, each a pattern of attacks and responses that a player can learn to read and counter. Over repeated attempts, a player profile accumulates a summary of how this player behaves: which strategies they handle well and which they struggle against.

If the strategic tier were allowed to act freely on this profile, it would re-weight the vocabulary toward whatever the player handles least well, sharpening the boss against the player's specific weaknesses between attempts. This is precisely the wrong outcome for the genre. The player is trying to learn a fixed system, and an agent that shifts to exploit them turns that system into a moving target. Effort that should yield mastery instead yields a boss that is always one step ahead, and the satisfaction the game is built to provide is lost.

The framework's response is not to remove the mechanism but to govern it tightly. The cap is set low, so no single strategy can come to dominate and the boss remains the same opponent from attempt to attempt. The cadence is set slow, or to zero, so the boss does not change underneath a player who is in the middle of learning it. In the limiting case the governor switches adaptation off, and the agent behaves exactly as a conventional, authored boss would. For a stable-model game, that is the correct setting, and the framework reaches it not by lacking the capability to adapt but by choosing not to apply it.

4.2 A Companion in a Living-World Game

Consider instead a companion character in a game whose appeal is a world that responds to the player. The agent is built identically: the same three tiers, the same kind of authored tactical vocabulary, the same strategic tier re-weighting it from a player profile. Here the profile summarizes the player's habits and tendencies over time, and the vocabulary is a set of authored behaviors the companion can express.

When the strategic tier re-weights this vocabulary toward the player, the companion comes to behave in ways that fit how this particular person plays. This is the outcome the genre wants. The player's sense that

the companion notices and responds to them is the experience on offer, and the adaptation is what produces it.

The framework governs this case loosely. The cap is set higher, so the companion’s behavior can concentrate meaningfully around the player rather than only shifting at the margins. The cadence is set faster, so the companion responds within a timescale the player will notice and attribute to being understood. The mechanism is unchanged from the boss; only the governor’s settings differ, and they differ because the source of satisfaction differs.

4.3 The Same Mechanism, Calibrated Differently

The two agents are the same construction. Each confines its language model to the strategic tier, each re-weights an authored vocabulary, and each is bounded by a cap and a cadence. What separates them is a judgment about what their respective games are for, expressed as two settings of the governor. The boss is governed tightly because mastery requires a stable target; the companion is governed loosely because responsiveness requires a moving one.

5 Discussion and Limitations

The framework is a design heuristic and an architecture, not an empirical result. It claims that the value of strategic player adaptation depends on what a game is for, that this can be predicted from where a system sits on the stable-model versus living-world spectrum, and that the resulting decision can be expressed through the governor’s two controls. The one claim it makes firmly is the architectural safety property: because the LLM only re-weights an authored vocabulary, it cannot produce behavior the designers did not write and test.

The framework is conceptual and, in this paper, unvalidated. We present no implementation, no metrics, and no player study, and its central prediction, that adaptation harms stable-model systems and helps living-world ones, is argued from the structure of those genres rather than demonstrated. It is also worth stating that the LLM here is not learning in any machine-learning sense. It is not trained or updated by play; it is re-prompted with a player profile and asked to re-weight authored options, so the adaptation is a change of emphasis, not a change in the model.

One question the framework raises but does not settle is whether players enjoy being read. Yannakakis and Togelius (2025) observe that a game could, in principle, come to know more about how a person plays than the person knows themselves. Our framework takes that capability as given and asks where it should be used, but it does not establish how players feel about it, and the same adaptation may read as attentive to one player and intrusive to another.

Finally, the framework does not introduce adaptive game AI, player modeling, or dynamic difficulty adjustment, all of which are mature areas surveyed in Section 2. Its contribution is the narrower, normative question of when an existing capability, made newly inexpensive by LLMs, should be applied at all.

6 Conclusion and Future Work

This work presents a conceptual framework for deciding when strategic player adaptation, now inexpensive to build with LLMs, is appropriate for a game. The framework combines a three-tier architecture that confines the LLM to a slow, strategic role and keeps it out of the real-time loop, a stable-model versus living-world heuristic that predicts whether adaptation helps or harms a given system, and a governor that turns that prediction into concrete settings for how strongly and how often the agent adapts. Its recurring claim is that the design problem is calibration rather than capability: the same mechanism serves one game and damages another, and the difference lies in how it is governed.

Future work could replace the illustrative cases of Section 4 with a working implementation, allowing the boss enemy and companion agents to be built rather than described. Player studies could then test the framework’s central prediction directly, measuring whether tightly governed adaptation preserves the experience of a stable-model system while loosely governed adaptation improves a living-world one. Such studies would also address the open question of whether players enjoy being read, which the framework raises but cannot settle. We see the present framework as the conceptual groundwork for that validated follow-up, and as a first treatment of a design space we would call, in its fuller form, that of stable models and living worlds.

References

- Cowley, B. U., & Charles, D. (2016). Adaptive artificial intelligence in games: Issues, requirements, and a solution through behavior-based general player modelling. *arXiv preprint arXiv:1607.05028*.
- Gallotta, R., Todd, G., Zammit, M., Earle, S., Lipapis, A., Togelius, J., & Yannakakis, G. N. (2024). Large language models and games: A survey and roadmap. *IEEE Transactions on Games*, 1–18. <https://doi.org/10.1109/TG.2024.3461510>
- Hu, S., Huang, T., Ilhan, F., Tekin, S., Liu, G., Kompella, R., & Liu, L. (2024). A survey on large language model-based game agents. *arXiv preprint arXiv:2404.02039*.

- Kambhampati, S. (2024). Can large language models reason and plan? *Annals of the New York Academy of Sciences*, 1534(1), 15–18. <https://doi.org/10.1111/nyas.15125>
- Kambhampati, S., Valmeekam, K., Guan, L., Verma, M., Stechly, K., Bhambri, S., Saldyt, L., & Murthy, A. (2024). LLMs can't plan, but can help planning in LLM-modulo frameworks. *Proceedings of the 41st International Conference on Machine Learning (ICML)*.
- Yannakakis, G. N., & Togelius, J. (2025). *Artificial intelligence and games* (2nd ed.). Springer.
- Zohaib, M. (2018). Dynamic difficulty adjustment (DDA) in computer games: A review. *Advances in Human-Computer Interaction*, 2018(1), 5681652. <https://doi.org/10.1155/2018/5681652>