Proceedings of the 4th International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/2026.39387

Behavior Trees in Action Games: A Review on Enhancing Players' Combat Experience

Siyao Wu[1,*], **Zhongyu Chang**[2], **Mingzhen Li**[3]

[1]*Northeastern University, Shenyang, China,*
[2]*University of Wisconsin Madison, Shaanxi, China*
[3]*Central South University, Changsha, China*
[*]*Corresponding Author. Email:15524294122@163.com*

Abstract: In modern action games, Behavior Trees (BTs) are common. They control non-player characters (NPCs). Modularity, flexibility, and easy maintenance make them useful. Prior work mostly deals with technical implementation. Few studies examine how BT design affects player combat experience. This review looks at recent papers from three sides: algorithmic optimization, NPC parameter design, and player experience evaluation. Learning Behavior Trees allow NPCs to adapt their behavior and decisions. Parameterized Behavior Trees support customization and difficulty tuning. For player experience, PENS, GEQ, GUESS, and NASA-TLX are used. These tools help evaluate player responses to AI-driven behavior. We compare current methods, point out gaps, and suggest future directions. The goal is combat AI that adapts better and focuses more on the player.

Keywords: Behavior Trees; Game AI; Player Experience

1. Introduction

Action games like *Dark Souls* and *Sekiro: Shadows Die Twice* attract many players. Their NPCs often use Behavior Trees (BTs) or similar systems. In fast combat, BTs have some advantages over Finite State Machines (FSMs). They are modular and hierarchical. This makes AI logic easier to organize, reuse, and maintain. Debugging can also be simpler during development [1, 2].

Most BT research deals with technical issues, such as tree structure, execution speed, and decision logic. Fewer papers consider player-side effects. The link between AI design, NPC behavior, and player experience is not clear. It remains unclear how BT design affects autonomy, competence, immersion, or cognitive load. Therefore, a technical improvement in BTs may not necessarily lead to better gameplay.

This review covers three topics: algorithmic mechanisms, NPC parameter design, and player-centered evaluation. For player outcomes, established frameworks include PENS [3], GEQ [4], GUESS [5], and NASA-TLX [6].

Three studies correspond to these perspectives. [7] examines adaptive Behavior Trees. [8] studies parameter passing and behavior reuse. [9] examines player perceptions of human and AI-controlled agents. Together, these studies connect AI algorithms, NPC behavior design, and player evaluation.

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.39387

1.1. Algorithmic perspective

Learning Behavior Trees are reviewed in this section. They adapt NPC behavior using observations of player behavior [7]. The study also examines the main results and limitations of this approach.

1.2. NPC parameter design perspective

Parameterized Behavior Trees are discussed in this section [8]. The discussion covers explicit parameter passing, reusable subtrees, PAR arguments, and lookup nodes. Their design is also compared with other AI approaches, with attention to the limitations of this method.

1.3. Player experience perspective

This section examines how players perceive AI-controlled co-players. The focus is on perceived humanness and social interaction [9]. These findings are also considered in relation to broader player experience dimensions, including PENS, GEQ, GUESS, and NASA-TLX.

2. Background

This section introduces the concepts needed to understand Behavior Trees in action game AI and their connection to player experience. Three topics are covered: Game AI, Behavior Tree structure, and player experience evaluation tools.

2.1. Game artificial intelligence

Game AI controls non-player characters (NPCs) and other autonomous agents. It enables them to make decisions according to the game state and player actions. In game development, AI systems need to operate in real time. They also need to remain controllable and flexible during design. Early games commonly used rule-based methods and Finite State Machines (FSMs) to control NPC behavior [1]. As game systems became more complex, these methods faced several problems. State explosion made systems harder to manage. Maintenance also became more difficult, while predefined rules limited behavioral variation and adaptability [10].

2.2. Behavior trees

A Behavior Tree (BT) is a hierarchical structure for controlling the behavior of an intelligent agent. It represents the logic used to select and execute actions. A typical BT contains control nodes, condition nodes, and action nodes [2]. Compared with FSMs, BTs organize behavioral logic in a tree structure. Complex behaviors can therefore be divided into separate modules. This makes the resulting logic easier to read, extend, and maintain.

BTs are widely used in game AI to organize NPC behavior [11]. Recent studies have extended BTs with learning mechanisms and parameter control. These extensions allow NPC behavior to vary according to different conditions. Learning Behavior Trees, for example, modify an existing BT based on observed player behavior [7]. This provides a way to address repetitive and predictable behaviors in manually scripted NPCs.

2.3. Existing player experience evaluation frameworks

As game AI develops, NPC behavior is evaluated not only by decision correctness and execution speed. Its effects on player experience have also become relevant. Player Experience (PX) refers to

DOI: 10.54254/2755-2721/2026.39387

players' psychological responses and feelings during gameplay. Common dimensions include autonomy, competence, immersion, and cognitive workload [12].

Several frameworks have been developed to measure player experience. The Player Experience of Need Satisfaction (PENS) framework evaluates the satisfaction of psychological needs, including autonomy, competence, and relatedness [3]. The Game Experience Questionnaire (GEQ) measures aspects such as immersion, tension, and challenge [4]. The NASA Task Load Index (NASA-TLX) is used to assess cognitive workload during interaction [6]. Together, these frameworks provide measures for examining the relationship between game mechanics, AI behavior, and player experience.

3. Algorithmic perspective

Tomai and Flores introduced Learning Behavior Trees (LBT) in [7]. LBT observes player behavior. This makes NPCs more adaptable. LBT does not rebuild the tree from zero. It adds Modifier nodes to the current structure. NPCs adjust their decisions using these nodes, based on what they saw players do. The original tree's hierarchy, interpretability, and maintainability stay intact.

Before LBT, three main approaches existed for NPC adaptation: dynamic scripting, evolutionary algorithms, and learning methods. Each has drawbacks. Dynamic scripting relies on hand-coded rules. Integrating these rules with existing game AI is often difficult. Evolutionary algorithms can create Behavior Trees automatically. The generated trees usually need human checking and refinement. Neural networks learn well, but they do not give the clear, editable structure that makes Behavior Trees attractive for game AI. Hand-authored Behavior Trees and Finite-State Machines follow fixed logic. Players can often guess NPC actions [1, 2]. LBT takes another route. It updates NPC behavior step by step using player trajectory data. It also preserves the structural strengths of Behavior Trees.

3.1. Algorithm

LBT learns player behavior from recorded trajectories. The learning starts from a manually built Behavior Tree. Each node has annotations for the control variables needed to run and evaluate that node's behavior. Given a recorded player trajectory, the NPC runs its own Behavior Tree and replays the player actions from the trajectory at the same time. At each decision point, the NPC compares its predicted behavior with the behavior in the player's trajectory. If they match, the NPC keeps its original decision logic. If they do not match, LBT tries to identify the cause. Then it changes the execution logic by adding Modifier nodes.

Modifier nodes are Decorator nodes. They attach to existing Behavior Tree nodes. When a deviation occurs, LBT first checks if the cause is a timing difference. If the NPC does the right action but earlier than the player, LBT adds a Delay Modifier above the leaf node for that action. The Delay Modifier stops its child node from updating for a learned duration. It returns a running state until the delay ends. Training samples link the current game state to the observed delay. This lets the NPC repeat the same timing change in similar future situations.

For deviations not caused by timing, LBT traces back through the tree to find the part responsible. LBT also provides four general-purpose Modifiers for different behavioral mismatches. Table 1 lists them.

After each modification, LBT replays the player trajectory to test if the modification reproduces the observed player behavior. If the test succeeds, the modification is kept. If not, the unexplained deviation is left for further analysis. Modifier nodes collect valid training samples from multiple player trajectories over time. More samples bring the learned NPC behavior closer to the observed

DOI: 10.54254/2755-2721/2026.39387

Table 1: General-purpose Modifiers in LBT

Modifier	Purpose
IgnoreCondition	Removes a condition that blocks the observed behavior.
RestartCaller	Restarts the parent when a running behavior blocks other options.
SetPreference	Learns the player's preferred values for control variables.
Suppress	Hides an unselected subtree so other behaviors can be checked.

player behavior. LBT repeats this process: compare behavior, trace deviation, modify the tree, verify by replay. This process adds varied behaviors to the original Behavior Tree.

From the player's view, LBT is expected to reduce repetitive and predictable NPC behavior through data-driven adaptation. It incorporates observed player behavior into NPC decisions. This can enrich player-NPC interactions and make the game feel more dynamic. However, it is still unclear whether this behavioral-level optimization actually improves the player's subjective experience. Controlled experiments with players are needed to verify this.

3.2. Experiment and evaluation

The authors evaluated LBT in a lightweight MMORPG simulation. The simulation used combat-oriented quest scenarios. They collected behavior data from 25 participants. Participants were divided into two groups and completed similar tasks in two areas. Each player's recorded trajectory served as an evaluation unit. The authors built two Behavior Trees, Btree1 and Btree2. The trees had equivalent functionality but different decision structures. The goal was to test if LBT could learn behavior from different tree structures.

The experiment compared two schemes. One was the full LBT adaptation. The other was a baseline using only the Delay Modifier. The main metric was the proportion of player actions that the model could not explain in each trajectory. Lower values mean the model explained more observed player behavior. Across all datasets and Behavior Tree structures, the full LBT model had lower unexplained-action ratios than the baseline. The difference was significant ($p < 0.01$). This shows that Modifier mechanisms beyond timing adjustments also help NPCs adapt. Comparisons across datasets and tree structures also suggest that LBT works with different behavioral scenarios and decision structures.

The results show that LBT improves behavioral match between NPCs and players. But the evaluation has limitations. It relies only on objective behavioral metrics. It was conducted in one simulation environment. It did not measure players' subjective gameplay experience directly. Therefore, the current results do not prove that diverse NPC behaviors from LBT improve perceived challenge, immersion, engagement, or enjoyment. Further player-centered experiments are needed. These experiments should establish a direct relationship between adaptive NPC behavior and player experience.

3.3. Contributions

For practical game AI development, LBT has clear advantages. It improves NPC behavioral adaptability.

LBT introduces a learning mechanism into traditional static Behavior Trees. This mechanism uses player trajectories. In the past, NPCs relied mostly on manually predefined rules. With LBT, NPCs

DOI: 10.54254/2755-2721/2026.39387

can adjust their decisions according to observed player behavior. This expands NPC behavioral capabilities. It also reduces repetitive and predictable behaviors. Such behaviors are common in scripted NPC systems. LBT also uses a non-invasive optimization strategy. It adds Modifier nodes to the existing tree. This preserves the structural advantages of Behavior Trees. It increases behavioral diversity without adding much development complexity.

From the perspective of the research gap, LBT offers a practical way to balance NPC behavioral diversity and engineering controllability. The original paper compared two schemes in the experiments. One scheme used the full LBT adaptation. The other used only the Delay Modifier as a baseline. The results suggest that Modifier mechanisms beyond the Delay Modifier provide additional adaptive effects. This provides an experimental reference for future research on Behavior Tree learning.

3.4. Limitations

LBT's modifier-based framework can improve NPC adaptability. But the evaluation mainly uses objective behavioral metrics. It does not capture players' subjective experiences. Therefore, it is difficult to determine whether increased behavioral diversity actually improves gameplay experience.

LBT's learning is also constrained by the initial Behavior Tree structure. Compared with end-to-end data-driven methods, LBT only adjusts existing logic locally. It cannot easily create entirely new decision branches outside the original tree. When player behavior deviates strongly from predefined logic, LBT may fail to adapt.

The framework mainly models discrete interactive behaviors. It has limited ability to represent continuous movement or open-ended exploration. The evaluation used a simplified MMORPG simulation. The participant sample was relatively small. Therefore, the generalizability of LBT to commercial games requires further validation. investigation and validation.

3.5. Discussion

LBT brings player behavior data into traditional Behavior Tree adaptation. It collects and observes player actions. Modifiers adjust the decision logic. This reduces some limitations of traditional Behavior Trees. NPC behavioral diversity can be increased without major changes to the existing tree structure.

Current LBT evaluations remain at the behavioral level. The framework does not directly link algorithmic improvements to subjective player experience. More player-centered experiments are needed. These experiments should collect player feedback and test whether adaptive NPC behavior improves gameplay experience.

Improving player experience through NPC behavior also requires work beyond algorithms. This includes design-level control of generated behaviors. LBT changes the decision process by observing player behavior. It mainly addresses the algorithmic side of NPC adaptation. Parameterized Behavior Trees provide another way to adjust NPC behavior through explicit parameters. These parameters can be used to control aspects of NPC behavior, such as personality, game difficulty, and combat dynamics. Design-level methods offer more flexibility for shaping NPC behavior. They can connect AI algorithms with actual gameplay experience. Player feedback on changes in NPC behavior remains important. Future research should examine the relationship among NPC algorithm design, behavior regulation, and player experience evaluation.

DOI: 10.54254/2755-2721/2026.39387

4. Parameterized behavior trees for flexible NPC design

Shoulson et al. [8] introduced a parameterization method for Behavior Trees. Nodes and subtrees can receive explicit arguments. Traditional Behavior Trees often store data on a shared blackboard. Managing this data becomes hard when several subtrees read or change the same values. Parallel branches may access the same data at the same time. The proposed method reduces this problem. Each parameterized subtree gets its own parameter scope. Required parameters are then exposed to the parent tree.

The framework describes an agent with two parts: traits and capabilities. Traits store information about the agent. Examples are strength or tiredness. Capabilities describe actions the agent can perform. Examples are walking or grasping an object. In a Behavior Tree, assertion nodes use traits. Action nodes use capabilities.

Node parameters come from three sources. A value can be written directly into the tree. It can come from the agent or the environment. The third option is a PAR argument. A PAR argument is passed into the subtree from another part of the tree. The paper gives an example: an agent's tiredness value can be used as input for different actions.

PAR arguments also support subtree reuse. A designer can build a subtree with arguments. The subtree is saved as a PAR tree. Later, a lookup node can call the subtree. The subtree's arguments appear as parameters of that lookup node. Parameter values can pass through several nested subtrees. This is similar to passing arguments through nested function calls in programming.

Compared with a non-parameterized Behavior Tree, this method makes data flow easier to follow. Without an explicit interface, values may be stored on a shared blackboard. Or designers may create similar subtrees for different objects or agents. With parameterization, the same subtree can be reused with different inputs. This reduces repeated behavior logic. It also makes it easier to see where each value comes from.

The paper explains the method through a haggling example. The scenario includes a buyer, a seller, and a flower pot. First, the authors create a reusable `PickUp` PAR tree with an `item` argument. The subtree checks whether the item's weight is less than or equal to the agent's strength. If the condition is true, the agent bends down, grasps the object, and stands up.

The larger event tree controls the haggling process. The price starts at seven, and the buyer tries to reduce it to four. Each haggling attempt has a success probability of 0.5. If the final price is below four, the buyer calls the `PickUp` subtree with the flower pot as the argument. The buyer then walks to the exit. The buyer and seller are treated as roles in the event rather than fixed global agents. This means other compatible agents can use the same event structure.

The example shows how parameterization changes subtree reuse. The `PickUp` logic does not need to be rewritten for every object or agent. Only the argument changes. However, the method does not create new behavior by itself. The tree structure, available parameters, and parameter interfaces are still defined by the designer.

Shoulson et al. also implemented the method in Topiary. Topiary is a graphical Behavior Tree authoring tool connected to the ADAPT simulation platform and Unity. The process begins with an agent model written as a C# class. Traits and capabilities are marked in the class and compiled into an agent DLL. Topiary reads this information and creates the available assertion and action nodes. Designers can then build PAR trees, enter parameter values, and reuse other PAR trees through lookup nodes. The final trees can be exported as C# scripts for Unity.

The paper mainly provides a design example and a tool prototype. It does not report controlled comparisons of development time, runtime performance, error rates, or player experience. The hag-

DOI: 10.54254/2755-2721/2026.39387

gling example shows that explicit parameter passing and subtree reuse are possible, but it does not measure whether these features improve gameplay. For action-game NPCs, similar parameterized subtrees could be used with different targets, objects, or agent roles. This is a possible application of the framework, not a result tested directly in the paper.

5. Player experience perspective

In [9], Tulk Jesso *et al.* examine how players identify and evaluate human and AI co-player behavior. It assesses whether NPCs can reproduce specific behaviors. It also asks whether players can identify a co-player from nonverbal behavior and how those behaviors relate to perceived humanness, trust, and social interaction. The researchers collected player experience data through a behavioral Turing test, in-game behavior logs, questionnaires, and interviews.

The study took place in *Don't Starve Together*. This open-world multiplayer survival game lets players gather resources, craft tools, and cooperate or compete with others. The researchers recruited 83 undergraduate students. They excluded 17 participants because of technical problems, game failures, or incomplete data, leaving 66 sets of data for analysis. Each participant interacted with one co-player for about 30 minutes without verbal communication. The co-players belonged to three groups: real human players ($n = 22$), NPCs using a simple behavior tree ($n = 22$), and social NPCs that combined a behavior tree with a Partially Observable Markov Decision Process (POMDP) ($n = 22$). The study therefore provides direct evidence about player perception in a multiplayer survival setting, but it does not directly examine hostile or combat-oriented NPC behavior.

5.1. Research framework and experimental design

The experiment included several methodological controls. The researchers disabled in-game chat to isolate behavioral cues. The game automatically recorded the distance between the participant's character and the co-player's character (approximately measured in centimeters from the on-screen distance) and the frequency of interaction. The records included giving items to or receiving items from the co-player, attacking or being attacked, and jointly hunting or fighting game creatures capable of retaliating. After the game, the researchers used questionnaires and interviews to collect participants' subjective evaluations of the co-player. Participants were randomly assigned to experimental conditions and had not previously been exposed to the game.

Although *Don't Starve Together* relies heavily on cooperative survival, the interactions in the experiment were not purely cooperative. Participants could follow, help, exchange resources with, ignore, compete with, or attack the co-player. The setting therefore allowed both cooperative and competitive social behavior. It was not the same as the structured hostile combat encounters typically involving enemy NPCs.

5.2. Experimental results and main findings

Participants did not reliably distinguish human players from AI in the behavioral Turing test. They identified the human co-player as human in 45% of cases, the simple NPC in 23% of cases, and the social NPC in 32% of cases. Accuracy was 61% for the human versus simple NPC comparison and 57% for the human versus social NPC comparison. Neither result was significantly above chance. Participants could observe the co-player's behavior, but they could not reliably determine the co-player's identity from that behavior alone [9].

Proximity was measured as the distance between the participant's character and the co-player's character. It served as an indicator of spatial interaction. The average distance was smallest for the

DOI: 10.54254/2755-2721/2026.39387

Table 2: Comparison of Perceived Humanness and Trust

Measure	Human Player	Social NPC	Simple NPC
Real-person rating (/7)	5.0 ± 2.2	5.1 ± 2.3	3.3 ± 2.3
Trust rating (/7)	4.1 ± 2.6	2.4 ± 1.5	—
Identified as human (%)	45	32	23

simple NPC (40.9 cm), largest for the human co-player (109.5 cm), and intermediate for the social NPC (87.0 cm). The distance between the simple NPC and each of the other two co-players differed significantly. The difference between the human co-player and the social NPC was not significant [9]. Distance, however, is not a direct measure of trust. A player may stay close to interact with or observe the other party, or to judge the other party's identity. Moving away may indicate distrust. The opposite is also possible: a player who trusts a cooperative partner may let that partner move away to complete a task without continuous monitoring. The paper therefore gives a cautious interpretation. The social NPC's navigation was closer to the human player's navigation than to the simple NPC's navigation. This does not show that the social NPC received more trust. The paper also notes that the simple AI stayed closest to the participant, although its ability to understand social behavior and social contexts was limited. Proximity to the simple AI did not necessarily indicate meaningful social interaction.

The interaction reciprocity analysis found differences in the co-players' interaction initiative. The scores for the human co-player and the simple NPC were $+16.2\%$ and $+18.5\%$, respectively. In these two conditions, participants initiated slightly more interactions than the co-player. The social NPC received a score of -33.6%. It initiated substantially more interactions than the participant. The social NPC was therefore more socially active. This result only shows that it initiated contact more often. It does not by itself show that participants trusted it more [9].

The researchers also used questionnaires to compare participants' ratings of co-player humanness and their trust in the different co-players. The results are shown in Table 2.

One main finding is that perceived humanness and trust are different dimensions. The social NPC received a real-person rating of 5.1/7, which did not differ significantly from the human co-player's rating of 5.0/7. The simple NPC received a lower rating of 3.3/7. Participants gave the social NPC a trust rating of 2.4/7, compared with 4.1/7 for the human co-player. Thus, in this experiment, the social NPC appeared more human but did not receive the same level of trust as the human player. The paper suggests that participants may have viewed some social NPC behaviors as random or unpredictable. This may have contributed to the difference, but the study did not directly test the causal link [9].

The qualitative analysis examined the behavioral cues that players used to judge humanness. Participants' reports that the co-player interacted with them were significantly associated with higher humanness ratings. Under the social AI condition, 32% of participants identified random or unpredictable behavior as a cue for their judgment [9]. Social interaction can therefore make an NPC appear more human to players. Interaction that lacks intelligibility may also affect how players evaluate the behavior.

Taken together, the results do not show that one NPC performed best on every measure. The social NPC appeared more human in navigation and social interaction and showed greater social initiative. Player experience also depended on how players understood these behaviors and whether they were willing to trust the NPC.

DOI: 10.54254/2755-2721/2026.39387

5.3. Contribution to behavior tree research

The study provides player-based evidence that a hybrid social-agent architecture composed of a behavior tree and a POMDP can produce behavior in a complex game environment that is difficult to distinguish from human behavior. Participants identified the social NPC as human in 32% of cases. This was above the 30% reference threshold commonly associated with the classic Turing test. The comparison requires caution because the study used a nonverbal behavioral Turing test rather than the original conversational Turing test. The result shows behavioral credibility in a specific game context. It does not show that the architecture has passed the classic Turing test.

Higher perceived humanness also did not produce a more positive overall evaluation. The social NPC appeared more human, but participants trusted it less than the human co-player. This matters for player-centered evaluation. Technical complexity in behavior trees and POMDPs may describe different aspects of a system than players' subjective evaluations do.

One explanation is that social interaction needs to be considered together with behavioral coherence. The social NPC initiated more interactions than the human co-player and the simple NPC. Participants also described some of its behavior as unpredictable. These findings do not show that more social interaction reduces trust. They suggest that increasing interaction frequency or behavioral complexity may not be enough when players cannot understand the logic behind an NPC's behavior.

Humanness and trust should therefore be treated separately. In this experiment, higher perceived humanness did not result in equally high trust. The evidence supports treating them as two distinct dimensions in player evaluation. It does not support the assumption that improving one will necessarily improve the other.

5.4. Limitations and methodological considerations

This study has a key limitation: its context. The work looks at player responses in *Don't Starve Together*. That game is multiplayer and survival-based. It allows both cooperation and competition. The study does not test enemy NPCs in a hostile combat system. So the findings may not apply directly to combat games. We also cannot assume that the link between unpredictability, humanness, and trust stays the same in a fighting setting.

This difference matters for behavior tree research. In a social survival game, unpredictable actions might make a co-player's social intentions unclear. In combat, the same feature could have a different role. It might introduce tactical variety. It could break repetitive attack patterns. Or it might raise the challenge. The study does not show whether unpredictability would help or hurt player evaluations in combat.

Future work could apply similar player-centered measures to combat-oriented settings. One example is fights like those in *Dark Souls*. That would test if the relationship between behavioral unpredictability and player perception changes across game types. It would also give more direct evidence for designing BT-based NPCs in hostile environments. Right now we rely on a social survival experiment, which is less direct.

The participant sample also limits what we can conclude. All participants were undergraduates. Each condition had only 22 people. The authors also mention that longer interactions would need more training and adjustments to the social AI's ability to sense and react to social cues [9]. So some of the observed unpredictability may come from this specific system. It may not be a general shortcoming of POMDP-based social reasoning.

DOI: 10.54254/2755-2721/2026.39387

6. Conclusion

This paper looks at behavior trees in combat experience from three angles. These are algorithmic optimization, NPC parameterization, and player experience evaluation.

The three studies show a clear direction in the research. Learning Behavior Trees aim at algorithmic adaptability. They observe player behavior to reduce deviations. Parameterized Behavior Trees use explicit parameters. This gives designers more control, reusability, and flexibility. The player experience study examines how players perceive and react to AI behavior. These studies together show that adaptability and design flexibility do not guarantee better player experience.

Learning Behavior Trees can cut down behavioral inconsistency by observing player behavior ($p < 0.01$). Parameterized design gives designers more flexibility to control and reuse NPC behavior. The player-centered study, however, finds that behavioral realism does not always lead to player trust. The social AI received a real-person rating close to that of a human player (5.1/7 versus 5.0/7). But its trust rating was much lower (2.4/7 versus 4.1/7; overall difference among co-player identities: $p = 0.045$). This outcome does not prove a universal link between humanness and trust. It suggests that behavioral intelligibility may matter alongside human-like behavior.

Across these three parts, evaluating NPCs requires more than looking at behavioral complexity and parameter flexibility. We also need to consider whether behavior is coherent, fits the character and game context, and is understandable to players. The social AI's survival behavior was mainly controlled by a behavior tree. Its social interaction was adjusted through a POMDP based on inferred social context. This design can increase behavioral diversity. Yet it may also produce strategic shifts that players find hard to explain. This interpretation fits the observation that the social AI was seen as random or unpredictable. But it remains a hypothesis. It is not a causal mechanism directly shown by the study.

This distinction matters for combat-oriented NPCs. Combat NPCs can use tactical variation and some uncertainty to reduce repetition and increase challenge. The study cannot show how such features would affect player experience in combat. So it is useful to separate behavior that is hard to predict from behavior that is hard to explain. The first may create tactical variation. The second may weaken players' understanding of and trust in the NPC's intentions.

These findings point to a cautious design principle. Human-like behavior is not the only sign of successful NPC design. Behavioral diversity, adaptability, coherence, and player understanding are separate factors to evaluate. They may interact differently across game genres. For social agents, intelligible intentions and reliable interaction may be especially important. For hostile agents, more tactical uncertainty may be acceptable. Empirical research is still needed to determine how important each factor is. Results from one game context do not directly apply to another.

Taken together, these studies suggest a possible gap between technical BT improvements and player-centered outcomes. Adaptive and parameterized architectures can increase behavioral diversity and design flexibility. Their effects on player experience may depend on whether behavioral changes stay consistent with the character and game context and are understandable to players. Adaptive components may introduce strategic changes that players find hard to explain. This interpretation is still a hypothesis. It is not a mechanism directly shown by the reviewed studies.

Future research could: (1) examine behavioral coherence when parameter changes or adaptive decisions alter NPC behavior; (2) investigate whether contextual, tactical, or narrative cues help players understand the AI's adaptation; (3) compare these effects across social, cooperative, competitive, and hostile game contexts; and (4) integrate player experience frameworks such as PENS, GEQ, GUESS, and NASA-TLX more systematically into behavior tree evaluation.

DOI: 10.54254/2755-2721/2026.39387

Behavior tree research can therefore move beyond technical complexity and study how players perceive and understand adaptive behavior. Combining algorithmic adaptability, designer control, and direct player evidence may offer a stronger basis for improving NPC combat experience.

Acknowledgment

We sincerely thank Prof. William Nace and our TA Vicky for their guidance and support. Siyao Wu, Zhongyu Chang, and Mingzhen Li contributed equally to this work and should be regarded as co-first authors.

References

[1] Ian Millington. *AI for Games*. 3rd. CRC Press, 2019.

[2] Michele Colledanchise and Petter Ögren. *Behavior Trees in Robotics and AI: An Introduction*. CRC Press, 2018.

[3] Richard M. Ryan, C. Scott Rigby, and Andrew Przybylski. "The Motivational Pull of Video Games: A Self-Determination Theory Approach". In: *Motivation and Emotion* 30.4 (2006), pp. 344–360.

[4] Wijnand A. IJsselsteijn, Yvonne A. W. de Kort, and Karolien Poels. "The Game Experience Questionnaire". In: *Proceedings of the FUGA Workshop*. 2013.

[5] Mikki H. Phan, Joseph R. Keebler, and Barbara S. Chaparro. "The Development and Validation of the Game User Experience Satisfaction Scale (GUESS)". In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. Vol. 60. 1. 2016, pp. 1–5.

[6] Sandra G. Hart and Lowell E. Staveland. "Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research". In: *Advances in Psychology* 52 (1988), pp. 139–183.

[7] Emmett Tomai and Roberto Flores. "Learning Behavior Trees from Demonstration". In: *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)*. 2014, pp. 1–8.

[8] Alexander Shoulson et al. "Parameterizing Behavior Trees". In: *Motion in Games*. Vol. 7060. Lecture Notes in Computer Science. Springer, 2011, pp. 144–155.

[9] Stephanie Tulk Jesso, William G. Kennedy, and Eva Wiese. "Behavioral Cues of Humanness in Complex Environments: How People Engage With Human and Artificially Intelligent Agents in a Multiplayer Videogame". In: *Frontiers in Robotics and AI* 7 (2020), p. 531805. DOI: 10.3389/frobt.2020.531805.

[10] Damian Isla. "Handling Complexity in the Halo 2 AI". In: *Proceedings of the Game Developers Conference (GDC)*. 2005.

[11] Alex J. Champandard. *Understanding Behavior Trees*. http://aigamedev.com/open/article/bt-overview/. Accessed: 2026-07-27. 2007.

[12] Josef Wiemeyer et al. "Player Experience". In: *Dagstuhl Reports* 6.12 (2016), pp. 65–78.