Player-Centric Procedural Content Generation: Enhancing Runtime Customization by Integrating Real-Time Player Feedback

Nancy N. Blackburn*
nancy.n.blackburn@utah.edu
University of Utah
Salt Lake City, UT, USA

M. Gardone*
m.gardone@utah.edu
University of Utah
Salt Lake City, UT, USA

Daniel S. Brown
daniel.s.brown@utah.edu
University of Utah
Salt Lake City, UT, USA

ABSTRACT

Game content creation poses significant challenges, particularly for indie developers and small teams, as it presents difficulties in scaling to meet diverse player preferences. Adaptable procedural content generation (PCG) provides a promising solution to this issue. Extensive literature exists on adaptable PCG techniques, encompassing both offline (during development) and online (during gameplay) approaches. Building upon this foundation, we propose a novel extension called Player-Centric Procedural Content Generation (PCPCG) as an additional tool for creating unique game experiences based on player preferences. In contrast to runtime adaptable PCG methods, which adapt or learn based solely on in-game data, PCPCG actively involves players in the learning loop by soliciting their feedback. Additionally, it shifts the focus from designers (as in mixed initiative (MI) and co-creative PCG) to the players as providers of learning data, thus operating during gameplay rather than during development. PCPCG possesses three key qualities: 1) real-time operation during gameplay, 2) active participation of players (not designers) in the learning loop, and 3) online learning from player feedback to create engaging and personalized content. It is important to differentiate PCPCG from content creation aids and in-game data adaptable PCGs. While PCPCG falls under the umbrella of adaptable PCG, it goes beyond relying solely on in-game data by incorporating valuable player feedback as a vital information source for content generation. PCPCG introduces a novel and promising approach to runtime procedural content generation by leveraging player feedback to create adaptive and personalized game content. While our proof of concept demonstrates the viability of PCPCG in a Pac-Man domain, further research is required to explore its limitations as the complexity of the possibility space increases.

CCS CONCEPTS

• **Human-centered computing** → *Interactive systems and tools*; • **Applied computing** → **Computer games**.

*These authors contributed equally to this research.

CHI PLAY Companion '23, October 10–13, 2023, Stratford, ON, Canada

ACM ISBN 979-8-4007-0029-3/23/10.
https://doi.org/10.1145/3573382.3616069

KEYWORDS

personalization; adaption; gameplay; preference learning; procedural content generation

ACM Reference Format:
Nancy N. Blackburn, M. Gardone, and Daniel S. Brown. 2023. Player-Centric Procedural Content Generation: Enhancing Runtime Customization by Integrating Real-Time Player Feedback . In *Companion Proceedings of the Annual Symposium on Computer-Human Interaction in Play (CHI PLAY Companion '23), October 10–13, 2023, Stratford, ON, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3573382.3616069

1 INTRODUCTION

Manually creating game content is a time-consuming task performed by expert game designers. Procedural content generators (PCGs) are used to address this challenge and enhance gameplay experience [8, 14]. Adaptable procedural content generation [25] falls into two categories: 1) offline, involving human refinement of generated content, and 2) online, adapting content during runtime. However, the online techniques have predominantly been applied to game AI behavior [28] and difficulty adjustment [32], and do not explicitly solicit player feedback.

To address this limitation, we propose Player-Centric Procedural Content Generation (PCPCG), which incorporates player preferences into the content creation process by actively soliciting players for their feedback during gameplay. Our research aims to answer key questions: (1) How can we cater to individual player preferences during runtime? (2) How can we apply preference learning with minimal agent training and sparse player feedback for PCGs? (3) How can PCPCG account for changing player preferences during runtime? (4) What is the best way to gather feedback from players that maximizes learning?

PCPCG aims to create immersive, personalized game experiences by leveraging in-game player feedback. It integrates runtime player feedback as a valuable source of information for content generation, continually adapting to evolving player preferences.

Our contribution is the proposal of PCPCG as a family of runtime PCG algorithms that create content based on individual player preferences. We establish criteria for these algorithms and apply preference learning to runtime adaptable PCG for level generation with a low dimensionality feature set. We present our methodology, findings, discuss study limitations, and provide an outlook on future research directions.

2 RELATED WORK

In this section, we explore adaptive procedural content generation, player-generator interactions, and preference learning algorithms from machine learning.

Co-Creative Adaptive PCG agents collaborate with human game designers to create content, allowing for offline refinement of generated content [2, 8, 11, 12, 16, 18]. Our focus is on runtime adaptation to individual player preferences, distinguishing our work from these offline approaches.

Dynamic Difficulty Adjustment (DDA) adjusts game levels based on player skill determined from in-game data [10, 13, 20] but does not consider player preferences regarding content itself. Our work directly asks players for their preferences and generates content accordingly, focusing on feedback directly from the player.

One Look Learning (OLL) [1, 28–30] adapts game content based on in-game data but has not been applied to PCG for game levels and does not learn from explicit feedback.

Player-Generator Interaction (Offline and Online). Player interaction with PCG systems can occur offline (e.g., parameterization, direct manipulation) or online (e.g., preference-based interactions) [24]. As it relates to our work, preference-based interactions generate game content based on inferred player preferences, either with or without the player's explicit knowledge. However, this approach has limitations in capturing complex player preferences and may not scale well to higher-dimensional problems like level or narrative generation. It is important to note that preference-based interactions can influence the PCG system but do not inherently possess the ability to adapt to changing player preferences over time. Our approach combines preference learning, sparse feedback, and player involvement to generate personalized game content in real-time.

Preference Learning (PL) and Human-in-the-Loop Machine Learning algorithms involve human feedback to learn subjective preferences that are often hard to model mathematically [4, 7, 17, 22, 23, 26]. However, most prior work assumes stationary preferences, whereas preferences over game content may change over time. Many algorithms aim to gather meaningful information with minimal disruption [6, 9, 21, 27]. However, previous approaches often require extensive interactive feedback, which may not align with player tolerance for interruptions. Our work seeks to learn human preferences over game content using sparse feedback.

2.1 Inspiration for PCPCG

The related works provide insights for the development of Player-Centric Procedural Content Generation (PCPCG), which combines preference learning, sparse feedback, and player involvement.

PCPCG utilizes runtime adaptation like OLL, DDA, and preference player-generator interaction PCGs. However, it explicitly involves the player by soliciting their feedback, addressing limitations of solely relying on in-game data. Our PCPCG agent is trained in real-time as the player plays, mitigating the issue of lengthy training sessions. Sparse feedback is collected non-intrusively, with a two-point Likert scale at the end of a level. By integrating preference learning into the PCG process and adapting it to an online context, PCPCG continuously learns and adapts to player preferences as they evolve, facilitating the generation of personalized game content aligned with individual player preferences. This approach has the potential to promote player engagement and enjoyment throughout the gameplay experience.

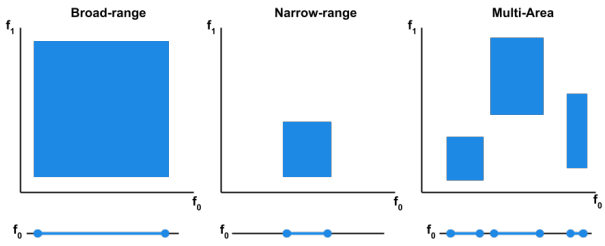


Figure 1: Player preferences shown in 1D and 2D. The player's preference equals the union of all preferred areas (shown as the shaded regions) across all features.

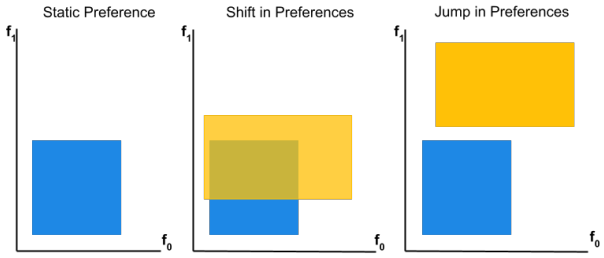


Figure 2: Player preference change types: static (the preference remains the same), shift (the new preference overlaps with the old preference), and jump (the new preference is in a completely new region from the old one with no overlap).

3 PCPCG FORMAL DEFINITION

In order to deploy an algorithm that can adapt to human preferences as the player interacts with the system with minimal training data and sparse feedback, the PCPCG (Player-Centric Procedural Content Generation) approach has three constraints: 1) Require only minimal training: the PCPCG algorithm aims to minimize the amount of training required from the player. Lengthy training sessions may not align with players' tolerance for interruptions and their decreased patience for extended training periods. 2) Learn during runtime: unlike traditional preference learning approaches, which rely on extensive pre-training data, the PCPCG algorithm is designed to learn and adapt in real-time as the player engages with the system. This online approach allows for continuous adaptation to changing player preferences during gameplay. 3) Utilize sparse feedback: the PCPCG approach seeks to gather meaningful feedback from the player while minimizing interruptions to their gameplay experience. Sparse feedback is obtained by requesting feedback infrequently and using simple data formats, such as a two-point Likert scale (like or dislike), presented to the player at natural break points in gameplay.

3.1 Modeling Player Preferences

In our implementation of PCPCG, we represent player preferences as zones within a set of features. Each feature has a range of values,

and the player's preference occurs as a subset of the feature's value range. We modeled potential preferences by defining three different ranges of preference: 1) *broad-ranges* representing one vast zone of preference, 2) *narrow-ranges* representing one small zone of preference, and 3) *multi-area ranges* where the player preference is the union of multiple disjointed zones (Illustrated in Figure 1). Additionally, we modeled these preferences across three different mutable types: *static* preferences, where the preference zone remains the same throughout the duration of play, and *dynamic* preferences, which can be further classified as *shift* or *jump*. Shift preferences occur when the preference zone shifts from the original zone but overlaps some of the old zone, while a jump in preferences occurs when the player's preferences move to an entirely new zone with zero overlap of the previous zone. This is illustrated in Figure 2.

At the start of the game, a player is presented content that is generated from a given feature vector. The feature vector's components span all features; each component is an element of a feature's range. The player has a certain probability of liking the given game content as $P(\Lambda)$ (See equation 1). $P(\Lambda)$ is calculated as the number

$$P(\Lambda) = \frac{F_{\text{satisfied}}}{F_{\text{total}}} \tag{1}$$

of components in the feature vector that satisfy the player's preference , where $F_{\text{satisfied}}$ is the number of features that satisfy the player's preference and F_{total} is the total number of components in the feature vector. When all preferences are satisfied, the probability of the player liking the content is one (i.e., $P(\Lambda) = 1$).

3.2 PCPCG Algorithm

The PCPCG algorithm is structured as follows: First, the player is presented with game content generated based on a given feature vector (for our case, the PCPCG always starts this process from the same feature vector). Second, once the player has completed the level, they are asked to provide feedback on the content. The system then stores what the player did or did not like and creates a new feature vector within some step distance away from the current feature vector. Finally, with the new values, new content is generated. The process repeats until the player ends the game (see Algorithm 1).

Algorithm 1: Basic PCPCG Framework

```
Data: oldPref: the prior preference used, rating: user's input
      on level, ss: step size for the next preference
Result: newPref
prefs ← (oldPref, rating) /* Record samples */;
newPref ← NULL;
for p ∈ oldPref do
    prefVal ← GetNextPreference(p, ss);
    while prefVal ∈ prefs.negative do
        prefVal ← GetNextPreference(p, ss);
    end
    newPref.append(p.name, prefVal);
end
```

The feedback PCPCG requests from the player only happens at a natural break in play, i.e., at the end of the level. To facilitate moving quickly back to play, the feedback solicitation is simple: a 2-point Likert scale is presented to the player asking how they felt about the level: "like" or "dislike." The PCPCG system stores a label for the content based on the player's selection. Gathering feedback through this simple form is a very noisy process, meaning that unless all potential features are satisfied, it is unclear whether a player will like or dislike the level. To store the player's feedback, we employed different approaches based on the type of preference. Under static preferences, a list structure was used. However, dynamic preferences required a way of forgetting old preferences, which led us to utilize a ring buffer.

Given the labels, PCPCG learned player preferences by utilizing a rejection-based sampling algorithm [5], as illustrated in Figure 3. Rejection-based sampling algorithms are commonly used for tasks such as GAN discriminators [3]. PCPCG starts from the same feature vector for each player. An ϵ-ball (epsilon ball) centered on the feature vector is created. PCPCG generates a level from the feature vector and, based on the generated level, the player likes (accepts) or dislikes (rejects) it after completing the level (win or loss). The player can be a human or a synthetic human. For our synthetic humans, the probability of liking the level is calculated as exactly described in Equation 1. Given the feedback, a label is applied to the ϵ-ball which stores the player's preference. If the player likes the feature vector, then PCPCG may select a feature vector from within the ϵ-ball region in a future iteration. If the player rejects the feature vector, then the ϵ-ball becomes a dead zone where PCPCG cannot create any new feature vectors. To generate a new feature vector, PCPCG samples a new feature vector within a predetermined maximum step distance from the current feature vector. This new feature vector is then used to create a new level for the next iteration of the algorithm.

4 METHODOLOGY

This section presents the methodology employed in the study, outlining the implementation of PCPCG (Player-Centric Procedural Content Generation) and the testing process.

4.1 Implementation of PCPCG

We started with Zigurous's openly available source code [31], which provided basic gameplay and art assets based on the original Pac-Man [19] game. To this we added the PCPCG algorithm. The PCG in PCPCG is grammar-based [24], where the possibility space comprises the set of design features to be manipulated. The preference learning agent introduces constraints on the possibility space by creating dead zones based on player feedback obtained during gameplay. The interpreter then utilizes the newly specified possibility space to randomly generate a single new level within those constraints, which is then presented to the player in-game.

4.2 Possibility Space

To demonstrate the proof of concept, we focused on manipulating a specific aspect of the game: the placement of pellets. This choice was made to keep the complexity manageable during the initial implementation. Various features were selected to represent the pellet placement.

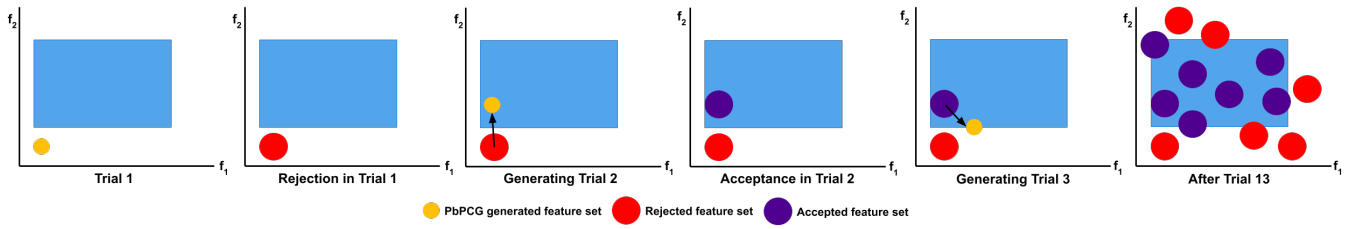


Figure 3: An illustration of how PCPCG performs preference learning through rejection-based sampling. Each trial is generated based on the last sample. The blue/lighter rectangle is the player's preferences, every point outside that region is not favorable. Red/Light circles indicate the sample was rejected; darker/purple circles indicate the sample was accepted sample.

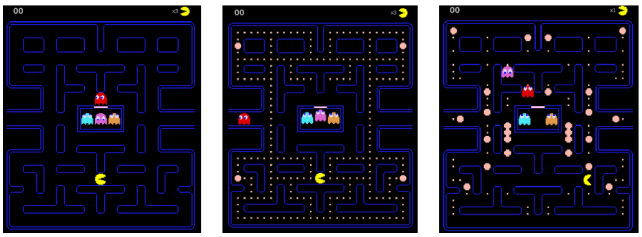


Figure 4: The original *Pac-Man* map without (left) and with (center) pellets versus a PCPCG generated layout (right). (*Pac-Man* art assets are provided by Zigurous.)

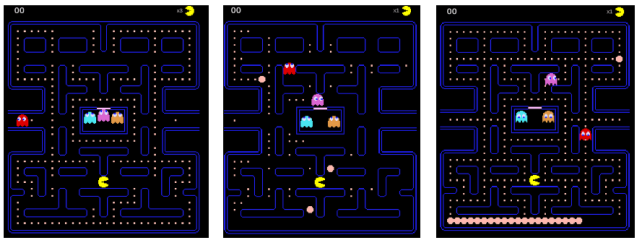


Figure 5: PCPCG generated layouts. Pac-Man has not eaten any of the pellets in these maps. (*Pac-Man* art assets are provided by Zigurous.)

(1) **Power pellet density** ($D_{pp} = \frac{pp}{tp}$, Range: $[0, 1] \in \mathbb{R}$): Ratio of power pellets (pp) to total pellets (tp). Density of regular pellets (rp) is implicitly calculated as $D_{rp} = 1 - D_{pp}$.
(2) **Total pellet count** ($C_{tp} = \text{rp} + \text{pp}$, Range: $[1, 294] \in \mathbb{N}$): Total pellets (tp) on the map. The minimum is one as zero represents the win state.
(3) **Total pellet density** ($D_{tp} = \frac{tp}{s}$, Range: $(0, 1] \in \mathbb{R}$): Ratio of total pellets to all possible pellet tiles on the map (s).
(4) **Symmetry/Asymmetry** (S, Range: $[0, 1] \in \mathbb{W}$): If true, pellets are drawn symmetrically on the map over the vertical axis. Symmetry over the horizontal axis is not tracked because the map is only vertically symmetrical.

4.3 Testing PCPCG

To evaluate the effectiveness of PCPCG, we used randomly generated synthetic humans [4] as players during runtime testing. This approach allowed us to simulate diverse player preferences and behaviors, providing a comprehensive evaluation of the system. Three types of synthetic humans were created representing different preference types: broad-range, narrow-range, and multi-area. The performance of PCPCG was assessed using recall and precision metrics. Static preference testing and dynamic preference testing with different memory models were conducted. The results of the testing are summarized in Table 1, indicating the system's performance in terms of recall and precision [15] for each agent type and feature.

Three types of synthetic humans were created representing the broad-range, narrow-range, and multi-area preference types and tested over static and dynamic mutable preferences. In this work, multi-area ranges consisted of two to four disjoint zones. There are two hyper parameters that exist for the purposes of exploration and knowing the size of the area around a rejected sample: step size and dead zone, respectively. The initial seeding for both static and dynamic agents included 294 total pellets, four power pellets, and a power pellet density of 0.0136. The seed used for bothstatic and dynamic preference testing was 42.

$$\text{recall} = \frac{TP}{TP + FN} \tag{2a}$$

$$\text{precision} = \frac{TP}{TP + FP} \tag{2b}$$

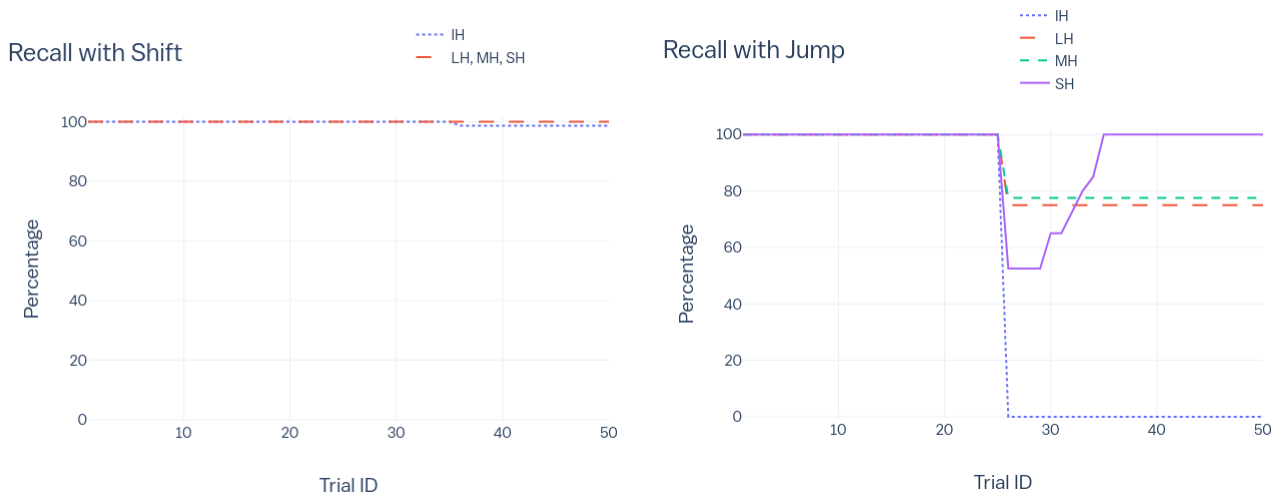
We evaluated PCPCG performance using recall and precision metrics [15], based on true or false positives/negatives. True positives (TP) are correctly labeled preferred areas, while true negatives (TN) are correctly labeled undesired areas. False positives (FP) are incorrectly labeled positive, and false negatives (FN) are incorrectly labeled negative. Using these labels, we calculated recall and precision. Recall measures preservation of relevant preference areas while excluding undesirable areas (Equation 2a). Precision measures the ratio of actual preferred areas over correctly labeled areas and unexplored areas (Equation 2b).

For static preference testing, we generated 10,000 synthetic humans for each of the three preference range types. Each synthetic human underwent ten iterations of the PCPCG learning algorithm (Algorithm 1) 100 times using a new PCPCG agent each time. Average precision and recall were calculated from these 100 runs. In static preference testing, we expected high recall but potentially lower precision due to naive exploration.

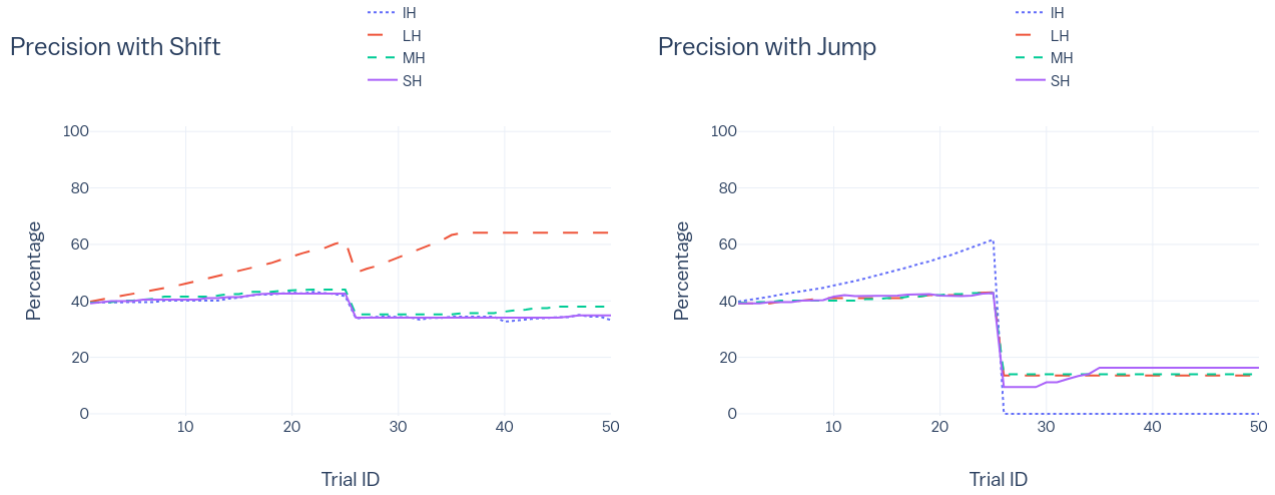
To evaluate dynamic preferences, PCPCG was tested using four memory models: an *infinitely growing list* (**IH**), *short-term* (n=25,

Table 1: This table shows that PCPCG performs well on static preferences with multi-area and broad-range types, achieving high recall values. Precision is lower in the narrow-range case due to the larger disliked area (increasing the likelihood of false positives).

	Average Recall			Average Precision		
Feature	**Multi-Area**	**Broad-Range**	**Narrow-Range**	**Multi-Area**	**Broad-Range**	**Narrow-Range**
Total Pellets	90.729%	89.31%	91.256%	50.650%	54.706%	13.586%
Pellet Density	90.654%	89.96%	98.065%	50.438%	54.794%	14.534%
Power Pellets	92.576%	96.34%	78.112%	51.449%	58.349%	11.916%



(a) Recall with shift and jump preference change.



(b) Precision with shift and jump preference change.

Figure 6: Tracking the ability of PCPCG agents to relearn a player's preference after a change in preferences in the 25th trial. These agents ability to relearn was tracked across four different memory models: *infinitely growing list* (IH), *short-term* (n=25, SH), *medium-term* (n=50, MH), and *long-term* (n=100, LH) horizon ring buffers. In shifting changes, all memory models demonstrate reasonable adaptability to the preference change. However, in jump changes, only the SH model successfully returns to its previous recall values, while the IH model fails to learn the new preference.

SH), *medium-term* (n=50, **MH**), and *long-term* (n=100, **LH**) horizon ring buffers. These memory models allowed us to examine how different temporal horizons influence PCPCG's adaptability. Testing was performed 25 times with the original preference set and 25 more times after a change in preference. Preference changes occurred immediately after obtaining feedback for the 24th level. In dynamic preference testing, we anticipated that the infinitely growing list would struggle to recover after a preference jump, while the other memory models would gradually recover over time. The short horizon ring buffer was expected to exhibit lower maximum recall and precision compared to the others.

The results are summarized in Table 1, showing the average recall and precision values for each agent type and feature. PCPCG performs well on static preferences with multi-area and broad-range types but exhibits lower precision with narrow-range preferences, possibly due to a larger disliked area and higher likelihood of sampling false positives.

In summary, the methodology involved implementing PCPCG and conducting testing with synthetic humans to evaluate its performance in capturing and adapting to player preferences.

5 RESULTS

The results in Table 1 show that PCPCG effectively learns preference regions with high recall. Precision is relatively high for multi-area and broad-range preferences, but lower for narrow-range preferences due to extensive exploration. With more iterations, PCPCG is expected to perform better in narrow-range preferences as more iterations would allow the agent to hone in on the small preference size.

For mutable preference with a shift style, Figure 6a exhibits a minor drop in recall (100% to 98.5%). In Figure 6b, PCPCG's precision remains stable after a shift, with MH and LH models showing recovery. The IH model struggles to recover due to its inability to forget old preferences.

In mutable preference with a jump style, the IH method fails to capture new preferences. The recall and precision of SH, MH, and LH models experience a drop, although not as severe as IH. The SH model exhibits a recovery pattern in recall but struggles in precision. More iterations are likely needed for precision improvement after a jump.

6 LIMITATIONS AND FUTURE WORK

PCPCG has limitations that can lead to interesting future work. The current implementation, being a proof of concept, requires further exploration to assess scalability and effectiveness in diverse scenarios.

One limitation is the reliance on real-time player feedback. While valuable for immediate incorporation of preferences, continuous feedback may be intrusive and bothersome for players. Human studies are necessary to understand these limitations and evaluate the benefits of this feedback approach. Testing PCPCG with human players is also essential to assess its performance and adaptability. Regarding the mechanism for feedback gathering, leveraging everyday interactions as implicit feedback, such as "Skip Level" for dislike and "Replay" for like, could mitigate bothersome feedback requests.

Future research should focus on improving preference learning precision. Alternative approaches, like active SVM learning, can identify specific features contributing to player dislike. Presenting previously liked and disliked content for replay in dynamic preference scenarios could indicate changes in preference and improve system behavior. Enhancing the representation of player preferences is worth exploring. Transitioning to a probability field representation, with values ranging from 0 to 1, would offer a more nuanced understanding. New feature set values could be influenced by the likelihood of a like, allowing exploration within disliked regions.

7 CONCLUSION

PCPCG is a proof of concept that successfully integrates real-time player feedback into games. Through experiments with synthetic humans, PCPCG demonstrated a high level of precision in learning static and dynamic preferences. To fully realize the potential of PCPCG in creating personalized game experiences, further research is required to address its limitations, with a particular emphasis on scalability. By overcoming these challenges, PCPCG has the capacity to deliver highly engaging and personalized game experiences that cater to individual player preferences.

REFERENCES

[1] 2004. Interesting games through stage complexity and topology. (2004). https://www.um.edu.mt/library/oar/handle/123456789/29588

[2] Alberto Alvarez, Jose Font, and Julian Togelius. 2022. Toward Designer Modeling Through Design Style Clustering. *IEEE Transactions on Games* 14, 4 (Dec. 2022), 676–686. https://doi.org/10.1109/TG.2022.3143800 Conference Name: IEEE Transactions on Games.

[3] Samaneh Azadi, Catherine Olsson, Trevor Darrell, Ian Goodfellow, and Augustus Odena. 2018. Discriminator Rejection Sampling. https://doi.org/10.48550/ARXIV.1810.06758

[4] Daniel S. Brown, Russell Coleman, Ravi Srinivasan, and Scott Niekum. 2020. Safe Imitation Learning via Fast Bayesian Reward Inference from Preferences. (2020). http://arxiv.org/abs/2002.09089

[5] George Casella, Christian P. Robert, and Martin T. Wells. 2004. Generalized Accept-Reject Sampling Schemes. *Lecture Notes-Monograph Series* 45 (2004), 342–347. http://www.jstor.org/stable/4356322

[6] Sean Andrew Chen, Jensen Gao, Siddharth Reddy, Glen Berseth, Anca D. Dragan, and Sergey Levine. 2022. ASHA: Assistive Teleoperation via Human-in-the-Loop Reinforcement Learning. *2022 International Conference on Robotics and Automation (ICRA)* (2022), 7505–7512.

[7] Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. (2017). http://arxiv.org/abs/1706.03741

[8] Martin Dahrén. 2021. The Usage of PCG Techniques Within Different Game Genres. (2021), 41.

[9] Jensen Gao, Siddharth Reddy, Glen Berseth, Nicholas Hardy, Nikhilesh Natraj, Karunesh Ganguly, Anca D. Dragan, and Sergey Levine. 2022. X2T: Training an X-to-Text Typing Interface with Online Learning from User Feedback. https://doi.org/10.48550/ARXIV.2203.02072

[10] Miguel González-Duque, Rasmus Berg Palm, and Sebastian Risi. 2021. Fast Game Content Adaptation Through Bayesian-based Player Modelling. arXiv:2105.08484 [cs, stat] http://arxiv.org/abs/2105.08484

[11] Matthew Guzdial, Nicholas Liao, Jonathan Chen, Shao-Yu Chen, Shukan Shah, Vishwa Shah, Joshua Reno, Gillian Smith, and Mark O. Riedl. 2019. Friend, Collaborator, Student, Manager: How Design of an AI-Driven Game Level Editor Affects Creators. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI '19*. ACM Press, Glasgow, Scotland Uk, 1–13. https://doi.org/10.1145/3290605.3300854

[12] Matthew Guzdial, Joshua Reno, Jonathan Chen, Gillian Smith, and Mark Riedl. 2018. Explainable PCGML via Game Design Patterns. arXiv:1809.09419 (Sept. 2018). arXiv:1809.09419 [cs] http://arxiv.org/abs/1809.09419

[13] Robin Hunicke and Vernell Chapman. 2004. AI for Dynamic Difficulty Adjustment in Games.

[14] Mikhail Jacob and Brian Magerko. 2015. Interaction-based Authoring for Scalable Co-creative Agents.. In *ICCC*. 236–243.

[15] M. Junker, R. Hoch, and A. Dengel. 1999. On the evaluation of document analysis components by recall, precision, and accuracy. In *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR '99 (Cat. No.PR00318)*. 713–716. https://doi.org/10.1109/ICDAR.1999.791887

[16] Antonios Liapis, Georgios N. Yannakakis, and Julian Togelius. 2014. Designer modeling for Sentient Sketchbook. In *2014 IEEE Conference on Computational Intelligence and Games* (Dortmund, Germany, 2014-08). IEEE, 1–8. https://doi.org/10.1109/CIG.2014.6932873

[17] Yi Liu, Gaurav Datta, Ellen Novoseller, and Daniel S Brown. 2023. Efficient Preference-Based Reinforcement Learning Using Learned Dynamics Models. In *International Conference on Robotics and Automation (ICRA)*.

[18] Pedro Lucas and Carlos Martinho. 2017. Stay Awhile and Listen to 3Buddy, a Co-creative Level Design Support Tool.. In *ICCC*. 205–212.

[19] Namco. 1980. *Pac-Man*. Arcade.

[20] Dvir Ben Or, Michael Kolomenkin, and Gil Shabat. 2021. DL-DDA – Deep Learning based Dynamic Difficulty Adjustment with UX and Gameplay constraints. https://doi.org/10.48550/ARXIV.2106.03075

[21] Siddharth Reddy, Sergey Levine, and Anca D. Dragan. 2022. First Contact: Unsupervised Human-Machine Co-Adaptation via Mutual Information Maximization. *ArXiv* abs/2205.12381 (2022).

[22] Dorsa Sadigh, Anca D. Dragan, S. Shankar Sastry, and Sanjit A. Seshia. 2017. Active Preference-Based Learning of Reward Functions. In *Proceedings of Robotics: Science and Systems (RSS)*.

[23] Daniel Shin, Anca Dragan, and Daniel S Brown. 2023. Benchmarks and Algorithms for Offline Preference-Based Reward Learning. *Transactions on Machine Learning Research* (2023).

[24] Gillian Smith. 2015. Game AI Pro 2, Procedural Content Generation An Overview.

[25] Julian Togelius, Georgios Yannakakis, Kenneth Stanley, and Cameron Browne. 2011. Search-Based Procedural Content Generation: A Taxonomy and Survey. *Computational Intelligence and AI in Games, IEEE Transactions on* 3 (Oct. 2011), 172–186. https://doi.org/10.1109/TCIAIG.2011.2148116

[26] Christian Wirth, Riad Akrour, Gerhard Neumann, Johannes Fürnkranz, et al. 2017. A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research* 18, 136 (2017), 1–46.

[27] Tengyang Xie, Akanksha Saran, Dylan J. Foster, Lekan Molu, Ida Momennejad, Nan Jiang, Paul Mineiro, and John Langford. 2022. Interaction-Grounded Learning with Action-inclusive Feedback. *ArXiv* abs/2206.08364 (2022).

[28] Georgios N. Yannakakis and John Hallam. 2004. Evolving opponents for interesting interactive computer games. (2004). https://www.um.edu.mt/library/oar/handle/123456789/22948 Accepted: 2017-10-24T08:23:35Z Publisher: MIT PRESS.

[29] Georgios N. Yannakakis and John Hallam. 2005. A generic approach for obtaining higher entertainment in predator/prey computer games. (2005). https://www.um.edu.mt/library/oar/handle/123456789/29602 Accepted: 2018-04-26T09:59:27Z Publisher: Charles River Media.

[30] Georgios N. Yannakakis and John Hallam. 2007. Towards Optimizing Entertainment in Computer Games. *Applied Artificial Intelligence* 21, 10 (Nov. 2007), 933–971. https://doi.org/10.1080/08839510701527580

[31] Zigurous. 2021. *Pac-man (2D) [Source Code]*. https://github.com/zigurous/unity-pacman-tutorial

[32] Mohammad Zohaib. 2018. Dynamic Difficulty Adjustment (DDA) in Computer Games: A Review. 2018 (2018), 1–12. https://doi.org/10.1155/2018/5681652