Playtrace Arc Search: A Tool to Explore and Evaluate Large Spaces of Playtrace Metrics Through User-Defined Curves

Samuel Shields[1], Noah Wardrip-Fruin[1] and Edward F. Melcer[2]

[1]University of California, Santa Cruz, 1156 High St, Santa Cruz, CA 95064, USA
[2]Carleton University, 1125 Colonel By Dr, Ottawa, ON K1S 5B6, Canada

Abstract

Playtraces are artifacts produced during playtesting that tell a story about how game systems operate and what actions players take at runtime. The playtrace contains relevant metrics in a game alongside the metrics' rising and falling throughout player progression. These curves inform designers about their players' experiences and open opportunities to implement player-adaptive design strategies. To help improve the iterative design process around playtrace analysis, we introduce the Playtrace Arc Search (PAS) tool. PAS allows designers to search through a large corpus of playtrace data to find the system metric curves that match their design intent by drawing a desired progression arc on a canvas and then using that to perform a curve-similarity search over all playtraces. PAS enables designers to see a set of playtraces as a summarized whole, then search against that whole to find specific, meaningful gameplay data that can confirm or reject their hypotheses about game and player performance. PAS showed success on an initial evaluation of 1,000 playtraces; this, combined with its game- and metric-agnostic capabilities, indicates that PAS might be a useful tool for designers to rapidly discover whether their design hypotheses are accurately played out in metric progression curves with desired levels of consistency.

Keywords

Game Balancing, Playtrace Analysis, Player Modeling, Dramatic Arc, Design Tools

1. Introduction

A playtrace from a video game is defined by Osborn et al. [1] as "a sequence of player actions corresponding to one play of the game." As players perform actions in a game, observable metrics are produced as a direct or indirect impact of changing the system. Such metrics might include parameters relating to a game goal (e.g., how many items have been collected), bug prevention (e.g., has a player reached a checkpoint in a race too early), or updating a player model (e.g., what level of difficulty the player is experiencing right now). The role of such metrics in design strategies such as Dynamic Game Balancing [2] as well as scoring games for quality in A/B testing [3] helps underscore the utility of metric production at both design time and runtime. These metrics are frequently trivially available due to their critical role in modifying game systems, and are also useful to provide system observability of a game system over a play session. Designers have the responsibility not only to tune input parameters to a game system to ensure a desired play experience, but also to utilize these observable metrics to iterate on their design strategy and confirm their hypotheses about runtime operations of games. The value of producing understandable and meaningful playtraces is, as such, a meaningful step in iterative game design — especially when it comes to human or automated playtesting.

That being said, playtrace collection and analysis rarely involve examining a single, predetermined session of gameplay data. Tools surrounding playtraces usually deal in aggregates of thousands to millions of traces compiled together and visualized in some format. For instance, heatmaps [4, 5], player modeling [6, 7], and balancing patches [8] are just a few examples of how playtrace composites are currently used to augment design strategies. While such tooling is valuable for pattern identification in playtrace datasets, it can put the designer's understanding of their desired, emergent play experience in

Joint AIIDE Workshop on Experimental Artificial Intelligence in Games and Intelligent Narrative Technologies, November 10-11, 2025, Edmonton, Canada.

samshiel@ucsc.edu (S. Shields); nwardrip@ucsc.edu (N. Wardrip-Fruin); edwardmelcer@cunet.carleton.ca (E. F. Melcer)
0000-0001-6372-5656 (S. Shields); 0000-0003-1964-7624 (N. Wardrip-Fruin); 0000-0003-1760-5279 (E. F. Melcer)

the back seat to a global summary of data. Furthermore, such approaches might make it challenging to identify interesting examples of playtraces that could become meaningful case studies for further design investigations. As such, a tool that allows a designer to show what their intended experience is for a set of playtraces and to understand how consistent (or, if desired, diverse) a set of traces is quickly could provide a valuable strategy to rapidly understanding system characteristics during gameplay. Supplementing this with the identification of important examples of their desired experience then provides such an approach with the opportunity to do deeper investigations of important games played and even perform parameter tuning based on a given playtrace match.

To address this gap in designer tooling, we introduce the Playtrace Arc Search (PAS) tool, which enables the searching of a large corpus of playtraces according to a user-defined curve or "arc". PAS reads a set of structured playtrace data, has the user select game and/or system metrics to analyze, then allows the user to "draw" a curve to search for playtrace arcs that fit their arc. It also allows a designer to quickly understand if a set of playtraces exhibit similar or diverse behaviors. In both of these modes, PAS provides an option for rapid iterative design cycles, where the identification of some systematic arc (such as drama, emotional intensity, difficulty) allows a designer to home in on important qualities of a specific gameplay session. By quickly showing a designer where to focus, PAS can tighten the playtesting loop, which is cited as one of the most expensive and time-consuming parts of the game development lifecycle [9]. In this work, we describe the system architecture of PAS, perform a brief set of evaluations on a corpus of 1,000 playtraces, and discuss the design implications of applying this tool in both manual and procedural design contexts. A screenshot of the tool can be seen in Fig. 2, and the tool is available for use on GitHub.[1]

2. Related Work

2.1. Co-Creative Tooling

Co-creative systems and research tooling involves the alteration of a designer's workflow to include computational systems, which helps produce work with greater speed, quality, or expressive range. [10, 11, 12, 13]. Such systems often either provide active assistance in creating an output (such as in Liapis et al. [14]'s *Sentient Sketchbook*, where a user-drawn map is used to generate a level) or in scoring and evaluating some user-defined artifact (as in Migkotzidis and Liapis [15]'s *SuSketch*, which provides predictive evaluation of a level design). Our system falls into the latter category, and lands somewhere between a creative tool and a data analytics platform. By allowing a designer to specify what types of emergent outcomes they desire, we afford a designer the ability to determine if their current game system is capable of the experiences they desire. If it is (or is not), a designer can then use specific playtraces to understand what parameters, tuning, and gameplay actions were critical to producing the playtrace they observed.

2.2. Parameter Tuning in Games and PCG

Video games are inherently parameterized systems. How high a character jumps or when an AI should perform an action (among many, many other scenarios) are all determined by a designer's decision on how the corresponding mechanic and parameter is tuned. As such, both static and procedurally generated content (PCG) are dependent on how a system is parameterized and tuned for a given interaction. The pursuit of automated parameter tuning and parameter tuning analysis is thus well researched and prominent in academic literature. Summerville et al. [16] identifies parameter tuning as one of the greatest unsolved challenges in PCG, and good approaches to parameter tuning provide the opportunity to greatly improve artifact quality. There are many approaches to performing automated parameter tuning — an algorithmic example is the usage of evolutionary search algorithms, which score and iterate a swath of parameters until some desired fitness is reached [17, 18, 19, 20]. An analytics approach is shown in Withington and Tokarchuk [21]'s approach to identifying *which* parameters and

[1] https://github.com/smshields/ArcSearch

metrics might be most valuable when performing Expressive Range Analysis [22]. The importance of understanding *which* parameters and *how* a designer should tune them are thus critical facets of an iterative game design loop, especially when it comes to systematic tuning for emergent or experiential game properties such as game balance. This identification and modification of parameters in turn depends on the quality and observability of system metrics that tell the designer how a game progressed — metrics which are commonly reported through playtrace data.

2.3. Playtrace Analysis

Playtrace collection and analysis has been a critical area of game design and research for well over a decade. Drachen and Canossa [23] make the case that gameplay metrics provide critical information about player behavior, and that any action a player can take can potentially be measured. Wallner [24] discuss the importance of large playtrace and player metric corpora, especially when it comes to the production of visualization tooling such as heatmaps. Such large sets of playtraces, however, are inherently difficult to parse and nigh impossible to investigate on a case-by-case basis [25]. This does not mean that investigating an individual playtrace is unimportant — the study of individual games between players has historically been critical to the study of games such as *Chess* [26] and *Go* [27] and is a common attribute of eSports (such as in *Starcraft* [28]) and professional gaming commentary [29, 30]. The ability to both understand trends at large in a set of playtrace data and identify critical examples to investigate provides a solid foundation for thorough analysis of playtesting data.

2.4. Player Modeling

Player modeling refers to the practice of making a system-defined model of how a player might behave, feel, or think before, during, and after gameplay [6]. Player modeling is key to understanding how a system should respond to the input provided by a given audience, and is a common practice during game design [31] and the construction of procedural systems [32]. Modeling is particularly relevant in adaptive game systems such as dynamic difficulty adjustment (DDA), where modeling perceived difficulty allows a designer to appropriately modify a game's properties to meet a player on their skill level [33]. This is exemplified by Valve's *Left 4 Dead* [34] AI director (AID), which uses a calculated metric for "emotional intensity" as a means to define gameplay curves. In pursuit of roughly sinusoidal cycles of rising tension, climax, and falling tension, system perception of emotional intensity drives enemy spawn volume and timing [35]. It is important to note that player modeling is often a designed, composite of many gameplay metrics. In the case of *Left 4 Dead* [34], there is no raw value measurement of "emotional intensity" broadcast from the player's limbic system; it is calculated from a combination of in-game metrics. Thus, we can easily visualize and operate different dimensions of player modeling through these composite metrics that are created by a game's designer. The ability to understand if your player modeling is 1) correctly tuned to player experience and 2) allows for the correct manipulation of system interaction is in turn critical for a designer to meet their intended experiential goals. A generic version of how a modeled metric changes throughout a playtrace can be seen in Fig. 1.

3. System Description

The Playtrace Arc Search (PAS) tool (Fig. 2) aims to help designers accomplish three goals:

1. Understand overall distribution and consistency of playtrace data
2. Search large sets of playtrace data for desired system arcs
3. Inspect individual playtraces of interest, with references to their source data

PAS aims to satisfy these goals for a specific drawn playtrace within seconds, allowing for multiple searches to be performed in quick succession. It accomplishes this by parsing a large set of structured JSON data and providing a point cloud representation of user-defined properties of said JSON data. The user can enable or disable this cloud on top of a canvas, which allows the user to draw a desired

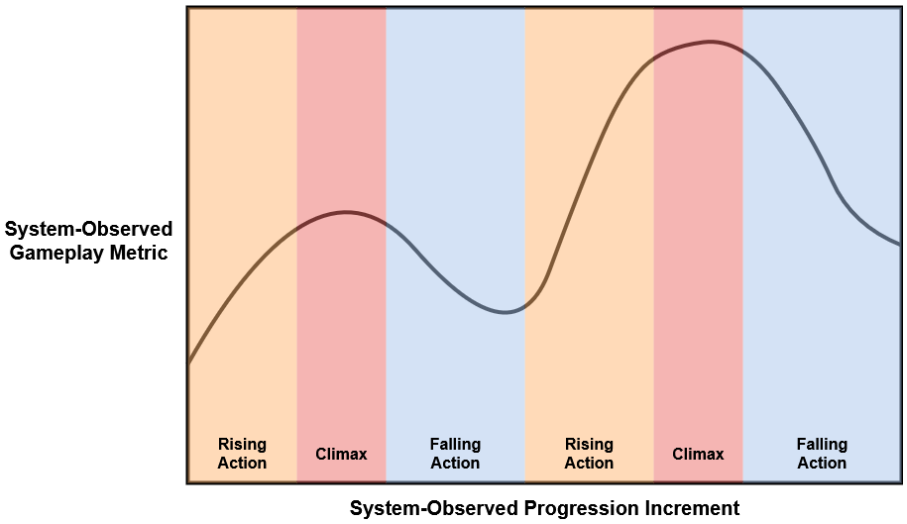

Figure 1: An example of how playtrace metrics are modeled through curves and used to influence designer understanding and gameplay systems. Some system-observed metric, usually derived from a player model, is meant to change over some incremental metric (e.g., time or number of games played) in order to understand, project, or modify a player's experience according to some curve. Such models have been used to great effect in Dynamic Difficulty Adjustment systems, such as the AI director in *Left 4 Dead* [35, 34]

curve that they'd like to search for within the playtrace corpus. After selecting search strategies using curve-matching approaches, a list of similarity-scored results (Fig. 3) is presented for the user to inspect. The system is web-native and uses graphing (Chart.js[2]), canvas (Fabric.js[3]), Fréchet Distance (Curve-Matcher[4]), and Dynamic Time Warping (Dynamic-Time-Warping[5]) libraries for user-input, visualization, and search strategy. The full code and demo for PAS is available online.[1]

3.1. Metric Selection, Logging, Formatting

PAS focuses on the analysis of how important gameplay metrics change throughout playtraces. These metrics are sometimes raw data (e.g., how much HP does a character have), but can also be productively calculated and composited into player-modeling metrics tied to player actions or progression. The metrics recorded at each step can be thought of as dependent variables, which PAS visualizes on the y-axis of its arc drawing. On the x-axis, a discrete progression measurement is used to map each consecutive data entry (e.g., gameplay tick). While a common progression measurement used for playtraces is time, other metrics can be used: player action frequency, level/goal completions, and so on. What is important is that these metrics are identified as meaningful by a designer and implemented into a game logging system that allows for post-hoc analysis.

PAS reads a folder from the file system containing an array of JSON files. The naming of each JSON should be a meaningful identifier corresponding with its playtrace. Examples of good identifiers include time of playtrace or the seed of the generated artifact — something displayed in the interface that will help the user quickly understand which playtrace they are looking at. The format of the JSON is shown in Listing 1. Of note is that you can have multiple discrete progression measurements and multiple dependent metrics for each progression measurement, allowing the user to switch between different playtrace inspections without reloading data. Discrete progression measurements are represented by their index in their array (i.e., discreteProgressionMeasurements[0] corresponds to the first recorded metrics in a playtrace). Dependent metrics must be numbers for visualization and analysis purposes. This data structure makes PAS game-agnostic — so long as designers define metrics for input (which is

[2]https://github.com/chartjs/Chart.js
[3]https://github.com/fabricjs/fabric.js/
[4]https://github.com/chanind/curve-matcher
[5]https://github.com/GordonLesti/dynamic-time-warping

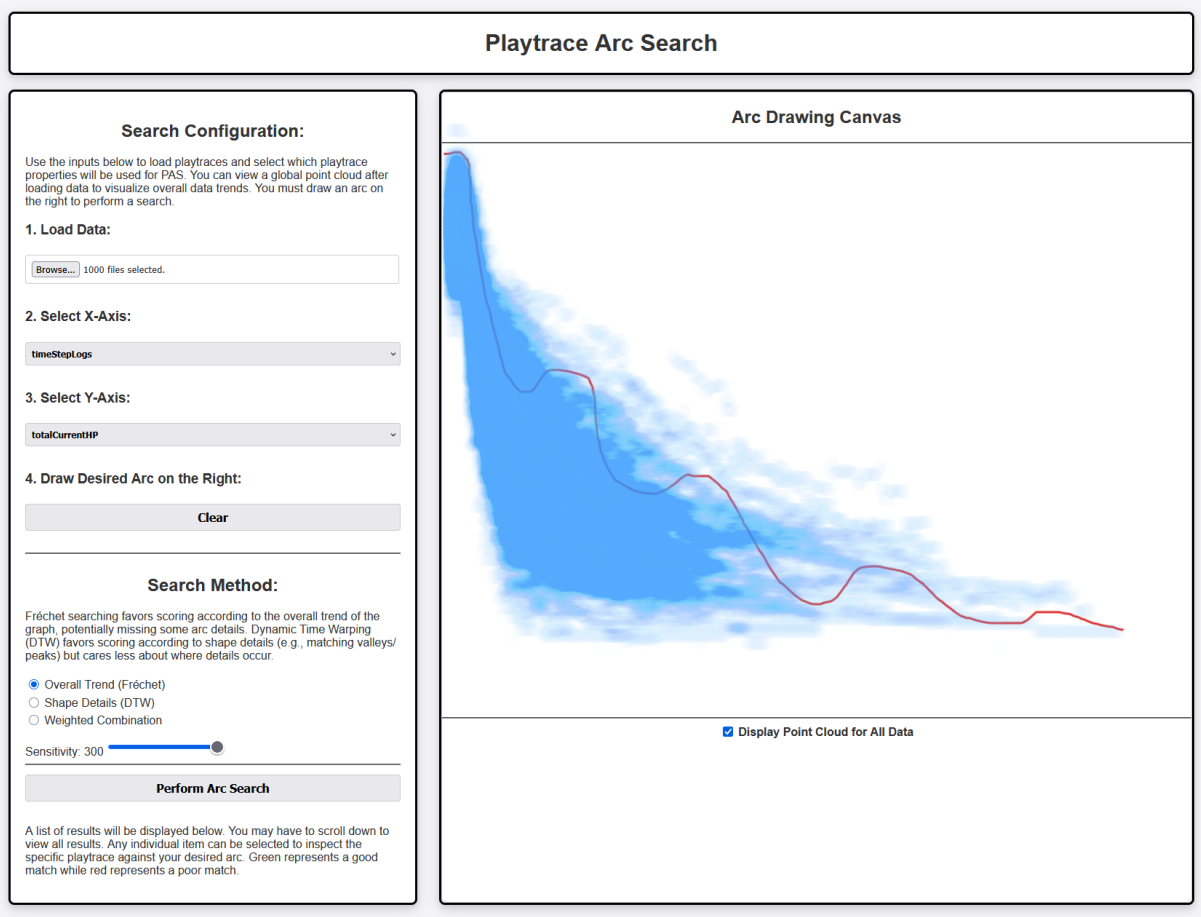

Figure 2: A screenshot of the Playtrace Arc Search (PAS) tool. The left panel allows a user to upload data, select the values to inspect, and the curve-matching strategy to use when evaluating playtraces. The right panel provides an (optional) point cloud graph of all playtraces, as well as a canvas for a designer to draw the curve (in red) that they are searching for in their data.

likely done as a byproduct of development anyways) PAS can be applied to a dataset.

Once data is correctly formatted, it can be loaded into the system using the PAS interface, and the x-axis and y-axis can be selected using dropdowns. After the playtraces are loaded and metrics are specified, a user can select a toggle to display a point cloud representation of the playtraces they have loaded.

Listing 1: An example input JSON schema to be used with PAS. Each object inside of "discreteProgressionMeasurements" should have the same structure. Only numbers can be used for playtrace analysis.

```
{
        //System accepts any number of potential x-value measurements
        "discreteProgressionMeasurements" : [
                //Each index in array corresponds with x-value on graph
                {
                        //Each value in a discrete object can be used as a y-value
                        //Each value will be present in PAS interface
                        "metric1": number,
                        "metric2": number,
                        "metric3": number,
                        ...
                },
                ...
        ],
```

```
15          ...
16  }
```

3.2. Search Strategy and Output

On the right-hand side of the tool, a canvas exists for a user to draw an arc on. This arc can be any shape, but must pass the vertical line test (no duplicate y values for the same x value), and must only consist of one continuous segment. The user can use the generated point cloud as a guide if desired. The line does not need to reach both ends of the canvas, as the line drawn will be transformed to fit within the vertical and horizontal bounds of any given playtrace to which it is being compared. With data loaded and the desired arc drawn, a user can select what search strategy they'd like to use: Fréchet Distance, Dynamic Time Warping (DTW), or a weighted combination of both. The Fréchet Distance strategy favors matching based on overall curve trend on a point-by-point basis [36], while DTW cares more about the shape and translated phase of curves and is more forgiving of translations of those shapes (e.g., rising/falling trends) [37]. Both are useful for different use cases — Fréchet Distance might help confirm an overall trend across all playtraces (all metrics are rising/falling in similar places) while DTW can help confirm that all playtraces experienced a similar pattern regardless of progression index that they occurred at. A weighted combination of both approaches is also available, allowing for a nuanced mixture of both search strategies. Each strategy scores every playtrace in similarity against the user-drawn curve from 0 to 1. Due to the transformation of the user-drawn curve to match the relative ranges of playtraces, a resampling is performed for both algorithms to standardize scoring. The density of this resampling can be specified using a sensitivity slider below the search method options. The search can be initiated by clicking the button below the search method options.

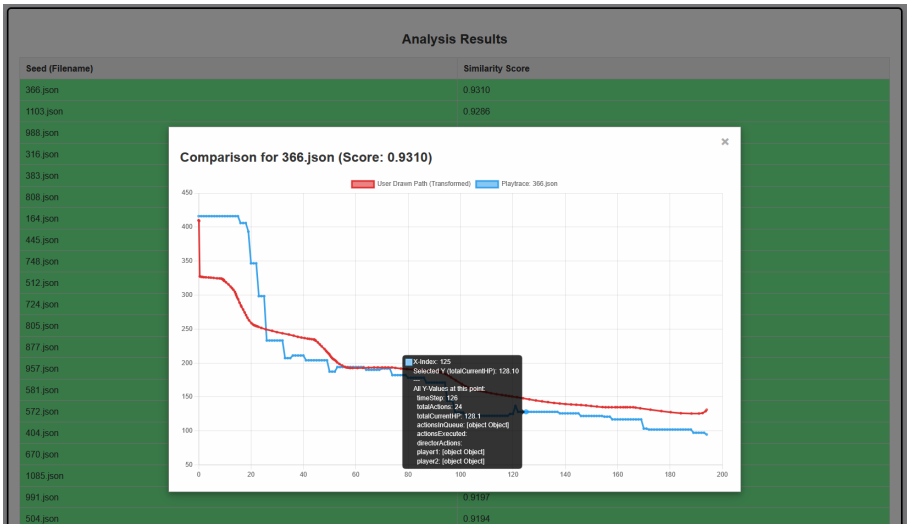

Figure 3: A view of the analysis results section with a playtrace inspector open. The similarity score shows how similar the drawn graph is to a given playtrace. Hovering over the playtrace produces all values provided within the input JSON for a given x value.

After performing the search, the user is presented with a ranked list of playtraces, sorted by their similarity to the drawn score (see Fig. 3). Each playtrace is labeled by its filename, and its score is shown in its row. Playtraces are colored on a gradient from green to red, with high scores being green and low scores being red. Clicking on an individual playtrace entry shows a graph that overlays the user-drawn arc with the playtrace, allowing for direct comparison in context. The user can hover over the playtrace, investigating all metrics included in the JSON file for the given discrete progression step.

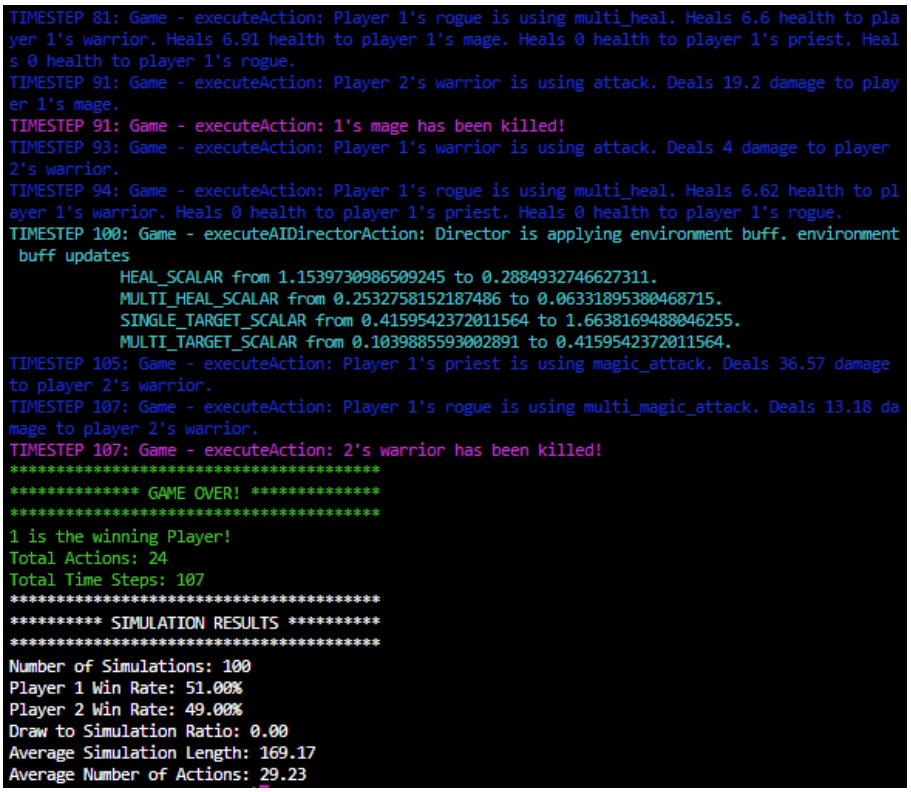

Figure 4: Console output from the automated playtesting platform used for evaluation, FighterDDA. This output highlights characters attacking and defeating one another while providing a basic metrics summary. Logging from each simulated fight is saved in JSON format for analysis by PAS.

4. Evaluation

To evaluate the efficacy of PAS, we ran an evaluation that sought to confirm three critical design use cases:

1. Can PAS correctly positively score a drawn arc corresponding with the point cloud?
2. Can PAS correctly score a drawn arc that does not follow the point cloud?
3. Can PAS correctly identify a graph with certain features from a large set?

To do this, we used a game testbed with automated playtester functionality to generate 1,000 playtraces [38]. From these playtraces, we generated a point cloud, drew arcs to match each condition, selected appropriate search strategies, and reviewed the results.

4.1. Methodology

We used a headless Turn-Based Role-Playing Game simulator named *FighterDDA* for the production of a 1,000 playtrace corpus of data [38]. We selected this system because of its usage of a popular game genre and ability to rapidly generate a large set of playtraces for analysis; collecting a similar volume of human-playtesting would require much more time and is unnecessary to confirm basic system operation. This system uses utility agents combined with an AI Director to simulate two teams of four characters fighting one another, with the game ending when all characters from a given player have no remaining health points. Characters are capable of attacking opponents, defending themselves, and healing teammates. The AI director applies environment changes in an attempt to modulate difficulty throughout gameplay. For the purposes of this evaluation, we chose to use the metric of the health of all characters summed together across each game tick. This metric provides us a view into the general pacing of the game and could help a designer understand if games are ending at appropriate times and

that games have some form of back-and-forth style of gameplay. A screenshot of the console output of a run of the data generation testbed is shown in Fig. 4.

To test the three cases listed above, we uploaded our corpus of playtraces to the system, selecting game ticks ("timeStepLogs") and total health ("totalCurrentHP") as our x-axis and y-axis, respectively. For case 1, we drew an arc that follows the densest areas of the point cloud, and ran analysis using the Fréchet strategy, as we are looking to see if we get matches for the overall trend of our drawn arc. For case 2, we drew an arc that inverts the densest areas of the point cloud, running the same Fréchet search strategy as in case 1. For case 3, we drew a curve that oscillated while matching the overall trend of the point curve, searching for playtraces that held some pattern of rising and falling current health throughout the game. We used a combined approach (both weights set at 0.5) for this case, as we both wanted the overall trend of the curve to be matched as well as the detection of specific curve features. We set system sensitivity at 300 for all cases. All drawn curves can be seen in Fig. 5.

4.2. Results

PAS worked as intended in all three cases listed in section 1. For use case 1 (Q1), we found high scoring of curves with near universal consistency, with scores ranging from 0.7367 to 0.9354. This shows that PAS is capable of matching the overall curve trends that are common within a large set of playtraces. For use case 2 (Q2), we found extremely low scoring of curves with universal consistency, with scores ranging from 0.0000 to 0.3542. This result indicates that PAS successfully is able to identify curves which are highly unlikely to appear within a data set. For use case 3 (Q3), we were able to identify a curve with both overall trend and prominent features close to our drawn curve, as well as see curves that matched poorly, with scores ranging from 0.6247 to 0.8603. This feature matching demonstrates PAS's ability to identify playtraces with both overall trend and specific features in a playtrace, helping identify exemplary playthroughs of a game. A visual showing the drawn arc and the highest/lowest scoring playtrace for each case is shown in Fig. 5.

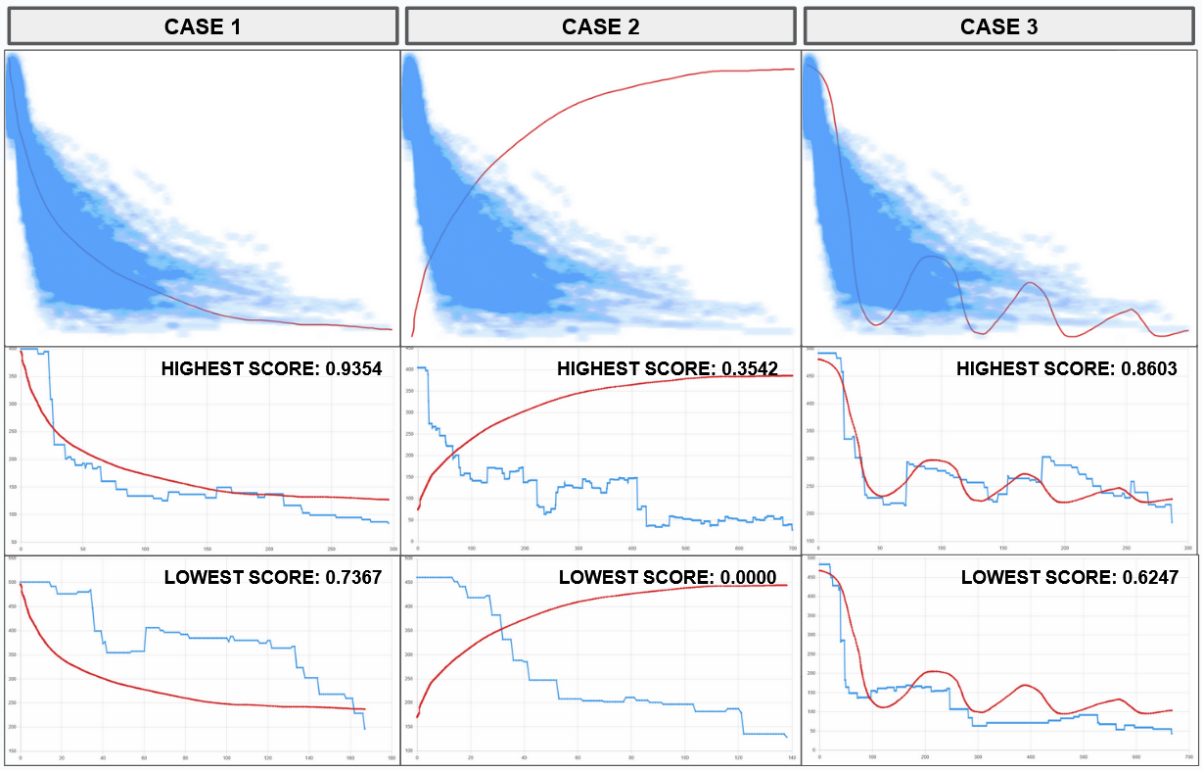

Figure 5: Results from the three case studies. Each column shows the designer's drawn graph, the highest matching playtrace, and the lowest matching playtrace. Cases 1 and 2 used Fréchet distance matching (Q1, Q2), while Case 3 used an evenly weighted combination of Fréchet distance and DTW matching strategies (Q3).

5. Discussion

5.1. Mapping Game States to Curves

PAS highlights a common facet of evaluating the quality of game systems and emergent experience — parameters and metrics of the system form curves over a progression, and these curves represent how situations and gameplay unfold. These curves can be viewed as game-system metaphors for dramatic arcs — the rising and falling of these curves over time can help a designer identify the "rising tension" or other systematic drama over a gameplay session [39].

It's important to note that progression does not just happen on a moment-to-moment, temporal basis. Such approaches might be unnecessarily fine-grained for understanding a game environment. A designer for a game that contains a large volume of small levels, such as a mobile puzzle game, might be more interested in metrics for each level completed over hundreds of levels instead of the minutiae of each individual level itself. The ability for PAS to scale based on the grain of the discrete increment provided allows the tool to be used across a variety of design use cases throughout the game lifecycle — you might investigate the moment-to-moment difficulty of a section of a game, or view how a player changes in skill level over thousands of multiplayer matches.

The ability to see a collection of curves and find important curves also highlights versatility in seeing the macro- and micro-trends of playtrace datasets. In our evaluation, we quickly saw that all games were ending within a similar curve space. This is a useful insight for the platform designer, as it shows that all games are ending appropriately and have a contained space (i.e., an important goal for this style of battle is that battles eventually end and have some level of consistency in how they play out). Meanwhile, it was reassuring to know that the individual character of curves with a tug-of-war pattern (health oscillating over time) was present, meaning that the games simulated were able to provide experiences that were not overly linear.

5.2. Confirming Designer Goals

Designers often use an "experience goal" (i.e., what experience should the player have) to drive their game development process [40]. These experience goals are impossible to know explicitly (as there is no direct access to player experience) and instead must be expressed through qualitative research or system-observable metrics defined through designer-specified player models. Understanding if a game metric is accurately predicting a player experience (and, in some implementations, impacting game state) correctly is key to meeting these player experience goals. Combining PAS analysis with qualitative data can quickly shed light on whether or not the correct system-metrics have been defined for an audience. For example, a designer might look at PAS and find confirmation for a system curve, but find that players do not qualitatively agree with the data. In such a case, the designer might need to go back to the drawing board to find a better system-derived method of quantifying player experience.

When it comes to PCG, Automatic Game Design, or games aimed at having high replayability, it may not be a positive thing to see a consistent set of playtraces. On the contrary, the designer of such systems or games might actually be looking for system arc diversity, hoping to see a wide range of curves across playtrace sessions [41]. When combined with an Expressive Range Analysis approach (such as in Shields et al. [42]'s tooling on FighterDDA), understanding the potential design space of system playtraces represents a novel and meaningful strategy to understanding system quality. PAS allows the visualization of such diversity at a glance with its point cloud, but also allows the designer to see if varied individual curves exist in the gameplay and what specific actions occurred during that playtrace to produce the curve in question. Consistency and diversity can be valid goals for different designers in different use cases, and PAS allows for the confirmation of either goal.

5.3. Tuning Through Parameter "Sweeping"

PAS is not only useful for looking at a static configuration of game systems — it can also be used for parameter tuning. This can be done by implementing an A/B testing strategy, where different parameter

tunings are provided for each playtesting session [3]. With this collection of data, a designer can then specify the output curve they were looking for, identify the playtrace(s) with the highest similarity, and use the parameter settings from those playtraces for future tuning refinement. The scoring from this system could even be implemented as part of a fitness function of an evolutionary loop, selecting games based on the quality of their system arc in addition to other metrics.

6. Limitations and Future Work

We believe that PAS has the potential to be a valuable tool for designers to rapidly iterate and evaluate their games based on systematic playtrace curves that might be difficult to parse or observe without it. That being said, there are clear limitations that could be addressed in future work. For one, a trial with game designers across a varied selection of games would greatly bolster the tool's real-world usefulness and ensure that it has the right usability and visualization approaches to aid in game design. Our evaluation confirms basic system functionality, but would benefit from being applied to a wider range of playtrace corpora, especially human-generated playtraces. Applying this form of evaluation to more complicated and a wider diversity of system measurements would also be useful. For example, it would be interesting to see how PAS could be applied directly to narrative spaces, to see if it could directly capture the dramatic arcs that it uses as a metaphor for system analysis. Integrating the arc-matching process into generative search strategies (i.e., into fitness functions) also represents an interesting design potential in looking for system quality when generating levels or games. Implementing a database adapter (so that data can be queried from a database rather than saving JSON files to disk) would also reduce requirements to improve/alter existing data logging systems in games and improve PAS compatibility overall. Finally, common arc patterns (such as the narrative arcs described in Reagan et al. [43]) could be provided to users as an alternative to drawing, effectively building on the work of Leong et al. [44]'s approach to mapping storysifting to Reagan et al. [43]'s definitions.

7. Conclusion

Playtrace Arc Search is a tool that allows designers to evaluate a large corpus of playtraces through a visualization tool and search for specific systematic "arcs" within their dataset. It provides visuals for the entire dataset as well as inspection of individual playtraces and data points alongside a scoring system that shows the similarity of a given dataset to a user-drawn arc. The tool contains multiple search strategies, and proved successful on searching through a corpus to find examples of relevant curves as drawn by a user. The combination of generic input, ability to rapidly iterate through arc searches and strategies, and range of visualization output indicates that PAS can be a useful tool for designers who wish to understand the quality of their game systems over time through playtesting data.

Acknowledgments

Special thanks to Robert Zubek, Jasmine Otto, and Oliver Withington for their feedback and advice on the tools described in this paper.

This material is also based upon work supported by the National Science Foundation under Grant No. 2202521. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Declaration on Generative AI

The author(s) used ChatGPT 4.5 and 5 in order to: Grammar and spell check, paraphrase and reword. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] J. Osborn, B. Samuel, J. McCoy, M. Mateas, Evaluating play trace (dis) similarity metrics, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, volume 10, 2014, pp. 139–145.

[2] G. Andrade, G. Ramalho, A. Gomes, V. Corruble, Dynamic game balancing: An evaluation of user satisfaction, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, volume 2, 2006, pp. 3–8.

[3] S. Hyrynsalmi, E. Klotins, M. Unterkalmsteiner, T. Gorschek, N. Tripathi, L. B. Pompermaier, R. Prikladnicki, What is a minimum viable (video) game? towards a research agenda, in: Conference on e-Business, e-Services and e-Society, Springer, 2018, pp. 217–231.

[4] P. Xenopoulos, J. Rulff, C. Silva, Ggviz: Accelerating large-scale esports game analysis, Proceedings of the ACM on Human-Computer Interaction 6 (2022) 1–22.

[5] A. Drachen, A. Canossa, Analyzing spatial user behavior in computer games using geographic information systems, in: Proceedings of the 13th international MindTrek conference: Everyday life in the ubiquitous era, 2009, pp. 182–189.

[6] G. N. Yannakakis, J. Togelius, Player modeling, in: Artificial Intelligence and Games, Springer, 2025, pp. 315–335.

[7] N. Liao, M. Guzdial, M. Riedl, Deep convolutional player modeling on log and level data, in: Proceedings of the 12th International Conference on the Foundations of Digital Games, 2017, pp. 1–4.

[8] J. Jiang, D. Maldeniya, K. Lerman, E. Ferrara, The wide, the deep, and the maverick: Types of players in team-based online games, Proceedings of the ACM on Human-Computer Interaction 5 (2021) 1–26.

[9] A. Zook, E. Fruchter, M. O. Riedl, Automatic playtesting for game parameter tuning via active learning, CoRR abs/1908.01417 (2019). URL: http://arxiv.org/abs/1908.01417. arXiv:1908.01417.

[10] A. Liapis, G. Smith, N. Shaker, Mixed-initiative content creation, Procedural content generation in games (2016) 195–214.

[11] S. Margarido, P. Machado, L. Roque, P. Martins, Boosting mixed-initiative co-creativity in game design: A tutorial, ACM Computing Surveys (2024).

[12] G. Lai, F. F. Leymarie, W. Latham, On mixed-initiative content creation for video games, IEEE Transactions on Games 14 (2022) 543–557.

[13] M. Kreminski, I. Karth, M. Mateas, N. Wardrip-Fruin, Evaluating mixed-initiative creative interfaces via expressive range coverage analysis., in: IUI Workshops, 2022, pp. 34–45.

[14] A. Liapis, G. N. Yannakakis, J. Togelius, Sentient sketchbook: computer-assisted game level authoring (2013).

[15] P. Migkotzidis, A. Liapis, Susketch: Surrogate models of gameplay as a design assistant, IEEE Transactions on Games 14 (2021) 273–283.

[16] A. Summerville, S. Snodgrass, M. Guzdial, C. Holmgård, A. K. Hoover, A. Isaksen, A. Nealen, J. Togelius, Procedural content generation via machine learning (pcgml), IEEE Transactions on Games 10 (2018) 257–270.

[17] S. Shields, R. Mawhorter, E. Melcer, M. Mateas, Searching for balanced 2d brawler games: successes and failures of automated evaluation, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, volume 18, 2022, pp. 189–198.

[18] M. Morosan, R. Poli, Automated game balancing in ms pacman and starcraft using evolutionary algorithms, in: Applications of Evolutionary Computation: 20th European Conference, EvoApplications 2017, Amsterdam, The Netherlands, April 19-21, 2017, Proceedings, Part I 20, Springer, 2017, pp. 377–392.

[19] R. Leigh, J. Schonfeld, S. J. Louis, Using coevolution to understand and validate game balance in continuous games, in: Proceedings of the 10th annual conference on Genetic and evolutionary computation, 2008, pp. 1563–1570.

[20] F. Rupp, K. Eckert, Geevo: Game economy generation and balancing with evolutionary algorithms,

arXiv preprint arXiv:2404.18574 (2024).
[21] O. Withington, L. Tokarchuk, The Right Variety: Improving Expressive Range Analysis with Metric Selection Methods, in: Proceedings of the 18th International Conference on the Foundations of Digital Games, ACM, Lisbon Portugal, 2023, pp. 1–11. URL: https://dl.acm.org/doi/10.1145/3582437.3582453. doi:10.1145/3582437.3582453.
[22] G. Smith, J. Whitehead, Analyzing the expressive range of a level generator, in: Proceedings of the 2010 Workshop on Procedural Content Generation in Games - PCGames '10, ACM Press, Monterey, California, 2010, pp. 1–7. URL: http://portal.acm.org/citation.cfm?doid=1814256.1814260. doi:10.1145/1814256.1814260.
[23] A. Drachen, A. Canossa, Towards gameplay analysis via gameplay metrics, in: Proceedings of the 13th international MindTrek conference: Everyday life in the ubiquitous era, 2009, pp. 202–209.
[24] G. Wallner, Play-graph: A methodology and visualization approach for the analysis of gameplay data, in: 8th International conference on the Foundations of digital games (FDG2013), Foundations of Digital Games, 2013, pp. 253–260.
[25] Y.-E. Liu, E. Andersen, R. Snider, S. Cooper, Z. Popović, Feature-based projections for effective playtrace analysis, in: Proceedings of the 6th international conference on foundations of digital games, 2011, pp. 69–76.
[26] P. W. Frey, P. Adesman, Recall memory for visually presented chess positions, Memory & Cognition 4 (1976) 541–547.
[27] J. S. Reitman, Skilled perception in go: Deducing memory structures from inter-response times, Cognitive psychology 8 (1976) 336–356.
[28] Blizzard Entertainment, Starcraft 2, [DIGITAL], 2010.
[29] A. Białecki, N. Jakubowska, P. Dobrowolski, P. Białecki, L. Krupiński, A. Szczap, R. Białecki, J. Gajewski, Sc2egset: Starcraft ii esport replay and game-state dataset, Scientific Data 10 (2023) 600.
[30] Z. Lin, J. Gehring, V. Khalidov, G. Synnaeve, Stardata: A starcraft ai research dataset, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, volume 13, 2017, pp. 50–56.
[31] A. Drachen, M. Seif El-Nasr, A. Canossa, Game analytics–the basics, in: Game analytics: Maximizing the value of player data, Springer, 2013, pp. 13–40.
[32] C. Pedersen, J. Togelius, G. N. Yannakakis, Modeling player experience for content creation, IEEE Transactions on Computational Intelligence and AI in Games 2 (2010) 54–67.
[33] A. Baldwin, D. Johnson, P. Wyeth, P. Sweetser, A framework of dynamic difficulty adjustment in competitive multiplayer video games, in: 2013 IEEE international games innovation conference (IGIC), IEEE, 2013, pp. 16–19.
[34] Valve South, Left 4 dead, [Windows, Xbox 360, macOS], 2008.
[35] M. Booth, The ai systems of left 4 dead, in: Artificial Intelligence and Interactive Digital Entertainment Conference at Stanford, 2009, 2009.
[36] H. Alt, M. Godau, Computing the fréchet distance between two polygonal curves, International Journal of Computational Geometry & Applications 5 (1995) 75–91.
[37] E. Keogh, C. A. Ratanamahatana, Exact indexing of dynamic time warping, Knowledge and information systems 7 (2005) 358–386.
[38] S. Shields, E. F. Melcer, FighterDDA: A simulation testbed for evaluating director-based dynamic balancing, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE), AAAI Press, 2025. In press.
[39] M. Mateas, A. Stern, Structuring content in the façade interactive drama architecture, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, volume 1, 2005, pp. 93–98.
[40] T. Fullerton, C. Swain, S. Hoffman, Game design workshop: Designing, prototyping, & playtesting games, CRC Press, 2004.
[41] T. X. Short, T. Adams, Procedural storytelling in game design, Crc Press, 2019.
[42] S. Shields, O. Withington, E. F. Melcer, Designer difficulties: Visualizing the possibility spaces

of dynamic difficulty adjustment systems, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE), AAAI Press, 2025. In press.
[43] A. J. Reagan, L. Mitchell, D. Kiley, C. M. Danforth, P. S. Dodds, The emotional arcs of stories are dominated by six basic shapes, EPJ data science 5 (2016) 1–12.
[44] W. Leong, J. Porteous, J. Thangarajah, Automated sifting of stories from simulated storyworlds., in: IJCAI, 2022, pp. 4950–4956.