Visualizing AI Playtesting Data of 2D Side-scrolling Games

Shivam Agarwal
University of Duisburg-Essen
shivam.agarwal@paluno.uni-due.de

Christian Herrmann
University of Duisburg-Essen
h.christian92@gmx.de

Günter Wallner
Eindhoven University of Technology
University of Applied Arts Vienna
g.wallner@tue.nl

Fabian Beck
University of Duisburg-Essen
fabian.beck@paluno.uni-due.de

***Abstract*—Human playtesting is a useful step in the game development process, but involves high economic costs and is time-consuming. While playtesting through artificial intelligence is gaining attention, it is challenging to analyze the collected data. We address the challenge by proposing visualizations to derive insights about level design in 2D side-scrolling games. To focus on the navigation behavior in the level design, we study the trajectories of computer agents trained using artificial intelligence. We demonstrate the effectiveness of our approach by implementing a web-based prototype and presenting the insights gained from the visualizations for the game *Sonic the Hedgehog 2*. We highlight lessons learned and directions to customize the approach for other analysis goals of playtesting.**

***Index Terms*—artificial intelligence, playtesting, visualization**

I. Introduction

Playtesting—the process of exposing players to a game to assess their behavior and experience with it [1]—has become commonplace in game development. However, playtesting with human players can be a time-consuming and costly task, which may limit its application, especially for smaller and independent studios (cf. [2], [3]). At the same time, the virtual worlds of navigation-based games are getting more complex, making it infeasible to exhaustively test them manually. There is an increased interest in the application of artificial intelligence (AI) to automate (parts) of the playtesting process. Indeed, research on using AI for the purpose of playtesting is proliferating ranging from game balancing [4], [5] to navigation [1], [2] and covering a wide variety of games, including board games (e.g., [4]), collectible card games (e.g., [5]), matching tile games (e.g., [6]) and others (e.g., [1], [3]). Moreover, AI allows to take different skills and goals of players into account to build agents with distinct behaviors (e.g., [7], [8]).

Just like with human playtesters, it is essential for game designers to understand the behavior of agents as artificial playtesters before allowing actionable decisions. Visualization can serve as an effective means to aid the analysis. While game AI research is already taking advantage of visualization (e.g., [9]–[12]), it is still less leveraged in games user research.

Complex maneuvers in 2D side-scrolling jump-and-run games make the level design and playtesting challenging for game designers. We propose a visualization approach showing the navigational behavior of agents for testing the level design. We implement our approach for a specific game (*Sonic the Hedgehog 2*) in a web-based visualization tool. Then, we discuss insights derived from the visualization and reflect on the challenges we encountered. These lessons learned can serve as inspiration for advancing this line of research.

II. Related Work

Several works have employed AI to assist in playtesting, taking benefit of agents exhibiting different gameplay behaviors. Guerrero-Romero et al. [7] discusses a methodology for automated testing based on a team of agents pursuing different goals such as winning, exploring, and collecting. Similarly, but approaching the topics rather from a games user research than AI perspective, Stahlke and Mirza-Babaei [1] also outline opportunities and challenges posed by using AI to train agents that mimic the navigational behavior of humans in evaluating level designs. Zhao et al. [3] discuss two fundamental metrics—skill and style—which human-like agents need to vary to model different behaviors. Similarly, Holmgard et al. [8] developed AI agents who behave differently based on their personas and was extended by Mugrai et al. [6] who developed procedural personas specifically for Match-3 games. Silva et al. [4], taking a more game-specific approach, used different agents to analyze which play styles are best suited for the different maps offered by the boardgame *Ticket to Ride*. Ariyurek et al. [13] used trained agents specifically to find bugs while Garcia-Sanchez et al. [5] used an evolutionary algorithm to create decks of cards in *HearthStone* to be played by an AI against human-designed decks. Chang et al. [12] proposed exploration techniques to amplify the coverage of game environments starting from human playtester data. Pfau et al. [11] explored the use of unsupervised solving to test riddles in adventure games for solvability.

These works show that using AI for playtesting have been explored from a variety of perspectives. Although some of the above mentioned works discuss the potential of visualization (e.g., [7]) or make use of data visualizations (e.g., [4], [8], [12]), the main focus is on the AI technology rather than on dedicated solutions for visualization of AI behavior. Some

notable exceptions include the work of Douglas et al. [10], who built saliency maps based on an agent's interpretation of the game environment to help convey the agent's decision making. Further, Agarwal et al. [9] proposed a timeline-based visualization of direct and indirect interactions between agents to help understand competition and collaboration strategies. However, these approaches are not targeted specifically for the purpose of playtesting. Most similar is the work of Stahlke et al. [2], who proposed a framework for configurable agent navigation behavior paired with a simple visualization of simulated data. However, unlike our approach, their visualization shows trajectories of only one agent at a time.

III. Approach

Our approach proposes visualizations to analyze level design showing navigational behavior from AI playtesting data. We focus our approach on a specific game and implement the proposed visualizations in a web-based prototype.

A. Use Case: Sonic the Hedgehog 2

To demonstrate visualizing AI playtesting data, we choose a 2D side-scrolling platforming game—*Sonic the Hedgehog 2* [14]. Sonic is the in-game character controlled by a human player. The goal of each game level is to finish it by making Sonic run from left to right while avoiding obstacles such as enemies or traps (which might kill Sonic). The action space in the game is simple: At any point, Sonic can either run (left or right), roll, duck down, jump, or perform a spin dash move (roll with high speed). Most of the enemies can be killed by jumping on or rolling through them. While advancing through the level, Sonic can collect different power-ups. Most notable are the rings that accumulate and protect Sonic from dying. If Sonic has no rings and gets hit by an enemy, it dies, loses a life, and has to start the level again.

B. AI Training Approach

We use the NeuroEvolution of Augmenting Topologies (NEAT) algorithm [15] in a Python implementation [16] for training agents to play the game. The algorithm implements a genetic approach to deep reinforcement learning where the structure of the neural network evolves over several generations. Each generation consists of 30 agents that play the game. The best-performing agents from a generation are selected for crossovers to make the next generation. A reward function determines the performance of each agent. Since the goal of the game is to reach the far right end of the level, we use a simple reward function based on the Euclidean distance between final position of the Sonic and the goal position.

We use the Open AI Gym environment [17] for the agents to play the game. The position of each AI agent at every frame during the game is recorded along with the received reward value for each game level. We also include the generation number of each AI agent in the data. The recorded data is then used to construct visualizations, as explained next.

C. Visualization

The specific goal of the visualization is to enable visual analysis of the navigation behavior of agents and to assess the difficulty of obstacles. To this end, assuming that different levels of training can reflect different levels of expertise of humans, we compared the behavior of agents from different training iterations (generations). Aside a plot of the overall agents' performance across the training process, we visualize the individual trajectories of the agents within the level for selected generations. An example of the visualizations is shown in Figure 1. The line chart at the top encodes the minimum (—), average (—), and maximum (—) reward values for the population of agents in each generation along the training process. Two maps of the selected level [18] are shown below the line chart. For each map, there is a dropdown menu to select a specific generation of agents. Showing many individual trajectories of agents would create clutter due to overlap and make it difficult to derive meaningful insights. Hence, taking inspiration from an existing approach [19], we calculate and overlay aggregated trajectories on top of the map, and extend the visualization by showing the rewards earned by agents across generations. Characteristic points are calculated by analyzing all the trajectories of agents in the selected generation and shown as white circles. The width of a black line between two circles denotes the travel frequency of agents between the two points in either direction.

IV. Discussion

Our approach is a step towards visualizing AI-based playtesting data of 2D side-scrolling games. In the following, we discuss what kinds of insights can already be found as well as what are lessons learned and directions for future work.

A. Insights

Figure 1 shows the aggregated trajectories for agents of generation 29 (Figure 1b) and 56 (Figure 1c). By examining generation 29, we see that the agents were able to overcome initial obstacles and travel far in the level. The thick black line near the first waterfall (Figure 1b1) indicates there was a lot of movement in this region. On a closer look, we see that the waterfall has some enemies and the preceding black lines are thinner than the one placed above the waterfall. Knowing the game mechanics, we can infer that near the waterfall, some agents of generation 29 got hit by the enemy in the waterfall and got knocked back, indicating that even some intermediate-level players might find the waterfall obstacle difficult (maybe due to the presence of enemies in the waterfall). The agents who learned to overcome this obstacle did not face trouble in crossing the second waterfall (Figure 1b2), suggesting that executing the maneuver to cross similar obstacles is not difficult to reproduce.

The line chart at the top (Figure 1a) shows that at least some agents learned to gain a maximum reward (complete the level) as early as in generation 3. It indicates that the level design is easy to navigate. Given that Emerald Hill Zone is

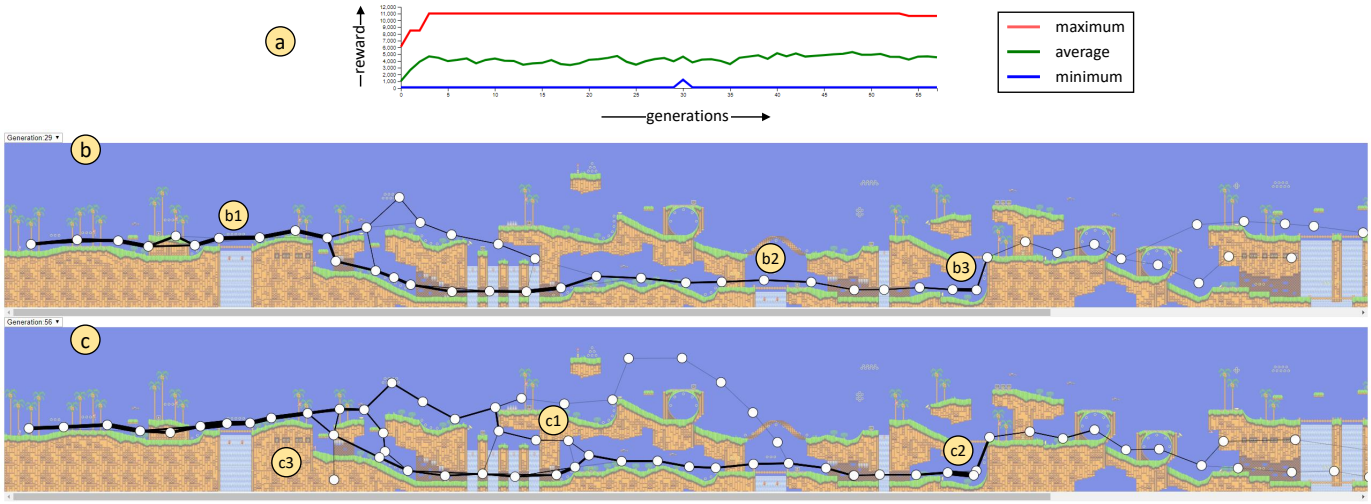

Fig. 1. Visualizations showing (a) a line chart of reward values received by agents across training process (generations), the aggregated trajectories of agents from (b) generation 29, and (c) generation 56 in *Emarald Hill Zone* level of the game *Sonic the Hedgehog 2*.

the first level this is likely also how it was envisioned in order to introduce players to the game.

In contrast to generation 29, some agents of generation 56 started to take the upper route of the level (Figure 1c). More agents in generation 56 discovered the secret path in the level (Figure 1c1). These observations indicate that secret paths in the level design can be discovered by more advanced players. Although few agents in generation 29 learned to perform the spin dash maneuver and cross the steep vertical wall (Figure 1b3), still many agents in generation 56 were not able to get past the obstacle, as shown by the thick black line before the wall (Figure 1c2). This indicates that learning to apply the spin dash maneuver in this situation may be difficult.

In contrast, the agents were not able to complete the *Aquatic Ruin Zone* level. The line chart in Figure 2 (top) shows that no agent in any generation earned the maximum possible reward implying that none of them was able to complete the level. This indicates the higher difficulty of the *Aquatic Ruin Zone* level. Additionally, although the average reward received per generation is slightly increasing, the maximum reward earned has no steady pattern. Even in generation 11 (Figure 2), only a few agents were able to travel farther as many of them died overcoming the obstacles or facing enemies (downward black lines indicate death). Similarly, Figure 3 shows the aggregated trajectory of agents from generations 11 and 53 for the level *Hill Top Zone*. We recognize a thick black line extending into the wide gap at the beginning of the level. Most of the agents, even in later generations, were not able to jump across the wide gap, suggesting that the obstacle at the beginning of the level might be more difficult to pass.

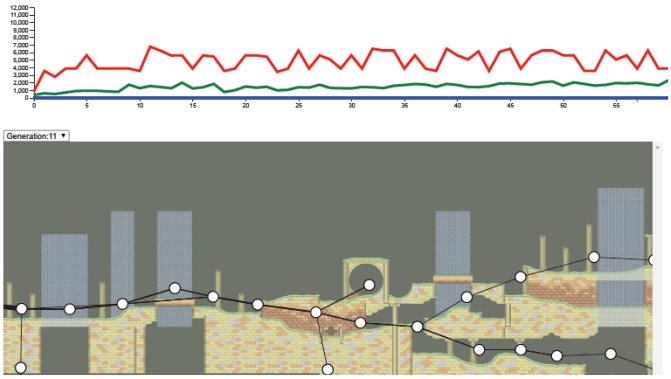

Fig. 2. The line chart (top) and aggregated trajectory visualizations (bottom) of agents from generation 11 in *Aquatic Ruin Zone* level of the game.

B. Lessons Learned

Insufficient Training or Hard Level? One of the challenges in using AI for playtesting is how to interpret the failures of agents. For instance, in Figure 3, we can see that many agents in later generations were not able to cross the wide gap. We learned that comparing agents across generations enables us to answer the question partially. For instance, selecting the generations with a peak in the line chart shows the aggregated trajectory of agents in the selected generation. The aggregated trajectory reveals whether the agents were able to overcome obstacles and travel farther in the level, or not.

Tailoring Reward Functions. Reward functions drive the behavior of agents. In our approach, we wanted to analyze the design of obstacles and hence used a simple reward function based on how far the agents traveled towards the goal position. Hence, the agents learned to move farther. However, as shown in Figure 1c3, we observe that the agents did not collect power-ups for which they had to travel backward. As such, the reward functions need to be tailored while training agents to exhibit the intended behaviors. The intended behaviors themselves can be based on different personas, representing different play styles of human players. For instance, agents trained with a simple reward function based on Euclidean distance (the one we used) would mimic a beginner, whereas assigning higher

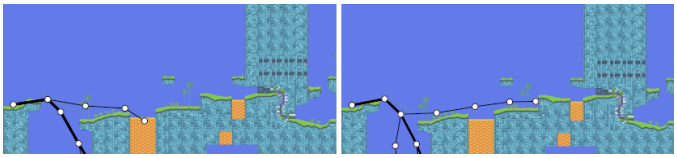

Fig. 3. Too hard or insufficient training?—Generation 11 (left) and 53 (right)

rewards for collecting power-ups (rings) could mimic play styles of expert players. The different navigational behaviors would be reflected in the trajectories, explorable through our visualization.

Improved Visualizations. We overlaid the aggregated trajectory on the map of the level. Although the technique depicts the movement of agents, it does not show other game-specific aspects. For instance, it does not show game or level-specific events occurring during the gameplay (e.g., earthquakes in the *Hill Top Zone*). The visualization needs to be customized to display such events. Additionally, calculation of characteristic points for aggregated trajectory used in the visualization can be improved. Currently, it does not take walls or other boundaries into account, which can be important for interpreting trajectories in some levels (e.g., in the *Metropolis Zone* level, which has a maze-like structure with narrow passages).

C. Future Work

In our approach, the start position for all agents was kept the same. However, to test obstacles in specific parts of the level design, it would be desirable to change the start position of the agents. As part of future work, the collection of playtesting data can be integrated with the visual interface. The integration will help game designers by giving them more control over different configurations, such as, modifying the number of agents in a generation, changing their start position, the learning rate of agents, and customizing reward functions. The visualizations can then be adapted to show the results of these configurations, e.g. by showing the segments of trajectories to focus on specific obstacles in level design, or including enemy activity to offer context for AI behavior [9]. Using different reward functions could create agents mimicking human players at different levels of expertise. Hence, differentiating between trajectories of such agents within the movement visualization could also help make comparisons between them.

V. Conclusions

We proposed an approach to visually analyze the playtesting data in 2D side-scrolling games, where artificial intelligence (AI) is used to play the game levels. We focused our approach on the game *Sonic the Hedgehog 2* and used aggregated trajectory visualization to analyze the navigation behavior of agents across training iterations. While working with the visualizations, we gained several insights about the design of three levels in the game. Although we study only one game, we envision that the approach can be generalized and extended to other 2D side-scrolling games. We shared lessons learned and discussed possible directions for visualizing the behavior of AI agents to support playtesting.

Acknowledgments

This research was partly funded by MERCUR (project: "Vergleichende Analyse dynamischer Netzwerkstrukturen im Zusammenspiel statistischer und visueller Methoden").

References

[1] S. N. Stahlke and P. Mirza-Babaei, "Usertesting without the user: Opportunities and challenges of an AI-driven approach in games user research," *Computers in Entertainment*, vol. 16, no. 2, 2018.

[2] S. . Stahlke, A. Nova, and P. Mirza-Babaei, "Artificial playfulness: A tool for automated agent-based playtesting," in *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, 2019.

[3] Y. Zhao, I. Borovikov, F. de Mesentier Silva, A. Beirami, J. Rupert, C. Somers, J. Harder, J. Kolen, J. Pinto, R. Pourabolghasem, J. Pestrak, H. Chaput, M. Sardari, L. Lin, S. Narravula, N. Aghdaie, and K. Zaman, "Winning is not everything: Enhancing game development with intelligent agents," *IEEE Transactions on Games*, vol. 12, no. 2, pp. 199–212, 2020.

[4] F. de Mesentier Silva, S. Lee, J. Togelius, and A. Nealen, "AI-based playtesting of contemporary board games," in *Proc. of the Int. Conference on the Foundations of Digital Games*, 2017.

[5] P. Garca-Snchez, A. Tonda, A. M. Mora, G. Squillero, and J. J. Merelo, "Automated playtesting in collectible card games using evolutionary algorithms," *Knowledge-Based Systems*, vol. 153, no. C, p. 133–146, 2018.

[6] L. Mugrai, F. de Mesentier Silva, C. Holmgård, and J. Togelius, "Automated playtesting of matching tile games," in *Proc. of the IEEE Conference on Games*, 2019, pp. 1–7.

[7] C. Guerrero-Romero, S. M. Lucas, and D. Perez-Liebana, "Using a team of general AI algorithms to assist game design and testing," in *Proc. of the IEEE Conference on Computational Intelligence and Games (CIG)*, 2018, pp. 1–8.

[8] C. Holmgard, M. C. Green, A. Liapis, and J. Togelius, "Automated playtesting with procedural personas through MCTS with evolved heuristics," *IEEE Transactions on Games*, vol. 11, no. 4, pp. 352–362, 2019.

[9] S. Agarwal, G. Wallner, and F. Beck, "Bombalytics: Visualization of competition and collaboration strategies of players in a bomb laying game," *Computer Graphics Forum*, vol. 39, no. 3, pp. 89–100, 2020.

[10] N. Douglas, D. Yim, B. Kartal, P. Hernandez-Leal, M. E. Taylor, and F. Maurer, "Towers of saliency: A reinforcement learning visualization using immersive environments," in *Proc. of the Int. Conference on Interactive Surfaces and Spaces*, 2019.

[11] J. Pfau, J. D. Smeddinck, and R. Malaka, "Automated game testing with icarus: Intelligent completion of adventure riddles via unsupervised solving," in *Extended Abstracts Publication of the Annual Symposium on Computer-Human Interaction in Play*, 2017, pp. 153–164.

[12] K. Chang, B. Aytemiz, and A. M. Smith, "Reveal-more: Amplifying human effort in quality assurance testing using automated exploration," in *Proc. of the IEEE Conference on Games (CoG)*, 2019, pp. 1–8.

[13] S. Ariyurek, A. Betin-Can, and E. Surer, "Enhancing the monte carlo tree search algorithm for video game testing," 2020.

[14] Sega, "Sonic the Hedgehog 2," Game [Sega Genesis], November 1992, sega, Tokyo, Japan.

[15] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[16] A. McIntyre, M. Kallada, C. G. Miguel, and C. F. da Silva, "neat-python," 2019, https://github.com/CodeReclaimers/neat-python Accessed: March, 2020.

[17] V. Pfau, A. Nichol, C. Hesse, L. Schiavo, J. Schulman, and O. Klimov, "Gym Retro," 2018, https://openai.com/blog/gym-retro/ Accessed: March, 2020.

[18] Sonic Retro, "Sonic the Hedgehog 2 (16-bit)/Maps," 2019, http://info.sonicretro.org/Sonic_the_Hedgehog_2_(16-bit)/Maps Accessed: March, 2020.

[19] G. Wallner, N. Halabi, and P. Mirza-Babaei, "Aggregated visualization of playtesting data," in *Proc. of the CHI Conference on Human Factors in Computing Systems*, 2019.