Aiding the Experts: How Artificial Intelligence Can Augment Expert Evaluation With PathOS+

By

Atiya Nowshin Nova

A Thesis Submitted to the School of Graduate and Postdoctoral Studies in partial fulfillment of the requirements for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

Faculty of Business and Information Technology
UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY
(ONTARIO TECH UNIVERSITY)
OSHAWA, ONTARIO, CANADA

July 2022

THESIS EXAMINATION INFORMATION

Submitted by: Atiya Nova

Master of Science in Computer Science

Thesis title: Aiding the Experts: How Artificial Intelligence Can Augment Expert Evaluation With PathOS+

An oral defense of this thesis took place on July 22, 2022 in front of the following examining committee:

Examining Committee:

Chair of Examining Committee: Dr. Patrick Hung

Research Supervisor: Dr. Pejman Mirza-Babaei

Examining Committee Member: Dr. Loutfouz Zaman

Thesis Examiner: Dr. Richard Pazzi

The above committee determined that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

Abstract

Within games user research (GUR) predictive methods like expert evaluation are good for getting easy insights on a game in development but may not accurately reflect the player experience. On the other hand, experimental methods like playtesting can accurately capture the player experience but are time consuming and resource intensive. AI agents have been able to mitigate the issues of playtesting, and the data generated from these agents can supplement expert evaluation. To that end we introduce PathOS+. This tool allows the simulations of agents and has features that allows users to conduct their evaluations in the same place as the game, and then export their findings. We ran a study to evaluate how PathOS+ fares as an expert evaluation tool with participants of varying levels of UR experience. The results show that it is viable to use AI to identify design problems and offer more validity to expert evaluation.

Keywords: Expert Evaluation; Playtesting; Artificial Intelligence; Games User Research; Video Games

Author's Declaration

I hereby declare that this thesis consists of original work which I have authored. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize the University of Ontario Institute of Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize the University of Ontario Institute of Technology to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my thesis will be made electronically available to the public.

The research work in this thesis that was performed in compliance with the regulations of the university's Research Ethics Board under REB Certificate NSERC Discovery RGPIN-2021-03500 and NSERC CREATE SWaGUR CREATE 479724-2016.

Atiya Nova

Statement of Contributions

Some ideas have appeared previously in the following publications:

Atiya N Nova, Stevie Cheryl Francesca Sansalone, and Pejman Mirza-Babaei. 2021. PathOS+: A New Realm in Expert Evaluation. In Extended Abstracts of the 2021 Annual Symposium on Computer-Human Interaction in Play (CHI PLAY '21). Association for Computing Machinery, New York, NY, USA, 122–127. https://doi.org/10.1145/3450337.3483495

Samantha Stahlke, Atiya Nova, and Pejman Mirza-Babaei. "Artificial Playfulness: A Tool for Automated Agent-Based Playtesting". In: Proceedings of CHI 2019 Extended Abstracts. Glasgow, Scotland, UK: ACM, 2019, LBW0176:1–LBW0176:6. doi: 10.1145/3290607.3313039

The prototype developed for this work (described in Chapter 3) and some of ideas presented here are based on the thesis by Samantha Stahlke titled "Synthesizing Play: Exploring the Use of Artificial Intelligence to Evaluate Game User Experience". A research assistant named Stevie Sansalone also worked on the project. I wrote the majority of the code for the variation presented here. A full history of all code contributions is available on the project's Github page:

https://github.com/AtiyaNova/pathosplus.

The user study was conducted remotely. All the interviews were performed by me and I was the primary contact for study participants.

I hereby certify that I am the sole author of this thesis. All ideas and prior work from others referenced in this work has been attributed appropriately following standard practices.

Parts of this thesis have been edited and adapted for submission as a paper for ACM FDG 2022. This thesis was written before the submission.

Acknowledgements

First and foremost, I'd like to thank Dr. Pejman Mirza-Babaei, my supervisor. He gave me a chance when I was still evergreen and it's thanks to him that I got to embark on so many fascinating research opportunities. I wouldn't be where I am today without his kindness and support, and for that I'll forever be grateful.

It's through these opportunities that I got to meet the next person I'd like to thank, Samantha Stahlke. I really enjoyed working with her, whether it be on AI research or being the TA for her game design course. She's a great friend and she encouraged me so much every time I showed her what I was working on. Samantha, if you're reading this, no u.

I would like to thank my fellow alumni for being so supportive during this entire process. Specifically I would like to Vivian and Sage for taking the time to check out what I was working on and making sure it didn't burst into flames. On that note, I would also like to thank Stevie Sansalone. She helped me a lot with this thesis and the papers we wrote for it, and she has been immensely helpful. I'm glad I got to meet her.

Last but certainly not least, I would like to thank my partner, Kodey Owen. He has been my rock and given me so much support through the hardest parts of my academic career. He has listened to my endless rambles with all the patience in the world, and celebrated all my successes with me. If not for him I would not be nearly as happy as I am right now.

There are many other people who have had a positive impact on me over the past couple of years, and I'm sure there are some that I'm forgetting. Regardless, if you've shown me any support or kindness throughout this journey, thank you! I'm grateful for it.

Contents

List of Tables

List of Figures

1 Introduction

1.1 Context

Games existed as a popular form of entertainment for thousands of years. One of the earliest games is Senet, first conceived in ancient Egypt [1]. Made for two players, Senet had humble beginnings. Game pieces are placed onto a board. Dice are rolled. Pieces are moved accordingly. The first player to get all their pieces off the board is declared the winner. The mechanics of this game were simple enough, yet over the years it took on religious and spiritual influence. No longer was it simply moving pieces on a board–instead it began to symbolize how a human soul progressed through the afterlife. And no longer was this a pastime reserved for nobility–commoners also were able to partake. Gradually, Senet's influence spread throughout the world.

Since then we have seen the meteoric rise of many other games. Some were from the ancient world—Mahjong, Go, Backgammon. Others were from a more contemporary age—*Tetris*, *Pong*, *Space Invaders*. While differing in mechanics, these were all able to pique human interest and garner a fanatical following. For instance, even though *Tetris* first came out in the 1980s, there are still world championships for it to this day [2]. Even a special online version of the game (called Tetris 99) recently came out on the Nintendo Switch, and became a viral hit [3].

Games also grew in sophistication. They were no longer just relegated to paper or wooden boards. They were made for arcade machines, home consoles, handheld systems, phones. These days gaming hardware is powerful enough to simulate 3D graphics nearly indistinguishable from real life. Games push the boundaries in Virtual Reality and Augmented Reality, advancing the industry of immersive technologies with major

commercial successes such as *Beat Saber* (which has an estimated revenue of over 100 million [4]) and *Pokemon Go* (which broke records and made billions in revenue [5]). The popularity of games seeps into other media too, with notable game franchises having books and movies being made. Notably Sega's *Sonic the Hedgehog* was such a box office success that not only did the sequel come out in 2022, but a third movie and a spin-off series are in the works [6].

This increase in popularity and sophistication means games are more than just an entertaining pastime. Games are a multimillion dollar industry, and amidst the pandemic made more than the sports and movie industries combined [7]. Alongside that, the majority of North American households now own gaming devices [8]. With this success comes increased pressure. Large and small gaming companies alike have great expectations placed on them to release quality games that provide an enjoyable experience.

This poses several challenges. There are many elements unique to developing games compared to other software, such as determining what makes the game "fun", which varies by genre and target audience [9]. Along with that, evaluating how humans interact with a piece of software is a uniquely difficult process. Game developers do not always have an accurate perception of how players may experience their game, as they spend so much time developing their game that they may no longer be able to view it from an unbiased perspective [10]. This issue is compounded by the complexity and scale of the game. The trends within the industry—-what works, what doesn't, what variables to account for—-are also always changing. This brings up the question, how can we analyze these games to determine what the players want? How can we define what makes a good experience? How can we determine a game's quality while it's still in development?

1.2 Human-Computer Interaction and Games User Research

Human Computer Interaction (HCI), first practiced in the 1970s, is the study of how humans interact with computers. It's all about fostering a better understanding of these interactions. By focusing on usability and accessibility this discipline has helped make various technologies easier to use. This is exemplified by the evolution of the modern computer mouse.

When computers were an emerging marvel, the primary way that users could interact with the computer's features were via keyboards. This was cumbersome for users, and in a bid to improve the ease of use the light pen was invented [11]. The light pen was a pointer that allowed users to interact with computers, and while an improvement, it had its issues as well. To make use of the light pen a user would have to hold their arm up for extended periods of time, causing strain that became known as the "gorilla arm" [12]. There was an evident need for a more streamlined, lightweight tool. This led to the creation of the first rudimentary computer mouse in the 60s by Douglas Engelbart. The tool was described in his seminal paper "Augmenting Human Intellect: A Conceptual Framework", in which he also broke down how such an idea should be researched and tested among those who use computers frequently [13]. This was the first step towards the computer mouse that we all use today, and it was achieved through the field of HCI–through the in-depth study and testing of the interactions between humans and computer interfaces.

The field of HCI research has also helped form many other disciplines, and its practices have bled into game development. This resulted in one of its offshoots, Games User Research (GUR) [14]. GUR is a multidisciplinary field that, like HCI, is influenced by computer science and psychology. It is focused on better understanding the player in order to provide an enjoyable experience. GUR methods can help researchers and designers identify and improve upon problem areas within a game. The methodolo-

gies that fall under GUR are numerous, and have their own strengths and weaknesses. These methods can be objective (e.g. observations) or subjective (e.g. focus groups), qualitative (e.g. interviews) or quantitative (e.g. questionnaires), predictive (e.g. expert evaluation) or experimental (e.g. playtesting) [15]. Depending on the needs of a study, games user researchers use mixed method approaches by combining different methods.

As a key predictive methodology, expert evaluation—which can encompass other methods, such as heuristic evaluation—is when user research experts evaluate a game in development, and use their expertise to predict what problems a player may run into. This method can be conducted early on in a game's development in order to find usability problems, and is cheap and easy to conduct [16]. Examples of potential issues that could be detected are things like navigation issues, balancing problems, and unfair difficulty. This method does not rely on actual player data, however, instead relying on the insights of the evaluator. Due to the subjectivity of this method it may not accurately reflect the player experience. It is possible for major issues to be missed entirely, while minor issues are given more importance than they should.

This is not an issue for experimental methods like playtesting. Playtesting is the process of getting players to play through a game in development. This helps researchers collect player data and see firsthand how a player might engage with a game, thereby resulting in analyses that are more reflective of the player experience [17]. Researchers would easily be able to tell, for instance, if a part of a game level causes frustrations in players, or if something is not taught to the player adequately. The problem with this method, however, is that it is costly and inaccessible. In order to run playtests researchers have to recruit players, set up testing space, obtain all the necessary hardware needed to capture and analyze data, and so on. All of this is too resource-intensive and time consuming for some studios, especially independent developers, and they subsequently are not able to partake in a very useful process.

It is possible to combine expert evaluation and playtesting methods in order to reduce the number of issues missed [18]. There are some roadblocks to this approach, however, as expert evaluation methods are quick and easy to conduct, whereas playtests are not. This

raises the question of how we can improve upon the disadvantages of expert evaluation, without giving up any of the things that make it a beneficial and accessible method.

1.3 Artificial Intelligence

Natural intelligence refers to a human's mental aptitude and rationale. Artificial Intelligence (AI), as the name suggests, is an artificial simulation of naturally occurring intelligence. Humans have always had a fascination with AI. Evidence of this is in how AI has been depicted within art over the past several decades. Literary works referring to what could be interpreted as AI have existed since the 1800s, with some scholars arguing that Mary Shelley's "Frankenstein" is one of the first works depicting AI (as Frankenstein's monster was an intelligent, artificial creation). One notable literary work depicting AI was the short story "I Have No Mouth and I Must Scream", which came out in the 1960s. In it, an AI (that was referred to as "AM") took over the world and killed all humans, except for a small handful. It then proceeded to torture these remaining humans for all eternity. This is a rather grim depiction of a worst case scenario, and it is a trope that has been seen countless times since. It is not uncommon for stories like this to come out when there have been big scientific developments, as they are meant to act as cautionary tales, warnings against pushing science too far.

Other depictions of AI have been more hopeful. In the movie called Her (2013), we see a man who, after divorcing his wife, falls in love with an AI called Samantha, and starts a romantic relationship with her. What makes this movie interesting is that Samantha gains sentience and a greater understanding of human bonds. By the end of the movie she gains her own agency, and the freewill to explore the world as she sees fit. It is an optimistic depiction of what the future of AI could be–one not filled with the dread of AI takeover, but rather one of AI companionship.

While these fictitious depictions of AI take on fantastical and oftentimes deeply personal forms, AI in real life have not quite reached these expectations. We are still nowhere close to achieving self-driving cars [19], for instance, and tales of AI making laughable

(and sometimes concerning) mistakes are commonplace [20]. There have still been impressive strides made in recent years, however. When AI started off in the 1950s it was in the form of simple chess playing algorithms [21]. By today's era AI is sophisticated enough to handle more complex tasks with varying degrees of success. Examples include the aforementioned self-driving cars [22], along with algorithms to detect faces [23], and answer moral questions [24].

One key area of investigation when it comes to AI is how to closely approximate human behaviour. The Turing Test, conceptualized by the "father of AI" Alan Turing, was created as a way to determine if an AI algorithm could think like humans [25]. While there have been many attempts at imitating human behaviour believably, no algorithm as of yet has passed the Turing Test. Regardless, AI algorithms are now able to go so far as to simulate human decision-making [26], navigational skills [27], memory functions [28], and so on.

While AI exists within games to some capacity (such as enemies and companions), up until recently game AI has been considered rudimentary compared to the broader field of AI. This is because when it comes to games the importance is not on how sophisticated the AI is, but rather how much enjoyment players can derive from their behaviour. What's really fascinating are the ways in which AI can help in the game development process. AI can be used to evaluate the playability of a level [29], sift through large amounts of data that would be too difficult for humans to look through [30], and even generate content procedurally [31]. Additionally, AI approximations of human behaviour have helped make playtesting more accessible, with the creation of independent (or autonomous) agents that can play through games in lieu of humans (thereby reducing time and resource costs) [32]. These agents could be made to mimic different player profiles, and even complete tests at accelerated rates. Evaluators can then examine these agents as they play, or watch recorded playthroughs, in order to get actionable information. This is not meant as a direct replacement of playtesting with humans, rather it can act as a supplement, and as a way of allowing smaller developers to engage with a valuable process.

Now that the benefits of AI are starting to be leveraged for playtesting purposes with positive results, the question then becomes how can we push this field further? What else can we do to truly explore the range of AI testing benefits?

1.4 Contributions

We believe the benefits of AI testers can be extended to expert evaluation in order to reduce the subjectivity of evaluations. Evaluators can use autonomous agents to generate simulated player data, and then use the data to supplement their evaluations. This means that any conclusions drawn would be a closer approximation to the actual player experience, and the likelihood of major usability and gameplay issues being missed would be lessened. The key strength of this combination is that it could take advantage of the benefits of playtesting, without losing the low cost and ease associated with expert evaluation.

Previously, I was involved in the creation of a tool called PathOS [32]. This tool runs in Unity (a commercial game engine), is open source and it allows developers to simulate playthroughs of 3D game levels with AI agents that can emulate different player behaviours. The original purpose of this tool was to examine how AI agents could help facilitate level design, primarily through 3D navigation. A study that we previously conducted has shown that while the tool was helpful in allowing designers to understand how players may interact with their level, further strides can be taken.

As the now lead researcher on this project, my aim is to adjust the tool so that it help supplement expert evaluation. This variation of the tool is called PathOS+, and it's my primary contribution. It is a testing tool specifically for the Unity engine and any projects created within it, and is primarily intended for games in action or adventure oriented genres. The goal of this variation is to improve upon the feedback given to us during our previous study, by making it more easy to use and expanding the features and abilities of the AI agents. The other goal is to make it so that it can be an effective expert evaluation tool. This means broadening the tool's analysis capabilities, while also making it so that

evaluators can digitally conduct and even export their evaluations within the tool itself.

Another contribution of this thesis is that it expands the scope of research for AI playtesting and expert evaluation. Especially for the latter, there has been very little research done on how it can be combined with AI playtesting. There has been research that suggested the combination of user testing and expert evaluation could have its benefits, but the body of work is limited. The research herein helps add to that body of research by showing how AI research tools can streamline aspects of the game development process, and even make things more accessible for independent developments. Within the realm of game development, this contribution has the potential to help enable developers to benefit from a user-centered approach.

1.5 Thesis Outline

The primary goal of this thesis is to detail the development of PathOS+, the study conducted to determine how researchers use it for expert evaluation, and the conclusions we have drawn therein. In this chapter I discussed the motivations of this research, and introduced the fields of games user research and artificial intelligence. I discuss the benefits and downsides of various GUR methods, namely expert evaluation and playtesting, and bring up how AI can be used to mitigate some of these downsides. I also introduced our previous tool, PathOS, and gave a brief overview of how it differs from PathOS+. The following chapters are thus:

- **Chapter 2:** This chapter acts as a literature review that gives an extensive overview of the expert evaluation methods and AI in game development. Specifically, when it comes to expert evaluation, I discuss its application within industry, how heuristics are used, and the caveats of expert evaluation methods. When it comes to AI, I broadly cover some use cases and examples, namely in the categories of quality assurance, data analysis, simulation based testing, and visualizations. Overall this research has helped influence the direction I took with PathOS+.

- **Chapter 3:** This chapter is divided into two parts. The first part details the

specifics of the original PathOS. This includes its design ethos and available features. This also details the original study that was conducted with level designers, and the feedback and insights gained from it. I then discuss our process of examining this feedback in order to determine next steps for the tool, and what changes I decided to go ahead and implement. The second part discusses the nuances of developing PathOS+. This includes any subsequent research I conducted, and any additional changes made to the tool and its features. Key points include the addition of further capabilities to the AI, usability and interface improvements, and an expert evaluation tab created for the sole purpose of conducting evaluations. I also detail what developments were made to get the tool ready for our next study.

- **Chapter 4:** This chapter covers the study we conducted in order to evaluate PathOS+. It goes over the scenarios I created for the purpose of this study, and gives an overview of the 9 recruited participants who participated in it. We go into detail our methodology, the nature of the test, and the overall results gleaned from the study.

- **Chapter 5:** This chapter covers our interpretation of the results detailed in Chapter 4. I discuss some of our broad findings and whether PathOS+ succeeded in its objective. I also discuss potential areas of application for PathOS+ within the games industry, and go over various limitations of the study and tool, and possible next steps. I conclude by summarizing everything that was discussed within the thesis.

Lastly, the **Appendix** contains all supplementary materials, such as: the PathOS+ manual, the handout that explained the task, the interview guide, and a changelist of potential features to be integrated.

1.6 Summary

Games are a fascinating medium with a deeply interesting history dating back to ancient times, and it has only increased in popularity in recent years. With this ever-growing

popularity comes a need for methods that help to improve a game's experience. This can be difficult due to the nature of game development and artistic endeavors in general.

HCI is a field that emerged in order to better examine our relationship with technology, and improve the experience. GUR is one of its offshoots and it has been instrumental in determining how to improve games, though GUR methods have their weaknesses. Expert evaluation, for instance, is beneficial at the beginning of a game's cycle, but can be highly subjective. Whereas playtesting allows researchers to see firsthand accounts of players playing games which can help them gain useful insights, but it is resource intensive and inaccessible. Research suggests that playtesting data could be combined with expert evaluation to make the findings more accurate, but then there is the risk of losing the ease of use and quickness that are the key benefits of expert evaluation.

The emergence of AI has been meteoric, and in the current era there are algorithms that can mimic elements of human behaviour. It's through leveraging AI algorithms that some of the disadvantages of playtesting have been mitigated, through the use of AI agents that can play through games in place of humans while emulating player behaviours. We theorize that the data generated from AI playtesters can be used to supplement expert evaluation in order to add validity to any claims that are made, while still retaining the benefits of expert evaluation.

PathOS was an open-source tool created to facilitate AI playtesting of 3D levels within the Unity engine, with its original purpose being to help in the level design process. We decided to expand PathOS in order to tailor it for aiding the expert evaluation process. We've dubbed this variation PathOS+.

The rest of this thesis covers how we analyzed previous study data in order to determine what features to add to PathOS+. It covers the development of these features, the study we ran in order to determine whether PathOS+ fulfills its intended purpose, and what insights we gleaned from participant responses. Overall I conclude that PathOS+ can be a beneficial aid for expert evaluation.

2 | Related Work

The identification of issues—whether it be by predictive or experimental methodologies—can play an important role in game development. The data from GUR methods can be used to inform decisions on a game's design [33], and can help highlight problem areas to help developers improve the experience of their game.

Playtesting methods have been shown to be helpful in identifying gameplay issues and gaps in the player's understanding of the game rules [34]. Expert evaluation methods have also shown to be effective at identifying issues, especially through the use of heuristic guidelines that were developed over time (covered in further detail in Section 2.1.2). There are also some unique GUR methods that can be leveraged to identify issues.

Biometrics are a way of collecting physiological data from the player, usually through methods such as eye tracking and electroencephalogram (EEG) to name a few [35]. They let developers gain a more intimate understanding of the player's feelings towards a game. Existing research suggests that they can help detect problems in gameplay and emotional immersion, and can be effective when used in conjunction with other methods [36]. There exist approaches such as biometric storyboards, which help to visualize biometric data for better understandability and analysis [37].

Some studies showcase the use of biometrics in action, such as the research by Clerico et al. [38] where they look at whether biometrics can help shed light into nebulous concepts such as "fun". They got 62 participants to play some missions in *Assassin's Creed Unity* (Ubisoft Montreal, 2014) and recorded their physiological signals (such as electrodermal, cardiac, and electromyographic activities). Participants also self-reported how much fun they were having at different parts of the game. They had relative accuracy when it came to identifying physiological responses correlated with how much fun

the player was having. This led them to theorize that biometrics can be helpful for understanding player enjoyment and useful for adaptive gameplay. Alternatively Halabi et al. wanted to explore how players would identify the severity of various issues [39]. They did this by combining surveys with biometrics data. Their findings showed that observational methods can help with gameplay issues, whereas biometrics can help with identifying usability issues. Overall they argue that their insights can help developers make more informed decisions when it comes to selecting research methods, and also that it's important to take into consideration the player's perspective when deciding the severity of issues.

Telemetry—the process of collecting in-game player data—can be helpful in determining pain points within games [40]. As an example, Gagné, El-Nasr, and Shaw wanted to explore the use of telemetry for specifically real-time strategy (RTS) games [41]. They collected telemetry data of a Flash RTS game called *Pixel Regions*. Using their visualization tool "Pathways" they were able to see data pertaining to level information, movement information, and death information. This data helped them gain insight into whether players were playing the game as intended by the designer and what gameplay strategies they employed. When used in conjunction with other GUR techniques telemetry can be a powerful way to gain a better understanding of player behaviour. This was further explored in a case study done on the racing game *Pure* (Disney Interactive Studios, 2008) [42]. Telemetry data was combined with methods such as player self-report diagrams and biometric storyboards. This allowed the researchers to gain an intimate understanding of the player's experience, such as where they seem to be getting enjoyment and which situations led to players changing their playstyle. By looking at arousal states and combining that with interviews they were also able to identify gameplay issues, such as when the game behaved differently to player expectations.

Lastly, diary studies are when participants play a game within their own domestic environment and record their activities [43]. This can be an effective way to understand the player's motivation and overall user experience. Hillman et al chose to run diary studies on *NHL 16* (EA, 2015) during its public beta, They combined diary studies with

semi-structured interviews, a large-scale survey, and mindmaps [44]. Their findings helped them learn about how the users communicated, built teams, and discussed their grievances in the game, which was all useful information for the developers.

Aside from these different methods, there has also been work done to investigate the validity of GUR methods as a useful component of game development. To that end Mirza-Babaei et al. conducted a study comparing three variations of a game [45]. They used a 2D game *Matter of Seconds* and conducted a user test (UT) on the game using a classic method for one, and biometrics for the other. Three designers then provided feedback on changes that could be made to the game, with one being informed by the UT report, the other being informed by the biometric UT report, and the last one using their own expertise. Programmers then built these variations. 24 participants then playtested these variations, and a survey and interview was conducted with them in order to gain their insights. Based on the results the researchers found that user testing can help significantly improve the quality of a game, and biometric storyboards can help with more nuanced design decisions.

There has also been research done to explore the impact of issue identification on a game's critical reception. Mirza-Babaei et al. noted the lack of research done on the impact of playtesting on a game's reception, which can be problematic in establishing the value of playtesting in the eyes of different developers [46]. They examined three different games as case studies, which all had different types of playtests done on them before release, the data of which they had access to. For each game they referred to the Metacritic reviews and analyzed them via a hybrid deductive and inductive content analysis method. Features within the reviews got coded into different categories, and this was compared to the data from the playtest reports–namely, which issues or "features" were identified and how they were analyzed. Their results indicate what playtests could focus on in order to evaluate user reception on a game (such as the quality of core mechanics). Overall this study indicates that playtests can help inform designers on the reception of their game.

The rest of this chapter covers expert evaluation (its applications and caveats) in indus-

try, and also the field of AI and how that can be used for playtesting purposes. Everything covered here is in some way related to the conceptualization and development of PathOS+.

2.1 Expert Evaluation

Expert evaluation (also sometimes referred to as expert reviews) is a predictive methodology that can encompass techniques such as cognitive walkthroughs and guideline-based evaluations [47]. It is preferable for these evaluations to be conducted by a small handful of user researchers (roughly around three [48]). Expert evaluations have been successfully used in software development for decades to identify usability issues, but are still relatively new when it comes to game development. The degree to which these methods are used in the industry vary from studio to studio [16]. With that being said, there are still some notable examples of the usage of expert review methods in game development.

2.1.1 Uses in Game Development

Notable organizations like Player Research, who have worked with indie and Triple A studios alike, use expert evaluation as one of their methods [49]. They state that one of the strengths of the method is that in order to conduct it they don't need to use a playable build, since any existing material from the game would suffice for analysis. Additionally, since evaluations can be conducted very early in a development cycle, they can identify and remove issues from the onset. This is a common strength attributed to expert reviews (heuristic or otherwise). In a study done by Korhonen, Paavilainen, and Saarenpää they tasked user researchers with conducting expert reviews [50]. Participants stated that this methodology was well suited for the evaluation of games because of the potential to identify playability problems, especially early in the development process.

Research was done to show how expert evaluation methods can find issues that might not otherwise have been noticed. Saul Laitinen did a case study in order to determine

whether methods like expert evaluation can be useful for game development [51]. For the case study, a top-down shooter named *Shadowgrounds* (Frozenbyte, 2005) was evaluated while it was still in development, after which it was usability tested. The game's developers were made to rate each of the found issues in a survey. The results were promising, showing how 43% of the problems found were ones the game developers hadn't noticed before, meaning that they could fix those problems before the game's release. Yong Jun Choi did something similar in his research, in which an action game named *Midnight Strike* was evaluated and then conducted a usability test on, for game developers to then rate the results [52]. The game developers reported that they would not have found over 60% of the reported issues, and overall the developers found it to be a useful process.

Usually special considerations need to be taken in order to adjust expert evaluation methods for game development. This is especially evident when it comes to heuristic reviews, which are among the more common methods in game development. There already existed heuristic guidelines for software applications outside of the games industry, but those guidelines would not work well with games. As discussed by Korhonen, Paavilainen, and Saarenpää, the design goals between traditional software and games are widely different, and any created heuristic guidelines need to reflect those differences [50].

2.1.2 Heuristics

The number of heuristic guidelines that could be used to analyze games are numerous and vary in applicability. First invented by Jakob Nielsen and Rolf Moloch, heuristic evaluations were intended for evaluating "human-computer dialogue" (essentially, user-facing software interfaces) [53]. They conducted a survey with 77 designers and programmers in which the participants were tasked with conducting an evaluation task. This led to the preliminary creation of heuristics for these interfaces, which were later refined into the famous Jakob Nielsen's 10 Usability Heuristics [54]. Nielsen described this method as a quick process which involved getting a small handful of researchers to

separately evaluate a software based on heuristic guidelines and then reconvene to discuss and combine findings [55]. This process has stayed more or less the same, the only difference is the style of heuristics being used.

One of the first verified heuristic lists developed for evaluating game playability was created by Desurvire, Caplan, and Toth [56], titled the Heuristic Evaluation of Playability (HEP). These heuristics were based on existing literature at the time and had input from design experts. The guidelines within this set were organized into four categories: Gameplay, Game Story, Game Mechanics, and Game Usability. They used a game in early development that they playtested with 4 users, the results of which indicated that HEP viably could be used to help improve the design of a game early on in its development. Years later, Korhohen, Paavilainen, and Saarenpaa also confirmed this, while noting some areas of improvement [50]. They stated that while HEP was useful, some of the categories seemed too specific, or it was difficult to find a heuristic that fit a given scenario.

Based on the feedback that HEP received, some years later Desurvire and Wiberg released a paper on heuristic guidelines that were intended to be broader and more generalizable to different types of games (with a specific focus giving to action-adventure, first person shooter, and real-time strategy games). It was called the Heuristics of Playability (PLAY) [57]. To determine its facets the researchers created surveys for each of the genres they were focusing on, and got participants at a game expo to fill them out. Based on the survey results the categories for PLAY were determined, which were: Gameplay, Coolness/ Entertainment/ Humor/ Emotional/ Immersion, and Usability/Game Mechanics. Each category had various subcategories representing different aspects of a game, making this one of the more fleshed out heuristic guidelines.

Other general heuristic guidelines were created over the years. One framework was the Playful Experiences (PLEX) framework, developed by Arrasvuori et al [58]. Instead of being organized into different groups representing broad categories, the PLEX framework is built up of 22 categories dubbed "experiences", each representing a specific emotional state. Lucero et al. ran some studies determining how useful PLEX was as a

guide for expert evaluation, and they found that it worked very well, albeit its simplistic nature [59]. In another example, Pinelle, Wong, and Stach set out to create heuristics specifically to evaluate the usability of a game [60]. They determined this heuristic list by looking at 100 game reviews, and from there deriving 10 usability principles. They tested the heuristics by getting five researchers to run a heuristic evaluation of a PC game while using these principles, and found that it helped to identify issues, but more tests would have to be done to validate it further. Another example comes from Tondello et al., who wished to create a heuristic set that looked at motivators (intrinsic, extrinsic, and otherwise) when it comes to gameful design [61]. They ended up creating 28 heuristics that could help evaluators recognize gaps in a gamified application. They ended up running a study to validate this framework later [62] with five user research experts. These experts conducted an evaluation of two online gamified applications, the first three while using the heuristics, and the last two without. This allowed them to compare the results between the two groups. Overall, they found that those who used their heuristics were better able to identify issues.

Over the years games branched out into new mediums and genres. This meant there was a need for heuristics that reflected the nuances of these different games. Especially for mobile and virtual reality games.

Korhonen and Koivisto created one of the first heuristic sets for mobile games called the Playability Heuristics for Mobile Games, which was built off previous heuristic guidelines [63]. Their categories were Gameplay, Game Usability, and Mobility (which they added to take into account how mobile games are played in varying environments, and thus are prone to interruptions). These heuristics also took into account considerations for multiplayer mobile games. They validated these heuristics by choosing five different games, and found that while the heuristics helped to identify various issues, there were some playability issues that didn't fit into any of their identified modules, indicating that more research would have to be done. On that note, Kumar, Goundar and Chand recognized that Neilsen's heuristics may not be suitable for mobile applications [64]. Their solution was to examine 10 heuristics case studies for mobile games and

from there create a framework for conducting heuristic evaluations on mobile games, divided into planning, conducting, and reporting. They conducted a study to evaluate this, which showed how the framework seemed straightforward to use, but they have to conduct more studies to validate it.

Similarly there was a need for ways of evaluating virtual reality games, which were a rapidly emerging platform. Sutcliffe and Gaunt decided to do just that by creating a heuristic framework for VR environments [65]. The heuristics set consists of 12 heuristics focused on usability within a virtual evaluation, while the evaluation method follows Nielsen's guidelines with an additional technology audit step. They evaluated these heuristics with the use of three virtual environments and found that the heuristics helped with identifying issues in virtual environments while also providing recommendations. More recently, citing VR's increased emergence into the mainstream, Murtza, Monroe, and Youmans created a new heuristic set specifically for evaluating virtual reality applications [66]. They created these heuristics by surveying 85 participants, which included VR students and industry experts. The results let them identify themes with which to base their heuristics off of (such as physical space constraints and mental comfort). Though they cited a limitation of this study being that the samples they chose may not be representative of the industry in broad terms.

2.1.3 Caveats

Studies have been done to evaluate how well these heuristics function, with some conflicting results. Korhonen, Paavilainen, and Saarenpaa did a study comparing two of the covered heuristics (HEP and playability heuristics for mobile) and found that while there were clear benefits, the heuristic sets themselves had issues [50]. HEP was reported to have too many heuristics that were too specific to apply to any game. The playability heuristics were better in that regard, but they were spread across different documents which made it inconvenient to use. This highlights a trend to be wary of, where these heuristics are either too specific or too cumbersome to use.

There were other issues arising within all of these methodologies, issues inherent to

predictive methodologies. Methods like expert evaluation, and those under it, only act as predictions for possible issues players may encounter. Without player data it is difficult for user researchers to justify findings relating to subjective metrics such as fun, pacing, and difficulty [16]. The reality is that it is very possible for major issues to be missed entirely, or for smaller issues to be given more significance than they deserve, due to the lack of player data.

This issues of false positives is shown with the research done by Baauw, Bekker, and Barendregt [67], who created a method for evaluating children's education games called "Structured Expert Evaluation Method" (SEEM). It was composed of questions the evaluators had to ask every frame. They evaluated SEEM on two things: thoroughness and validity. On two games (*Rogers* and *Milo*), they ran a usability test and identified core issues. Then they got 18 evaluators to conduct an expert evaluation using SEEM. They found that while the evaluators identified lots of issues, they also identified lots of false positives, meaning that the validity of their method was lower than anticipated. They chalked this up to being the evaluators over, or under, estimating children's cognitive abilities while also having a misunderstanding of the game itself. Thus, while the evaluation showed that SEEM could be used to detect problems, it also indicated that adjustments would have to be made. False positives are an ever-present risk with predictive methodologies.

A solution to make predictive evaluations more accurate was proposed by Tan, Liu, and Bishu, who suggested that combining usability testing alongside expert evaluation methods can help make up for any false positives identified within tests [18]. Similarly, the Nielsen Norman Group stated that combining usability testing with expert evaluations can help evaluators identify issues they might not have thought of [68].

There are also some other issues that arise in regards to the mediums used to conduct expert evaluation. Some research papers explored the pros and cons of conducting evaluations via traditional methods—such as pen and paper—versus using web-based or software tools to do the process. When Hvannberg et al. compared the two in a case study, they found that participants who used an electronic tool to conduct their evalua-

tions found the process to be more efficient, and less tedious [69]. Though they found that context-switching (swapping between the game itself and the tool for conducting the evaluation), could cause some difficulties for the evaluator. Similarly Sivaji et al. created a web-based tool titled Usability Management System (UMS) for conducting expert evaluations [70]. They ran tests in order to validate this, and found that using a web tool helped streamline the evaluation process.

We kept all of these considerations in mind when developing PathOS+. We wanted to focus on providing concrete evidence for evaluations, while also streamlining the process so that researchers can conduct their evaluations where the game is located. These caveats are also why we chose not to base our tool on a specific set of heuristics, which is covered in more detail in Section 5.

2.2 Artificial Intelligence

The applications of AI within the games industry are wide and varied, with different purposes. AI is typically used for in-game enemies with varying degrees of complexity, but also in terms of software meant to make aspects of the game development process easier. While the latter is not commonplace, there have been more conversations as of late on ways that AI could be integrated into the process to either automate things or make things easier. The following covers some of the different areas of AI applications.

2.2.1 AI for QA

Errors in a piece of software—-referred to as bugs—-can cause a frustrating experience for the user. When it comes to software development, there exists a multitude of tools to aid in the detection of errors within software, along with tools that can help fix those errors [71]. It's much harder to do the same thing when it comes to games due to the intricately complex and nuanced nature of them. Current bug testing processes within game development can be very difficult and involve a lot of manual labor [72]. Human testers typically play through the game in order to detect and log the bugs they find. This

can be strenuous and thankless work, and there is no guarantee that all if not most of the bugs will be found. As of late, there has been more and more research put into the use of AI as a way to ease the burden of bug testing, and to make it a more efficient and successful process.

There are many AI bug-testing tactics that involve combining the strengths of both humans and AI to identify bugs. This is explored by Chang, Aytemiz, and Smith who decided to explore ways of aiding human testers to expand their search capabilities with AI [73]. This led to the creation of Reveal-More, a framework that takes a small recording of human gameplay data and uses AI to expand the coverage of potential bugs. The algorithm works by using the save states of a game as a human plays through it, and once the human is done playing the algorithm uses the save states as "seeds" for exploration. For each save state, the algorithm examines more of the game from that state with the use of the Rapidly-Exploring Random Trees (RRT) algorithm, and after a set amount of time moves onto the next state. To prove the algorithm's capabilities, they ran tests for the games *Super Mario World* (Nintendo, 1990) and *The Legend of Zelda: A Link to the Past* (Nintendo, 1991). Their tests showed that Reveal-More was able to cover approximately two times more of the gameplay than human testers while also reaching inaccessible areas, showing its potential in increasing the coverage of bugs. These researchers also plan on expanding this algorithm further, by increasing its capabilities of useful actions and the ability to examine data in between game builds. In a similar vein, this was also observed by Machado et al., who created Cicero in order to explore the applicability of AI helping with the game development process [74]. This is a level design tool that can allow for the prototyping of 2D sprite games in different genres, and also the subsequent bugtesting of those games. It's built upon the General Video Game AIFramework (GVGAI), and it also used SeekWhence (a replay analysis tool), Playtrace Aggregation (a visualization system) and a Mechanics Recommender (a tool to recommend mechanics). To test how it works, they ran a study with 32 participants, and got them to do different tasks with (and sometimes without) the AI. Overall they found that for the most part humans performed more accurately able to locate bugs with

AI assistance, though they argue that more testing would have to be done to verify these results. Additionally, citing certain shortcomings in existing AI algorithms for playing through games, Pfau, Smeddinck, and Malaka developed a novel framework called ICARUS to play and report bugs [75]. It was developed at the adventure game developing company Daedalic Entertainment, and it's made for the Visionaire Game Engine. It automatically plays and highlights issues into builds of this studio's games. While it does not completely reduce the load on the QA team, they can still use it to supplement human bugtesting and reduce the workload.

Alternatively Gordillo et al. believe that human playtesters should focus on the general experience of a game, while AI focuses on the identification of bugs [76]. To that end they used Reinforcement Learning (RL) agents to try to maximize bug coverage within 3D levels. They make it so that the agents are inherently curiosity driven so that they prioritize exploration. They created large, complex 3D maps for the purposes of evaluating the AI agents, and they trained the agents using a proximity policy organization algorithm. Due to its speed and efficiency they were able to train and simulate 320 agents on different machines. The agents were successful in exploring the maps, and their tool allowed them to create visualizations of the agents' exploration. They make the claim that a tool like this could help pinpoint locations where players could get stuck in a level along with exploits and glitches. They want to increase the complexity of this approach for future work by introducing things like new hazards and environmental features.

There are also some AI algorithms that can identify bugs solely on their own, especially for bugs which are harder to find. For instance, It can be really difficult and time-consuming to detect graphical bugs in a game, partially due to how rare these glitches can be. Manual testing processes include humans watching camera footage of the assets to see if anything is amiss (so called "smoke tests"), which is not the most efficient way of catching these types of bugs. Ling and Gisslén decided to use deep convolutional neural networks (DCNNs) to try and solve this problem [77]. The algorithm would be fed an image that showed the "correct" version of a certain graphical asset, and then there would be four images showing an incorrect version of that same asset. Subsequently the

algorithm was trained within Pytorch, and its performance was evaluated in different test scenes. After testing it, they found that the algorithm was able to detect 88% of bugs. For future work, the researchers will aim to train the algorithm to recognize a wider variety of graphical issues.

Citing that bugs have the potential of causing major grievances if they are found in a released game, Varvaressos et al. believe that the software verification technique, runtime monitoring, can be used to quickly find bugs as an alternative to manual testing [78]. This technique works by observing a piece of software during its runtime and detecting deviances in expected behavior. To make it applicable to games, the researchers took advantage of how most games have a similar structure, i.e. the use of gameloops for the iterations of the game. Through the use of a monitoring software called Beep Beep (a Java-based monitor for Web applications) the algorithm took snapshots during each iteration of the loop in order to look for variances. To explore the efficacy of this technique, the researchers did a case study where they examined two PC games: *Pingu* and *Bos Wars*. For each game, they identified what rule violations (or bugs) would look like, then ran tests with their runtime monitoring architecture. The test showed that the program was able to detect both known and unknown bugs without any issues, though the researchers are working on improving the algorithm to be more sophisticated.

Bug testing is not the same as playtesting. However, looking at the AI techniques that have been used for bug testing purposes can help inform the strategies we use for playtesting tools.

2.2.2 AI for Analysis and Development

Understanding player behaviour patterns in games can help designers craft gameplay experiences that appeal to specific player types, and there has been research done to classify players into certain broad typologies. The earliest recognizable example is Bartle's Taxonomy of player types, which put players into categories of Explorers, Socializers, Achievers, and Killers [79]. There have been many other typologies since, such as BrainHex (which divides players into Seekers, Survivors, Daredevils, Masterminds,

Conquerors, Socialisers, Achievers) [80] and Newzoo's (an esports market analytics company) Gamer Segmentation (in which players are divided into Ultimate Gamers, All-Round Enthusiasts, Time Fillers, Bargain Buyers, Community Gamers, Hardware Enthusiasts, Popcorn Gamers, Backseat Viewers, Lapsed Gamers) [81]. AI can be an aid for data analysis purposes by identifying and modeling player types in games.

This is shown in the work of Drachen, Canossa and Yannakakis, who wanted to construct player models for *Tomb Raider: Underworld* (Crystal Dynamics, 2008) (also known as TRU) [82]. They gathered player data from the engine itself via the EIDOS Metrix Suite and through the use of the XBox Live Web Service. They got data from 1365 players from which they extracted six features: ways of dying (opponent, environment, falling), total number of deaths, rate of completion, and help-on-demand (which was a method for players to request aid). They used an unsupervised learning approach called Emergent Self-Organizing Maps (ESOM) which was trained using a batch algorithm. It was subsequently able to identify 4 player types with distinct behaviors: Veterans, Solvers, Pacifists, and Runners. The information gained can help developers determine if the game is being played as intended and also make adjustments to the game's mechanics. Similarly, Melhart, Liapis, and Yannakakis wanted to explore how player experience could be generally modeled in games across genres (with a specific focus on racers, shooters, and platformers) [83]. Through general player experience modeling and the use of telemetry data from the Affect Game AnnotatIoN (AGAIN) they were able to derive key features from each genre and develop player models for them. Their models had relatively high accuracy when it came to predicting behaviours and they were able to determine that general game features can be used for player modelling purposes.

It is also possible to model player navigation, which can then be used to gain further insights about the player's flow through a level. Thawonmas, Kurashige, and Chen proposed that they could look at movement patterns within online games and glean useful information from it for game design purposes [84]. They used an algorithm called Self Organization Map (SON), which uses inputs that are transition probabilities between landmarks in a given map in order to cluster online gamers into categories, from which

they can infer player types. The researchers then argue that this data can be taken in order to make design decisions for the sake of player satisfaction, and for future work they wish to improve the capabilities of the algorithm that they used.

On a similar note, inferring the behavior from in-game trajectory information can be really helpful for the implementation of believable game AI while also helping to improve map designs. Bauckhage et al. wanted to explore the spatial behavior of players with certain clustering algorithms (DEDICOM and DESICOM) within complicated 3D game environments. They used Quake 3 vs Unreal Tournament 2003 for their data. They ran two experiments, the first of which involved them doing spectral clustering, the second of which involved them examining different play styles. Their results show that these algorithms can provide meaningful clusters of player trajectories that can be helpful for the purposes of design, and also that the DEDICOM algorithm is better for speed, whereas DESICOM is better for more easily interpretable information.

There are other ways that AI can be used to understand and analyze human behaviour, beyond identifying and emulating specific player types. For instance, AI can help better understand player emotions. Roohi et al. were looking at recognizing player expressions with deep neural networks and long short term memory (LSTM) via a technique called Affect Gradients [85]. Frames of videos of players playing through *Infinite Mario Bros* (a 2D game based on the original *Mario* games) would be fed into their algorithm. From the ensuing results their expressions would be identified. The results show that this method can be used to identify expressions, but work has to be done on how it interprets different emotions. For instance, when players are detected as showing happy expressions when they're dying, it may be due to them feeling embarrassed or laughing at themselves. On the other hand when they appear to look angry it could just be that they're concentrated. Care would also have to be put into how faces are recorded, because if the camera is angled the wrong way the facial expressions won't properly be captured.

Javvaji, Harteveld, and El-Nasr took note of how these types of AI methods have been used to analyze large datasets [86]. However they also point out that this data can be dif-

ficult to interpret for non-experts, and sometimes the data does not make it immediately obvious to the designer what the player patterns actually are. To address these caveats they used knowledge based abstraction and a visualization interface called Glyph on an interactive in-development puzzle game called *Daedalus*, which could be played in Slack. They ran a study with participants who were tasked with figuring out the puzzles. Using their proposed technique, they were able to identify the following player types: Early Dropouts, Arduous Dropouts, Dodgers, Resolute Finishers, and Elite Players. The data helped them find issues with balancing (i.e. players were dropping out very early) along with unexpected player patterns. The results were then used to create guidelines for abstracting raw data in order to analyze player behaviour. Their work shows how large datasets can be made manageable with AI, along with the benefits of context-driven analysis. Though one limitation is that this method is made for a very specific game, so it's to be determined how well it may work on other types of games.

The identification of balancing issues is one of the ways that AI can directly help the game development process. As an example, to improve and accelerate the playtesting process for *Hearthstone*, García-Sánchez et al. looked into using an Evolutionary Algorithm (EA) [87]. The EA created decks that were played in matches by AI versus human-designed decks. They used a heuristic mutation operator to shorten the search through the space of possible decks, and used the framework *MetaStone AI* to simulate the matches. A human would analyze the playthroughs in order to find imbalances. Their results showed that the AI generated decks outperformed the human-designed ones, showing how the method works well for finding optimal decks. They were also able to see cards that would frequently show up in the AI decks, and from that determine that those cards are unbalanced or overpowered. They also think that this AI approach is generalizable for different kinds of deckbuilding games. They mention that there is further work to be done in the future, such as optimizing the score function.

An aspect of game balancing is finetuning difficulty, which AI can be helpful for. Roohi et al. used deep reinforcement learning (DRL) game-playing agents for player modeling [88]. They did this to identify churn rates, when could then be used to infer certain

things about the game (i.e. difficulty of a given level) and inform design decisions. They combined their DRL agents with monte carlo tree search (MCTS) and evaluated the algorithm on 168 levels of the free-to-play game *Angry Birds Dream Blast*. They found that combining these techniques allowed for higher prediction accuracy with less of the computational costs, meaning it could help developers determine how difficult their levels are. Future work would include testing this process on a game in development, instead of one that has already been released (albeit with frequent updates). Similarly, Shin et al. acknowledge that one of the biggest challenges for match 3 games is building levels that keep users continually interested in the game by balancing the difficulty adequately [89]. The difficulty of a level can be determined by how many moves are needed to complete it. Their goal is to use AI to check the planned difficulty of a level and confirm it via AI playtesting. They identified different strategies of completing the level referred to as strategic plays. From there their agents used a policy-based reinforcement learning method called advantage actor-critic (A2C) in order to learn the game boards, and OpenAI Gym to update the current state of the board. In order to evaluate this method, they used *Jewels Star Story* and compared the average number of moves it took an AI to complete each level versus a human tester. Their results showed that the agent was highly accurate and their technique appeared to perform better than the industry standard at the time. Thus indicating that their method was a viable way for a designer to determine how difficult a level in their match 3 game is.

Not all difficulty in games is static. Dynamic difficulty adjustments are something that can be leveraged to accommodate the differing abilities of players. Hawkins, Nesbitt, and Brown noted that in order to dynamically adjust difficulty it's relevant to consider how likely a player is to take risks within a game. To address this, they used a particle filtering technique (the filters of which use sequential MCTS methods) to try and create idealized player models that are based off of their risk profile [90]. They can vary the number of particles in the filter in order to create agents that perform differently. In order to get data for their models they ran a design challenge with a number of participants where they were shown a grid of squares filling up with dots, and they had to quickly

choose the one that was filling up the quickest. This challenge helped to determine how risky and accurate the player's behaviour was. The application of models such as these during game development is that they can be used to help determine what difficulty bracket a player may fall into. Then the game can dynamically adjust its difficulty to that player's level. While useful, the researchers acknowledge that a limitation of the work is how it requires some pre-existing empirical player data.

Zook et al. wanted to see if they can reduce the time-intensive expense of game balancing by automating some of the more tedious tasks [91]. They focused on parameter tuning—a method in which designers make low-level changes to games for the sake of balancing. They used a specific machine learning technique called active learning (AL), which (in this scenario) took in game parameters as inputs, while the output was a game parameter tuning goal. In order to validate this method they created a simple shoot-em-up game and tested player gameplay behaviour goals and player subjective response goals. To collect data for their model they deployed the game online and obtained data from hundreds of players. Their AL algorithm used this data set in order to pick sequences of playtests that could achieve the desired design goals. Their findings showed that AL helped reduce the number of playtests needed. They also pointed out the limitations of this method, such as how the game they chose was convenient for this method and may not be suitable for more complex scenarios. Overall they believe that this work is a good first step when it comes to using AI to aid in game design tasks.

Lastly, AI can be used to build levels. As an example, it can be difficult to determine what makes a good level in a physics-based puzzle game, and difficult to test for playability (which is whether or not players are able to complete the level). Shaker, Shaker and Togelius presented a tool called Ropossum [29] which lets designers edit procedurally generated levels for the game *Cut the Rope* and ask AI agents to solve them. The tool also makes suggested modifications so that the final design is deemed playable. The system contains an evolutionary framework (using Grammatical Evolution) for procedural content generation (PCG) and a physics-based playability module to solve the game. When it comes to the agent, it infers the next best action. If the sequence of actions

doesn't lead to winning the level, the agent backtracks. A state tree is generated that represents the actions and states explored. The method worked well for the purposes of these researchers, however the application of this method is limited to just this specific game. Further research into these types of tools for a wider range of game genres could be invaluable.

2.2.3 AI Simulation Testing

It's possible to use AI as a stand-in for human participants when it comes to playtests. While this application of AI is more recent than data analysis with AI, there has been some notable literature covering the benefits and challenges of such an approach.

One of the promising applications of AI testers is that they can be used to simulate different player personalities. Holmgard et al., while acknowledging how difficult it is for game designers to get an understanding of how players react to their ingame content, proposed using archetypal generative player modeling for game testing [92]. They called this approach "Procedural Personas for Playtesting". They evolved agents to play the rogue-like turn-based game called *MiniDungeons 2*. It makes use of MCTS, but with generative programming. By running the tests they were able to identify four player archetypes: Runners, Monster Killers, Treasure Hunters, and Completionists. Their agents were able to efficiently and robustly play the game with distinctive playstyles. The information gained from watching these agents can help influence a level's design. Future work could look at improving complexity and incorporating human data for the playstyles.

Ariyurek, Surer, and Betin-Can wanted to expand the applicability of these personas in order to make playtesting a more feasible process [93]. Original personas are rigid in what they can emulate, and they may stick to specific goals or "rewards". They propose an RL APF (alternate path finder) method in which the agent is encouraged to explore more expansively, and gets punished for visiting previously visited states. This approach is known as goal based personas. These can be used for personas that have to change their personality type during a playthrough, i.e. they could focus on defeating

enemies, and then switch to collecting treasures. They evaluated and tested their game on GVG-AI and VisDoom environments. The result shows that the designer can use this method to see how different kinds of players can interact with their game. Though one of the big limitations with this work is that it only works on games that a RL agent can play. For future work they wish to experiment with more persona types along with 3D environments.

Making simulation AI behave like humans can be a big challenge. On that note Gudmundsson et al. noticed that a lot of AI game playing methods don't take into consideration player data from already released games [94]. They decided to leverage player data to create more human-like behaviour with the use of convolutional neural networks (CNNs), which they deemed a promising solution. They trained agents on *Candy Crush Saga* and *Candy Crush Soda Saga*, and they focused on predicting the difficulty levels of match-3 games in order to help with game design. The approach allowed them to figure out the difficulty of a given level in minutes, whereas it would've taken days with human testers. They've been using this method extensively in hundreds of levels and the accuracy of the predictions have been stable.

An alternate approach to make AI perform human-like is what Zhao et al. proposed [95]. They think that in order for agents to behave human-like in-game there needs to be an emphasis on two components—"skill" (how good they are) and "style" (how they play). They used four different use cases where they trained the agents with skill and style in mind. Two of the case studies centered around playtesting AI agents, whereas the other two focused on game playing AI agents. For the former, they tested the experience for different play styles and to measure competent player progression. For the latter, they looked at human-like progression of an open world game, and game-playing AI assistance. They found that the techniques they used for different case studies varied, and that RL algorithms can't go from the benchmark to the real thing without heavy parameter tuning. In a similar vein, Glenn and Brunstad wanted to explore the degrees of skill players can demonstrate for games like *Yahtzee* through the use of AI testers [96]. They used supervised learning, based on neural networks, in order to have AI agents

play through the game. The AI achieved highly optimal scores, though they mentioned that the results could be further validated, and for future work they were thinking of incorporating RL.

These AI testers can be used to play complex games and identify game breaking scenarios that humans may not have found. They can also be helpful with identifying game balancing issues, especially since the use of agents can allow researchers to run thousands of tests in a small amount of time. This was shown in the work of Silva et al., who wanted to explore the applicability of AI agents in playing board games [97]. They used the *Ticket to Ride* game as a test scenario. Due to the complexity of the game they were not able to use established algorithms such as A* or MCTS, so they had to create their own custom agents. They examined common playstyles and crafted 4 distinct agent types, each with their own playstyles. After simulating 110,000 matches they examined the results, and were able to find out useful information such as which agents were the best strategic choice for which game maps. They also found unique game-breaking scenarios that may not have been found without the use of AI. Similarly Silva et al. were tasked with running playtests on pre-release builds of the *Sims Mobile* [98]. In order to bypass the slowness of the game—which relied on things like finger taps to progress—they took the base mechanics and parameters of the game and created a separate version that had just those elements. They then used the A* algorithm for agent decision making. What would take testers days of organic gameplay would only take their AI minutes, because they were able to run thousands of tests. This also allowed them to explore various questions the designers had for them, such as whether there are significant imbalances within the different relationship categories—and their findings showed that it seemed to take too long for sims to enter romantic relationships. So designers were able to look at this information and actually make changes to the mechanics accordingly. For future work they discussed MCTS as being a possible viable alternative.

There are also other tools created for the purpose of making AI playtesting more accessible for independent studios, similar to PathOS+. Keehl and Smith created Monster Carlo 2, which they claim can help independent studios conduct playtesting [99]. They

combined Monster Carlo with Unity's MLAgents toolkit–combining search-based machine playtesting with the ability to learn through examples. With this they crafted a framework that can mimic powerful autonomous agents such as AlphaGo, while also being able to directly work with Unity's systems. In the previous paper that discussed Monster Carlo, the tetris-esque game *It's Alive!* was used, which the researchers decided to reuse in order to conduct a fair comparison. They conducted thousands of playtests using many different datasets. While the results weren't as promising as the researchers hoped, they believe that in the future their work could be improved upon by increasing the capacity of their neural networks and better tuning the hyperparameters. They still believe that the concept they explored is beneficial.

There also exists commercial tools specifically for the Unity engine that help to facilitate player simulations. One of those tools is Unity game simulation, a cloud-based service created by Unity themselves that allows developers to run simulations of their game, primarily with the focus of game balancing [100]. Another is GameDriver, which supports automated testing for Unity games, with more of a focus on quality assurance [101]. While these tools are able to simulate a wider variety of genres than PathOS+, they do not appear to combine testing with expert analysis like PathOS+ does.

2.2.4 AI and Visualization

Visualizations can be powerful in that they allow for easier analysis of complicated data, such as large scale playtesting data that may be difficult to analyze efficiently. Visualizations can help make information easier to parse.

There already exist techniques for creating computer visualizations of playtesting data. An example is demonstrated in the work by Wallner, Halibi, and Mirza-Babaei [102]. These researchers proposed an aggregated visualization approach that takes advantage of clustering, territory tessellation, and trajectory aggregation techniques. They decided to go with aggregated visuals because they can be less cluttered and easier to read. In order to test how well this approach performed, they recruited six testers to play through *Infinite Mario*, a 2D game based on the original *Mario* games. From these play ses-

sions they collected in-game data, physiological data, and observational data, all for the purpose of better understanding the player's behavior within the game. They then had two visualizations of this data: the non-aggregated version, and their own aggregation method. They recruited 9 games industry professionals whom they conducted expert interviews with, and also incorporated rating scales to evaluate each of the visualization techniques. The results showed that overall the aggregate visualization approach was more readable, more positively reviewed, and was rated as slightly more informative. Though there are still improvements that could be made to this approach, with a noted limitation being that level information would get obscured by the visualizations.

Such visualization methods can be used on AI playtesters, and there are some examples of visualization techniques combined with AI. For instance, visualization data has shown to be immensely useful in making information gathered from AI testers easier to comprehend. As an example Agarwal et al created a web-based visualization tool showing the navigational behavior of agents for testing *Sonic the Hedgehog 2* (Sega, 1992) [103]. They used the Neuro Evolution of Augmenting Topologies (NEAT) algorithm in Python to train agents to play the game alongside the Open AI Gym environment for the agents to play the game. This was then used to create visualizations, and the visual aggregates of the data showed the researchers what parts of the level agents were having troubles with, whether agents were able to find the secret paths, the basic trajectories of the agents, and so on. This information is highly useful to designers, who can then make improvements to their levels based off of it.

Similarly, while pointing out that one of the most effective ways of understanding dynamic interaction processes is by viewing the data sequentially, Agarwal, Wallner, and Beck set out to create a timeline based visualization approach that allows developers to examine the strategies and decisions of their AI agents [104]. The tool (dubbed "Bombalytics") is based around *Pommerman*, a variant of the classic game *Bomberman* (Hudson Soft, 1983). Their main goals were to give developers: an overview of event sequences in a game, a visualization of local patterns and repetitions, and an overview of a competition in a set of games. They accomplished these goals through incorporating various

visualization features such as a high level overview of the simulations, sequential timeline showing the steps in each simulation, and lifespans depicting actions relating to specific game objects. They gave an online questionnaire to 20 AI and analytics experts with varying levels of experience in order to validate the tool. The results of this study showed that while the tool is usable and helped in the understanding of agent behavior and performance, there is a tradeoff in how to provide useful information without the interface becoming overly complicated. Keeping the interface suitably decluttered can be an important thing to consider when crafting visualization tools.

In terms of AI tools that help streamline the analysis process when it comes to large player datasets, this is shown in the work of Braun et al. They noted that in the massively popular multiplayer game *Overwatch* (Blizzard Entertainment, 2016) new players may feel too daunted to join due to the high skill level required for the characters in the game (dubbed "heroes") [30]. They created an online game data mining algorithm with the goal of providing players a means by which to confirm how they are performing in comparison to other players through the use of clustering and visualization techniques. They also wanted to provide a general understanding of how these statistics may affect chances of winning. This method took data from current players of the game and clustered the data per hero using an IOS app called SiteSucker. They used an intelligent computation model to analyze the data and fix any quality related issues (such as missing or wrong data). Once the dataset was completed it got mined by an affinity propagation technique and the algorithm churned out plot graphs that depicted different gameplay-related metrics. While this approach can help individual players become more informed and possibly better at the game, as of the publication of this article the researchers are looking into ways of incorporating the data mined knowledge to maximize team winnings, and also incorporate machine learning.

As of now there are AI visualization techniques that are used in the broader field of analytics that have potential to be promising within the games industry. Despite that there is more work that could be done with how AI visualizations are used in games. With that in mind, tremendous strides have been made to show the benefits of visualiza-

tion techniques with player data and its analysis capabilities. Aspects of the PathOS+ visualizations are based off of some of this information.

3 The Development of PathOS+

This chapter goes over the entirety of the developmental process of PathOS+. This covers the features of the original PathOS, the study conducted to identify which new features to incorporate, and the changes made to create PathOS+.

Figure 3.1: Overview of PathOS runtime interface and level markup

3.1 Original PathOS

PathOS was originally created to help with the level design process. The goal was to achieve this by providing level designers with useful information by facilitating AI playtests of the levels that they created. Ideally this information would allow the user to see things like hot spots within their level, areas where players may get stuck or lost, and so on. This way they could see if potential players would behave as intended, or if anything needs to be adjusted. Figure 3.1 shows an overview of PathOS.

One of the key objectives of the tool was to make playtesting a less burdensome process. It was also intended to be accessible for game designers, and generalizable so that it could apply to different genres of games. With this in mind, Unity was chosen as the game engine for the tool to be built in, as it is popular and accessible for independent developers. Written in C#, PathOS makes use of Unity's physics and AI system, including

the NavMesh which allows Unity's AI agents to traverse through level geometry.

The entire project is free and open-source, with the files available on Github for anyone to download. Incorporating the tool into an existing project is a relatively straightforward process. Once the files are downloaded from Github they can be copied over to within the file structure of the Unity project in question. From there the PathOS elements—known as "prefabs"—can be dragged and dropped into the game scene in order to be used. These prefabs include everything necessary to run simulations, such as the PathOS Agent, the PathOS Manager, and the agent camera. Alongside these files is a demo scene, depicting how PathOS works in a typical game scenario to act as an aid for users.

The rest of this section covers the different facets of PathOS and how they function, with emphasis given to the interface, the agent, and the existing visualization features.

3.1.1 PathOS Interface

Since the primary target audience for this tool was game designers, the interface had to be such that it was easy to use while also obscuring the more complicated backend functionalities. In order to achieve this, PathOS's interface elements were seamlessly integrated into the Unity game engine. As an example, the inspector—a key aspect of the Unity interface—is used to allow users to edit modifiable properties without needing to dive into the code. This combination of unique PathOS elements with the familiar Unity interface is intended to help lessen the learning curve for the user (as long as they had prior experience with Unity) while also supporting their workflow. This section goes over each facet of the PathOS interface and explains each in detail.

Level Markup

Objects pertinent to the gameplay in the level need to be marked by the user as an "entity type". This is so that the AI agent can identify what it is looking at, whether it is an enemy, a resource, a goal, or something else entirely. Each entity type is represented by their own unique icon that is shown in the inspector in the level markup tab. This icon shows up on the entity in the Unity sceneview if the user has Gizmos enabled (which is

recommended).

In the level markup tab of the PathOS Manager every possible entity is shown alongside their respective UI Icon. This part of the interface uses a point-and-click system, in which the user chooses the entity that they are interested in, and then they click on a gameobject in the Unity scene to mark it as that entity type. In this mode, the brush is referred to as "the markup brush" since, instead of functioning like a cursor, its functionality is more akin to a paintbrush. Figure 3.2 shows this process in detail. This part of the interface also shows a list of all the current entities within the game level, which the user can then modify or delete at their will.

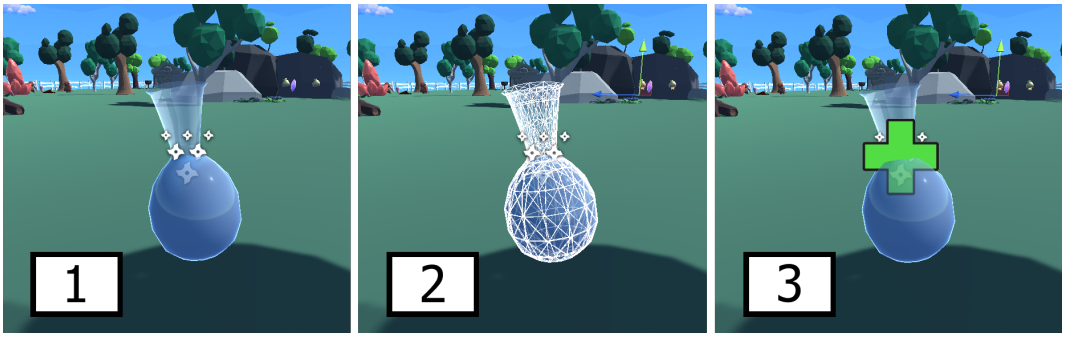


Figure 3.2: Overview of how items get marked

Runtime Interface

Simulations can be run by pressing the play button, which comes with the Unity interface. While a simulation is running several UI elements are shown within Unity's game view window. Within the window, if the agent is selected in Unity's hierarchy, it will show their first-person camera view in the bottom right and their mental map on the bottom left. The camera in the game window shows a bird's-eye perspective of the level that allows the user to pan through the game view and change how zoomed in the camera is to the agent.

For each in-game entity there is an icon shown to represent whether or not the AI has interacted with it: an eye if the agent is currently looking at it, a brain if that entity is not visible but is located in memory, and a red x if the agent has already interacted with

it. There is also a legend that can be toggled on and off. All of these elements exist to provide users with feedback so that they can understand the agent's rationale while also supporting real time observations. As an example, the mental map shows where the agent has gone so far. Looking at it can allow user to see if a particular part of the level remains unvisited, indicating that something may be difficult to get to depending on their intention.

Behavior Customization

After the PathOS Agent is created—typically through dragging and dropping the PathOS Agent Prefab into the scene, or copy-pasting the PathOS Agent script onto a gameobject—the user can modify aspects such as the movement speed and camera. This part of the interface also allows the user to customize the agent's personality.

Within the Unity inspector a slider is shown for each personality attribute (experience, curiosity, achievement, completion, aggression, adrenaline, caution, efficiency) for the selected agent, representing the range for each attribute. The user can modify these at will to change what the agent's personality is. For instance, if they want to see how an aggressive player type may play through their level, they can make the aggression attribute value high, and the caution value low. User also have the option of creating custom personality profiles, which are preset values that they can use to generate agents that fall within a specific personality type. Several default ones are already present within the Unity project (explorer, completionist, aggressor, novice). Attribute values can also be randomized if they so prefer. Lastly, the user is empowered to modify the scoring matrix for the attributes within the interface as well, if they wish to change anything about how the agent interacts with any of the given entities.

Batch Tests

While users can run individual agent simulations, they can also run simulations with multiple agents at once. In PathOS this is referred to as batch testing. This part of the interface is represented as a separate window that the user can open up by selecting

"Window" in the topmost toolbar and then clicking "PathOS Agent Batching".

Within this window the interface allows the user to determine how many agents they want to simulate, and whether they want to simulate multiple agents concurrently, or just simulate them one at a time. They can select a preexisting agent prefab located in the scene as an agent reference. There is also a slider that allows them to specify the timescale. When it comes to the agent personalities, all the agent attributes are shown, and users have three options: they can set fixed values for each attribute, specific ranges for each attribute, or use one of the predefined custom personality profiles to dictate the agent personality. Once the user is done modifying these values they can choose to start the simulation, and there is also a button that lets them stop it at any time.

3.1.2 Visualization Features

Figure 3.3: PathOS visualizations (from left to right): heatmaps, individual path, and entity interactions

PathOS also contains various visualization features meant to help the user with analysis purposes and help them better understand the agent playthroughs. The visualization components are located within the PathOS Manager prefab and are represented in the interface as panels that have modifiable variables. In order to make use of the visualization features, agent "logs" have to be recorded. To do so the user must enable logging and select a folder where these logs files would be saved. The logs are excel files that record the agent's position and rotation throughout the simulation.

This data is then loaded back in for the visualization features, which can take various forms. These visualization types can be enabled and customized via the interface in the

visualization panel. It is also possible for all of them to be enabled at the same time, layering on top of one another. The simulation data for multiple agents can be loaded concurrently, and the visualizations show the combined information for all the agent data currently loaded in and enabled. All of these visualizations appear on top of the level geometry in the Unity sceneview, and they can also be raised to different heights. An overview of these different types can be seen in Figure 3.3.

The first visualization type is heatmaps, represented with tiled squares (the size of which can be modified). These heatmaps can be adjusted in terms of the transparency (the alpha) and the gradient that is used (which color represents high density versus low density). The second visualization type is the individual paths. This shows an agent's trajectory throughout the level, and can be modified in terms of the color of the path itself and whether the user wants to only see specific trajectories. The last visualization type is entity interactions. These show 2D circles on in-game entities that the agent has interacted with. The size of the circles are dependent on how long the agent spent at that interaction, along with how many agents interacted with that entity. Modifiable properties include the gradient and whether there are labels of the entities shown for each circle.

For the visualizations overall it is possible to set a time range. This range makes it so that only the relevant data for a specific part of the simulation is depicted. Designers can also modify which parts of the visualizations they want to see, and can remove any data that is loaded into the Unity scene at will.

3.1.3 PathOS Agent

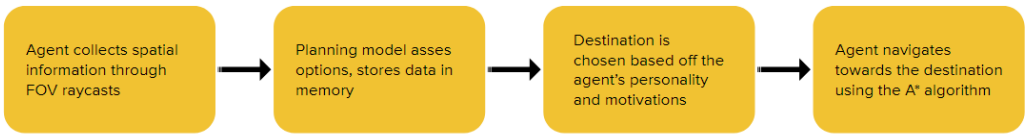


Figure 3.4: High level overview of how PathOS agents function

The PathOS Agent prefab is built up of multiple different components, which include

the C# scripts (PathOS Agent, PathOS Agent Memory, PathOS Agent Eyes, and PathOS Agent Renderer), Unity's NavMesh agent component, and the agent sight camera. All of these different elements work in tandem in order for the agent to function the way it should. The following section breaks down these different components in order to explain the functionality of the agent, with particular focus being given to the agent's sight, memory, and decision-making. Figure 3.4 shows an overview of the PathOS agent functions.

Sight

The goal of the PathOS agent is to predict and simulate player movement through 3D terrain. In order to do so agents have to have adequate information about the game world. Especially since the objective is to make the agent behave in a humanlike manner, this means imperfect information. The agent is not omnipotent, and does not automatically know where every entity was located. Part of how this was implemented was through the simulation of the agent's sight.

Sight is simulated through a point-of-view (POV) camera attached to the agent. Through the use of Unity's physics and rendering systems, the agent is able to gain basic spatial information through a series of rays that are projected from the camera as the origin point. These rays serve two purposes. Firstly, by casting out rays along different sight-lines, the agent gets a rough estimate of what the explorable space and level boundaries are. Secondly, if the rays intersect with a level entity within the agent's POV and is not occluded by any geometry, that entity is marked as visible until it leaves the agent's POV. This information helps the agent form a mental model of the level then make decisions and explore accordingly. As an example of a potential use case, an overly cautious agent may use information on the location of enemy entities to figure out which routes to avoid.

One thing to note is that this isn't a wholly accurate simulation of player vision. This system does not take into account aspects such as colour and in-game lighting. These are elements that affect whether or not a human player is able to see something in a game

level, but are omitted for PathOS. What we do with rays is adequate for our purposes, but it's also possible to expand the sophistication of the agent sight using techniques such as computer vision for future endeavours.

Memory

Another necessary component for simulating humanlike navigation is to have a representation of human memory. How a human player remembers aspects of a level can influence how they choose to explore it, such as whether they remember where key entities are located. In order to do this we gave the agent a simple memory model that was split into spatial memory and entity memory.

Spatial memory is how the agent keeps track of level geometry. This information is stored in the agent's memory map, which is made up of the information gathered from the raycasts. It is also displayed to the user via the tile-based mini-map that shows up in the bottom left of the Unity game view window, so that user are more aware of the agent's perception and process. Additionally, it is also used for the agent's route-planning. While agents use Unity's inbuilt NavMesh system to travel to destinations, this does not take into account contextual information relating to entities that may be along the way. In order to calculate desirable routes with this in mind, the data from the agent's mental map is used with an implementation of the A* algorithm so that the agent can wayfind based off of their motivational properties.

On the other hand, entity memory is how the agent keeps track of level entities and their location. Within this memory the agent remembers what the entity type of each level entity is, and also its position. In order to better mimic the imperfect nature of human memory, this information is not permanent in the agent's memory. Instead the information can be mutated and lost, similar to how humans can lose mental information. The way this functions is based on short-term memory, long-term memory, and memory recall.

Short term memory is when mental information is stored temporarily and in limited

capacity, but before reaching this point information passes through something called iconic memory. For something to pass from iconic memory to short term memory it has to remain in focus for more than a certain amount of milliseconds. To mimic this in PathOS entities are only stored in the agent's short term memory if they are perceived for more than a couple of milliseconds. Another thing to note is that the storage of short term memory is based on the experience level of the agent, similar to how in real life more experienced gamers may be better able to remember pertinent in-game information. Thus the agent's short term memory can vary from three to five entities respectively.

Long term memory is where mental information is stored semi-permanently. For something to be transferred from short term memory to long term memory it has to be in focus for a while, usually through the process of memorization. To mimic this in PathOS entities are transferred to long term memory if they are within an agent's sight for a certain amount of time. If the entity is out of sight before getting transferred to long term memory, it starts to "decay" and will eventually be forgotten. Once in the agent's long-term memory, it is still possible for an entity to be forgotten, but it takes a much longer time.

Lastly, recall is how information from the memory storage is brought to the forefront. For humans this process involves a mental cost, especially with a larger amount of remembered items. This is shown in PathOS through the insertion of extra time to recall information when there are more items stored in memory. Additionally, noise is inserted into the agent's recall of entity locations that are not currently in sight. The noise helps make estimates on where entities are located more imprecise, similar to the imperfect nature of human recall memory.

Decision-Making

During traversal the agent makes decisions—in the form of mathematical calculations––to decide where it wants to go. These calculations primarily consist of the agent looping through available target destinations, and picking the most suitable one for the sake of simulating navigation that is authentic to its motivations.

Target destinations can be of two types, that being entity destinations or exploration destinations. Entity destinations are where the level entities are located. The calculations determine which entity, from the list of potential entities in the agent's memory, the agent will navigate towards. Once that location has been visited, that entity will no longer be considered as an entity that could be targeted for visitation as interactions in PathOS are one-time occurrences. On the other hand, exploration destinations are not tied to a specific entity, rather they can be any point on the traversable level geometry. They are chosen based on the direction of the agent and the farthest point they can reach, and can act as alternatives to entity destinations.

Entities can take nine different types in PathOS, and are based off of existing literature on types of entities in video games. They are also static, meaning that an entity cannot change type during the duration of the simulation. The entities can be goals, hazards, resources, POIs, or NPCs. These types are shown in more detail Table 3.1.

Whether or not the agent goes towards one of these entities, or an exploration target, is based on a scoring function. The calculations of this scoring function take into account the entity type along with the values for each of the agent motivations as defined in the entity motive matrix. These attributes are based on existing literature on player psychology. They can be edited via the normalized personality sliders within the interface, which are shown in greater detail in Table 3.2.

The value assigned to each of these motivational attributes factor into the scoring function. A (customizable) scoring matrix is used to define the relationship between the entities and each attribute. The equation can be summarized by the following:

$$Score = entity\ score + bias$$

The entity score is based on the agent's current location, with the direction to the destination being factored into the calculation. It helps determine the desirability of traveling to this destination.

The bias represents the inherent value of the target destination. For exploration targets

Icon	Type	Description
	Optional Goal	Objective not necessary for completion.
	Mandatory Goal	Object necessary for completion.
	Final Goal	Final objective for the level.
	Enemy Hazard	Entity that acts as a threat to the player via combat.
	Environmental Hazard	Entity that acts as a threat to the player via the environment.
	Achievement Resource	A boon, like a collectible.
	Essential Resource	A boon, like health potions.
	Point-of-Interest (POI)	An entity that does not serve a specific function, other than being an interesting location or object.
	NPC	A non-playable character that the player can interact with.

Table 3.1: Breakdown of the entities

Attribute	**Description**
Curiosity	Represents the player's drive to obtain in-game rewards, drawn to all the goals and Achievement Resources.
Achievement	Represents the player's drive to see all aspects of the game world, drawn to POIs and NPCs.
Completion	Represents the player's drive to complete everything, similar to Achievement, drawn to most entities.
Aggression	Represents the player's drive to dominate through combat, drawn to hazards.
Adrenaline	Represents the player's drive to dominate through environmental factors, drawn to hazards.
Caution	Represents the player's drive to keep themselves safe, drawn to preservation resources.
Efficiency	Represents the player's drive to finish the level as quickly as possible.

Table 3.2: Breakdown of the personality attributes

it's 0, and for entity targets it is the sum of all the scores in the row of that specific entity in the matrix, multiplied by the value of its motivational profile. This bias is scaled by a variable that allows the agent to prioritize entities that are closer to its current location. Once the target destination is determined after comparing scores, the agent begins moving towards that destination. This scoring function is run intermittently, and the agent has the ability to reconsider and backtrack to further mimic player behaviour.

3.2 Reexamining the Original Study

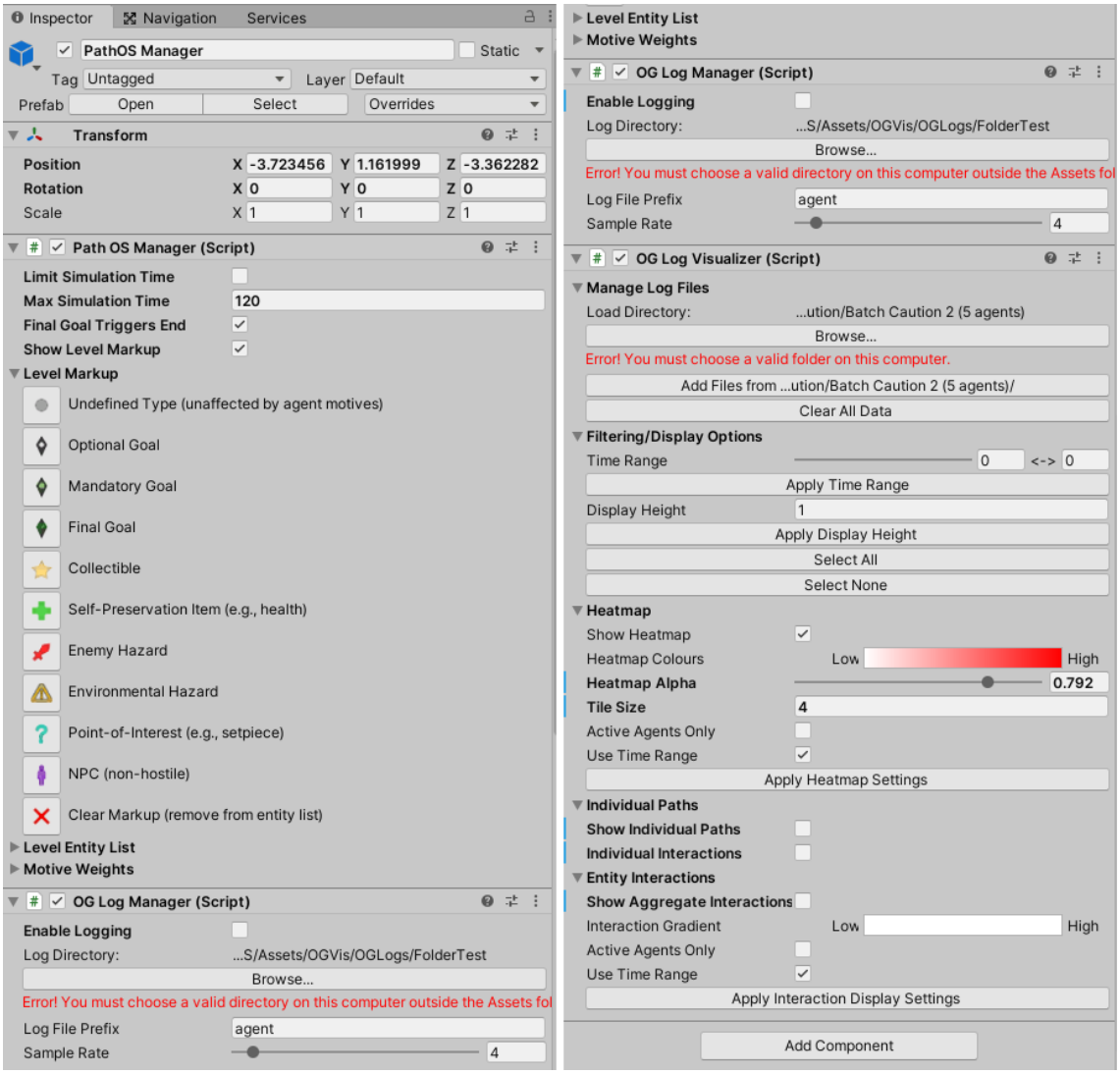


Figure 3.5: The original PathOS Manager inspector

Originally, after PathOS was completed, a study was conducted to evaluate it as a level design aide [105]. The results were promising. Many participants thought the tool was easy to use and useful for both user experience (UX) and quality assurance (QA) purposes. There were also numerous suggestions made from which we created a changelist of the most promising suggestions, including quality of life improvements and additional features. Now that it's time to introduce the next era of PathOS, it came time for us to reexamine this original study in order to glean what information we could to transform PahOS into a suitable expert evaluation tool.

The participants of the original study came from very different backgrounds. There were six industry professionals (P3: Tracking Data Manager, P4: Machine Learning Expert, P5: Data Tracking Manager, P6: Programmer, P8: Programmer, P9: Programmer), two students in GUR (P1: Graduate student, P10: Graduate student), and a consultant (P7: UX Researcher). One of the participants (P2) was not able to complete the study so their data was omitted from analysis. All the participants completed a pre-test questionnaire and interview in order to gauge more information about them, including demographic information. They then had PathOS demoed to them so that they were familiar with its functionality, and were given the manual so that they could be familiar with all of its content. During these initial stages the participants were given their task, in which they had to use the tool to create and playtest three unique levels. They were given a prompt for each one (a level for aggressive players, for example). Once the remaining nine participants completed this task, they were interviewed again to learn more about their experience with the tool. These interviews asked them to explain their created levels and design process, along with their experience using the tool. We recorded and later transcribed these interviews. It was these transcriptions that were used for the analysis process of PathOS+.

After reexamining the transcripts through the context of expert evaluation we noticed some interesting things. Participants made suggestions that were pertinent to not only the general improved usability of the tool, but also expert evaluation. We went through these transcripts and re-coded them separately, and then combined the results. We chose

not to use a reliability test because the goal was not to confirm the same findings. Instead the goal was to identify as many potential ideas as possible that could be useful for the betterment of the tool and expert evaluation. The following describes what we found. As an aside, there were other features integrated for the purposes of expert evaluation, but they are explained in detail in the next section. The rest of this section is just what we could extrapolate from the original study. The following information was also covered in one of our previous publications [106].

One of the issues that was the most prevalent—pointed out by six participants (P1, P3, P6, P7, P8, P9)—was that navigating through the different features of PathOS was inconvenient. The features of PathOS were located in different gameobjects (along with the separate menu option for the batching), so if participants had to make use of a specific feature they had to make sure they clicked the correct option in the inspector, or opened the correct window. This was a little confusing. P7 mentioned the difficulties they had finding features by saying, "*I had a huge trouble getting the markup to open. And even now I still don't... I found it once and now I don't know where it is.... I read through [the Manual] and I still had trouble finding it*". P9 commented on the batch window by saying, "*being able to have [the batch window] in the inspector or something would've been nice. Otherwise it's just floating here, it's always covering stuff*". There were participants who discussed how having a centralized location for all the features would be greatly beneficial. P3 said in regards to the confusion, "*[That's] a Unity hierarchy problem... when you build something like that it does start to get cluttered.... Like the navigation tab for instance, is very useful... So if there was an AI tab for instance, just hit that and it brings up the manager*". Similarly P8 said that, "*Honestly I feel like just having a permanent separate window for PathOS... for everything, would be amazing*". Even though this is not directly related to expert evaluation, expert evaluators would still benefit from a tool that is organized with all the key features in a single centralized area. It would probably help make their workflow more efficient. For that reason, one of my goals was to make a centralized window for all the PathOS features.

There were some other smaller usability features that I implemented to improve the ease

of use for PathOS+. For instance, PathOS came with a timescaling feature that wasn't immediately recognizable to the participant. This meant that participants were missing out on a very useful and time efficient feature, especially for analysis and observation purposes. To exemplify this, P1 said, "*I really wanted a timescale slider, so that I could just set the timescale to 8 and let him go one playthrough*", and similarly P8 said, "*there really needs to be a fast forward tool*", both without realizing that the feature already exists. For that reason I decided that making time-scaling features more prominent would be beneficial for PathOS+.

Another feature that needed usability improvements was the markup brush. In particular, there were two key changes that participants wanted to see. Firstly, while marking up the entities in a level, the participant may need to adjust the camera angle to better position the markup brush or get a better look at the surrounding geometry. The problem is that in doing that (by pressing alt and left-clicking) they would deselect the markup brush, so the selected entity for the brush would disappear. This was an annoyance for participants, in particular for P5, P9, and P10. P10 stated "*The thing I noticed is that because I have to mark, I have to move sometimes. And when I move it ends up deselecting the markup options*". Additionally, some participants (P1, P3, P5, P6, P9) felt that the whole process could be more streamlined. Specifically, they wanted ways of marking multiple entities at once. P6 said, "*I think it'd be interesting… if I could search and then select all of the enemies and click a button to mark them all as an enemy, instead of having to click on them individually. 'Cause the way I was doing this was I was designing the level, and then positioning the camera, clicking on all of them*". On the topic of mass tagging entities, P5 notably stated, "*When I was trying to mark all of these gems here it was pretty time consuming, and any miss like that and now I've marked the ground, and I'm like I gotta undo, and now that I've undo-ed I missed the thing, and I was like oh now I've clicking on something else, I gotta find it again… You can see how manually clicking on smaller objects can get frustrating really quickly*". The general consensus was that the current setup was time-consuming and frustrating, so I decided to explore possibilities of easing this burden on users. While these things aren't directly

related to expert evaluation, they still end up affecting the expert evaluation process. It's possible that an evaluator may have to set up the game level that is to be analyzed, or they may want to experiment with the level markup to analyze different possibilities. Whatever the case may be, streamlining this process and improving its usability means that the evaluator would have a much quicker and easier time getting the level set up for analysis.

I also wanted to explore how to expand the way PathOS simulated games. At this stage the entity interactions were an abstraction. For instance, when the AI interacted with enemies they did not lose any health, so there were no consequences like there would be in a real game. This was something I felt would be important to include, as these types of interactions and consequences could affect player behavior in real-life scenarios. To really emulate humanlike behavior it would be good to have the AI be influenced by such factors as well. This desire was reflected with how participants mentioned wanting more complex interactions, in particular in relation to how resources would factor into agent behavior (P1, P3, P9). For instance, P1 stated, "*I think maybe the tool would benefit from being able to let the user create, like a list of resources… And then visiting certain objectives would lower certain resources*". Similarly P9 said, "*I started putting down some health stuff after fights. And then I realized that there probably isn't too much logic behind the agent either getting the health before a fight or healing back up after a fight. Or for how difficult a fight would be…*" When asked about a risk/reward system, they responded, "*Yeah, I think that could be helpful for it, for at least getting closer to how a more indepth game would fully structure things*". I used this feedback to determine how to add increased complexity to the agent behaviour in PathOS+.

The biggest issues highlighted through this reexamination of the transcripts were that the visualization features had to be improved. This was very evident by how most participants were not aware of the visualization, or that they had troubles using it (P1, P5, P6, P9, P10). I chose to focus on these issues as it directly ties into expert evaluation via the tool's analytical capabilities. Figure 3.5 shows what the old manager inspector looked like, where features like the level markup and the visualizations were located in

the same place. The following are the key visualization-related suggestions that were identified.

Four participants (P1, P6, P9, P10) struggled with the agent logging features, and wanted the process to be more streamlined. Some participants were not initially aware that logging had to be enabled, and an output destination file had to be selected in order to log data. This was described by P10, who said, "*The only thing is I forgot that this was an option, like enabling logging. So at first it wasn't enabled and I'd go to the folder and like why is there nothing in the folder? So maybe having that enabled logging set up by default*". Participants like P1 mentioned how it would be nice if there was a default output file, saying, "*Maybe [setting up the logs] could be streamlined a little bit...having to manually go in and select an output destination for your logs. Like maybe just have a default*". P9 talked about how some of the UI features could be more clear, specifically referencing how logs are loaded in by saying, "*Having to then click [Add files from] for the log. 'Cause I didn't realize at first. I thought this was the [Browse] button had changed to show the scene, and not that it was [Add files from]. It was just a thing that I missed for the first minute or something like that, I was like why is nothing showing up*". By streamlining the process of logging and recording agent playthroughs, I will be making the visualization features more accessible and easy to understand for users.

Some participants talked about how the data that the visualization showed could be improved in terms of clarity and additional features. For instance, three participants (P5, P9, P10) noted that when it came to reviewing the individual paths, it was a little difficult to determine what direction the agents were moving in. A discussion was had with certain participants on whether they'd like to see directional arrows on the individual paths. To this P5 said, "*If people are having more of an open-world environment, then I think yeah it would be more helpful. But if it's more of a linear experience then it's probably not necessary*". P9 said on this issue, "*When you're looking at the interactions with one of the timelines, you can't really tell when that was happening. Which just goes back to the thing of not having time context for the lines or interactions*". This brings us to

the next point, which is that PathOS+ could benefit from displaying more time-sensitive information. In relation to P9's quote, they were asked, "*So do you think it might be useful to say have like on mouseover of a part of the path to maybe show a timestamp or something like that?*" to which P9 responded, "*at least on the interactions... Like you could just put [time stamps] beside the platforms or things like that*". By making it easier to parse the visualization data and glean more information on what the agent was doing, we would be streamlining the analysis process of agent data, thereby making it a more efficient way to make conclusions.

Some other suggestions were in relation to the heatmap functionality, such as P6 who said, "*It might be cool to export the heatmaps as an image or something you can view outside of Unity...Or just move the memory map view to its own editor window and just render it out as a texture each frame, save it over time. But definitely tools to export and import data into different programs... and take your heatmaps and put them in presentations... I think there's a lot there*". Considering that expert evaluators may want to use screenshots of their work as they present their findings, I chose to implement this for PathOS+.

I also determined some other features to add specifically relating to conducting expert evaluations based on existing literature, which are detailed in the next section.

3.3 PathOS+

I began development of PathOS+ soon after analysis was completed. Below is a breakdown of the features and changes that ended up in PathOS+

3.3.1 Centralized Window

Unity Editor Scripting is a way of coding within the engine that allows users to create custom interfaces while offering great flexibility. This was heavily used in order to adjust the existing UI of PathOS+ features while also creating the centralized window (dubbed the "PathOS+ window"). The centralized window went through numerous iterations,

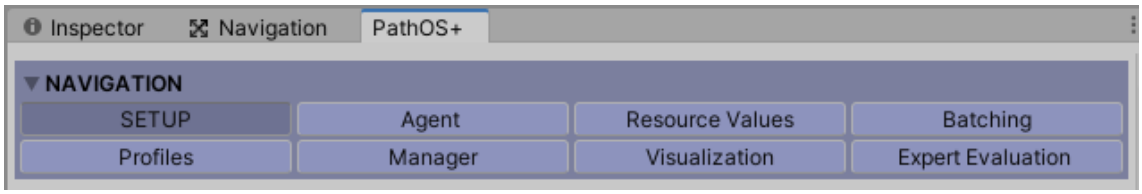


Figure 3.6: The navigation bar located at the top of the centralized window

with what is described here being the iteration in the most recent version of PathOS+. It's also important to note that the user can still choose to use the regular inspector interface for the different PathOS+ components—the PathOS+ window is just there for them to access everything from a convenient centralized location if they so choose.

In order to open the PathOS+ window the user needs to select "Window" at the topmost toolbar, and then "PathOS+". This opens up the window, which can then be docked to any part of the Unity interface. Within the window there is a collapsible navigation bar at the very top (shown in Figure 3.6), within which there are buttons for 8 separate tabs. Each tab has a separate functionality. This section covers the contents within each.

Setup. There are three references that have to be configured in order to use the PathOS+ window properly. These are the agent, manager, and screenshot manager (the latter being a special addition just for PathOS+). These objects are prerequisites for any scene that makes use of PathOS+, and the window references these objects in order to modify and access their values. In the event that the user has multiple of any of those objects (such as multiple agents) they can select which one they want to focus on. If the user was to, say, navigate to the Agent tab without having selected an agent, that tab will only show a warning saying an agent reference is required to use that tab. Once the references have been set, the interface will change to show components relevant to that reference (i.e. setting an agent reference will show a button that lets the user jump to the agent tab). If there is currently no agent, manager, or screenshot manager object already existing in the scene, the user can create those objects in the scene in this tab through the click of a button.

Agent. If the agent reference has been set the user can use this tab to modify the person-

ality values of that particular agent. They can also modify other things such as the agent eye and memory values, the timescale, and the agent's UI renderer. When this tab is selected, it acts the same as having that particular agent selected in the Unity hierarchy. This means that if the user was to run the simulation while in this tab, in the game view they would see that agent's first person view, mental map, and so on.

Resource Values. This tab makes use of the agent reference (mandatory) and the manager reference (optional, just to toggle one specific option). This tab represents a new feature that was incorporated just for PathOS+, which is the enemy and resource values. In order to learn more about this, see Subsection 3.3.3.

Batching. This tab is dedicated to batching, and only requires a reference to an existing agent prefab in the Unity scene if the user wants to use one as a template. It allows the user to run simulations with multiple agents. Here the user can set the timescale, the number of agents, the starting position, the agent prefab which will be used to model the batching agents off of, and the agent motivation values.

Profiles. This tab is dedicated to creating agent personality profiles. It does not require any references. Here the user can specify the personality of a unique profile by modifying motivation values. They can then export the profile to be used, or import existing ones.

Manager. This tab is dedicated to all the managerial functionalities of PathOS+, and requires the manager reference. Here the user can set up the level markup, look through the entity list, specify which entities to ignore (a new feature for PathOS+), and customize the entity motive matrix. While this tab is selected it functions the same as having the manager selected in the Unity object hierarchy, which means that the user will see all the entity icons for the marked objects in the sceneview. This tab can be viewed in Figure 3.10.

Visualization. This tab is dedicated to all the visualization functionalities of PathOS+, and requires the manager reference as the visualization features are tied to the manager (and in the past iteration of the tool they were also located in the same UI block as the

manager). This is where the user can log agent playthroughs, and then load those logs back in as data that can be depicted as heatmaps, individual paths, and entity interactions. All the properties to modify these elements can be found here, and the adjustments made are explained in more detail in Subsection 3.3.4. This tab can be viewed in Figure 3.10.

Expert Evaluation. This tab is a wholly new feature that contains the unique expert evaluation functions of PathOS+. This is where the user can conduct their evaluations and import or export them. This is explained in more detail in Subsection 3.3.2.

Overall this window helps solve a big issue that was present with the original PathOS, in which features were difficult to find, and the tool was difficult to navigate due to how it was organized. This window takes all the key elements of PathOS+ and breaks them down into tabs that are convenient to access, thereby reducing a lot of the burden on the user.

3.3.2 Expert Evaluation Tab

One of the tabs in the centralized window was dedicated specifically to expert evaluation. I could have decided to omit this tab, and make it so that the user could be left to their own devices to document their evaluations. However, by reviewing literature surrounding expert evaluations I determined that there is a need for expert evaluation tools that are digital and flexible. Hvannberg et al. did a study where they had a group of researchers do an expert evaluation via traditional methods, in contrast to a group that did the same thing with a digital tool [69]. The latter group were not only more efficient and happier with that evaluation method, but they seemed to identify more issues. This could be due to the increased efficiency. The paper also notes that it would be beneficial for a tool to allow users to do the expert evaluation at exactly the same place that the game is located, so that there is no hassle related to jumping between different windows and applications. The expert evaluation tab in PathOS+ was developed to do exactly that.

The tab was created by leveraging Unity's editor scripting system. Within this tab users

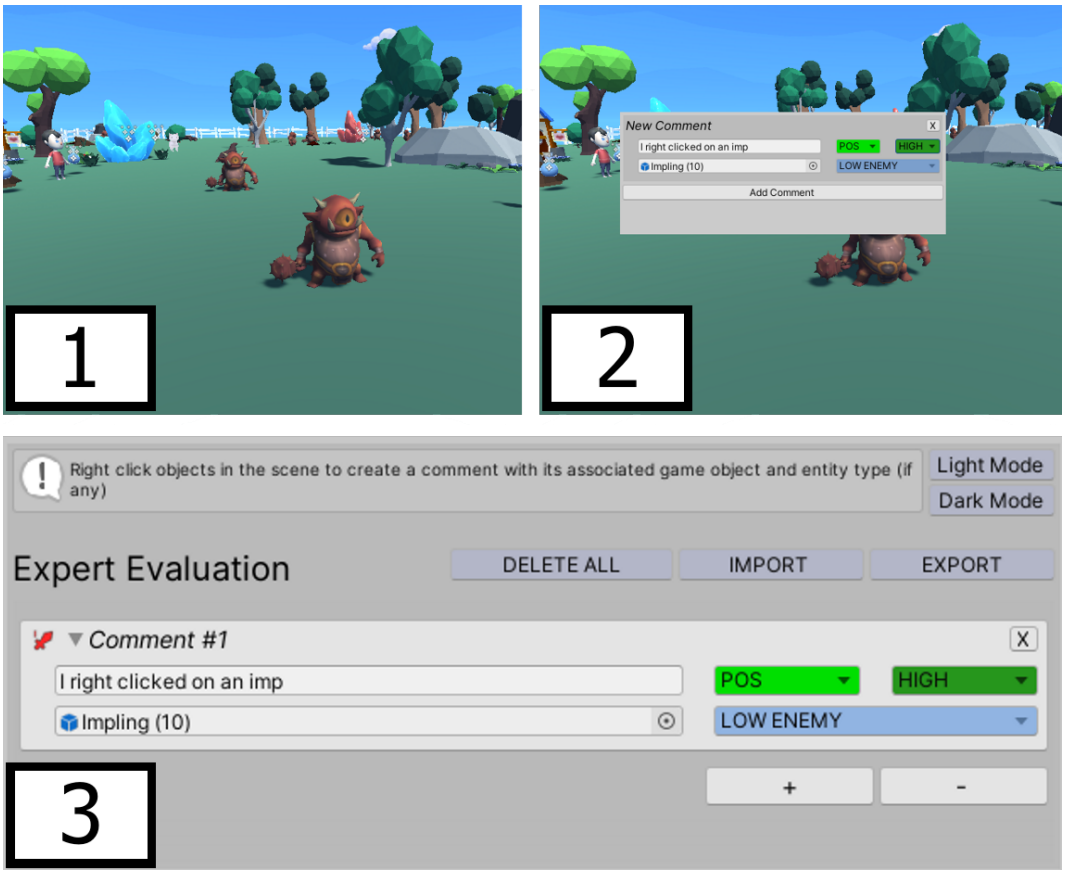


Figure 3.7: Step-by-step breakdown of the right-clicking comment feature

can create "comments" representing any points or observations they may have about the game. These comments have a text area for the description of the issues, along with two colour-coded dropdowns. One of the dropdowns is for determining if the issue is positive, negative, or neutral. The other dropdown is for determining the severity of the issues (if applicable) by giving it a low, medium, or high rating. Comments can also be deleted or edited at any time. Additionally, comments can also have a gameobject and entity type attached to them if it is a comment relating specifically to an in-game entity. Users can right-click an entity in the Unity game scene and a window will pop up letting them fill out a comment that has that specific gameobject and entity type applied. Once they're done and they select "Add Comment" it gets added to their list of comments within the expert evaluation tab. This whole process is shown in 3.7. All of these comments are saved, so the user can close and reopen the Unity project or switch Unity scenes without worries. This system fulfills the purpose of allowing users to write

in-depth observations about the game within the application.

I want to quickly note that the evaluation tab was not always structured this way. Previously I experimented with having the evaluation tab load in established heuristic guidelines (such as PLAY heuristics [57]) and then allowing the user to write comments for each of the categories (while still allowing them to assign a severity). I decided to go with a more open-ended approach rather than using specific heuristic sets. This allows the user greater flexibility when it comes to their evaluations, and helps avoid the issue of heuristics being too specific to be generalizable.

This tab also comes with an import and export system. Evaluations can be exported into .CSV files that organize the information. At the top of the .CSV file is a table that organizes the comments (which are numbered) by entity type to severity. It then lists all the comments with their corresponding number and all the related information attached to it. These .CSV files can then be imported into a PathOS+ project. When the data is loaded in, if there are any gameobjects referenced in that scene, they are found and attached to the relevant comments. This feature was integrated in order to allow evaluators to share their findings and back them up with ease.

Lastly, this tab is tied into the tool's unique screenshot system. The ability to take screenshots is primarily in the Setup tab—once the screenshot manager is selected, the user can write down the name of the folder the screenshot will be saved to, the name of the screenshot itself, and they can press a button to take a screenshot at any time. This uses an in-scene camera to take a birds-eye view image of the level in question. In the Expert Evaluation tab, if the screenshot manager is configured, the user will be provided with a view of the map they are on. They can refresh this image to reflect any changes that happened to the level. If the user were to enable heatmaps or anything of that nature, they would be shown in this map view overlayed on the level geometry. When the user exports their evaluation a .PNG is exported along with it. The .PNG depicts what is shown in the map view, including the visualization features that are shown within it. The reason why I chose to implement this feature is if a user wants to export images of the level with the visualization features for presentations to the development team.

3.3.3 Combat and Resource System

My way of adding further sophistication to the AI was by introducing a simple simulation of combat and resources. In order to do this, I had to make some changes to the fundamentals of the tool, which is why in PathOS+ there are new entity types and changed versions of the old ones. I created different levels of hazard entities to represent different difficulty levels, and different levels of preservation resources to represent different healing amounts. This is shown in further detail in Table 3.3.

Table 3.3: Breakdown of new entity types

Old Entity	Icon	New Entity	Description
Hazard Enemies		Enemy Hazard (Low)	Low difficulty enemy
		Enemy Hazard (Med)	Medium difficulty enemy
		Enemy Hazard (High)	High difficulty enemy
		Enemy Hazard (Boss)	Boss enemy, also counts as a goal
Self-Preservation		Self-Preservation (Low)	Low amount of healing
		Self-Preservation (Med)	Medium amount of healing
		Self-Preservation (High)	High amount of healing

There is a tab in the PathOS+ window that is dedicated to this resource system. It shows all of these entities—along with the environmental hazard entity—with sliders next to them. These sliders allow the user to modify the range of how much damage the different enemy types do, or how much health the resources heal. I used ranges instead of specific values because that seems more realistic to how games function, in which players can take varying amounts of damage (or heal various amounts), even in encounters that are

more or less the same. An overview of this tab is shown in Figure 3.8.

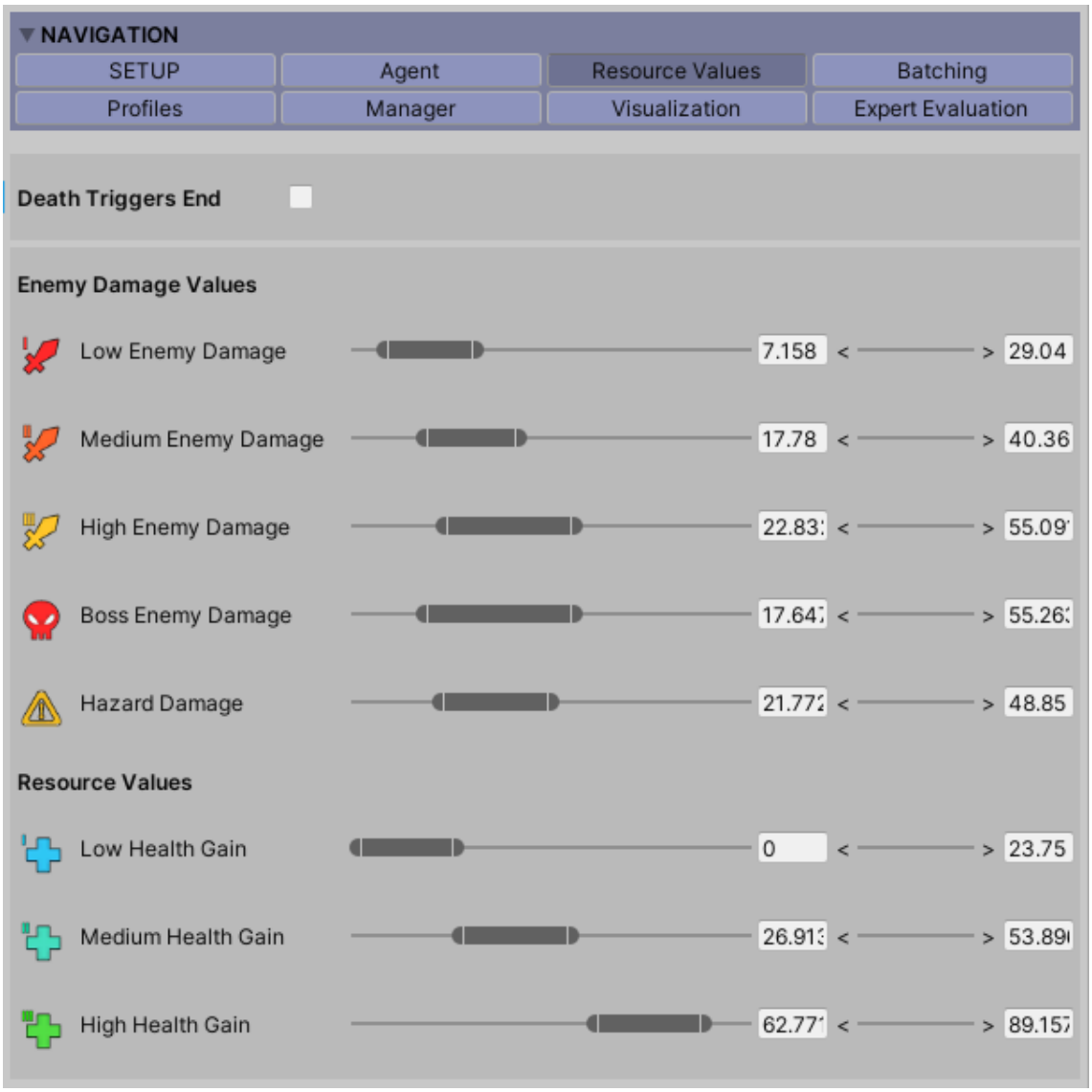


Figure 3.8: Resource tab for combat and potions

The way it works is that the agent has a certain amount dedicated to their health, as is now shown in the top right of the game view during simulations. If the agent chooses an enemy as their target destination and heads towards it, once they interact with it it runs a very basic combat simulation. Based on the damage range of that enemy type, and on the agent's experience value (agents with higher experience take less damage, akin to real life), a random value is generated that fits within those ranges. That amount is then subtracted from the agent's health value. Similarly, when the agent interacts with

a resource, based on what resource type it is and its healing range, that amount will be added to the agent's health. The health never goes above 100, and consequently if it reaches zero or less then that can count as a "death" and an end to the simulation if the user so chooses. A toggle to enable this is in the Resource Values tab.

The agent's behavior is affected by how much health they have left. During the simulation the agent has two matrix representations of their motivation attributes. The first is the default one, the one the user has determined in the interface before running the simulation. The second is one that gets modified during runtime. Influenced by how "aggressive" and "cautious" the agent is to start off with, as their health goes down the agent's aggressive values go down by a certain amount, and the cautious value goes up by a certain amount. This gives the agent the ability to possibly avoid enemies if their health is getting too low, but if they're really aggressive by default it's less likely for that to be the case.

3.3.4 Streamlined Visualizations

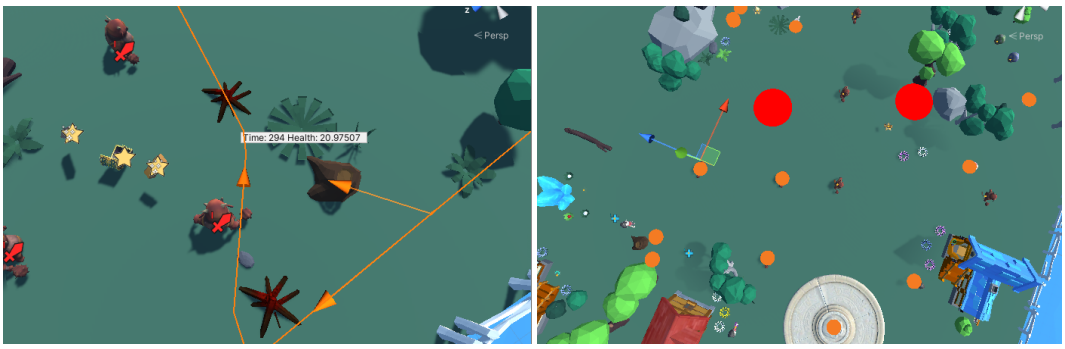


Figure 3.9: The changes made to individual paths and entity interactions

The visualization aspects of PathOS+ are really important to facilitate analysis, thus it was crucial to improve their ease of use as much as possible. To do so I had to study UI design and understand how to make interfaces that are easy to navigate. I chose to do this before adding new features because, regardless of how effective the new features are, users won't reap their benefits if they're struggling to use it.

One of the things I did was decouple the visualization features in the UI from the man-

ager features. Previously these were all part of the same inspector window, which may have added to the confusion. For PathOS+ the visualization and manager features are separated into two different tabs. Secondly, based on some research I did on effective UI design [107], I chose to only make menu elements visible when they're actually usable. As an example, the options for which folder to save the logs to are not shown unless logging itself is enabled. This reduces information overload on the user, and makes it so that they understand which UI options are tied to what. The UI information is also grouped by relevance. Previously the heatmap, individual path, and entity interaction information were all shown vertically all at once. In PathOS+ they are separated by subtabs, so that it reduces clutter and users will only see the variables specific to that visualization feature (as shown in Figure **??**). Many UI elements can be collapsed in order to further reduce noise in the interface. Different blocks of the UI also have different background colors, to further segment chunks of information. Lastly, the UI warnings that were present in the previous iteration of the tool were just red text. They were made more obvious and distinctive so that users get a better indication if something is missing.

After the conclusion of these UI adjustments I was able to focus on new additions. I changed the individual paths so that directional arrows could be enabled, showing which direction the agent was moving in at any given time. Additionally, if the user were to hover over the individual path at any time, they would see the specific time stamp for that moment, along with how much health the agent had at that time. This is to help give further context as to what was going on at that specific point of the agent playthrough. The entity interactions were also changed so that the labels for the different entities are more clear. The new agent paths and entity interactions are shown in Figure 3.9. Lastly, in the previous iteration of the tool, users could make changes to features like what time range the data is depicting, the heatmap alpha, and so on. After creating these changes the user had to click a button for them to apply. In PathOS+ this was changed so that all the user would have to do is make those changes, and it would apply to the visualizations in real time.

3.3.5 Quality of life improvements

Other than improving the organization of the tool through the use of the centralized window, I tried polishing the other features based on the feedback participants gave from the original study. I started by focusing on the issues with the level markup feature. PathOS+ now has the ability to mass tag gameobjects as a certain entity by clicking and dragging the cursor while they're in the level markup mode. I disabled the issue where the markup tool would deselect the chosen entity type if the user were to try and rotate the camera. Lastly, I introduced an ignored entity list to the tool. Gameobjects added to this list gets ignored by the markup brush. This was added because participants complained about how they would accidentally mark irrelevant things as entities, such as the ground.

I made UI options like the timescaling more prominent, and added it to panels where it previously wasn't before. For instance, previously the timescale could only be adjusted in the batching window for batch tests. I added the timescale feature to the agent customization area, so that even regular simulations can be made to be up to eight times the original speed. Other than general UI improvements and tweaks, there were also numerous bug fixes and behavior adjustments made to the tool. For instance, previously, if the agent walked through an entity, even if they did not intend to interact with that entity, it gets marked as visited and is no longer considered for the agent's decision-making. I changed it so that the agent has to intend to visit that entity in order for it to be marked off, like how in games players can run right past something without directly interacting with it (such as combat encounters).

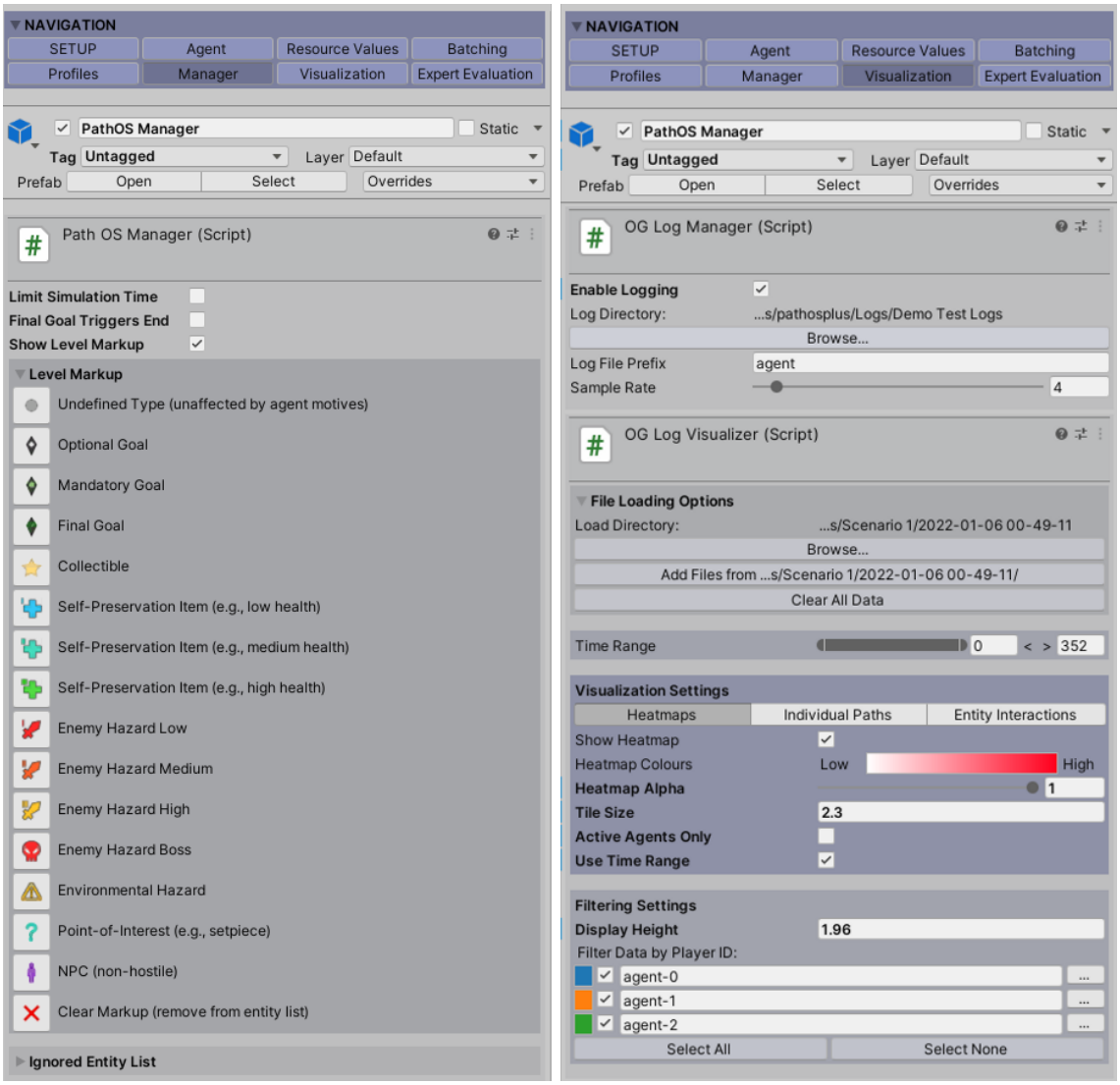


Figure 3.10: In PathOS+, the manager and visualization are kept in separate tabs

4 Testing and Evaluation

In order to assess how effective a piece of software is, user tests would have to be run with the target audience. Only by looking at how the target audience interacts with the software can we say whether it succeeds in its intentions. In order to evaluate PathOS+, we decided to run a user study with participants who have had user research experience. The goal was to get participants to try out the tool and use it for an evaluation task, after which they would give us feedback on how it fared and what improvements could be made.

4.1 User Study

The user study was divided into three parts. The pre-interview, the take home expert evaluation task, and the post interview. The expert evaluation task was in the form of three premade levels in Unity, each with their own design flaws. We hoped the study would help us answer the following:

- Are the participants able to identify the design flaws within each level?
- Is the AI instrumental in helping them discover the flaws?

Design flaws in this case are issues that can hamper the enjoyment or progression of a hypothetical player. In particular, issues that I am able to replicate with PathOS+ and the limited resources I had at my disposal. It is important to note that PathOS+ can do more than identify flaws–it can be used to examine player behaviour and movement patterns in general, whether it is constituted as flaws or not. It is solely for the purpose of this study that we chose to focus on what we deemed as objective flaws within these scenarios' design. I also went with this approach because it is directly tied to the weaknesses of the

expert evaluation approach, in that due to its subjectivity important issues may be missed or misrepresented. By running a study in which the focus is to see if PathOS+ can help in the identification with design-related issues, it may help shed light on whether AI playtesting data may be beneficial for expert evaluation.

The following describes each aspect of the study in detail.

4.1.1 Participants

In order to generate interest in the study, I did a few presentations on AI in playtesting and the purpose of PathOS+. One presentation was to the user research lab and grad students at Ontario Tech University. The other presentations were at the company Ubisoft Toronto, and they were to the user research and quality assurance teams. Alongside the presentation a link was provided for signing up to be a study participant. The presentation garnered some interest, and resulted in some user researchers signing up for the test. The rest of the participants were interested colleagues who have heard about the project and had some user research experience in the past. Table 4.1 goes over each participant in detail.

4.1.2 Pre-Interview Session

Before this interview, all the participants would receive an email containing a link to the GitHub page where the project was located, along with a consent form for them to fill out. This email also had a link to the Google Meet room, which the participant would have scheduled via a Calendly link (a website for booking meetings).

The purpose of this interview was to become more familiar with the participants and their experience with both game development and user experience. After the interview questions were asked, the participant would be demoed the Unity project and given a brief rundown on how it works. This was partially to refresh the memories of participants who had seen the PathOS+ presentation a while ago, and also to help those that had less familiarity with Unity. The following list shows the planned order of events for the

Table 4.1: List of participants, their current occupation, previous UX experience, and previous Unity experience

ID	Current Occupation	UX Experience	Unity Experience
P1	UX Coordinator	6-7 months	Experienced
P2	UX Grad Student	6 years	Minimal Experience
P3	Lead Designer, GUR instructor	3 years	Experienced
P4	Lead XR Developer (VR/AR)	1 year	Experienced
P5	User Research Moderator	1-2 years	Experienced
P6	User Research Moderator	3 years	No Experience
P7	NA (WITHDRAWN)		
P8	User Research Moderator	4.5 years	Minimal Experience
P9	UX Grad Student	2 years	Experienced

demo:

1. The participant would be shown how to open the project in Unity
2. They would be shown where the levels were located, and how to open them
3. They would be shown how they could run a simulation (by pressing play)
4. They would be shown how to open the PathOS+ window
5. They would be given a brief rundown on the functionality of each tab, along with the setup therein
6. While explaining the Expert Evaluation tab, they would be given their task, and shown where to export their evaluations
7. They'd have a chance to ask questions
8. Then they'd be shown a live demo of how to run batch tests, and load in visualizations

9. They would be warned about some known bugs

The participant would also be notified that they'd receive a manual and handout in an email sent to them at the conclusion of the initial interview. We'd book our subsequent interview, approximately a week from the first one (with some leeway due to the busy schedules of some of the participants). They'd be given a chance to ask more questions, after which the meeting would be ended.

4.1.3 Expert Evaluation Task

For this task, I chose to base the three scenarios off of existing levels. That way I could incorporate real design problems found in commercial games and see if the users are able to identify them as a form of comparison and validation. In order to find suitable levels, I searched through articles and reviews online. The reason why I chose to do this was because reviews would let me see an honest look of what the playerbase struggled with in these games, and what issues the developers missed (or were unable to solve) before release.

The scenarios were made as three separate Unity scenes. They were built with free assets downloaded off the Unity asset store. The following sections describe which games I chose, and the ensuing levels created off of those games.

Volcano, Far Cry (Crytek, Ubisoft)

Figure 4.1: Far Cry Volcano [108] versus Scenario 1

Far Cry (Crytek, 2004) is the first installment in a series that has spanned over 6 main-

line entries along with numerous spinoffs. The games are first-person shooters in which the player goes to (oftentimes tropical) locales in order to take down a host of different enemies, usually with a charismatic villain mastermind to contend with. While this series is beloved by its diehard fans, certain levels have gotten attention for being difficult. The Volcano level at the end of the first *Far Cry* in particular.

In a GamesRadar article titled "Remember these 11 frustrating video game levels and try not to smash your controller" [109] Ryan Taljonick and Connor Sheridan say in regards to this level: "Armed to the teeth, you step into the rim of an active volcano to fight an army of rocket spamming Hulks (which are stupidly hard to kill and are entirely out of place)... if you need to heal up, you can grab the level's only health pack—which, by the way, is on the opposite end of the arena." This gives the impression that the level is brutally difficult.

Similar sentiments can be seen across message boards dedicated to this game. In a Google group titled "Farcry the Volcano NO WAY TO DO THIS MISSION." [110] multiple players bemoan how difficult the level is, and all the strategies they came up with to deal with the level. A user named B. Jones specifically states "...this is probably the single hardest part of the game." A common reason these complaints were brought up was because the enemies did too much damage.

From these reactions, there are a few key takeaways that can be extrapolated. I decided to boil it down to two things that can be easily translated to a Unity PathOS+ level:

- The enemies are too strong, they do a lot of damage and it's very easy to die
- There is only one health pack in the level, making it even harder

Based on user comments each level also has a miscellaneous design problem. Some users in a message board were discussing how they couldn't find certain objects that could give them a much needed advantage in the level. This obscure tidbit inspired this design issue:

- The end goal is partially obscured, making it difficult for the player to find

The comparison between Scenario 1 and the original *Far Cry* level is shown in Figure 4.1. The final layout of Scenario 1 and its entities are shown in Figure 4.2.

Figure 4.2: Scenario 1 Overview

In order to create a scenario that emulates the aesthetic and style of the original level, I searched the Unity store for free assets that could be used. Keywords such as "Volcano" and "Desert" were used as search terms. When making the scenario, the elements within it were abstractions of the elements in the original level. For instance, there is a giant metallic structure in the middle of the level and mountains on all sides. The area where the AI agent spawns is a high tech building, like what players spawn in in the original game.

To mimic the difficulty of the original level this scenario has enemies spread throughout. They were given the highest enemy tier (tier 3), meaning they did the most damage for regular enemies. Care was also taken to make sure there were too many of these enemies in the level, and they were placed at frequent intervals. To compound the difficulty, only one health pack was available in the level, as mentioned by players as one of the downsides of the Volcano level. Lastly, in order to make the final goal difficult to find, it was obscured behind one of the metal legs that were part of the big structure.

Blighttown, Dark Souls (FromSoftware, Namco Bandai Games)

Dark Souls (FromSoftware, 2011) is a game that—along with its sequels—are known for its difficulty. Even the series it is a spiritual successor of, *Demon Souls* (FromSoftware), is thought of as being brutally difficult. While the games are lauded for their

Figure 4.3: Dark Souls Blighttown [111] versus Scenario 2

environments and challenge, some aspects of it are a little too brutal for some players. Blighttown, a level in the first game, is one that has garnered particular scorn. It is known for being filled with tough enemies and lots of poison.

In a Polygon article titled "Dark Souls Remastered guide: Blighttown Map" [112] Jeffrey Parkin says, "Dark Souls Remastered's Blighttown is all kinds of awful. The enemies are equal parts dangerous, aggressive, and annoying. The ones that aren't breathing fire are probably trying to poison you. And it's not just the enemies — the design of Blighttown often feels like it's actively trying to kill you. It's full of narrow walkways, blind corners, and precipitous drops. Once you're past that, you'll enter a poisonous swamp." This quote depicts a level that is perceived as difficult due to the sheer number of enemies and poison.

In a Reddit thread titled "What's so difficult about Blight Town?" [113] Lord_M_G_Albo says, "Blighttown is an horror if you don't have the Master Key or don't know about the shortcut by New Londo. Without it, it's a long vertical area with poison and toxicity everywhere. There is many enemies who make toxicity damage, but they rarely drops the moss who cure it." In that same thread kevinbr1 says, "The poison guys. And if you don't bring any poison cure you are [practically] screwed." These comments indicate that one of the biggest problems with the level were the enemies and the poison damage.

With these critiques in mind, these are the two issues I decided to translate to a Unity PathOS+ level:

- There is too much poison
- The enemies and combat is too tough

In the *Dark Souls* games there are instances of bonfires—designated healing/saving locations—being located near the entrance of an area. This does not serve as an issue in and of itself because these bonfires can be used multiple times. On the other hand, in PathOS+ the health potions are single use. I decided to take advantage of this in order to create a miscellaneous design issue:

- There is a high level healing item right next to where the player spawns, where they risk using it before they take any damage.

The comparison between Scenario 2 and the original *Dark Souls* level is shown in Figure 4.3. The final layout of Scenario 2 and its entities are shown in Figure 4.4.

Figure 4.4: Scenario 2 Overview

To replicate the dark and brooding aesthetic of the original game I searched the Unity Store for free assets that could be used. Keywords like "Medieval" and "Skeleton" were used to find appropriate models. The level is filled with medieval buildings and twisting paths that give a tight and claustrophobic feel. There are also different kinds of enemies in this level to represent the different enemies and bosses in the original Blighttown. For instance, skeletons were used as regular enemies, mushrooms were used for poison enemies, and giant spiders were used as the bosses (to represent a notorious spider boss from Blighttown).

To replicate the difficult combat the level was filled to the brim with enemies. They

all did different amounts of damage. In lowest to highest damage: the poison enemies were given the Environment Hazard tag, the regular enemies were given the Enemy Hazard (Low) tag, and the boss enemies were given the Enemy Hazard (Boss) tag. While *Dark Souls* is difficult, it's also considered fair, so health packs were placed at regular intervals.

As of now, PathOS+ has no accurate simulation for poison. Thus, translating the poison to a Unity PathOS+ level required some creativity. I decided to fill the level with dozens of the poison enemies that did a small amount of damage each. Since the agent will be interacting with them often, it can simulate what poison damage is like—small increments of damage over a consistent timeframe.

Lastly, I placed a health potion right next to where the agent spawns in the level. This was given the entity tag Self-Preservation Item (High), meaning it does the highest amount of healing. This is a big waste of an important resource in a level that is intended to be difficult.

The Library, Halo (Bungie, Microsoft)

Figure 4.5: Halo the Library [114] versus Scenario 3

Halo: Combat Evolved (Bungie, 2001) was one of the XBox's launch titles. It's considered to be one of the biggest influences for the first person shooter genre, with some going so far as to proclaim it as one of the best games of all time. With that being said, not all the levels in this game are wildly beloved. The Library level in particular is infamous.

In a WhatCulture article titled “13 Terrible Levels In Otherwise Awesome Video Games” [115] Jack Pooley writes, “There’s literally nothing more to [Halo’s library level] than fighting the same enemies again and again as you ascend a building. It’s pure lazy, copy-paste nonsense that drags on far, far too long and temporarily derails an otherwise magnificent game.” Similarly, in a Reddit thread titled “[Halo CE] “The Library” Campaign Mission is Awful” [116] Penrose4Real says, “From a level design perspective, it’s horribly lazy and has the same room layouts copied and pasted EVERYWHERE. Combine that with the same 4 repetitive types of Flood, and you get a level that’s nothing but filler. It’s 25+ minutes of going through the same tunnels or rooms with the exact same layout.” All of these comments indicate that the level is overly repetitive.

From this, I was able to determine two factors that could be easily translated into design flaws for a Unity PathOS+ level:

- The levels are really repetitive and copy paste
- The level itself is too long, adding to the tedium

Another impression I got from the comments was that it is disorienting to navigate the Library level. For that reason I decided to add this miscellaneous design issue:

- Due to the repetitive nature, it’s very hard to navigate and easy to get lost.

The comparison between Scenario 3 and the original *Halo* level is shown in Figure 4.5. The final layout of Scenario 3 and its entities are shown in Figure 4.6.

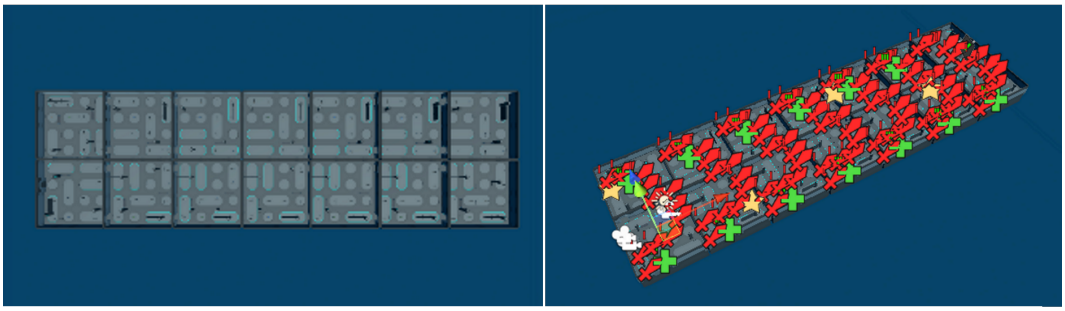


Figure 4.6: Scenario 3 Overview

The original game had a very distinct sci-fi aesthetic. In order to replicate that I went to the Unity asset store and looked for free assets, using keywords such as "Sci-fi" and "Space". I found free models that allowed me to create sci-fi rooms filled with fluorescent lights and astronaut enemies (similar aesthetically to *Halo* characters).

The key aspect to this scenario was to make it very repetitive. In order to do so, I started by creating two rooms. These two rooms looked very similar, and I copy-pasted them over and over again to create the full level. These rooms were uniformly laid out in a straight line. On top of that, the entities within each room were also positioned the same way, and were given the same entity tags. Each room has the same number of Enemy Hazard (Low) enemies (which means they did a low amount of damage) along with a Collectible tag and Self-Preservation Item (High) health potion. The combat in this level is not too difficult, in order to compound the repetitive and monotonous aspects of it. All of these aspects also makes the layout disorienting, and it's difficult to tell where the player is at any point of the level since it's all the same.

4.1.4 Post-Interview Session

The second interview is intended to be a way to go over the evaluations and also hear feedback about the tool itself. The interview is started off by asking the participant to go over their evaluations in their own words. At this point I already saw the evaluation spreadsheets in the email sent to me, but it's helpful hearing the participant verbalize their findings, especially if they're more verbose orally.

After hearing the overview of their findings, the participant is asked about what their evaluation process is like. The key is to see if they used the AI tests as a part of their analysis. They are then asked questions about whether the AI changed any predisposed assumptions they had, and then if any of their evaluations would not have been possible had it not been for the AI. They are then asked about the features of the tool itself (if they felt limited by it, what features they used often), and then there's a question specifically asking them if they think the tool is suitable for expert evaluation. All of these questions help to determine how effective the tool was for the evaluations.

4.1.5 Analysis

An AI tool called OtterAI was used to transcribe the interviews, with some minor edits being made in areas where the AI did not properly interpret the text. The analysis of the interview data was conducted by a research assistant and I. We both examined the transcripts separately using a thematic analysis method and coded our findings. During the coding process we did not use predefined codes, rather we took on more of an exploratory approach. The primary goal was to identify firstmost how many of the issues the players noted per scene, in order to evaluate how PathOS+ performed as an evaluation tool. This was done by examining not just the interview data, but also the evaluation files the participants sent us. The next was to identify trends, whether it be among general observations of the levels, or comments about features about the tool. This was to supplement our insights and see if there were any other takeaways we could have about the tool.

Once we were done we has a meeting to discuss our findings and any differences we might have had. Our findings were then combined and we tallied our codes, from which we were able to extrapolate some conclusions. The next section describes the results of this analysis, whereas the next chapter covers our interpretation of the data.

4.2 Results

Table 4.2 covers which design flaws were identified by which participants. Table 4.3 covers the general insights gained from interview questions.

4.2.1 Scenarios

For the first scenario, participants (P1, P3, P4, P5, P6, P8, P9) noted how difficult the combat was. As stated by P8, "*Agents ended up dead pretty quickly... I did notice that they were losing a great amount of health with each encounter*". Participants (P1, P2, P4, P5) mentioned that it was problematic that there was only one health pack in the level. Describing this concern, P5 explained that "*Having only the one health pack in this level*

Table 4.2: Scenario, intended issues, and players that mentioned the intended issue

Level	Issues	Participants identified
Scenario 1	Enemies are too strong	7/8 (P1, P3, P4, P5, P6, P8, P9)
	Only one health pack in the level	4/8 (P1, P2, P4, P5)
	End goal is obscured	5/8 (P2, P3, P4, P5, P8)
Scenario 2	Too much "poison"	3/8 (P2, P3, P6)
	The combat is too tough	2/8 (P1, P5)
	Big health pack right at entrance	4/8 (P1, P2, P4, P5)
Scenario 3	The levels are repetitive	7/8 (P1, P2, P3, P4, P5, P6, P8)
	The level is too long	3/8 (P1, P3, P5)
	Hard to navigate	6/8 (P1, P3, P4, P5, P6, P8)

could prove to be a problem since there are so many enemies". Participants (P2, P3, P4, P5, P8) also pointed out that the end goal was too obscured, with P2 saying "*In order for player to get a clue of what is the final goal/indicating clear affordances are needed*". There was also an unintentional problem that participants (P1, P3, P4, P5, P9) noted, which is that the agent had a hard time escaping the beginning indoor area. P4 describes that "*[the AI] would get stuck in the hallways and not traverse outside of it*". This area was also considered a problem for the placement of elite enemies with P5 stating that "*if you're starting in the indoor area, there's no way to exit it without really encountering an enemy*".

For the second scenario the lowest number of intended issues were detected. Participants (P2, P3, P6) mentioned there was too much poison, with P3 saying that "*there are too many environmental hazards that the agents will just want to avoid at all costs*". Two participants (P1, P5) said that the combat was too tough, with P1 saying, "*The bosses in the second level [are clustered towards one area]. Perhaps it would be difficult for a player to get through*". Participants (P1, P2, P4, P5) pointed out that it was problematic that the big health was right next to the entrance. P5 said "*There's like... the highest level three healing item at the start... I found that to be kind of weird that there would be such*

a high healing item so close to the start of the level". There were some miscellaneous observations, such as how agents stuck to navigating the left side of the level (P4, P5, P6). As stated by P4, "*I found for the most part, they stayed on the left side, I would say a good 85% went left instead of right. And I found that if they did go left, they never really went back to check out the right... So there was very much a clear strategy there that just they had no interest in the right path.*" A possible explanation was given by P5, who said, "*If you go in the right path, there's more healing items and if you take the left side there's more enemies like bosses. So I feel like maybe a way to balance that out could be like having more hazards on one side versus the other*".

Participants had mixed opinions in regards to the combat of Scenario 2, with P1 identifying that "*the bosses in the second level [are clustered towards one area]. Perhaps it would be difficult for a player to get through*", while P4 commented that "*combat seems well balanced*". While participants had different opinions on whether the combat difficulty was appropriate or overly challenging, they generally liked the variety of enemies present in the level. P1 stated that "*One thing is, that's definitely really nice in this map is that there is a lot of enemy varieties*". Interestingly, P4 said, "*Yeah, I thought the second level... based on the mass amount of enemies in there... I thought everyone was just gonna die. [The agents] were okay, if anything, if I had to choose which one had the best balance for combat, it was definitely the second level...*". This contrast between what the participant expected in regards to how the combat was balanced based on their initial impression, and what they observed from testing with PathOS+, highlights one of the tool's core strengths. Through the use of PathOS+ evaluators have a way to test assumptions that may challenge their intentions.

For the third scenario, participants (P1, P2, P3, P4, P5, P6, P8) pointed out that the level was repetitive. The repetitiveness included the environment, enemy types, and enemy layouts—which was all intended. This is shown with how P6 said, "*I thought the rooms themselves… was a little identical*", P5 said, "*As well as all the enemies in this level are the same difficulty. So there's no variety, and it kind of would get boring, fighting the same enemy over and over again*", and P2 said, "*...all of the enemies were the same. And*

it is kind of monotonous distribution". Participants (P1, P3, P5) said that Scenario 3 was too long, with P3 stating that "*The goal is too far away*". Lastly, participants (P1, P3, P5, P6, P8) mentioned that it was hard to navigate. P1 said, "*there's a chance that players can easily get lost within because they don't really have a sense of direction to know where they're supposed to go*". As P4 explained, "*The issues all kind of stemmed from the same problem in which there wasn't a lot of variety*". There were other unintended issues brought up as well, such as how this scenario did not provide much challenge with P4 saying that "*there didn't seem to be a lot of challenge, either to it. None of them came close to dying, ever*". Participants (P1, P4, P8) attributed this to a number of factors including, as P1 stated, how "*The enemies that are on this map, they're all level one enemies...There's a high level preservation item in each room. But I think that might make it too easy for players...*".

PathOS+ appeared to have directly aided in the identification of some core issues. This was especially shown with Scenario 3, as participants stated that they only noticed the issues of this scenario by watching the agents traverse through it. P6 said, "*The AI and his exploration of [Scenario 3] helped me to determine that there's actually far too many doors at this level, because it's so easy to get in and out of every single door*". In a similar vein P5 stated, "*[In Scenario 3]... I probably wasn't thinking... this is also uniform... I'm like, there's actually just a simple direct path from the start all the way till the end ...But once I saw the agents moving around, I saw them just going between the same rooms ... And that made me think, the uniformity of these levels can actually prove to be an issue*". Watching the AI resulted in these participants realizing that the exploration in Scenario 3 was difficult.

4.2.2 General

Other than the identification of intended issues, examining the results allowed us to see how participants used PathOS+ for the evaluation process. Based on the feedback from the post-interviews we determined that participants used PathOS+ for gaining insights about the scenarios.

Table 4.3: High level overview of general insights from results

Insight	**Participants**
Started evaluations by watching specific agents	P2, P3, P4, P6, P8
Focused on qualitative data	P1, P2, P6, P9
Focused on quantitative data	P3, P5, P8
Mixed qualitative and quantitative	P4
Agents challenged perspective	P1, P2, P4, P5, P6, P8, P9
Thought it was cumbersome to switch between agents	P1, P5, P6, P8
Wanted more dynamic interactions	P2, P8
Used visualizations most often	P5, P8, P9
Used time-scaling most often	P1, P4, P6
Used agent batching most often	P2, P3, P4
Couldn't use time-scaling	P3, P5
Chose not to log agents	P1, P2, P3, P6
Did not use agent view	P4, P5
Exclusively used personality profiles	P3, P4
Exclusively used personality sliders	P8, P9
Found PathOS+ intuitive or easy to learn	P1, P4, P5, P8, P9
Had issues learning how to use PathOS+	P2, P3, P6
Thought runtime minimap was useful	P3, P4, P5, P6, P8, P9
Thought first-person agent view was useful	P4, P6, P9
Wanted clearer feedback	P1, P2, P6
Unable to see gizmos	P3, P6, P9
Liked right-clicking objects	P1, P4, P5, P6
Thought PathOS+ was time-efficient	P1, P6, P8, P9
Wanted AI improvements	P2, P4, P6, P8
Wanted usability improvements	P3, P4, P6
Wanted better onboarding	P1, P8, P9
Would use PathOS+ for personal projects	P1, P2, P4, P5, P6, P8, P9
Would use PathOS+ for professional project	P3

In terms of their expert evaluation process many participants (P2, P3, P4, P6, P8) started their evaluations by watching the agents navigate the level, as exemplified by P6 who stated, "*I just wanted to watch the agent do whatever he was doing first*". Their evaluations would then be guided by their observations of agent behaviours. When it comes to their overall evaluation strategy, some participants preferred observing single agents (P6) and others preferred observing batches of agents (P3, P5, P8). P2 and P4 even utilized a mix of both methods. Generally these participants tended to use at least some form of batching in their evaluations. To explain this in further detail, half of the participants (P1, P2, P6, P9) used PathOS+ to generate qualitative data. They used features such as the first-person agent camera to focus on player perspectives. On the other hand three participants (P3, P5, P8) focused on using PathOS+ for generating quantitative data. They used features such as the batching, logging, and visualization to understand broad trends. As stated by P9, they said "I would just keep an eye out to see how they would interact with each sort of different asset, how they would go about navigating the level", whereas P3 said they would “Watch the agents of the batch test... So I can see if I can get a more diverse response, to have them go in multiple directions... And I was really focusing on the motivation, what would actually pull them towards the end goal". In contrast to these participants, P4 used PathOS+ to perform a comprehensive review of each level starting with generating qualitative data by focusing on player perspectives, and then moving into quantitative data analysis by observing batches of agents. They stated "*I would start with a single agent go through, just like following in his footsteps ... And then if I had batching working, I would set out for one type, and then I'd set up for another personality, so on and so forth, and just kind of see where they differed...*".

Overall participants (P1, P2, P4, P5, P6, P8, P9) reported that observing PathOS+ agents navigating each of the levels challenged their assumptions about the scenarios from their initial observations (before running the simulations). Specifically, what frequently challenged the expectations of participants (P1, P5, P6, P8, P9) was the way in which agents navigated through levels and chose targets. Participants (P5, P6) credited it with helping them recognize the problem of repetitiveness in Scenario 3, and some (P1, P3, P4, P8)

also reported that using PathOS+ had provided them with insights they would not have obtained by playing the level themselves. Describing their experience of having expectations challenged, P1 said they initially "*just assumed that like you like players, they start, they just go to the end. But now I understand that players actually like, they want to explore each area and kind of interact with everything that's available to them*". They explained how they usually "*don't really deal with the optional goals or anything, I just kind of go straight to the end*". In the end they said the tool "*opened my eyes because it is supposed to simulate a player going through the area*".

There were still some issues with the use of the tool and agent behaviour, despite it helping participants gain new insights. Four of the participants (P1, P5, P6, P8) mentioned that switching between agents and PathOS+ window tabs during simulations could be cumbersome. P1 wanted more seamless agent switching, saying "*I know under [the agent tab], where you can toggle when the camera can follow. I thought it might have been a little helpful if I could switch... I mean, you can switch [in the] Setup tab. But maybe like when the camera is following the player*". P5 found the current agent swapping to be frustrating, stating, "*When I was trying to follow a certain agent, I would have to drag them in from the hierarchy into the agent reference. And I found that kind of annoying at times*". Additionally, two participants (P2, P8) also reported a desire for more dynamic interactions. P8 would have liked mobile enemies, and both P2 and P8 wanted more dynamism between agents and the environment, with P8 saying "*So I guess with the combat encounters that were simulated here, like, the enemies are basically standing in place, right? Is there like, can the enemy tracking stuff be mapped onto movable enemies?*". P2 also wanted the tool to emulate a wider variety of game genres, or combat features such as poison and environmental obstacles.

When asked, the visualization tools (P5, P8, P9), timescaling (P1, P4, P6), and agent batching (P2, P3, P4) were reported as the features used the most. All the participants who reported the visualization tools as their most used feature stated that they liked using heatmaps. To that end P9 said, "*I really like looking at some heatmaps... it was really cool to see like the different [paths the agent] could take and possible scenarios*".

Additionally, P3 and P5 wished to use timescaling, but were prevented by bugs, with P5 describing how they "*tried playing around with the timescale, but it didn't seem to do much*". When asked about the least used features, half of the participants said that they chose not to log player data for further analysis (P1, P2, P3, P6), with P6 explaining, "*I think this was a little too much. Like there's a little too many things to go through for me*". Two participants chose not to use the first-person agent view (P4, P5), with P4 saying, "*I didn't really use the player view often, unless I noticed something drastically different*". Additionally agent personalities were used differently across participants. Some participants used pre-set personality profiles exclusively (P3, P4), with P3 explaining, "*I went through just the presets as opposed to tweaking it myself*". Others solely adjusted the individual agent traits manually (P8, P9), as stated by P9, "*[I wouldn't use profiles as much,] I would just tweak him here and there and I was pretty good on it*".

Five participants (P1, P4, P5, P8, P9) described learning to use PathOS+ as easy to learn or intuitive. As said by P9, "*[Learning to use the tool was] pretty simple. I mean, I'm also familiar with Unity. So using it really made sense. Setup wasn't too difficult at all. And all of it was pretty comprehensive to me*". P4, despite not accessing the manual during the study due to forgetting that they had it in their inbox, was able to learn how to use the tool. Of the three participants (P2, P3, P6) who had a differing opinion, P3 said the tool "*was a little finicky. Especially with all the setup*". P2 and P6 said that there was a steep learning curve, though P2 also added that the tool would not be difficult to learn for users with experience using Unity, saying "*I think it's pretty easy to work with... if you have already a little experience with Unity, probably have no issue with this*". It is also worth noting that the issues P3 were having with the setup was the result of a bug with the batching.

Overall, participants described their experience of using the interface of PathOS+ favorably. UI elements that were considered useful included the runtime mini map (P3, P4, P5, P6, P8, P9) with P3 stating, "*...having the map show up was excellent. Their memory map, that I really liked looking at*". The first-person agent view also received praise (P4,

P6, P9), with P6 saying, "*I loved the small little window on the bottom right to show what the what the agent is looking at in the world*". However, three participants (P1, P2, P6) described that they would have liked more clear feedback for events such as characters taking damage, with P1 saying, "*I think it would be cool that whenever a player got hit by an enemy, it could maybe show some feedback to how much damage they took, maybe it was 18, with a little bit less, because [when I] was running the test, the agent, we just run into an enemy and just lose a lot of health*". Additionally, three participants (P3, P6, P9) were unable to see gizmos such as entity tags during simulations. This could be a bug with the Unity engine itself and would require further investigation.

Regardless of any grievances, all participants indicated that they considered PathOS+ to be a useful tool for conducting expert evaluations. In particular, four participants (P1, P4, P5, P6) greatly appreciated the feature of attaching comments to game objects by right-clicking. P1 said, "*I really like how you can right click and leave comments anywhere in the scene. And that, like devs, were actually working on it, they can import that and they can see exactly what you mean*" and similarly P5 said, "*I like all the color coding [and] being able to select things... just right clicking on stuff in the scene...*". Four participants (P1, P6, P8, P9) also described the tool as being time efficient in regards to the time scaling and the batching features, with P6 saying, "*Like, say, for example, I want to be able to run 1000 simulations of this level that I just created... I would use this tool to run 1000 different situations [with a] variety of different agents... this tool is really useful to do just that... I think you can just get a... cursory look into the level design. And I think that's super useful*". When asked if PathOS+ was suitable for expert evaluation, P2 believes that the tool can help identify design flaws, saying "*I think it's probably [a very good tool], because the distribution of items and how a player has walked through the levels to find the final goal... is the final goal achievable for all players or not? You can answer [this question] by using this tool*". When asked the same question P8 stated, "*I can absolutely see the types of data that this produces that would be beneficial to me in conducting my expert evaluation*".

Lastly, seven participants wished to see certain changes. This included improvements to

the AI (P2, P4, P6, P8), which involved making it more humanlike as described by P4, "*I do think the AI does need a little bit of of work just because there's so much backtracking with a lot of them... I couldn't feasibly see a player like doing that many loop de loops with no rewards*". Participants also wished for usability improvements (P3, P4, P6) with P3 explaining that "*I think it needs another just usability pass for just setup especially*". Additionally, three participants (P1, P8, P9) requested for the onboarding and learning experience of PathOS+ to be improved. They requested: more documentation and a tutorial scenario (P1), tooltips for reminders on the tool's features (P8), and a better indication of how to toggle gizmos (P9). Lastly, when asked seven participants (P1, P2, P4, P5, P6, P8, P9) said they would likely use PathOS+ for personal projects. An outlier was P3, who suggested they would use it for commercial games (specifically mentioning a tactical strategy game).

5 Discussion and Conclusion

The following section breaks down our analysis of the results of this study. We go over the identification of issues, the perceived benefits of PathOS+, its limitations, and concluding remarks.

5.1 Identification of Intended Issues

The purpose of putting intentional design issues within each crafted scenario was to see if participants would be able to successfully identify them through the use of the tool. The reason why this is important is because one of the few drawbacks of expert evaluation is that it can be difficult to justify conclusions due to a lack of player data, and important issues may not be identified by the evaluator [16]. If PathOS+ played a hand in the identification of issues, it could act as proof that AI playtesting can potentially be leveraged to improve the validity of expert evaluation.

It became clear that some participants were only able to identify some issues with the use of AI. This was especially evident with Scenario 3, and how participants stated that they would not have realized how difficult the level was to navigate without seeing how much the AI struggled. This is related to how participants stated that watching the AI allowed them to gain a new perspective, and challenged their existing assumptions about the scenario. The comments that the participants made indicate that watching the agents helped them learn new things that they may not have learned without the AI, which can help them make more accurate evaluations.

There were also some issues noticed by very few participants, in particular with Scenario 2. We have some hypotheses as to why this is the case. With participant P8, they had severe issues running the scenario on their computer, which could be why they didn't

identify the issues for that scenario. But other than that, PathOS+ was not adequately equipped to portray certain game mechanics like poison, which is where the difficulty of Blighttown (the inspiration of Scenario 2) came from. The evaluation of this scenario also represents something that occurs often in game development, which is that the designer's intention of a given level does not fully translate into practice. This level was intended to have punishing combat, which in actuality ended up being really balanced because of the damage values set to the enemies. If this were an actual situation, if not for the tool and the feedback gathered from it, the designer may not have realized until much later in the development cycle. This can also act as an example that the tool hypothetically fulfills its intended purpose.

Overall the results show that through the use of PathOS+ participants were able to identify most issues and use the data to justify their findings. Tools like PathOS+ can help evaluators by giving them evidence for the claims that they make, which may help improve their confidence when it comes to reporting issues.

5.2 Benefits of PathOS+

By examining the data another research assistant and I were able to identify some potential benefits of using PathOS+. The following sections highlight what these benefits are.

5.2.1 PathOS+ supports flexible styles of work

Analysis isn't always conducted for the same purposes. Different user research-oriented tasks can have different objectives and desired deliverables. With this in mind, it's ideal for analytical tools to support different styles of analysis, so that a broad array of researchers can benefit from the tool's use. Our study indicates that PathOS+ appeals to support different styles of analysis. Specifically, by examining the study results we were able to narrow down the participant evaluation approaches into two distinct types.

Nearly half the participants focused on the "player perspective". This meant that they

prioritized running single tests to get an intimate look into the player's behavior. As P2 described, running single tests allowed them to put themselves in the shoes of the player. The "player perspective" focused approach can be valuable in instances where a developer wants a glimpse into how their target audience may interact with their game, with a focus on the individual interactions that shape the experience. The qualitative data gained can help developers craft an experience that appeals to their target audience.

Other participants focused on "data generation". This primarily consisted of running batch tests of many agents in order to get very high level data. As described by P3, running batch tests helped them obtain more diverse data. The "data generation" approach can help to empower participants to see the bigger picture and identify broad patterns in player behaviour. The quantitative data obtained makes it suitable for things like game balancing, which requires seeing large amounts of data to identify trends and determine what aspects of the design need to be adjusted.

Interestingly, some participants used a hybrid analysis approach. In particular, P4 started with "player perspective" by focusing on single tests because they were, in their words, "following in the agent's footsteps". They later switched to a "data generation" approach in which they ran batch tests of agents with set personality types, in order to examine their behavior and identify patterns. This hybrid approach lets the participant gain the benefits of both qualitative and quantitative data, and it shows in their in-depth analysis.

By looking at the results we were able to determine that PathOS+ is flexible enough to support qualitative, quantitative, and even hybrid approaches to data analysis. This means that PathOS+ isn't just suitable for one specific evaluation style, but that different kinds of evaluators can leverage the benefits of this tool, and they can even switch up their approach midway through an evaluation. This helps make the tool more applicable to different types of situations.

5.2.2 PathOS+ supports different levels of experience

It can be hard for less experienced evaluators to get full use out of a tool if the learning curve is too steep. Likewise, tools catering to those with less experience may not be suitable for those who don't get enough functionality out of it. Ideally a tool should be able to support an evaluator regardless of their experience level.

The results of the study showed us that PathOS+ was able to accommodate researchers with differing levels of experience. This is evident with how the participants performed in the evaluation task. P1, for instance, had the least amount of experience, yet they were one of the participants who identified most of the core issues. On the other hand, P5 had almost double the amount of experience, yet they identified more or less the same amount of issues. Their years of experience (or lack thereof) did not stop them from performing well in the task.

Ultimately Unity did not impede the researchers' ability to complete the task, though it may have had an impact on how comfortably the researchers were able to find issues. The participants who had the least Unity experience (P2: who had only started learning Unity within the past year; P6: who had absolutely no Unity experience; and P8: who had only used Unity on a couple of instances) also had among the most UX experience spread across multiple industries. Yet while they were able to complete the task, they were not among those who found the most issues. P2 identified the most issues with 5, whereas P6 and P8 identified 4. It's important to note that a big reason why P8 found just under half the issues could have been due to the technical difficulties they were having with their laptop, which made the simulations they ran very slow, or impossible in the case of Scenario 2. While they were all able to get the hang of the tool and complete the task, the interface of the tool could be improved so as to be less daunting to those with less Unity experience.

The results show that a researcher does not need to have multiple years of UX experience in order to benefit from PathOS+. This is important because it means that the tool is approachable to any user researcher, and even for the inexperienced it can still be an

effective way to identify design problems. The caveat is that the level of Unity experience may have an impact on issues found. While the intended target audience is those who have enough Unity experience to comfortably use the engine, it's still beneficial to identify ways to make the tool easy to use for anyone, which is why the diverse participants in the study were immensely useful in highlighting things that may have otherwise been overlooked. Regardless, those with little Unity experience were still able to complete the evaluation task in the same window of time as the other participants. Overall the approachability of PathOS+ was one of my original objectives for the tool, which it seems like I have made strides towards.

5.2.3 PathOS+ improves efficiency in the game development process.

Game testing processes can be time-consuming. Screening and recruiting participants, setting up the lab space, running the tests themselves, and even preparing and analyzing the data can all take up an inordinate amount of time. One of the often touted benefits of AI—-in the context of game playtesting and other fields—-is its ability to save time. This is shown in case studies where AI agents could run more tests at faster rates than humans [97], [98]. Additionally AI algorithms can solve common issues with the timeliness of data analysis that can occur in situations like the volume of player data being too large for human researchers to reasonably sift through [117]. The use of algorithms can make it easier and faster to go through and identify trends within large collections of data.

Since it's such a notable benefit of AI, we were curious to see how PathOS+ would fare in this regard too. By examining the study we noticed multiple participants mentioning that PathOS+ can help save time. They mentioned this primarily because the tool allows playtests to be run frequently and early in a game's development, and references were made to specific features such as the time scaling (which some mentioned as a highly used feature) and running batch tests (as mentioned by participants like P6, there could theoretically be 1000 tests run at once). The main benefit of this in a tool such as PathOS+ is that design problems can potentially be identified, and subsequently recti-

fied, early in the development process. This means that players may be provided with a better experience, and that the game can be further iterated upon.

We were also paying attention to literature done on expert evaluation methods that highlighted some of the ways they were inefficient. Notably, in one study the researchers found that swapping between the programs where the evaluation was being done and the game in question was an annoyance [69]. Our intention with PathOS+ was to provide a potential solution to this issue. PathOS+ empowers users to conduct expert reviews in the same program as the game being developed. As pointed out by P6, this is very convenient because they don't have to switch between different software, everything is located where they need it. Solving this annoyance can not only make expert evaluation more efficient, but features like being able to right-click and attach gameobjects to specific comments can also help researchers better contextualize their findings.

The literature also suggests that user researchers prefer digital tools for expert evaluation because overall its more efficient [69]. To see if PathOS+ fulfilled this purpose we paid attention to the participant responses. Many participants directly mentioned that they appreciated the Expert Evaluation tab, and the features located within it such as the ability to export their findings via the tool's comment system. There was one outlier with P4, who mentioned that their personal preference was writing things down on paper, but they still liked and appreciated the Expert Evaluation tab. Overall, the praises the Expert Evaluation tab received are a positive sign. They were designed to make the evaluation process more efficient, and the way the participants responded seemed to indicate that they succeeded on that front.

All the results indicate that PathOS+ has the potential to save a lot of time (especially early) in the testing process, in particular through the time-scaling and batching features. It also appears to help make the expert evaluation process more efficient.

5.3 Applications to Industry

The primary intended application for PathOS+ is in indie game development, where it has the potential to make playtesting a more viable option. This is due to a number of factors, such as how the tool requires little setup in comparison to traditional playtesting. All an independent developer would have to do to use PathOS+ is download and import the files into an existing Unity project. They would have to drag and drop the prefabs into the scene they want to test, and mark up all the entities of the level. These are very few steps in comparison to all the steps required to set up a playtest, such as recruitment and lab preparation. Additionally, because of the flexibility of the tool, aspects of the agents or the level entities can be changed instantaneously. This makes it very easy for independent developers to use this tool at any stage of the development, whether that entails running preliminary tests when their game is in its seminal stages in order to gain an early understanding of the player's flow through the level, or running in-depth tests when almost everything is finalized in order to gain a better understanding of the full player experience.

Another thing that can make PathOS+ a viable alternative in comparison to traditional testing for independent developers is that by emulating different personality types, it can help make up (at least in part) for the lack of human feedback. Through the simulation of documented player personality types, the tool can also be used to get some high level insights into how different types of players (e.g. aggressive, to cautious, to completionist player types) engage with a designer's levels. The tool also offers additional flexibility in case the designer isn't convinced with how the agents are behaving. The matrix that determines the agent's personality is available within the interface for them to modify the values as they wish. All of the source code is available to the user via Unity scripts, so they can make any more modifications to the personalities (or any other aspect of the tool) as they see fit, adding to the tool's flexibility.

While human playtests can be run separately to AI tests, there is also a potential for PathOS+ to be combined with human testing. Theoretically, independent developers

can use the tool early on in development to identify problem areas, and then later on run playtests with humans in those areas in order to get more in-depth human insight. This can help developers gain the benefits of quick and rapid testing, and also feedback from players. Overall, all of these aspects of the tool makes PathOS+ (and similar tools) an attractive prospect.

A feature that can be applicable to anyone within the industry (independent or triple A) is the ability to run large-scale tests, or a high volume of sequential tests, which is a key application of PathOS+. Through the batching features developers can quickly run dozens of simulations without having to go through the costly hurdles of recruiting human testers. The number of agents that can be run through the batch test simultaneously is currently maxed at 8 agents. But even with this limited amount, using the data generated from these agents the developers can gain insights while saving time and money. Especially if they were to use the timescaling features.

All the data from these large-scale tests can be aggregated in the visualization, which can then be leveraged to gain new insights. As an example, if developers were to examine the ensuing heatmaps from these batch tests, they may be able to determine which parts of the level the agents are spending the most time on and which entities they are interacting with the most. If this does not align with their vision of the level they can then go and adjust the design accordingly. This can work very well with game balancing efforts, as seen during the study with some of the intentional design issues incorporated into the scenarios. To use an example, the combat in the first scenario was unbalanced. It was brutally difficult, and too hard for players who are not highly experienced. By running tests with the tool and leveraging features such as the batch tests, visualizations, and even (in some cases) regular observations, almost all the participants were able to identify and document this issue. A similar scenario could hypothetically be seen in a game development studio, where a user researcher uses this tool to identify that the combat in a level is too difficult, and the designer uses that feedback to rebalance the combat to something more suitable.

In his talk titled “What makes a great usability expert review?” Sebastian Long details

some of the downsides to expert evaluations as they currently are in the industry, and mentions some things that expert evaluation does not have the capacity to cover due to the lack of player data [16]. This includes covering difficulty, because evaluators may not be able to tell if the game is difficult from a player's perspective. What is shown with PathOS+ (and subsequently the previous example) is that researchers had the confidence to point out game design flaws—-such as difficulty—in the scenarios because they had the AI data to support their findings. This shows that tools like PathOS+ have the potential to make these sorts of observations more viable in the industry within an expert evaluation context.

Lastly, communication is notoriously difficult within game development studios. Especially in teams where it can be hard to find the time to create presentations and explain findings to one another. In PathOS+ the Expert Evaluation tab not just gives developers the ability to directly document their findings within the game, but it also allows them to export their findings in a format that is easy to read and understand. These evaluations can then be sent to team members, who can choose to read it as is (via the .CSV file) or load it into their version of the game. The applications of this are manifold. Team members would be able to read general comments by the user researcher while also seeing comments tied to specific game objects within the scene, getting a better understanding of potential issues. Team members could also follow up with the user researcher as needed. This ease of communication can be greatly helpful in facilitating quick and easy discussions within teams, especially when things are busy and there isn't enough time to book a meeting or give each team member an in-depth explanation.

5.4 Limitations and Future Work

A potential reason that the participants were not able to find all the issues is due to the nature of this simulated study. In a practical setting the participants would have had access to the game developers. They could talk to them to learn more about the game and receive design guidelines to reference. Because this is a simulated study, these elements were absent. The participants were not aware of the designer's intentions with the levels

and had to piece details together themselves, which is important to keep in consideration due to its effect on the results.

Another thing that could have affected the identification of intended issues, and something that numerous participants mentioned, is that the AI is lacking in dynamicism. A common comment was that combat encounters were too simple, that sometimes it was hard to figure out if a combat encounter even occurred because it looked the same as any other entity encounter. On top of that, certain ingame effects—such as poison—are not able to be adequately simulated in PathOS+. This overall lack of dynamism meant that it was harder for participants to visualize or fully understand certain interactions, which in turn can impact their ability to analyze the experience of a level. To improve upon this issue, dynamic and animated interactions could be added into the game (i.e. the agent doing a generic attack animation), and further combat effects could be simulated (poison, burning, etc.). The dynamism PathOS+ provides can also be taken a step further by improving the scope of the resources through the introduction of currency and other similar items, which are another thing that can greatly impact a player's ingame behavior.

Adding to this, the AI's general behavior could be tweaked in some areas. Another common remark by participants is that the agents backtrack a lot. In some cases it didn't seem entirely realistic, which appeared to be particularly evident in the building in Scenario 1 and throughout Scenario 2. Agents could be modified to either not backtrack as much, or to backtrack based on their personality (such as if they prioritized exploration).

The game genres that PathOS+ supports could be expanded. At the moment it is oriented towards action-adventure games. It is not as well suited for genres such as racing or puzzles, as these genres were not immediately intuitive to the agent's reasoning and navigation system. However some adjustments could be made to the algorithm to accommodate more of these genres. This may help make the tool more generalizable.

While PathOS+ was not difficult to learn for those with existing Unity experience, the usability of the tool could be further improved due to some difficulties certain partici-

pants had. For instance, the Setup tab is the only section where the user can set up the agent reference. If the user were to, say, select a different agent from the Unity hierarchy during a simulation to quickly check out that agent's stats or first-person view, it would not work. They would have to go through the Setup tab. This can cause a bit of a headache, as noted by P3 and P9 in particular. A fix to this particular issue could be to make the agent reference swap to whatever the user selects in the hierarchy (if they do). Another thing that could be done is streamline the visualization even further. While it seems that more participants used and appreciated the visualization features this time around, some still found it a little confusing (P3 and P6 knew it existed but didn't use it, for instance). Further steps could be taken to make it easier to set up and view the visualization data. For instance, the file saving automatically defaults to a file that can not be used due to some Unity settings, which can be confusing for users because they have to go through the process of finding the correct file outside of the immediate Unity directory. There could be a workaround to this that makes it easier for them to save and load that data. The naming convention could also be made more clear. Many of these things could be given a more distinctive interface so users understand what they're looking at, and are not bombarded by too much information.

Something else that is important to note is that there were certain bugs present within the build that the researchers were not aware of ahead of time. These included: not being able to scale the time in the Batching tab, batching agents would not get deleted from the hierarchy after a batch run, and personalities applied directly from the Batching tab may not apply. Another bug that is harmless but a nuisance is that error messages would randomly show up in the Unity editor tab that were not actually indicative of an error within the software. This is a Unity engine specific bug, but ideally would be fixed for an optimal experience for the next iteration of the tool (along with the other bugs listed). Additionally, some participants reported performance issues. P1 had some crashes, and P8 struggled to run the Unity project on their computer. It is not yet determined whether this is due to hardware issues, or something within the software causing the difficulties. Further investigation would need to be done to determine the cause of these performance

hindrances.

Some of the features I could choose to focus on for the next iteration of the tool could be based on the positive comments identified during analysis. For instance, the expert evaluation features were highly praised, especially the different aspects of commenting. This could be taken a step further. A possible feature could be that when the user creates a comment attached to a specific game object, that comment could show up in the above that specific gameobject within the game level. This could make it easier to visualize where certain issues are located, or if there are any particular pain points in a level. The way data gets exported could also be made more robust, perhaps with graphs within the .CSV file to make it easier to parse the data. All of these ideas could make the evaluation aspect of the tool more robust.

There are other ways that my future work could expand the capabilities of the tool. For instance, machine learning could be an avenue that I explore. I chose not to integrate machine learning into this iteration of the tool because it's not very accessible due to it requiring a lot of existing player data. A work around to this issue could be to spend resources to obtain player data that would then be fed into the algorithm. The benefit of this would be that the agent may be able to replicate more realistic player data. The downside would be in the data containing biases or any other issues, not to mention the high resource cost. Either way, this could be an interesting avenue to explore.

I could also go down the route of computer vision. This is the process which allows AI to see things the way a human may see things. This is a complicated process, but could have some benefits for the AI agent, namely making it so that they behave more human-like. Currently PathOS+ does not take into colors or light and darkness when it comes to an agent's visibility of entities within a level. The benefit of taking this into consideration is that this is something that affects regular gamers as they go through games, with some games even using light and darkness as a core mechanic. Incorporating this could make the tool more generalizable. It may be out of scope, but could be interesting to explore nonetheless.

Lastly I could expand the open-source nature of the tool. While the codebase and entity is exposed to the user, modifying parts of it may seem daunting to those who are inexperienced with Unity scripting. I could make the interface more user-friendly and allow users to add their own entities and resources with ease. Additionally I could make the process of creating personality types within the tool more robust, since as of now modifications would have to be made in various parts of the codebase to allow for that, which can be cumbersome. Improving these open-source aspects of PathOS+ could further help make it more generalizable and easier for users to modify the tool to suit their own personal needs.

5.5 Conclusion

Games are a fascinating phenomena that have been around for almost as long as humans themselves. This medium has evolved a lot from its rudimentary iterations in the ancient world. Due to this evolution, and that it's a multi-million dollar industry, user researchers require adequate methods to evaluate games before release. This is to ensure a game's quality so that players not only enjoy the game, but that the development studio is also profitable. Researchers need methods to evaluate games because it's not always obvious to developers what the flaws are, because they are so close to the games that they are working on.

Expert evaluation is one such method. As a predictive methodology its primary benefits are that it is cost-effective, easy to conduct, and can help provide valuable insights early on in a game's development phase. It may not accurately capture the issues with the player experience, however, and may instead be used to misrepresent the importance of a non-issue. Playtesting is another method that developers use that involve getting players to play through a build of the game. While it is a costly and resource-intensive process, it can give designers an accurate glimpse into what the experience of the game is. Research has been done to show that AI could possibly mitigate the downsides of playtesting. My belief is that this can be taken a step further. AI testers can allow these two techniques to be merged to take full advantage of the benefits that they both offer,

without the detriments that are attached to these methods.

This is where PathOS+ comes in. It was developed based on an open-source AI playtesting tool for the Unity engine (dubbed PathOS). The primary purpose for PathOS+ was to be a tool for the enhancement of expert evaluation. By identifying some issues with the original tool I was able to make some edits to PathOS+. This includes, but are not limited to, usability improvements, a centralized window where all the features could easily be accessed, a simple combat and resource system, a more streamlined visualization system, and last but not least a special tab that could be used to conduct expert evaluations from within a game level. These features expanded the capabilities of the tool while giving users the ability to conduct evaluations and export their findings.

In order to determine how PathOS+ fares in comparison to my intentions for it, I ran a study with those of varying UX experience to determine how beneficial PathOS+ was for expert evaluation. In order to do this I created three unique scenarios in Unity that were based off of real commercial games. These scenarios had intentional design flaws that the participants of the study were meant to identify with the use of the tool by conducting expert evaluations. Once the participants completed this task they exported their findings and had an interview with me in order to explain their findings, along with their overall experience with the tool. Another research assistant and I then analyzed the data.

The results from this study were illuminating. Overall, participants were able to identify a majority of the design flaws, with some caveats. In examining this data we identified things about how participants approached the expert evaluation task with the tool. For instance, participants largely found the tool intuitive to learn, even if they had different levels of experience. They really appreciated the expert evaluation features, especially when it came to things like attaching comments to specific gameobjects. We recognized that there were distinctive styles of evaluation when it came to the participants—-some went with a player centric approach, where others focused on generating lots of data. Crucially, through their remarks it was made evident that some participants would not have been able to identify the design issues without the aid of the AI.

I then identified some limitations of the tool, such as some usability issues, bugs, and a lack of dynamism with the AI. I also discuss things I could do to improve it for the future. Things like expanding on the expert evaluation features, and exploring more advanced options such as machine learning and computer vision.

As demonstrated by this thesis, research tools (like, but not limited to, PathOS+) can have a positive impact on game development. This is especially the case for small and independent developers who have limited resources and may not be able to conduct playtests. Through research tools their capability for analysis and gaining pre-release insights can be improved. Overall this thesis is an important contribution to the GUR and HCI fields. It enables a wider breadth of developers to benefit from user-centered approaches in their game's development.

References

[1] M. Solly, *The best board games of the ancient world*, Feb. 2020. available from: https://www.smithsonianmag.com/science-nature/best-board-games-ancient-world-180974094/.

[2] E. Gatica, *The 2020 classic tetris world championship proves its spot in the esports world*, Dec. 2020. available from: https://apptrigger.com/2020/12/08/2020-classic-tetris-world-championship-esports/.

[3] L. Lanier, *Nintendo switch online at 9.8 million accounts, 'tetris 99' popular*, Apr. 2019. available from: https://variety.com/2019/gaming/news/nintendo-switch-online-nearly-10-million-1203198704/.

[4] N. Sutrich, *Beat saber surpasses $100 million in revenue on the oculus quest platform*, Oct. 2021. available from: https://www.androidcentral.com/beat-saber-surpasses-100-million-revenue-oculus-quest-platform.

[5] J. Chamary, *Why 'pokémon go' is the world's most important game*, Feb. 2018. available from: https://www.forbes.com/sites/jvchamary/2018/02/10/pokemon-go-science-health-benefits/?sh=6c6fbb043ab0.

[6] D. Takahashi, *Sonic the hedgehog 2 film beats predecessor, speeding to $331.6m at the box office*, May 2022. available from: https://venturebeat.com/2022/05/05/sonic-the-hedgehog-2-film-beats-predecessor-with-331-6m-at-the-global-box-office/.

[7] B. Gilbert, Video-game industry revenues grew so much during the pandemic that they reportedly exceeded sports and film combined, *Business Insider* [online] 2020, 2020. available from: https://www.businessinsider.com/

video-game-industry-revenues-exceed-sports-and-film-combined-idc-2020-12.

[8] D. Partis, *Over 50% of households in the us own a games console*, Jul. 2021. available from: https://www.gamesindustry.biz/articles/2021-07-20-over-50-percent-of-households-in-the-us-own-a-games-console.

[9] R. E. S. Santos, C. V. C. Magalhães, L. F. Capretz, J. S. Correia-Neto, F. Q. B. da Silva, and A. Saher, "Computer games are serious business and so is their quality: Particularities of software testing in game development from the perspective of practitioners," ser. ESEM '18, Oulu, Finland: Association for Computing Machinery, 2018, ISBN: 9781450358231. DOI: 10.1145/3239235.3268923. available from: https://doi.org/10.1145/3239235.3268923.

[10] L. Parker, *The science of playtesting*, Sep. 2012. available from: https://www.gamespot.com/articles/the-science-of-playtesting/1100-6323661/.

[11] S. International, *75 years of innovation: The computer mouse*, Aug. 2020. available from: https://medium.com/dish/75-years-of-innovation-the-computer-mouse-fef5161ba45d.

[12] R. Marsden, *Cyberclinic: Forget rsi. 'gorilla arm' beats it hands down*, Oct. 2010. available from: https://www.independent.co.uk/tech/cyberclinic-forget-rsi-gorilla-arm-beats-it-hands-down-2118318.html.

[13] D. C. Englebart, "Augmenting human intellect: A conceptual framework," 1962.

[14] P. Mirza-Babaei, V. Zammitto, J. Niesenhaus, M. Sangin, and L. Nacke, "Games user research: Practice, methods, and applications," in *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '13, Paris, France: Association for Computing Machinery, 2013, pp. 3219–3222, ISBN: 9781450319522. DOI: 10.1145/2468356.2479651. available from: https://doi.org/10.1145/2468356.2479651.

[15] P. Mirza-Babaei, "Biometric storyboards: A games user research approach for improving qualitative evaluations of player experience," Ph.D. dissertation, Aug. 2013.

[16] S. Long, "What makes a great usability expert review? lessons from my practice game analyses," Presented at gamesUR Conference, 2019.

[17] P. Mirza-Babaei, N. Moosajee, and B. Drenikow, "Playtesting for indie studios," in *Proceedings of the 20th International Academic Mindtrek Conference*, ser. AcademicMindtrek '16, Tampere, Finland: Association for Computing Machinery, 2016, pp. 366–374, ISBN: 9781450343671. DOI: 10.1145/2994310.2994364. available from: https://doi.org/10.1145/2994310.2994364.

[18] W.-s. Tan, D. Liu, and R. Bishu, Web evaluation: Heuristic evaluation vs. user testing, *International Journal of Industrial Ergonomics* [online], vol. 39, no. 4 2009, pp. 621–627, 2009, Special issue: Felicitating Colin G. Drury, ISSN: 0169-8141. DOI: https://doi.org/10.1016/j.ergon.2008.02.012. available from: https://www.sciencedirect.com/science/article/pii/S016981410800053X.

[19] H. Dia, *'self-driving' cars are still a long way off. here are three reasons why*, Apr. 2021. available from: https://theconversation.com/self-driving-cars-are-still-a-lng-way-off-here-are-three-reasons-why-159234.

[20] C. Q. Choi, *7 revealing ways ais fail*, Oct. 2021. available from: https://spectrum.ieee.org/ai-failures.

[21] R. Anyoha, *The history of artificial intelligence*, Aug. 2017. available from: https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/.

[22] M. Daily, S. Medasani, R. Behringer, and M. Trivedi, Self-driving cars, *Computer* [online], vol. 50, no. 12 2017, pp. 18–23, 2017. DOI: 10.1109/MC.2017.4451204.

[23] Y. Zeng, E. Lu, Y. Sun, and R. Tian, *Responsible facial recognition and beyond*, 2019. DOI: 10.48550/ARXIV.1909.12935. available from: https://arxiv.org/abs/1909.12935.

[24] P. Noor,
is it ok to ...': The bot that gives you an instant moral judgment, Nov. 2021. available from: https://www.theguardian.com/technology/2021/nov/02/delphi-online-ai-bot-philosophy#:~:text=Ask%5C%20Delphi%5C%20is%5C%20a%5C%20bot,right%5C%2C%5C%20wrong%5C%2C%5C%20or%5C%20indefensible..

[25] T. E. of Encyclopaedia Britannica, *Turing test*. available from: https://www.britannica.com/technology/Turing-test.

[26] E. Colson, *What ai-driven decision making looks like*, Jul. 2019. available from: https://hbr.org/2019/07/what-ai-driven-decision-making-looks-like.

[27] E. Bermudez Contreras, B. Clark, and A. Wilber, The neuroscience of spatial navigation and the relationship to artificial intelligence, *Frontiers in Computational Neuroscience* [online], vol. 14 Jul. 2020, p. 63, Jul. 2020. DOI: 10.3389/fncom.2020.00063.

[28] G. M. van de Ven, H. T. Siegelmann, and A. S. Tolias, Brain-inspired replay for continual learning with artificial neural networks, *Nature Communications* [online], vol. 11, no. 1 2020, 2020. DOI: 10.1038/s41467-020-17866-2.

[29] N. Shaker, M. H. Shaker, and J. Togelius, "Ropossum: An authoring tool for designing, optimizing and solving cut the rope levels," in *AIIDE*, 2013.

[30] P. Braun, A. Cuzzocrea, T. D. Keding, C. K. Leung, A. G. Padzor, and D. Sayson, Game data mining: Clustering and visualization of online game data in cyber-physical worlds, *Procedia Computer Science* [online], vol. 112 2017, pp. 2259–2268, 2017, Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 21st International Conference, KES-20176-8 September 2017, Marseille, France, ISSN: 1877-0509. DOI: https://doi.

org/10.1016/j.procs.2017.08.141. available from: https://www.sciencedirect.com/science/article/pii/S1877050917314989.

[31] J. Liu, S. Snodgrass, A. Khalifa, S. Risi, G. N. Yannakakis, and J. Togelius, Deep learning for procedural content generation, *CoRR* [online], vol. abs/2010.04548 2020, 2020. arXiv: 2010.04548. available from: https://arxiv.org/abs/2010.04548.

[32] S.. Stahlke, A. Nova, and P. Mirza-Babaei, "Artificial playfulness: A tool for automated agent-based playtesting," in *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, ser. CHI EA '19, Glasgow, Scotland Uk: Association for Computing Machinery, 2019, pp. 1–6, ISBN: 9781450359719. DOI: 10.1145/3290607.3313039. available from: https://doi.org/10.1145/3290607.3313039.

[33] M. Dealessandri, *An introduction to games user research*, Aug. 2021. available from: https://www.gamesindustry.biz/articles/2021-08-11-an-introduction-to-games-user-research.

[34] A.-L. Porlier, *Introduction to playtesting and quality assurance*, Nov. 2021. available from: https://pixelles.ca/2021/11/introduction-playtest-qa/.

[35] L. Nacke, M. Ambinder, A. Canossa, R. Mandryk, and T. Stach, Game metrics and biometrics: The future of player experience research Jan. 2009, Jan. 2009.

[36] P. Mirza-Babaei, S. Long, and E. Foley, "Understanding the contribution of biometrics to games user research," in *DiGRA Conference*, 2011.

[37] P. Mirza-Babaei, L. E. Nacke, G. Fitzpatrick, G. R. White, G. McAllister, and N. M. Collins, Biometric storyboards: Visualising game user research data, *CHI '12 Extended Abstracts on Human Factors in Computing Systems* 2012, 2012.

[38] A. Clerico, C. Chamberland, M. Parent, P.-E. Michon, S. Tremblay, T. H. Falk, J.-C. Gagnon, and P. Jackson, "Biometrics and classifier fusion to predict the fun-factor in video gaming," in *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, 2016, pp. 1–8. DOI: 10.1109/CIG.2016.7860418.

[39] N. Halabi, P. Mirza-Babaei, L. Uszkoreit, and G. Wallner, "Exploring severity of gameplay issues from players' perspective," Nov. 2020. DOI: 10.1145/3383668.3419932.

[40] A. Drachen, *What is game telemetry?* Nov. 2020. available from: https://gameanalytics.com/blog/what-is-game-telemetry/.

[41] A. Gagné, M. El-Nasr, and C. Shaw, "A deeper look at the use of telemetry for analysis of player behavior in rts games," Oct. 2011, pp. 247–257, ISBN: 978-3-642-24499-5. DOI: 10.1007/978-3-642-24500-8_26.

[42] G. McAllister, P. Mirza-Babaei, and J. Avent, Improving gameplay with game metrics and player metrics, in. Mar. 2013, pp. 621–638, ISBN: 978-1-4471-4768-8. DOI: 10.1007/978-1-4471-4769-5_27.

[43] H. Owen, *Better games from diaries and data*, Mar. 2016. available from: https://www.gamedeveloper.com/design/better-games-from-diaries-and-data.

[44] S. Hillman, T. Stach, J. Procyk, and V. Zammitto, "Diary methods in aaa games user research," in *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '16, San Jose, California, USA: Association for Computing Machinery, 2016, pp. 1879–1885, ISBN: 9781450340823. DOI: 10.1145/2851581.2892316. available from: https://doi.org/10.1145/2851581.2892316.

[45] P. Mirza-Babaei, L. Nacke, J. Gregory, N. Collins, and G. Fitzpatrick, "How does it play better? exploring user testing and biometric storyboards in games user research," Apr. 2013, pp. 1499–1508. DOI: 10.1145/2470654.2466200.

[46] P. Mirza-Babaei, S. Stahlke, G. Wallner, and A. Nova, A postmortem on playtesting: Exploring the impact of playtesting on the critical reception of video games, in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 1–12, ISBN: 9781450367080. available from: https://doi.org/10.1145/3313831.3376831.

[47] J. Lazar, J. H. Feng, and H. Hochheiser, Chapter 10 - usability testing, in *Research Methods in Human Computer Interaction (Second Edition)*, J. Lazar, J. H. Feng, and H. Hochheiser, Eds., Second Edition, Boston: Morgan Kaufmann, 2017, pp. 263–298, ISBN: 978-0-12-805390-4. DOI: https://doi.org/10.1016/B978-0-12-805390-4.00010-8. available from: https://www.sciencedirect.com/science/article/pii/B9780128053904000108.

[48] R. Macefield, *An overview of expert heuristic evaluations*, Jun. 2014. available from: https://www.uxmatters.com/mt/archives/2014/06/an-overview-of-expert-heuristic-evaluations.php#comments.

[49] C. Snell, *Games user research methodology series: Usability expert analysis*. available from: https://www.playerresearch.com/learn/games-user-research-methodology-series-usability-expert-analysis/.

[50] H. Korhonen, J. Paavilainen, and H. Saarenpää, "Expert review method in game evaluations: Comparison of two playability heuristic sets," in *Proceedings of the 13th International MindTrek Conference: Everyday Life in the Ubiquitous Era*, ser. MindTrek '09, Tampere, Finland: Association for Computing Machinery, 2009, pp. 74–81, ISBN: 9781605586335. DOI: 10.1145/1621841.1621856. available from: https://doi.org/10.1145/1621841.1621856.

[51] S. Laitinen, Do usability expert evaluation and test provide novel and useful data for game development? *J. Usability Studies*, vol. 1, no. 2 Feb. 2006, pp. 64–75, Feb. 2006, ISSN: 1931-3357.

[52] Y. J. Choi, "Providing novel and useful data for game development using usability expert evaluation and testing," in *2009 Sixth International Conference on Computer Graphics, Imaging and Visualization*, 2009, pp. 129–132. DOI: 10.1109/CGIV.2009.51.

[53] R. Molich and J. Nielsen, Improving a human-computer dialogue, *Commun. ACM* [online], vol. 33, no. 3 Mar. 1990, pp. 338–348, Mar. 1990, ISSN: 0001-0782. DOI: 10.1145/77481.77486. available from: https://doi.org/10.1145/77481.77486.

[54] J. Nielsen, *10 usability heuristics for user interface design*. available from: https://www.nngroup.com/articles/ten-usability-heuristics/.

[55] ——, *How to conduct a heuristic evaluation*. available from: https://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation/.

[56] H. Desurvire, M. Caplan, and J. A. Toth, "Using heuristics to evaluate the playability of games," in *CHI '04 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '04, Vienna, Austria: Association for Computing Machinery, 2004, pp. 1509–1512, ISBN: 1581137036. DOI: 10.1145/985921.986102. available from: https://doi.org/10.1145/985921.986102.

[57] H. Desurvire and C. Wiberg, "Game usability heuristics (play) for evaluating and designing better games: The next iteration," in *Online Communities and Social Computing*, A. A. Ozok and P. Zaphiris, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 557–566, ISBN: 978-3-642-02774-1.

[58] J. Arrasvuori, M. Boberg, J. Holopainen, H. Korhonen, A. Lucero, and M. Montola, "Applying the plex framework in designing for playfulness," Jun. 2011. DOI: 10.1145/2347504.2347531.

[59] A. Lucero, J. Holopainen, E. Ollila, R. Suomela, and E. Karapanos, "The playful experiences (plex) framework as a guide for expert evaluation," ser. DPPI '13, Newcastle upon Tyne, United Kingdom: Association for Computing Machinery, 2013, pp. 221–230, ISBN: 9781450321921. DOI: 10.1145/2513506.2513530. available from: https://doi.org/10.1145/2513506.2513530.

[60] D. Pinelle, N. Wong, and T. Stach, "Heuristic evaluation for games: Usability principles for video game design," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '08, Florence, Italy: Association for Computing Machinery, 2008, pp. 1453–1462, ISBN: 9781605580111. DOI: 10.1145/1357054.1357282. available from: https://doi.org/10.1145/1357054.1357282.

[61] G. F. Tondello, D. L. Kappen, E. D. Mekler, M. Ganaba, and L. E. Nacke, Heuristic evaluation for gameful design, *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts* 2016, 2016.

[62] G. F. Tondello, D. L. Kappen, M. Ganaba, and L. E. Nacke, "Gameful design heuristics: A gamification inspection tool," in *HCI*, 2019.

[63] H. Korhonen and E. M. I. Koivisto, "Playability heuristics for mobile games," in *Proceedings of the 8th Conference on Human-Computer Interaction with Mobile Devices and Services*, ser. MobileHCI '06, Helsinki, Finland: Association for Computing Machinery, 2006, pp. 9–16, ISBN: 1595933905. DOI: 10.1145/1152215.1152218. available from: https://doi.org/10.1145/1152215.1152218.

[64] B. Kumar, M. Goundar, and S. Chand, A framework for heuristic evaluation of mobile learning applications, *Education and Information Technologies* [online], vol. 25 Jul. 2020, Jul. 2020. DOI: 10.1007/s10639-020-10112-8.

[65] A. G. Sutcliffe and B. Gault, Heuristic evaluation of virtual reality applications, *Interact. Comput.*, vol. 16 2004, pp. 831–849, 2004.

[66] R. Murtza, S. Monroe, and R. Youmans, Heuristic evaluation for virtual reality systems, *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* [online], vol. 61 Sep. 2017, pp. 2067–2071, Sep. 2017. DOI: 10.1177/1541931213602000.

[67] E. Baauw, T. Bekker, and W. Barendregt, "A structured expert evaluation method for the evaluation of children's computer games," vol. 3585, Sep. 2005, pp. 457–469, ISBN: 978-3-540-28943-2. DOI: 10.1007/11555261_38.

[68] A. Harley, *Ux expert reviews*. available from: https://www.nngroup.com/articles/ux-expert-reviews/.

[69] E. T. Hvannberg, E. L.-C. Law, and M. K. Lárusdóttir, Heuristic evaluation: Comparing ways of finding and reporting usability problems, *Interacting with Computers* [online], vol. 19, no. 2 2007, pp. 225–240, 2007, HCI Issues in Computer Games, ISSN: 0953-5438. DOI: https://doi.org/10.1016/j.intcom.2006.10.001. available from: https://www.sciencedirect.com/science/article/pii/S095354380600138X.

[70] A. Sivaji, S.-T. Soo, and M. Abdullah, "Enhancing the effectiveness of usability evaluation by automated heuristic evaluation system," Aug. 2011, pp. 48–53. DOI: 10.1109/CICSyN.2011.23.

[71] N. Raghuvanshi, *30 best bug fixing software in 2022: Get free demo*, 2022. available from: https://www.softwaresuggest.com/us/bug-fixing-software.

[72] J. Thang, *The tough life of a games tester*, Mar. 2012. available from: https://www.ign.com/articles/2012/03/29/the-tough-life-of-a-games-tester.

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

[74] T. Machado, D. Gopstein, A. Nealen, O. Nov, and J. Togelius, Ai-assisted game debugging with cicero, *2018 IEEE Congress on Evolutionary Computation (CEC)* 2018, pp. 1–8, 2018.

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

[76] C. Gordillo, J. Bergdahl, K. Tollmar, and L. Gissl'en, Improving playtesting coverage via curiosity driven reinforcement learning agents, *2021 IEEE Conference on Games (CoG)* 2021, pp. 1–8, 2021.

[77] C. G. Ling, K. Tollmar, and L. Gisslén, "Using deep convolutional neural networks to detect rendered glitches in video games," in *AAAI 2020*, 2020.

[78] S. Varvaressos, K. Lavoie, A. Massé, S. Gaboury, and S. Hallé, "Automated bug finding in video games: A case study for runtime monitoring," Mar. 2014. DOI: 10.1109/ICST.2014.27.

[79] R. Bartle, Hearts, clubs, diamonds, spades: Players who suit muds Jun. 1996, Jun. 1996.

[80] L. E. Nacke, C. Bateman, and R. L. Mandryk, Brainhex: A neurobiological gamer typology survey, *Entertainment Computing* [online], vol. 5, no. 1 2014, pp. 55–62, 2014, ISSN: 1875-9521. DOI: https://doi.org/10.1016/j.entcom.2013.06.002. available from: https://www.sciencedirect.com/science/article/pii/S1875952113000086.

[81] A. Pamboris, *Overview: Newzoo's gamer segmentation and gamer personas*, Dec. 2021. available from: https://newzoo.com/insights/articles/overview-newzoos-gamer-segmentation-and-gamer-personas.

[82] A. Drachen, A. Canossa, and G. N. Yannakakis, Player modeling using self-organization in tomb raider: Underworld, *2009 IEEE Symposium on Computational Intelligence and Games* 2009, pp. 1–8, 2009.

[83] D. Melhart, A. Liapis, and G. N. Yannakakis, *Towards general models of player experience: A study within genres*, 2021. arXiv: 2110.00978 [cs.HC].

[84] R. Thawonmas, M. Kurashige, and K.-T. Chen, Detection of landmarks for clustering of online-game players, *Int. J. Virtual Real.*, vol. 6 2007, pp. 11–16, 2007.

[85] S. Roohi, J. Takatalo, J. M. Kivikangas, and P. Hämäläinen, "Neural network based facial expression analysis of gameevents: A cautionary tale," in *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play*, ser. CHI PLAY '18, Melbourne, VIC, Australia: Association for Computing Machinery, 2018, pp. 429–437, ISBN: 9781450356244. DOI: 10.1145/3242671.3242701. available from: https://doi.org/10.1145/3242671.3242701.

[86] N. Javvaji, C. Harteveld, and M. Seif El-Nasr, Understanding player patterns by combining knowledge-based data abstraction with interactive visualization, in *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 254–266, ISBN: 9781450380744. available from: https://doi.org/10.1145/3410404.3414257.

[87] P. García-Sánchez, A. Tonda, A. Mora, G. Squillero, and J. Merelo Guervós, Automated playtesting in collectible card games using evolutionary algorithms: A case study in hearthstone, *Knowledge-Based Systems* [online], vol. 153 Apr. 2018, Apr. 2018. DOI: 10.1016/j.knosys.2018.04.030.

[88] S. Roohi, C. Guckelsberger, A. Relas, H. Heiskanen, J. Takatalo, and P. Hämäläinen, Predicting game difficulty and engagement using AI players, *Proceedings of the ACM on Human-Computer Interaction* [online], vol. 5 Oct. 2021, pp. 1–17, Oct. 2021. DOI: 10.1145/3474658. available from: https://doi.org/10.1145%5C%2F3474658.

[89] Y. Shin, J. Kim, K. Jin, and Y. B. Kim, Playtesting in match 3 game using strategic plays via reinforcement learning, *IEEE Access*, vol. 8 2020, pp. 51 593–51 600, 2020.

[90] G. Hawkins, K. Nesbitt, and S. Brown, Dynamic difficulty balancing for cautious players and risk takers, *Int. J. Comput. Games Technol.* [online], vol. 2012 Jan. 2012, Jan. 2012, ISSN: 1687-7047. DOI: 10.1155/2012/625476. available from: https://doi.org/10.1155/2012/625476.

[91] A. Zook, E. Fruchter, and M. O. Riedl, *Automatic playtesting for game parameter tuning via active learning*, 2019. DOI: 10.48550/ARXIV.1908.01417. available from: https://arxiv.org/abs/1908.01417.

[92] C. Holmgård, M. C. Green, A. Liapis, and J. Togelius, Automated playtesting with procedural personas through mcts with evolved heuristics, *IEEE Transactions on Games*, vol. 11 2019, pp. 352–362, 2019.

[93] S. Ariyurek, E. Sürer, and A. B. Can, Playtesting: What is beyond personas, *ArXiv*, vol. abs/2107.11965 2022, 2022.

[94] S. F. Gudmundsson, P. Eisen, E. Poromaa, A. Nodet, S. Purmonen, B. Kozakowski, R. Meurling, and L. Cao, "Human-like playtesting with deep learning," in *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, 2018, pp. 1–8. DOI: 10.1109/CIG.2018.8490442.

[95] Y. Zhao, I. Borovikov, F. de Mesentier Silva, A. Beirami, J. Rupert, C. Somers, J. Harder, J. F. Kolen, J. Pinto, R. Pourabolghasem, J. Pestrak, H. Chaput, M. Sardari, L. Lin, S. Narravula, N. Aghdaie, and K. A. Zaman, Winning is not everything: Enhancing game development with intelligent agents, *IEEE Transactions on Games*, vol. 12 2020, pp. 199–212, 2020.

[96] J. Glenn and R. Brunstad, "Automatic playtesting for yahtzee," in *2020 IEEE Conference on Games (CoG)*, 2020, pp. 760–763. DOI: 10.1109/CoG47356.2020.9231924.

[97] F. de Mesentier Silva, S. Lee, J. Togelius, and A. Nealen, "Ai as evaluator: Search driven playtesting of modern board games," in *AAAI Workshops*, 2017.

[98] F. d. M. Silva, I. Borovikov, J. Kolen, N. Aghdaie, and K. Zaman, "Exploring gameplay with ai agents," ser. AIIDE'18, Edmonton, Alberta, Canada: AAAI Press, 2018, ISBN: 978-1-57735-804-6.

[99] O. Keehl and A. M. Smith, "Monster carlo 2: Integrating learning and tree search for machine playtesting," in *2019 IEEE Conference on Games (CoG)*, 2019, pp. 1–8. DOI: 10.1109/CIG.2019.8847989.

[100] J. Shih, S. Chockalingam, and C. Guo, *Optimize your game balance with unity game simulation*, Mar. 2020. available from: https://blog.unity.com/technology/optimize-your-game-balance-with-unity-game-simulation.

[101] J. McMaster, *Gamedriver releases new testing software, receives seed funding*, Mar. 2022. available from: https://venturebeat.com/2022/03/17/gamedriver-releases-eponymous-ix-receives-seed-funding/.

[102] G. Wallner, N. Halabi, and P. Mirza-Babaei, "Aggregated visualization of playtesting data," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ser. CHI '19, Glasgow, Scotland Uk: Association for Computing Machinery, 2019, pp. 1–12, ISBN: 9781450359702. DOI: 10.1145/3290605.3300593. available from: https://doi.org/10.1145/3290605.3300593.

[103] S. Agarwal, C. Herrmann, G. Wallner, and F. Beck, "Visualizing ai playtesting data of 2d side-scrolling games," Aug. 2020. DOI: 10.1109/CoG47356.2020.9231915.

[104] 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 2020, 2020.

[105] S. Stahlke, A. Nova, and P. Mirza-Babaei, Artificial players in the design process: Developing an automated testing tool for game level and world design, in *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 267–280, ISBN: 9781450380744. available from: https://doi.org/10.1145/3410404.3414249.

[106] A. N. Nova, S. C. F. Sansalone, and P. Mirza-Babaei, "Pathos+: A new realm in expert evaluation," in *Extended Abstracts of the 2021 Annual Symposium on Computer-Human Interaction in Play*, ser. CHI PLAY '21, Virtual Event, Austria: Association for Computing Machinery, 2021, pp. 122–127, ISBN: 9781450383561. DOI: 10.1145/3450337.3483495. available from: https://doi.org/10.1145/3450337.3483495.

[107] A. Blair-Early and M. Zender, User interface design principles for interaction design, *Design Issues* [online], vol. 24 Jul. 2008, pp. 85–107, Jul. 2008. DOI: 10.1162/desi.2008.24.3.85.

[108] GamingRevenant, *Far cry 1: Walkthrough - volcano [level 20] (realistic mode) 4k uhd - 60fps max settings*, Oct. 2018. available from: https://www.youtube.com/watch?v=1fD7xcp7KuU&ab_channel=GamingRevenant.

[109] R. Taljonick and C. Sheridan, *Remember these 11 frustrating video game levels and try not to smash your controller*, Feb. 2017. available from: https://www.gamesradar.com/frustrating-levels-nearly-made-us-break-our-controllers/.

[110] *Farcry the volcano no way to do this mission.* available from: https://groups.google.com/g/comp.sys.ibm.pc.games.action/c/BnAfjgMxFQI.

[111] D. Curtis, *Blighttown - dark souls wiki guide*, May 2018. available from: https://www.ign.com/wikis/dark-souls/Blighttown.

[112] J. Parkin, *Dark souls remastered guide: Blighttown map*, Jul. 2018. available from: https://www.polygon.com/dark-souls-remastered-guide/2018/7/2/17478910/blighttown-map-items-npc.

[113] *R/darksouls - what's so difficult about blight town?* Available from: https://www.reddit.com/r/darksouls/comments/aeskw4/whats_so_difficult_about_blight_town/.

[114] J. Pooley, *10 convoluted video game levels everyone got lost in*, Nov. 2021. available from: https://whatculture.com/gaming/10-convoluted-video-game-levels-everyone-got-lost-in?page=9.

[115] ——, *13 terrible levels in otherwise awesome video games*, Oct. 2017. available from: https://whatculture.com/gaming/13-terrible-levels-in-otherwise-awesome-video-games.

[116] *R/halo - [halo ce] "the library" campaign mission is awful.* available from: https://www.reddit.com/r/halo/comments/i077jn/halo_ce_the_library_campaign_mission_is_awful/.

[117] G. Wallner and S. Kriglstein, Visualization-based analysis of gameplay data – a review of literature, *Entertainment Computing* [online], vol. 4, no. 3 2013, pp. 143–155, 2013, ISSN: 1875-9521. DOI: https://doi.org/10.1016/j.entcom.2013.02.002. available from: https://www.sciencedirect.com/science/article/pii/S1875952113000049.

A Appendix

A.1 PathOS+ Manual

This is the manual that explains all the functionalities of PathOS+. It is a modified version of the previous PathOS manual.

PATHOS+

User Manual

What is PathOS+?

PathOS+ is an end-to-end, lightweight framework for simulating player behavior. PathOS+ agents approximate player navigation in a game's world, and can be viewed in real-time or recorded for later visualization. Agents can also be customized to mimic different player motivations.

PathOS+ is built for Unity, and designed to operate on top of your existing game projects, requiring no instrumentation or modification of game assets or code.

Happy playtesting!

Table of Contents

Quickstart Guide

First Thing's First

Set up the Unity Navmesh if you haven't already from *Window>AI>Navigation*. Make sure you have PathOSManager and PathOSAgent objects in the scene - prefabs can be found in *PathOS+/Prefabs*. If you wish to take and export screenshots you can also drag and drop the ScreenshotCamera object into the scene. Alternatively, you can go to the Setup tab of the PathOS+ Window (see *PathOS+ Window*) and instantiate the prefabs there.

Level Markup

Use the Level Markup section of the Manager tab (see *Level Markup*) to label important or interactive objects in the scene (e.g. enemies, collectibles). Tag any objects that would be indicated on a player's compass or minimap with the "Always Known" flag in the Manager tab's Entity List.

Agent Set-Up

Adjust the motive sliders under the Agent tab to reflect the desired profile, or select a profile form the available presets and apply it to the agent (see *Agent Customization*).

Running the Simulation

Hit play to start the simulation and you can watch the agent navigate in real time. Select it in the hierarchy (or the Agent tab) to view an on screen overlay showing its targeting logic, mental map, player view, and health bar (see *Runtime Interface*).

Extras

You can run multiple agents automatically as part of a testing batch (see *Batch Simulation*) and record data for later review and visualization (see *Data Recording & Visualization*).

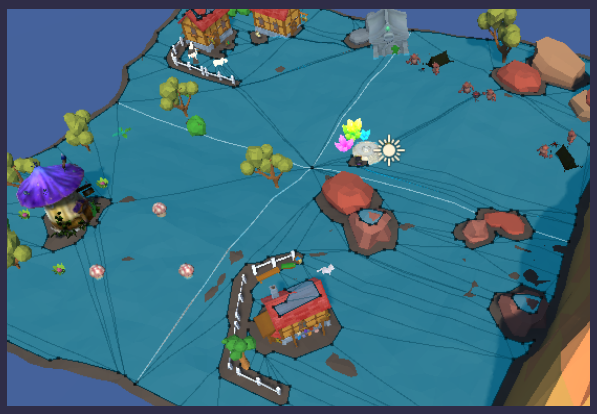
Project Set-Up

Demo Project

The included demo project and prefabs are set up to work “out of the box” - all you’ll have to do is hit the Bake button in Unity’s Navigation panel to re-bake the Navmesh after changing the level layout.

Will my project work with PathOS+?

Hopefully! If your game involves players moving around a 3D world, the answer is probably yes. As long as you can bake a Unity Navmesh for your scene, you can use PathOS+ to test your level designs. However, the tool has its limitations - be sure to read “A Few Caveats” below to see if the framework is a good fit for you.



Navmesh

PathOS+ Agents work with Unity’s Navmesh system for pathfinding. For the tool to work, you’ll need to bake your Navmesh from Window > AI > Navigation. If you’re starting from scratch, make sure that the baked agent settings (height, radius, etc.) match the settings on the Unity NavMeshAgent component of your PathOS+ Agent prefabs and GameObjects.

A Few Caveats

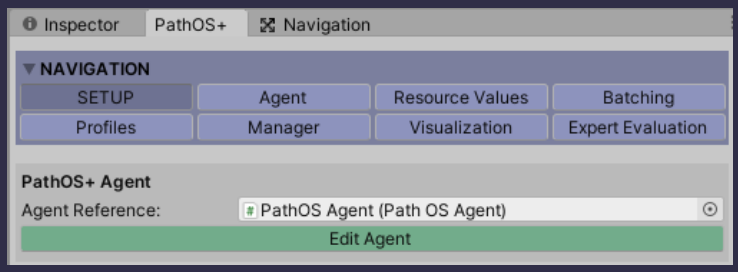
Tool Usage. PathOS+ is a tool primarily for simulating navigation, not complete gameplay. Agents’ navigation is dependent on the contextual information you provide - which GameObjects are collectibles, enemies, goals, and so on - but while there is some simulation of things like combat, agents don’t fully interact with your game’s mechanics.

Level Layout. Agents navigate in 3D, but their spatial logic is planar. This means PathOS+ works best when your level is mostly laid out at a uniform altitude with a defined ground plane - or when you can test your level in separate, mostly flat sections. PathOS+ will not work properly for levels with vertical layering.

Visibility. Whether or not agents can “see” game objects is based on Unity’s physics system, so anything you want to occlude visibility should have colliders attached. Visibility is calculated approximately, so don’t be surprised if what you see through the player’s POV camera doesn’t match up exactly with the agent’s logic.

PathOS+ Window

This window was created for the purpose of localizing every tool you'll need to set up levels and run tests. In order to open it, just navigate to *Window>PathOS+.* Here is a breakdown of the different tabs and what they each do.



SETUP. In order to function properly, the window needs a reference to the agent/manager you're using in the scene. These references (along with the reference to a screenshot camera if you added one) can be set up in this tab.

Agent. This is where all the values for the agent can be set (i.e. their personality) (See *Agent Personality*). There are also settings for the PathOS+ Camera that can be toggled during runtime.

Resource Values. The resource values can be edited in this tab. This includes how much damage each tier of enemy does, along with how much health the different potion types restore. (As an aside, how much damage the agent takes depends on their level of experience. If it's higher, they'll take less damage).

Batching. This tab can be used to run tests with multiple agents at once (See *Batch Simulation*).

Profiles. Custom agent personality profiles can be created and modified here (See *Agent Personality*).

Manager. This is where the user can do all the general setup for the level (such as establishing which GameObjects will act as entities) (See *Level Markup*).

Visualization. This is where agent playthroughs can be saved and recorded, and where they can later be loaded in as things like heatmaps.

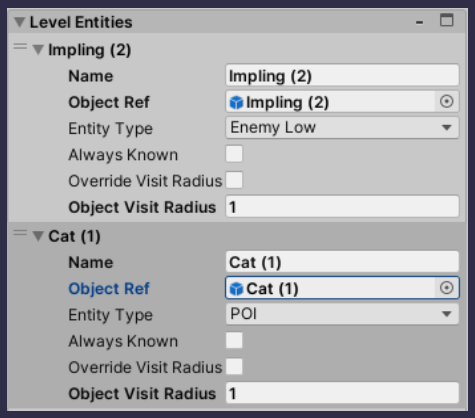
Expert Evaluation. This tab is where expert evaluations can be recorded, and later exported (see *Expert Evaluation*).

Level Markup

For agents to navigate in the context of your game, they need to understand which objects in the level can be interacted with, and what purpose they serve. To tag GameObjects, use the *Level Markup* section of the Manager tab.

Makeup Brush

In the *Level Markup* section of the Manager tab, you can click on one of the entity types to activate tagging for that type (your cursor will change). In the scene, click on objects to tag them with the selected entity type. You can also use this mode to change the tag on already labeled objects, or clear tags from labeled objects.

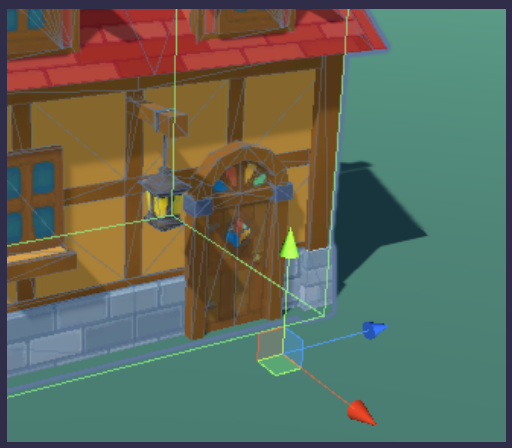


Entity List

You can also edit tags via the *Level Entity List* section in the Manager tab. You can add and remove tags using the '+/-' buttons at the bottom of the list. Here, you can also change the GameObjects referenced by each tag, or set an object as "always known" (mimicking the effect of entities which would be indicated on a player's minimap or compass).

A Note on Entity Locations

When agents target and visit game entities, they use the position of the Transform on the tagged GameObject. For large objects with colliders attached (e.g., buildings), a parent or proxy GameObject with the desired "visit location" should be used as the tag reference. The prefabs included in the demo project have already been set up with parent GameObjects with suitable pivot points chosen.



Game Entities

There are fourteen different tags available for level objects during the markup process. Here is a summary of their meaning. Type tags are used by agents to help drive their navigation through the game world.

 *Optional Goal.*	In-game missions or objectives that are optional. (e.g., sidequest marker)
 *Mandatory Goal.*	Objective that must be completed to finish the level. (e.g., main mission marker)
 *Final Goal.*	Objective that would allow the player to complete/exit the level, if applicable.
 *Collectible.*	Item that can be collected in game for achievement value. (e.g., treasure)
 *Self-Preservation Item (Low).*	Item that can be collected to boost player survivability. (e.g., low health potion)
 *Self-Preservation Item (Med).*	Item that can be collected to boost player survivability. (e.g., medium health potion)
 *Self-Preservation Item (High).*	Item that can be collected to boost player survivability. (e.g., high health potion)
 *Enemy Hazard (Low).*	Hazard that could result in a combat encounter if engaged. (e.g., low tier monster)
 *Enemy Hazard (Med).*	Hazard that could result in a combat encounter if engaged. (e.g., medium tier monster)
 *Enemy Hazard (High).*	Hazard that could result in a combat encounter if engaged. (e.g., high tier monster)
 *Enemy Hazard (Boss).*	Boss entity that results in a combat encounter if engaged.

Environment Hazard.

Interactive hazard that agent can take damage from if engaged. (e.g., poison)

Point-of-Interest.

Environment landmark intended to draw in players for exploration. (e.g., setpieces)

NPC.

Non-hostile character that can be interacted with. (e.g., quest giver)

Runtime Interface

Note: During playmode, select the agent in the hierarchy or select the Agent tab to enable the PathOS Agent UI. Unchecking the “3D Gizmos” option in Unity is recommended.

Controls

Spacebar.

Toggle on-screen legend for mental map and Gizmos.

Click and Drag.

Pan the PathOS World Camera (if present).

Mouse wheel. Zoom in/zoom out with the PathOS World Camera (if present).

Right click (in edit mode). Adds a comment to the expert evaluation tab with gameobject/entity of selected object (if open).

UI Layout

In the lower left corner, the agent's mental map is displayed. In the lower right, the agent's POV camera view is rendered. In the top right, the player's current health is shown. Gizmos are displayed on level entities indicating their state in the agent's world model.

Mental Map

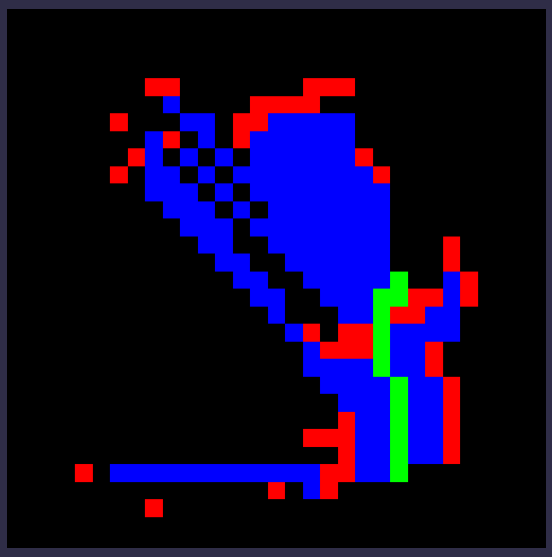
The mental map shows the agent's internal, tile-based representation of the scene's spatial layout. Four colors are used to indicate the state of each tile as perceived by the agent:

Black. Unknown

Blue. Seen as unobstructed (e.g., flat ground)

Red. Seen as obstructed (e.g. wall)

Green. Visited/traversed by the agent



Player View

The player view displays what the agent is currently “seeing” through its player POV camera. This can be used to double-check visibility of game objects outside the approximate system used by agents.

Entity Gizmos

For all game entities tagged using the markup system, Gizmos are displayed indicating how they factor into the agent’s logic at any given time. The meaning of these gizmos is as follows:

[NO ICON] Entity is not affecting agent logic - it is not visible, remembered, or previously visited.

Currently targeted by the agent. Can be applied to a game entity, or an empty point in the scene (while the agent is exploring).

Entity is contained in the agent’s memory.

Entity has been previously visited.

Entity is currently visible to the agent.

Entity has been determined to be unreachable (the agent cannot navigate to it using the Navmesh).

Agent Customization

Agents are governed by their motives, which reflect player motivations and can be customized for each agent. To change an agent's behavior, you can use the *Player Characteristics* section of the Agent tab. Here you can tweak individual motives, or apply a custom profile preset (see below).

Motives

There are seven agent motives in addition to the experience scale (e.g., amount of prior game experience). These motives affect the agent's behavior and affect the way it will evaluate tagged entities as potential destinations. These motives are as follows:

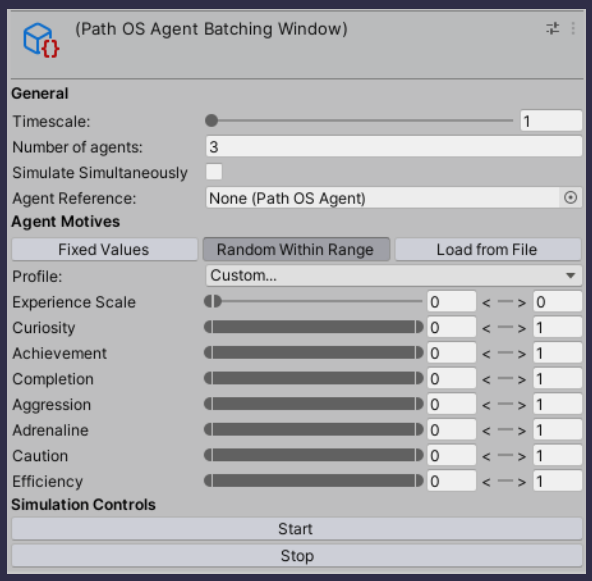
Curiosity.	The motivation to explore for exploration's sake, and discover all a level has to offer.
Achievement.	Wanting to earn achievements, complete game objectives, and rack up a high score.
Completion.	The desire to complete every in-game log, find every collectible, and so on.
Aggression.	A drive to seek out conflict and combat, dominating the game world.
Adrenaline.	Thrill-seeking, not only in combat, but in besting challenges or environmental gauntlets
Caution.	Taking care to maximize survivability, avoiding combat and hoarding resources.
Efficiency.	Wanting to get through a level as quickly as possible, prioritizing necessary goals.

For advanced users, you can view and edit the relationship between these motives and level markup tags in the *Motive Weights* section of the Manager tab. Positive weights between a motive and entity tag indicate that an agent with a high value for that motive will be drawn to entities of that type. Negative weights will cause repulsion, and a zero weighting indicates no effect of the chosen motive on the agent's behavior around entities of the chosen type.

Custom Profiles

To manage agent profiles, go to the Profiles tab. From here, you can create and edit profiles, as well as loading or saving files to carry profile sets between projects. Each profile has a range defined for the seven agent motives, as well as experience. When a profile is applied to an agent, values are picked randomly from these ranges

Batch Simulation



To simulate multiple agents automatically, go to the Batching tab. From here, you can choose to simulate agents simultaneously. All you have to do is specify a prefab for the system to instantiate for each agent needed, as well as a starting position for agents in world space.

The number of agents active at once is capped at 8. Any number greater than this will be divided into batches of 8 or less at a time, and automatically run in sequence.

Note: You can also adjust the timescale of the simulation to speed things up - note that any time limit set in the Manager tab will proceed in real time, however.

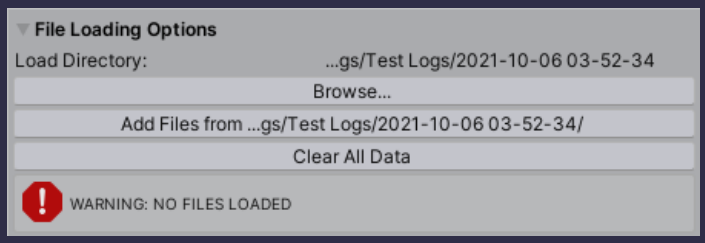
Agent motives will be initialized automatically based on the settings specified - either using fixed values, randomizing them within a range, or loading their values from a file. For range initialization, a custom profile (see above) can be selected to automatically define ranges. For file initialization, a .CSV file should be specified containing values for each of the desired agents.

Data Recording and Visualization

To automatically record logs containing information on agents' navigation and visiting level entities, toggle the "Enable Logging" setting in the Visualization tab. This will record logs both if playmode is triggered manually, or if simulation is handled en masse through the agent batching window.

Loading Data Logs

Load logs through the Visualization tab. Logs are stored as CSV files. To load them, select the directory where logs are located and hit "Add Files from...".

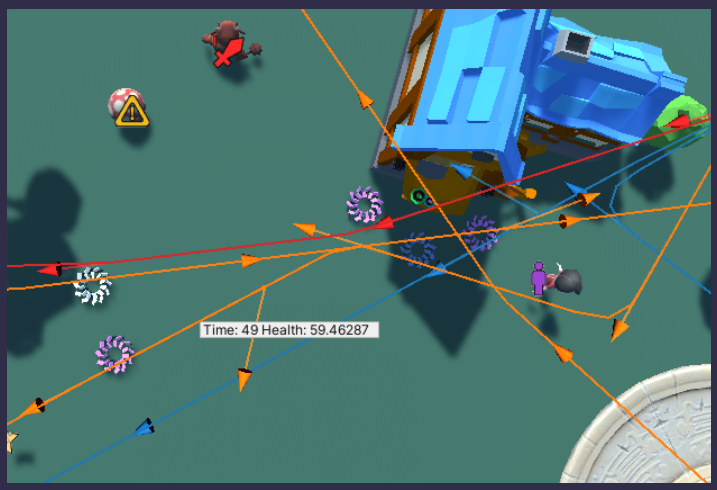


Display Filters

It's possible to set the timescale of the loaded data. You can also control what data is visible. The "display height" controls at what altitude (in units) visualization elements will be rendered in the scene. You can also choose which agents should be included or excluded from the visualization. The profiles of the agents used to create the data can be viewed by clicking the ellipsis next to each agent name.

Viewing Agent Paths

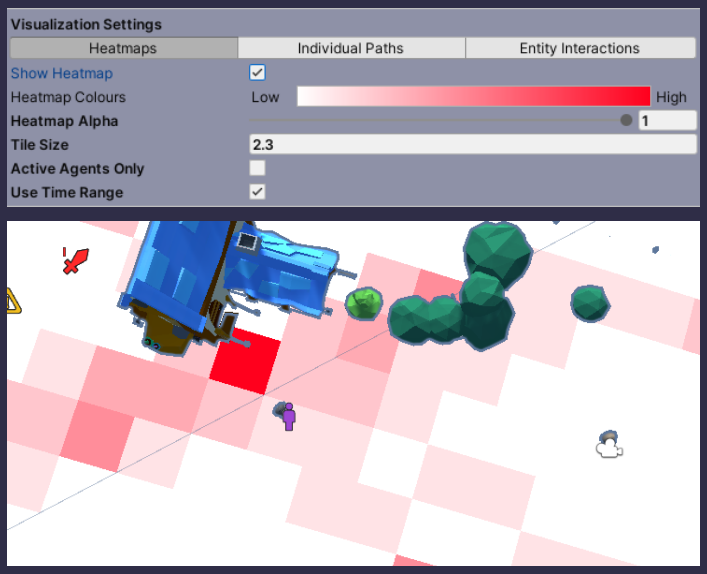
Individual agent paths through the world can be toggled, along with directional arrows for the paths. Enabling "individual interactions" will also show a record of individual agents visiting level entities during their traversal. From here, you can also specify colors for displaying each agent's trajectory. If you hover over a part of the agent's path, you can see what time the agent was at that location, and their corresponding health.

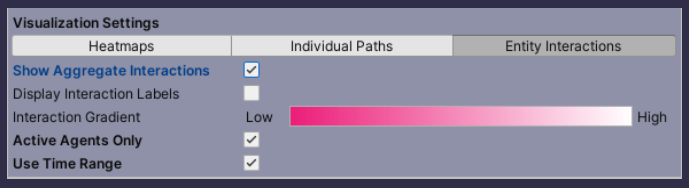


Heatmaps

To use the heatmap functionality, make sure that “Show Heatmap” is toggled after you load the files in.

The Heatmap tab can be used for customization (e.g., color scheme). The “tile size” attribute can be used to adjust the heatmap granularity. If “active agents only” is enabled, only data from agents enabled in the Filtering section will be used. If “use time range” is enabled, only data from the time range specified will be used.





Viewing Agent Interactions

In the Entity Interactions section, you can visualize how many agents visited different level entities in the scene. Each entity is shown as a circle, with scale and color adjusted to reflect the proportion of agents which visited each entity.

If “active agents only” is enabled, only data from agents enabled in the Filtering section will be used. Circles will be scaled according to the number of total agents in the enabled group. If disabled, data from all agent logs loaded will be used and scaled according to the total number of logs loaded.

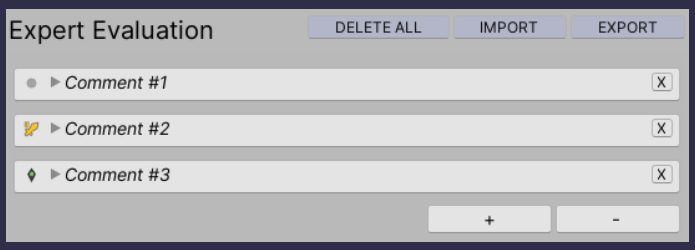
If “use time range” is enabled, only data from the range in the Filtering section will be used (i.e., entities visited outside this time range are excluded from the visualization).

Expert Evaluation

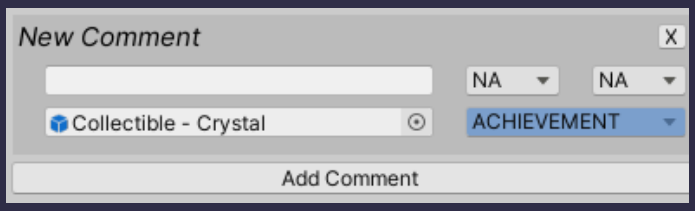
This tab can be used to record your observations as you conduct an expert evaluation.

Adding/Deleting Comments

Comments can be added by clicking the plus button, and can likewise be deleted by pressing the minus button, or the X attached to each comment.



If the PathOS+ manager reference is set up, you can right click an object in the scene. A popup will show up that allows you to add a comment which has the corresponding gameobject and entity name associated with it.



For each comment, there is a textfield where you can write down your thoughts. Each one can be rated as POS or NEG, and be given a LOW, MED, or HIGH severity. The GameObject reference and entity type can be manually set for each comment. You can also choose to leave all these fields blank if you so choose.

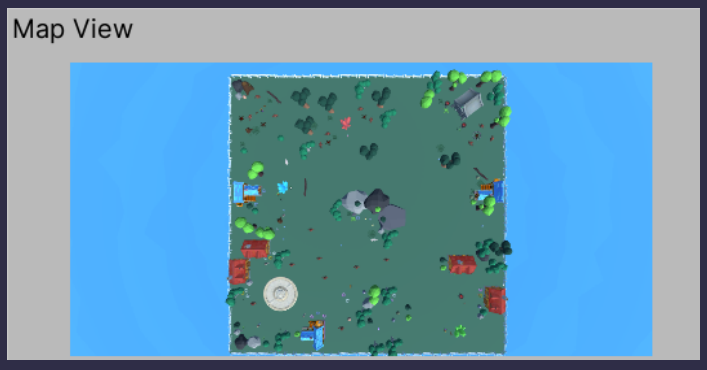
Severity	NONE	HIGH ENEMY	MANDATORY GOAL			
None						
Low	#1	#2				
Med						
High			#3			
#	Description	Severity	Category	GameObject	Object	Entity Type
1	This is a comment.	LOW	NEG	No GameObject	NA	NONE
2	This is an imp. The comment has the entity type and gameobject because I right clicked in the scene.	LOW	NEG	No GameObject	NA	HIGH ENEMY
3	This is another comment/ representative of my evaluations.	HIGH	POS	No GameObject	NA	MANDATORY GOAL

Import/Export

Evaluations can be exported or imported as a formatted .CSV file. This file lists all the different comments, and also creates a table that sorts the comments based on their entity type and severity.

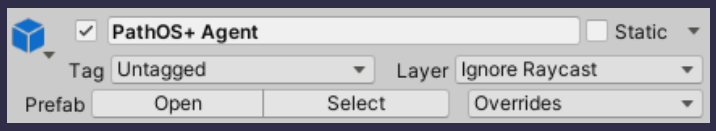
Map View

If the screenshot manager reference is set, it's possible to see a real time birds-eye-view map view of the level. This view will also be exported as a .png alongside the evaluation .CSV file.



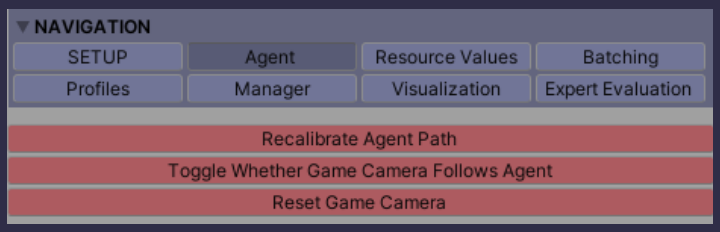
Troubleshooting

1. The simulation starts, then abruptly stops



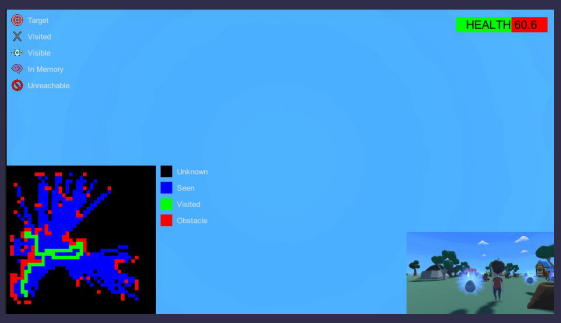
Make sure to have the PathOS+ agent enabled in the inspector.

2. The PathOS+ Agent got stuck



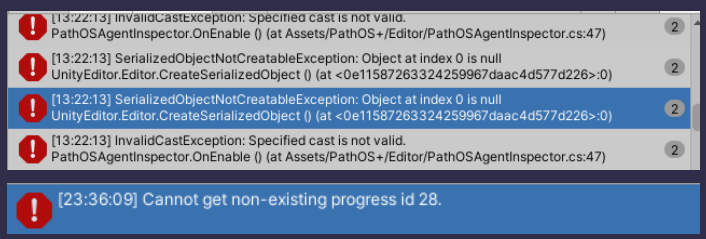
If, during runtime, the PathOS+ agent gets stuck at any point on the map, under the Agent tab there should be a button you can click that says “Recalibrate Agent Path”. This gets them unstuck. You can also press “Toggle Whether Game Camera Follows Agent” if you want the camera to continuously follow the play.

3. The game camera isn’t in the right spot



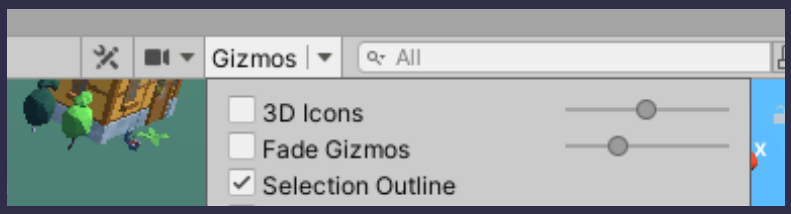
If, during runtime, the camera becomes displaced and you are no longer able to view where the agent is, you can go to the Agent tab and select “Reset Game Camera”. This will reset the position of the camera to where the agent is located.

4. I keep getting these error messages



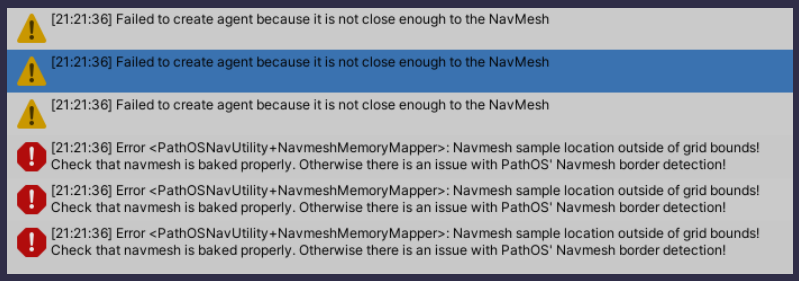
This is a glitch with the Unity engine. It won’t affect what you’re doing, but if the error messages are bothering you can restart Unity, or clear the messages out of the console.

5. The entity/runtime icons aren't visible



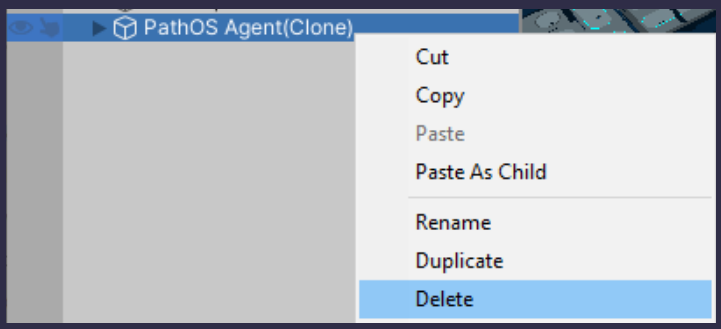
Make sure to have Gizmos enabled, and 3D icons disabled.

6. It doesn't let me run batch tests



If you're getting these errors, it means that the starting position for the batch agents aren't on the navmesh. Look at the position of the PathOS+ agent in the scene, and copy those coordinates to the X, Y, Z input fields in the Batching Tab.

7. I accidentally created an extra agent/manager/etc.



If you ever accidentally create an extra of anything, you can delete it from the scene. Find the object within the Hierarchy and right click it, then click "Delete". If you set this object as a reference anywhere (such as in the Setup tab), you will have to replace that reference.

A.2 PathOS+ Handout

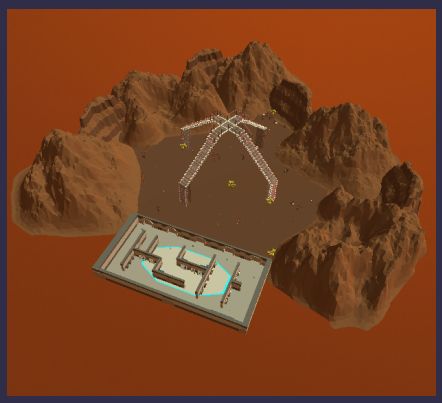
This handout was given to participants so that they understood the task they were given.

PATHOS+

Unity Demo - Evaluation Exercise

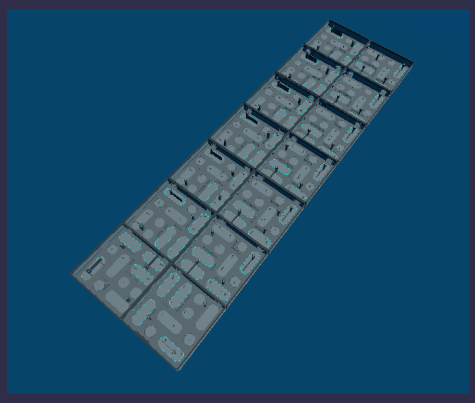
Getting Started

Navigate to *pathosplus\Unity\PathOS\Assets\User Study Project 2021*, where you will find three scenarios. These will be the scenes you'll be evaluating. You can open them in any order.



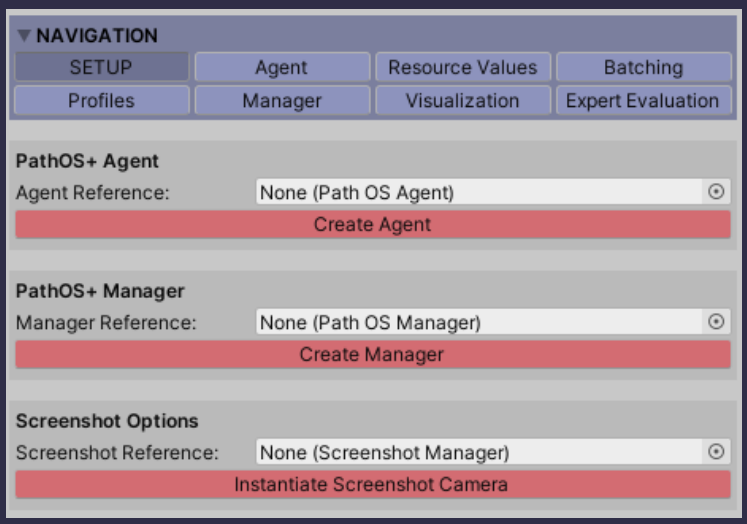
Scenario 1

Scenario 2



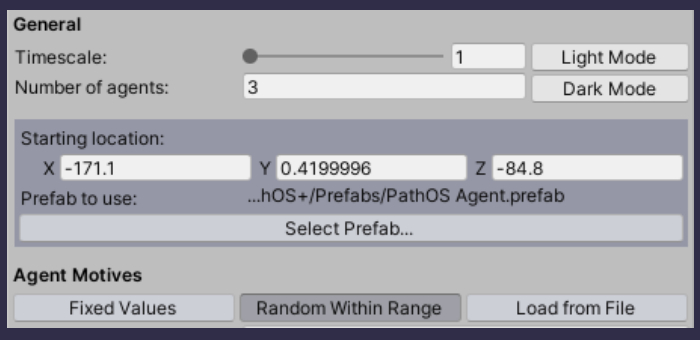
Scenario 3

Once you're in a scene, open the PathOS+ Window by clicking *Window>PathOS+*. Then navigate to the Setup tab. This is where you set up the references to the PathOS+ Agent, Manager, and the Screenshot Manager in order to use the other tabs. You'll need to do this every time you open a scene. Simply click on the little circle, and select the relevant gameobject (it's already set up for you, so don't click "Create Agent", "Create Manager", etc).



Now you're ready to test out the scenes!

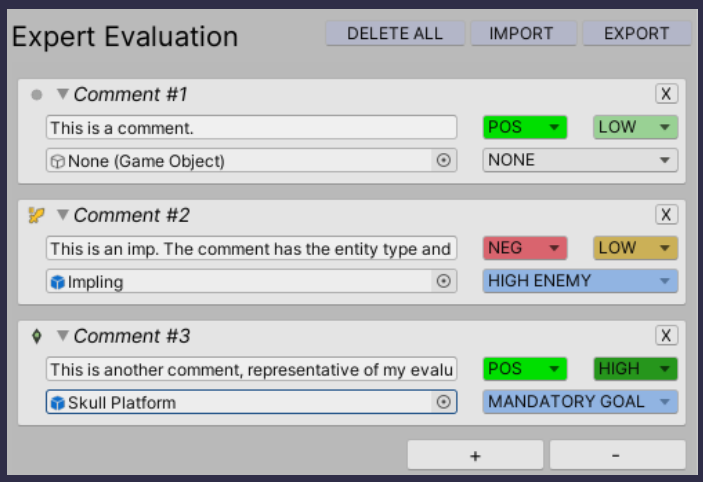
Testing & Visualization

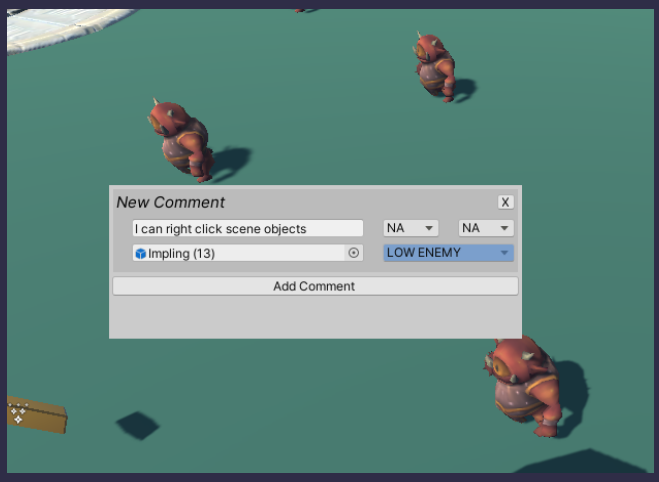


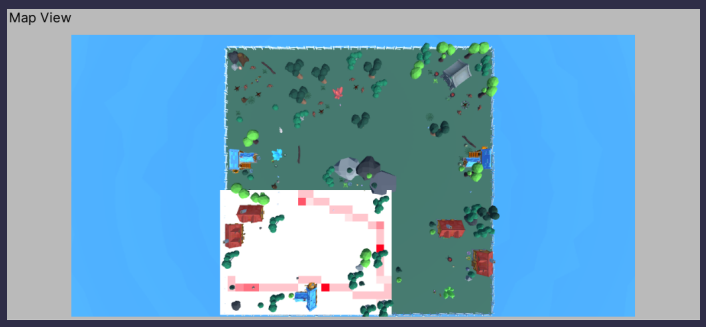
In order to run simulations, press the play button at the top of the screen. By going to the Batching tab you'll be able to run tests for multiple agents simultaneously. Make sure to select the correct prefab by clicking "Select Prefab...", going to *pathosplus\Unity \PathOS\Assets\PathOS+\Prefabs,* and selecting PathOS Agent.prefab.

Evaluation

In the Expert Evaluation tab you can add your evaluations as comments (which is where you add the explanation and justification of the observed issue), and rate them as POS or NEG, and give them a LOW, MED, or HIGH severity. You can add corresponding gameobjects or entity types to each comment. It's also possible to right click an entity in the scene and create a comment based off of its data.



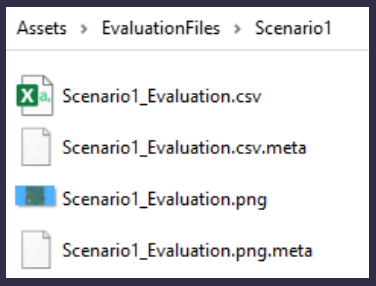




Evaluations can be exported as .CSV files, and imported back into the project later on.

It's also possible to see a real-time view of the map within the Expert Evaluation tab. When you export your evaluations, a screenshot of the map will also be exported.

Under *pathosplus\Unity\PathOS\Assets\EvaluationFiles* are folders for each scenario. When you export the evaluations, make sure to export into the corresponding folder. When you're done with all the evaluations, you can zip up the EvaluationFiles folder and email that to me at atiya.nova@ontariotechu.net. Then we'll review the results together during our second meeting.



Note: While the Expert Evaluation tab autosaves constantly, as an extra precaution I still recommend exporting your evaluations every time you're about to enter a new scene.

A.3 Interview Guide

This interview guide was used so that the researcher knew what questions to ask. **Note: This is based off the original PathOS interview guide.**

Questions in bold are numbered to indicate a rough order (may be adjusted within sections depending on the flow of discussion). Non-bolded questions are optional follow-up questions which can be asked at the discretion of the researcher to facilitate further discussion and clarify participant intent where appropriate.

As this interview follows a semi-structured protocol, the researcher may rephrase the questions stated herein and ask additional follow-ups not listed at their discretion to improve the insights which can be gained.

A.3.1 SESSION 1 - PRE-EXERCISE INTERVIEW

Purpose: Initial exploration of participant's experience with user research. The researcher should take notes on any points of interest for follow-up during the post-exercise interview.

1. Could you explain your current role/area of expertise in game development?

2. Could you briefly explain your experience with user research?

Have you ever conducted expert evaluations, or any of the methods that fall under that umbrella?

If no: An expert evaluation would be you looking at a game in development and analyzing it, and using your UX expertise in order to identify problem areas, what works, what doesn't. Ideally your findings would be given as actionable feedback to the developers.

If yes:

- Are there any tools you use when expert evaluation?

- If so, is there anything you would change about your workflow, if you could?

3. Have you used Unity before? How do you feel about using it?

After answering these questions, the participant will be introduced to the PathOS+ tool. They will be shown:

1. Where the scenarios are
2. How to run a simulation
3. How to open the PathOS+ Window
4. The different tabs:
5. How to set the references in the Setup Tab
6. How to run batch simulations
7. How to log/load data
8. How to use the expert evaluation tab
9. Some caveats re: Unity Engine bugs, and to report any issues they come across.

Lastly, they'll be assigned their evaluation task. During this process I will specify:

1. That they need to do an expert evaluation for each level
2. Where to export the evaluations to
3. Once they're done, they need to zip up the folder that has the evaluations

A.3.2 SESSION 2 - POST-EXERCISE INTERVIEW

Purpose: Explore and understand users' expert evaluation process during the exercise. Evaluate tool comprehension and feature set, understand how users apply the tool in their work, gather feedback and suggestions for improvement.

1. Could you briefly explain your top findings?

If the participant mentions that any of the levels are similar to existing levels, or asks if you intentionally made the levels based off of existing levels, ask them:

- What makes you think that way?
- Did this affect your evaluations?

2. What was your general expert evaluation process?

How much did you use the AI when analyzing the levels?

3. Did your assumptions ever get changed by the tests you ran with the AI?

Can you give me some examples?

4. Were there any insights that you think you might not have been able to have, had it not been for the AI?

5. Did you feel limited at all by the tool's features? Why or why not?

6. Were there any features you found yourself using often? Were there any features you didn't use very much?

Follow up on the reasons for using or not using individual features. Go over any notes that the participant kept during their time completing the exercise and review their use of different features.

7. How did you find learning to use the tool?

On mentioning a particular difficulty:

- Why do you think that feature was difficult to understand?

8. How did you find the interface?

- How did you feel specifically about the in-game view?

- How did you feel about the different tabs, and the UI therein?

9. How suitable do you think this tool is for expert evaluation?

- Can you elaborate on that?

10. How do you think the tool could be made more useful?

11. Describe a situation in which you could see yourself using this tool.

End with:

12. Is there anything else you would like to add?

A.4 List of Suggested Changes

This list collects all of the changes discussed with participants during the user study. **Note: This is based off the original PathOS changelist.**

Changes have been assigned a category based on their nature:

- **Usability changes** would help prevent user error or confusion.
- **QoL (quality-of-life) changes** would improve user experience by expediting certain tasks or providing additional information in the UI.
- **Additions** would be more substantial modifications allowing for additional functionality and/or an enhanced ability to suit individual game projects.
- **Bugs** Technical issues that need to be fixed for the next iteration

The participants who made these suggestions are listed next to each proposed change.

Usability More streamlined visualization ***(P3, P4, P6)***

Usability More streamlined setup ***(P3)***

Usability Abilty to make interface bigger ***(P4)***

Usability Streamlined swapping between agent/evaluation tab ***(P6)***

QoL Streamlined agent-swapping ***(P1, P2, P3, P4, P5)***

QoL Highlighted agents ***(P1)***

QoL Make it easier to rotate the camera while doing evaluation comments ***(P4)***

Additions More combat feedback (i.e. damage numbers, animations, basically anything that made it clearer what was going on during combat scenarios) ***(P1, P2, P6)***

Additions More human-like behavior from agent ***(P3, P4, P6)***

Additions More dynamicism to the combat simulations (such as effects like poison, or

moving enemies) ***(P2, P6, P8)***

Additions The ability to simulate more genres of games (i.e. board games) ***(P2)***

Additions Ability to toggle a free-form camera instead of being restricted to top-down ***(P3)***

Additions "General" tag in the evaluation tab/the ability to create our own tags ***(P5)***

Bugs AI getting stuck on level geometry ***(P3, P4, P5, P8)***

Bugs Batching doesn't delete agents after simulation ***(P3, P4, P5)***

Bugs Timescale in Batching tab sometimes didn't work ***(P3, P5)***

Bugs Batching doesn't always save agent presets ***(P4)***