Check for updates

ACM DIGITAL LIBRARY | Association for Computing Machinery | acm open

Latest updates: https://dl.acm.org/doi/10.1145/3410404.3414249

RESEARCH-ARTICLE

Artificial Players in the Design Process: Developing an Automated Testing Tool for Game Level and World Design

SAMANTHA NICOLE STAHLKE, Ontario Tech University, Oshawa, ON, Canada

ATIYA N NOVA, Ontario Tech University, Oshawa, ON, Canada

PEJMAN MIRZA-BABAEI, Ontario Tech University, Oshawa, ON, Canada

Open Access Support provided by:

Ontario Tech University

PDF Download
3410404.3414249.pdf
31 January 2026
Total Citations: 21
Total Downloads: 685

Published: 02 November 2020

Citation in BibTeX format

CHI PLAY '20: The Annual Symposium on Computer-Human Interaction in Play
November 2 - 4, 2020
Virtual Event, Canada

Conference Sponsors:
SIGCHI

https://doi.org/10.1145/3410404.3414249
ISBN: 9781450380744

Artificial Players in the Design Process: Developing an Automated Testing Tool for Game Level and World Design

Samantha Stahlke
Atiya Nova
Pejman Mirza-Babaei
samantha.stahlke@ontariotechu.ca
atiya.nova@ontariotechu.net
pejman.Mirza-Babaei@ontariotechu.ca
Ontario Tech University
Oshawa, Ontario, Canada

ABSTRACT

Iterative user-centred design has become a standard approach for developing interactive products. This process relies on prototyping and usertesting as early as possible to deliver a positive user experience and ensure that final products align with designers' intentions. The video game industry has benefited greatly from this design approach, with games user researchers adapting and introducing various evaluation techniques. However, repeatedly creating game builds suitable for usertesting (i.e., high-fidelity prototypes) is time-consuming and expensive. Moreover, recruiting users and conducting evaluation sessions are labour-intensive tasks. These challenges are especially pressing in the evaluation of game level and world design, where designers may wish to evaluate many alternatives or rapidly measure the impact of many small design changes on a game's ability to deliver the intended experience. To support developers grappling with these challenges, we have developed PathOS, a novel tool for simulating testing sessions with agents that model player navigation. This paper reports on our development objectives, implementation, and a user study with professional game developers to assess PathOS' application in a level design context. Our results demonstrate the ability of an automated testing tool to enhance the workflow of practicing developers.

CCS CONCEPTS

• **Human-centered computing** → **Human computer interaction (HCI)**; **User studies**; • **Computing methodologies** → *Artificial intelligence.*

KEYWORDS

Human-computer interaction; Usertesting; Playtesting; Game development; Artificial intelligence

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CHI PLAY '20, November 2–4, 2020, Virtual Event, Canada
© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8074-4/20/11…$15.00
https://doi.org/10.1145/3410404.3414249

ACM Reference Format:
Samantha Stahlke, Atiya Nova, and Pejman Mirza-Babaei. 2020. 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 (CHI PLAY '20), November 2–4, 2020, Virtual Event, Canada.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3410404.3414249

1 INTRODUCTION

Over the past decade, the games industry has advanced substantially in terms of technology, design, and business practices. From the introduction of new platforms (e.g., mobile, VR, AR), to novel publishing models (e.g., Free-to-play, season passes), and the inclusion of new audiences (e.g., kids, older adults), developers are rapidly adapting to provide engaging experiences for players across these new design frontiers. In focusing on the creation of fulfilling player experiences, design techniques which involve players in the development process (e.g., player-centric and participatory design) have been receiving more attention from game designers. These efforts have helped to establish the field of games user research (GUR) among academic and industry practitioners, with the goal of understanding players and their behaviour to help developers optimize and guide player experience closer to their design intent.

GUR fundamentally relies on iterative design and development processes, where prototypes created by game developers are evaluated by testing with players drawn from a game's target audience. Insights gained from this process are used to help developers improve their designs until a desirable experience is achieved for the end user. However, this process is time-consuming, laborious, and resource-intensive. Hence, many academic and industry researchers have focused on improving and optimizing this process. Such efforts can be roughly classified by their focus on a particular stage of the GUR process: 1) Triangulating user research methods to improve data collection (e.g., [43, 47]); 2) Improving data analysis processes (e.g., [34, 36]); 3) Communication of results (e.g., [58, 59]); 4) Creation of tools addressing the entire GUR pipeline from data collection to visualization of results (e.g., [17, 51]).

The work discussed here focuses on the development and evaluation of PathOS, a tool for evaluating level design in video games by simulating navigation in a virtual world with customizable AI players. There are a number of reasons why we believe a tool such as this can make a significant contribution in supporting game developers. First, usertesting sessions are time-consuming (a typical test may

take an hour per player, per session), and carry the requirements of lab space, moderation, and participant compensation. Additionally, recruiting enough participants to cover the wide range of skills and gameplay preferences extant in a game's target market is challenging. Lastly, any testing with real players demands the availability of a suitable game build, which can be time-consuming to create and typically impossible to modify "on-the-fly". Realistically, these requirements impose limitations on the number of testing rounds that can be run during a typical development cycle.

As a support to help alleviate these concerns, automated testing can supplement an established user research process, or serve as an attractive alternative for developers unable to afford frequent usertesting. Simulated testing sessions require only a computer, can be accelerated to provide results on a trivial timescale, and can run multiple agents in parallel (e.g., test with multiple "players" at once on a single machine). By leveraging player modelling, we can simulate a wide range of player types. The ability to test with agents inside a game's development environment (e.g., without needing a finished build) may help support designers in the early stages of development, when changes are being made rapidly and a game's mechanics may not be fully implemented. Taken together, these benefits may allow developers to evaluate their designs more often, and ultimately create a more successful product. It is not our intent to argue that automated testing can replace testing with real players, rather, that it can support and expand existing processes particularly in the early phases of design.

This paper continues our previous work on AI in GUR and the early ideation of our testing framework. In our previous publications, we highlighted the opportunities and challenges of AI-driven approaches in GUR [49] and provided an initial proposal for the PathOS tool [50]. This paper details our complete development process and associated user studies as we explore the application and limitations of our automated testing tool.

1.1 The PathOS Framework

The PathOS framework is a utility for predicting player navigation in digital games by simulating virtual agents which coarsely model the trajectory of human players. The tool allows users to watch and collect data on agents navigating a game's world, either in real time or via post-simulation visualization. Our primary target users for PathOS are game designers aiming to improve their level and world design, particularly in the early stages of development. PathOS may also be used by user researchers as a supporting tool for the expert evaluation of a game's design.

When we refer to level and world design, we mean the process of laying out a virtual space comprising terrain, props, interactive objects, and so on, whether this is a closed level or a larger open world. Our tool aims to help designers and researchers answer basic questions about how players will navigate a game's world, such as:

- *Will players follow the intended path through this level?*
- *Do players miss out on or avoid content placed in certain areas?*
- *How many collectibles will the average player find in this level?*
- *Where will players spend most of their time?*

PathOS is designed with flexibility for different projects and design workflows in mind, with utilities for real-time observation, post-simulation visualization, batching agents *en masse*, and customizing agents to reflect a game's target audience. The tool is freely available and open-source, to support the efforts of small development studios and independent creators.

The ultimate aim of the PathOS framework is not to replace testing with human users, but rather to allow designers to pursue more informed iteration of their creations using minimal resources. It is our hope that PathOS can contribute to the game development process by providing a versatile solution to support the evaluation of level designs. With this work, our primary contribution is to demonstrate the benefit of an automated evaluation tool in the design process of practicing game developers.

2 BACKGROUND

Artificial intelligence (AI) research has always had a strong relationship with games, with games serving as a testbed for the development of novel algorithms. In the 1950s, some of the earliest AI algorithms were developed to play board games [57]. Continual advancement eventually led to the development of systems capable of defeating human world champions, such as IBM's famed Deep Blue in chess [11] and the othello-playing program Logistello [10].

Over time, the complexity and applicability of AI increased beyond the bounds of simple board games. Modern applications of AI include autonomous vehicles [5], curating content on social media [42], and the increasing popularity of voice-command systems [41], to highlight a few examples. AI has also found many applications specific to GUR, promising reductions in human labour and resource requirements for various parts of the GUR process. We can broadly group these efforts into two categories: supporting data analysis, and simulation-driven testing.

2.1 AI for data analysis in GUR

Within GUR, AI has helped extend the limits of what researchers can accomplish, augmenting human analysis through intelligent interfaces and helping build player models from large collections of data. AI can help improve the efficiency of many of the tedious tasks associated with data analysis, such as processing large and disparate datasets. Cheong et al. [12], for instance, created ViGLS, a system that automatically generates visual gameplay summaries based on logs of ingame actions. ViGLS uses an action planner in order to infer causal relationships between game events, employing heuristics to select events for inclusion in the provided summaries.

Difficult tasks in analyzing data from individual players, such as recognizing emotional responses, may also be supported through the use of AI. Mandryk and Atkin [34] created a tool for assessing player emotion based on physiological data through the use of a fuzzy logic expert system. Machine learning (ML) has also been used as a tool for estimating player emotion in a less invasive manner. Roohi et al. [46], for example, explored the use of neural networks to recognize players' facial expressions during gameplay.

Other tools have used AI to understand and adapt to user needs during the analysis process. Brown et al. [9], used ML to predict user behaviour and performance based on their interactions during a visual search task. Such insights may be used to create tools which adapt to the needs of a user in real time. Gotz and Wen

[20], for instance, created a rule-based adaptive visualization interface that analyzes user behaviour in order to recommend alternate visualizations more suitable to a user's current task.

Investigations of user behaviour often rely on extracting insights from massive collections of player telemetry data (e.g., movement in a game's world, time to complete tasks, character deaths, etc.). With data from thousands of players, manual analysis may prove impossible. Recently, ML has been investigated as a tool to create models of player behaviour based on such datasets.

Melhart et al. [37] used preference learning to predict player motivations based on gameplay data. Drachen et al. [16] investigated several unsupervised clustering algorithms to classify players and assist in identifying common patterns of behaviour in World of Warcraft (Blizzard Entertainment, 2004). Thawonmas, Kurashige, and Chen [52] used game telemetry data to identify distinct groups of players based on their navigation patterns.

A related task is the idea of behavioural prediction; that is to say, based on the analysis of existing data, can we predict future user actions? Such predictions may be valuable in projecting user experience for the purposes of evaluation and in creating adaptive game interfaces. Mahlmann et al. [32] used high-level gameplay data from thousands of players in Tomb Raider: Underworld (Crystal Dynamics, 2008) to predict player retention via expected total gameplay time and final level reached. ML has also helped with lower-level predictions such as the inference of player objectives. Weber and Mateas [60] used regression models to predict player behaviour in Starcraft (Blizzard Entertainment, 1998). The authors argued that such models could help improve strategy game AI agents, minimizing the need for human-authored behaviours.

2.2 Simulation-driven game testing

AI has also been used as a proxy for human users in user experience (UX) and quality assurance (QA) testing. Through the creation and observation of AI agents that obey a game's rules and/or approximate player behaviour, game designers can better understand how different players will respond to a given game. Several different methods have been used to model player behaviour; we refer the reader to Justesen et al. for a comprehensive review of recent work in simulating gameplay through deep learning [23]. The relatively novel concept of evaluation through deploying agent-based testing has already demonstrated promise as a means to reduce the resource and labour requirements of game testing.

Though QA is a distinctly separate domain from UX research, we can certainly learn from tools created for the automation of game QA testing. Pfau, Smeddinck, and Malaka [44] created ICARUS, a system for exhaustive QA testing of point-and-click adventure games through reinforcement learning. Ariyurek, Betin-Can, and Surer [1] developed gameplay agents governed by RL, Monte Carlo Tree Search (MCTS), and inverse RL to mimic human behaviour in simple action-adventure games, with the goal of identifying bugs. The authors found in general that agents were capable of matching or even outperforming human testers in terms of the errors found.

Another testing application is the idea of evaluating playability, that is, objective features of a game's content, such as whether it is possible to complete a given level, how many possible solutions exist for a given puzzle, and so on. This is of particular interest for procedurally generated context, or rapid iteration on a game's rules. In such cases, exhaustive testing with humans may be impractical. Keehl and Smith [26, 27] developed a MCTS-based Unity framework for gameplay simulation dubbed Monster Carlo. This tool allows designers to quickly test their changes to a game and ensure that it is still playable. Several such tools are developed for game-specific applications, such as the automated balance-testing system developed for The Sims Mobile (Maxis Redwood Shores, 2018) [15]. Another game-specific tool developed by Shaker et al. [48] explored the automated generation and evaluation of levels for the puzzle game *Cut the Rope* (ZeptoLab, 2010).

Commercial studios have also developed solutions for automated testing of gameplay and level design, which are typically proprietary and heavily tailored towards a specific game context. Gudmundsson et al. describe the use of convolutional neural networks and MCTS to evaluate the difficulty of different level designs in *Candy Crush Saga* (King, 2012). Similarly, Rovio employed reinforcement learning to assess level difficulty in *Angry Birds Dream Blast* (Rovio Entertainment, 2018) [35]. These endeavours are far from limited to mobile puzzle games; gameplay bots were also developed for simulating large-scale playtests in *Battlefield V* (Electronic Arts, 2018). More general solutions have also been developed, such as Unity's open-source ML Agents Toolkit [21], perhaps the most similar work described here to our own in terms of intent.

Turning to the more complicated task of UX evaluation, questions surrounding UX are far more subjective than their QA and playability counterparts. Here, the task is creating agents that "think like humans". Several approaches have been proposed to model behaviour for agents serving as simulated testers. Borovikov and Beirami [7] used Markov models trained on human gameplay to generate data for a neural network simulating human-play styles in an open-world shooter game. Tremblay et al. [54] created a pathfinding tool to predict player trajectories in stealth games, allowing designers to compare results with their intended path through a level. The use of imitation learning to more accurately model human behaviour by incorporating data from real players in training gameplay agents has also been explored. Dossa et al. [22], for example, used a blend of imitation and reinforcement learning to create human-like gameplay agents for simple arcade-type games.

Beyond behavioural modeling, agents have also been explored as a means to actively "critique" game content. Liapis et al. [29] created *procedural personas* controlled by single-layer neural networks to mimic different player archetypes in a dungeon crawler game. Based on predicted gameplay, these personas provide judgements on the quality of game levels. This concept was further expanded upon by Mugrai et al. [39], who created MCTS agents that mimicked human play-styles to provide an estimation of level difficulty.

3 TOOL DESIGN AND DEVELOPMENT

As discussed in section 1, our motivation for this work is to help game creators inform their level design process. For the tool to be as useful as possible for its intended purpose and audience, we established the following design objectives:

- **Reduce the burden of playtesting:** Reduce the effort and resources needed to acquire data on player behaviour, by providing a simulated approximation.

- **Accessibility for developers:** Make it easy for developers to access and integrate the tool with their projects.
- **Ease of use for designers:** Provide a front-end interface for level designers to easily use the tool as part of their existing workflow, without any need for programming expertise.
- **Generalizability:** Maximize versatility by designing the tool to work for any 3D game navigation. Provide developers with the ability to modify or augment the functionality of the tool to suit unique project needs if necessary.

The following subsections explore the technical implementation of the framework, the behavioural model employed by PathOS agents, and the tool's designer-facing features and workflow.

3.1 Implementation

To minimize the technical overhead associated with integrating PathOS into a development project, we decided to develop the framework as an extension for an existing game engine. We created PathOS as an extension for Unity[1], a popular free commercial game engine. We chose Unity as a platform for creating PathOS due to its popularity among independent developers and creators.

The tool is written in C#, and queries Unity's existing physics and navigation mesh (or *navmesh*; a representation of a level's virtual geometry) systems to support low-level object visibility checks and pathfinding operations. To minimize PathOS' "technical footprint" on a project, all required set-up can be accomplished by dragging and dropping scripts and prefabricated objects included with the framework (this adheres to Unity's standard workflow for editing a game scene). No modification or instrumentation of project code is required to use the framework, and no modification of existing objects in a scene is necessary.

PathOS is free and open-source. A Unity project containing the source code and all necessary assets to use the tool in an existing project, as well as the tool's user manual, is available on Github [2].

3.2 Agent behaviour

PathOS use a simple model of player perception, memory, and decision-making. This model can be described as a cognitive architecture [53] integrating subsystems aimed at simulating sight and information processing to solve a selection problem (where to travel in the game world). Our approach is similar in spirit to the GOMS (goals, operators, methods, and selection rules) model used to analyze user tasks in classical HCI research [28].

As PathOS agents move in a game's world, they "see" level geometry and interactive objects, which are stored and filtered through the agent's memory. Agents use information from memory to continually re-evaluate their in-game destination based on a motivation profile. Each of these stages (perception, memory, and destination selection) is explained in more detail later in this subsection.

The type of various in-game entities present in a scene (e.g., enemies, health packs, etc.) is key information in an agent's decision-making process. This is supported by a simple tagging system whereby designers assign predefined types to game objects according to their in-game function. The tags provided in the framework (see Table 1) are based on existing game design literature.

[1]https://unity.com
[2]https://github.com/samanthastahlke/pathos

Individual variation in agent behaviour is determined by a simulated player motivation profile, informed by existing models of player typology (e.g., [4, 33, 40]). Agents have seven motives, each of which can be assigned a weight to produce the desired profile (see Table 2). Additionally, agents have an "experience" parameter which can slightly improve the agent's memory capacity and storage, meant to mimic the skill-based player characteristics (e.g., *Decisiveness*, *Control Skill*) discussed by Cowley [14].

Each motive has a vector associated with it assigning a base favourability value to each type of game entity given in Table 1. For instance, the favourability of enemies for the caution motive is very low, whereas the favourability of survival resources is high. Taken together, the vectors for each motive comprise a weighted scoring matrix used in selecting in-game destinations (described later in this section). Default values for this matrix are included with the tool, though it is editable by designers to reflect game-specific needs (e.g., estimated level of combat risk, etc.).

Given this understanding of how interactive objects and agent motivations are represented, we can move on to examine each stage of agents' decision-making process.

Perception model. Agents are equipped with a player point-of-view (POV) camera meant to mirror players' in-game view. An object in the scene is considered visible if it is contained in the POV camera frustum, not occluded by any scene geometry (determined through raycasting), and exceeds a minimum threshold for apparent on-screen size. Further information about level geometry and obstacles on the navigation mesh is gathered through multiple line-of-sight raycasts originating from the POV camera.

Memory model. Agent memory consists of two main parts; *spatial memory* and *entity memory*. Spatial memory contains a representation of level geometry (i.e., navigable terrain, obstacles), built from information obtained from line-of-sight raycasts as described above. This is represented via a tile-based map, meant to imitate the "minimap" often available to players as they reveal more of a game's world through their travels. This memory map is displayed as part of the on-screen UI (see subsection 3.3).

Entity memory contains a record of the location of in-game objects as well as their type, as tagged by the level designer (see *Entity types and agent motives* below). For objects to enter an agent's memory, they must be visible for a certain amount of time on-screen. This is meant to imitate the transfer of visual information from immediate or "iconic" memory [8] to short-term memory (STM) in humans, a process which occurs on a sub-second timescale [19, 30]. Agent STM size is limited to 3-5 entities, based on the capacity of human STM [13, 38]. Above this limit, entities not in view can be forgotten, with the oldest impressions discarded first in accordance with the idea that human memory decay is mostly dependent on time elapsed [3, 45]. Entities which are visible for an extended period or repeatedly enter the agent's field of view are transferred to long-term memory (LTM), following the rehearsal model of human STM-LTM transfer [2].

When recalling information about game entities outside the agent's field of view, their remembered location is subject to slight random variation, similar to the noise introduced during recall of sensory information in humans [31]. Like human players, agents can forget and misremember information as a result.

Table 1: Entity tags used in the PathOS framework.

Name	Description	Refer to...
Optional Goal	Objective marker not necessary for level completion (e.g., sidequest mission markers, optional puzzles)	Objective [18], optional goal [6], goal point [6]
Mandatory Goal	Objective marker necessary for level completion (e.g., main mission marker)	Objective [18], committed goal [6], goal point [6]
Final Goal	Objective which triggers the end of a given level	Objective, outcome [18]
Enemy Hazard	A hostile character, etc. which could incite combat	Enemies, agents, combat [6]
Environment Hazard	Physical hazard which could damage the player	Deadly traps, obstacles [6]
Survival Resource	Item which can be picked up for some benefit to player survival, such as health packs	Resource [18], power-up [6], resource [6]
Collectible	Item which can be picked up to contribute to achievements or unlocking content	Resource [18], objective [18], collecting [6], pick-ups [6]
Point-of-Interest (POI)	Environmental feature, landmark, or setpiece intended to capture players' visual interest	Exploration [6]
Non-Player Character (NPC)	Non-hostile character who may be interacted with for story purposes, completing a mission, etc.	Characters, agents [6]

Table 2: Agent motives comprising individual motivation profiles.

Name	Description	Refer to...
Curiosity	Drive to explore the extents of the game world; particularly drawn to POIs and NPCs	Explorer [4], seeker [40], curiosity-cognitive [33], immersion-discovery [61], adventurer [56]
Achievement	Desire to earn game achievements and accomplish feats of skill, endurance, or luck; drawn to goals and collectibles	Achiever [4], achiever [40], optimisation [14]
Completion	Motivation to finish a game "100%", completing every checkbox along the way; drawn to most interactive objects	Completionist [24], thoroughness [14]
Aggression	Wanting to exert dominance over the game world and seek out combat; particularly drawn to enemies	Killer [4], aggression [14], conqueror [40], competitor [24], brawn [55], mercenary [56]
Adrenaline	Seeking the thrill of challenge, looking to take risks; particularly drawn to environmental hazards and enemies	Daredevil/survivor [40], daredevil [56], challenge [33]
Caution	Averse to danger; repelled by hazards, drawn to survival resources	Caution/resourcing [14]
Efficiency	Desire to meet end objectives as quickly as possible; particularly drawn to mandatory and final goals	Speed [14], achievement-advancement [61], mastermind [40]

Destination planning. When planning a route through the game world, agents can select from two types of destinations: *entity destinations* and *exploration destinations*. An entity destination represents a particular in-scene object (e.g., mission marker, collectible, etc.), whereas an exploration destination represents a direction for travel without a singular object at its terminus. The favourability of a destination is assessed by assigning a score (given in Equation 1):

$$destination\ score = \sum (M \bullet W_i * d) + bias \quad (1)$$

The scoring equation consists of two parts: a sum of scores for all interactive entities the agent would pass along a straight path to the candidate destination, and a bias term for the destination itself.

Beginning with the summation, consider a case where the agent would pass two enemies and a collectible en route to its destination. Scores for each of these three entities will factor into the overall favourability of the destination. The score for each component of this sum is dependent on the agent's motivation profile, the scoring matrix described at the beginning of this section, and the agent's distance to the entity. M is the agent's motivation profile represented as a column vector consisting of the strength of each motive for the agent, on the interval $(0, 1)$. W_i is a row taken from the weight matrix corresponding to the type of a given entity. The dot product of these two terms is then scaled by d, a factor proportional to the inverse square of the distance between the entity and the agent, which is additionally affected by the angle between the agent's linear trajectory to the destination and a line drawn between the agent and the entity in question. This ensures that entities which are further "off course" in reaching a given destination have a lesser impact on that destination's final score.

For exploration destinations, the bias term is set to zero. For entity destinations, the bias term is calculated in the same way as an individual entity score within the summation term, with an additional slight bonus added to incentivize prioritizing interactive targets over "exploration" targets.

While navigating the game world, agents continually re-evaluate all entities contained in memory as potential entity destinations,

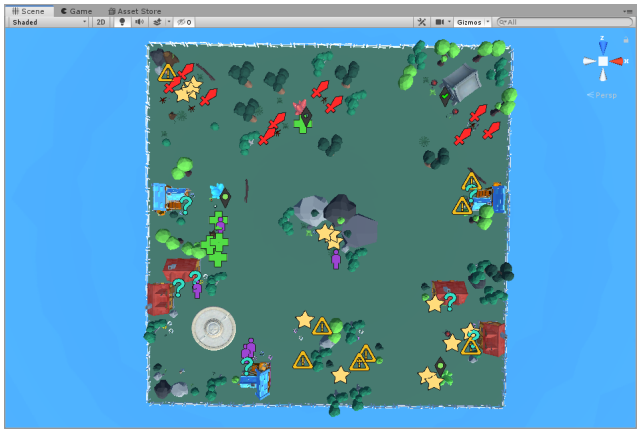

Figure 1: A screenshot of the "level markup" in PathOS. Icons are used to show the tags applied to game objects.

using trajectories cast out around the agent in a radial pattern as potential exploration destinations. If a destination is evaluated as having a higher score than the current target, it is selected as the new target and the agent is re-routed accordingly. A slight degree of stochasticity is introduced here to inject uncertainty into the decision-making process when two or more destinations are evaluated as equal or nearly equal; this attempts to mimic the imprecision in decision-making when multiple alternatives are seemingly equal in terms of favourability.

3.3 User interface

In keeping with our goal of making the tool easy to use for game designers, the PathOS user interface is integrated into Unity's existing Editor environment, the interface through which designers edit "scenes", or parts of a game's world. The front-facing functionality of the tool is described throughout this subsection.

Level markup tool. The level markup utility (Figure 1) allows designers to tag interactive objects (e.g., enemies) with their in-game function, informing agent decision-making (tags are described in subsection 3.2). This is implemented as a point-and-click system providing visual indicators of object tags, as well as an editable list of tagged objects for fine-tuning options (e.g., whether an object's location would always be known to a player).

Behaviour customization and profiles utility. Individual agents can be placed into the Unity scene as Unity game objects. Motivation profiles can be specified through a customized panel in the Inspector, Unity's existing interface for modifying script parameters. An additional utility for editing custom preset profiles by specifying ranges of values for each motive is also available. Here, users can load, save, and edit presets which are made available to the agent Inspector. From the agent Inspector, users can select a preset and apply it to the agent, automatically randomizing the agent's motives within the ranges allowed by the profile specified.

Runtime interface. If PathOS agents are active in the scene, the Play button in the Unity editor can be used to begin simulating agents in real-time. Users can select an agent in the Unity scene hierarchy to view an on-screen overlay displaying information about the contents of the agent's field-of-view, memory contents, and current destination (shown in Figure 2).

Figure 2: A screenshot of the PathOS runtime overlay. At bottom left, the agent's spatial memory map is displayed. At bottom right, the agent POV camera is shown. Icons are displayed on interactive objects to show their state in an agent's memory (seen, visited, in memory, targeted).

Figure 3: The three visualizations available for analyzing agent data. Top left: Individual path visualization. Top right: Bubble visualization for entity interactions. Bottom: Heatmap. Each visualization is customizable in terms of colour, filtering data, etc.

Visualization. A logging and visualization utility is provided to view agent behaviour after a simulation is completed. Three different visualizations are provided to review and analyze agent behaviour; individual pathtraces through the scene, a heatmap visualizing the time spent by agents in different areas, and a bubble visualization showing the proportion of agents which visited each interactive entity in the scene. Each of these visualizations provides several options to customize its appearance, filter different agents from the collection of loaded logfiles, and slice the data shown according to a particular time window. Screenshots showing the alternative visualizations available are given in Figure 3.

Batching utility. To test with many agents simultaneously or in sequence, an additional batching utility is provided. From the batching control panel, users can specify the number of agents to simulate, the profile preset or range of motivation values to use for initializing each agent, and a timescale for accelerating simulation. This functionality is used in conjunction with data logging to facilitate analysis of data from agents tested *en masse*.

4 USER STUDY

We conducted an evaluation of PathOS with game development professionals, to assess our target users' use and opinions of the tool. Our primary concerns were understanding the experience of using the tool in a level design context, and gathering feedback on what improvements could be made to fulfill users' professional needs. Our evaluation was guided by the following key questions:

- *How can PathOS contribute to the game design process?*
- *How can PathOS be improved to better suit the needs of game developers?*

Study Procedure. Our study consisted of three parts; an introductory interview (pre-exercise interview), a take-home level design exercise, and a post-exercise interview.

The pre-exercise interview session lasted approximately 30-45 minutes and was conducted either in-person or remotely according to participant availability. After providing consent, participants filled out a demographics questionnaire and answered questions regarding their design process. The researcher then introduced participants to PathOS and the task they would be asked to complete. Participants were given a brief demonstration of the tool's operation and core features by the researcher. They were also provided with a handout explaining the take-home level design exercise, the PathOS user manual, and a video demo of the tool that they could consult during the exercise if they wished to do so.

For the take-home level design exercise, participants were asked to create two levels for a hypothetical action-adventure type game, one of which should incentivize players to explore and one which would require players to complete goal markers while fighting through enemies. A Unity project was provided with the framework already installed and a collection of "prefab" assets that could be used to build each level. Participants were instructed to use PathOS in any way they chose during their design process, using the features they found helpful or wanted to explore on their own. Participants completed this exercise on their own time, and were told to spend approximately 2-3 hours in total on the exercise, though they were free to experiment with the tool for longer if they wished. Participants were also given a simple first-person controller in Unity to facilitate "testing" their levels on their own, depending on their personal preferences and process.

The post-exercise interview session lasted approximately 40-60 minutes, and was conducted either in-person or remotely, as with the pre-exercise interview. At the start of the session, participants shared the levels they created and explained their process in designing each level and using the tool. Following this, participants completed a semi-structured verbal interview exploring their experience in learning and using the PathOS framework. After completion, participants were debriefed and thanked for their time.

Table 3: Demographics and self-reported experience working with the Unity engine for study participants.

ID	Age	Occupation	Unity Exp.
P1	29	Graduate Student	Advanced
P2	N/A (Withdrawn)		
P3	25	Tracking Data Manager (Mobile)	Intermediate
P4	29	ML Expert (AAA)	Intermediate
P5	26	Data Tracking Manager (Mobile)	Novice
P6	23	Programmer (Indie)	Advanced
P7	24	UX Researcher (Consultant)	Intermediate
P8	23	Programmer (AAA)	Intermediate
P9	23	Programmer (Indie)	Advanced
P10	23	Graduate Student	Intermediate

Participants. We recruited ten participants, all with at least 3 years of relevant experience in game design/development and who had either completed an undergraduate degree specializing in game development and/or worked in the games industry. A summary of participant demographics and experience is given in Table 3. One participant (P2) withdrew during the study due to a time conflict, leaving nine participants that completed the study in total.

Of these nine participants, eight completed the study as described above. One participant (P4) had independently worked with the framework for approximately two months before being recruited for the study (as PathOS is freely available online). Due to time constraints for this participant and their existing experience with the tool, they completed only a modified version of the post-interview session (skipping the first interview and exercise entirely).

No major irregularities occurred with any of the participants during the study. One participant (P7) only created one level for the exercise due to personal time constraints. Another (P8) initially misunderstood the tool's purpose, understanding it to be a tool primarily for developing game character AI rather than game testing, although this had been explained during the pre-interview. The participant had nonetheless been using the tool to test and make adjustments to their design, and discussion with the researcher early in the post-interview session clarified this point before questions regarding the tool's utility were asked. It is therefore assumed for the purposes of analysis that neither of these anomalies had a significant impact on our evaluation's outcome.

Analysis Procedure. We performed in-depth analysis on post-exercise interview transcripts only, as the pre-interview contained no evaluative questions and we did not aim to establish any relationship between pre- and post-exercise interview responses. However, pre-exercise interview data helped to establish common design and development practices mentioned by participants for the purposes of discussing our results from post-exercise interviews.

We created two spreadsheets (*Summary* and *Features*) to analyze post-exercise interview data. Each spreadsheet contains a series of categories associated with quotes extracted from post-interview transcripts. Categories for each spreadsheet were created deductively based on the content of post-interview questions (*Summary*) and the features of the tool (*Features*). All categories could contain multiple comments.

The *Summary* spreadsheet is intended to capture a high-level synopsis of participants' experience using the tool and how they integrated it into their process; categories were created deductively based on the interview questions asked: *Process*, *Level Changes*, *Liked Features*, *Issues*, *Wishlist*, *Learning*, *Usefulness*, *Application*, *Use Cases*.

These categories are not mutually exclusive, since, for example, a comment praising the utility of a particular feature could be placed in both the "Usefulness" and "Liked Features" categories.

The *Features* spreadsheet is meant to collect comments made by participants regarding individual features. Categories were generated deductively based on the feature set of the tool: *Level Markup*, *Agent Personality Adjustment*, *Agent Behaviour*, *Runtime Gizmos*, *Mental Map*, *Agent View*, *Runtime UI (Other)*, *Data Logging*, *Heatmap*, *Path Vis*, *Entity Vis*, *Vis (Other)*, *Batching*, *Profiles*, *Inspector (Other)*.

During analysis, each category was further subdivided into comments that were positive, negative, or suggested changes. Subcategories were mutually exclusive (i.e., a single comment could not be both positive and negative), though higher-level categories were not mutually exclusive (e.g., a comment discussing two features in the same sentence could be assigned to categories for each of those features).

Two researchers completed both spreadsheets independently, assigning codes to excerpts taken from the post-interview transcripts. Following this, researchers met to ensure that 100% agreement was reached. We did not consider the inclusion of comments which were worded slightly differently (since participants would often repeat their thoughts during natural discussion) or differently trimmed (e.g., shorter or longer excerpts chosen) as disagreement.

For the *Features* spreadsheet, any comments initially assigned to different categories were discussed and re-assigned to the most appropriate category. For the *Summary* spreadsheet, no discrepancies were encountered in the interpretation of responses, and quotations selected by both researchers were combined to produce a final list of excerpts for inclusion in the discussion of results.

5 RESULTS

Participants demonstrated a great deal of creativity and variety in terms of the levels they created, often theming the levels they created or telling a story about the intended experience they wanted players to have. To gain a clearer understanding of PathOS' contributions to the game development process in practice , we asked participants to describe their use of the framework during the design exercise.

All eight participants that completed the exercise indicated that they had used the tool to test their designs. Four participants stated that they never (P6, P7) or rarely (P1, P9) tested their creations themselves, relying entirely on AI agents. Of the remaining participants, two stated that they frequently used the tool during the design process, while the remaining two participants began using the tool after completing the initial layout of their levels. Four participants (P1, P6, P9, P10) explored PathOS' logging and visualization capabilities while completing the exercise, while the remaining participants exclusively observed agents in real time.

Five out of eight participants (P1, P3, P6, P8, P9) made changes to their levels during the exercise as a result of observing agents' behaviour. Changes made by participants included geometry adjustments to prevent unintended shortcuts, layout changes meant to give players more options, and placing additional interactive objects. Of the three participants that did not make changes, one (P5) had specific ideas of what they would change in their levels if they had more time, based on agents' behaviour ("could've also put some bigger points of interest [...] lead the AI closer"). One other participant (P10) stated that they "could see a use for making some changes as [they] noticed the behaviour with the agent", but noted they were more interested in exploring the visualization utilities with the time they had.

The following subsections examine participant comments, exploring insights derived from the *Summary* (subsection 5.1) and *Features* (subsection 5.2) spreadsheets described in the previous section.

5.1 Overall impressions

Participants were generally very positive when discussing their experience using the PathOS framework. Key strengths highlighted in participant comments include:

- **Usefulness** (P3: "[PathOS] improves the process a lot. Without this tool I don't think I could've made something that I'm as happy with in such a short amount of time", P10: "It definitely helps give the designer [an idea] of how they want the level to go and important points that they want a user to experience")
- **Interface design** (P1: "It was all super intuitive, easy to use", P5: "It's very clean, and very concise with its information")
- **Ease of setup** (P1: "It was pretty straightforward learning how to use the tool", P7: "It's pretty easy to plug and play")
- **Technical implementation** (P4: "It kinda worked out of the box [...] to download code that works, and does, you know, something, is already impressive", P6: "I don't actually have any clue how you would approach this. If you wanted to be more like an actual player [...] this is great")

First-time user experience. No substantial difficulties were encountered by participants in the course of setting up and learning how to use the framework. It is worthy to note that participants received a brief tutorial on the tool during the pre-interview session, and given access to the tool's documentation. However, this instruction is in line with what is typically available for development tools online. Participants commented positively on their learning experience (P3: "Once I got started, I didn't refer to the video again, it was very intuitive", P9: "Able to figure out most of it just by trying to do something"). P4, who did not receive the tutorial session (since they had worked with the tool independently), remarked that "it was easy to see how to replicate the stuff in the [webinar]." The tool's documentation and walkthrough video were also well-received (P1: "user manual and handout helped a lot", P7: "mostly very clear"). Two participants (P5, P10) noted that the quantity of features and settings could be intimidating (P5: "It's a little overwhelming at first, but I think if you take it in strides that works", P10: "It's a lot to take in at first"), but noted that the modular design of the tool and documentation available mitigated this factor (P5: "It's great because you can choose to do only some of the features [...]

and have it still work", P10: "Having that outline on the side really helped").

Perceived usefulness. All nine participants indicated that they felt the tool was useful when asked, citing several different individual reasons. A common idea discussed was the potential time and/or labour savings created by using the tool (P1: "[...] come back in an hour later and have all of your data [...] you don't have to sit there and watch it run", P3: "I can see results, take action on those results, and it's been five minutes. That's extremely valuable.", P9: "Much faster than me having to go and test everything"). One participant (P3) also alluded directly to the burden of recruitment, saying "I can get an AI to run through it without me having to bug the three people that are near me that can actually test something, and realistically you can only get [humans] to test it once." The value of insights made from observing agents was also discussed as a factor in the tool's utility (P1: "Being able to see the general flow of where players go, how long they spend at each place. I think that's really useful.", P6: "I think I got a lot of feedback from the AI about the layout of the level", P10: "With this agent you see what the thinking process is").

Participants also commented on the generalizability and ease of integrating of the framework as a factor in its utility (P4: "It's something that can be extended very easily. And for, probably a lot of games, it doesn't need to be extended.", P6: "[...] would work with any game and not cause any issue because it's not actually modifying any game objects", P7: "quite a bit of versatility with the different genres it could be used with"). Lastly, several participants remarked on the framework's positive contribution to their general design process (P3: "It worked really well for my workflow, or my design flow, where I want to build something, test something, see how it works in that moment [...] it helped me prototype a lot faster", P7: "I think this could be a very good prototyping tool for level designers", P10: "[The markup tool] helps the designer also keep track of [their design], like having these collectibles and objectives").

Application areas. When asked whether the tool would be more useful for playtesting or quality assurance (QA) applications, most participants felt the tool could be useful for both UX-oriented questions (P1: "Just for general level flow sort of thing", P7: "Figuring out an approximation of what your player is going to do") and finding bugs (P5: "Help alleviate a lot of the trivial tasks that QA needs to go through when it comes to checking collisions and stuff like that"). Five participants (P3, P4, P6, P8, P10) expressed no strong opinion on whether the tool was more useful for one application or the other. The remaining four participants were split on whether they felt the tool would be more well-suited for playtesting (P1, P7) versus QA (P5, P9). Interestingly, reasons given for both sides of this argument stemmed from feelings that agents were less suited to mimic the behaviours of people playing "for fun" (P9: "Real people are weird and do weird stuff") or "for testing" (P7: "QA testers do some weird things that you don't expect").

Potential use cases. When asked about how they might use PathOS in their own work or personal projects, participants described a number of different scenarios. These situations included testing for open-world games (P8: "I know AAA always [needs] collectibles and corners and this would be amazing for it", P10: "Like an open-world game [...] I'd like to implement this to get those quick tests, see if this is the right placement of objects); independent or "indie" projects (P3: "A small exploration game. And I have maybe 5 people who can playtest it for me every now and then [...] just being able to have the dummy run through it, super helpful"); game jams (P1: "There's no time to get actual playtesters during a game jam. So, being able to run like a hundred agents through your level just to see how it flows"); coverage testing (P4: "If I had defined [how] interactions with NPCs work, and they go talk to NPCs, and that is the only thing necessary to solve quests, is it possible for an agent to 100% the game? [...] As a human, to test that, that would be really tedious and take a lot of time"); and procedural level generation ("I think if I were to create a procedurally generated level this tool would let me test all of the parts [...] I can see this tool being really great for procedural levels").

5.2 Feature-level impressions

For a brief explanation of the functionality of each of PathOS' features, the reader is referred to subsection 3.3.

Level markup. All nine participants commented positively on the level markup tool (P3: "I can see that it's a tool for designers", P5: "intuitive [...] clean and clear", P10: "very easy to select a markup and paint [...] I liked how there was an entity list I could go and modify"). However, all nine participants also encountered minor usability or quality-of-life issues with the tool (e.g., P3: "somewhat tedious to tag all of the [entities]"), suggesting that small modifications could help prevent user error and/or improve their experience (e.g., to prevent misclicking on objects, P9: "put them on a different layer right, and not have it be selected"). With the exception of one comment describing the optional entity list interface as "ridiculously huge" for complex levels, all negative (i.e., issue-related) comments relating to this feature were classed as low priority.

Agents. Participants commented positively on the process of changing agent personalities (P7: "clear to use", P8: "you can change the weights of the agent to modify whatever your needs [are]"). Two participants (P8, P10) suggested that tooltips for agent motives could help users to more immediately understand the meaning of different simulated player characteristics. In terms of agent behaviour, participants were mixed on whether they found agent behaviour believable (P6: "still a far cry from how a player would actually interact", P7: "I estimated what I expected players to do [...] generally the AI responded as expected"). P8 noted a fast-forward option for simulating single agents as a much-needed feature: "I was having a miserable time just watching them. You know the late stages of design? [...] Like come on. Just get there." (It should be noted that users are not required to watch agents in real-time, and that a fast-forward option is available for batch simulation. However, since some users prefer to watch agents "live", the inclusion of this feature for single agent testing could improve flexibility.)

Runtime interface. The completeness of information available in the runtime overlay was noted as a key positive (P4: "easy to explain why the agent was doing something", P10: "everything I needed to know about the agent"), along with its clarity (P6: "Understandability, information was great", P7: "pretty easy to understand

what's going on"). Negative comments in this category discussed minor issues with finding settings to customize the display, or in one case, potentially reduced accessibility of the agent's "mental map" display for users with colour vision impairments.

Logging and visualization. Some concern was expressed regarding implementation of the data logging feature, with multiple participants suggesting improvements to the process (P5: "having a default directory might be good", P9: "being able to rename what the folder is, like it's made by the logger"). One participant (P5) did not initially realize that logging needed to be enabled for visualization to work, noting that "some sort of prompt would be helpful to remind me to enable logging".

The visualization system was well-received, particularly in terms of the heatmap (P4: "I think the heatmap thing was really useful") and individual path visualization (P1: "the individual routes [were] useful", P10: "I liked that there were different colours and you can select the agent you want to focus on"). Negative comments in this category cited confusion regarding settings (e.g., referring to forgetting to adjust display transparency, P10: "I was confused I couldn't see the heatmap") or the encoding of information (P10: "a little hard to tell maybe the direction [of the agent] sometimes [in the path visualization]").

Batching and profiling. Few participants commented on the batching and profiling features, as many participants had not used them extensively. Positive comments were made on both batching (P1: "useful for running a lot of simulations") and profiles (P5: "I can click apply profile and it saves me a ton of time"). One participant (P10) expressed concern with the batching interface, noting that it could be difficult to navigate the Unity scene hierarchy and find individual agents during the simulation for the purposes of viewing their individual information in the runtime overlay.

General interface. Participants were positive on PathOS' integrated Unity Inspector user interface (P1: "super intuitive", P7: "I liked the organization of your menu systems", P10: "if you want to customize, it's there for you"). Two participants, however, noted that it could be difficult to know what to click at times, citing an overabundance of text in the Inspector (P7) or a lack of clarity in expanding parts of the interface (P8, noting that after some time using the tool, it became "pretty straightforward").

5.3 Changelist

We compiled a changelist for improvements to the framework based on participant comments in the *Wishlist* category of the *Summary* spreadsheet, and comments classified as *Negative* or *Change* from the *Features* spreadsheet. Key changes focused on general usability improvements, better suiting users with different needs, and additions to enhance the tool's functionality.

A few key changes to improve the tool's usability aim to prevent error in using the level markup tool: automatic restoration of the interface state after manipulating the Unity scene camera, the ability to selectively prevent objects such as level geometry from being accidentally tagged, and adding a precision indicator to the markup cursor to improve accuracy. Another change suggested was a prompt reminding the user to change their Unity Gizmos settings to ensure that the runtime overlay is displayed with optimal settings. A reminder prompt was also suggested for the logging utility, to prevent users from forgetting to record data.

Users also suggested smaller "quality-of-life" changes to save time in using the framework; firstly, the inclusion of support for tagging multiple objects at once using the markup tool, and secondly, adding a fast-forward option for individual agent simulation. Several small changes were also suggested for the logging and visualization systems: providing a default output directory for logs, adding a directional indicator to path visualizations, and filtering agents included in a visualization by their motivation profile.

Changes to improve the tool's accessibility were discussed with one participant (P7), who stated that their background in UX consulting makes them more cognizant of potential accessibility issues. Suggested changes included the addition of icons to the Inspector interface to make it easier for dyslexic users to navigate, and an alternate colour scheme for the "mental map" component of the runtime interface (see Figure 2) to make it more useful for users with colour vision deficiency.

Lastly, to inform future development, we also discussed substantial additions to the tool's functionality to improve its utility. The most frequently requested enhancements to the framework were support for dynamic game entities (e.g., entities that could spawn or change type at runtime), custom resources that could affect agent logic (e.g., health, ammunition, etc.), and an API that could allow for programming custom agent behaviours. Several participants also suggested that the Unity interface for the tool could be condensed into a control panel providing a central point of interaction for PathOS.

6 DISCUSSION

With the PathOS framework, we aimed to provide developers with an affordable and useful tool to inform their level design process. In the course of its evaluation, we sought to answer the following two questions: *How can PathOS contribute to the game design process?* and *How can PathOS be improved to better suit the needs of game developers?* These questions are explored in the following subsections.

6.1 How can PathOS contribute to game development?

We can understand the potential role of PathOS in the development process by examining three key points surrounding its stages of use: the types of projects it can be used for, integration into those projects, and its utility in practice.

With respect to PathOS' suitability for different projects, technically speaking, the framework could be used for any game project in Unity where players will navigate a 3D world in a first- or third-person perspective. Since the current version of the tool uses a 2D representation for agents' spatial memory, it is primarily intended for cases where most navigation occurs relative to some ground plane (e.g., open world, single-level dungeon, etc.). However, extending this representation to 3D for better adaptation to heavily vertical level design is part of our planned future work.

User study participants expressed a multitude of different use cases they felt were feasible with the framework in its current state, as discussed in subsection 5.1. Project scenarios suggested

by participants included open-world games (P8, P10), role-playing games (P4, P7), game jams (P1), an indie exploration game (P3), and procedural content generation (PCG) (P6), among others. Some of these examples (e.g., game jams and PCG) were surprising to us, in that they were not scenarios we had imagined as possible use cases in designing the framework. This suggests that the framework has achieved, at least in part, its goal of extendability and applicability to diverse game projects.

Regarding framework integration, this process can be challenging for any development tool, depending on its learning curve and the level of overhead associated with its installation and any necessary modifications of existing project content. From a design standpoint, we have attempted to minimize the burden of the framework for new users on each of these points (e.g., by providing drag-and-drop agent "prefabs" in Unity, enabling customization through Unity's Inspector, and requiring no modification of existing game code). Participants largely confirmed that we were successful in achieving this goal (P3: "extremely easy to use", P4: "worked out of the box", P6: "not cause any issue because it's not actually modifying any game objects").

Pivoting to our central inquiry, the utility of PathOS in practice, we can consider the broad intended use of the framework, as well as specific tasks that the tool can support during the design process. Our original intent with PathOS was to serve as a surrogate of sorts for playtesting in the early stages of design, between testing iterations, or in other scenarios where a lack of resources or other limitations made human testing infeasible. Although most participants agreed with this sentiment, many expressed that they felt the tool could be equally useful for QA, with two participants stating they felt the tool was actually more useful for QA applications (see subsection 5.1). This was a surprising and welcome result, as we had not considered the potential QA implications of PathOS.

Some of the QA tasks suggested by participants included checking for collision glitches, coverage testing of NPC interactions, and adding performance data (e.g., framerate, GPU load) to visualizations. Given that seven out of nine participants suggested that the tool could potentially be useful for QA even in its current state, this application could represent a secondary "target market" of sorts for PathOS within the development community. Further development of the tool to support tasks such as those described here is thus a worthwhile avenue to explore in future work.

More in line with the primary focus of the framework, when discussing UX-oriented goals, participants repeatedly returned to the idea of validating level "flow", or the path taken by players/agents through a game's world. Specific tasks performed by participants during the design exercise included identifying and repairing unintended shortcuts, opening up new paths to support greater player freedom, placing game objects to complement predicted player behaviours (e.g., placing health pickups at locations visited by agents prior to combat interactions), and simply confirming that players could follow the intended path to "finish" a given level. Participants supported their work in completing these tasks both through real-time observation of agent behaviour and by visualizing data from agents simulated in groups.

Some participants preferred to watch agents in real time, commenting on the completeness of the information overlay available, whereas others focused almost exclusively on visualization, highlighting the heatmap and path visualizations. In both cases, participants felt that the insights they were able to extract were useful, suggesting that both modes of operation are helpful depending on the individual and their current task or design context.

When discussing the framework's utility in a general sense, participants commented positively on the information they were able to obtain (e.g., P6: "got a lot of feedback from the AI about the layout of the level"). In discussing the value of the tool for developers, participants cited its ability to save time (P3: "see results [and] take action on those results, and it's been five minutes") and financial resources (P7: "a good tool for people who want to start with user research but don't have the funds"). Participants also suggested that the tool could help alleviate challenges associated with recruitment, such as a lack of participants (P3: "pool of potential playtesters [for an indie project] is extremely limited"), or concerns of confidentiality (P5: "[...] NDA secure products so you're pretty hesitant to show it to players. So this might be a really good supplement to that").

We can summarize our findings regarding PathOS' contribution to the development process as follows: PathOS provides value for developers by offering an approximation of player behaviour which can be used to extract actionable feedback at an extreme reduction in resource requirements compared to human usertesting.

6.2 How can PathOS be improved?

As discussed in subsection 5.3, we have compiled a list of changes discussed with participants to improve PathOS, as well as more major additions to the framework's functionality. Here, we will focus on the more substantial enhancements to the tool, with the chief goals of enriching the simulation of player interactions, and expanding support for dynamic game environments.

Currently, agent interaction in PathOS is divorced from game mechanics; when agents "visit" an entity in the game world, no game logic is executed. While this reduces the framework's technical footprint on a project, it limits users' ability to understand how those interactions may affect player behaviour over time (e.g., a series of difficult fights causing even aggressive players to adapt stealthy behaviours). To support such cases, participants suggested including an API for developers to add in custom behaviours (e.g., combat simulation) when agents interact with game objects. Additionally, it was suggested that the resources available to players could be made available to agents, with designer-customizable resources that could be affected by specific interactions (e.g., losing health after interacting with an enemy, and gaining health after interacting with a first-aid kit). Conceivably, such a system could be used to support reinforcement learning as a driver for agent behaviour, in conjunction with the current destination planner.

In addition to allowing for richer agent interaction, the framework could also be extended to better adapt to the dynamic nature of game worlds, where the interactive content of a level may substantially change as a result of player behaviours (e.g., killing an enemy results not only in the enemy dying, but in a new area becoming accessible). Changes suggested to support this goal include support for "spawners" that produce entities during runtime, and conditional logic for the availability of interactive objects (e.g., talking to an NPC results in a new optional goal for players).

All of these additions have a common goal: promoting functional customizability of the framework, allowing it to more specifically adapt to individual game projects with fleshed-out mechanics and level logic. This could enable designers to answer more complex questions about player behaviour and user experience, for instance, simulation of combat could allow for a basic assessment of difficulty in addition to level flow. This could also serve to extend the utility of the tool beyond the early stages of the level design process, making it more useful as a tool to be applied throughout development.

7 LIMITATIONS AND FUTURE WORK

The PathOS tool is still fundamentally a proof-of-concept for automated testing, rather than a final product. The chief limitation of this work is that we cannot make definitive claims regarding the accuracy of PathOS' behavioural prediction, since we have yet to compare agent behaviour with data from human players. Future work will need to provide an empirical validation of our behavioural model with data from human players. By testing levels with both human players and PathOS agents, we may use data from human players to assist in tuning parameters of our simulation model. Past work in user prediction for interface navigation demonstrated the use of approximate bayesian computation as a means to improve model accuracy based on data from real users [25]. An additional consideration specific to this work would be aligning the motivation profiles of simulated agents with their human counterparts, perhaps by employing existing player typology surveys (e.g., [40]).

We are also limited in our ability to assess the long-term use and potential value of this tool, since the present study only examined short-term use in practise. Furthermore, our evaluation here focuses on the subjective perception of PathOS' utility. A more complete evaluation could include a comparison of the design process and its output between designers with and without access to the framework. Comparative measures might include the time taken to create and refine different level designs, and the perceived quality of created levels by human participants. This would allow us to better understand where the tool's strengths lie in terms of reducing labour requirements versus improving design quality.

Our current prototype is inherently limited by the simplicity of its perceptual model, which is based on level geometry and does not account for the audiovisual cues that influence player navigation. Future work could examine the use of computer vision techniques to better model how human players perceive a game's world. Another limitation is that interactive objects are treated as static; that is to say, fixed in position and always-available. Interactions with these objects (e.g., puzzle-solving, combat) is also abstracted, which means that the model cannot account for behavioural differences resulting from the outcome of an interaction and its cognitive load on players. In the early stages of design, prior to the implementation of game logic, this can be an adequate approximation for the tool's intended purpose. However, for the tool to be useful throughout the development process, it will need to better support the dynamic nature of "real" game environments.

7.1 Future Work

Moving forward, further development on the PathOS framework will focus on enhancing its functionality and implementing changes to improve its usability and support user efficiency. Our first steps will be to implement the improvements derived from discussion with study participants, outlined in subsection 6.2. With respect to framework additions, our top priority will be the inclusion of support for dynamic game entities and custom programming of agent behaviours, to make the tool more suitable for longer-term use with individual game projects. Eventually, we hope to explore opportunities to augment the complexity of the framework's perceptual and behavioural models, such as using computer vision to "play from pixels" to better evaluate the visibility and visual design of interactive entities for human players, or using reinforcement learning to help govern agent behaviour.

Future studies with the framework should aim to evaluate the accuracy of agent behaviour, as described at the beginning of this section. Additionally, case studies with commercial game studies could be conducted to assess the framework's role in the development process over a longer term, as well as its adaptability to different projects in practise. Lastly, such long-term studies could help us to determine what modifications or extensions to the framework would be most beneficial to developers in a real-world context.

8 CONCLUSION

The evaluation of user experience in games is a difficult task, made all the more challenging by the increasing complexity and size of game systems and virtual worlds. Playtesting is critical in determining how real users will experience a game, but carries significant costs in terms of labour and financial resources and can only be conducted when a playable build is ready (often tool late to make fundamental design modifications). Additionally, participant recruitment can be a lengthy process that may prove detrimental in the already compressed timelines of development iteration.

To overcome these challenges, developers and game researchers are exploring the use of automated techniques to expedite the process of playtesting, or serve as a supplement to existing methods. In the spirit of such endeavours, we created PathOS, an open-source framework for predicting player navigation in video games by modelling human perception, memory, and decision-making in a virtual environment. A preliminary evaluation of our tool with developers found that it succeeded in providing users with information they found valuable to their design process.

In future development and evaluation of the PathOS tool, we hope to augment its functionality to support better customization for individual projects, and collaborate with game studios to assess its use in commercial contexts. With respect to AI-driven testing in general, the novel nature of such work makes it difficult to speculate how exactly it will evolve in the coming years. It is our hope that efforts such as ours will help to make games user research more accessible, empowering developer creativity and providing even more imaginative worlds for players to explore and enjoy.

ACKNOWLEDGMENTS

Samantha Stahlke would like to thank Svetlana Stahlke for her mentorship and support. This work has been supported by the Ontario Tech University Research Excellence Award, the Vector Institute, and the Natural Sciences and Engineering Research Council of Canada (NSERC Discovery GPIN-2014-05763).

REFERENCES

[1] Sinan Ariyurek, Aysu Betin-Can, and Elif Surer. 2019. Automated Video Game Testing Using Synthetic and Human-Like Agents. *IEEE Transactions on Games* (2019), 21 pp. https://doi.org/10.1109/TG.2019.2947597

[2] R C Atkinson and R M Shiffrin. 1971. The control of short-term memory. *Scientific American* 225, 2 (1971), 82–90. http://www.ncbi.nlm.nih.gov/pubmed/5089457

[3] Pierre Barrouillet and Valérie Camos. 2012. As Time Goes By: Temporal Constraints in Working Memory. *Current Directions in Psychological Science* 21, 6 (2012), 413–419. https://doi.org/10.1177/0963721412459513

[4] Richard Bartle. 1996. Hearts, Clubs, Diamonds, Spades: Players who suit MUDs. *Journal of Virtual Environments* 1, 1 (1996), 19 pp. http://mud.co.uk/richard/hcds.htm

[5] Keshav Bimbraw. 2015. Autonomous Cars: Past, Present and Future - A Review of the Developments in the Last Century, the Present Scenario and the Expected Future of Autonomous Vehicle Technology. *ICINCO 2015 - 12th International Conference on Informatics in Control, Automation and Robotics, Proceedings* 1 (01 2015), 191–198. https://doi.org/10.5220/0005540501910198

[6] Staffan Bjork and Jussi Holopainen. 2004. *Patterns in Game Design.* Charles River Media, Rockland, MA, USA.

[7] Igor Borovikov and Ahmad Beirami. 2018. Imitation Learning via Bootstrapped Demonstrations in an Open-World Video Game. In *2018 NeurIPS Workshop on Reinforcement Learning under Partial Observability*. 3 pp.

[8] Claire Bradley and Joel Pearson. 2012. The Sensory Components of High-Capacity Iconic Memory and Visual Working Memory. *Frontiers in Psychology* 3 (2012), Article No. 355. https://doi.org/10.3389/fpsyg.2012.00355

[9] Eli T. Brown, Alvitta Ottley, Helen Zhao, Quan Lin, Richard Souvenir, Alex Endert, and Remco Chang. 2014. Finding Waldo: Learning about Users from their Interactions. *IEEE Transactions on Visualization and Computer Graphics* 20, 12 (2014), 1663–1672. https://doi.org/10.1109/TVCG.2014.2346575

[10] Michael Buro. 2003. *The Evolution of Strong Othello Programs.* Springer US, Boston, MA, 81–88. https://doi.org/10.1007/978-0-387-35660-0_10

[11] Murray Campbell, a. Joseph Hoane Jr., and Feng-hsiung Hsu. 2002. Deep Blue. *Artificial Intelligence* 134, 1-2 (2002), 57–83. https://doi.org/10.1016/S0004-3702(01)00129-1

[12] Yun-Gyung Cheong, Arnav Jhala, Byung-Chull Bae, and R. Michael Young. 2008. Automatically Generating Summary Visualizations from Game Logs. In *Proceedings of AIIDE 2008*. AAAI Press, Stanford, California, USA, 167–172. https://doi.org/10.5555/3022539.3022570

[13] Nelson Cowan. 2001. The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences* 24, 1 (2001), 87–114.

[14] Benjamin Cowley and Darryl Charles. 2016. Behavlets: a method for practical player modelling using psychology-based player traits and domain specific features. *User Modelling and User-Adapted Interaction* 26, 2-3 (2016), 257–306. https://doi.org/10.1007/s11257-016-9170-1

[15] Fernando de Mesentier Silva, Igor Borovikov, John Kolen, Navid Aghdaie, and Kazi Zaman. 2018. Exploring Gameplay with AI Agents. In *Proceedings of AIIDE 2018*. AAAI, 7 pp. arXiv:1811.06962.

[16] Anders Drachen, Christian Thurau, and Christian Bauckhage. 2013. A Comparison of Methods for Player Clustering via Behavioral Telemetry. In *Proceedings of FDG 2013*. 245–252. arXiv:1407.3950.

[17] Brandon Drenikow and Pejman Mirza-Babaei. 2017. Vixen: Interactive Visualization of Gameplay Experiences. In *Proceedings of FDG 2017 (FDG '17)*. ACM, 3:1–3:10. https://doi.org/10.1145/3102071.3102089

[18] Tracy Fullerton. 2008. *Game Design Workshop: A Playcentric Approach to Creating Innovative Games.* https://doi.org/10.1007/s13398-014-0173-7.2

[19] Karl R. Gegenfurtner and George Sperling. 1993. Information transfer in iconic memory experiments. *Journal of Experimental Psychology: Human Perception and Performance* 19, 4 (1993), 845–866. https://doi.org/10.1037/0096-1523.19.4.845

[20] David Gotz and Zhen Wen. 2009. Behavior-Driven Visualization Recommendation. In *Proceedings of IUI 2009*. 315–324. https://doi.org/10.1145/1502650.1502695

[21] Arthur Juliani. 2017. Introducing: Unity Machine Learning Agents Toolkit. https://blogs.unity3d.com/2017/09/19/introducing-unity-machine-learning-agents/ Library Catalog: blogs.unity3d.com.

[22] Rousslan Fernand Julien Dossa, Xinyu Lian, Hirokazu Nomoto, Takashi Matsubara, and Kuniaki Uehara. 2019. A Human-Like Agent Based on a Hybrid of Reinforcement and Imitation Learning. In *Proceedings of IJCNN 2019*. IEEE, Budapest, Hungary, 1–8. https://doi.org/10.1109/IJCNN.2019.8852026

[23] Niels Justesen, Philip Bontrager, Julian Togelius, and Sebastian Risi. 2020. Deep Learning for Video Game Playing. *IEEE Transactions on Games* 12, 1 (March 2020), 1–20. https://doi.org/10.1109/TG.2019.2896986 Conference Name: IEEE Transactions on Games.

[24] Adam S. Kahn, Cuihua Shen, Li Lu, Rabindra A. Ratan, Sean Coary, Jinghui Hou, Jingbo Meng, Joseph Osborn, and Dmitri Williams. 2015. The Trojan Player Typology: A cross-genre, cross-cultural, behaviorally validated scale of video game play motivations. *Computers in Human Behavior* 49 (2015), 354–361. https://doi.org/10.1016/j.chb.2015.03.018

[25] Antti Kangasrääsiö, Kumaripaba Athukorala, Andrew Howes, Jukka Corander, Samuel Kaski, and Antti Oulasvirta. 2017. Inferring Cognitive Models from Data using Approximate Bayesian Computation. In *Proceedings of CHI 2017 (CHI '17)*. ACM, Denver, Colorado, USA, 1295–1306. https://doi.org/10.1145/3025453.3025576

[26] Oleksandra Keehl and Adam M. Smith. 2018. Monster Carlo: An MCTS-based Framework for Machine Playtesting Unity Games. In *Proceedings of CIG 2018*. IEEE, Maastricht, 8 pp. https://doi.org/10.1109/CIG.2018.8490363

[27] Oleksandra Keehl and Adam M. Smith. 2019. Monster Carlo 2: Integrating Learning and Tree Search for Machine Playtesting. In *Proceedings of CoG 2019*. IEEE, London, United Kingdom, 8 pp. https://doi.org/10.1109/CIG.2019.8847989

[28] David Kieras. 2004. GOMS Models for Task Analysis. In *The Handbook of Task Analysis for Human-Computer Interaction*. Lawrence Eribaum Associates Inc., Mahwah, New Jersey, USA, 83–116.

[29] Antonios Liapis, Christoffer Holmgard, Georgios N. Yannakakis, and Julian Togelius. 2015. Procedural personas as critics for dungeon generation. In *Lecture Notes in Computer Science*, Vol. 9028. 331–343. https://doi.org/10.1007/978-3-319-16549-3_27

[30] Vincent Di Lollo. 1977. Temporal characteristics of iconic memory. *Nature* 267, 5608 (May 1977), 241. https://doi.org/10.1038/267241a0

[31] Wei Ji Ma, Masud Husain, and Paul M. Bays. 2014. Changing concepts of working memory. *Nature Neuroscience* 17 (2014), 347–356. https://doi.org/10.1038/nn.3655

[32] Tobias Mahlmann, Anders Drachen, Julian Togelius, Alessandro Canossa, and Georgios N. Yannakakis. 2010. Predicting Player Behavior in Tomb Raider: Underworld. In *Proceedings of CIG 2010*. IEEE, 178–185. https://doi.org/10.1109/ITW.2010.5593355

[33] Thomas W Malone. 1981. Toward a Theory of Intrinsically Instruction Motivating. *Cognitive Science* 5, 4 (1981), 333–369. https://doi.org/10.1207/s15516709cog0504_2

[34] Regan L. Mandryk and M. Stella Atkins. 2007. A fuzzy physiological approach for continuously modeling emotion during interaction with play technologies. *International Journal of Human Computer Studies* 65, 4 (2007), 329–347. https://doi.org/10.1016/j.ijhcs.2006.11.011

[35] David Mason and Asko Relas. 2019. How Rovio teaches Angry Birds to fly in the cloud using ML. https://www.youtube.com/watch?v=U-NsVcDKU0Y

[36] Ben Medler, Michael John, and Jeff Lane. 2011. Data Cracker: Developing a Visual Game Analytic Tool for Analyzing Online Gameplay. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 2365–2374. https://doi.org/10.1145/1978942.1979288

[37] David Melhart, Ahmad Azadvar, Alessandro Canossa, Antonios Liapis, and Georgios N. Yannakakis. 2019. Your Gameplay Says It All: Modelling Motivation in Tom Clancy's The Division. In *Proceedings of CoG 2019*. IEEE, 8 pp. https://doi.org/10.1109/CIG.2019.8848123

[38] Earl K. Miller and Timothy J. Buschman. 2015. Working memory capacity: Limits on the bandwidth of cognition. *Daedalus* 144, 1 (2015), 112–122. https://doi.org/10.1162/DAED_a_00320

[39] Luvneesh Mugrai, Fernando de Mesentier Silva, Christoffer Holmgård, and Julian Togelius. 2019. Automated Playtesting of Matching Tile Games. In *Proceedings of CoG 2019*. IEEE, London, United Kingdom, 7 pp. https://doi.org/10.1109/CIG.2019.8848057 arXiv: 1907.06570.

[40] Lennart E Nacke, Chris Bateman, and Regan L Mandryk. 2014. BrainHex: A Neurobiological Gamer Typology Survey. *Entertainment Computing* 5, 1 (2014), 55–62. https://doi.org/10.1016/j.entcom.2013.06.002

[41] Fredrik E B Norberg and Zelal Yildirim. 2018. The implementation of Voice Command in Smart Homes.

[42] Aristea Papadimitriou. 2016. *The Future of Communication: Artificial Intelligence and Social Networks.* Master's thesis. Malmö University.

[43] Falko Weigert Petersen, Line Ebdrup Thomsen, Pejman Mirza-Babaei, and Anders Drachen. 2017. Evaluating the Onboarding Phase of Free-ToPlay Mobile Games: A Mixed-Method Approach. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play* (Amsterdam, The Netherlands) *(CHI PLAY '17)*. Association for Computing Machinery, New York, NY, USA, 377–388. https://doi.org/10.1145/3116595.3125499

[44] Johannes Pfau, Jan David Smeddinck, and Rainer Malaka. 2017. Automated Game Testing with ICARUS: Intelligent Completion of Adventure Riddles via Unsupervised Solving. In *Proceedings of CHI PLAY 2017 Extended Abstracts*. 153–164. https://doi.org/10.1145/3130859.3131439

[45] Timothy J. Ricker, Evie Vergauwe, and Nelson Cowan. 2016. Decay theory of immediate memory: From Brown (1958) to today (2014). *Quarterly Journal of Experimental Psychology* (2016). https://doi.org/10.1080/17470218.2014.914546

[46] Shaghayegh Roohi, Jari Takatalo, J. Matias Kivikangas, and Perttu Hämäläinen. 2018. Neural Network Based Facial Expression Analysis of GameEvents: A Cautionary Tale. In *Proceedings of CHI PLAY 2018 (CHI PLAY '18)*. ACM, Melbourne, VIC, Australia, 429–437. https://doi.org/10.1145/3242671.3242701

[47] Mike Schaekermann, Giovanni Ribeiro, Guenter Wallner, Simone Kriglstein, Daniel Johnson, Anders Drachen, Rafet Sifa, and Lennart E. Nacke. 2017. Curiously Motivated: Profiling Curiosity with Self-Reports and Behaviour Metrics

in the Game "Destiny". In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play* (Amsterdam, The Netherlands) *(CHI PLAY '17)*. Association for Computing Machinery, New York, NY, USA, 143–156. https://doi.org/10.1145/3116595.3116603

[48] Mohammad Shaker, Noor Shaker, and Julian Togelius. 2013. Evolving Playable Content for Cut the Rope through a Simulation-Based Approach. In *Proceedings of AIIDE 2013*. 72–78.

[49] Samantha Stahlke and Pejman Mirza-Babaei. 2018. Usertesting Without the User: Opportunities and Challenges of an AI-Driven Approach in Games User Research. *ACM Comput. Entertain* 16, 2 (2018), 18 pp. https://doi.org/10.1145/3183568

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

[51] Chek Tien Tan, Tuck Wah Leong, and Songjia Shen. 2014. Combining Think-Aloud and Physiological Data to Understand Video Game Experiences. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 381–390. https://doi.org/10.1145/2556288.2557326

[52] Ruck Thawonmas and Masayoshi Kurashige. 2007. Detection of landmarks for clustering of online-game players. *The International Journal of Virtual Reality* 6, 3 (2007), 11–16. http://140.109.19.187/pub/thawonmas07_landmark_ijvr.pdf

[53] Kristinn Thórisson and Helgi Helgasson. 2012. Cognitive Architectures and Autonomy: A Comparative Review. *Journal of Artificial General Intelligence* 3, 2 (2012), 1–30. https://doi.org/10.2478/v10229-011-0015-3 Publisher: Sciendo Section: Journal of Artificial General Intelligence.

[54] Jonathan Tremblay, Pedro Andrade Torres, Nir Rikovitch, and Clark Verbrugge. 2013. An Exploration Tool for Predicting Stealthy Behaviour. In *Proceedings of AIIDE 2013*. 34–40.

[55] Anders Tychsen and Alessandro Canossa. 2008. Defining personas in games using metrics. In *Proceedings of Future Play 2008*. 73–80. https://doi.org/10.1145/1496984.1496997

[56] Jukka Vahlo, Johanna K. Kaakinen, Suvi K. Holm, and Aki Koponen. 2017. Digital Game Dynamics Preferences and Player Types. *Journal of Computer-Mediated Communication* (2017). https://doi.org/10.1111/jcc4.12181

[57] H.Jaap [van den Herik], Jos W.H.M. Uiterwijk, and Jack [van Rijswijck]. 2002. Games solved: Now and in the future. *Artificial Intelligence* 134, 1 (2002), 277 – 311. https://doi.org/10.1016/S0004-3702(01)00152-7

[58] Günter Wallner, Nour Halabi, and Pejman Mirza-Babaei. 2019. Aggregated Visualization of Playtesting Data. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, Article 363, 12 pages. https://doi.org/10.1145/3290605.3300593

[59] Günter Wallner and Simone Kriglstein. 2012. A Spatiotemporal Visualization Approach for the Analysis of Gameplay Data. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Austin, Texas, USA) *(CHI '12)*. Association for Computing Machinery, New York, NY, USA, 1115–1124. https://doi.org/10.1145/2207676.2208558

[60] Ben G. Weber and Michael Mateas. 2009. A data mining approach to strategy prediction. In *Proceedings of CIG 2009*. 140–147. https://doi.org/10.1109/CIG.2009.5286483

[61] Nick Yee. 2006. Motivations of Play in MMORPGs - Results from a Factor Analytic Approach. *CyberPsychology & Behavior* 9, 6 (2006), 772–775. https://doi.org/10.1089/cpb.2006.9.772