Proceedings of the Fourteenth Artificial Intelligence and
Interactive Digital Entertainment Conference (AIIDE 2018)

Modelling Player Understanding of Non-Player Character Paths

Mengxi Xoey Zhang, Clark Verbrugge
McGill University
Montréal, Canada
xoey.zhang@mail.mcgill.ca, clump@cs.mcgill.ca

Abstract

Modelling a player's understanding of NPC movements can be useful for adapting gameplay to different play styles. For stealth games, what a player knows or suspects of enemy movements is important to how they will navigate towards a solution. In this work, we build a uniform abstraction of potential player path knowledge based on their partial observations. We use this representation to compute different path estimates according to different player expectations. We augment our work with a user study that validates what kinds of NPC behaviour a player may expect, and develop a tool that can build and explore appropriate (expected) paths. We find that players prefer short simple paths over long or complex paths with looping or backtracking behaviour.

Introduction

The difficulty of a stealth or combat game is strongly affected by player knowledge of enemy movements. Previous playthroughs, outside knowledge, as well as general gaming experience allow players to make increasingly strong assumptions about non-player character (NPC) behaviours, and thus better predict or identify safer or more strategic movement options in game levels. Modelling player assumptions and expectations of NPC movements provides an insightful mechanism for understanding how players may attempt to solve a level, particularly over repeated play. It also offers potential for adapting game difficulty by dynamic modifications that either conform to or violate expectation.

In this work we describe an algorithmic approach to estimating and representing player knowledge of NPC movements. We base our design on a uniform representation of partial observations, applying algorithmic and heuristic techniques to build different estimations of position and expected movement. We do not aim at open, unbounded prediction; rather, our focus is on how players may fill the gap between two observations, reflecting the use of prior, partial knowledge that provides consistent, but not identical information, as may be acquired from repeated playthroughs.

The full extent of individual player knowledge is of course beyond what can be acquired from simple game logging. Our approach is aimed at building the core techniques that enable an exploration tool to better understand and experiment with what information becomes available to a player, given specific NPC routes, level geometry, and incomplete observation. We thus define a baseline system that allows for exhaustive modelling of possible NPC positions, constrained by the gap-time and filtered by knowledge of level geometry. Players may also attribute movement characteristics or make assumptions about NPC behaviours as well. Our design naturally incorporates different constraint models that reduce pathing possibilities to better represent player expectation.

Algorithmic and representational design is supported by a non-trivial experimentation and visualization tool built into Unity®. This allows for flexible exploration of different, constrained models of player expectation given different segments of NPC observation. The choice of path constraints we offer is further justified by data gathered from a small user study.

Our work is intended to facilitate design exploration of NPC movements and geometry, and to provide a tool for further understanding of how gameplay may be learned and optimized through repeated player experience. Specific contributions of this work include the following:

- We construct a generic model of player path observations which accommodates a variety of possible observation sources and expectation constraints.
- We provide an algorithm for computing an exhaustive representation of possible movements, as well as extensions that either work through our formal representation or incorporate simple constraints to reduce the set of possible behaviours according to different models of player expectation.
- Our approach is implemented and demonstrated in a non-trivial tool within the open-source Unity® framework. This allows for flexible visualization and exploration of expected movement under different constraints.
- Our design approach is supported by a small-scale user study that shows players generally expect NPC to follow simple, non-looping paths with minimal backtracking.

Background and Related Work

Stealth is one of many genres of video games that calls for players to develop and employ strategies to avoid dynamic, patrolling or static enemy agents. This style of video game typically requires the player to hide from the non-player enemies in order to successfully achieve a goal.

In such games NPCs frequently behave in a predetermined and predictable way. This presents a puzzle context, where (among other gameplay mechanisms) the player uses current in-game and previous observations to predict future NPC movements in order to help evade enemies or hide from them. This knowledge may come from current, in-game observations, observations made in previous playthroughs, as well as from external resources and general player experience in other, similar games. Accumulation of this knowledge is essential to the strategic decisions made in solving a stealth level.

Existing research in path modelling is primarily focused on understanding and predicting player motion in order to control NPC behaviour. This is useful for designing agents that can interact with users creatively (Singh et al. 2016), apply basic strategy to intercept enemies (Tastan, Chang, and Sukthankar 2012), attempt to appear "human-like" in understanding how and where a player may move (Hladky and Bulitko 2008), and (going outside of games) as a means of estimating pedestrian behaviour (Yin et al. 2016). Models of human motion are often based on learning AI systems, but may also incorporate other approaches, such as use of particle systems for estimating occluded position (Weber, Mateas, and Jhala 2011), and steering systems (Tastan and Sukthankar 2011) for physically motivated movement constraints. Further, heuristic constraints can be derived based on looking for similarities in observed movement pattern (Aparicio, IV and Caban 2009), as well as gameplay-specific properties, such as a quantified notion of the "danger" or "risk" inherent to, and thus affecting different path choices (Tremblay, Torres, and Verbrugge 2014).

Modelling the "opposite" perspective, that is what a player expects of NPC movement, is less common, although possible with generic planning systems that do not make assumptions about human or AI agency (Geib et al. 2016). Particle approaches and other techniques that generate probabilistic occupancy maps can also be used for making position estimates. The stealth game *Third Eye Crime* (Isla 2013), for example, is well known for incorporating positive and negative knowledge from an AI's perceptual system (Isla and Blumberg 2002) into the gameplay, although again in terms of modelling the player. Since players may also attribute human-like properties to NPC movement (Heider and Simmel 1944; Roemmele et al. 2016), modelling players and modelling NPCs from a player perspective likely has some degree of synergy, although stealth NPC movement tends to have predictable, algorithmic properties, designed to fit the game play or a visual objective.

Other, more widely scoped models of player behaviour have also been performed. Knowledge of a player's tactics or strategy, for example, allows for a user-specific AI response, whether the goal is to make a game more competitive or more entertaining (Bakkes, Spronck, and van Lankveld 2012). This of course extends broadly, motivating a generic taxonomy (Smith et al. 2011) to help clarify the many approaches. Our approach is in this terminology a universal, descriptive reaction model, interpreted, although also grounded in user data.

Path prediction and reconstruction can also be applied to real-life situations beyond video games. This includes diverse contexts, such as security monitoring, using a dynamic oriented graph to identify moving objects and help predict abnormal behaviour (Duque, Santos, and Cortez 2007), as well as in observing individual animal behaviour, using state–space modelling to study animal movement, biogeography and spatial population dynamics (Patterson et al. 2008).

Representing Movement Knowledge

Knowledge of enemy positions and movements can be acquired from various sources. Wikis and strategy guides may give locations and routes (with widely varying levels of detail), 3rd party game videos provide partial sources, previous attempts by the player can provide multiple sets of observations, and even single-game observations of cyclic behaviour can give multiple data points on the same route.

Representing and combining this information introduces an initial challenge. Our approach is to use a uniform model that reduces knowledge of path behaviour to a series of specific, ordered and time-stamped observations, no matter the source. We assume (repeated, deterministic) enemy movement is partially captured by segments of prior observation. This allows us to structure player expectation in relation to their knowledge as a process of filling in the (unknown) gaps between (known) segments, for which we can use different algorithmic solutions. Search for more constrained solutions within this space is then bounded, and the same formalism also lets us easily incorporate a variety of constraints that may heuristically limit the scope of expectation.

In this section we formalize our basic representation, building a model that incorporates both knowledge and its absence. The section following then shows how expectation can be computed, and limited by adding assumed observations and imposing search constraints.

Segment and Path Abstraction

Our game domain Σ is a discrete two-dimensional grid, extended into a continuous, positive time dimension: $\Sigma \subseteq \mathbb{N}^2 \times \mathbb{R}^+$. NPCs follow polygonal, obstacle-avoiding routes through the space, and at each point in time they have a defined orientation (field of view). We form observations of an NPC movement as a series of *points* that record position, time, and orientation. We will refer to elements of point p using the notation $p(x)$, $p(y)$, $p(t)$, and $p(\theta)$.

Generally, an observation is continuous over a span of time, implying an infinite set of data points. We are, however, mainly interested in points where the observed target changes their behaviour. We thus structure path observation into *segments*, assuming constant motion or rotation between two points, and creating new segments when behaviour changes. Observation is not assumed to be complete, and thus a series of observed segments will have gaps, starting when the player loses track of their target and ending when they regain sight of the same target.

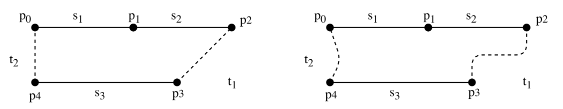


Figure 1: Example of a partial path with different ways to fill in unknown gap segments

Definition 1 *A* segment *s is a pair of points $\langle p_a, p_b \rangle$ where $p_a(t) \leq p_b(t)$.*

Segments are meant to describe ordered pieces of a path, and the condition ensures the start of the segment precedes the end of the segment. We generally refer to p_a as *start*(s) and p_b as *end*(s), i.e. the beginning and end of the segment s respectively.

Segments may occur in two forms, indicating either straight-line fragments of a path (*path* segment), or the gaps between such fragments (*gap* segment). The former constitute subsets of continuous observation, and for simplicity assume that either $(p_a(x), p_a(y)) = (p_b(x), p_b(y))$, or $p_a(\theta) = p_b(\theta)$—i.e., either position changes or orientation changes, but not both. (Neither changes for a waiting agent, but time must increment.) Gap segments represent a period of missing observation that may include multiple, arbitrary motions, and are thus only constrained by advancing time.

The duration of a segment is important to understanding potential movements. In this work, we will commonly refer to the time elapsed between *start(s)* and *end(s)* as $\Delta t(s)$, or just Δt if s is implied. The length of a path segment is given by as *length(s)*, and we refer to $\Delta t_{min}(s)$ (or Δt_{min}) as the minimum time required to get from *start(s)* to *end(s)*, at maximum speed.

Segments can be composed into *valid paths* by forming ordered sequences.

Definition 2 *A* valid path *is an ordered list of $n \geq 1$ consecutive path segments $S = \{s_0, s_1, \cdots, s_{n-1}\}$, such that end$(s_i)$ = start(s_{i+1}), $\forall i < n - 1$. A valid path may be cyclic if end(s_{n-1}) = start(s_0), assuming modulo time.*

A given valid path may form a *complete path*, in which case all segments are path segments. We are of course more interested in *partial* paths, which contain gaps.

Definition 3 *A* partial path *is a valid path including at least one gap segment, starting with a path segment, and without consecutive gap segments. The final segment may only be a gap segment in the case of cyclic paths.*

We require gap segments to be bracketed by path segments in order to bound the scope of non-observation. Note as well that the condition on consecutive gaps is a simplification, not a constraint, since adjacent gap segments can be trivially merged and reduced to a single gap segment.

Figure 1 shows two (cyclic) partial paths, each with two gap segments t_1, t_2. The dotted lines show possible behaviours that may have occurred during the gaps. The idea is that, although the path segments of the two partial paths are identical, the gap segments can be filled in with different potential path segments.

NPC Movements Our design rests on the idea of knowledge being acquired incrementally, building expectation up based on past observation. This reflects the origin of our interest in stealth and combat games, where in order to present a solvable puzzle enemy NPCs often behave in a predictable way that forms a repeated pattern of movement. Even within this, however, different movement models can be used. We assume a simple model: NPCs can only either move forward at a fixed speed, rotate while standing at a single spot, or wait. Actual movement follows a simple Manhattan movement model, making discrete steps to adjacent tiles in the grid space. More complex movement models with variable speed, combining rotation and moving, and so forth are left for future work.

Expectation: Filling in the Gaps

Once we have a valid partial path, we can build models that estimate NPC position, filling in gap segments with possible path segments. Below we discuss different approaches, as well as a tool we built to visualize the results.

Conservative Approach Our baseline approach gives a full representation of every position the NPC could have occupied. We iterate through each gap segment g_i in the partial path. Given that the amount of time elapsed $\Delta t(g_i)$, and naïvely assuming that the NPC did not vary in displacement speed, we can calculate all potential positions the guard may have occupied between, and for how long.

Our method associates each position with labels, one representing the earliest time an NPC may arrive at that location having departed from the start of the gap segment, and one for the latest time they must leave it in order to reach the end point of the gap segment.

The labelling algorithm is essentially a BFS performed on both the beginning and end points separately. We flood out from the starting point *start*(g_i), computing for each tile a value l_a as the minimum time the tile could be entered. Then we perform the same procedure from *end*(g_i), computing l_b as the minimum time required to reach the end. For a given tile τ, we can define $w_\tau = \Delta t(g_i) - (l_a + l_b)$, which gives the maximum time an NPC can possibly wait at that position before continuing if they must reach *end*(g_i) within $\Delta t(g_i)$. By sorting the tiles by their w_τ value and colour-coding them, we can see a heat-map effect. Figure 2 shows an example.

This representation has the advantage of compactly representing reachable locations, while still enabling us to recover the (exponential) set of feasible paths by conducting a search from start to end of the gap segment, entering a neighbouring tile τ only if it satisfies $\Delta t - t_e - l_b > 0$, where t_e is the time taken on the path so far. Note that this computes minimal paths when $\Delta t = \Delta t_{min}$. Figure 3 shows an example of the search space for different Δt, using the 3rd dimension as time. We used blue blocks to show the additional reachable tiles when incrementing Δt.

Conforming to Expectations In most games, stealth games in particular, NPCs move in constrained, structured ways. Guards, for example, are expected to patrol an area, moving efficiently between locations, perhaps with pauses

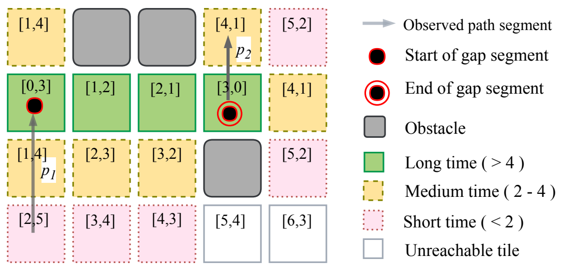


Figure 2: End result of the labelling algorithm showing each *tile* labelled with [l_a,l_b]. Colouring of each tile is based on $\Delta t = 7$ and shows the maximum amount of time the character can wait on that tile.

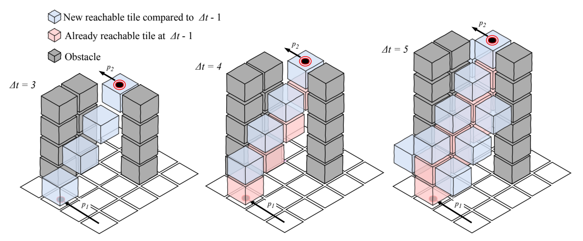

New reachable tile compared to Δt - 1
Already reachable tile at Δt - 1
Obstacle
Δt = 3
Δt = 4
Δt = 5
p_1
p_2

Figure 3: Resulting search space based on increasing the amount of time elapsed without changing the position of start or end of a gap segment.

to look around, but with limited backtracking. The set of paths and positions a player may expect an NPC to traverse is thus less than the set of reachable positions computed in the baseline conservative approach. We want to narrow down the possibilities to paths that are more likely, based on how a player may expect an NPC to actually move.

There are a number of different factors that can affect a player's expectation of possible NPC movement. Optimality is an obvious criterion—given two points, a rational expectation is that the NPC followed a minimal path. When the Δt exceeds Δt_{min}, however, the extra time must be (presumed to be) spent waiting, in detours or loops, or both. Other factors, such as number of turns, proximity to obstacles, may also be considerations.

Figure 4 shows a small example. In this scenario, the two blue paths follow minimal routes and must incorporate some amount of waiting, with the dark blue path also including more rotations. The red path follows a more circuitous route, trading an assumption of waiting for sub-optimal routing.

To better predict the path taken by a NPC, we want to look at a selection of possible paths. As seen in the conservative approach, the larger the difference between Δt and Δt_{min}, the more paths there are to choose from, since for every unit of time added, more possibilities emerge. Ideally, we can search the solution space for paths that are simple first, with a baseline assumption that players expect NPC movements are unlikely to be sporadic, irregular or random.

Simple paths The ideal simple path does not involve doubling back or revisiting tiles multiple times while minimizing number of turns. We augment minimal distance with

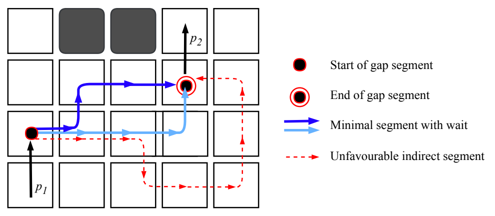


Figure 4: Example of a gap with $\Delta t = 8$. The two blue paths are equally minimal in length (4), but imply the NPC paused at some point to make up the 4 extra time units. The dotted, red path may or be less favourable in being indirect, even if it implies continous movement. Turning here is assumed instantaneous.

a requirement for a minimal number of turns as well—"straight" paths are intuitively simpler. The definition of straight of course depends on the movement model. In our Manhattan-based model, we have many equivalent paths in terms of distance, that may nevertheless be seen as more or less simple depending on how jagged, or turn-intensive they are.

This gives us two properties we can use to reduce the set of expected paths (and thus positions), either minimizing distance, or number of segments of constant movement. Our interpretation of *(ideal) simple* paths combines these, as the subset of minimal distance paths that are also constructed from as few path segments as possible. As previously mentioned, the set of minimal distance paths is easy to compute in our representation by assuming $\Delta t = \Delta t_{min}$. For reasonable gap sizes and obstacle density, reducing that set to minimize segments is also straightforward.

Waiting Other constraints on expected pathing behaviour can be based on similar properties. Pausing or waiting is a potential movement behaviour, common in many stealth contexts. Player expectation may thus also be informed by the existence or distribution of excess travel time. If NPCs do not generally pause (or vary speed), then a gap segment must be assumed to include detours of some form, and overly quick paths will be excluded from the expectation set. Path search in this context cannot terminate unless the time used matches the gap time: $t_e = \Delta t$.

Alternatively, if waiting is a possibility, the distribution of pauses becomes a factor—a character that repeatedly stops and starts is less likely than one that waits less frequently. Tiles that can be part of paths with a single waiting event of total duration w are trivially found as ones for which $w = \Delta t - (l_a + l_b)$, and since (maximum) waiting time is part of our representation we can also easily incorporate different allocations of waiting into a path search.

Interesting Sites A further source of expectation exists in geometric, or content-driven properties of the game space. An NPC guard in a museum may be expected to pass by specific artwork, or frequently return to a chair or guard station. Interest points are clearly game-dependent, and are best manually identified. In our implementation, when no interest point is defined the algorithm assigns corners and

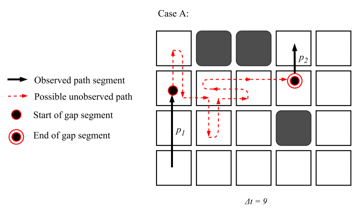


Figure 5: Example of two possible paths to fill the gap segment between observed path segments p_1 and p_2 without wait time.

tiles around obstacles to be points of interest, as heuristically important occlusion and observation points in a stealth context. To calculate a path that favours interest points, we first calculate all reachable interest point tiles τ_i with $\Delta t - (l_a + l_b) > 0$, then we simply consider paths that include interest points where this value is largest, therefore using up time most efficiently.

Interest points may also be incorporated by introducing an artificial path segment, locating the NPC at an interesting site at some time within the gap segment. The time values (and orientation) are less determined, but can be bounded, since for a given tile in the interesting area arrival and departure must be between l_a and $\Delta t - l_b$ respectively.

In the example shown in figure 5, we try to fill in a plausible path for the gap between observed path segments p_1 and p_2, with a duration of $\Delta t = 9$ in this case. Because $t_{min} = 3$ and is significantly smaller than Δt, if waiting has not been previously observed then a shortest path may not be plausible, and we would expect a player to consider longer paths that do not require wait time. The paths shown in cases A and B both fulfill this requirement. The latter, however, both explores more interesting area (the lone obstacle), and has fewer turns, arguably making it more probable.

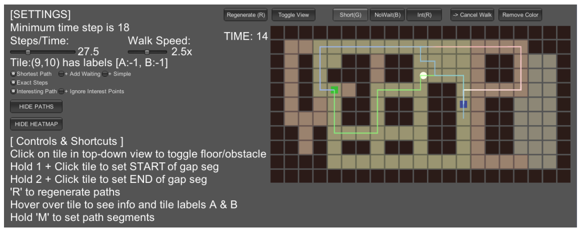
By analyzing previously seen path segments, we can determine whether a player may have observed that an NPC has tendency to take simple paths, interesting paths, or wait, etc. This allows us to focus on specific properties when generating segments used to fill in gap segments.

Geometry A player's awareness of obstacles also of course affects their expectation of pathing possibilities. This is easily incorporated into our approach by relaxing the base level representation—gaps in geometric knowledge reduce path constraints, expanding the underlying conservative set of (expected possible) reachable positions.

Path Segment Generation Tool

The geometry, set of guard motions, and expected player entry and observation points affects how a player will approach a game puzzle. In order to understand the effects of different observations we created a tool in Unity®, which allows us to construct simple game levels and visualize the conservative results. This also allows us to filter and generate example paths segments that respect our various constraints.

Figure 6 shows a screenshot of the interface. The tool expects a discretized level as input, indicating which tiles



[SETTINGS]
Minimum time step is 18
Steps/Time: 27.5
Walk Speed: 2.5x
TIME: 14
Tile:(9,10) has labels [A:-1, B:-1]

[Controls & Shortcuts]
Click on tile in top-down view to toggle floor/obstacle
Hold 1 + Click tile to set START of gap seg
Hold 2 + Click tile to set END of gap seg
'R' to regenerate paths
Hover over tile to see info and tile labels A & B
Hold 'M' to set path segments

Figure 6: A screenshot of the tool built to analyze gap segments.

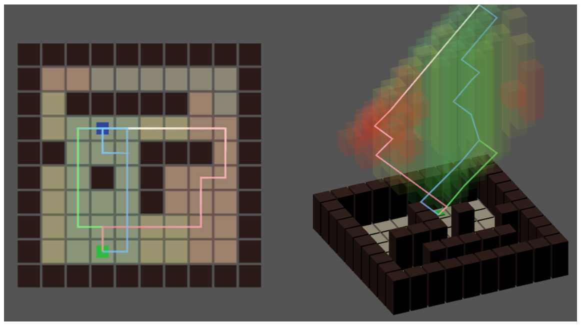


Figure 7: Two separate views: top-down and 3D-rotating side view of paths generated using our tool for the gap segment indicated by the green and blue squares.

represent pathable space and which represent obstacles; for easy experimentation tile designations can also be interactively modified. Within this space one can specify a gap segment by selecting specific start and end tiles, with Δt for the gap adjusted using a slider. This dynamically fills in and colours tiles reachable using the base conservative algorithm, forming a heatmap of possible positions according to maximum waiting time. This can be better viewed in a 3D rotating view, where the user can visualize the length and wait time possible at every tile as a column in the time dimension rendered above the level space.

Individual constraints can then be selected using checkboxes, and a path respecting each resulting property is generated as example routes. Following the properties described earlier, this shows shortest paths, no-waiting paths, and interesting paths. Figure 7 shows examples of three different generated paths for the same gap segment (green for shortest, blue for no-waiting, and red for interesting). The colouring of the vertical columns in the 3D view indicates when the character can enter the tile beneath it, while the height of each is calculated by $\Delta t - (\tau.l_a + \tau.l_b)$. Although the algorithm computes all optimal paths, the program only displays one random path when multiple equivalent ones exist for each selected constraint.

Figure 8 shows a slightly larger example with three generated paths, following different constraints. Within the visualization we can also animate characters following each path, allowing us to see relative differences in movement. We used this visualization as a basis for the different examples shown to users in our human study, below.

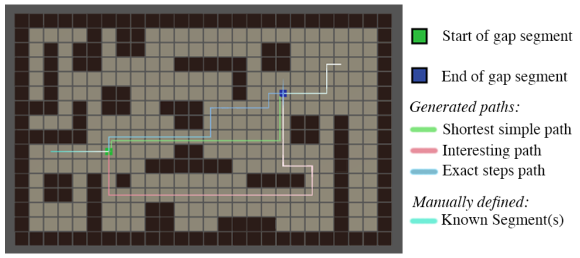


Figure 8: Example of shortest simple path, exact steps and interesting paths generated by our tool on a larger map, with $\Delta t = 26$, manually defined segments and a hidden heatmap.

User Study

We validate our choice of constraints on path expectation through a simple user study. Using the tool that we built, we generated different paths with different features in various level setups and created video recordings to simulate an NPC taking these generated paths. We removed the NPC from view during gap segments and asked survey participants to choose which path the NPC had likely taken during those segments. Letting respondents draw their own paths was not feasible due to the complexity of imposing path and time constraints for each gap segment.

Our survey consisted of 10 questions. Each question presented an option between two generated paths that could have filled the gap segment presented. These paths had a combination of different characteristics: short/long, simple/complex, interesting, with cycles, with backtracking, with or without obstacles, etc. Through a Reddit post, we gathered a total of 104 responses and conducted the study under the hypothesis that players have a preference between different pathing behaviours. The survey was designed to verify whether players preferred certain movement patterns over others, and it is available online for inspection (Zhang and Verbrugge 2018).

Results Table 1 summarizes data from the survey results. As expected, players anticipate NPC movement to be more simple than complex. 75.0% and 68.3% of those surveyed viewed the simple paths as more likely when presented in levels with and without obstacles respectively, given a simple path and a complex path of equal length ($p < .005$). After introducing waiting, the majority still expected the NPC to take the simpler and shorter path, preferring waiting behaviour to more elaborate pathing: 67.7% over long or interesting paths, and 61.8% over paths with loops ($p < .025$).

We also found that 72.1% of respondents were consistent with their preference when picking the most probable path taken, possibly based on their preconceptions of how NPCs should move or behave, whether it is simple or complex. Interestingly, if we show the player a longer path segment before the gap segment where the NPC engages in waiting behaviour, 58.7% picked the path that includes waiting to fill in the gap. When shown a NPC engaging on a longer, more interesting path before the gap, 67.3% chose the longer path to fill in the gap instead ($p < .005$). This indicates that players take into consideration previous observations and what they know does influence their expectations, and we can reject the null hypothesis of players not having a preference in NPC movement.

Table 1: Partial Summary of Survey Results

	Preference			
Features	Obstacle	No Obstacle	With Waiting	
Simple/Short	75.00%	68.27%	67.31%	61.54%
Complex	25.00%	31.73%	-	-
Interesting	-	-	32.69%	-
Looping	-	-	-	38.46%

When participants are given relatively short path segments to observe before the gap segment, those who chose a path with a loop over one with backtracking consists of 55.8% of those surveyed, which does not indicate an obvious preference ($p > .05$). There was also no significant preference between long interesting paths and patrolling back-and-forth behaviour (51.92% to 48.08%). In these cases, there is not enough evidence to reject the null hypothesis.

Although 67.3% of respondents claim to play video games often, they were not asked the genre of games they play. Due to the nature of our study, previous video game knowledge likely influenced subject responses. Other confounding factors in our study may include the non-randomized ordering of the questions, as well as priming the participants by introducing the idea of waiting within the survey, although this was necessary to avoid confusion. More data in more contexts would help clarify these issues. Our overall conclusion is that players generally expect shorter and simpler ("ideal") paths, and otherwise adapt their expectations to reflect what they have previously observed.

Conclusion and Future Work

Games are frequently attempted more than once, and understanding the learning curve is an important direction in improving design. Our work defines an approach to modelling observation-based knowledge of NPC movements that offers ample flexibility to represent different kinds of player expectation, and can be used within a tool for exploring how level design and prior knowledge may interact. A user study confirms players have constrained expectation of NPC movements, also affected by prior experience.

A number of interesting future directions are possible from our work. Actual game integration with player tracking data would of course be useful, and would let us validate that our computed expectations correlate with player behaviours. We are also interested in incorporating ambiguity into the model, reflecting imprecision and lossiness in player observation. Introducing negative path segment knowledge would also be interesting, as knowing where a NPC is not at different time frames also contributes to accurately modelling what a player knows, what they will predict, and thus how they will behave.

Acknowledgments

This work supported in part by NSERC grant 249902, and by the COHESA project, through NSERC Strategic Networks grant number NETGP485577-15.

References

Aparicio, IV, M., and Caban, D. R. 2009. Distance-based spatial representation and prediction systems, methods and computer program products for associative memories. US Patent 7,574,416.

Bakkes, S. C.; Spronck, P. H.; and van Lankveld, G. 2012. Player behavioural modelling for video games. *Entertainment Computing* 3(3):71–79.

Duque, D.; Santos, H.; and Cortez, P. 2007. Prediction of abnormal behaviors for intelligent video surveillance systems. In *2007 IEEE Symposium on Computational Intelligence and Data Mining*.

Geib, C.; Weerasinghe, J.; Matskevich, S.; Kantharaju, P.; Craenen, B.; and Petrick, R. 2016. Building helpful virtual agents using plan recognition and planning. In *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Heider, F., and Simmel, M. 1944. An experimental study of apparent behavior. *The American Journal of Psychology* 57(2):243–259.

Hladky, S., and Bulitko, V. 2008. An evaluation of models for predicting opponent positions in first-person shooter video games. In *IEEE Symposium On Computational Intelligence and Game*, 39–46.

Isla, D. A., and Blumberg, B. M. 2002. Object persistence for synthetic creatures. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 3*, AAMAS '02, 1356–1363. ACM.

Isla, D. 2013. Third Eye Crime: Building a stealth game around occupancy maps. In *Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Patterson, T. A.; Thomas, L.; Wilcox, C.; Ovaskainen, O.; and Matthiopoulos, J. 2008. State–space models of individual animal movement. *Trends in Ecology & Evolution* 23(2):87–94.

Roemmele, M.; Morgens, S.-M.; Gordon, A. S.; and Morency, L.-P. 2016. Recognizing human actions in the motion trajectories of shapes. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*, 271–281.

Singh, K. Y.; Davis, N.; Hsiao, C.-P.; Jacob, M.; Patel, K.; and Magerko, B. 2016. Recognizing actions in motion trajectories using deep neural networks. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Smith, A. M.; Lewis, C.; Hullet, K.; Smith, G.; and Sullivan, A. 2011. An inclusive view of player modeling. In *Proceedings of the 6th International Conference on Foundations of Digital Games*, 301–303.

Tastan, B., and Sukthankar, G. 2011. Leveraging human behavior models to predict paths in indoor environments. *Pervasive and Mobile Computing* 7(3):319–330.

Tastan, B.; Chang, Y.; and Sukthankar, G. 2012. Learning to intercept opponents in first person shooter games. In *IEEE Conference on Computational Intelligence and Games (CIG)*, 100–107.

Tremblay, J.; Torres, P. A.; and Verbrugge, C. 2014. Measuring risk in stealth games. In *Proceedings of the 9th International Conference on the Foundations of Digital Games*.

Weber, B.; Mateas, M.; and Jhala, A. 2011. A particle model for state estimation in real-time strategy games. In *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 103–108.

Yin, Q.; Yue, S.; Zha, Y.; and Jiao, P. 2016. A semi-Markov decision model for recognizing the destination of a maneuvering agent in real time strategy games. *Mathematical Problems in Engineering* 2016. Article ID 1907971.

Zhang, M. X., and Verbrugge, C. 2018. Path prediction study. https://goo.gl/forms/9KbNednVIZvxyHGs2.