Biologically-Inspired Gameplay: Movement Algorithms for Artificially Intelligent (AI) Non-Player Characters (NPC)

Rina R. Wehbe*, Giovanni Riberio, Kin Pon Fung, Lennart E. Nacke, Edward Lank

Cheriton School of Computer Science, HCI Games Group, The Games Institute, University of Waterloo

ABSTRACT

In computer games, designers frequently leverage biologically-inspired movement algorithms such as flocking, particle swarm optimization, and firefly algorithms to give players the perception of intelligent behaviour of groups of enemy non-player characters (NPCs). While extensive effort has been expended designing these algorithms, a comparison between biologically inspired algorithms and naive directional algorithms (travel towards the opponent) has yet to be completed. In this paper, we compare the biological algorithms listed above against a naive control algorithm to assess the effect that these algorithms have on various measures of player experience. The results reveal that the Swarming algorithm, followed closely by Flocking, provide the best gaming experience. However, players noted that the firefly algorithm was most salient. An understanding of the strengths of different behavioural algorithms for NPCs will contribute to the design of algorithms that depict more intelligent crowd behaviour in gaming and computer simulations.

Keywords: Games User Research (GUR), Biological Algorithms, Non-player Characters (NPCs), Artifical Intelligence (AI), Movement in Games

Index Terms: H.5.2 [User Interfaces]: User Interfaces—Graphical user interfaces (GUI); H.5.m [Information Interfaces and Presentation]: Miscellaneous

1 INTRODUCTION

The heart of video games is the user experience. Developers need to make tough decisions to ensure the game is both challenging and entertaining keeping players in the state of optimum flow experience [7]. One challenge presented to developers is the creation of algorithms that control non-player characters (NPCs). Among many other parameters, one of the primary aspects of NPC behaviour involves how these characters move during gameplay. The creation of realistic movements is necessary for individual NPCs whose movement can be inspired by some individual character goal. On the other hand, if the NPC is a member of a group of NPCs, some form of group-based algorithm for movement is necessary to give the appearance of purposeful group behaviour. These group-based movement algorithms have collectively been labelled *flocking algorithms*.

While significant research effort has gone into the design of flocking algorithm variants (blocks, particle swarms, and firefly, for example), it is difficult to determine how important these algorithms are in gameplay. By design, their goal is to create more realistic NPC group behaviours, but do they? And if they do, does that affect player perception of character realism and intelligence? Is engagement and enjoyment in games also affected? Do players even notice the change in NPC behaviour? While it is true that these algorithms have been contrasted with realistic group behaviours to validate their efficacy [35], we are aware of no work that has actually asked the above fundamental questions about the effect of these algorithms on the player experience. This question is particularly interesting given that the ultimate goal of these algorithms is to enhance realism. Therefore, the contribution of our work is to test if the changes between the flocking algorithms affect user experience.

In this paper, we explore player response to flocking algorithms through two experimental studies. First, we invite users to play a game where enemy types are controlled by four different algorithms: Flocking, Particle Swarm Optimisation (PSO), and Firefly Algorithms, and a control condition where NPCs in groups move in a straight line toward their objective. We do this to assess the effect the algorithms have on the player's experience. Our results reveal differences between algorithms. Players identified the particle swarm optimisation (PSO) algorithm as their most preferred algorithm, but it was also ranked as the easiest algorithm. In contrast, flocking was less preferred by users but rated as the most difficult algorithm; they felt that groups of enemies whose movements were controlled by Flock were not as predictable, thereby improving realism. However, the differences were not enough to cause a significant difference in measures of engagement in gameplay. Players seemed little impacted by the differences between the different group behaviour algorithms in-the-moment.

To determine whether algorithms were truly of limited import or if potential confounds in experimental design (e.g. players preferred conditions that were easiest because their success increased, algorithmic parameters were poorly tuned, or the game design was lacking) were resulting in limited impact on measures of immersion in game play, we conducted a follow-on Mechanical Turk study. The follow-on study further validates the limited utility of NPC coordinated movement algorithms within our game platform, and provides additional evidence that aspects of game play such as story and aesthetics seem more important to overall measures of enjoyment in gaming than do flocking algorithms.

While a superficial interpretation of our results might, at first, argue that flocking algorithms have limited impact on game play, it is the case that algorithms created significant differences in perceived difficulty. One challenge with game play is the calibration of difficulty levels to player skill, and flocking algorithms, with their impact perceptions of difficulty, may be useful in engineering incremental difficulty adjustments in games, thus preserving challenge and engagement in the long term.

2 RELATED WORK

Maintaining presence and immersion in computer games is a core component of creating a positive Player Experience (PX) [35]. Previous research has demonstrated the importance of the behaviour of artificial intelligence (AI) on maintaining immersion in games [26, 35]. For Game AI, this has meant developing the (often imperfect) behaviour of non-player (usually enemy) characters [35]. Examples of these behaviours include character movement behaviours such as navigation or pathfinding.

The creation of game environments has been separated into two distinct design approaches in past literature [12, 24]. The first is *scripting*, which requires game developers to pre-specify every path

*e-mail: rrwehbe@uwaterloo.ca

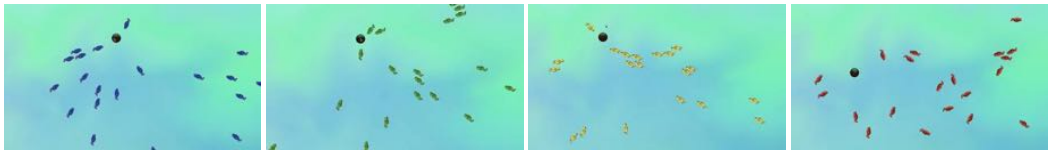

Figure 1: The above images show the original version of the game with each algorithm represented by a coloured fish. In S1 users played the game. These algorithms were recorded for gameplay videos in S2.

of movement, interaction, and game objects that a player will experience throughout the game. The content for this gameplay must be created and include NPC behaviour, scripts, location in area, among many other factors. In pre-created, scripted interactions the linearity of progression causes a direct path through gameplay limiting the ability to create open worlds, flexible game play, and rich combinations of human and NPC characters. The negative factors of this static content approach extend beyond game play: they actually begin at preconception of the game narrative and through the development process. Overall, because scripting must fully specify all in-game behaviours of NPCs, the process creates a high resource cost in both programmer hours, and development time.

In contrast, another approach developers can take towards game design is called *emergence* [12]. Emergence involves developers defining general, global rules for interactions between game objects, and then allowing the gameplay to emerge as the player proceeds through the game. As a simple example, consider Conway's *Game of Life* [5] which illustrates the advantage of created emergence in games. An algorithm can be a simple rule set which can mimic complexity of a life form, and thereby leading to more emerging gameplay. Advantages of emergent design can be balanced with static content generation. Commonly, the spawning of enemy characters in an area can be randomly generated with enemies moving in accordance to predefined algorithms. For example, the Left 4 Dead series [28, 29] combines NPC movement and horde behaviour with narrative elements, and atmospheric game environments.

2.1 Biologically-Inspired NPC Movement Algorithms

Biologically-inspired algorithms mimic behaviours found in the natural world to solve optimisation problems. The Flocking algorithm was developed by Craig W. Reynolds in 1987 [17] and simulates the movement of a flock of birds or school of fish. At every iteration, an agent must decide which direction to move in by taking three factors into account: alignment, cohesion, and separation. Alignment can be defined as an agent wanting to move in the same general direction as it's neighbouring agents are moving. Cohesion means that an agent wants to move towards the average position of it's neighbours. However, separation means the agent does not want to be too close so as to avoid collisions and hence, the agent will want move away from the average position of any neighbours to which it deems itself too near. Reynolds' later 1999 paper [18] added steering factors to his agents, allowing for them to seek roosts (i.e., function optima) in the search space.

In 1995, Kennedy and Eberhart developed the Particle Swarm Optimisation (PSO) algorithm [13], which took some inspiration from Reynolds' work and also from the way in which insects swarm. The PSO algorithm determines an agent's direction of movement by considering two factors. The first is the best position the agent has found at some point in the past, and the second is the global best position that has been thus far determined by all the agents as a unit.

The Firefly algorithm, developed in 2009 by Xin-She Yang, [34] was built upon the PSO algorithm and partially inspired by the way in which fireflies flash their lights to communicate. With this algorithm, each agent is assigned a 'brightness' value which directly corresponds to how good that agent's current solution is. Agents are then attracted to other agents who are brighter than they are and will move towards them. However, this attractiveness of one agent to another is proportional. Two agents that are further apart will appear less bright to each other and hence will find each other less attractive.

Games have taken advantage of the research on biological algorithms to simulate enemy movements. Flocking, PSO, and Firefly were all algorithms developed to simulate natural movement phenomena. In contrast to the mechanical approach for scripted movements, algorithms allow for emergence, or dynamic behaviour [15]. The use of emergence is strategically employed by game developers to create more unpredictable, immersive, and believable enemy characters and worlds.

2.2 Forced Choice Experiments

The goal of this paper is to assess the effectiveness of a set of biologically inspired algorithms used to define movement for collaborating groups of characters in video games. Overall, we wish to determine whether or not there exists a difference between these algorithms, and, in particular, if that difference has a meaningful impact on game play. To do this, we conduct two studies: A first study explores gameplay and finds limited effects on immersion. A second study seeks to determine whether differences are observable in real-world gameplay. In psychology, the experimental protocol that determines whether discrimination between phenomena exists, even at a subtle level, is a just-noticable difference (JND) study or signal detection study [16, 25]. We leverage these concepts in a forced choice protocol for our second study.

A noticeable difference is the minimum change in a stimulus that can be detected 50 percent of the time [36]. This protocol measures the confusion of the subject as they try to discern small stimulus differences [23, 27]. In the past, this protocol has been used in the HCI literature for a multitude of reasons from graphical fidelity [3, 11] to lag control [32, 33]. JNDs in games user research (GUR) is found in studies of game performance. JNDs were used to select appropriate tempos for a thesis on the effect of music tempo on game performance [14], and used as performance indicators in discrimination tasks which showed video-game players to possess benefits in multisensory processing [9]. JNDs have also been used in the development of serious games which aim to motivate the calibration of HCI tools [10]. More recently it has been proposed that JNDs can be used to help create meaningful variations of game constant by creating cognitively-grounded procedural content [2].

JNDs leverage force choice for signal detection [16,25], a concept we leverage to assess algorithmically guided enemy behaviour have, to the best of our knowledge, not been explored to date.

3 Evaluating In-Game Algorithms

One challenge with assessing end-user perspective on AI in games is that each game includes slightly different AI behaviours and each game also has different story lines, graphics, and gameplay characteristics, all of which significantly influence realism in the game. To control for individual game characteristics, this section first introduces a bespoke game, a simple game of 'tag', which is then leveraged during both phases of the study. Next we describe the study protocol for our first study.

Our study is structured as a 4-factor within-participants mixed-methods study. We explore four algorithms as our independent variables (Control, Flocking, PSO, Firefly). We measure the dependent effects on user-experience, immersion, feelings of realism through self-report questionnaires, and interviews.

3.1 Game

The bespoke game created for this study is called Tag-o-rithms, so named due to the testing of different algorithms in a tag-like game. The game consists of four levels, each of which has its own distinct enemy behaviour based on a biological algorithm. The levels contain 20 *boids*, rigid-body objects with applied steering algorithms, which are pursuing the player's character in an attempt to 'tag', i.e. make contact with or collide with, the player's character. The player's character is represented as a black circle.

Every level consists of one minute of gameplay in which the user evades twenty enemy characters by dragging their player character around the screen with the mouse. Mouse tracking leverages the standard cursor acceleration algorithm. Each time the player is hit by an enemy, the hit counter is incremented by one and the enemy that made contact with the player vanishes and then 'respawns', i.e. reappears, at a random on-screen location. If an enemy moves off the side of the screen, it reappears on the opposite side at the same relative height it disappeared. The player character cannot move off screen. Should the user move their mouse off their character at any point, the character will remain stationary until the user reacquires their character by moving the mouse cursor to their character and begins dragging again. Figure 1 shows four different levels of gameplay, each with a different flocking algorithm.

Enemies in each level all move at the same speed. The vector representing the velocity of an enemy can be defined by the equation

$$v' = \alpha \frac{(1-\beta)v + \beta dir}{\|(1-\beta)v + \beta dir\|} \quad (1)$$

where the new velocity, v' is a normalized combination of the old velocity and the new direction of movement, *dir*, which is determined by the algorithm governing each respective enemy type. β is a constant that determines the weight of the contribution of the new direction and α a constant that is multiplied by the normalized direction to give speed. The github repository of the game is available at *github.com/rinarene*

To maintain consistent scale across platforms, the game is defined based upon its width and height. The boids were 0.066width x 0.059height game units (gu) large relative to the overall width and height of the display. Move speed for boids was 0.1width gu/sec. The applied unity-Rigidbody 2D had 0gu/sec Linear Drag, 0.05width gu/sec Angular Drag, 0 gu/sec Gravity Scale.

3.2 Participants

Participants ($n = 21, 5$ female) were recruited from our university and our local community. Participants were required to be over the age of 18: Ages ranged from 18-40. The majority ($n = 10$) of participants were between 21-25 years of age, with the second highest majority - 26 to 30 years - consisting of 6 participants. No restrictions were made based on skill, gameplay experience, or level of education. All but one participant, reported playing daily(10) or weekly(10). Three self-described as causal or infrequent gamers, 11 as recreational or regular gamers, and 7 as avid gamers.

3.3 Independent Variables

The algorithm applied to each boid (one enemy character in the group) is the independent variable of the study. For simplicity and to control for bias, variations in condition were denoted by enemy colour. In review: blue is the Control condition, green the Flocking algorithm, yellow the PSO algorithm, and red, the Firefly algorithm. For consistency, in game-design we measure games as proportional relationships (e.g. this object is twice the size of object one). This allows for games to be scaled and displayed on multiple screens of different sizes, resolutions, and configurations. Therefore we present our measurements in game-units.

3.3.1 Control: Blue

In the control condition (blue) at every iteration each enemy determines the direction of the shortest straight path to the player character and moves in that direction. An enemy character in this level is unaware of the other enemies and it's behaviour is in no way influenced by other enemy characters' motion. The control condition is representative of the kinematic movement control algorithms that were once commonly implemented [15].

3.3.2 Flocking: Green

The Flocking algorithm is implemented for the green enemies The direction of movement in this case is controlled by four factors; alignment, cohesion, separation, and target seeking. The first three are weighted equally. Target seeking is weighted slightly more than the other three as we found this was necessary in order to have the enemies sufficiently interested in moving towards the player character's location.

3.3.3 Particle Swarm (POS): Yellow

The yellow enemies are our implementation of the PSO algorithm Each enemy is aware of it's own previous best position and the global best position and uses these factors to determine direction of movement. A position is determined to be best if the distance between the enemy and the player when calculated is shortest. As the player is constantly moving, the distances of all best positions must be recalculated at every increment.

3.3.4 Firefly: Red

Lastly, we have the red enemies. In this condition, enemy movements are dictated by the Firefly algorithm. The 'brightness' of every enemy is determined by the enemies distance to the player; the closer an enemy is, the brighter they are. As previously stated, the attractiveness of one agent to another is proportional to the distance between them. An agent that is further away will appear less bright. We have simplified this into a neighbour radius and enemies are only able to see other enemies that are within their radius. Therefore, enemies will not be attracted to other enemies that are far away, regardless of their brightness. It should be noted, that the radius used is same size as that used earlier in the Flocking algorithm.

3.4 Protocol

Before receiving consent, participants were given a brief introduction to the game and controls. After written consent was obtained, participants played each condition (Control, Flocking, PSO, and Firefly) which were presented in a random order. Participants were asked to fill out two standard GUR questionnaires after each play period.

After all conditions were played, participants were asked to fill out a bespoke exit questionnaire. Immediately after completing the exit questionnaire participants took part in a one-on-one semi-structured interview with the researcher. The researcher asked the nine questions displayed on table 1; the semi-structured nature of the interview, however, allowed the interview to probe for additional details, clarifications, or relevant life-experience data to fully explore perceptions of flocking algorithm behaviour.

3.5 Quantitative Measures: Game Score and Questionnaires

One obvious measure of game play efficacy is the overall score a player obtains in any game. In this game, each time a boid made

Table 1: All questions asked of participants druing their semi-structured interviews conducted after completing exit questionnaires.

Semi-structured Interview Questions
Q1: What did you think of the game?
Q2: Can you tell me about about how you felt about the different levels?
Q3: Was there a difference between the [red, blue, green, yellow] enemies?
Q4: Was any level harder or easier?
Q5: Did the game remind you of any off-the shelf/released/popular games? Or games you have played previously?
Q6: Do you think there was a difference between the enemies movement patterns?
Q7: Can you describe in one word, or supply a name for the different enemy types?
Q8: We built the game to reflect different biological algorithms (straight at you, flocking, swarm, and firefly). If you had to match the movement patterns to an algorithm which would you pair?
Q9: Any other overall feelings about the game?

contact with a player, the player's score was incremented, meaning that the goal was to score as low as possible. Alongside scores, we also wish to capture player experience in the game, and we do this using two standard questionnaires and an exit questionnaire.

To measure a self-report of player experience after each level (flocking algorithm), we use the Self Assessment Manikin (SAM) [1] and the Player Experience of Needs Satisfaction (PENS) [8, 19, 20]. Both the SAM and the PENS are established questionnaires that have been shown to correlate well with aspects of immersion and enjoyment. The SAM allows players to give a rating of Pleasure, Arousal, and Dominance by asking players to identify the emotions they feel by selecting the best fitting picture on a visual Likert scale. The SAM allows for a break down of complex emotions. For example, high pleasure, high arousal, and low dominance may indicate a happy and excited player with a low feeling of self efficacy.

PENS [8, 21, 22] measures overall satisfaction with gameplay along dimensions of immersion, challenge, satisfaction, and usability via a set of 20 questions. For example questions ask: "When playing the game, I feel transported to another time and place." (immersion), "My ability to play the game is well matched with the game's challenges."(difficulty and challenge), or "Learning the game controls was easy." (satisfaction and usability). We focus particularly on the sections that correlate with immersion and presence. Again, we operationalize immersion and presence to refer to the feeling of being involved in the game and of being in the game world respectively. [22].

Participants completed by the SAM and PENS after each level of the game, i.e. after experiencing each individual flocking algorithm.

After participants completed all four levels, the participants were asked to fill out a final exit questionnaire. This custom questionnaire collected comparisons between conditions. Specifically, participants were asked which condition was most preferred versus least preferred and which condition was hardest versus easiest.

4 Results

The results of study 1 included game metrics in the form of enemies hit, questionnaire data, and interview data.

4.1 Scores

The object of the game was to avoid being hit by the NPC enemies in the different algorithm conditions. Figure 2 graphs the average scores by condition.

To analyze the scores (categorical discrete count data) for significant differences within participants, we used a repeated measures Analysis of Variance (ANOVA) in IBM SPSS Statistics v24. Given that the sphericity assumption was violated, we applied a

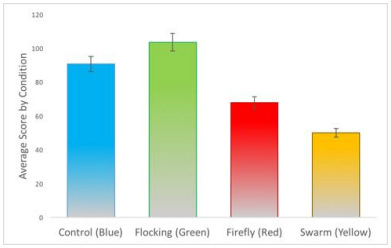

Figure 2: Average in game scores by condition.

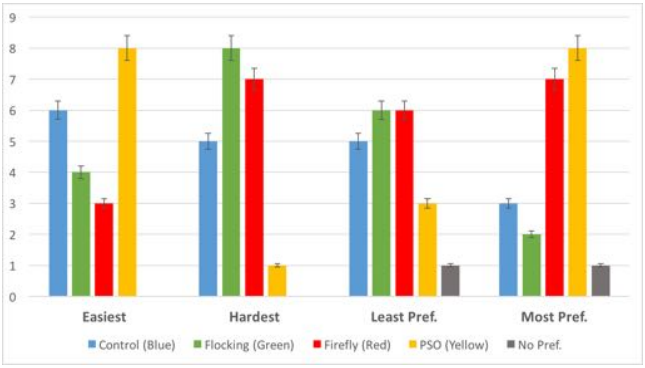

Figure 3: Participant ranking of algorithms on exit questionnaires: easiest versus hardest and least versus most preferred algorithms.

Greenhouse-Geisser correction. Corrected ANOVA was significant ($F(2.347, 12357.008) = 18.260, p < 0.0001$). The differences were further supported by a Poisson regression ($x^2(3) = 27.894, p < 0.0001$). Scores were best (lowest) in swarm followed by firefly and worst (highest) in flocking.

4.2 Questionnaires

Recall that there were two questionnaires tested after each condition, the SAM and PENS, that rated in-the-moment immersion and enjoyment, and an exit questionnaire that asked participants to contrast algorithms. Considering, first, the SAM and PENS data, we tested the data collected using all of a Poisson Regression, a test for the significance of the fit of the discrete data model, and Friedman's ANOVA. In all cases, we find that both the PENS and the SAM scores are not significantly different for each algorithm tested. Given the within-subjects design of our experiment, we find it unlikely that algorithms are resulting in meaningful in-the-moment differences in game play.

On the exit questionnaires, we do find a difference between algorithms in overall ranking. Participants reported liking the PSO the best, followed closely by Firefly enemies. Enemies liked least were almost equally divided (Flocking and Control). When asked about the hardest enemy, participants reported that the Flocking enemies were most difficult, followed by the Firely. PSO was reported to be the easiest to defeat, followed by Control. Figure 3 details this information.

4.3 Interviews

The interviews reveal some insight into the player experience associated with each algorithm tested.

Given our exit questionnaire data, one factor we explored in our interviews was whether the FireFly algorithm (2nd most preferred and 2nd most difficult) might be a good compromise between preference and challenge. Although participants felt that the Firefly

algorithm was salient, the general consensus did not indicate that the effect of the Firefly algorithm on player experience would be positive. Players were very divided in their feedback. P6 describes:

"The red enemies for some reason caught more my attention because they will move themselves their individual movement was more quickly and I feel more tension, I need to focus more on specific groups of fishes where they were moving around where I could not even touch them or be in their way of moving." P6

In contrast, it was also reported that: *"The red ones were very peaceful for me, very relaxing. I did not even notice if they were move individually; I just saw them move around the screen. I didn't pay much attention to their individual movements. The yellow ones make me feel more happy more active, even friendly like if they were my friends and I can touch them and they will not do anything bad to me but then red ones were like dangerous. Don't even get close to me because they were like. I saw them as very, very dangerous . I try to look at the specific figure of the fish and I think I saw they were having like teeth like piranhas. Different kind of fish and enemies then the red and the yellow. I don't know if they are different or not but I saw they looked a little different. And then the blue ones were not dangerous but I feel mad at them. It was anger that I felt when I played against them."*

Participants also felt that the salience of the red enemies made them unnerving. Dialogue conveyed frustration with the novelty they perceived coming from some of the algorithms. P8 was quoted saying: *"...well I'd call the red the 'crazies' or the *** fish."*

The mixed reports were also evident in feedback regarding the necessary skill needed to avoid each enemy. *"It seemed like very much up to chance a lot of the time, just hoping that there wouldn't a ton of fish coming at me from all different directions ... you know, I couldn't handle it. I didn't feel like skill is involved too much, except for the yellow level I felt like I was able to use skill but for red and green I felt like I was just overwhelmed had no choice but to hit the fish all the time."* P4

4.4 Synthesis of Results

In synthesizing our results from this section, we find that PSO was considered the most-preferred condition for game play, but was also ranked as the easiest. In contrast, Flocking was the least preferred (even below control), but was also considered to be the hardest condition.

Recall that the goal of our experiment was to analyse which algorithms increased user engagement in games. Interestingly, the more realistic an algorithm is – particularly in a tag-style game of the kind we created – the more challenging that algorithm should be. The challenge with interpreting data from our initial game-play study is, therefore, as follows: *Are participants ranking PSO highly because it is the most engaging algorithm, or because it is the easiest algorithm to defeat?*

More generally, conventional wisdom in game design would indicate that artificial intelligence, and, in particular, NPC character behaviour, is very important for gamer engagement and enjoyment [30]. This, then, begs the question of whether algorithm effects or confounds (e.g. player dominance, algorithm tuning, game design) might be impacting questionnaire data such that the SAM and PENS data are unrevealing. To explore this question in detail, we present a follow-on study that eliminates game play and instead focuses on perceptions of NPC group behaviours.

5 Study II: Understanding User Preferences

To further understand the results of S1, we conduct a 3-part study. First, we seek to understand if there is a noticeable difference between the algorithms using a forced choice method. Forced choice is a common psychological testing protocol, it is where absent a real preference all groups would be equally likely. If there is a noticeable difference, force choice detects subtle deviations between categories [16, 25]. Using this protocol, we assess if optimization of two of the preferred algorithms (Flocking and PSO) creates a better user experience. Furthermore, since games are rich complex environments, we wish to test ecological validity by also offering game examples from two real world games (an earlier version and the most recent version of a specific game series) and contrasting these real world games with our simple bespoke game. The study is performed on a crowdsourcing platform, Mechanical Turk (M.Turk), for convenience.

5.1 Participants

Participants were recruited from M.Turk. The participants identified as being from the USA. On average participants were 34 with a range of 22-57. Because our primary goal was to cross-reference results with our first study, we recruited 21 participants, a sufficient number to determine whether and if confounds exist in our initial study data. Participant were remunerated $\$4/hour$ for their participation. Overall, 11 participants specified they played often , 5 moderately ($> 1hr/week$), 2 sometimes ($> 1hr/month$), 1 seldom ($> 1hr/6months$), and 2 Never (almost never, or don't play).

5.2 Protocol

As noted above, in this study we explore three aspects of algorithm design. First, to further validate our first study with additional participants, we repeat the algorithms and design from our first study on the mechanical turk platform but using video rather than game play (to eliminate easiness of condition as a factor that increases preference). Next, we examine in more details algorithmic parameters of two algorithms from the first study, the most preferred (PSO/Swarm) and the most challenging (Flocking). Finally, to test whether game attributes might colour judgments, we ask participants to assess AI algorithms as they exist in two popular computer games.

5.2.1 Study Protocol Overview

We test noticeable difference using the a forced choice protocol; we present two algorithmic options to participants and then forces participants to choose one algorithm.

We deployed our study through Amazon's Mechanical Turk [1]. Participants were first given the information letter and consent form. With participant consent, the game was presented as a series of minute long gameplay videos with a micro-questionnaire between rounds.

The MTurk task had an embedded Google form with link provided. After asking basic demographic questions (age, gender, gameplay experience), the form allowed for users to view videos of two of the tested algorithms and choose the most convincing algorithm by asking participants to choose "*the videos that have AI which seems more realistic, intelligent, or have intention.*" We begin with pairwise comparisons of each algorithm and then do one final overall rating question in each section. The sections were:

Algorithm Comparisons Comparison of the algorithms of s1 (Control, Flocking, PSO, Firefly)

Flocking and PSO Variations here we contrast parameters associated with Flocking and Swarm

Game Examples Finally, we contrast two Off-the-Shelf Game Examples (L4D series and Grand Theif Auto Series) with two different levels of AI sophistication (from an early release and a recent one) for each game are contrasted.

For example, as per a JND experimental design, every permutation of pairs of videos was presented to the user to choose between (e.g. Flocking vs. Control, Control vs. PSO, etc.). At the end of

[1] https://www.mturk.com/

Table 2: Specifications for each video and the adjustments made for comparisons in Game Units.

Video	Type	Neighbour	Swarm Speed	Separation
1	PSO	3	6	1
2	Flocking	4	6	2
3	PSO	3	6	1
4	Flocking	4	6	2
5	PSO	3	6	1

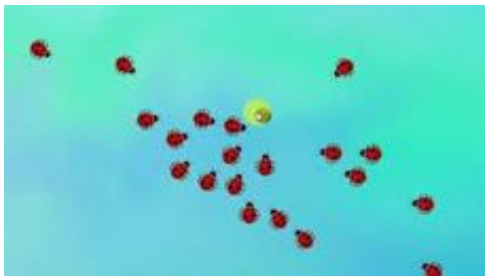

Figure 4: In the optimization of the algorithms all boids were changed to bugs to ensure that any found differentiation by participant were from algorithm movements.

the section, users must rate all options (e.g. Control, Flocking, PSO, and Firely) into 1st, 2nd ... nth placements. This two-part rating process verifies consistency in user ratings.

5.2.2 Flocking and PSO Variations

A second confound associated with realism in algorithms designed to control groups of NPCs during game-play is the specific tuning of individual algorithms. In this section, we focus on two standard algorithms, PSO and Flocking, and we explore how individual parameters of the algorithms might impact their realism, again using a JND design.

Both PSO and Flocking have a set of variables, subject to manipulation, that are common between the two algorithms. While Neighbour Radius, Swarm Speed, and Separation Radius can all be varied, we focus on the density of NPCs within the vicinity of the player by varying the neightbour radius and the separation radius. Table 2 defines the variations in variables we explore in Game Units. By changing the weight factors we hypothesize that the algorithm can be optimized to find a point which results in the best user feedback.

As a control condition, the PSO in Video One weightings were not set (all equal weights of 0) for alignment, cohesion, separation, and target seeking. For all other iterations of flocking and PSO (video 2-5) alignment, cohesion, separation, and target seeking were set to $0.6, 0.6, 0.2, 0.4, 0.5$ game units respectively. In all cases forward movement was set to 0.4 game units.

Identical to the first phase, algorithms were evaluated in pairs, head-to-head, with forced choice followed by an overall ranking to assess consistency in participant responses.

5.2.3 Game Examples

For ecological validity, we last test off-the-shelf published games to understand if a user's perceptions of algorithms may be impacted by the overall aethetics of the in-game experience to such an extent that minutia of character movement becomes unimportant. Since games offer a rich environment, we choose two main series from two different genres. The first series: Left4Dead (L4D) is a post-Apocalypse zombie first-person-shooter (FPS). From this series we choose L4D1 and L4D2. The second series, Grand Theft Auto (GTA), features an approximation to a real-world crime story line. From this series we choose GTA San Andreas (third-person shooter)and GTA5 (FPS). Both these series have one updated and one original AI configuration. While it is difficult to map the AI algorithms from the games onto our specific, research-based AI algorithms, there are clear differences in AI behaviour, and we wanted to determine whether these differences were detectable by our participants.

6 Mechanical Turk Results

6.1 Algorithm Comparison

For the algorithms: Control, Flocking, PSO, and Firefly; we begin with a pairwise assessment. Figure 7 illustrates the voted placements. The votes reveal noticeable differences between algorithms. In first place: blue or the control condition; Yellow (PSO) in second; followed by Green, Flocking; and last Red - Firefly.

To test if the rankings are significantly different, we used a Friedman's test. We found significant differences across values $X^2(3) = 31.686, p < 0.01$. All algorithms differed significantly from one another in ranking (there were no ties). Posthoc tests (Wilcoxon Signed Ranks Test) reveal the following for Flocking and Control $z = -3.891, p < 0.001$ Negative signed rank (-) for PSO-Control a non-significant relationship of $z = -1.628, p > 0.103$ (-), Firefly-Control $z = -3.757, p < 0.001$ (-), for PSO-Flocking $z = -2.611, p = 0.09$ (+), Firefly-Flocking $z = -1.954, p = 0.051$ (-), and finally Firefly-PSO $z = -3.170, 0.002$ (-).

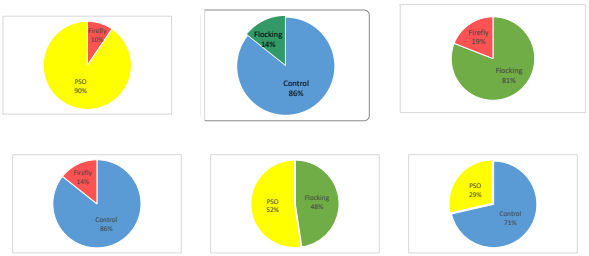

Figure 5: One-on-One comparison of the different Algorithms

6.2 Flocking and PSO Variations

We compare the variations of the PSO algorithm and the Flocking algorithm. Testing Flocking against PSO reveals that the relationship between the two algorithms established in SII original algorithm comparison stands: Flocking is second to PSO according to user rating. Figure 7 (PSO in yellow and Flocking in Green) illustrates the relationship with the PSO in yellow predominant in 1st and 2nd places and the green Flocking algorithms dominating 4th and 5th place. Using the same protocol as above, a Friedman's test was used to look for significant differences between ratings. One participant was excluded due to not following instructions. Differences were significant for the different optimizations: $n = 20, x^2(4) = 19.080, p = 0.001$. Regardless of the specific parameter values we use, PSO consistently performs better than flocking. The post-hoc Wilcoxon Signed Rank Test reveals significance between videos 1 and 2 $z = -19.79, p = 0.048$ (-) based on negative rankings, videos 1 and 4 $z = -2.462, p = 0.014$ (-), videos 2 and 3 $z = -2.774, p = 0.006$(+), videos 2 and 5 $z = -2.487, p = 0.013$(+), videos 3 and 4 $z = -3.014, p = 0.003$ (-), and finally, videos 4 and 5 $z = -2.486, p = 0.013$ (+).

6.3 Game Examples

To test that the game itself was not confounding the results of the study, we test off-the-shelf game examples. From the chosen examples, we see that the L4D series AI is preferred over the GTA AI. This finding is illustrated by Figure 7. Comparing the game series

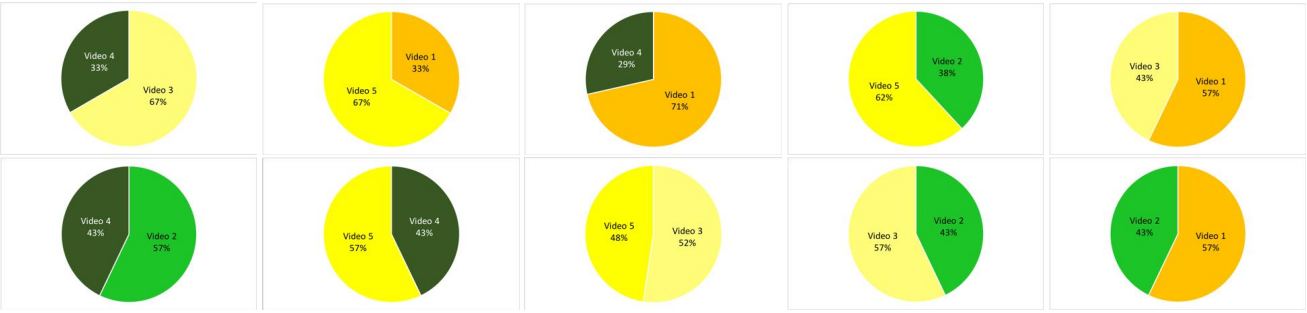

Figure 6: One-on-one comparison of Flocking and PSO variations.

with each other, we can see that L4D1 and L4D2 are equally rated by players; similarly, GTASA and GTA5 have similar ratings despite the differences in AI behaviour. Using the same protocol to test with a Friedman's test, significant differences were found between ranking of game examples: $n = 21, X^2(3) = 16.486, p = 0.001$. The post-hoc Wilcoxon Signed Ranks Test demonstrates not noticeable differences between series franchises, and reveals significant differences between game series with GTASA-L4D $z = 0.344, p = 0.001$ (-), GTASA-L4D2 $z = -2.833, p = 0.005$ (-), GTA5-L4D $z = -2.686, p = 0.007$(-) GTA5-L4D2 $z = -2.554, p = 0.11$ (-).

7 Discussion

Many interactive systems are designed such that the user interacts with or is affected by AI systems. In games, these systems frequently underlie the behaviour of NPCs, and, in particular, enemy NPCs. Research effort into these algorithms is premised on the assumption that improvements in AI algorithms that control NPC behaviour will improve the in-game experience.

7.1 TAKE-AWAY 1: Difficulty is Not a Direct Predictor

In our initial study, our participants reported liking the NPC opponents controlled by the particle swarm optimization algorithm (i.e. the yellow enemies) the most, followed by NPC opponents controlled by the firefly algorithm (red enemies). The PSO enemies were also considered the easiest to defeat. The PSO enemies may have been the easiest because the PSO algorithm directs movement according to best global position, not according to player trajectory or position. This may give players more room when fleeing these enemies. Despite the correlation between preference and low difficulty indicated by game scores and ease ratings, interview data in our first study focuses on the salience of movement. Our results indicate that further study may show that difficulty alone does not fully account for ratings of enemy intelligence. Following up with Study 2 provides perspective on the aforementioned results of study 1. Players do notice differences (even modest ones). Despite the fact that our studies demonstrate that these differences are not easily articulated. Unlike productivity applications, games purposefully need to create challenge that is calibrated to player skill [31]. Calibrated challenge in games keep players in a state of flow or optimized experience. The concept of optimized experience from psychology [6] describes a state where the task is its own reward, with a reduced sensation of time. Over time during game play, if enemies become too predictable, the game will not properly calibrate challenge and the experience will become less optimized, too easy. We believe that our studies – particularly the difficulty rankings of algorithms noted in study 1 – demonstrate that NPC flocking algorithms may be an effective tool for preserving this balanced difficulty and therefore, a sense of flow.

7.2 TAKE-AWAY 2: Direct Movement Can Be the Best Initial Choice

Study 2 results articulate clearly that participants do see discernible differences between algorithms of movement. However between the optimized algorithms, when asked to identify which algorithm seemed to be most intelligent or purposeful in movement participants consistently rated the control condition as most realistic, intelligent, or intentional. In the control condition the boid-enemies move directly towards the player. Any addition of a biological algorithm movement pattern gave the perception of decreased intelligence. Participants in our second study also did not significantly differentiate between L4D1 and the updated AI of L4D2 [4]. Games are rich environments (with sound, art, textures, etc.). The systematic higher ratings for the L4D series supports the idea that AI algorithms matter less to players' perceptions of intelligence. Furthermore, in S1 the lack of notice or comprehension seen in the qualitative data indicates that game players may not be able to fully comprehend the distinction between different AI behaviours in-the-moment. In the absence of distinction, an early instantiate of group behaviour can be direct movement toward an objective. This seems purposeful to a player, is immediately understandable, and can therefore represent early agency by NPCs. Undoubtedly, over time, this algorithm would become too basic, but initial game encounters can and should be immediately obvious and sensible. Anticipation, ambushes, and other subtle, indirect forms of NPC coordination can wait until players develop increased familiarity with the subtleties of interaction.

8 Conclusion

As we note in the introduction, artificial intelligence is increasingly applied in computer games to simulate agency and intelligence in artificial characters, i.e. non-player characters or NPCs. This paper specifically explores one aspect of NPC behaviour, flocking algorithms, that control the coordination among members of a group of NPCs. We present, to the best of our knowledge, a first study examining how the contrasting behaviours of three different flocking algorithms affect player perception of realism, player preference, and player evaluation of difficulty. Overall, our initial results argue that one cannot simply assume that any individual algorithm is better than any other algorithm; instead, these algorithms exist as one tool for game designers as they seek to create realistic end-user experiences during game play.

Acknowledgments

We thank Megan Antoniazzi for her contributions to the idea and early project. We thank NSERC, SSHRC, and the Games Institute at University of Waterloo for supporting this study.

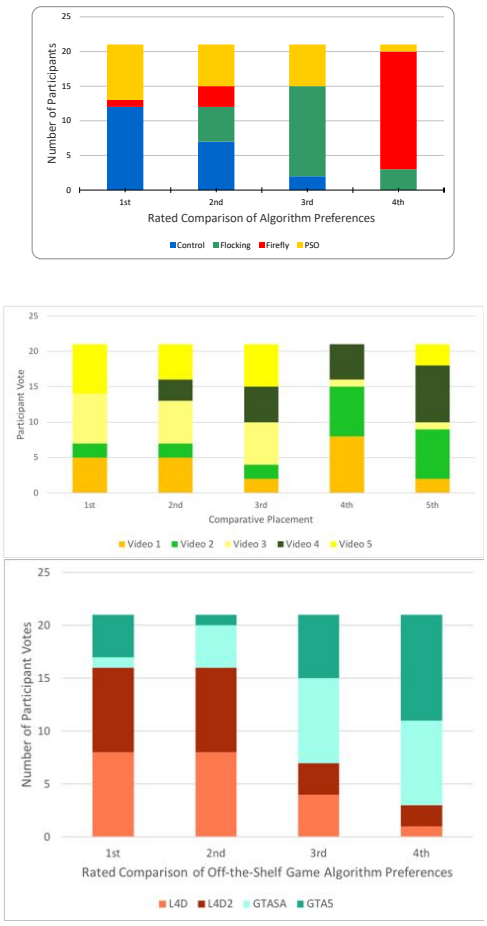

Figure 7: Comparison of the different (a) Algorithms, (b)Weighted Optimizations of the Swarm and Flocking Algorithms, and (c) off-shelf Game Algorithms

REFERENCES

[1] M. M. Bradley and P. J. Lang. Measuring emotion: the self-assessment manikin and the semantic differential. *Journal of behavior therapy and experimental psychiatry*, 25(1):49–59, 1994.

[2] R. E. Cardona-Rivera. Cognitively-grounded procedural content generation. In *Games@ AAAI*, 2017.

[3] I. Cheng and W. Bischof. A perceptual approach to texture scaling based on human computer interaction, 2006.

[4] V. D. Community. Artificially intelligent(ai) systems of l4d, 2011.

[5] J. Conway. The game of life. *Scientific American*, 223(4):4, 1970.

[6] M. Csikszentmihalyi and I. Csikszentmihalyi. *Beyond boredom and anxiety*, vol. 721. Jossey-Bass San Francisco, 1975.

[7] M. Csikszentmihalyi and I. S. Csikszentmihalyi. *Optimal experience: Psychological studies of flow in consciousness*. Cambridge university press, 1992.

[8] E. L. Deci and R. M. Ryan. Intrinsic motivation and self-determination in human behavior. 1985. *Consultado en septiembre*, 2013.

[9] S. E. Donohue, M. G. Woldorff, and S. R. Mitroff. Video game players show more precise multisensory temporal processing abilities. *Attention, perception, & psychophysics*, 72(4):1120–1129, 2010.

[10] D. R. Flatla, C. Gutwin, L. E. Nacke, S. Bateman, and R. L. Mandryk. Calibration games: Making calibration tasks enjoyable by adding motivating game elements. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pp. 403–412. ACM, New York, NY, USA, 2011. doi: 10.1145/2047196.2047248

[11] J. R. Flynn, S. Ward, J. Abich, and D. Poole. Image quality assessment using the ssim and the just noticeable difference paradigm. In *International Conference on Engineering Psychology and Cognitive Ergonomics*, pp. 23–30. Springer, 2013.

[12] J. Juul. *Half-real: Video games between real rules and fictional worlds*. MIT press, 2011.

[13] J. Kennedy and E. R. Particle swarm optimization. In *Proc. IEEE International Conf. on Neural Networks*, vol. 4, pp. 1942–1948, 1995.

[14] D. Lawrence. The effect of musical tempo on video game performance, 2012.

[15] I. Millington and J. Funge. *Artificial intelligence for games*. CRC Press, 2016.

[16] J. A. Nevin. Signal detection theory and operant behavior: A review of david m. green and john a. swets' signal detection theory and psychophysics. 1. *Journal of the Experimental Analysis of Behavior*, 12(3):475–480, 1969.

[17] C. W. Reynolds. Flocks, herds, and schools: A distributed behavioral model, in computer graphics. *ACM SIGGRAPH Computer Graphics*, 21(4):25–34, 1987.

[18] C. W. Reynolds. Steering behaviors for autonomous characters. In *Game Developers Conference*, vol. 1999, pp. 763–782, 1999.

[19] R. M. Ryan and E. L. Deci. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary educational psychology*, 25(1):54–67, 2000.

[20] R. M. Ryan and E. L. Deci. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American psychologist*, 55(1):68, 2000.

[21] R. M. Ryan, C. S. Rigby, and A. Przybylski. The motivational pull of video games: A self-determination theory approach. *Motivation and emotion*, 30(4):344–360, 2006.

[22] R. M. Ryan, C. S. Rigby, and A. Przybylski. The motivational pull of video games: A self-determination theory approach. *Motivation and Emotion*, 2006. doi: 10.1007/s11031-006-9051-8

[23] S. S. Stevens. *Psychophysics: Introduction to its perceptual, neural and social prospects*. Routledge, 2017.

[24] P. Sweetser and J. Wiles. Scripting versus emergence : issues for game developers and players in game environment design. *International Journal of Intelligent Games and Simulations*, 4(1):1–9, 2005.

[25] J. A. Swets. *Signal detection theory and ROC analysis in psychology and diagnostics: Collected papers*. Psychology Press, 2014.

[26] R. Tamborini, M. Grizzard, N. David Bowman, L. Reinecke, R. J. Lewis, and A. Eden. Media enjoyment as need satisfaction: The contribution of hedonic and nonhedonic needs. *Journal of Communication*, 61(1):10251042, 2011.

[27] R. Teghtsoonian. Psychophysics: Introduction to its perceptual, neural, and social prospects, 1975.

[28] Valve Corporation. *Left4Dead*. Game [Windows, Xbox 360, OS X], 2008. Valve Corporation, Bellevue, Washington, USA.

[29] Valve Corporation. *Left4Dead2*. Game [Windows, Xbox 360, OS X], Linux, 2009. Valve Corporation, Bellevue, Washington, USA.

[30] R. R. Wehbe, E. Lank, and L. E. Nacke. Left them 4 dead: Perception of humans versus non-player character teammates in cooperative gameplay. In *Proceedings of the 2017 Conference on Designing Interactive Systems*, pp. 403–415. ACM, 2017.

[31] R. R. Wehbe, E. D. Mekler, M. Schaekermann, E. Lank, and L. E. Nacke. Testing incremental difficulty design in platformer games. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pp. 5109–5113. ACM, 2017.

[32] J. Xu and B. W. Wah. Concealing network delays in delay-sensitive online interactive games based on just-noticeable differences. In *2013 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6. IEEE, 2013.

[33] J. Xu and B. W. Wah. Consistent synchronization of action order with least noticeable delays in fast-paced multiplayer online games. *ACM Trans. Multimedia Comput. Commun. Appl.*, 13(1):8:1–8:25, Dec. 2016. doi: 10.1145/3003727

[34] X. S. Yang. Firefly algorithms for multimodal optimization. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Arti-*

ficial Intelligence and Lecture Notes in Bioinformatics), 2009. doi: 10.1007/978-3-642-04944-6

[35] G. Yannakakis. Game ai revisited. In *Proceedings of the ACM Computing Frontiers Conference*, p. 285292, 2012.

[36] R. M. Yaremko, H. Harari, R. C. Harrison, and E. Lynn. *Handbook of research and quantitative methods in psychology: For students and professionals*. Psychology Press, 2013.