Towards the Design of a Human-Like FPS NPC using Pheromone Maps

Amir Yahyavi
School of Computer Science
McGill University
Montreal, Quebec, Canada,
Email: amir.yahyavi@cs.mcgill.ca

Jonathan Tremblay
School of Computer Science
McGill University
Montreal, Quebec, Canada,
Email: jtremblay@cs.mcgill.ca

Clark Verbrugge
School of Computer Science
McGill University
Montreal, Quebec, Canada,
Email: clump@cs.mcgill.ca

Bettina Kemme
School of Computer Science
McGill University
Montreal, Quebec, Canada,
Email: kemme@cs.mcgill.ca

***Abstract*—Non-player characters (NPCs) in video games tend to be easily recognized by human players, reducing the sense of immersion and limiting the complexity of character interactions. In this paper we study various aspects of the NPCs performance and how it differs from human players. We provide categorization and metrics for quantifying some aspects of the NPCs performance and provide an in-depth analysis of the behavior of NPCs. We detail how movements, interactions, use of items, and relying on static decision-making schemes result in markedly different behaviors from humans in the popular FPS *Quake III*. In addition, we propose a framework relying on a special kind of influence map, a *pheromone map*, which can lead to a more adaptive human-like behavior. These maps can efficiently give a summary of the events in the game world, be adaptive in nature, and be effectively used in the decision making process of NPCs.**

I. Introduction

The ability to play with other online players is essential in the design of many games. To ensure a rich and interactive game world, games often rely on the use of non-player characters (NPCs). NPCs can play alongside or against human players or be used to simply populate the game world. They should be able to play in teams and handle different goals and strategies. Ideally, they should also show human-like movements and interactions, giving players the sense of an active and well populated virtual world.

Within the first person shooter (FPS) genre, many rely on variations of the area awareness system (AAS) that provides navigation information for the NPCs [1]. These systems give a simplified 3D map of the world where different *areas* are calculated as simplified volumes so navigation within an area is simply point-to-point movement, and reachability can be easily determined. While navigation information is very useful for tasks such as path optimizations, the navigation itself relies on decision-making algorithms often in the shape of a simple decision tree and fuzzy logic. Combined together they determine the long-term and short-term goals of the AI in the game [2]. For example, in Quake III, the NPC selects, using predetermined preferences, a target item such as a weapon. It also tries to pickup any useful items along the way to his chosen target.

Although this produces functional NPCs, a drawback to this approach is that static decision-making tends to result in overdetermined and unrealistic behavior. In terms of movement, some areas end up being overused by the NPCs while some areas are never explored. With respect to use of weapons and other game items, more important differences become visible, clearly distinguishing NPCs from humans. For example, while reaction speed means that human players will usually avoid use of slower acting distance weapons such as sniper rifles in close or rapid combat, NPCs do not necessarily suffer from the same limitations. These constraints are not a result of limitations of the weapons or the game world in any obvious and ahead-of-time sense, but are due to limitations of the human players themselves, and so cannot always be easily extracted from the game mechanics.

Inspired by these examples and backed by traces collected from several gaming sessions in Quake III, we discuss a number of metrics that can be used to measure the performance of the AI in comparison to a human player (we refer to the characteristics that the AI exhibits as its performance). Furthermore, we provide an in-depth analysis of how these decision-making algorithms can lead to substantial differences in behavior of NPCs and humans. As a potential solution, we discuss how special kinds of influence maps, *pheromone maps,* can be used to adaptively change the behavior of NPCs, making them more human-like. In this model each interesting entity or event is assigned a given attractiveness leading to the generation of pheromones that spread in the game world and fade over time. This attraction is then integrated into the process of decision making of NPCs, giving the AI a useful summary of the dynamic game state. Our main contributions are:

1) We discuss several categories of metrics that can be used to measure NPC performance and how it differs from human players.
2) We provide in-depth analysis of differences between human and typical AI players based on a popular FPS game AI. Detailed *Quake III* game traces are used to compare various aspects of gameplay. Our results give quantitative insight into how NPC and player behaviors differ.
3) We describe a lightweight, adaptive framework that takes temporal and spatial aspects of the game into account in the decision making process. *Pheromone maps* are designed to allow for a dynamic and adaptive AI gameplay.

II. Background and Related Work

As a proxy for real opponents and team-mates, players expect NPCs to be reactive to player actions, the game

environment, and be adaptive in their behavior according to the player's preferences and performance [3]. In fact, in many cases, it is desirable that the NPC shows a behavior very similar to that of a human player.

To better understand why NPCs act the way they do, we consider the following underlying mechanisms: (1) The basic underlying mechanisms, such as decision trees, that are shared by many of the NPCs and are used to implement the AI. (2) The way the game is able to adapt to the capabilities of the player.

Essentially, a game is an state of *conflict* that is typically resolved by the player achieving certain goals [4]. This conflict closely affects the design of any Non Player Character (NPC). NPCs are expected to offer a challenge to make the game more interesting, and this challenge is supposed to be related to the skill of the players. In many games such as *Quake III*, the player chooses himself the gameplay difficulty at the beginning of the game, e.g., *I Can Win, Bring It On, Hurt Me Plenty, Hardcore* and *Nightmare* in Quake III. NPCs have to be reactive to the player's actions and the environment and have to adapt to the player's preferences and performance.

A. Decision Making:

There are several mechanisms that are used to model the decision process of a NPC such as decision trees, hierarchical state machines, behavior trees, fuzzy logic, *etc.*, each with their advantages and disadvantages [2]. For example, decision trees in which every branch node is a condition and leaf node is an action, are simple to represent and use but does not scale well and is hard to modify.

In *Quake III*, as a typical game AI implementation, the agents use multiple decision trees embedded inside a state machine [1]. The state machine represents states an agent can reach in the game such as *Battle Fight* or *Seek Long Term Goal*. At every think frame, the agent goes through the network of states and finds the most appropriate one, best suiting the agent's current situation. In each state, there exist a structure of *if-then-else*, representing the decision making process of that agent in that particular state. Moreover, inside the structure new states can be reached. For example, in the state *Seek Long Term Goal*, using the agent's interests coming from fuzzy variables, the agent will select a goal, such as picking up a weapon. Human-like, opportunistic behavior is encoded by temporarily switching to a *Seek Short Term Goal* state if the agent notices a useful item along the way. Once the short term goal is completed it goes back to the *Seek Long Term Task* state.

Another development is to use *influence maps* to guide AI behavior. Influence maps can be used to determine who is in control of a certain part of a map [5]. Conceptually, influence maps have some similarity with potential fields used in robotics to define motion control. It refers to a desired position b in the world where the robot wants to reach. In this case, the robot at position a is treated as a particle and gets under the influence of an attractive field U from position a to b [6]. This field represents the free-space of the environment. Obstacles are modeled as negative charges and they repulse the particle. Thus, the robot is exposed to forces that can be formulated as $U(q) = U_b(q) - \sum U_{obstacle}(q)$.

This concept has been applied to games as well, with some authors arguing that a potential field approach can be a viable and cheaper option to replace more conventional, A* path finding [7]. In [8], an algorithm is proposed that determines where is the best place for Pac-Man to move, based on attractiveness of potential fields, taking into consideration the moving *ghosts* and dots to eat. In [5], the authors build influence maps into a tree form, where every leaf is a representation of the game state in form of an influence map. From there a genetic algorithm is used to investigate the best strategy. In [9], the authors use flocking behavior common in real-time strategy games (RTS), and employ information from an influence map to increase the safety of units.

Influence maps have rarely been used in FPS games, mostly focusing on RTS games [10]. In addition, most uses of these maps in decision making have been limited, utilizing a simple distance based approach to determine areas under own or enemy's influence, for tasks such as building a new building. We argue that they can efficiently be applied to FPS games as well, and be used in different tasks and levels of decision making by NPCs.

B. Game Adaptivity:

Game adaptivity has received a considerable amount of research attention from the community [11]. In general, to provide adaptivity, the game first monitors the actions of the players, computes an abstract value of their performance/preferences, and makes changes to the game environment according to those values. These parameters can include player's rank, accuracy, and time taken for task completions. The game can adapt to these parameters, for example, by making it harder for an experienced player by improving the AI performance.

The simplest form of adaptivity is given by offering a player various difficulty levels, controlled by the player, to play the game so that he can adapt the game according to his skills [12]. Most games, e.g., Quake III, provide such controls where the capacities of the NPCs depend on the chosen difficulty level. A more dynamic approach, known to players as the *rubber band*, can be described as a negative feedback loop [4]. This loop tends to keep its environment stable by dampening the differences; e.g., by allowing computer-controlled opponents to catch up no matter how far behind they are. An example is the *Mario Kart* racing game series by *Nintendo*, which handicaps players in proportion to their relative position. A better form of adaptivity is found in the *Left 4 Dead* series of games, which uses a dynamic balancing system that models the game intensity as experienced by the player, in form of a metric based on particular actions within the game. When a threshold is reached, the game's pace slows down by removing opponents, increasing the interval of their appearance, and decreasing their accuracy [13].

While gameplay can be improved using such techniques, they highlight the differences between NPCs and humans, for example, by unusually increasing the accuracy. Our interest in this work is to use more natural methods of adapting NPC behavior, e.g., using pheromone maps, to reduce the glaring differences between human players and NPCS.

III. Performance Metrics

Many metrics for measuring the performance of the game NPCs can be developed. One aspect that is often hard to define is how *natural* the NPCs reactions are in response to various game events. This happens as a result of the player imagining how he himself or other human players would react to the same situation and how it is different from what the NPC has shown. Given that most players differ from each other in their playing style, as long as these differences fall inside an acceptable margin of the human players, they are deemed acceptable. Note that here we are focusing on NPCs that take human roles in the game and are expected to behave like one and does not include other kinds of NPCs. In addition, extreme cases of NPCs, e.g. NPCs that are used as a near perfect player with perfect aim, mobility, etc., and are specifically designed for their non-human characteristics are not considered.

To be able to quantify these differences we divide them into the following categories: (1) Mobility, (2) Item Pickups, (3) Fighting Strategy, (4) Use of Items, (5) Difficulty Level, and discuss how they can determine the performance of the game NPC. This categorization is based on the design of the different modules in *Quake III* AI [1] and we believe is representative of many games in the FPS genre.

Note that here we are ignoring teamwork as a part of the NPCs task. Many NPCs are used as teammates for the players and are expected to support the player in the game or perform various other tasks (e.g., defend a certain location). This, however, is quite game specific and given fairly basic support from *Quake III*, we ignore it here and instead focus on death match type games where each player is independent from others and its aim is to simply win the game by getting points through eliminating other players.

Here, we introduce a number of performance metrics in order to quantitatively compare the performance of human and NPC players. Note that the metrics we propose are designed to be useful for comparing different AI strategies and are not necessarily indicative of a *better* approach in general as game developers may choose to differentiate NPC behavior from human players by choice depending on the game and playing scenario. We later, in Section IV, analyze these measurements in *Quake III*.

Movements: One basic but defining task of the players is movements. While providing basic movements e.g., randomly, is easy, it affects all other behaviors of the NPC as it define what enemies it will face, what items it can pickup and in what strategic locations it will end up. One main drawback of use of AAS files and optimized path finding algorithms such as A* is that it makes NPC movements repetitive and fairly predictable. While long term strategic goals of the NPCs determines their final destination, the path they take to get to their destination is fairly predictable. In addition, even though game events, such as becoming under fire and escaping, and short term goals do make the behavior appear more complex it is usually not enough. Moreover, it prevents the NPCs from exploring certain parts of the maps which can be particularly useful depending on different game styles and skills of other players.

Metrics: (1) To quantitatively, measure the differences between human and NPC movements we divide the game world into cells and measure how often they are used by NPCs and humans. Each cell maintains a count for humans and NPCs which is increased every time a player passes through the cell depending on its type. In a binary space partitioning (bsp) based approach the original bsp maps can be used as cells. This can be used as a tool for developers to highlight the different movement patterns. In addition, a difference map that calculates the difference between the two counts on each cell can be used as a quantifiable metric.

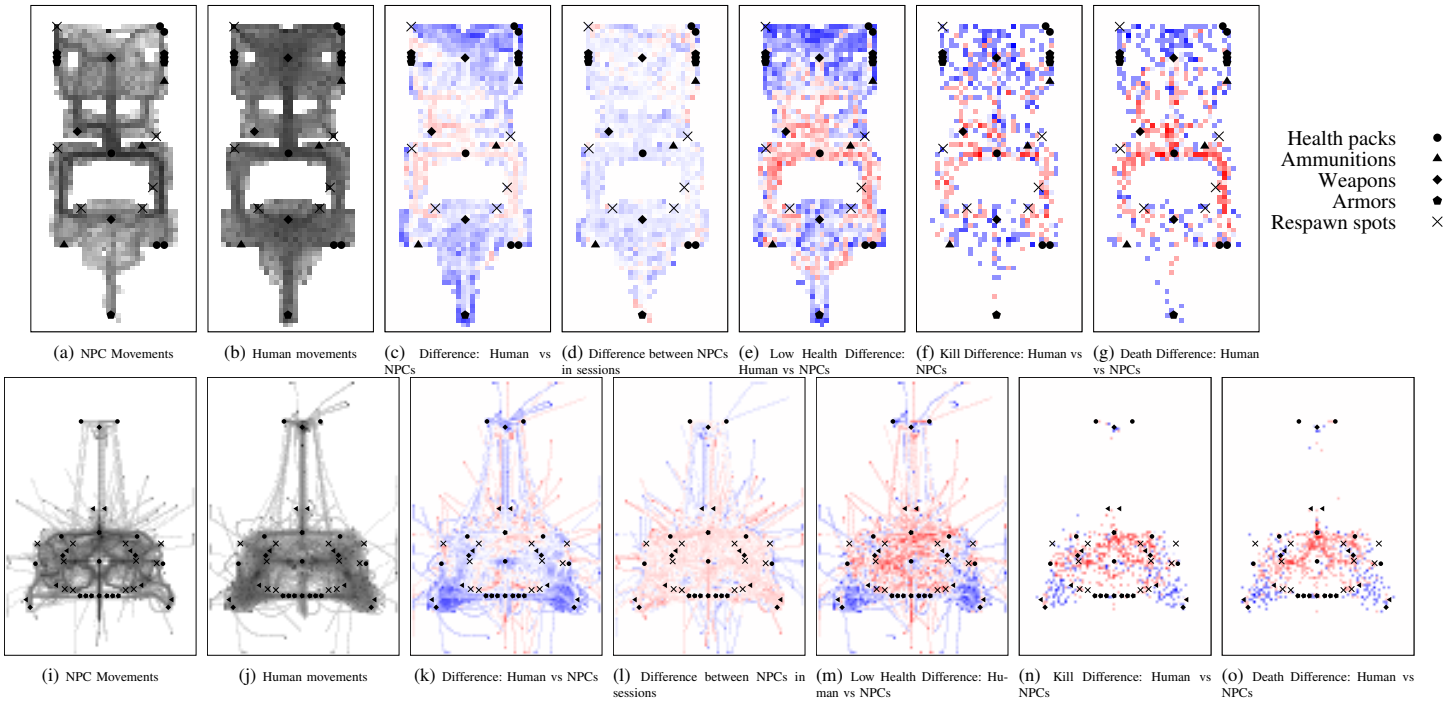
(2) The previous metric provides an overview, of how on *average* players move and does not highlight individual cases. One of the main differences in movements that is jarring to players is during their close interactions. How human players and NPCs move when in close combat can act as another measure for comparison. This can be calculated as a function of whether the players is under attack or not and as a function of the time passed since the interaction.

Item pickups: What game items the players pickup affects their gameplay. These items may include different weapons, ammo, health packs, armor, etc. For example, it affects the path the players choose (e.g., deciding to pick up a certain item). Moreover, a player with good armor and health can afford to be more aggressive, and the type of weapon (e.g., long range, short range, melee) dramatically affects player's behavior.

Metrics: An indicator is how players act when they are in a particular state, e.g., in low health. Players are expected to take more defensive instances when they are in low health and are vulnerable or go to location where health pack, armor, or similar items exist. Therefore, a difference map as discussed before but taking into account a particular state (e.g., $health < threshold$) can act as a measurement of their performance. Similarly, the frequency with which players pick up items and the percentage of picking up different items, can also serve as a measure of performance.

Battles: Players are expected to be able to take advantage of strategic locations. These locations typically provide a good vantage point, access to powerful weapons and items, or a good cover or a hiding place from attacks. In addition, players are expected to avoid hazardous locations that makes them open to attacks and unable to defend. One typical problem with NPCs is that players find a weak spot in the game and exploit repetitive behavior of NPCs (e.g., passing through the same path) to gain easy points by repetitively attacking the NPC in the same spot. Note that these spots change during the game and depending on other players behavior, weapons, and locations. For example, a path that is not necessarily very dangerous can become one if another player takes cover in a good vantage point and uses an sniping weapon. A human player would learn after one or two interactions to avoid such a place until the threat is eliminated but most NPCs lack such a long term strategic decision making process.

Metrics: Strategic decision making of bots is fairly hard to measure without considering the specific situation and since a single best solution rarely exists. However, we can measure how on average NPCs performs. We do this by comparing locations that human and NPC are typically killed or are able to score a point. These places highlight the vantage points good for attacking others as well as vulnerable locations.



(a) NPC Movements (b) Human movements (c) Difference: Human vs NPCs (d) Difference between NPCs in sessions (e) Low Health Difference: Human vs NPCs (f) Kill Difference: Human vs NPCs (g) Death Difference: Human vs NPCs

(i) NPC Movements (j) Human movements (k) Difference: Human vs NPCs (l) Difference between NPCs in sessions (m) Low Health Difference: Human vs NPCs (n) Kill Difference: Human vs NPCs (o) Death Difference: Human vs NPCs

Fig. 1. Experimental sensitivity analysis of AntAI using a 12-player trace in a *deathmatch* game from Quake III in the `q3dm01` (first row) and `q3dm17` maps (second row). Difference maps are normalized logarithmic representation of differences between gaming sessions. Intense colors, blue for players and red for NPCs, shows a considerable difference in behavior, while pale color shows lower differences or no event in that area of the map.

Use of items: How players use certain items, greatly affects how they play. For example, a sniping weapon, given high accuracy at a distance and long reload times, requires finding a good vantage point and hiding from other weapons that are less accurate but faster at a close range. In addition, most human players would find use of sniper gun at a close range very difficult while NPCs would not have that problem and it can lead to a tangible different in their behavior. Another important factor is that most players have certain weapons of choice. This affects their choice of paths (to find this weapon or ammo) and where they are most likely to go. They also show great differences in their accuracy when using their favorite weapon in comparison to others. NPCs however do not have such behaviors, unless it is explicitly programmed, which often is not.

Metrics: To measure the use of game items we consider one of the most important category of items, i.e., weapons. How often players use different weapons (percentage of each weapon's use) as well as how often they are successful at targeting people can serve as an indicator of their performance.

Difficulty level: To provide different levels of challenge to players with different skills, the performance of the NPCs needs to be adaptable. This, however, is mostly accomplished by improving the accuracy of NPCs in targeting players. While this is acceptable to a certain level, e.g. expert human players have very good accuracy, it has its limits and after a certain point this performance becomes too unrealistic as NPCs can have near-perfect aim, given that NPCs can use trajectories based on game physics, even for weapons such as rocket launchers that are often hard to aim as they take a longer time to reach a moving target. Another way to improve this performance can be through improving the strategic thinking of the NPCs and their resulting movements.

Metrics: Difficulty can be measured as the efficiency of players in achieving their goals. In a deathmatch game that goal is to score points by eliminating other players. When NPC difficulty is increased most games resort to improving the accuracy of their weapon use which can be measurable. Furthermore, strategic decision making can be measured as it affects movement and use of strategic locations by using various difference maps as discussed before.

IV. *Quake III* Analysis

In this section we provide a thorough analysis of NPC behavior based on decision-trees and state machines as used in many of today's games. In particular we show that while it yields somewhat complex behaviors, this behavior is considerably different from the behavior of human players, and often less versatile. In addition, the NPC's behavior might seem predictable, erratic, or irrational to a human player.

For the purpose of the analysis, we have selected *Quake III*, a popular FPS game, given its popularity and availability of its source code. While newer games use more complex decision making processes, many rely on the same techniques used in Quake III (e.g., Left 4 Dead 2009 [13]), therefore, we believe the analysis can be representative of the FPS genre of games. We have used two popular maps in the game, and played the game in a total of ten gaming sessions using players of various levels of skill. *P* represents players in figures and tables, *NPC* represents NPC of difficulty level *Hurt me plenty*, and N-NPC is NPC with *Nightmare* level. Around ten gaming sessions are performed where each session lasts for around 20-30 minutes and includes 6-12 players of types: players, NPCs, and both. This is a typical case for most *Quake III* gaming sessions and

TABLE I. DIFFERENCES IN USAGE, DISTANCE, AND ACCURACY OF IMPORTANT WEAPONS BETWEEN PLAYERS VS. NPCS

Map	Weapon	P % Shot	NPC % Shot	P % Kill	NPC % Kill	P Avg. Dist.	NPC Avg. Dist.	P % Acc.	NPC % Acc.	N-NPC % Acc.
q3dm01	MachineGun	60.4%	43.4%	32.2%	20.1%	324.2	149.1	5.0%	3.8%	5.4%
	Shotgun	5.28%	34.5%	5.1%	28.0%	253.9	203.8	9.0%	6.6%	8.6%
	RocketLauncher	9.4%	11.5%	30.8%	37.6%	273.7	185.4	30.4%	27.0%	32.2%
	PlasmaGun	24.7%	10.4%	31.8%	14.1%	256.3	178.4	12.0%	11.1%	12.4%
q3dm17	MachineGun	49.6%	60.7%	15.7%	21.8%	451.9	343.6	4.1%	3.3%	5.0%
	Shotgun	20.8%	19.5%	10.7%	12.7%	226.5	374.6	6.7%	6.0%	8.0%
	RocketLauncher	26.5%	19.0%	60.3%	61.2%	259.7	270.9	29.5%	30.1%	37.6%
	RailGun	2.9%	0.65%	12.6%	4.2%	782.9	488.6	55.9%	60.8%	66.7%

while the study is not exhaustive, it can be fairly representative of the general gaming behavior in *Quake III*.

In our results, we emphasize the differences between NPCs and human players in the categories discussed:

Movements: Figures 1a and 1i depict the movements of NPCs in several gaming sessions. Darker areas represent more activity in that part of the map. Note that the figures use logarithmic presence in the map essentially showing a power law distribution in the use of some regions. As shown, NPCs tend to use specific paths and locations in the map as indicated by the AAS files. This is a result of the decision making process and not a limitation of the accessible areas as the NPCs *can* use other paths and locations but often don't. Figures 1b and 1j show the same map for human players. As one can observe, players use the maps more uniformly. While popular locations still exist, as Figure 1j particularly shows for the larger, more complex `q3dm17` map, they are significantly different from the paths chosen by NPCs. Figures 1c and 1k highlight these differences in player and NPC movements in several gaming sessions. Blue regions are mostly used by players, while red regions are mostly used by NPCs with the intensity showing the amount of difference. White shows the same amount of activity (or lack of it) in the region. The figures clearly indicate how players and NPCs use different regions in the game.

Figures 1d and 1l show the differences in AI behavior in different gaming sessions. Even in different gaming sessions and playing against human players with various skills, NPCs show minimal or no difference in their behavior, lacking adaptability to the players' skills and preferences.

Item pickups: Figures 1e and 1m further highlight the differences in movements between players and NPCs when the entity has low health (30 from 100). Players are much more likely to move towards items such as healthpacks and armors. Particularly, NPCs on average where 10% less likely to pick up an item during the game in comparison to humans.

Battles: Players are able to exploit strategic locations, giving them a good vantage point, access to powerful weapons, or take cover from attacks. This difference, and lack of adaptivity is further used by players as shown in Figures 1f and 1n highlighting differences in locations where players and NPCs score by killing opponents. Players tend to go to locations providing powerful items, e.g., rocket launchers, and with a good vantage point, e.g., top level of the map, to perform attacks targeting those in lower levels. This is also shown in death locations in Figures 1g and 1o. NPCs are usually targeted in lower levels and smaller corridors in the game. In gaming sessions in which both players and NPCs were playing, players were able to take advantage of predictability and lack of adaptivity of NPCs, and attack NPCs in bottlenecks while maintaining strategic locations near powerful weapons and with a good vantage point.

This difference is further shown in the *aggressiveness* of the players. Human players are more likely to follow their target even if the target is shooting back. This behavior increases as a function of the time elapsed since last attack by target. In addition, while players die on average in a game at the same rate, 4.34 vs. 4.50 deaths for a NPC per minute in `q3dm1` map, they achieve 1.15 vs. 0.82 kills per death.

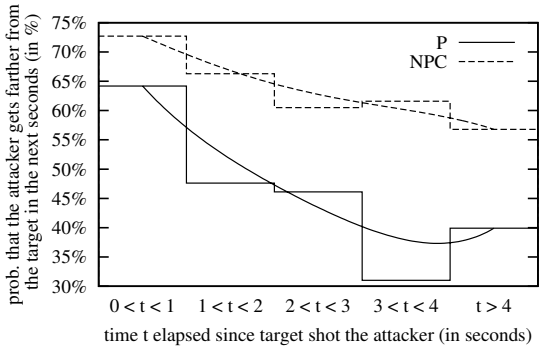


Fig. 2. When engaged in a fight, players tend to get away from each other. However, when attackers are not being shot back, they move towards their target more often.

Use of items: Another significant difference between human players and NPCs is in their use of items. Some items, due to their nature, are hard to use by human players in specific situations. For example, a sniper rifle or its equivalent in Quake III, the Rail Gun, is hard to use in close quarters combats (CQB). Players, as shown in Table I, use this weapon at substantially higher distances than NPCs do. These values also depend on the map. While the use of RocketLauncher in map `q3dm17`, which features large open areas, is very similar to the players, it is different in map `q3dm1`, where the world consists of rooms and corridors and there is higher risk of damaging oneself when using the weapon.

Difficulty level: Lack of adaptability is further shown, when different NPC difficulty levels are used. While, as shown in Table I, the accuracy of attacks are increased along with the difficulty, our difference maps showed no significant difference between movements of NPCs of different levels, similar to the case in Figures 1d and 1l. Similarly, most of the other game

statistics remained the same. This shows that NPCs are not using a better decision making process when the difficulty is increased.

A. Bot Detection

While the main focus of this paper is on improving the performance of the game AI, this analysis can also be employed in detecting when a bot is playing the game instead of a human player. One of common forms of cheating in the games is by having an NPC play the game while the player is away, in order to easily gain experience points or improve one's ranking [14]. A range of techniques have been used to detect these cheat types [15], [16], [17]. Our analysis can be used to extract the necessary features (e.g., movements, interactions, etc) to be used in statistical analysis and machine learning techniques that can classify the players into human or (cheating) NPC players.

V. AntAI Algorithm

Here we propose AntAI, our *vision* of how a more human-like NPCs can be designed using *pheromone maps*. Pheromone maps, inspired by ant colony systems and a form of influence maps, are used to model online events in the game world in a lightweight and efficient manner. We argue that this model can be then used to improve the behavior of the NPC, allowing it to make more strategic and human-like decisions. We have previously used these pheromone maps successfully in improving predictions of human player movements [18], [19]. In such a case the maps are used to model points of interest in the game that attract or repel the player and help in determining which direction the player will move to in the future. For further implementation details please see [19].

In AntAI, points of interest (e.g., weapons, items, other players, events, and specific locations) are treated as ants that generate pheromones, modeling their relative influence. Pheromones are chemicals, whose concentration is coded by a signed floating point number, that exert influences on players. They spread in the game world and fade over time, therefore capturing the geometrical and temporal aspects of interest. We first discuss how these maps are generated, and then, how they can be used to improve the decision making process by the NPCs.

Pheromone Maps As common to most games, AntAI assumes a game world divided into non-overlapping *cells*, e.g., regular grids (as shown in Figure 3), AAS, or open area graphs, typically used for tasks such as path finding, collision detection, or graphical rendering. We denote by C the size of a cell in game world unit. The management of pheromones and the computation of attraction forces exerted by them is performed at the granularity of a cell:

The NPC computes the concentration of pheromone (represented in grayscale in Figure 3) in each cell and computes and sums the corresponding influences. For the sake of scalability, only the cells in a limited region around NPC are considered, e.g., a fixed-size square represented with dashed lines in Figure 3. In a distributed implementation this would be the Area of Interest (AOI) of the NPC.

- **Generation**: In each frame, each point of interest within a cell, be it an avatar or an object, generates a given amount of pheromone related to its attractiveness to the NPC. Events such as kills and deaths can also generate pheromone. Attractiveness is a function of the characteristics of the object and possibly the current state of the considered avatar (a health pack would be more appealing to a wounded player). This amount is added to the concentration of the cell. The maximum concentration of a cell can be capped ($\mathrm{ph}_{\max}$) to limit the attractiveness of any single cell at a given frame.
- **Evaporation**: In order to limit in time the attraction of previous positions of points of interest, pheromones fade in time, meaning that their concentration is decreased at the beginning of each frame. This is necessary to capture the dynamic nature of the game as popular locations and items change over time. Exponential decays, i.e., removing a fixed percentage of the old pheromones at the beginning of each frame, have been successfully used in previous works on ant colonies (e.g., Max-Min ant colonies [20]). Beyond its simplicity and its effectiveness, such an evaporation model ensures that the total amount of pheromones in the game world does not grow to infinity over time. The evaporation speed should reflect the pace of the game.
- **Dissemination**: As pheromones spread, the concentration of pheromone in neighboring cells are mutually dependent. At the beginning of each frame (after the evaporation step), a given amount of pheromone is simultaneously removed from each cell and evenly dispatched to its neighboring cells. The size and shape of this neighborhood depend respectively on the predetermined speed of pheromones dissemination and the game world topology (e.g., wall, hills, *etc.*). For example, obstacles may block the dissemination of pheromones to avoid attraction to unreachable areas.

These phenomena are captured by the following recursive expression of the concentration of pheromone in a cell, for a given player P, at frame t:

$$\mathrm{ph}_t(\text{cell}) = \overbrace{\varepsilon\ \mathrm{ph}_{t-\delta t}(\text{cell})}^{\text{evaporation}} + \overbrace{\sum_{\text{entity}\in\text{cell}} \text{Attr.}(\text{entity}, NPC)}^{\text{generation}} + \underbrace{\sum_{\text{c}\in\mathcal{N}(\text{cell})} \frac{\varepsilon\cdot\gamma}{|\mathcal{N}(\text{c})|}\mathrm{ph}_{t-\delta t}(\text{c})}_{\text{incoming dissemination}} - \underbrace{\varepsilon\cdot\gamma\ \mathrm{ph}_{t-\delta t}(\text{cell})}_{\text{outgoing dissemination}}\ , \quad (1)$$

where ε is the evaporation factor (percentage of pheromones that remain after evaporation), γ is the dissemination factor (percentage of pheromones that spread in the neighboring cells), and $\mathcal{N}(\cdot)$ is the set of a cell's neighboring cells. The attractiveness of a player to itself is set to zero. This is shown in Figure 3 in the area around the trajectory of a moving avatar: some pheromones remain and some spread around its previous positions; all pheromones fade.

Attraction: The amount of pheromone generated for items of interest, denoted by the *attraction* function in equation, is

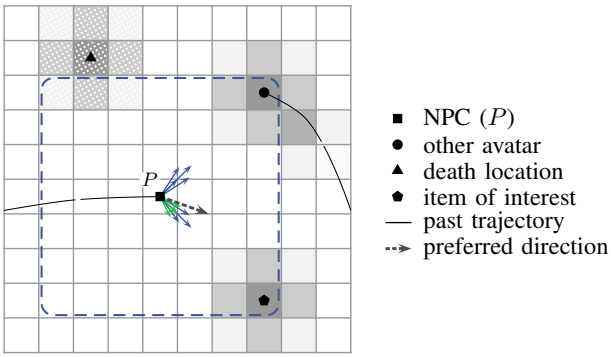


Fig. 3. Overview of AntAI: the game world is divided into cells by using a regular square grid. Each cell contains a certain amount of pheromone, represented grayscale (crosshatched for negative values). Attraction forces are directed towards the attracting cells and their intensity is proportional to the pheromone they contain.

dependent on the NPC's *state* and *goals*. For example, when the health of the player is low, the value generated for a health pack is higher than other items attracting the player to it's location. However, this is substantially reduced after the health pack is picked up. The effect of this increase is quickly faded through evaporation, allowing new interests, such as picking up a powerful weapon for attack, to be taken into account.

Number of maps & Overhead: While a single pheromone map can provide a summary of the game world, multiple maps can also be used taking only specific items or events into account. In the former, a single map exists for each NPC, where the current state of the NPC at each frame is taken into account in deciding how much pheromone is generated for each item or event inside that map. However, in the latter, the state of the player is not taken into account when the pheromones are generated for each map. In this case, to take individual state of NPCs into account, each map receives a coefficient according to each NPCs current status, e.g., the healthpack map receiving a higher coefficient for an injured NPC. Many different maps may be generated for different interests such as death locations, kill locations (using different weapons), successful item pickups (e.g., healthpack), strategic item pickups (e.g., flag in capture the flag), and player movements.

The number of these maps is a design decision: higher number of maps gives more detailed information about the game state at higher overhead costs. One big advantage of using different maps, without taking individual state into account, is that information can be shared between all NPCs and is suitable for server systems. On the other hand, single maps generated per NPC can allow the map to be much smaller, e.g., size of AOI, and be distributed among players. A combination of the two approaches may also be used: some maps are shared and each NPCs has one or more exclusive maps. In general pheromone maps are easily implemented and have relatively low overhead. Furthermore, vectorization of the matrix can be used to allow hardware accelerated technologies to be used.

Here, we explain our *vision* on how to use one or more such maps in the decision making process and how it can improve the performance of the NPCs.

A. Decision Making

Pheromone maps can be used to improve different aspects of decision making by the NPCs. We list most common tasks performed during the decision making process of a FPS NPC and how the pheromone maps can be applied. They are easily integrated in the current decision making process of the NPCs and are not meant to replace them.

Long term goals: Long term goals of NPCs in a FPS game, can be goals such as attacking an area in the game, defending the base, providing support for teammates, or maintaining strategic locations. The pheromone map can provide a summary of locations under attack as well as vulnerable positions (with regards to visibility and game mechanics) to help the NPC choose the target destinations. For example, when the player is in attack state it can help choose goals that are more strategically vulnerable. Note that while many options may exist on the map, their attractiveness is dependent on the amount of pheromone for that location as well as their Euclidean distance from the player as shown in equation 2, where k is a configuration parameter depending on the game mechanics. These forces are then summed up as shown in Figure 3 (when multiple maps exist, the values are first multiplied by map coefficient) and used to decide on a target.

$$\|Attraction(\text{cell}, \text{NPC}_{pos,t})\| = \frac{\text{ph}_t(\text{cell})}{d(\text{cell}, \text{NPC}_{pos,t})^k} \ , \quad (2)$$

Short term goals: An example of short term goals would be picking up an item while pursuing a long term goal. These decisions in particular are related to the current state of the player. For example, a player with low health tries to pick up a health pack on it's way to attack the enemy base. The pheromone map provides necessary information for choosing between items and selecting a suitable path to pick up the healthpack while minimizing the threats.

Battles: Pheromone maps can be used to improve the battle performance of the NPCs. A pheromone map highlighting death and kill locations can help choose better locations for fighting and defense: locations with higher death score are more dangerous while kill locations provide a good vantage point. Furthermore, this is calculated according to whether the NPC is in attack or defense mode. Similarly, this can take the weapon being used into account, highlighting where kills and death occur when players are using a particular weapon. This also helps with making the NPC's usage of weapons follow the normal game pattern among players. Pheromone trails left by the players help with chasing the enemies or supporting teammates. When the player is wounded and in retreat the map highlighting health pickups would direct the player, while avoiding death locations.

Path finding: As shown in the previous section, current NPC players choose predictable paths not taking the current state of the game into account. The pheromone maps highlight the paths being taken by other players, allowing the NPC to choose more human-like paths. In addition, it allows the NPCs to avoid bottlenecks and strategically vulnerable locations, reducing their predictability.

Difficulty level: Pheromone maps enable the NPC to make more strategic decisions in the game. Therefore, it allows the NPC to act smarter when the difficulty level is increased by the player instead of only increasing the accuracy of the NPCs' attacks. It allows easier control of the difficulty level: the amount of information and details, e.g., number and size of maps, used in the process of decision making can determine the difficulty level of the NPCs.

State transitions: So far we have discussed how to better perform goals and tasks decided by the already existing decision making process. In most games offensive and defensive roles of NPCs do not adapt to the current state of the game but are assigned at the beginning of the session. Pheromone maps can help determine whether the NPC should act in an offensive or defensive state, act as a supporting companion of the leader of the team, chase, or retreat given the current state of the game. For example, a large amount of negative pheromone left by attack of the enemies would trigger a state transition of the NPC to a defensive role if he's the closest to that location and the values are above a certain threshold.

VI. Conclusion & Future Work

In this paper we analyze how FPS NPCs' reliance on traditional decision-making approaches can result in showing significant differences from human players. We categorize different types of NPC behavior and how it can differ from human players. Furthermore, we propose measurable metrics for NPC performance and perform such analysis on popular FPS game *Quake III*. In addition, we proposed a framework using pheromone maps, a special kind of influence map, to improve various aspects of NPCs' behavior. Our *preliminary* tests with this system have been promising and we plan to further complete the implementation and show the improvements according to measurements in our analysis.

References

[1] J. M. P. V. Waveren, "The *Quake III* Arena Bot," Master's thesis, University of Technology Delft, Netherlands, 2001.

[2] I. Millington and J. Funge, *Artificial Intelligence for Games*, 2nd ed. Morgan Kaufmann, 2009.

[3] R. Hunicke, "The Case for Dynamic Difficulty Adjustment in Games," in *Proc. of the ACM SIGCHI Int. Conf. on Advances in computer entertainment technology ACE*, 2005, pp. 429–433.

[4] K. Salen and E. Zimmerman, *Rules of Play: Game Design Fundamentals*. The MIT Press, 2003.

[5] C. Miles and S. Louis, "Towards the Co-Evolution of Influence Map Tree Based Strategy Game Players," in *IEEE Sym. on Comput. Intell. and Games (CIG)*, may 2006, pp. 75 –82.

[6] G. Dudek and M. Jenkin, *Computational Principles of Mobile Robotics*, 2nd ed. Cambridge University Press, 2010.

[7] J. Hagelbäck and S. J. Johansson, "Using Multi-Agent Potential Fields in Real-Time Strategy Games," in *Proc. of the Int. Joint Conf. on Autonomous agents and multiagent systems*, 2008, pp. 631–638.

[8] N. Wirth and M. Gallagher, "An Influence Map Model for Playing Ms. Pac-Man," in *IEEE Sym. on Comput. Intell. and Games (CIG)*, 2008, pp. 228 –233.

[9] H. Danielsiek, R. Stuer, A. Thom, N. Beume, B. Naujoks, and M. Preuss, "Intelligent Moving of Groups in RTS Games," in *IEEE Sym. on Comput. Intell. and Games (CIG)*, dec. 2008, pp. 71 –78.

[10] A. J. Champandard, "The Mechanics of Influence Mapping: Representation, Algorithm & Parameters," http://aigamedev.com/open/tutorial/influence-map-mechanics/, 2011.

[11] R. Lopes and R. Bidarra, "Adaptivity Challenges in Games and Simulations: A Survey," *IEEE Trans. Comput. Intellig. and AI in Games*, vol. 3, no. 2, pp. 85–99, 2011.

[12] J. Chen, "Flow in Games (and Everything Else)," *Commun. ACM*, vol. 50, no. 4, pp. 31–34, Apr. 2007.

[13] M. Booth, "The AI systems of Left 4 Dead," in *AI and Interactive Digital Entertainment (AIIDE)*, 2009.

[14] A. Yahyavi and B. Kemme, "Peer-to-peer architectures for massively multiplayer online games: A survey," *ACM Computing Surveys*, 2013, to appear.

[15] P. Golle and N. Ducheneaut, "Preventing Bots from Playing Online Games," *Computers in Entertainment*, vol. 3, no. 3, p. 3, 2005.

[16] T. Schluessler, S. Goglin, and E. Johnson, "Is a bot at the controls?: Detecting input data attacks," in *Proc. of Int. ACM SIGCOMM Workshop on Network & System Support for Games (NETGAMES)*. ACM, 2007, pp. 1–6.

[17] K. Chen, H. K. Pao, and H. C. Chang, "Game bot identification based on manifold learning," in *Proc. of Int. ACM SIGCOMM Workshop on Network & System Support for Games (NETGAMES)*. ACM, 2008.

[18] A. Yahyavi, K. Huguenin, and B. Kemme, "AntReckoning: A Pheromone-Based Dead Reckoning Algorithm for Games," in *Proc. of Int. ACM SIGCOMM Workshop on Network & System Support for Games (NETGAMES)*, 2011.

[19] A. Yahyavi, K. Huguenin, and B. Kemme, "Interest Modeling in Games: The Case of Dead Reckoning," *MMSJ*, 2012.

[20] T. Stützle and H. Hoos, "Max-Min Ant System," *Future Generation Computer Systems*, vol. 16, pp. 889–914, 2000.