DEPARTMENT OF COMPUTER SCIENCE
SERIES OF PUBLICATION C
REPORT C-2008-50

Framework for evaluating believability of non-player characters in games

Tero Hinkkanen, Jaakko Kurhila and Tomi A. Pasanen

UNIVERSITY OF HELSINKI
FINLAND

Framework for evaluating believability of non-player characters in game

Tero Hinkkanen[1], Jaakko Kurhila[2] and Tomi A. Pasanen[1]

[1]Gamics Laboratory
Department of Computer Science
P.O. Box 68, Fin-00014 University of Helsinki, Finland

[2]Department of Computer Science
P.O. Box 68, Fin-00014 University of Helsinki, Finland

Technical report, Series of Publication C, Report C-2008-50
Helsinki, July 2008, 10 pages

Abstract

We present a framework for evaluating believability of characters in first-person shooter (FPS) games and look into the development of non-player character's user-perceived believability. The used framework is composed of two aspects: firstly, character movement and animation, secondly, behavior. Examination of three different FPS games yields that the newer the game was, the better the believability of characters in the game. Moreover, the results from both the aspects of the framework were mutually balanced through all games examined.

Computing Reviews (1998) Categories and Subject Descriptors:

I.2.1 Artificial Intelligence: Applications and Expert Systems — games
J.4 Social and Behavioral Science — psychology, sociology

General Terms:
First-person shooter (FPS), believability, movement, animation, behaviour

Additional Key Words and Phrases:

1 INTRODUCTION

First-person shooter (FPS) games have been popular ever since their first release in the early 1990's (*Hovertank 3D* 1991, *Wolfenstein 1992, and Doom 1993*). The games are usually straightforward in a sense that the target is to navigate the player's character through different levels of the game and accomplish different tasks. Normal task is to move from point A to point B and shoot everything that moves or tries to shoot back. The view to the game world consists of a split screen where the narrow lower part of the screen is showing player's health and ammo, and the upper large part of screen represents player's eye view to the game world (the large part is the player's only mean to monitor other characters in the game and draw conclusions about them). Depending on the game, in-game characters run by the player or by the computer can have human-like constraints or not. Because games are made for players' enjoyment, not every part of real-world laws and rules are included in the games. For example, a player must be able to win even the most superior enemies in the game alone [8].

Main reason for the popularity of FPS games has been their relatively high-level graphics together with a breathtaking pace of interaction. Nowadays, however, players have started to expect even more realism in games, such as unpredictability. Because of the games in this genre, many significant improvements on the game activities have been attached to characters run by the computer, in other words "non-player characters" (NPCs) which the player's character will meet during the game.

It can be said that the ultimate goal of an NPC is to be indistinguishable from a character run by a player. However, recent studies show [2, 14] that players will notice if their opponent is controlled by a computer rather than another human player, or if the opponent is too strong or too weak compared with another human player. According to the studies, the elements increasing the *believability* of NPCs as human players are natural movement, mistakes and gestures during the game, character appearance and character movement animation.

An NPC can be seen as an intelligent agent trying to do its best (rational action) in the current stage [21]. While choosing the rational action for an NPC, artificial intelligence (AI) tries at the same time to be as entertaining as possible, using even "cheap tricks" [16, 24]. Cheap tricks are permitted by the players as long as the player stays relatively convinced of the rationality of the actions.

In this paper, we take a look at how the game industry has been promoting non-player character's (NPC) believability in FPS games and compile a framework for evaluating believability of the NPCs. We start by looking at the elements which build believability in the next section and present the framework in Section 3. In Section 4, we apply our framework to three FPS games revealing improvements along the age of games. We conclude with final remarks in the last section, Section 5.

2 BUILDING BELIEVABILITY

Based on different studies among players [2, 14], the believability of the NPCs is most influenced by (1) the game environment where the NPCs appear, (2) another character or player which the NPC is compared to, and (3) the players' cultural background and age. Because we are aiming to a general framework we skip the last item and divide NPC's believability into three main categories: movement, animation and behavior. Next we consider how game developers have tackled each of these.

2.1 Movement

In FPS games, NPCs' movement usually tries to emulate humans' natural movement: finding the obvious shortest path and reacting to the game environment. One typical way of helping an NPC to find the shortest path is to build a navigation mesh onto the game map. Game designers plant varying number of navigation points or nodes onto the map. When the NPC searches for the shortest way to its destination, it actually calls for the search algorithm of the game to find the shortest path between the two navigation points: the one where the NPC is, and the one where it's going to go.

The most commonly used search algorithm is A* [16] and its variations. In some FPS games, designers have eased NPCs' pathfinding algorithms by outlining the area(s) where NPCs can move. This reduces the search space significantly and thus quickens the search. However, if an NPCs' destination is not within its range of movement or otherwise out of its reach, A* has to search trough every node in the search space, which particularly in large maps demands a great amount of CPU time. In case that game designers have not considered this option, the NPC calls for the pathfinding algorithm over and over again thus slowing the game down. In these cases, NPCs have been killed off so that the CPU time will not be wasted [9].

Even though the performance of computers has been rising continuously, optimizing algorithms are still needed to guarantee the smooth running of ever bigger games [25]. Reducing the navigation mesh is a simple and fast way to increase the speed of A*s, but it leads to a sparse search space and thus to a clumsy and angular NPCs' movement.

A good solution to optimize the searches is to cut down their number. This can be reached in two different ways: one is to reuse old searches for other NPCs and the other is to limit the flooding of A*. Flooding is understood as the extensive widening of pathfinding around the optimal path.

Even if A* can not find a path to a location, it is not wise to erase this search result from the computer's memory. In games where there are several NPCs, it is likely that some other NPC is going to search for a similar or even the same route at some point of the game. Now if the failed results are already in the memory, it saves a lot of the CPU time when there is no need to do the same search again [4]. Keeping a few extra paths in the memory does not notably limit the amount of free memory during the game.

By using path lookup tables, it is possible not to use any pathfinding algorithms during the game at all [2]. Every possible path will be stored in the lookup table, which is loaded in the memory while the game begins. Even though the tables will be quite large, it is still faster to look for a path straight from the table rather than search for the best route to the location. Major problems with the path lookup tables are that they require completely static maps and a lot of free memory.

Humans tend to move smoothly, that is, they attempt to foresee the upcoming turns and prevent too sharp turning. NPCs' paths can be smoothened in several ways. One is to use weighted nodes, in which case an NPC is surrounded by four sensors [3, 15]. Once a sensor picks up an obstacle or the gradient changes of the area, the sensor's value is increased. If the value exceeds the sensor's limit value, the NPC is guided away from the direction of the sensor.

Because an NPC knows which nodes it is going use in its path, it is possible to foresee where the NPC has to turn. Smoothening these turns by starting the turn before the node, game developers have been able to increase NPCs' believability [18]. Combining foreseeing the turn with string-pulling [3, 27], in which every node n_x is removed if it is possible for an NPC to go directly from node n_{x-1} to n_{x+1}, produces a very smooth, human-like movement for NPCs.

When several NPCs exist simultaneously, one must pay attention to how they move in groups. Problems occur when two or more NPCs try to use the same node at the same time. This can be solved by using reservations [17] in nodes, so that the first NPC reserves the node to itself, and the other NPCs have to find alternative paths. The reservation can be done by increasing the cost of one node so high, that A* ignores it while finding a path.

If several NPCs have to go through one particular node at the same time without having alternative paths, it can form a bottleneck for NPCs' smooth movement. One way to solve this is to let NPCs go through the bottleneck in a prioritized order [23]. This leaves low-priority NPCs to wonder around while waiting for their turn.

2.2 Animation

Most of the ***animations*** used by NPCs are made with one of the following three methods [2]. One is to draw or otherwise gain the frames of the entire animation and then combine them into to one sequence. The other method is to draw a couple of keyframes and later on morph them with computers into one

smooth animation. The third way is to connect sensors to a person and record the person's different moves onto the computer. Then these moves are fitted to a drawn character.

Each one of these methods has the same flaw: Once an animation is done, it can only be changed by recording it again. This obviously can not be done during the game. By recording several different animations for NPCs' one action, it is possible to change between different animations if the same action occurs again and again. This, however, only prolongs the obvious, which is that player will notice if dozens, or even a few, of NPCs limp or dies with in precisely the same way.

Using hierarchically articulated bodies or skeleton models, NPCs' animations can be adjusted to fit different situations and actions [2]. The skeleton models can also be fitted to different NPCs only by changing the model's appearance and size. The use of the skeleton models reduces the amount of memory needed for animations, because every animation is now done when needed instead of using pre-recorded sequences.

NPCs' appearance is very important while their believability is looked into. If gaps occur between the limbs and the torso, or other oddities can be seen in NPCs' appearance, it decreases their believability. While the polygon mesh is added over the skeleton these flaws can be avoided by paying attention to how the mesh is connected to the skeleton and by adding padding between the skeleton and the mesh [2].

The animation controller (AC) has an important role considering the NPC's animations. The AC decides what animation is played with each action and at what speed. In case of two animations are played sequentially, the AC decides at what point the switch happens. Some animations have a higher priority than others. Showing the death animation overcomes every other animation, because it is the last animation any NPC will ever do.

If NPCs are made with skeleton models, the AC needs to decide which bones are to be moved and how much in order to gain believable movement for an NPC. Some animations or movements can be shown simultaneously with other movements. These include, for example, running animation for the lower part of the body and shooting animation for the upper part while face movements for yelling are shown in the NPC's face.

Animations are even used to hide programming bugs in the games. In *Half-Life* when a player throws a grenade amongst a group of enemy NPCs, NPCs' pathfinding does not always find paths for NPCs to run away from the immediate explosion. Programmers at Valve Software could not localize this bug but they could see when the bug occured [13]. They programmed NPCs to duck and cover every time this bug appeared, and this solution was warmly welcomed by players saying it added an extra touch of human behavior to NPCs.

2.3 Behavior

Making mistakes is human, therefore it is not to be expected that any NPC's actions are flawless. Intentional mistakes, such as two NPCs talking loudly to each other or an NPC's noisy loading of guns, reveal the NPC's location to a player before he/she can even see it. NPCs' far too accurate shooting tends to frustrate the players so it is recommended that the first time an NPC sees the player's character, it should miss it thus giving the player time to react and shoot back [5, 13].

NPCs need reaction time for different actions to be able to imitate the physical properties of human [5, 14, 20]. These are made by adding one second delay for each action NPCs have, thus making them appear as if they were controlled by other players.

Both predictability and unpredictability are natural for human players [22]. In FPS games, this becomes apparent when either too weak or too powerful weapons are chosen in the game. Emergent behavior (EB) offers more unpredictability for NPCs. In EB, no simple reason can be given for the NPC's known actions and therefore the result of the action can benefit either the NPC or the player's character.

Emergent behavior occurs mostly when timers and goal-based decisions are used to control NPCs' behavior [19, 22]. Emergent behavior can also be a result from several small rules that NPCs follow. A

good example of this is flocking [3, 10, 19]. In flocking, every member of a flock or a group follows exactly the same rules, which can produce more action than the sum of these rules dictates.

Moreover, NPCs should take notice of other NPCs and their actions, and be aware of their existence. If game programmers so desire, NPCs can give support to each other or search for cover together [13]. In the worst case, a guard can walk over his fellow guard without even noticing his dead corpse on the ground [14]. However, it has been stated that the most important thing for NPCs to notice is to avoid friendly fire [20, 26].

NPCs can "cheat" by using information they possibly could not obtain in real life. These include locations of ammo and health in the game, the location of the players' characters' or even the characters' health and fighting capabilities [13, 22]. This information can be programmed for the player's benefit, too. By letting the player's character's health to drop to near zero and then by changing the NPCs from ultimate killing machines to sitting ducks, the game can give the player a feeling of a sudden success and thus keep him/her playing the game longer.

Lately, cheating of NPCs has been reduced by game programmers in order to give the player and the NPCs equal chances to survive in the game. At the same time, NPCs' abilities to autonomously search for health and ammo through different game levels and remember where it has or has not been have increased. Thus the change has been from cheating to more human-like NPCs [6, 12].

NPCs' behavior is mostly controlled by finite state machines [3, 2, 7, 28]. In addition to state machines, trigger-systems and scripts are used in state transitions. A more developed version of the state machine is a hierarchical state machine, in which every state is divided into smaller state machines which have their own states and state transitions [2].

3 DESCRIPTION OF FRAMEWORK

A framework for evaluating the believability of characters is a means to evaluate user-perceived NPC believability in FPS games. It should be noted that this framework is intentionally limited to provide simplicity and universality in use.

The framework is composed of two main aspects: firstly movement and animation, secondly behavior. It is based on programming techniques and algorithms used in different FPS games. This framework does not take a stance on how some requirement has been executed, but only whether or not it has been implemented so that the player can perceive it.

The basic element of NPCs' movements and animations is that any NPC can find the most suitable path to its destination. In most cases, NPCs' destination is the current location of the player's character. NPCs' path may not be the shortest, but it must be a reasonable suitable path. Because game maps are divided into smaller blocks to prevent too large search spaces, an NPC has to be able to cross these borders especially after it has noticed the player's character.

When NPCs move, they must move smoothly and be capable of avoid running into both static and dynamic obstacles. The player will not be convinced of NPCs' believability if it cannot move around a barrel or wait for a moving vehicle to move out of its way. When two or more NPCs move together, they must pay attention to each other to avoid collisions.

When observing NPCs' animations, three different things are of importance. First, one should note whether there are several pre-recorded animations for one action or not. Secondly, a shift from one pre-recorded animation to another must be fluent so that no unrealistic movements are made in between. Third, NPCs appearance must be done well enough so that no gaps can be seen between their limbs or other unnatural design is apparent.

Tables 1 and 2 show the specific propositions that are used in evaluating the believability of NPC characters. Propositions equal points, and the points are added into a score. Some propositions are viewed to have a greater impact on the believability. Therefore, some rows in Tables 1 and 2 are counted for doubling the score, i.e. the points for a single requirement can be 2 instead of 1. The importance of some requirements over others is based on the view taken in this study.

Table 1. Scores for movement and animation

Requirement for NPC	Points
NPC can find the most suitable path for its destination.	1
NPC's movement is not limited to a certain area, such as one room.	1
NPC's movement is not clumsy or angular.	2
NPCs are aware of each other and do not collide with each other.	1
NPC can avoid any dynamic or static obstacle in game field.	2
NPC has different animations for one action.	1
Shifting from one animation to another is fluent.	1
NPC's appearance is done carefully and no unnatural features can be found in it.	1
Total	**10**

NPCs' behavior is based on human's natural behavior. NPCs can and should make mistakes, and a way to make sure of this, is to program them to make intentional mistakes, vulnerabilities and reaction times. Emergent behavior gives a good illusion of an NPC being controlled by a human instead of a computer, and thus if possible, it should be present.

Taking notice of other NPCs can best be seen whether or not NPCs can avoid friendly fire. It is difficult to see if an NPC cheats. If an NPC does not collect ammo or health during the game, revealing their location to the NPC does no good to it. Instead, revealing the location of the player's character is easier to notice. If an NPC knows exactly when the player comes behind the corner, or an NPC shoots the player's character without the player noticing the NPC first, the NPC has cheated (at least it defined to be so).

Because of FPS games are typically fast paced, characters are in constant move. While an NPC moves, just like the player's character, it is hard for it to aim correctly and shoot at the target. Pausing for a moment before shooting at the player gives the player a fair chance to hide or shoot back. All this is based on information of human reaction times and aiming capabilities.

Finally NPCs' behavior should be logical and human. Even though it is desirable for an NPC to act unpredictably, running away from the combat which it was obviously going to win leaves the player perplexed. Running away from the combat which you are going to lose is human, but this characteristic feature of human behavior is not a typical action of an NPC – NPCs tend to fight till their untimely end.

Table 2. Scores for NPC's behavior

Requirement for NPC	Points
NPC makes intentional mistakes.	2
NPC has human-like reaction times.	2
NPC behaves unpredictably.	1
NPCs are aware of each other.	2
Cheating in a manner that player can not detect it.	1
Bad aim when seeing player for the first time	1
Logical and human behavior	1
Total	**10**

The overall score for an NPC is made by multiplying scores from both aspects. Therefore, the overall score is always somewhere between 0 and 100. It is good to note that even if a game scores, say, fair scores of 5 from movement and animation and 5 from behavior, its overall score will be as low as 5 * 5 = 25. Correspondingly, if a game receives an overall score of 81, it should gain very high $\sqrt{81} = 9$ on average from both tables.

Therefore, we split the multiplied score finally into one dimension with five grades with text labels: sub-standard (score of 0-9), weak (10-29), satisfactory (30-54), good (55-79) and excellent (80-100). Labeled grades are included because the scores become intuitively more understandable, compared to the mere numeral score.

The thresholds for each labeled grade differ from each other, because when the overall score is the result of the two multiplied sub-scores, it is more likely to gain a score from somewhere in the middle than a very low or a very high score. By changing the limits of the grades or the importance of a requirement would give different results than those described in this paper.

Despite what the overall grade an NPC receives, it is easy to see whether or not its main aspects are in balance between each other. If they are, a player may place NPCs believability higher than what it really is. Correspondingly, even if overall grade of an NPC is high but the aspects scores differ much, NPCs may seem more unbelievable to a player than what the grade suggests.

The chosen policy to multiply scores results into a zero score if one of believability aspects gives a zero score. Any self-respecting game developer should not release an FPS game which does not meet even one requirement of both aspects, because it shows nothing but negligence towards NPC believability.

4 APPLYING FRAMEWORK

We examined three different FPS games published between 1993 and 2001 by our framework. They were *Doom* (1993), *Quake II* (1996) and *Tom Glancy's Ghost Recon* (2001). The games were chosen because they represent the timeline of FPS game development from the player's viewpoint. The case studies were conducted with PC-versions of the games by playing the single player mode using the medium level of the games (Doom 3/5, Quake II and Ghost Recon 2/3). Possible differences of NPCs' believability caused by the levels of difficulty or multi-player vs. single-player modes are not included in the evaluation.

Doom received points as follows:

Table 3. Scores for Doom from movement and animation

Requirement for NPC	Points
NPC can find most suitable path for its destination.	1
NPCs are aware of each other and do not collide with each other.	1
NPC's appearance is done carefully and no unnatural features can be found in it..	1
Total	**3**

Table 4. Scores for Doom from behavior

Requirement for NPC	Points
NPC makes intentional mistakes.	2
Cheating in a manner that player can not detect it.	1
Total	**3**

The combined overall grade for *Doom* is 3*3 = 9, which is sub-standard. The scores from both aspects appear to be in balance.

Quake II received points as follows:

Table 5. Scores for Quake II from movement and animation

Requirement for NPC	Points
NPCs are aware of each other and do not collide with each other.	1
NPC can avoid any dynamic or static obstacle in game field.	2
Shifting from one animation to another is fluent.	1

NPC's appearance is done carefully and no unnatural features can be found in it.	1
Total	**5**

Table 6. Scores for Quake II from behavior

Requirement for NPC	Points
NPC makes intentional mistakes.	2
NPC has human-like reaction times.	2
Cheating in a manner that player can not detect it.	1
Total	**5**

The combined overall grade for *Quake II* is 5*5 = 25, which is weak. The scores from both aspects appear to be in balance.

Tom Glancy's Ghost Recon received points as follows:

Table 7. Scores for Ghost Recons movement and animation

Requirement for NPC	Points
NPC can find the most suitable path for its destination.	1
NPCs are aware of each other and do not collide with each other.	1
NPC can avoid any dynamic or static obstacle in game field.	2
NPC has different animations for one action.	1
Shifting from one animation to another is fluent.	1
NPC's appearance is done carefully and no unnatural features can be found in it.	1
Total	**7**

Table 8. Scores for Ghost Recons behavior

Requirement for NPC	Points
NPC makes intentional mistakes.	2
NPC has human-like reaction times.	2
Poor targeting when seeing player for the first time	1
Logical and human behavior	1
Total	**6**

The combined overall grade for *Ghost Recon*s is 7*6 = 42, which is satisfactory. Aspects are only 1 point apart from each other, so they are relatively well-balanced.

5 SUMMARY

Defining artificial intelligence has never been easy during its over 50-year-old history. Today AI research is based upon defining intelligence as intelligent behavior. Despite the fact that the first AI studies were done with board games, gaming has not been the driver in modern academic AI research. Contrary to academic AI research, game AI development has pursued to create an illusion of intelligence instead of trying to create one, ever since the 1970's when the first arcade games were introduced.

The first two decades in computer games were mostly attempts to increase the quality of graphics of the games, instead of concentrating on what was behind the glittering surface. Ever since the first FPS games came to the market in the early 1990's, NPCs' believability has gained more and more attention in the development of the games. The ultimate goal is that no player could distinguish a human player from a computer controlled one.

The means to improve NPCs' believability can be divided into three: movement, animation and behavior. Various algorithms and programming methods have been introduced and used by the game industry to improve NPCs' believability.

In this paper, we described a framework for evaluating the user-perceived believability of NPCs. The framework is divided into two main aspects, which both can be judged independently. The overall grade which an NPC or a game receives from the evaluation comes when the scores from both main aspects are multiplied together. The grade can be anywhere between 0 and 100 and is divided into five verbal grades: sub-standard (0-9), weak (10-29), satisfactory (30-54), good (55-79) and excellent (80-100).

We applied the framework to three FPS games and the overall scores were *Doom*: 9 (sub-standard), *Quake II*: 25 (weak) and *Tom Glancy's Ghost Recon*: 42 (satisfying). Based on these results, it can be concluded that the investments of the game industry on NPCs' believability since the 1990's has produced results: the newer the game, the more believable the characters.

The framework is simple, but it is aimed to serve as a first step in an area of great importance: to construct a neutral and general framework for evaluating contents of digital games. Similar framework can easily constructed for different games with emphasizes altered as needed. The results obtained are twofolded: first to evaluate existing games and second to influence to future games

REFERENCES

[1] Christian Baekkelund, Academic AI Research and Relations with the Games Industry. In AI Game Programming Wisdom 3, edited by Steve Rabin, Charles River Media Inc., 2006, pp 77-88.

[2] Penny Baillie-De Byl, Programming Believable Characters for Computer Games. Charles River Media Inc., 2004.

[3] Mat Buckland, Programming Game AI by Example. Wordware Publishing, Inc., 2005.

[4] Timothy Cain, Practical Optimizations for A* Path Generation. In AI Game Programming Wisdom, edited by Steve Rabin, Charles River Media Inc., 2002, pp 146-152.

[5] Delwin Clarke ja P. Robert Duimering, How Computer Gamers Experience the Game Situation: A Behavioral Study. ACM Computers in Entertainment, vol 4, no 3, June 2006, article 6.

[6] Dongkuy Chong, Tolga Konik, Negin Nejati, Chunki Park and Pat Langley, A Believable Agent for First-Person Shooter Games. *In Artificial Intelligence and Interactive Digital Entertainment Conference*, pp 71-73. June 6 – 8, 2007, Stanford, California.

[7] Dan Fu ja Ryan Houlette, The Ultimate Guide to FSMs in Games. In AI Game Programming Wisdom 2, edited by Steve Rabin, Charles River Media Inc., 2004, pp 283-302.

[8] Matt Gilgenbach, Fun Game AI Design for Beginners. In AI Game Programming Wisdom 3, edited by Steve Rabin, Charles River Media Inc., 2006, pp 55-63.

[9] Dan Higgins, Pathfinding Design Architecture. In AI Game Programming Wisdom, edited by Steve Rabin, Charles River Media Inc., 2002, pp 122-132.

[10] Geraint Johnson, Avoiding Dynamic Obstacles and Hazards. In AI Game Programming Wisdom 2, edited by Steve Rabin, Charles River Media Inc., 2004, pp 161-168.

[11] John E. Laird, It Knows What You're Going to Do: Adding Anticipation to Quakebot. *Proceedings of the Fifth International Conference on Autonomous Agents 2001*, May 28-June 1, 2001, pp 385-392. Montréal, Quebec, Canada.

[12] John E. Laird, Research in Human-Level AI Using Computer Games. Communications of the ACM, vol. 45, nro 1, January 2002, pp 32-35.

[13] Lars Lidén, Artificial Stupidity: The Art of Intentional Mistakes. In AI Game Programming Wisdom 2, edited by Steve Rabin, Charles River Media Inc., 2004, pp 41-48.

[14] Daniel Livingstone, Turing's Test and Believable AI in Games. ACM Computers in Entertainment, vol. 4, nro. 1, January 2006.

[15] Mike Mika and Chris Charla, Simple, Cheap Pathfinding. In AI Game Programming Wisdom, edited by Steve Rabin, Charles River Media Inc., 2002, pp 155-160.

[16] Alexander Nareyek, AI in Computer Games. ACM Queue, February 2004, pp 59-65.

[17] Jeff Orkin, Simple Techniques for Coordinated Behaviour. In AI Game Programming Wisdom 2, edited by Steve Rabin, Charles River Media Inc., 2004, pp 199-205.

[18] M. Pinter, Realistic Turning between Waypoints. In AI Game Programming Wisdom, edited by Steve Rabin, Charles River Media Inc., 2002, pp 186-192.
[19] Steve Rabin, Common Game AI Techniques. In AI Game Programming Wisdom 2, edited by Steve Rabin, Charles River Media Inc., 2004, pp 1-14.
[20] John Reynolds, Team Member AI in an FPS. In AI Game Programming Wisdom 2, edited by Steve Rabin, Charles River Media Inc., 2004, pp 207-215.
[21] Stuart Russell ja Peter Norvig, Artificial Intelligence – A Modern Approach, second edition, Prentice Hall, 2003.
[22] Bob Scott, The Illusion of Intelligence. In AI Game Programming Wisdom, edited by Steve Rabin, Charles River Media Inc., 2002, pp 16-20.
[23] David Silver, Cooperative Pathfinding. In AI Game Programming Wisdom 3, edited by Steve Rabin, Charles River Media Inc., 2006, pp 99-111.
[24] Paul Tozour, The Evolution of Game AI. In AI Game Programming Wisdom, edited by Steve Rabin, Charles River Media Inc., 2002, pp 1-15.
[25] Paul Tozour, Building a Near-Optimal Navigation Mesh. In AI Game Programming Wisdom, edited by Steve Rabin, Charles River Media Inc., 2002, pp 171-185.
[26] Paul Tozour, The Basics of Ranged Weapon Combat. In AI Game Programming Wisdom, edited by Steve Rabin, Charles River Media Inc., 2002, pp 411-418.
[27] Paul Tozour, Search Space Representations. In AI Game Programming Wisdom 2, edited by Steve Rabin, Charles River Media Inc., 2004, pp 85-102.
[28] Billy Yue and Penny de-Byl, The State of the Art in Game AI Standardisation. Proceedings of the 2006 international conference on Game research and development, ACM International Conference Proceeding Series, vol 223, pp 41-46.