Personality-based Adaptation for Teamwork in Game Agents

Chek Tien Tan and **Ho-lun Cheng**
Department of Computer Science
National University of Singapore
3 Science Drive 2, Singapore 117543
{tanchekt, hcheng}@comp.nus.edu.sg

Abstract

This paper presents a novel learning framework to provide computer game agents the ability to adapt to the player as well as other game agents. Our technique generally involves a personality adaptation module encapsulated in a reinforcement learning framework. Unlike previous work in which adaptation normally involves a decision process on every single action the agent takes, we introduce a two-level process whereby adaptation only takes place on an abstracted actions set which we coin as agent personality. With the personality defined, each agent will then take actions according to the restrictions imposed in its personality. In doing so, adaptation takes place in appropriately defined intervals in the game, without disrupting or slowing down the game constantly with intensive decision-making computations, hence improving enjoyment for the player. Moreover, by decoupling adaptation from action selection, we have a modular adaptive system that can be used with existing action planning methods. With an actual typical game scenario that we have created, it is shown that a team of agents using our framework to adapt towards the player are able to perform better than a team with scripted behavior. Consequently, we also show the team performs even better when adapted towards each other.

Introduction

Artificial Intelligence (AI) in games has evolved much in recent years, with various work being performed in the area of synthesizing intelligent behaviors in game agents (Hussain & Vidaver 2006; Horswill & Zubek 1999; Geramifard, Chubak, & Bulitko 2006; McDonald *et al.* 2006; White & Brogan 2006). A game agent is defined here as a fictional character in the game world. It can be either a player controlled character (PC) or a non-player character (NPC). NPCs either can be computer opponents that the PCs go against, allies that fight alongside the players or even innkeepers whom the players trades with.

Whilst there exists a vast number of challenges present in this area of research, we will focus on ally NPCs that adapt towards the PC as well as amongst each other. We find that this aspect is lacking as most current game AI research aims to create better opponents to challenge the player. However, it should be noted that our framework can be easily extended for the adaptive AI of the opponents as well. In this paper, we embark upon this area by introducing a new perspective of adaptivity using a concept we coin personality-based adaptation.

Copyright © 2007, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Motivation

From highly scripted ally NPCs in Blizzard's *Diablo II* (Blizzard 2006a) to highly dynamic and interactive ones in their recent *World of Warcraft* (Blizzard 2006b), AI in ally NPCs have improved much over the years. However, the ally NPCs in *World of Warcraft* still need lots of manual instructions in order for them to work the way the player expects, according to the player's unique strategy. This inadvertently places another restriction on the game, which is the number of NPCs a player can control, since it becomes rather impossible to control a whole team of NPCs when the population gets large.

In other modern games that boast intelligent agents like Sierra's *No One Lives Forever* (Sierra 2006), a group of ally NPCs "play" alongside the player in combat scenarios. The AI in these agents allow them to know how to avoid friendly cross-fires, automatically take up cover positions, and even perform flanking maneuvers on the enemies instinctively. However, the ally agents perform and react in the same way regardless of whoever is playing the game. Take for example a poorer player who tends to die when not enough agents are protecting him. Then, in certain scenarios where some agents always decide to flank the enemy, it leaves him with less ally protection and hence he is more likely to die repeatedly, resulting in frustration. On the other hand, an expert player may be frustrated when in certain situations his teammates move too slowly and carefully. In general, most AI in current games may react automatically to new unintended situations, but does not tailor to different styles and levels of players.

Our Approach

The work in this paper is driven by the grand goal of creating adaptive game agents that *automatically conform to individual styles and strategies of different players*, so as to effectively achieve common goals with the players. In this

paper, we generalize the goal of the agents to be able to *conform to another agent, regardless of whether it is the PC or NPC*. We also extend our work for each agent to be able to *conform to multiple agents*. In short, we have devised agents that adapt to:

- the PCs' personalities
- the other NPCs' personalities in a team

This enables adaptation to be possible in current massively multi-player games in which PCs and NPCs are mixed in a team, for example in the *Guild Wars* game series (NCsoft 2006) where multiple players combine with different classes of NPCs for game missions.

The approach presented in this paper is based on a learning framework centered around an adapter module which assigns different personalities to game agents, based on the personality of another agent. We introduce a novel way of injecting adaptivity in that we decouple the computationally intensive machine learning process from the low level action selection process. Other than saving precious game processing cycles, this also ensures modularity which enables our adaptive system to work with various action selection methods from simple rule-based action selection systems to newer paradigms like Goal-Oriented Action Planning (GOAP) (Orkin 2004).

In the remaining sections of this paper, we will first review work related to our area of research. We then define agent personality and formally define the problem. Thereafter, our solution framework is presented. Next, we show our experimental results as empirical proof of our concept. Lastly, we conclude the paper along with our plans for future work.

Related Work

Synthesizing intelligent game agents (Horswill & Zubek 1999; Hussain & Vidaver 2006; Geramifard, Chubak, & Bulitko 2006; McDonald *et al.* 2006; White & Brogan 2006) has been one of the core areas in game AI research. In an older study (Horswill & Zubek 1999), attention was focused on agents that could automatically choose actions that helped them to survive and attack. Hussain and Vidaver (Hussain & Vidaver 2006) used an evolutionary control mechanism originally intended for unmanned ground vehicles to control an NPC in a modern commercial game. In the work done by Mcdonald et al (McDonald *et al.* 2006), they devised a five-level architecture as an abstracted framework to enable inter-operability amongst different agent controllers. Common to these studies are the issues of usefulness and believability in the individual agent behaviors. We believe that the area of multi-agent intelligence is lacking in game agent research. Although the work done by Geramifard et al (Geramifard, Chubak, & Bulitko 2006) is directed at multiple game agents, they only concentrated on the pathfinding problem in real-time strategy games. However, White and Brogan (White & Brogan 2006) has presented a method to produce that employs a self-organizing Kohonen map coupled with reinforcement learning effective game-play in multiple agents in RoboCup simulated soccer. Nevertheless, their work did not address the issue of adaptability towards the human player. On the whole, these studies make use of decision-making systems that need perform heavy computation for each action the agents take, and frequently update the knowledge in the system, hence possibly causing disruptions to the actual game-play. This are the deficiencies the work in this paper targets to alleviate.

In player-centric approaches in games, though existing literature is scarce, there has been some exploratory work based on the notion of player modeling (Charles *et al.* 2005; Thue & Bulitko 2006; Yannakakis & Maragoudakis 2005). It can be also noted that all of them are rather recent, demonstrating the infancy of this research area. Charles et al (Charles *et al.* 2005) presented an extensive exploration of various ways to gather a player's profile, and then described ways of making use of it in different aspects of the game. Their work serves as a good overview for an introduction to common player modeling concepts. In the work done by Thue and Bulitko (Thue & Bulitko 2006), they introduced a concept which they term as goal-directed player modeling, in which they made use of the fact that quest goals are always known in advance in RPG games. Since quest goals are abstracts that roughly define what steps the player needs to take to accomplish them, this is used to predict the player's future actions. One limitation is that their method only applies to quests in RPG and puzzle type games. In the work done by Yannakakis and Maragoudakis (Yannakakis & Maragoudakis 2005), they predicted players' actions based on a Bayesian network with a subset of attributes of the current world state as the inputs. They were partially able to predict player actions in previously unseen circumstances. Generally, these studies uses techniques that aim to predict specific user actions ahead of time, and thereby act on them. This might be extremely tedious and resource intensive as player actions take place almost every second, again disrupting game-play.

Agent Personality

For any agent in the game (PC or NPC), its agent personality consists of a set of allowable actions, with each action tagged with a relative weight value as shown in Figure 1. Each action in the set is tagged with a certain value between 0 and 1. Formally defined, if P is the set of all agent personalities, the agent personality, $P_k \in P$, of an agent k is a function that assigns a value to each action.

$$P_k : A \rightarrow [0, 1], \tag{1}$$

where A is the set of all allowable actions.

As expected, this simply means that when an action selection mechanism is applied, the value determines the chance of choosing that action in view of other simultaneous actions. That is to say, we normalize their values to determine a probability that it can be chosen. Hence, to exhibit a certain personality, it means a combination of different action values in the set. As can be seen, this is a practical and flexible way to define personality in agents. A change in personality inevitably leads to a change of actions being exhibited, hence affecting the agent's behavior. Intuitively, this is also true in humans, as our personalities directly translate into the actions we take in life. As will be shown in the next few sections, adaptation only takes place on the personality, and

action planning can take place thereafter, independent of the adaptation process. In a simple sense, to use the personality with modern action planning paradigms like Hierarchical Task Networks (HTNs) (Wallace 2004) or GOAP (Orkin 2004), it means coupling each primitive action in the respective systems with the value as defined here. Also note that some actions like *"goto"* or *"die"* may be always executed whenever it is required of the agent to perform them. Hence they can be set to be non-adaptable and always given a value of 1, since they do not occur simultaneously with other actions. They are shown in Figure 1 as the non-shaded actions. In this paper, the issue of how we obtain the personality for a PC is not covered as it is not the focus in this paper.

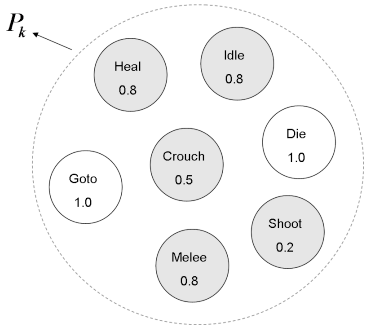

Figure 1: An example agent personality: Each action is tagged with a relative weight value that can be evaluated into a probability of choosing it. The shaded ones are the adaptable actions whilst the unshaded ones are non-adaptable (always having value 1).

Problem Formulation

A computer game can be represented as a world with attributes that describe it. An *attribute* is made up of an identifier and a value. Let the set of all attribute identifiers be I and the set of all attribute values be V. Depending on where each $v \in V$ is assigned to each $i \in I$, we would have a different *state* in which the world can exist.

$$\Gamma = \{\gamma : I \rightarrow V\}. \tag{2}$$

Therefore, the set of all states, Γ, is the set of functions $\gamma : I \rightarrow V$. For example, in a role playing game (RPG), we have $I = \{\text{monster\_health}, \text{player\_gold}, \ldots\}$ and $V = \{0, 50, 100, 888, \ldots\}$. In an arbitrary state γ, we have $\gamma(\text{monster\_health}) = 100$ and $\gamma(\text{player\_gold}) = 888$, which means that in this state the monster has full health of 100% and the player has 888 units of gold currencies.

As from the previous section, we define the set of all actions as A. An example can be $A = \{\text{melee}, \text{shoot}, \text{heal}, \ldots\}$ as previously seen in Figure 1. In the game world, the states can be changed by executing certain actions that alter the mapping of some attribute identifiers with their values (in other words, alter γ).

Now, we define the following functions:

1. An *adapter function*, ζ_k, that determines the agent personality, P_k of a certain kth game agent, via the agent personalities, P_i, of one or more other agents (which can be a PC or NPC), as well as the set of all actions, A.

$$P_k = \zeta_k(P_1, P_2, \ldots, P_i, \ldots, P_n, A), \tag{3}$$

   where $k = 1..n$ and $k \neq i$, with n being the total number of game agents.

2. A *behavior function*, β, that, for each kth game agent, generates a behavior plan, $B_k \in A^*$, from the set of all actions, A, given the agent's personality, P_k, a set of goal attributes, $I_g \subseteq I$, a goal state, $\gamma_g \in \Gamma$, and the current state, $\gamma_0 \in \Gamma$, of the world. The result, B_k, is basically a sequence of actions chosen from A.

$$B_k = \beta(A, P_k, I_g, \gamma_g, \gamma_0). \tag{4}$$

3. An *execute function*, ξ that executes a set of behavior plans such that the current world state, γ_0, is being changed. It means the output is a state of the world as a result of the behavior plans being executed.

$$\gamma_1 = \xi(B_1, B_2, \ldots, B_k, \ldots, B_n, \gamma_0), \tag{5}$$

   where $\gamma_1 \in \Gamma$ is the resultant world state.

Therefore, represented mathematically, the problem is to construct the adapter functions, ζ_k, such that the error, E, is minimized.

$$E = \sum_{g \in I_g} |\Delta_g(\gamma_1(g), \gamma_g(g))|, \tag{6}$$

where $\Delta_i(\gamma_x(i), \gamma_y(i)) = W_i(\gamma_y(i) - \gamma_x(i))$, $i \in I$ and $\gamma_x, \gamma_y \in \Gamma$, basically a weighted function to balance the contribution of each attribute.

Here we can see that the smaller E is, the closer it gets to the actual goal, and hence is a measure of how effective our adaptive system is in achieving the game goal.

Personality-based Adaptation Framework

At a macro level, our framework consists of the following steps as shown in Figure 2. It is basically a constant cyclic process that enables online training for the adapters.

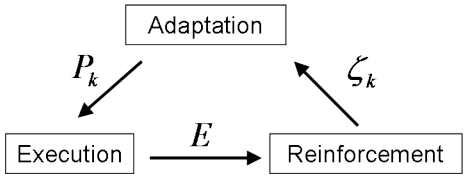

Figure 2: The adaptive game loop: a cyclic process that enables online training for the adapters.

Adaptation

The game loop starts with the adaptation process, which is also the core of our framework . During this process, each NPC agent, which is individually assigned an adapter module, activates its respective adapter to generate a personality of its own. An illustration of the adapter can be seen in Figure 3. It receives the personalities of the PC(s) as well as the

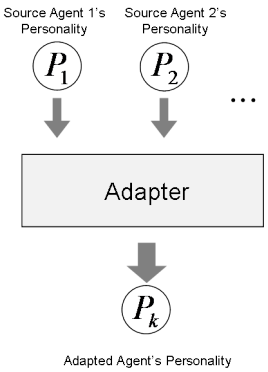

Figure 3: The adaptation process: for each agent, its adapter takes in either one or more agent personalities from other agents to generate its own personality.

rest of the NPCs as input, and generates its own personality, based on those. In this paper, the adapter is implemented via feed-forward neural network with a single hidden layer, although it can be any function approximation system in general. The back-propagation algorithm (Orr, Schraudolph, & Cummins 1999) is used for the network's learning processes. We chose neural networks because it can handle non-linear arbitrary relationships well, and deals well with large dimensions of input. This allows us to easily extend the adapter to large action sets as well as multiple agents. The network is initialized based on training with some synthetic data we have crafted, but the main learning is done online, at convenient intervals in the game. After each agent obtains its own personality, P_k, the game proceeds to the execution step.

Execution

Execution basically means the game is being played. It is worth pointing out that the end of execution does not always mean a "game over" or "you win" scenario, although it can be defined this way. Execution should end at a suitable predefined time in the game which does not disrupt game-play, for example, the next stage in a shooting game or quest transitions in a role-playing game. At the end of execution, the error, E, is calculated for the next step, namely, the reinforcement stage.

Reinforcement

Based on reinforcement learning terminology (Sutton & Barto 1998), a reward or punishment is given according to the feasibility of the results after execution ends. In our case, E is our punishment-to-reward ratio to the reinforcement learning system, with a value of 0 being a perfect result and a value of 1 being the worse result. The policy maker here would be the adapters collectively. This E is passed on to each adapter to enable it to update the weights of the neural network. At the end of reinforcement, basically an updated adapter, ζ_k, is being generated for each agent, to get them ready for the adaptation process.

Common to any online learning framework, we need to address the problem of exploration versus exploitation, namely, at each loop, the adapters choose whether to exploit the learnt knowledge and generate a well adapted personality (as of the current loop), or to generate a random personality to possibly create new learning instances for the adapters. We will adopt the standard ϵ-greedy algorithm to tackle this issue, with $\epsilon = 0.2$. This basically means that at the adaptation step, each adapter will generate a random personality with probability ϵ, and exploit the knowledge to generate the best adapted personality otherwise.

We can see that the adaptation and reinforcement processes only happen in between executions, hence eliminating disruptions to the main game-play. Also, the occasional exploration mechanism (choosing a new random personality) adds variety to NPC behaviors which also enhances the entertainment value for players (Spronck 2005).

Empirical Evaluation and Discussion

In order to evaluate our framework in a real game environment, we have created a typical action game scenario built on the Truevision3D 6.2 game engine. A screenshot is as shown in Figure 4. Although the results to be shown here are based on an action game close to that of an FPS, the framework can be applied to any game in general.

Game Mechanics

The goal of the game is to kill all the zombies in the game world. The PC is a marine equipped with a laser gun with limited range. The player is accompanied with 5 ally NPCs carrying the same type of gun. The PC and his NPCs can also run up close to melee damage the opponent. The zombies are packed in groups and when any of the agents are in range, they will approach the respective agent and melee attack by biting them. The game ends when either all zombies are killed or the PC is killed. An agent's melee attack does double the damage compared to shooting because the agent exposes itself to the attacks of the zombies and attracts more zombies towards it. Also, all attacks have a chance of hitting critically for twice the damage.

Figure 4: A screenshot of our experimental game scenario. The PC is in the center, and each of the other NPCs in white are busy with their own tasks chosen. The opponents are the zombies scattered over the map in packs.

In our experiments, we will only allow the actions *melee*, *shoot* and *heal* to be adaptable. An example personality of that used in our experimental setting is as shown in Figure

1. Our action planning system for the experiments is generally a rule-based system with the NPCs following the PC wherever he goes. At each zombie encounter, the NPCs will choose to either run up close and melee the zombie, stay far and shoot it, or if an agent's health is less than full, decide whether to heal him. And the choice of these three actions are dependant on the probabilities defined in the agents' personalities.

Experiments

To enable a large enough sample size of experiments, we have scripted the PC to allow numerous repetition runs. Initially, we run the game scenario 500 times with scripted NPCs with each NPC personality being fixed at random values. Next, we make 500 iterations again with each NPC having its adapter which only adapts to the PC. In a third set of 500 iterations, the adapters will adapt to the PC as well as all the other NPCs in the team. In each iteration, the positions of the zombies are re-shuffled randomly (but with all sets of experiments having the same random seed to ensure fairness). Hence this experiments aim to test whether the personality adapted NPCs are better than scripted ones, and also to test the online adaptability of our framework.

The attributes used in the calculation of E in our experiments are the total number of ally NPC agents dead and the total number of zombies alive, with a weightage of 0.3 and 0.7 respectively. These two attributes are the most straightforward metrics to determine the success rate of the game. More weightage was given to the former attribute because the goal of the game is to eliminate the zombies, and that the latter attribute is more of a factor to determine teamwork effectiveness, though it does have an effect on the game's goal.

The results are as shown in Figure 5. The agents using our framework that only adapt to the PC performs better as compared to scripted agents, with the adaptive agents producing a total average of $E = 0.576$ versus a total average of $E = 0.628$ for scripted ones. Consequently, when using agents that adapt to all other agents, it performs vastly better than both the scripted and the agents that only adapt to the PC, with a total average of $E = 0.393$. This might be due to the fact that the team of agents that adapt to each other has implicitly learnt cooperative behavior with the entire team, whilst the team of agents that only adapt to the PC only learns how to complement the PC.

We also include 2 further experiments to show the adaptability cross different player personalities. In Figure 6, it shows the results where we fixed the melee and healing attributes at 0.5 each respectively, and vary the shooting probability from 0 to 1. Similarly, for the results in Figure 7, we fix the melee and shooting attributes at 0.5 whilst varying the healing attribute from 0 to 1. In terms of the actual averages, for the first experiment as in Figure 6, it is $E = 0.533$ for scripted agents, $E = 0.392$ for agents that adapt only to the PC, and $E = 0.340$ for agents that adapt to each other. For the other experiment as in Figure 7, it is $E = 0.538$, $E = 0.441$, and $E = 0.416$ in the same order. Hence the same conclusion as those deduced from Figure 5 can be drawn here also, now generalized to more instances

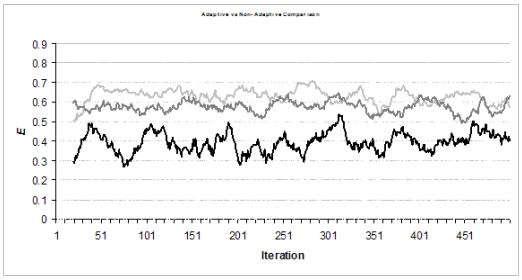

Figure 5: Experimental Results: Plot of E against number of iterations. The lightest colored line at the top depict the results with non-adaptive agents, the darker line below it shows the results with adaptive agents but only towards the PC, and the darkest line at the bottom shows the results with adaptive agents that adapt to all other agents. The lines are moving averages of 20 iterations so as to eliminate the noise of the actual iterations.

of the PC personalities, namely that the team-based adaptive agents perform better than player-only adaptation and scripted agents in various configurations of the PC personalities. However, we see that the improvements gained in healing is lesser comparatively. As shown in Figure 7, as the player's healing attribute get higher, the performance of both types of adapted agents gets poorer and closer to the scripted agents, likely due to the fact that he gets killed more often (and when the PC is killed, it marks the end of an execution). This is because in the game mechanics, the healer is especially vulnerable to death as the zombies are scripted to attack healers first because they pose a greater threat.

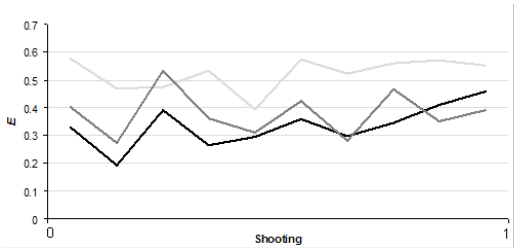

Figure 6: Experimental Results: Plot of E against the shooting probability. The lightest colored line at the top depict the results with non-adaptive agents, the darker line below it shows the results with adaptive agents but only towards the PC, and the darkest line at the bottom shows the results with adaptive agents that adapt to all other agents.

Conclusion and Future Work

To conclude, we have presented our initial efforts to devised a framework that enables player-centric adaptability of game agents. We have also introduced a novel concept of personality in game agents as a pre-step before actual action selection. In doing so, we reduce computational overhead as adaptation only takes place on the personality, and personal-

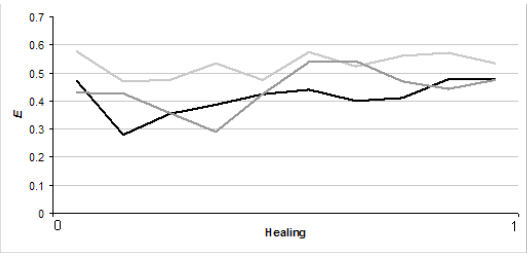

Figure 7: Experimental Results: Plot of E against the healing probability. The lightest colored line at the top depict the results with non-adaptive agents, the darker line below it shows the results with adaptive agents but only towards the PC, and the darkest line at the bottom shows the results with adaptive agents that adapt to all other agents.

ity adaptation only needs to be performed at convenient interludes in the game. It also enables modularity in the adaptive framework so that it can be used with various state of the art action planning methods. Moreover, with the personality defined commonly across both PCs and NPCs, the source of personality adaptation can be either the PCs or NPCs. We have also shown that our work can be easily extended to facilitate a network of adaptation needed in group scenarios. In modern multi-player games, it is common to see a mixture of PCs and NPCs "playing" together, hence this will be inevitably useful in recent games.

An immediate extension to our work can be applying the framework to the NPC opponents in a game. For example, the NPC opponents can take in the PCs personality and consequently adjust its attack power to create an appropriate level of challenge for different players. We can also include the opponent personalities as extra inputs to the ally NPCs, as a further source of environmental knowledge to deduce better personalities for themselves. As future work, we will be embarking on an evaluation of better machine learning paradigms to implement the adapter module and the reinforcement process, so as to increase the improvements we gain. We also would like to investigate the feasibility of pairing our personality-based adaptation system with state of the art action planning systems like HTNs and GOAP. Another aspect we hope to enhance in the future is the inclusion of an enjoyment variable in our empirical evaluations. In this work, our definition of "goodness" is only confined to whether the NPCs can complete goals feasibly. We recognize that in commercial games, entertainment value is what defines the "goodness", hence we hope to evaluate the improvements we have attained in this aspect next time. As it is very difficult to define and evaluate enjoyment in a technical sense, we will be employing a survey method, whereby users will actually play our game and be asked to fill out a questionnaire thereafter.

References

Blizzard. 2006a. Diablo ii. Accessed April 12, 2006. Available via http://www.blizzard.com/diablo2/.

Blizzard. 2006b. World of warcraft. Accessed April 12, 2006. Available via http://www.worldofwarcraft.com/.

Charles, D.; Kerr, A.; McNeill, M.; McAlister, M.; Black, M.; Kcklich, J.; Moore, A.; and Stringer, K. 2005. Player-centred game design: Player modeling and adaptive digital games. *In Proceedings of the Digital Games Research Conference* 285,298.

Geramifard, A.; Chubak, P.; and Bulitko, V. 2006. Biased cost pathfinding. *In Proceedings of the Artificial Intelligence and Interactive Digital Entertainment conference* 112,114.

Horswill, I., and Zubek, R. 1999. Robot architectures for believable game agents. *In Proceedings of the 1999 AAAI Spring Symposium on Artificial Intelligence and Computer Games, AAAI Technical Report SS-99-02*.

Hussain, T. S., and Vidaver, G. 2006. Flexible and purposeful npc behaviors using real-time genetic control. *In Proceedings of The IEEE Congress on Evolutionary Computation* 785,792.

McDonald, D.; Leung, A.; Ferguson, W.; and Hussain, T. 2006. An abstraction framework for cooperation among agents and people in a virtual world. *In Proceedings of the Second Conference on Artificial Intelligence and Interactive Digital Entertainment*.

NCsoft. 2006. Guild wars. Accessed April 12, 2006. Available via http://www.guildwars.com/.

Orkin, J. 2004. *Applying Goal-Oriented Action Planning to Games*. Hingham, Massachusetts: Charles River Media, first edition.

Orr, G.; Schraudolph, N.; and Cummins, F. 1999. Cs-449: Neural networks lecture notes. Accessed December 20, 2005. Available via http://www.willamette.edu/ gorr/classes/cs449/intro.html.

Sierra. 2006. No one lives forever 2. Accessed April 12, 2006. Available via http://nolf.sierra.com/.

Spronck, P. 2005. A model for reliable adaptive game intelligence. *In IJCAI-05 Workshop on Reasoning, Representation, and Learning in Computer Games* 95,100.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. Cambridge, Massachusetts: The MIT Press.

Thue, D., and Bulitko, V. 2006. Modeling goal-directed players in digital games. *In Proceedings of the Artificial Intelligence and Interactive Digital Entertainment conference* 285,298.

Wallace, N. 2004. *Hierarchical Planning in Dynamic Worlds*. Hingham, Massachusetts: Charles River Media, first edition.

White, C., and Brogan, D. 2006. The self organization of context for multi agent games. *In Proceedings of 2nd Annual Conference on Artificial Intelligence in Interactive Digital Entertainment*.

Yannakakis, G. N., and Maragoudakis, M. 2005. Player modeling impact on players entertainment in computer games. *In Springer-Verlag: Lecture Notes in Computer Science* 3538:74.