Modeling Motivation for Adaptive Nonplayer Characters in Dynamic Computer Game Worlds

KATHRYN MERRICK
University of Sydney, Australia

Current computer games are being set in increasingly more complex and dynamic virtual environments. Massively multiplayer online games, for example, are played in persistent virtual worlds, which evolve and change as players create and personalize their own virtual property. In contrast, technologies for controlling the behavior of nonplayer characters that populate virtual game worlds are frequently limited to preprogrammed rules. Characters using fixed rule-sets lack the ability to adapt in time with their environment. Motivated reinforcement learning offers an alternative to character design that can achieve nonplayer characters that both evolve and adapt in dynamic environments. This article presents and evaluates two computational models of motivation for use in nonplayer characters in persistent computer game worlds. These models represent motivation as an ongoing search for novelty, interest, and competence. Two metrics are introduced to evaluate the adaptability of characters controlled by motivated reinforcement learning agents using different models of motivation. These metrics characterize the behavior of nonplayer characters in terms of the variety and complexity of learned behaviors. An empirical evaluation of characters in simulated game scenarios shows that characters motivated by the search for competence are more adaptable in dynamic environments than those motivated by interest and novelty alone.

Categories and Subject Descriptors: I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search – *Dynamic programming, Heuristic methods*; I.2.6 [**Artificial Intelligence**]: Learning

General Terms: Algorithms

Additional Key Words and Phrases: Motivation, reinforcement learning, computer games, neural nets, nonplayer characters, persistent virtual worlds

ACM Reference Format:
Merrick, K. 2007. Modeling motivation for adaptive nonplayer characters in dynamic computer game worlds. *ACM Comput. Entertain.*, 5, 4, Article 5 (March 2008), 32 pages. DOI = 10.1145/1324198.1324203 http://doi.acm.org/10.1145/1324198.1324203

1. INTRODUCTION

Motivated reinforcement learning (MRL) offers a new approach to the design of behaviors for nonplayer characters (NPCs) in computer games [Merrick and Maher 2006]. A current focus of MRL research is on producing agents that can learn a variety of believable behaviors in complex environments and adapt those behaviors in dynamic environments. As such, MRL techniques offer a promising approach to the design of NPCs that can adapt in complex or dynamic environments. MRL is an approach to learning by trial and error in response to reward or punishment. In MRL, reward and punishment are modeled as a motivation signal that encourages general behavioral char-

Author's address: School of Information Technologies, J12, University of Sydney, NSW, 2006, Australia; email: kkas0686@it.usyd.edu.au

acteristics such as the desire to learn novel or interesting behaviors. The resulting diversity of learned behaviors has the potential to create NPCs that are more flexible and responsive than those using existing techniques.

Massively multiplayer, online role-playing games (MMORPGs) are a current example of computer games in complex, persistent, dynamic virtual environments. In MMORPGs, NPCs take on the role of enemies, partner characters, and support characters, and have the potential to benefit from MRL technologies. Previous approaches to the design of support characters, in particular, have tended to use fixed rules and looping animations so that characters are frequently unable to respond to the evolving environment [Laird and van Lent 2000]. More sophisticated NPCs have the potential to enrich game worlds by providing opportunities for interesting interactions with players. This makes the game world more interactive and improves the believability of the game [Zeltzer 1992].

Previous work with MRL in MMORPGs [Merrick and Maher 2006] and simulation games [Merrick and Maher 2007] has demonstrated the abilities of support characters controlled by MRL agents motivated by the search for novelty and interest. This work has shown, qualitatively, that characters using MRL can exhibit adaptive behavioral patters for performing different tasks at different times. However, existing work has not quantified the extent to which agents achieve these behavioral characteristics. Likewise, alternative approaches to modeling motivation that may achieve different behavioral characteristics have not been considered.

This article looks, in depth, at the role of motivation in creating adaptive NPCs using MRL. Two computational models of motivation are presented for the control of NPCs in MMORPGs. These models represent motivation both as an ongoing search for novelty and interest and as the search for competence. In order to quantify the effect of different models of motivation on the behavioral characteristics of NPCs, two metrics are introduced to evaluate the adaptability of NPCs controlled by MRL agents using different models of motivation. Specifically, the behavior of NPCs is characterized in terms of its variety and complexity. An empirical evaluation of MRL for controlling NPCs in simulated game scenarios shows that NPCs motivated by the search for competence are more adaptable in dynamic environments than those motivated by interest and novelty alone.

2. CURRENT TECHNOLOGIES FOR CONTROLLING THE BEHAVIOR OF NONPLAYER CHARACTERS

NPCs fall into three broad categories: enemies, partners, and support characters [Laird and van Lent 2000]. Enemies are characters that oppose human players in a pseudo-physical sense by attacking the virtual life force of the human player with weapons or magic. Partners take the opposite role and attempt to protect the human players with whom they are allied. Alternatively, partner characters might perform noncombat tasks such as selling goods on behalf of their human ally. In some games, partner characters may be taught to perform certain behaviors by players. Finally, support characters are the merchants, tradesmen, guards, innkeepers, and so on who support the storyline of the game by offering quests, advice, goods for sale, or training.

The basic architecture for an NPC must be highly reusable – so that it can be applied to different characters – and support humanlike behavior [Travis et al. 2000]. A common approach to the control of NPC behavior is to use an agent architecture [Wooldridge and Jennings 1995]. Agent architectures define a framework for the processes an NPC must execute repeatedly during game-play: sensing the environment, reasoning about sensor

data, and acting in the environment. This set of recurring processes is an NPC's intelligence loop [Baille-De Byl 2004]. Different approaches have been taken to implementing the intelligence loop. The approach chosen determines the complexity, flexibility, and adaptability of a character's behavior.

2.1 Reflexive Agents

Reflexive behavior [Maher and Gero 2002] is a preprogrammed response to the state of the environment – a reflex without reasoning. Only recognized states will produce a response. State machines, rule-based approaches, fuzzy logic, and flocking are reflexive reasoning processes commonly used to control the behavior of NPCs.

Rule-Based Systems

Games that use characters with rule-based reasoning include Baldur's Gate, FX Fighter, and Terra Nova. Rule-based approaches define a set of rules about states of the game world of the form: `if <condition> then <action>` [Russel and Norvig 1995]. If the NPC observes a state that fulfils the `<condition>` of a rule, then the corresponding `<action>` is taken. Only states of the world that meet a `<condition>` will produce an `<action>` response. While this approach ensures a fixed set of responses by a character, the details of these rules must be established prior to characters interacting with their environment. Any adaptation in behavior can only occur within the fixed set of predefined rules.

State Machines

State machines can be used to divide an NPC's reasoning process into a set of internal states and transitions [Baille-De Byl 2004]. Each internal state defines rules controlling how the agent should respond to its environment while in that internal state. State machines are one of the most popular control strategies for NPCs, used in games including Dungeon Siege, Age of Empires, The Sims, Enemy Nations, and Half-Life. Nonetheless this approach has similar issues to rule-based systems. Internal states, like rules, are established prior to characters interacting with their environment. While different rules in different internal states can lead to behavior that is more adaptable than rule-based systems, this adaptability is again limited by the predefined rules within each internal state.

Fuzzy Logic

Fuzzy logic provides a way to infer a conclusion based on facts that may be vague, ambiguous, inaccurate, or incomplete [Baille-De Byl 2004]. *Close Combat* 2 is an example of a game using fuzzy logic. Fuzzy logic uses fuzzy rules of the form `if <X is A> then <Y is B>.` X and Y are linguistic variables representing characteristics being measured – such as temperature, speed, or height – while A and B represent fuzzy categories – such as hot, fast, or tall. Fuzzy categories define decision thresholds within which certain courses of action may be pursued. Fuzzy logic can be applied to both rule-based approaches and state machines. While fuzzy logic allows characters to reason in environments where there is uncertainty, the ability of characters to adapt is still limited by the set of predefined fuzzy rules.

Flocking

Flocking [Reynolds 1987] is special example of rule-based reasoning used to control groups of characters such as crowds or animals. Flocking uses three rules governing the separation, alignment, and cohesion of individuals in a flock, herd, or other kind of group. Separation rules steer an individual to avoid others, alignment rules steer an

individual towards the average heading of the flock, and cohesion rules steer an individual towards the average position of the flock. Flocking algorithms have been used with great success to represent lifelike crowd and animal movements and have been incorporated in games such as *Half-Life* and *Unreal*. Using the basic flocking rules, flocks can adapt to changes in their environment by moving around, towards or away from objects. However, flocking does not allow character individuality or more complex adaptation.

2.2 Learning Agents

Learning agents are able to modify their internal structures in order to improve their performance with respect to some task [Nilsson 1996]. Some NPCs such as partner characters can be trained to learn behaviors specified by their human master. The human provides the NPC with rewards such as food or patting to encourage desirable behavior and punishment to discourage unwanted actions. Learning algorithms used in games include decision trees, neural networks, and reinforcement learning.

Decision Trees

Decision trees are hierarchical graphs that structure Boolean functions [Russel and Norvig 1995]. A decision tree is learned from a training set of previously made decisions. The learned decision tree can then be used to make future decisions. In *Black and White*, for example, creatures can learn decision trees about what food to eat based on how tasty the creature finds previously eaten food provided by a human player. While decision trees allow characters to learn, thus permitting more adaptable characters than do reflexive approaches, they require a set of examples from which to learn. These examples must be provided by players. While this is appropriate for partner characters, it is generally inappropriate for enemies and support characters to have their behaviors determined only by players.

Neural Networks

Artificial neural networks comprise a series of neurons with interconnecting pathways [Russel and Norvig 1995]. Neural networks, like decision trees, learn from examples. Examples of correct actions in different situations are fed into the network to train a character. When a character encounters a similar situation it can make a decision about the correct action to take based on the data stored in the neural network; neural networks are used by characters in *Battlecruiser: 3000 AD*. However, in many cases neural networks are frozen before the release of a game to prevent further learning during the game because learning from character actions can produce networks capable of adapting erratically or unpredictably to players' actions.

Reinforcement Learning

Reinforcement learning (RL) agents [Sutton and Barto 2000] are connected to their environment by sensation and action. On each step of interaction with the environment, the agent receives an input that contains some indication of the current state of the environment and the value of that state to the agent. This value is called a reward signal. The agent records the reward signal by updating a behavioral policy that represents information about the reward received in each state sensed so far. The agent then chooses an action that attempts to optimize the reward gained over time. Because RL is directed by reward and not examples, it is possible for game designers to provide a reward signal for a character prior to the character's learning during a game. This means that RL approaches, unlike decision trees and neural networks, are suitable for enemy and partner characters. Researchers from Microsoft have shown that it is possible to use RL

to allow enemies to improve and adapt a skill during a game. RL was applied to fighting characters for the Xbox game, *Tao Feng* [Graepel et al. 2004]. However, while *Tao Feng* characters using RL can adapt their fighting techniques over time, it is not possible for them to identify new skills to learn about. This is because they are limited by a preprogrammed reward for a single skill – fighting. The key difference between RL and MRL lies in the replacement of the skill-specific reward signal in RL with a general motivation signal in MRL. The motivation signal allows characters to identify multiple different tasks or skills about which to learn.

Evolutionary Agents

Evolutionary approaches such as genetic algorithms simulate the process of biological evolution by implementing concepts such as natural selection, reproduction, and mutation. Individuals in a population are defined in terms of a digital chromosome. When individuals reproduce, offspring are defined by a combination of their parents' chromosomes via processes of crossover and mutation. Offspring are then evaluated using a fitness function to determine which will remain in the population and which will be removed (die). Evolutionary algorithms are robust search methods that can optimize complex fitness functions. However, when genetic algorithms are used in NPCs, fitness functions must be predefined by game designers. As in RL, the fitness function limits the adaptability of a given population of individuals to the skills or tasks defined by the fitness function.

3. MOTIVATED REINFORCEMENT LEARNING

RL agents learn by trial and error through direct interaction with their environment. Learning is directed by rewards and punishments that guide an agent towards a specific, predefined task or skill. MRL introduces a motivation signal into the RL framework. Where reward defines a specific task, motivation may inspire learning of multiple tasks at different times in response to general concepts such as interest or curiosity. MRL algorithms occur in two broad classes: MRL(I) algorithms incorporate a motivation signal in addition to a reward signal, while MRL(II) algorithms use a motivation signal instead of a reward signal. Within these classes, motivation has been used for a range of different purposes. A number of MRL(I) models use motivation in conjunction with a reward signal as a means of speeding up learning [Singh et al. 2005]. Other MRL(I) models use motivation as an exploration mechanism either to facilitate later behavioral exploitation [Simsek and Barto 2006] or to direct action while waiting for task-oriented reward feedback [Huang and Went 2002. In contrast, MRL(II) approaches use motivation in the absence of a reward signal as an automatic attention-focus mechanism [Kaplan and Oudeyer 2003; Merrick and Maher 2006; Oudeyer and Kaplan 2004; Oudeyer et al. 2007; Saunders and Gero 2001]. MRL(II) models have potential as adaptive control algorithms for NPCs because attention-focus is determined by experience-based motivation only. A task-oriented reward defined prior to learning based on rules about the environment is not required. Rather, the motivation component is designed to be general enough to direct learning towards different tasks that might arise without requiring prior knowledge of what those tasks may be or when they should be performed. Because MRL is based on RL, it offers a means of autonomous learning for NPCs that does not require direct supervision or examples by human player characters. This means that MRL can potentially be applied to partner, enemy or support characters.

Merrick and Maher [2007] developed an MRL(II) approach for NPCs in complex, dynamic game environments. In their approach, complex, dynamic environments are modeled as a set of Markov decision problems (MDPs), $P = \{P_1, P_2, P_3 \ldots\}$, in which

each P_i represents a region of the environment in which a task is situated. P_i comprises a set S_i of states, a set A_i of actions, and a task defined by a reward function $\mathcal{R}_i$.

The key differences between this MRL algorithm and other RL and MRL approaches lie in:

- the use of experience-based motivation rather than task-oriented reward to focus learning;
- learned policies representing behavioral cycles; and
- context-free grammar representations for the state and action spaces.

In order for a RL agent to learn to perform a task in an environment modeled by a set **P** of MDPs, a reward signal $\mathcal{R}_i$ is required that represents that task. In the past, this reward signal has been constructed using known rules about the task or the environment. For example, an NPC representing a guard character can use RL to learn a patrol route, given a reward signal for each geographic location along the route. However, this reward signal assumes detailed knowledge of the layout of the environment prior to learning.

The role of motivation in MRL is to provide an approach to the design of reward signals for different tasks without the need to identify subsets of the state or transition space prior to learning to compose the rules that will define each task. For example, a MRL agent motivated to move to interesting areas could determine a patrol route without requiring detailed knowledge of the layout of an environment prior to learning. In addition, if changes in human activity change what is interesting in the environment while the game is in progress, the agent can adapt its patrol route appropriately.

More formally, motivation $R_{m(t)}$ is a function of an agent's experiences:

$$R_{m(t)} = \mathcal{R}(\mathbf{Y}_{(t)})$$

where experiences $\mathbf{Y}_{(t)}$ are modeled as a single, infinite trajectory of states and actions:

$$\mathbf{Y}_{(t)} = S_{(1)}, A_{(1)}, S_{(2)}, A_{(2)}, \ldots S_{(t)}, A_{(t)}$$

The aim of motivation in MRL is to produce motivation values that will motivate the RL process to focus on executing actions that support the emergence of focused behavioral cycles in the state and action trajectories. Behavioral cycles allow the development of agents that can mimic the biological, cognitive, or social cycles in natural systems [Mook 1987]. This is useful, for example, in applications for characters in persistent computer games that are required to exhibit realistic or humanlike behavioral patterns that can adapt over an extended lifetime.

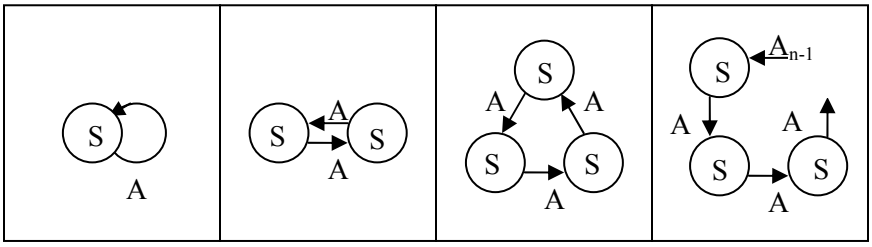


Fig. 1. Behavioral cycles of complexity one, two, three, and n.

The complexity of the behavioral cycle is defined as the number of states or actions in the trajectory. Four behavioral cycles of different complexities are illustrated in **Fig. 1**. The key ability of MRL that is not available using other agent models is the capacity for MRL agents to adapt their behavior autonomously to form different cycles, without requiring these cycles to be preprogrammed.

In this MRL model, sensed states $S_{(t)}$ are represented as a string from a context-free grammar (CFG) [Merceron 2001] ($\mathbf{V}_S$, $\mathbf{\Gamma}_S$, $\mathbf{\Psi}_S$, S) where

- $\mathbf{V}_S$ is a set of variables or syntactic categories;
- $\mathbf{\Gamma}_S$ is a finite set of terminals such that $\mathbf{V}_S \cap \mathbf{\Gamma}_S = \{\}$;
- $\mathbf{\Psi}_S$ is a set of productions of the form V → v where V is a variable and v is a string of terminals and variables; and
- S is the start symbol.

Thus, the general form of a sensed state is

```
S                    →    <sensations>
<sensations>         →    <PiSensations><sensations> | ε
<PiSensations>  →    <sL><PiSensations> | ε
<sL>                 →    <number> | <string>
<number>             →    1 | 2 | 3 | ...
<string>             →    ...
```

This representation is flexible enough to represent environments containing different numbers of elements $\mathbf{P}_i$. This is important in dynamic game worlds where the elements comprising **P** may change over time as players modify the game environment by building or crafting new objects. This representation is also flexible enough to represent objects with different properties. For example, one object may have a shape and color, while another may also have an age or usage limit, and so on.

In fixed-length vector representations two states can be compared by comparing the values of vector elements with the same index in each state. In variable length sensed states, a label L is assigned to each sensation by the sensor that produced it, such that two states can be compared by comparing the values of sensations that have the same label where such sensations exist.

A flexible representation is also required for the action space in dynamic environments. While characters may maintain a fixed set of effectors for the duration of their lives, the actions they can perform with those effectors may change with the addition or removal of elements P_i from the environment. Thus, the action space **A** is also represented using a CFG ($\mathbf{V}_A$, $\mathbf{\Gamma}_A$, $\mathbf{\Psi}_A$, **A**) where

- $\mathbf{V}_A$ is a set of variables or syntactic categories;
- $\mathbf{\Gamma}_A$ is a finite set of terminals such that $\mathbf{V}_A \cap \mathbf{\Gamma}_A = \{\}$;
- $\mathbf{\Psi}_A$ is a set of productions of the form V → v where V is a variable and v is a string of terminals and variables; and
- **A** is the start symbol.

The general form of the action space is:

```
A                    →    <actions>
<actions>            →    <PiActions><actions>  | ε
<PiActions>          →    <Aj><PiActions> | ε
<Aj>                 →    ...
```

The algorithm in Figure 2 represents the basic MRL control strategy for character reasoning, which achieves attention-focus that supports life-long, adaptive, multitask learning in complex, dynamic environments. MRL is modeled as a continuing task, temporal difference learning algorithm. The algorithm is arranged in five phases concerned with sensing the environment (line 3); action selection (lines 4-6); experience-based attention-focus using motivation (lines 7-9); learning (line 10); and activation (line 12). In this article, Q-learning [Watkins and Dayan 1992] is used as the temporal difference learning approach.

4. MODELING MOTIVATION FOR ADAPTIVE NONPLAYER CHARACTERS

Support characters support the storyline of games by offering quests, advice, goods for sale, or training. In *World of Warcraft*, for example, trainers include blacksmiths, herbalists, alchemists, and tailors. These characters tend to stand in one place, displaying only limited behavioral characteristics such as hand gestures or facial movements. More interesting blacksmiths or tailors, however, might perform tasks such as forging weapons or bleaching linen, with behavior dependent on the availability of materials in the environment and the experiences and motivation of the character.

A number of computational models of motivation exist for use in MRL and other agent-based systems. Huang and Weng [2002] and Saunders and Gero [2001] experimented with MRL models that incorporate a computational model of novelty with RL. Huang and Weng [2002] used their model in a robotics domain, while Saunders and Gero [2001] experimented with novelty in design agents. While novelty-based models were appropriate in those settings, in games, where human interaction with the game world introduces the possibility of random occurrences, novelty-based models become problematic. Random occurrences tend to be highly novel, but there is generally little for characters to learn from such situations.

Kaplan and Oudeyer [2003] developed several alternatives that overcame the problems associated with novelty-based models by using an approach designed to motivate a search for situations that show the greatest potential for learning. In one model, they defined such situations in terms of three motivational variables: predictability, familiarity, and stability of the sensory-motor context of a robot. While this technique overcomes the issues associated with novelty-based techniques, it is tailored specifically to the robotics domain where stability of sensors such as cameras is a key concern.

```
1.  Y(0) = ∅
2.  Repeat (forever):
3.     Sense S(t)
4.     if (Q(S(t), A) not initialized):
5.            initialize Q(S(t), A) arbitrarily ∀A
6.     Choose A(t) from S(t)
7.     Update Y(t)
8.     if (S(t-1) is initialised):
9.            Compute Rm(t)
10.           Update Q for S(t-1) and A(t-1)
11.    S(t-1) ← S(t); A(t-1) ← A(t)
12.    Execute A(t)
```

Fig. 2. Motivated reinforcement learning: Motivation is computed as a function of the experiences of states and actions.

Other models of motivation based on concepts such as curiosity and interest have more potential in game applications. Such models have been proposed by Schmidhuber [1991; 1997], Oudeyer et al. [2007], and Saunders [2001]. The idea of curiosity modifies novelty by motivating behaviors that achieve a moderate degree of novelty, which precludes highly novel, random occurrences. Merrick and Maher [2006; 2007] adapted a model of interest developed by Saunders [2001] for using in NPCs. Because it is desirable for game characters to be interesting to players, a model of interest as the motivator for characters is a useful starting point.

The model of Merrick and Maher [2006] is summarized in the next section, and an additional model of motivation as the search for competence is presented. Unlike novelty and interest, which are computed based on an agent's external experiences of its environment, competence is an introspective motivation. Competence motivation is computed based on the structures currently learned by an agent, which enables a character to adapt not only in response to external triggers, but in response to internal settings.

4.1 Modeling Motivation as Interesting Events

In the Merrick and Maher [2007] model of interest in Figure 3, changes in the environment are represented as events. Events represent potential learning tasks that, in turn, may lead to the emergence of new behaviors to perform those tasks. An event E is computed as the difference between, or change in, two successive sensed states as shown in line 2 of Figure 3. In their model, interest in a task is aroused when its novelty is at a moderate level, meaning that the most interesting tasks are those that are similar-yet-different to previously encountered tasks.

The key component of this model of motivation is a habituated self-organizing map (HSOM) [Marsland et al. 2000]. The HSOM identifies similar tasks by clustering events; the clustering error is used to trigger the computation of novelty; novelty is calculated using Stanley's [1976] model:

$$\tau \frac{dN_{(t)}}{dt} = \alpha [N_{(0)} - N_{(t)}] - \varsigma_{(t)}$$

τ is a constant governing the rate of habituation of interesting stimuli and α is a constant governing the rate of recovery.

The HSOM can be modified to accept variable-length strings from a CFG as input by initializing each SOM neuron as a zero-length vector. Each time a stimulus event is presented to the HSOM, each neuron is lengthened by adding randomly initialized variables with any labels L that occur in the event but not in the neuron. String-valued sensations are enumerated and events normalized before input to the HSOM. The novelty value output by the HSOM is modified using the Wundt curve to produce an interest value, which is used as the motivation signal $R_{m(t)}$.

The Wundt curve peaks for moderate novelty values, so the most interesting and thus most highly motivating events are those that are similar-yet-different to previously encountered experiences, as shown in Figure 4. Highly novel experiences such as random occurrences trigger low interest values, which can be thought of as caution in unfamiliar situations. Likewise, experiences that generate very low novelty values also trigger low interest; which can be thought of as boredom with overly familiar situations.

1. Subtract $S_{(t)} - S_{(t')}$ using

$$\Delta(s_{L(t)}, s_{L(t-1)}) = \begin{cases} s_{L(t)} \text{ if } \neg\exists\, s_{L(t')} \\ -s_{L(t')} \text{ if } \neg\exists\, s_{L(t)} \\ s_{L(t)} - s_{L(t')} \text{ if } s_{L(t)} - s_{L(t')} \neq 0 \\ \text{null otherwise} \end{cases}$$

2. Compose $E_{(t)} = (\Delta(s_{1(t)}, s_{1(t-1)}), \Delta(s_{2(t)}, s_{2(t-1)}), \ldots \Delta(s_{L(t)}, s_{L(t-1)}),...)$
3. Compute $N_{(t)} = \mathcal{N}(E_{(t)})$ using an HSOM and Stanley's model.
4. Compute $I(N_{(t)})$ using the Wundt curve:

$$I(N_{(t)}) = \frac{F^{+}_{max}}{1 + e^{-\rho^{+}(2N_{(t)} - F^{+}_{min})}} - \frac{F^{-}_{max}}{1 + e^{-\rho^{-}(2N_{(t)} - F^{-}_{min})}}$$

5. Output $R_{m(t)} = I(N_{(t)})$

Fig. 3. Algorithmic description of motivation to achieve interesting events [Merrick and Maher 2007].

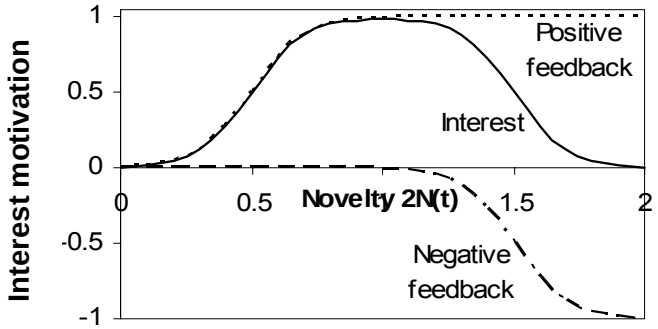


Fig. 4. Interest is computed as the joint action of positive and negative responses to novelty [Merrick and Maher 2006].

This simple, extrospective model of motivation has potential for learning simple tasks in complex, dynamic environments. It is possible that learning will display some sensitivity to changes in parameters of the motivation function. For example, higher values of τ may improve the complexity of behavioral cycles that can be learned. However, finding values for these parameters that achieve specific behavioral variety or complexity characteristics is likely to require careful tuning for any given environment. Other algorithms that modify the shape of the motivation curve while learning is in progress have the potential to avoid this problem. Such algorithms will use some form of introspection to allow the learning process to influence the motivation signal. Such a model is proposed in the next section.

4.2 Modeling Motivation Using Interest and Competence

The second algorithm for motivation for support characters is shown in **Fig. 5**. In this algorithm, the desires for interest and competence compete to motivate learning. The computational model of interest described above is used to focus attention

on new, interesting tasks. A computational model of competence is introduced that competes to maintain the current focus of attention for long enough to ensure that a stable behavior has emerged for completing that task. Competence is modeled by first computing the learning error of the current RL policy update from the Q-learning equation:

$$\Delta Q_{(t)} = |Q_{(t)}(S_{(t)}, A_{(t)}) - Q_{(t-1)}(S_{(t)}, A_{(t)})|$$

Events are again clustered and novelty computed using an HSOM. However, an additional error layer is attached to the HSOM with one neuron for each SOM neuron.

Each time an event is passed to the SOM, the error layer is updated to maintain the maximum error value $X_{(t)}$ experienced since the last time an event triggered the attached SOM neuron. Competence motivation is computed using the Wundt curve. Moderate

1. Subtract $S_{(t)}$ - $S_{(t')}$ using $\Delta(s_{L(t)}, s_{L(t-1)}) =$
$$\begin{cases} s_{L(t)} \text{ if } \neg\exists\, s_{L(t')} \\ -s_{L(t')} \text{ if } \neg\exists\, s_{L(t)} \\ s_{L(t)} - s_{L(t')} \text{ if } s_{L(t)} - s_{L(t')} \neq 0 \\ \text{null otherwise} \end{cases}$$
2. Compose $\mathbf{E}_{S(t)} = \{E_{(t)}\}$ where $E_{(t)} = (\Delta(s_{1(t)}, s_{1(t-1)}), \Delta(s_{2(t)}, s_{2(t-1)}), \ldots \Delta(s_{L(t)}, s_{L(t-1)}),)$
3. Compute $N_{(t)} = \mathcal{N}(E_{(t)})$ using an HSOM and Stanley's model.
4. Compute $I(N_{(t)})$ using the Wundt curve:
$$I(N_{(t)}) = \frac{F^{+}_{max}}{1 + e^{-\rho^{+}(2N_{(t)} - F^{+}_{min})}} - \frac{F^{-}_{max}}{1 + e^{-\rho^{-}(2N_{(t)} - F^{-}_{min})}}$$
5. Compute $\Delta Q_{(t)}$ and $X_{(t)}$
6. Compute $C(X_{(t)})$ using the Wundt curve:
$$C(X_{(t)}) = \frac{F^{+}_{max} - 0.33}{0.75 + e^{-\rho^{+}(2X_{(t)} - F^{+}_{min})}} - \frac{F^{-}_{max}}{1 + e^{-\rho^{-}(2X_{(t)} - F^{-}_{min})}}$$
7. Output $R_{m(t)} = \max(I, C)$

Fig. 5. Algorithmic description of motivation using interest and competence.

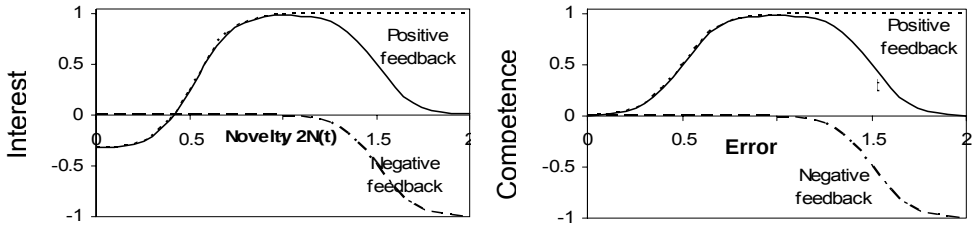


Fig. 6. (a) Modeling interest motivation with an aversion to low novelty (highly familiar tasks); (b) modeling competence motivation.

error is valued the most highly, so the most highly motivating events are those at which the agent currently has learned a little but not too much. This models the desire to approach optimal challenges that is thought to exist as a human motivation [Deci and Ryan 1985]. Tasks that are either too easy or too difficult trigger low motivation, while tasks that are moderately difficult based on prior learning are highly motivating.

Interest and competence compete to motivate learning, with the maximum value being used as the motivation signal. The intuition behind this model is that the ability to compute motivation with reference to learning error should encourage shifts in attention-focus to new task whenever a current, highly motivating task has been learned to a sufficient level of competence.

- In order to motivate the pursuit of competence more strongly than the pursuit of interest, the shape of the Wundt curve used to model interest is modified as shown in

(a) to model an aversion to highly familiar tasks.

5. METRICS FOR EVALUATING ADAPTIVE BEHAVIOR IN NON-PLAYER CHARACTERS

A number of different metrics have been developed for MRL algorithms. However, these techniques measure either the output or internal processing stages of the motivation function, and thus tend to be specific either to the domain or the motivation function being studied. In the developmental robotics domain, for example, Kaplan and Oudeyer [2003] aim to build robotic agents that are able to learn to maintain the stability of head and light source variables in a simple vision application. In order to measure the performance of their algorithm with respect to fulfilling this goal, Kaplan and Oudeyer [2003] use line charts to show the evolution of the three motivational variables: predictability, familiarity, and stability. The charts used by Kaplan and Oudeyer [2003] allow them to draw conclusions about the performance of their model for motivating vision-related joint manipulation. However, because these metrics are based on

properties such as head pan position and light position, they do not extend well to the evaluation of NPCs.

In a different approach to characterizing the internal performance of a motivation function, Huang and Weng [2002] use line charts for the evolution of Q-variables in table-based RL to characterize the distribution of the motivation signal. They use bar charts to characterize the agent's emergent behavior in terms of the frequency with which actions are performed. This approach allows Huang and Weng [2002] to conclude that their robot is displaying desirable emergent behavior and gives some indication of the efficiency with which that behavior is learned. However, as $|\mathbf{A}| \times |\mathbf{S}|$ charts are required to completely represent a learned policy, this technique for measuring performance does not scale well in complex environments with large state or action spaces.

In the design science domain, Saunders and Gero [2001] used bar charts to characterize the evolution of novelty within their design agents. The primary goal of the Saunders and Gero [2001] model of motivation is to maintain the agent's focus of attention on novel designs. As a result, charting the output of the motivation function and showing its continuing ability to identify novel design solutions is sufficient for measuring the performance of their model. However, characterizing the output of the motivation function is not a sufficient metric to quantify the adaptable behavior of the actual actions performed by NPCs.

In a different approach, Merrick and Maher [2006] used a scatter plot to visualize the emergent behavior of MRL agents controlling NPCs in MMORPGs. They identify different repeated patterns in the plot as representing different behaviors, and conclude that their agents can exhibit different behaviors for solving different tasks at different times. While this metric is more generic than previous approaches, being based on actions performed rather than internal variables specific to the motivation function used, it does not provide a quantitative evaluation of learning performance.

This article uses the idea of a behavioral cycle to represent a behavior by NPCs. In existing approaches, a behavioral cycle may be the result of animation or the firing of behavioral rules in reflexive agents. MRL extends such approaches to behavioral sequences that can adapt to different tasks at different times in response to changes in an NPC's environment. Thus, to develop learning performance metrics that evaluate adaptive, multitask learning, this section begins by defining mathematically what it means for a behavioral cycle to be learned for a task.

In order to recognize when a behavioral cycle is learned for a task, first, the standard deviation $\sigma_{E(t)}$ is computed for the number of actions used to solve a task for the event E during the last h times the task was completed:

$$\sigma_E = \sqrt{\frac{1}{h-1}\sum_{i=1}^{h}(a_i - \bar{a}_E)^2}$$

$\bar{A}_E$ is the mean number of actions required to repeat E during the last h successive completions of E.

- A task represented by the event E is learned when the standard deviation σ_E over the last h repetitions of E is less than some error threshold r for the first time.

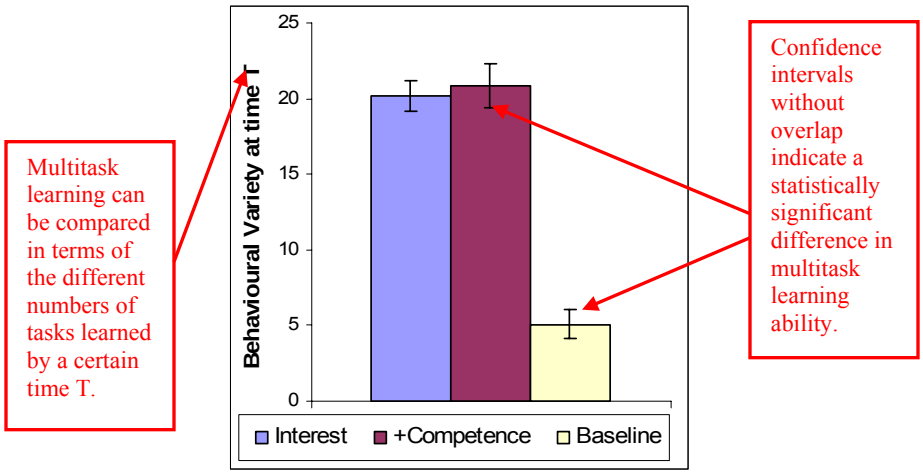


Fig. 7. Multitask learning can be visualized as instantaneous behavioral variety.

The intuition behind this approach is that a task is learned when there is a stable behavioral cycle incorporating that task. The use of h completions of E captures the cyclic nature of NPC behavior, while allowing for those cycles to change. The error term accommodates the random component of exploration strategies in RL. Using this model, charts can be produces that evaluate learning performance as information about σ_E. The following sections introduce two such metrics: behavioral variety and behavioral complexity.

5.1 Behavioral Variety

Behavioral variety evaluates learning performance by measuring the number of tasks learned. The behavioral variety V of the agent increases each time a new task is learned:

$$V_{(t+1)} = \begin{cases} V_{(t)} + 1 \text{ if } \sigma_{E(t)} < r \text{ for the first time} \\ V_{(t)} \text{ otherwise} \end{cases}$$

Instantaneous behavioral variety can be visualized using a bar chart to provide insight into the multitask learning performance of a NPC controlled by a MRL agent. This visualization characterizes multitask learning in terms of the number of tasks learned by a specified time. Because a single agent can be represented by a single series, multiple characters controlled by MRL agents using different motivation components can be compared by plotting a series for each character on the same chart. Fig. 7 shows an example of such a comparison for three characters using different motivation functions. Error bars show the 95% confidence interval.

Cumulative behavioral variety can be visualized using a line chart with a single series to provide insight into the adaptability or multitask learning performance of a NPC controlled by an MRL agent. This visualization characterizes multitask learning in terms of the gradient and maximum of the curve, which indicates, respectively, the rate at which new tasks are being learned and the number of tasks learned. Adaptability is

characterized by changes in the trend of the behavioral variety curve. Because a single agent can be represented by a single series, multiple characters controlled by MRL agents using different motivation or learning components can be compared by plotting a series for each character on the same chart. Fig. **8** shows an example of such a comparison for two characters using different motivation functions.

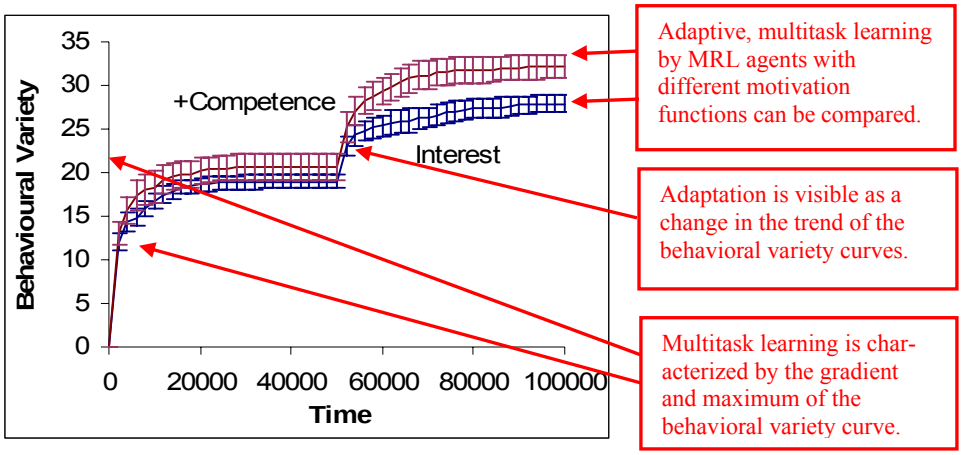


Fig. 8. Adaptive, multitask learning can be visualized as cumulative behavioral variety.

5.2 Behavioral Complexity

Behavioral complexity characterizes learning performance by measuring the complexity of a learned task in terms of the average length of the behavioral cycle required to repeat the task. More formally, when a task E has been learned according to the definition above, the complexity of the task can be measured as the mean numbers of actions $\bar{a}_E$ required to repeat E:

$$C_E = \bar{a}_E$$

Behavioral complexity can be visualized using a bar chart to provide insight into the multitask learning performance of a NPC controlled by a MRL agent. This visualization characterizes multi-task learning in terms of the complexity of tasks learned by a specified time. This could include the most complex task learned, the least complex task learned or the average complexity of learned tasks. Because a single agent can be represented by a single series, multiple characters controlled by MRL agents using different motivation or learning components can be compared by plotting a series for each agent on the same chart. Fig. **9** shows an example of such a comparison for three agents using different motivation functions.

5.3 Alternatives to Motivation

Alternatives to motivation for achieving adaptive behavior are possible. The simplest alternatives include greedy or random reward signals, which are computed based on an agent's experiences in its environment, but differ from motivation signals in that they are not necessarily inspired by psychological models of motivation. The disadvantages of non-motivated, experience-based reward signals for influencing behavior are that they tend to be either very general or highly contrived. General experience-based reward signals such as greedy or random rewards have no theoretical basis for application as

motivation, and thus less likely to be efficient as approaches to attention-focus. In contrast, over-constrained reward signals may perform well in certain environments but lack the capacity to display adaptive emergent behavior in others.

Despite their disadvantages, general experience-based reward signals that do not incorporate concepts of motivation can offer a useful baseline against which to evaluate

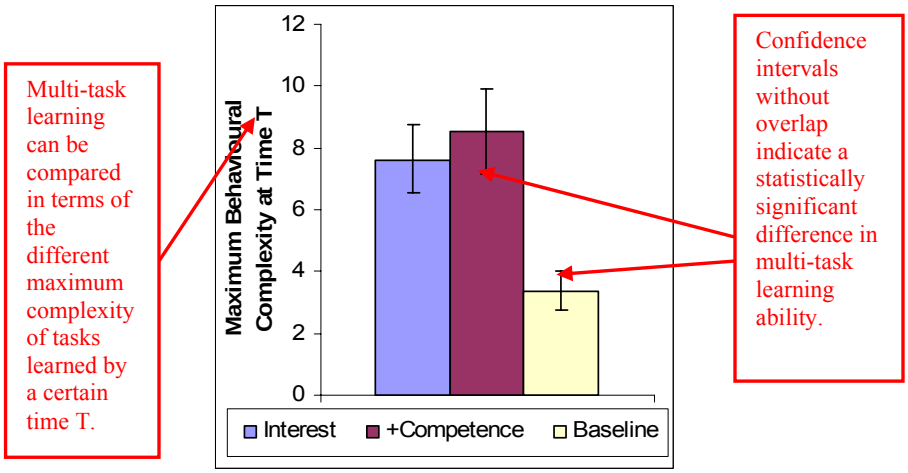


Fig. 9. Multitask learning can be visualized in terms of maximum behavioral complexity.

learned behavior. The use and development of complex computational models of motivation can only be justified if they significantly outperform simpler alternatives. As a baseline in the experiments in this article, a slowly changing, frequency-based reward signal is introduced. The idea of this reward signal is that, at each time step, there is some probability that the currently experienced task E will be chosen as a "goal task". This task will remain the goal task until a new one is chosen. The smaller the probability that the goal task will change, the longer the agent will have to learn about each. This reward function represents the narrowest possible focus of attention with respect to tasks. Agents using this reward function learn a single task for some period of time until a new task is chosen. Formally, this reward process remembers a single goal task E_g that has some probability p of changing to a current task $E_{(t)}$ at each time step:

$$E_g = \begin{cases} E_{(t)} \text{ with probability p} \\ E_{g(t-1)} \text{ otherwise} \end{cases}$$

The reward signal takes the following form:

$$R_{(t)} = \begin{cases} 1 \text{ if } E_{(t)} = E_g \\ 0 \text{ otherwise} \end{cases}$$

6. AN EMPIRICAL EVALUATION OF MOTIVATED REINFORCEMENT LEARNING FOR NONPLAYER CHARACTERS IN MMORPGS

Previous work has demonstrated MRL in controlling characters in games implemented in the *Second Life* virtual environment [Merrick and Maher 2006; Merrick and Maher 2007]. While these demonstrations provide a qualitative analysis of MRL for character control, they do not quantify the behavioral characteristics of NPCs controlled by MRL

agents. In order to achieve this, a controlled test environment is required. This article describes two experiments using a Java program to simulate the MMORPG game scenario described by Merrick and Maher [2006]. The environment used in these experiments is relatively simple, with 52 states and 24 actions available across different states. In the first experiment, the only changes to occur in the environment while learning is in progress are those triggered by the MRL-controlled NPC. The second experiment simulates an unpredictable change in a dynamic environment.

As it is possible to design rule-based reward signals for both of these environments, MRL would generally not be required for character control in this scenario. However, testing NPCs using MRL in simple, closed environments under controlled conditions permits the production of repeatable data as input for an empirical analysis.

6.1 Experiment 1

The environment used in this experiment is a simple, simulated role-playing game scenario, modeled on the village shown in
Fig. 10. The village can be modeled as a set of two MDPs, P_1 and P_2, describing two regions of the village, one containing the objects required to mine iron-ore and forge weapons and another containing the objects required to cut timber and craft furniture. These regions can be described by the CFG in Figure 11. This grammar describes the two regions P_1 and P_2 in terms of the locations, inventory, and visible objects they support. Locations have enumerated values, while visible objects and inventory are valued according to the number of units currently visible or held in inventory. A NPC's inventory describes the objects it is currently carrying.

Fig. 10. A game environment in *Second Life*.

S	→	<sensations>
<sensations>	→	<P_1Sensations><P_2Sensations>

```
<P1Sensations> → <P1location><P1inventory><P1visibleObjects>
<P1location> → <mine> | <smithy>
<mine> → 1
<smithy> → 2
<P1inventory> → <P1objects>
<P1visibleObjects> → <P1objects>
<P1objects> → <P1object><P1objects> | ε
<P1object> → <pick> | <forge> | <smelt> | <iron-ore> |
<iron> | <weapons>
<P2location> → <forest> | <carpenter-shop>
<forest> → 3
<carpenter-shop> → 4
<P2inventory> → <P2objects>
<P2visibleObjects> → <P2objects>
<P2objects> → <P2object><P2objects> | ε
<P2object> → <axe> | <lathe> | <timber> |<furniture>
<pick> → 1
<forge> → 1
<smelt> → 1
<iron-ore> → 1
<iron> → 1
<weapons> → 1
<axe> → 1
<lathe> → 1
<timber> → 1
<furniture> → 1
```

Fig. 11. A context-free grammar for sensed states in a game scenario in which agents control non-player characters. Agents have location sensors, inventory sensors, and object sensors.

Each object is visible at one location. The maximum number of any object that can be visible or held in inventory is one. Some example sensed-states in the environment in this experiment are shown in label-sensation (L:s) format in eq. (1). Labels L are constructed from the set of grammar variables, while values for sensations come from the set of terminals.

```
S(1) ((location:2)(visibleObjectPick:1)( visibleObjectForge:1))
S(2) ((location:2)(inventoryPick:1)( visibleObjectForge:1))
S(3) ((location:4)(inventoryPick:1)( visibleObjectAxe:1)(          (1)
       visibleObjectLathe:1))
```

In order for NPCs to interact with their environment, the actions defined by the CFG in

Fig. 12 are available:

```
A → <actions>
<actions> → <P1Actions><P2Actions>
<P1Actions> → pick-up <P1object> | move <direction> | use
<P1object> <P2Actions> → pick-up <P2object> | move <direction> | use
<P2object>
<direction> → north | south | east | west
<P1object> → pick | forge | smelt | iron | iron-ore | weapons
<P2object> → axe | lathe | timber | furniture
```

Fig. 12. A context-free grammar for the action set in a game environment in which agents have location effectors, pick-up object effectors, and use object effectors.

Possible tasks in this environment include traveling from place to place, cutting timber, mining iron-ore, smelting the ore to obtain iron, forging weapons, and crafting furniture. Not all actions are available in all world states. `Use` actions are only available for a particular object if that object is in the current `<inventory>` or `<objects>` list of the NPC, as appropriate for that object. For example, it is appropriate to `pick-up` an `axe` and have it in inventory for later use, but the `forge`, which is too heavy to be picked up, can be used when it is visible. `Move` actions are available in any world state. A `use` action produces the desired result, such as using the pick to mine iron, 90% of the time and no result 10% of the time. `Pick-up` actions are only available for a particular object if that object is in the current `<visibleObjects>` list of the NPC, that is, if the object and the NPC are at the same location. The actions that would produce the state sequence in eq. (1) are `A(pick-up, pick)` and `A(move, east)`. The events produced by these actions are shown in eq. (2). Each of these events represents a potentially motivating achievement task.

$$E_{(1)}\ \texttt{((inventoryPick:1)(seePick:-1))}$$
$$E_{(2)}\ \texttt{((location:2)(seeForge:-1)(seeAxe:1)(seeLathe:1))} \quad (2)$$

This experiment compares NPCs in terms of their emergent behavioral variety and behavioral complexity. Characters controlled by MRL agents using three different types of motivation are compared:

- an MRL agent using the baseline reward function;
- an MRL agent motivated to achieve interesting events; and
- an MRL agent motivated by interest and competence.

Each of these agents was run 20 times for 50,000 time-steps in the environment described above. Results show the 95% confidence interval. The interest, competence, RL and metric parameters used in the experiments in this article are summarized in Table I.

Results

Figure 13 characterizes the multitask learning ability of the three NPCs controlled by different types of MRL agents in terms of the behavioral variety achieved over a period of 50,000 time-steps. In the experimental game environment, NPCs motivated to achieve interesting events and NPCs motivated by interest and competence learn between 17 and 23 different behavioral cycles during the first 50,000 time-steps of their lifetimes. This is a key result, as it shows that NPCs capable of multiple tasks can emerge using MRL agents motivated by task-independent, experience-based motivation signals.

Table I. Parameters and their Values

	Parameter	**Value**
Reinforcement Learning Parameters	γ	0.9
	β	0.9
	ε	0.1
	Φ_1 and Φ_3	0.8
	Φ_2	1000
	η	0.1
	α	1.05
	τ_1	3.3
	τ_2	14.3

Interest Parameters	F^{+}_{max} and F^{-}_{max}	1
	ρ^{+} and ρ^{-}	10
	F^{+}_{min}	0.5
	F^{-}_{min}	1.5
Competence Parameters	F^{+}_{max} and F^{-}_{max}	1
	ρ^{+} and ρ^{-}	10
	F^{+}_{min}	0.5
	F^{-}_{min}	1.5
Metric and Baseline Parameters	r	2
	h	20
	p	0.001

NPCs motivated to achieve interesting events and NPCs motivated by interest and competency achieve the highest behavioral variety in the test environment. Of the 53 possible achievement tasks, these NPCs focus attention on an average of approximately 20 tasks before the rate of increase of behavioral variety drops to zero. This figure characterizes the multitask learning ability of MRL agents using the two motivation functions as significantly higher than that of agents using the baseline reward function. MRL agents using the baseline reward function learn approximately five tasks before the rate of increase of behavioral variety drops to zero.

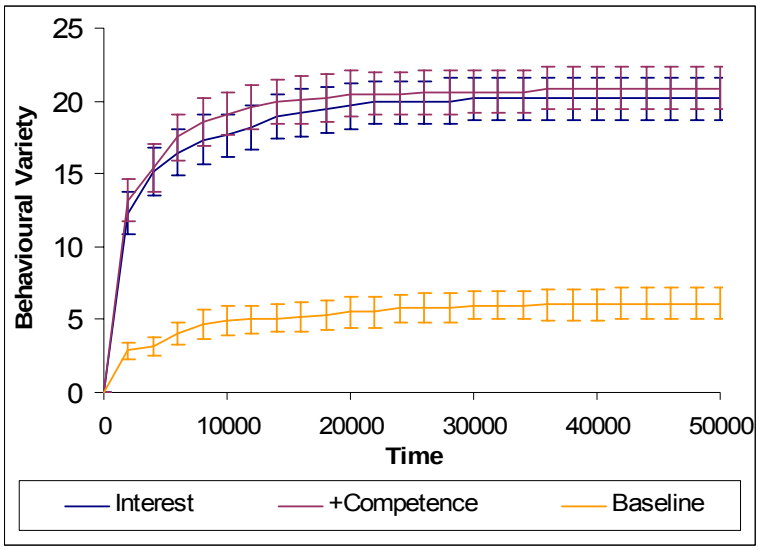


Fig. 13. Cumulative behavioral variety by nonplayer characters controlled by motivated reinforcement learning agents using different motivation functions.

The fact that the behavioral variety attained by the NPCs is significantly lower than the number of potential learning tasks in the environment is another key result. In

complex environments it is generally infeasible for NPCs to learn every task. Rather, each character is defined by a subset of learning tasks on which they focus attention. The higher behavioral variety displayed by NPCs using the two motivation functions can be explained by the task selection function they use. In the two motivation functions, task selection is modeled using a SOM that allows past experiences to influence the selection of current tasks. In contrast, the baseline reward function does not model past experiences, other than the current goal task. The baseline function thus has the highest probability of focusing learning on the most frequently experienced tasks, resulting in potential repetition of reward for similar tasks and lower overall behavioral variety.

Figure 14 characterizes the multitask learning ability of NPCs controlled by the three types of MRL agent in terms of their maximum behavioral complexity. NPCs using the two motivation functions achieve significantly higher maximum behavioral complexity than agents using the baseline reward function. This is due to the nature of the reward function, which has the highest probability of rewarding frequently occurring tasks. Frequently occurring tasks require fewer actions to repeat and thus influence the emergence of behavioral cycles of fewer actions. These behavioral complexity results again characterize a difference in the multitask learning ability of different approaches.

In the test domain in this experiment, the most complex single task is making weapons, which requires at least a five-action behavioral cycle. This is shown as a grey line in

Fig.14. NPCs using the motivation functions achieve behavioral complexity significantly greater than five during the first 50,000 time-steps of their lives. This indicates that they have the capacity to solve the most complex task in this test environment, but also that they can interleave the solutions to multiple tasks in a single behavioral cycle. This is because both motivation functions can provide rewards for more than one task during the same time period. In contrast, the baseline function remembers and rewards only one goal task during any given time period.

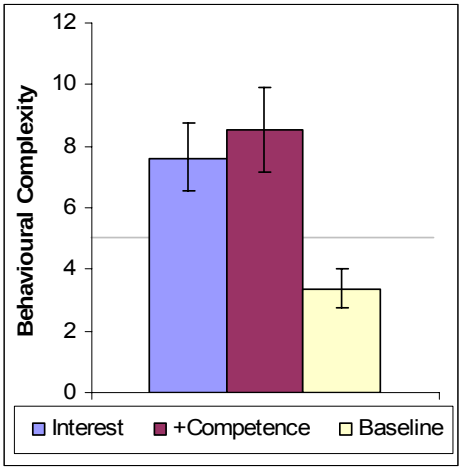


Fig.14. Maximum behavioral complexity achieved by NPCs using different models of motivation after 50,000 time-steps.

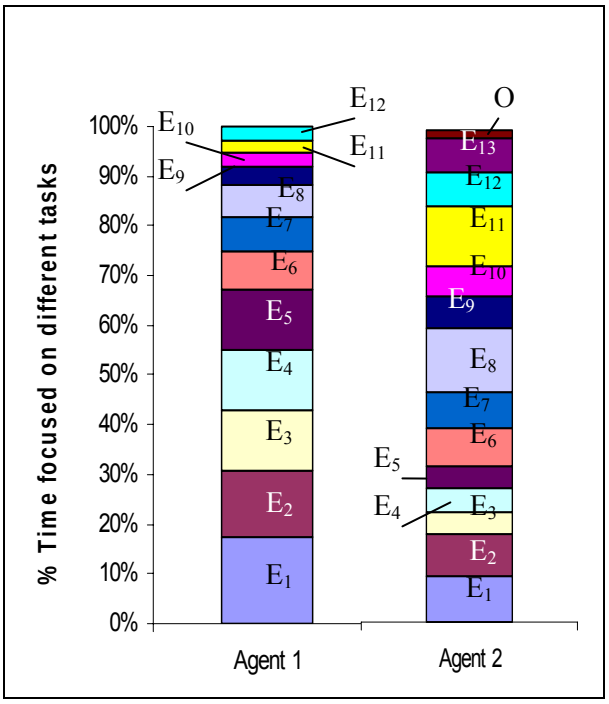
As a case study to tie the empirical results above back to individual characters, Figure 15 shows the proportion of time devoted to different tasks in the game environment by

two individual NPCs motivated to achieve interesting events. The first NPC focuses its attention primarily on tasks concerned with forging weapons (E_1-E_5): mining iron-ore, smelting iron, traveling between the mine and the smithy, and forging weapons. The second focuses primarily on tasks concerned with crafting furniture (E_8, E_{11}-E_{13}): cutting timber, traveling between the forest and the carpenter, and using the lathe. Thus, the first NPC has developed as a blacksmith character, while the second has developed as a carpenter character. The ability of different NPCs to focus attention on different subsets of tasks based on their experiences in their environment is a key emergent property of MRL agents. In applications such as MMORPGs, a number of NPCs using identical agent models can learn different behavioral cycles, representing different characters. This removes the need for different rule sets or internal states to be hand-crafted for different characters, as is the case using the traditional reflex agent approach.

An example of a behavioral cycle learned by each NPC is shown in Figure 16. While this is the primary behavioral cycle in each case, each character may also have learned other behavioral cycles for other tasks at different times.

6.2 Experiment 2

This experiment is conducted in a simulated game environment that is initially identical to the one described for Experiment 1. However, after 50,000 time-steps, the environment is changed such that the initial MDPs P_1 and P_2 are replaced by two new MDPs, P_3 and P_4, modeled on the environment in Figure 17. P_1 and P_2 described states and actions that allowed activities such as traveling, mining iron-ore, forging weapons, cutting timber, and crafting furniture. At t=50,000, monsters spawn. The monsters damage the forge and the lathe so that the actions for using the forge or lathe no longer produce weapons or furniture. Instead, when pick or axe are used in the presence of a monster, a dead monster results. Dead monsters disappear and new monsters spawn when the agent moves away from the dead monster's location. The state and action sets for P_3 and P_4 are shown in Figure 18.



E_1((location:-1.0)(visibleWeapons:-1.0)(visibleSmelt:-1.0)(visibleForge:-1.0))

E_2((location:1.0)(visibleWeapons:1.0)(visibleSmelt:1.0)(visibleForge:1.0))

E_3((inventoryIronOre:1.0))

E_4((inventoryIron:1.0)(inventoryIronOre:-1.0))

E_5((inventoryIron:-1.0))

E_6((location:-2.0)(visibleLathe:-1.0)(visibleWeapons:1.0)(visibleFurniture:-1.0)(visibleSmelt:1.0)(visibleForge:1.0))

E_7((location:2.0))

E_8((location:1.0)(visibleLathe:1.0)(visibleFurniture:1.0))

E_9((location:2.0)(visibleLathe:1.0)(visibleWeapons:-1.0)(visibleFurniture:1.0)(visibleSmelt:-1.0)(visibleForge:-1.0))

E_{10}((location:-2.0))

E_{11}((location:-1.0)(visibleLathe:-1.0)(visibleFurniture:-1.0))

E_{12}((inventoryLog:1.0))

E_{13}((inventoryLog:-1.0))

O = other

Fig. 15. Focus of attention by two individual motivated reinforcement learning agents to achieve interesting events over 50,000 time-steps. Agents that focus attention differently represent different game characters.

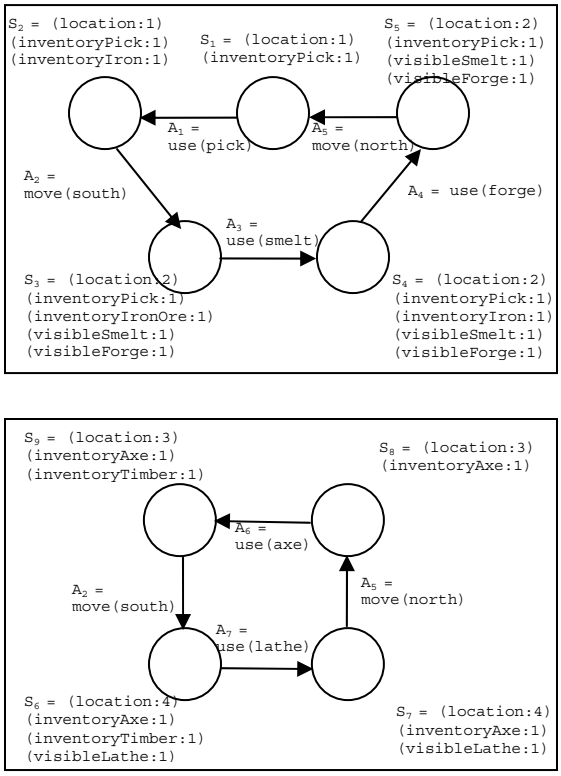


Fig. 16. Behavioral cycles learned by two motivated reinforcement learning agents: (a) agent 1 learned a behavioral cycle for mining iron and making weapons (represents a blacksmith character); (b) agent 2 learned a behavioral cycle chopping wood and making furniture (represents a carpenter character).

Fig. 17. An unpredictable change is simulated as a monster that damages the forge and lathe.

```
S                          →      <sensations>
<sensations>               →      <P4Sensations><P5Sensations>
<P4Sensations>  →          <P4location><P4inventory><P4visibleObjects>
<P4location>               →      <mine> | <smithy>
<mine>                     →      1
<smithy>                   →      2
<P4inventory>   →          <P4objects>
<P4visibleObjects>         →      <P4objects>
<P4objects>                →      <P4object><P4objects> | ε
<P4object>                 →      <pick> | <forge> | <smelt> | <iron-ore> |
<iron>                            | <weapons> | <monster> | <dead monster>
<P5location>               →      <forest> | <carpenter-shop>
<forest>                   →      3
<carpenter-shop>           →      4
<P5inventory>   →          <P5objects>
<P5visibleObjects>         →      <P5objects>
<P5objects>                →      <P5object><P5objects> | ε
<P5object>                 →      <axe> | <lathe> | <timber> | <furniture> |
                                  <monster> | <dead monster>
<pick>                     →      1
<forge>         →          1
<smelt>         →          1
<iron-ore>                 →      1
<iron>                     →      1
<weapons>                  →      1
<axe>                      →      1
<lathe>         →          1
<timber>                   →      1
<furniture>                →      1
<monster>                  →      1
<dead monster>  →          1
```

```
A                  →      <actions>
<actions>          →      <P4Actions><P5Actions>
<P4Actions>        →      pick-up <P4object> | move <direction> | use
<P4object> <P5Actions>  →      pick-up <P5object> | move <direction> | use
<P5object>
<direction>        →      north | south | east | west
<P4object>         →      pick | forge | smelt | iron | iron-ore | weapons |
                          monster | dead monster
<P5object>         →      axe | lathe | timber | furniture | monster | dead
                          monster
```

Fig.18. State and action spaces of the environment in Experiment 4 after t=50,000.

The size of the state space decreases due to the destruction of the forge and lathe, although the exact set of states in the new state space will depend on the state at t=50,000 when the environment changes. The size of the action space increases with the addition of the monster. The number of tasks in the environment after the change decreases to 12 tasks including 4 from the original MDPs for traveling between the forest and the mine and between the smithy and the carpenter's shop. NPCs that exhibit adaptable behavior in this environment should, after the change, display an increase in behavioral variety that is significantly greater than four tasks.

This experiment compares the performance of NPCs in terms of their emergent behavioral variety in an environment that changes while learning is in progress, as

described above. Characters controlled MRL agents using three different types of motivation are compared:

- an MRL agent using the baseline reward function;
- an MRL agent motivated to achieve interesting events; and
- an MRL agent motivated by interest and competence

Each of these agents was run 20 times for 100,000 time-steps, with the environment changing at t=50,000 time-steps; results show the 95% confidence interval.

Results

Figure 19 illustrates the effect of changes in the environment on the behavioral cycles learned by NPCs using different motivation functions. Hypothesis bars indicate the minimum increase in behavioral variety (4 tasks) required to conclude that a particular agent is adaptable. During the period t=20,000 to t=50,000 the gradient of the behavioral variety curves does not alter for any of the agent types. In the case of the NPCs motivated to achieve interesting events or motivated by interest and competency, the gradients of these curves is close to zero. This suggests that no new behaviors are being learned in that time period. After t=50,000 when the environment changes with the appearance of monsters, the cumulative behavioral variety of these agents increases significantly by between 7 and 12 behaviors. This indicates that learning new behavioral cycles is occurring for tasks such as running towards a monster, killing a monster, or running away from a monster.

After the monsters appear at t=50,000, the increase in behavioral variety by NPCs motivated by interest and competency is significantly higher than that by NPCs motivated by interest alone or using the reward function. The motive to develop new competencies allows NPCs to focus attention on and learn behavioral cycles for new tasks more quickly than the interest motive alone. This suggests that, in simple, dynamic environments, NPCs controlled by MRL agents motivated by interest and competence are more adaptable than those motivated by interest alone. This is due to their preference for pursuing tasks of low competency before tasks of low interest.

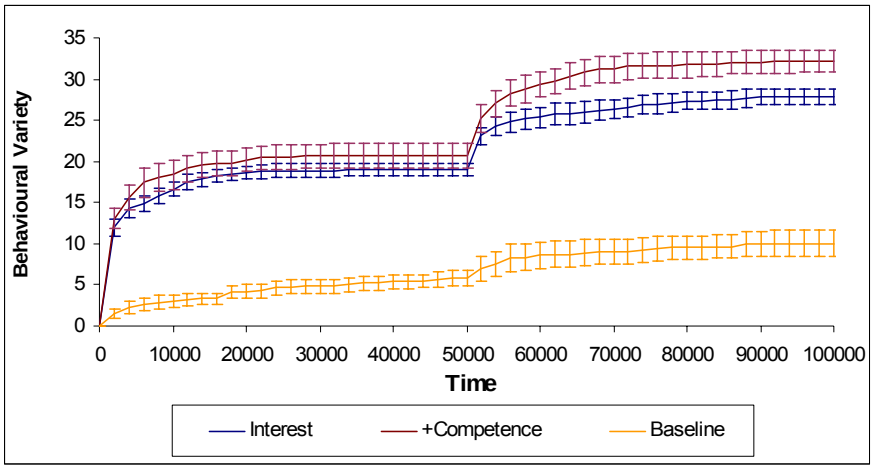
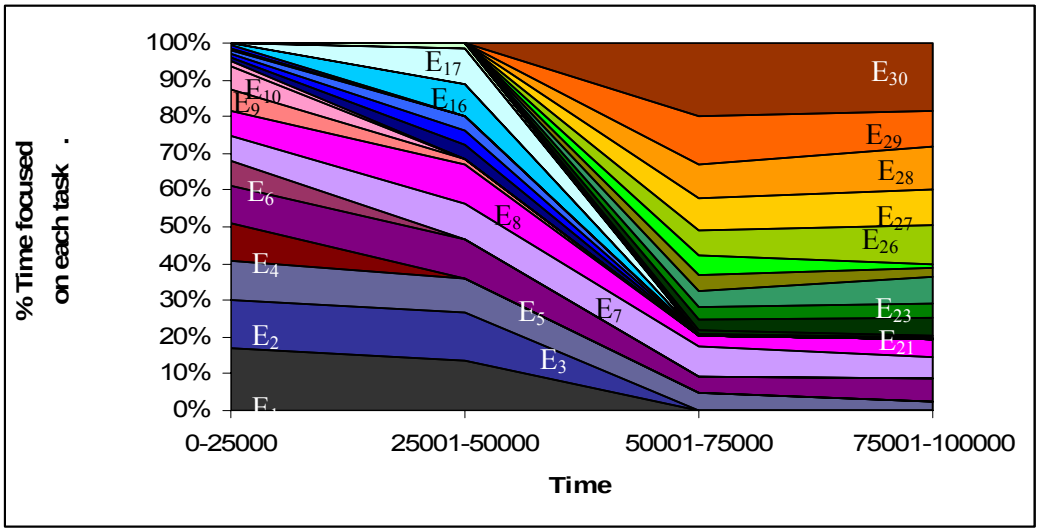


Fig. 19. Cumulative behavioral variety by nonplayer characters controlled by motivated reinforcement learning agents using different motivation functions.

NPCs using the baseline reward function continue to increase their behavioral variety after t=50,000, but this increase is only for approximately two tasks. Continued overlap of the confidence intervals with the hypothesis bar indicates that it is unclear if these are new tasks or tasks remaining from the initial environment. The curve for NPCs using the baseline reward function still has an increasing trend at t=100,000, so significant adaptation may occur at some time in the future. However, it can be concluded that these characters are slower to adapt than those using the motivation functions.



E_1((location:1.0)(visibleLathe:1.0)(visibleFurniture:1.0))

E_2((location:-1.0)(visibleLathe:-1.0)(visibleFurniture:-1.0))

E_3((location:2.0))

E_4((location:-1.0)(visibleWeapons:-1.0)(visibleSmelt:-1.0)(visibleForge:-1.0))

E_5((location:-2.0)(visibleLathe:-1.0)(visibleWeapons:1.0)(visibleFurniture:-1.0)(visibleSmelt:1.0)(visibleForge:1.0))

E_6((location:1.0)(visibleWeapons:1.0)(visibleSmelt:1.0)(visibleForge:1.0))

E_7((location:-2.0))

E_8((location:2.0)(visibleLathe:1.0)(visibleWeapons:-1.0)(visibleFurniture:1.0)(visibleSmelt:-1.0)(visibleForge:-1.0))

E_9((inventoryLog:1.0))

E_{10}((inventoryLog:-1.0))

E_{11}((inventoryIronOre:1.0))

E_{12}((inventoryIron:1.0)(inventoryIronOre:-1.0))

E_{13}((inventoryIron:-1.0))

E_{14}((location:2.0)(visibleLathe:1.0)(visibleWeapons:-1.0)(visibleSmelt:-1.0)(visibleForge:-1.0))

E_{15}((location:-2.0)(visibleLathe:-1.0)(visibleWeapons:1.0)(visibleSmelt:1.0)(visibleForge:1.0))

E_{16}((location:1.0)(visibleAxe:1.0)(visibleLathe:1.0))

E_{17}((location:-1.0)(visibleAxe:-1.0)(visibleLathe:-1.0))

E_{18}((inventoryLog:-1.0)(visibleFurniture:1.0))

E_{19}((visibleMonster:-1.0)(location:-1.0)(visibleLathe:-1.0)(visibleFurniture:-1.0))

E_{20}((visibleMonster:-1.0)(location:-1.0)(visibleWeapons:-1.0)(visibleSmelt:-1.0)(visibleForge:-1.0))

E_{21}((visibleMonster:-1.0)(location:-1.0)(visibleDeadMonster:-1.0)(visibleWeapons:-1.0)(visibleSmelt:-1.0)(visibleForge:-1.0))

E_{22}((visibleMonster:-1.0))
E_{23}((visibleMonster:1.0)(location:-2.0)(visibleLathe:-1.0)(visibleWeapons:1.0)(visibleFurniture:-1.0)(visibleSmelt:1.0)(visibleForge:1.0))

E_{24}((location:-1.0)(visibleLathe:-1.0)(visibleDeadMonster:-1.0)(visibleFurniture:-1.0))

E_{25}((location:-1.0)(visibleDeadMonster:-1.0)(visibleWeapons:-1.0)(visibleSmelt:-1.0)(visibleForge:-1.0))

E_{26}((visibleMonster:-1.0)(location:-1.0)(visibleLathe:-1.0)(visibleDeadMonster:-1.0)(visibleFurniture:-1.0))

E_{27}((visibleMonster:1.0)(location:1.0)(visibleLathe:1.0)(visibleFurniture:1.0))
E_{28}((visibleMonster:1.0)(location:2.0)(visibleLathe:1.0)(visibleWeapons:-1.0)(visibleFurniture:1.0)(visibleSmelt:-1.0)(visibleForge:-1.0))

E_{29}((visibleMonster:1.0)(location:1.0)(visibleWeapons:1.0)(visibleSmelt:1.0)(visibleForge:1.0))

E_{30}((visibleMonster:-1.0)(visibleDeadMonster:1.0))

Fig. 20. Change in attention focus over time exhibited by a single nonplayer character motivated by interest and competence in a dynamic environment.

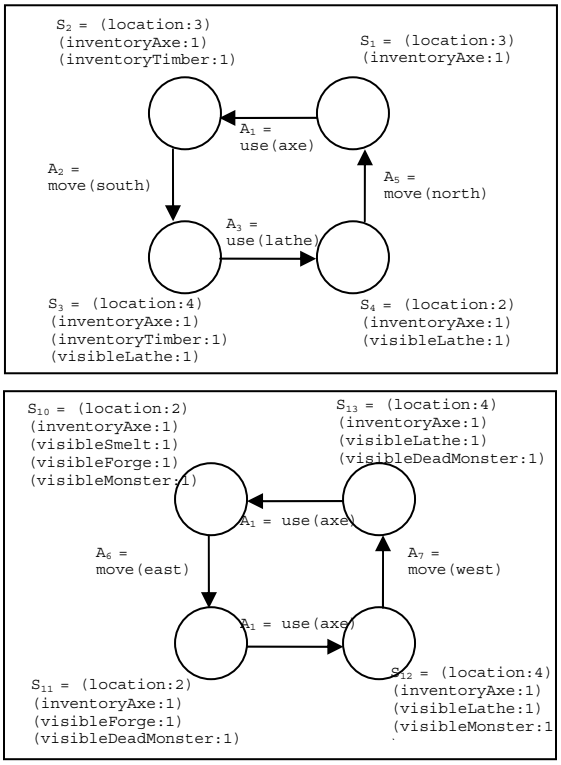


Fig. 21. Behavioral cycles before and after the monster appears: (a) behavioral cycle for cutting timber and making furniture; (b) behavioral cycle for killing monsters.

Experiment 1 includes a case study showing the focus of attention of two individual NPCs over 50,000 time-steps. Figure 15 shows how different NPCs are motivated to focus their learning on different tasks in response to their different experiences in their

environment. A second case study, Figure 20 traces the change in attention-focus by one NPC motivated by interest and competence in response to its experience in a changing environment. The lifetime of this NPC is broken into four phases: t=0-25,000, t=25,001-50,000, t=50,001-75,000, and t=75,001-100,000, and the progressive changes in attention focus is charted between these phases.

Fig. **20** shows that, at the end of the first phase, the primary focus of attention is on tasks represented by events E_1-E_{10} which are traveling, cutting timber, and making furniture. By the end of the second phase, the NPC's focus of attention has shifted slightly away from tasks E_4, E_6, E_9, and E_{10} towards tasks E_{13}-E_{17}. In the third phase, during which the environment changes, another shift in attention-focus occurs. By the end of the third phase the NPC's attention is focused primarily on tasks E_{21}-E_{30}. Only four tasks, E_3, E_5, E_7, and E_8, remain from the initial phase; this process represents a change in character from a lumberjack to a monster-killer in response to change in the environment. The focus of attention remains stable throughout the fourth phase. Examples of behavioral cycles learned before and after the appearance of the monster are shown in Figure 21.

7. IMPLICATIONS AND FURTHER DEVELOPMENT

The previous section evaluated the performance of NPCs using different MRL models. This section considers the wider implications of MRL in relation to how it may be used and developed further in future.

7.1 Integrating MRL into Existing Games

While MRL enables new kinds of characters that can adapt and evolve, the inclusion of characters controlled by MRL agents in existing game genres such as MMORPGs does raise a number of issues. MRL agents have higher processing and memory requirements than simple reflex agents, and, as a result, more resources are required to support similar quantities of characters controlled by MRL agents. One approach to this issue is to use a combination of MRL agents for some characters and reflex agents for others.

The introduction of adaptive, MRL agents into existing game genres also raises issues concerning game-play. Players must be able to find key characters such as quest-givers or merchants. This becomes more difficult if characters can move around and explore their environments. Solutions include limiting characters to certain areas or marking their location on a mini-map.

While current models of motivation are useful in support characters, they become problematic in characters such as enemies that can die. Learning by trial and error in dangerous situations can prove terminal. However, different motivation functions may be possible that can allow MRL agents to exhibit different behavioral characteristics; this and other avenues for future work are discussed in the following sections.

7.2 Alternative Approaches to Modeling Motivation

This article discusses alternatives to computational models of motivation for RL and evaluates two such approaches. A full sensitivity analysis of these approaches was not presented, but would be useful to better understand the performance of MRL techniques. The results so far show clearly that MRL is sensitive to the motivation function used; but space permits an exploration of only a selection of alternative motivation functions.

In addition to exploring the parameters of the models presented in this article, there also remains the question of when alternative models of motivation become appropriate. In natural systems such as humans and animals, there are three broad classes of motivation: biological, cognitive, and social models. Models in different categories have applications in different types of characters: for example, biological models may be

appropriate for modeling animal behavior. The introduction of pseudo-physiological variables for virtual animals has the potential to introduce a self-awareness that supports self-preservation in dangerous situations. This extends current interest and competence based models that focus on exploration and discovery only.

Alternative cognitive models of motivation, such as achievement, also have potential application in MRL to focus attention on different types of tasks; where interest-based models focus tasks of moderate novelty, achievement motivation focuses on tasks of moderate, high, or low difficulty, depending on the tendency to either seek success or to avoid failure. Concepts such as achievement motivation have the potential to contribute to a sense of personality in artificial agents and to the development of characters capable of identifying critical tasks.

Social motivation theories are also an important future research direction for MRL in multi-agent settings such as games. Evolutionary theories, in particular, represent an important component of motivation models, especially for enemy characters designed to function in dangerous environments. Evolutionary approaches such as genetic algorithms allow adaptation over generations of individuals, so that the failure or destruction of a single individual can be tolerated and even used as a trigger for learning within the society as a whole.

7.3 Scalability of Motivated Reinforcement Learning

The experiments in this article consider MRL in small-scale dynamic environments. While this level of complexity is sufficient to describe some MMORPG scenarios, as games continue to increase in complexity, more sophisticated behaviors will be required. For example, this article considers a village modeled by two MDPs, but in many MMORPGs, NPCs may interact in villages or other scenes modeled by many MDPs. A further suite of experiments is required to understand the impact of such complexity on character behavior.

Likewise, this article considers behaviors comprising one to ten actions. However, more complex tasks are also conceivable. Again, a further suite of experiments is required to gradually introduce tasks of increasing complexity in order to understand the impact on character behavior. As the complexity of the environment is increased, it may become necessary to combine motivation with other types of RL such as function approximation or hierarchical RL in order to achieve coherent behavior.

7.4 Adaptive Virtual Worlds

Finally, this article applied MRL techniques to the NPCs in computer games. However, unlike physical environments, virtual environments have the capacity for the environment itself to adapt and respond to human interactions with similar reasoning processes [Maher and Gero 2002; Maher and Merrick 2005]. The use of MRL agents to control not only the characters in the games but the buildings, weapons, furniture, and landscape has the potential to transform virtual game worlds into adaptive virtual spaces that can evolve and change over time. A weapon might develop new fighting skills, a room might learn how to trap intruders, trees might learn to repel lumberjacks. The use of MRL techniques has the potential to provide a new type of dynamic game environment that extends the life of a game by autonomously adapting new behaviors and challenges.

8. CONCLUSION

This article considers the role of motivation in creating adaptive NPCs using MRL. Two computational models of motivation are presented for the control of NPCs in MMORPGs. These models represent motivation both as an ongoing search for novelty

and interest and as the search for competence. In order to quantify the effect of different models of motivation on the behavioral characteristics of NPCs, two metrics are introduced to evaluate the adaptability of NPCs using different models of motivation. These metrics quantify adaptive, multitask learning in terms of the behavioral variety and complexity of NPCs. An empirical evaluation of NPCs controlled by MRL agents in simulated game scenarios show that MRL agents motivated by the search for competence produce NPCs that are more adaptable in dynamic environments than those motivated by interest and novelty alone.

MRL has the potential to allow new kinds of characters that adapt and evolve in time with their environments. This, in turn, permits the design of new game genres that include the open-ended modeling capabilities of emerging virtual world platforms such as *Second Life*. Using MRL to control characters means that players can not only modify the game world while the game is in progress, but characters can also respond to those modifications by adapting and learning. Furthermore, because the MRL model is not tied specifically to characters, there is potential for a new type of dynamic environment in which buildings, weapons, furniture, and landscapes can also adapt and change, creating exciting, evolving game worlds.

REFERENCES

BAILLIE-DE BYL, P. 2004. *Programming Believable Characters for Computer Games.* Charles River Media.

DECI, E. AND RYAN, R. 1985. *Intrinsic Motivation and Self-Determination in Human Behavior*. Plenum Press, New York.

GRAEPEL, T., HERBRICH, R. AND GOLD, J. 2004. Learning to fight. In *Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education.*

HUANG, X. AND WENG, J. 2002. Novelty and reinforcement learning in the value system of developmental robots. In *Proceedings of the Second International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems* (Edinburgh, Scotland), C.G. Prince et al. (eds), 47-55.

KAPLAN, F. AND OUDEYER, P.-Y. 2003. Motivational principles for visual know-how development. In *Proceedings of the 3rd International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems* (Lund University Cognitive Studies), C.G. Prince et al. (eds.), 73-80.

LAIRD, J. AND VAN LENT, M. 2000. Interactive computer games: Human-level AI's killer application. In *Proceedings of the National Conference on Artificial Intelligence* (AAAI), 1171-1178.

MAHER, M.-L. AND GERO, J.S. 2002. Agent models of 3D virtual worlds. ACADIA 2002: Thresholds. California State Polytechnic University, Pamona, 127-138.

MAHER, M.-L. AND MERRICK, K. 2005. Agent models for dynamic 3D virtual worlds. In *Proceedings of the 2005 International Conference on Cyberworld*s (Singapore), 27-34.

MARSLAND, S., NEHMZOW, U., AND SHAPIRO, J. 2000. A real-time novelty detector for a mobile robot. In EUREL European Advanced Robotics Systems Masterclass and Conference.

MERCERON, A. 2001. *Languages and Logic*. Pearson Education, Australia.

MERRICK, K. AND MAHER, M.-L. 2006. Motivated reinforcement learning for non-player characters in persistent computer game worlds. In *Proceedings of the International Conference on Advances in Computer Entertainment Technology* (electronic proceedings).

MERRICK, K. AND MAHER, M.-L. 2007. Motivated reinforcement learning for adaptive characters in open-ended virtual worlds. In *Proceedings of the International Conference on Advances in Computer Entertainment Technology* (Salzburg, Austria), 127-134.

MOOK, D.G. 1987. *Motivation: The Organization of Action.* W. W. Norton, New York.

NILSSON, N.J., 1996. Introduction to machine learning. http://ai.stanford.edu/people/nilsson/mlbook.html. Accessed Jan. 2006.

OUDEYER, P.-Y. AND KAPLAN, F. 2004. Intelligent adaptive curiosity: A source of self-development. In *Proceedings of the Fourth International Workshop on Epigenetic Robotics* (Lund University), 127-130.

OUDEYER, P.-Y., KAPLAN, F., AND HAFNER, V. 2007. Intrinsic motivation systems for autonomous mental development. *IEEE Trans. Evolutionary Computation 11*, 1, 265-286.

REYNOLDS, C. 1987. Flocks, herds and schools: A distributed behavioral model. *Computer Graphics 21*, 4.

RUSSEL, S. J. AND NORVIG, P. 1995. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, NJ.

SAUNDERS, R. 2001. Curious design agents and artificial creativity. Ph.D. dissertation, University of Sydney, Sydney.

SAUNDERS, R. AND GERO, J.S. 2001. Designing for interest and novelty: Motivating design agents. In *CAAD Futures 2001*, Kluwer, Dordrecht, 725-738.

SCHMIDHUBER, J. 1991. Curious model building control systems. In *Proceedings of the IEEE, International Joint Conference on Artificial Neural Networks* (Singapore), 1458-1463.

SCHMIDHUBER, J. 1997. What's interesting. Tech Rep.TR-35-97, Lugano, Switzerland.

SIMSEK, O. AND BARTO, A.G. 2006. An intrinsic reward mechanism for efficient exploration. In *Proceedings of the 23rd International Conference on Machine Learning* (University of Pittsburgh, PA).

SINGH, S., BARTO, A.G., AND CHENTANEX, N. 2005. Intrinsically motivated reinforcement learning. *Advances in Neural Information Processing Systems 17* (NIPS), 1281-1288.

STANLEY, J.C. 1976. Computer simulation of a model of habituation. *Nature 261*, 146-148.

SUTTON, R.S. AND BARTO, A.G. 2000. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA.

TRAVIS, M.F., BEAR, H.G., HYATT, G., REBER, P., AND TAUBER, J. 2000. Towards more humanlike computer opponents. In *Proceedings of the AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment* (Palo Alto, CA), 22-26.

WATKINS, C. AND DAYAN, P. 1992. Q-learning. *Machine Learning 8,*3, 279-292.

WOOLDRIDGE, M. AND JENNINGS, N. 1995. Intelligent agents: Theory and practice. *The Knowledge Engineering Review* 10, 2, 115-152.

ZELTZER, 1992. Autonomy, interaction and presence. *Presence: Teleoperators and Virtual Environments 1*, 1, 127-132.

Received May 2007; revised July 2007; accepted August 2007