AUTOMATIC ACQUISITION OF ACTIONS FOR ANIMATED AGENTS

Adam Szarowicz [1]
CIS School, Kingston University
Penrhyn Road
Kingston upon Thames, KT1 2EE
United Kingdom
a.szarowicz@kingston.ac.uk

Marek Mittmann [1,2]
Institute of Informatics
Silesian University of Technology
ul. Akademicka 16
PL 44-100 Gliwice, Poland
mittmann@ps.edu.pl

Paolo Remagnino
CIS School, Kingston University
Penrhyn Road
Kingston upon Thames, KT1 2EE
United Kingdom
a.szarowicz@kingston.ac.uk

Jarosław Francik [1,2]
Institute of Informatics
Silesian University of Technology
ul. Akademicka 16
PL 44-100 Gliwice, Poland

CIS School, Kingston University
Penrhyn Road
Kingston upon Thames, KT1 2EE
United Kingdom
jfrancik@ps.edu.pl

KEYWORDS
Q-learning, intelligent agents, lifelike characters.

ABSTRACT

The generation of animated human figures especially in crowd scenes has many applications in such domains as the special effects industry, computer games or for the simulation of the evacuation from crowded areas. Automation in action creation eliminates the need for human labour, which shortens the task of generation of the crowd scenes and also reduces the costs. This paper addresses a shortcoming of the architectures designed for creation of animated scenes with autonomous agents by proposing a module for automatic acquisition of new high-level actions. Agents use reinforcement learning to acquire these actions and the chosen algorithm is the deterministic version of Q-learning. This allows for easy definition of the task, since only the ultimate goal of the learning agent must be defined. Generated actions can then be used to enrich the animation produced by the animation system. The paper also compares results achieved when training agents with forward and inverse kinematics control.

INTRODUCTION

The creation of animated scenes involving interacting characters is a problem in such applications as film post-production and special effects, computer games or event simulation in crowded areas. Crowd scenes, created by dedicated intelligent systems or by human animators, usually rely on a number of high-level actions assigned to the avatars and performed at specific times. In scenes, where the fine detail is not a crucial factor, generation of those actions could be automated. This paper focuses on reinforcement learning as a means of extending such intelligent systems. The proposed solution addresses the elimination of the tedious animation job performed by human animators and directors to create new sets of more complex actions. It also provides a way of easy scripting and parameterisation of generated animation. Such scripting is difficult to achieve when working with sequences acquired by applying manual animation or motion capture techniques. On the other hand automatically generated sequences can later be incorporated into existing animation tools.

The problem of automatic creation of computer characters has been addressed by researchers trying to make the animated characters more intelligent (Funge 1999; Isla et al. 2001). The main problem with those architectures is that they miss features needed to generate realistic crowd scenes (interaction, details of the cognitive architecture (Funge 1999), bias on animal-like creatures in the C4 architecture (Isla et al. 2001; Blumberg et al. 2002) and therefore leave a broad scope for research into the problems involved, especially with human-like creatures. The growing popularity of agent-based architectures and methodologies brought new discoveries into the field of autonomy, distribution and interaction (Rao and Georgeff 1995; Wooldridge et al. 2000; Wood and DeLoach 2001; Mylopoulos et al. 2001; Winikoff et al. 2001) and gave a new opportunity to apply recent advances in AI to the problem of automatic animation generation. However there has been a very limited application of those systems into the field of computer animation.

An example of a system trying to solve this problem is FreeWill (Szarowicz et al. 2002) upon which we based our implementation. FreeWill proposed an architecture suitable to create intelligent and realistic animation by incorporating elements found in both animation-driven systems and distributed (multiagent) solutions. Each avatar participating in the animation consists of an intelligent agent together with a body layer, which is responsible for handling the visible part of the agent.

FreeWill utilises the idea of high-level actions, which are the same as plan libraries often used in the agent literature. These high-level actions (e.g. shaking hands with another avatar or opening a door) contain sequences of simple actions, which can be readily copied and pasted into the action sequence generated by each agent. Therefore the quality of the simulation highly depends on the number of different

[1] Supported by British Council and Polish Committee for Scientific Research as Polish-British Research Partnership Programme project no 239.
[2] Supported by the Polish Committee for Scientific Research under the grant no 4 T11C 024 24 (2003).

high-level actions an agent has at its disposal. An off-line automatic acquisition of those actions would greatly improve the results obtained from the system.

The goal of the presented work is to propose a method for such an automatic acquisition of high-level actions based on machine learning. It should be faithful enough to be applied in crowd scenes thus relieving the human animator of some of the most tedious tasks. A sample action that we have succeeded to automatically generate and acquire consists of opening a door and walking through it.

We are assuming that the avatar is able to perform a number of low-level actions and there is a high-level goal defined for the agent to achieve. The problem solution will include a sequence of low-level actions, which allow the agent to achieve the goal. We will call this sequence a high-level action.

MACHINE-LEARNING BASED ACTION ACQUISITION

A fully automated acquisition of high-level animation actions requires very little user intervention. Only goals for the learning task must be defined, while performance adaptation is unsupervised. Reinforcement Learning (RL) methods (Sutton and Barto 1998; Mitchell 1997) fulfil these criteria. RL is a Machine Learning technique whereby autonomous software (the agent) learns by trial and error which action to perform by interacting with the environment. Explicit precompiled models of the agent or environment are not required. It is sufficient to define states and actions available for them. Although not one of the requirements of the technique, one can imagine that *a priori* knowledge could also be incorporated. For instance both physical and dynamic constraints could be imposed making states and actions partially available or completely unavailable. The Machine Learning technique does learn, incrementally a *reactive* model for each one of the defined states, affected only by the models of neighbouring states. At each discrete time step, the agent selects an action given the current state and executes the action, causing the environment to move to the next state. The agent receives a reward that reflects the value of the action taken given that the agent is in the current state. The objective of the agent is to maximise the sum of rewards received when starting from an initial state and ending in a goal state. One incarnation of RL is Q-Learning (Watkins 1989). The objective in Q-learning is to generate Q-values (quality values) for each state-action pair. At each time step, the agent observes the state s_t, and takes action a. The choice of actions in early stages is usually random (any action may be selected from the possible actions set) and becomes more informed as the agent learns more about the environment (agent prefers actions which give higher rewards thus exploiting its knowledge). After executing an action the agent then receives a reward r dependent on the new state s_{t+1}. The reward may be discounted into the future, that is rewards received n time steps into the future are worth less by a factor γ^n than rewards received in the present (the discount factor expresses the confidence the agent has in the current policy: the further in the future the smaller the confidence). Thus the cumulative discounted reward is given by

$$R = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots + \gamma^n r_{t+n} \qquad (1)$$

where $\gamma \in [0,1)$ is constant for the entire simulation. If $\gamma = 0$ the agent only considers the immediate results of its actions and thus the reward is not discounted. The Q-value is updated at each step using the update equation (1) for a deterministic Markov Decision Process (MDP) as follows:

$$Q_n(s_t, a) \leftarrow r_t + \gamma \max_{a'} Q_{n-1}(s_{t+1}, a') \qquad (2)$$

A sequence of actions ending in a terminal state is called an epoch or iteration.

Q-learning can be implemented using a look-up table to store the values of Q for a relatively small state space. Neural networks are also used for the Q-function approximation (Bertsekas and Tsitsiklis 1996; Haykin 1999).

Reinforcement Learning has been applied to create successful board games implementations (Nicol and Schraudolph 1994; Thrun 1995), with unmanageable state spaces. Backgammon is the most successful example (Tesauro 1994). Reinforcement Learning has also been used in robotics to control one or more robotic arms (Davison and Bortoff 1994; Schaal and Atkeson 1994) and in animation to create motivational and emotional states for human-character interaction (Yoon et al. 2000), no results have hitherto been published on applying RL to control complex biped actions either in simulated or robotic environments.

The only architecture of the above-mentioned, which explicitly employs machine learning is C4, where machine learning techniques allowed the animated creatures to acquire new skills, especially in the form of training a home pet (a dog).

For the purpose of Q-learning our agent was assigned a number of simple actions, which are listed below:

- **Forward kinematics control**
  - Rotate arm up / down by $\Delta\alpha$
  - Rotate arm forward / backward by $\Delta\alpha$
  - Rotate forearm by $\Delta\alpha$
  - Move forward / backward by $\Delta\alpha$
- **Inverse kinematic control**
  - Move palm by Δx, Δy, Δz
  - Move forward / backward by Δx

where $\Delta\alpha = 20$ degrees, $\Delta x = 35$ cm for walk (the size of a single step) and $\Delta x = \Delta y = \Delta z = 5$ cm for the motion of a hand, $\gamma = 0.95$.

It was also assigned a goal of getting itself behind a closed door. Its task was to learn a way of doing so. The only represented states were those of the agent, the state of the door was represented externally as a variable to reduce the size of the state space. The door knob was not rotated in the experiments, although it is relatively easy to add such an action without much increase in the simulation time (in another task currently under investigation the agent has to grab an object).

Positive rewards were given to the agent whenever it managed to get through the door successfully, whereas negative rewards were used to prevent it from performing illegal

moves, e.g. outstretching a joint or colliding with an obstacle. For this purpose a quick and efficient way of detecting collisions was necessary, as collision detection algorithms were not a part of the animation package.

The agent's perception of the ambient environment was limited to detecting the collisions between its arms and exterior objects. This approach is sufficient in most cases, as the collisions practically determine constrains for the physically possible movements (at least these not resulting from internal, i.e. biomechanical conditions). The desired collision detection algorithm should avoid testing all the polygonal faces in both objects for overlap. Instead, much more efficient solutions are based on spatial volumes (volumes that entirely enclose objects). Tightness of fit between an object and its bounding volume is crucial for the precision of the collision test. The simplest, yet widely used bounding volumes are spheres. However, when objects are in close proximity this approach is imprecise or it requires strong spatial subdivision techniques applied in a hierarchical manner. Another solution is axis aligned bounding boxes method (AABB). This method also produces relatively large bounding boxes if the underlying objects are not axis-aligned, and that was the case with the avatar's arms. Therefore we have chosen the OBB (oriented bounding boxes) approach (Arvo and Kirk 1989) as a good trade-off: they are a snug fit and the method is not very computationally intensive. Applying the separating axis theorem proposed by Gottschalk (1996; Gottschalk et al. 1996) it is enough to make just 15 tests to determine if the boxes overlap.

Two ways of character control were tested: forward kinematics and inverse kinematics. The state space for the first case consisted of approximately 12000 states distributed across 4 dimensions (2 degrees of freedom for the left arm, 1 for the forearm and 1 for backward/forward walk). The second simulation comprised less than 50000 states (a 3-dimensional cube of x,y,z positions around the avatar's hand, the last dimension was walk along one axis), eight simple actions were available at each time step. These were three rotations -- two for the arm and one for the forearm -- in two directions and walk along one axis for the forward kinematics case (2*3+2) and hand motion along 3 spatial axes in two directions for each axis plus walk for the inverse kinematics case (2*3+2). The difference in the number of states for the two modes was caused by the necessity for greater sampling of the space in the case of inverse kinematics control. Number of states across each dimension was chosen to provide sufficient sensitivity but also to eliminate as many unnecessary states as possible. Therefore only reasonable angles for joint movements were selected, these were taken from human joint contraints: forarm can only rotate by about 180 degrees around the x-axis, arm 270 degress around the x-axis (forward/backward) and 180 degrees around the y-axis (up/down). Two additional states were added for each joint to represent the illegal motions, so called forbidden states (e.g. for the forarm rotation -20 degrees and 200 degrees would be the forbidden states). For walking only the route through the door was represented as walking to the door could easily be achieved within the FreeWill system.

THE FRAMEWORK

The experimental apparatus consists of an application communicating with 3DStudio Max software package, which was used to model the three-dimensional scene. The characters were created by the Biped Plugin.

The main part of the framework is an external application written in C++ that processes all simulations. It controls the scene and acquires information about object interactions through the COM interface, which is exposed from within the 3DStudio package by means of a dedicated script (a max script). This script contains definitions of functions for manipulating the objects and defines the appearance and initial positions of objects placed in the scene. Since the communication through COM interfaces is not fast enough, some critical components have been moved to a plug-in, that directly communicates with the max script engine. This solution is very efficient, but inconvenient to implement, so only necessary functions have been implemented in this way. The final sequence of actions is saved as a script controlling the avatar and the scene objects. Animation created with that script can then be rendered.

Recently another software architecture has emerged. The KINE+ framework (Francik 2003) makes the main application independent of the 3DStudio Max concerning the simulation task. The new framework contains a biped model compatible with 3DStudio, and has collision detection built-in. However it is not yet fully integrated with the main solution, we have strong grounds to believe that it will significantly improve its efficiency. Additionally, it supports animation output format compatible with MoCap.

RESULTS

The results obtained are depicted below. Figure 1 shows a few shots from the actual animation generated from the sequence of simple actions learned by the agent. At the beginning of the simulation the agent has no knowledge about the appropriate actions, and as the simulation progresses it gradually improves its performance. Figure 2 shows an average number of low-level actions performed per epoch as a function of time. Initially the numbers are low: `inexperienced' agent immediately encounters negative terminal states: collisions and forbidden states (see Section 2). It then gradually explores the state space until eventually the best path is found. This is when the number of actions performed stabilises (until perhaps the agent finds an even better path or decides to explore again).

The stabilisation is more stable in the case of the forward kinematics (FK) control (for this method of control the agent had fewer ways of fulfilling the goal) and the solution size is about 5 simple actions. For the state space defined above (4 dimensions, 12000 states) the solution was found in about 250 iterations (epochs), on a computer with Pentium 4 1,50GHz processor, 0,5GB RAM, and with the scene redraw switched off this was about 30 seconds. Because the animation obtained still relied on unnatural moves (lack of arm rotation) which could not be eliminated within the action set given a bigger state space was also investigated. Two additional degrees of hand freedom (hand and arm rotation around the z-axis) were added which made the total

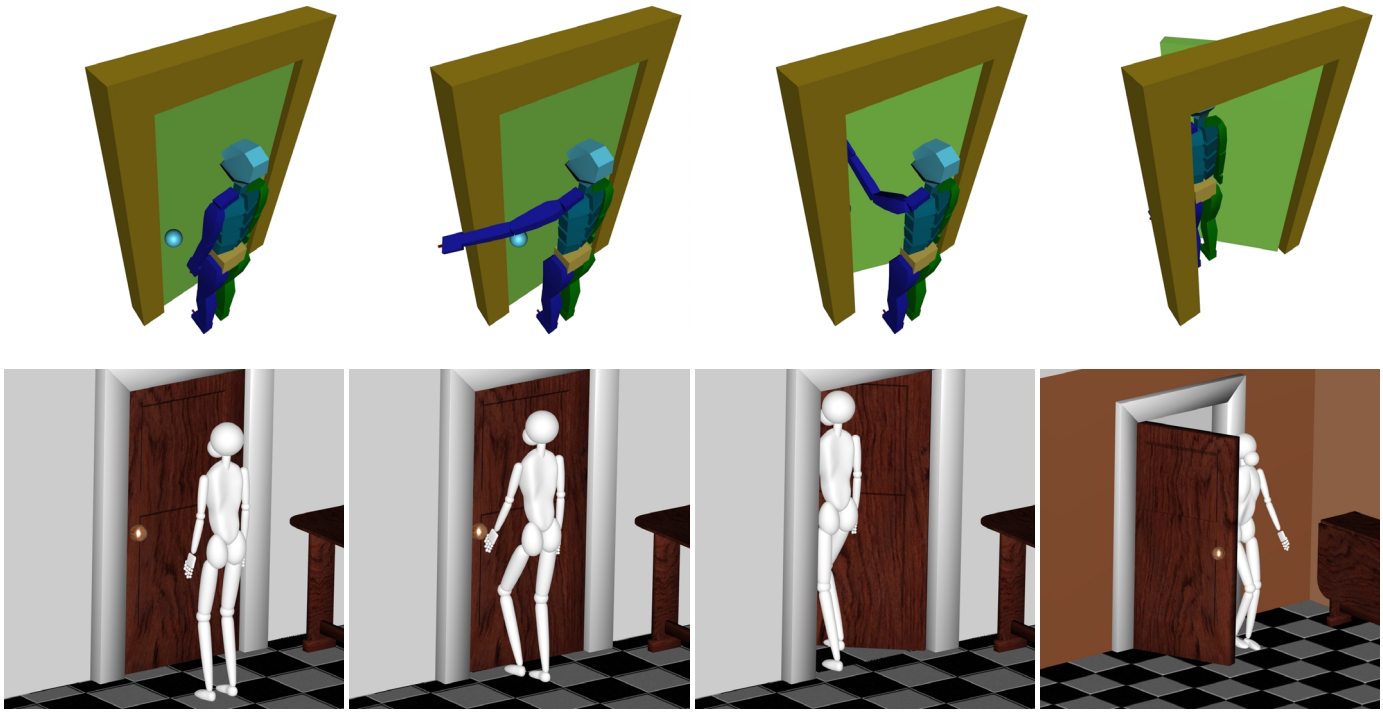

Figure 1: Learning avatar (above), animation derived from the best solution found (below)

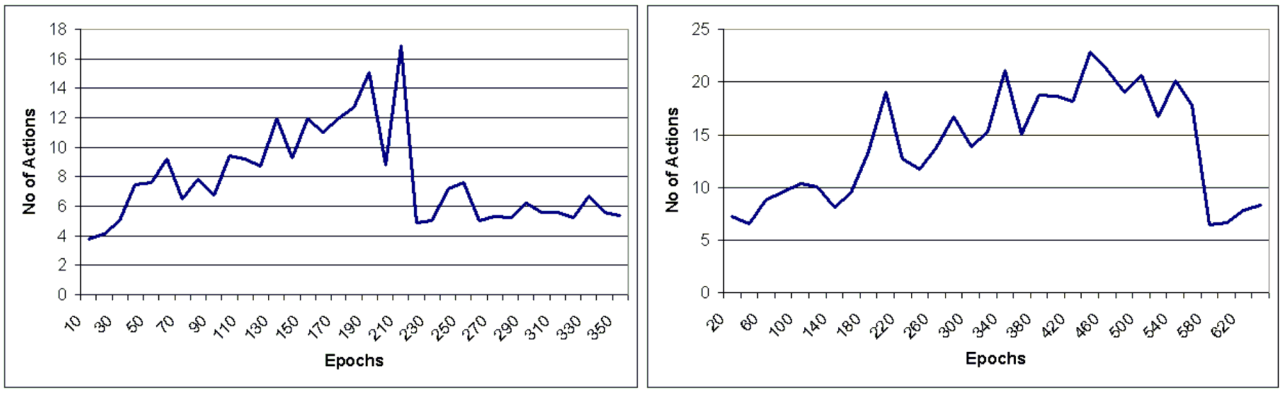

Figure 2: Learning curves (left – forward kinematics control, 12000 states; right – inverse kinematics, 50000 states)

number of states of over 2 million and 12 actions per state. In this case the solution was usually found in about 1500 iterations (80 seconds), but some initial exploration was enforced on the agent -- for the first several hundred iterations the agent started each iteration in a different, randomly chosen state. The obtained solutions were very similar.

For the inverse kinematics the stabilisation threshold is slightly higher than in the previous case: 6-7 simple actions (more actions are necessary because of smaller steps). Inverse kinematics generates more natural motion (joint angles, position of limbs in space) during the simulation thus implicitly rejecting some of the unnecessary states. It also requires fewer constraints to be checked against during the simulation (e.g. in terms of limits on joint angles), the representation was more natural and no extensions to it were necessary. However it takes longer to learn (approx. 800 iterations in 90 seconds). Also, decreasing the sampling rate for the IK control was increasing the number of iterations necessary to find the solution. We believe this was due to loss of precision by the avatar when performing the actions, that is, because of the rare sampling of the space, the avatar had to be very `lucky' to come across the door knob, open the door and fulfil the goal. The solution found was comparable to that acquired for the FK control - both the general motion of the agent and the timing were similar.

Although adding just two degrees of freedom increased the learning time from 30 to 80 seconds the resulting state description allows to simulate most manoeuvres of a single upper limb. Therefore further extensions of simulations involving one hand will not turn this solution intractable. Current experiments with an avatar learning how to lift an object confirm this statement.

CONCLUSIONS AND FUTURE WORK

In this paper a way of learning sequences of low-level actions to achieve a goal of an animated agent has been explained. The agent has been controlled using both forward and inverse kine-

matics and the learning algorithm applied was Q-learning. This algorithm proved to be sufficiently effective to learn new actions by a virtual agent with several degrees of freedom. Another benefit given by this technique is that automatically generated sequences can easily be scripted and parameterised and used in other animation tools. Application of a solution to a different character could be done by adjustment of the resultant motion parameters (e.g. for some types of motions only some of the angles have to be modified), it is also possible to learn a new sequence for each distinct character.

The created animation sequences are faithful enough to be applied in a crowd scene. Additionally our results can also be applied in the field of robotics, provided that the robot can already perform more basic actions such as walking. The experiments show that although inverse kinematics control takes longer to reach the solution it is easier to program (fewer dimensions in the state space, less different low-level actions). On the other hand scaling up is easier for the forward kinematics (the representation of states is more consistent). Additionally, because the technique generates multiple solutions, different sequences can be used by the avatars in the final crowd scene to perform the same task. This way of introducing randomness into the scene would generate more realistic results than the current techniques relying on phase offseting.

However, adding too many degrees of freedom to the presented technique will eventually create a very substantial state space with long simulation times and therefore a more compact representation is required. Therefore our next step will be an application of neural networks for state approximation. Other learning techniques (such as genetic programming) will also be applied to the constructed framework to compare results achieved from different methods.

So far we have only experimented with avatars interacting with static objects. A bigger challenge would be to try to learn interaction between agents – e.g. passing an object. The experiments presented in this paper provide good foundation for attempting that challenge.

REFERENCES

Arvo J. and D. Kirk. 1989. *A survey of ray tracing acceleration techniques. An Introduction*.

Bertsekas D. P. and J. N. Tsitsiklis. 1996. *Neuro-Dynamic Programming.* Athena Scientific.

Blumberg B. M., M. Downie, Y. Ivanov, M. Berlin, M. P. Johnson and B. Tomlinson. 2002. "Integrated learning for interactive synthetic characters". *ACM Transactions on Graphics* Vol. 21, No 3, pp. 417–426.

Davison D. E. and S. A. Bortoff. 1994. "Acrobot software and hardware guide". Technical Report Number 9406. Systems Control Group, University of Toronto, Toronto, Canada.

Francik. 2003. "A Framework for Programmatic Control of Animation of Human Avatars". *Studia Informatica,* to appear.

Funge J. D. 1999. *AI for Games and Animation. A Cognitive Modeling Approach.* A K Peters Natick.

Gottschalk S., M. C. Lin and D. Manocha. 1996. "OBBTree: A Hierarchical Structure for Rapid Interference Detection". *Proceedings of ACM SIGGRAPH*, New Orleans, Lopp. 171–180.

Gottschalk S.. 1996. "Separating axis theorem". Technical report TR96-024. Dept. of Computer Science, UNC, Chapel Hill.

Haykin S. 1999. *Neural Networks.* Prentice Hall.

Isla D., R. Burke, M. Downie and B.M. Blumberg. 2001. "A Layered Brain Architecture for Synthetic Creatures". *Proc. of 17th Joint Conf. on Artificial Intelligence IJCAI-01*, Seattle, USA, pp. 1051–1058.

Mitchell T. M. 1997. *Machine Learning*. McGraw Hill.

Mylopoulos J., M. Kolpand and J. Castro. 2001. "UML for Agent-Oriented Software Development: The Tropos Proposal". *Proc. of the 4th Inmt. Conf. on the Unified Modeling Language*, Toronto, Canada.

Nicol N., Schraudolph, P. Dayan and T.J. Sejnowski. 1994. "Temporal difference learning of position evaluation in the game of Go". In *Proceedings of Advances in Neural Information Processing Systems Conference*, San Mateo, CA, pp. 817–824.

Rao A. S. and M. O. Georgeff. 1995. "BDI Agents: From Theory to Practice". *Proc. of the 1st Conf. Conference on Multiagent Systems ICMAS95*.

Schaal S. and Christopher Atkeson. 1994. "Robot juggling: An implementation of memory-based learning". *Control Systems Magazine* No 14.

Sutton R. S. and A. G. Barto. 1998. *Reinforcement Learning: an Introduction*. MIT Press.

Szarowicz A., J. Amiguet-Vercher, P. Forte, J. Briggs, P.A.M. Gelepithis and P. Remagnino. 2001. "The Application of AI to Automatically Generated Animation". *Advances in AI, Proceedings of the 14th Australian Joint Conf. on Artificial Intelligence*, Springer LNAI 2256, pp. 487–494.

Tesauro G. 1994. "TD-Gammon, a self-teaching backgammon program, achieves master-level play". *Neural Computation* Vol. 6, No 2, pp. 215–219.

Thrun S. 1995. "Learning to play the game of chess". *Advances in Neural Information Processing Systems* (G. Tesauro, D. S. Touretzky, and T. K. Leen, editors), The MIT Press, Cambridge, MA.

Winikoff M., L. Padgham and J. Harland. 2001. "Simplifying the Development of Intelligent Agents". *Advances in AI, Proceedings of the 14th Australian Joint Conf. on Artificial Intelligence*, Springer LNAI 2256, pp. 557–568.

Wood M. and S. A. DeLoach. 2001. "An Overview of the Multiagent Systems Engineering Methodology". In *Agent-Oriented Software Engineering*, LNAI 1957, Springer.

Wooldridge M., N. R. Jennings and David Kinny. 2000. "The Gaia Methodology for Agent-Oriented Analysis and Design". *Autonomous Agents and Multi-Agent Systems*, Vol. 3, No 3, pp 285–312.

Yoon S.-Y., B. M. Blumberg and GG. E. Schneider. 2000. "Motivation Driven Learning for Interactive Synthetic Characters". In *Proceedings of Autonomous Agents Conference*, 2000.