Layering and Heterogeneity as Design Principles for Animated Embedded Agents

A. Szarowicz [a], J. Francik [a,b,1], M. Mittmann [b], P. Remagnino [a]

[a] *Digital Imaging Research Centre, Kingston University, UK*

[b] *Institute of Informatics, Silesian University of Technology, Gliwice, Poland*

Abstract

Animation of three-dimensional digital characters is still a major hurdle in the production of films and video games. This paper analyzes the technology of autonomous embedded agents as a solution to this challenge. Multi-layering and heterogeneity are chosen as desirable principles for the design of digital characters. The FreeWill+ animation framework has been developed with these principles in mind. Its general structure as well as some of its special features are presented, including communication issues and learning capability.

Key words: animated agents, multi-layering, lifelike characters, three-dimensional animation, learning systems

1 Introduction

The animation of three-dimensional characters has become an important issue for postproduction industries (for instance special effects geared for movies and games) (Parent, 2002). While current computer graphics technology can supply very realistic results, the control of animated characters (anthropomorphic or animal-like) is still a problem to be solved. The research community endeavors are mainly focused on the creation of realistic animats [2]. It seems quite natural to implement autonomous characters as autonomous agents, which we

Email addresses: a.szarowicz@kingston.ac.uk (A. Szarowicz), jfrancik@ps.edu.pl (J. Francik), mittmann@ps.edu.pl (M. Mittmann), p.remagnino@kingston.ac.uk (P. Remagnino).

[1] Work supported by European Commission as a part of Marie-Curie fellowship.

[2] *animated* characters take many names, we will use all of them interchangeably as the basic concepts are a common denominator to all proposed definitions

will call *animated* agents. Just like any *animated* creature, they consist of a body capable of animation, but unlike them they can also be endowed with an artificial mind. As such they are capable of independent planning, and by *embedding* intelligence they provide a vital augmented level of automation used to *give life* to human and animal like characters. This augmentation enables creativity and greatly enhances the efficiency and flexibility when used with both commercially available animation packages and dedicated frameworks.

Our 3D digital character can be seen as an intelligent agent embedded in a virtual body through which it can perceive and act upon its digital surroundings. While this approach is similar to robotics, it is free from many problems typically encountered in robotics research. The behavior and motion principles to which synthetic characters must obey are limited only by their creator's imagination and not by real physical constraints. This also alleviates problems related to uncertain sensing mechanisms and imperfect control and focuses on creation of a character body and mind. On the other hand, individual behavior may by itself be quite difficult to model, even in its most elementary form. Features like environment awareness, purposefulness (goal-orientation), attitude, and, maybe the most important, the viewer's subjective illusion of life should be considered crucial, and integrated in the model.

The level of difficulty increases significantly when several individuals are involved; further increase occurs when a flock (or crowd) animation is modeled. In this paper we put forward an architecture that combines multiple goal-oriented communicating individuals and their behaviors to form a group (a flock or a crowd). More traditional *teaming* systems promote the collective (crowd or flock) itself to agent status, creating a holistic semi-autonomous being, and this is achieved at some expense of the individual autonomy.

This paper presents some aspects of an agent-based architecture called FreeWill+, which supports creation of autonomous virtual creatures. It is capable of generating a crowd of digital characters, which are able to autonomously behave inside a digital scene, fulfill their goals, maintain beliefs and execute actions. These actions include both reactive movements as well as more deliberate and complex behaviors, with complex, co-operative tasks as the eventual aim of the project.

In the next section agent technology will be characterized and categorized, and the main distinctive features will be analyzed in relation to their relevance to character animation. A couple of agent-based animation solutions will be reviewed. In Section 3 the leading role of multi-layering and heterogeneity as design principles will be justified. These principles have been applied in the FreeWill+ system. Subsequent sections cover the general architectural outline of the system (Section 4), and a chosen functionality area: the ability of agents to learn (Section 5).

2 Agent Systems and Animated Agents

There are almost so many definitions of the notion of *agent* (or, more precisely, *intelligent agent*) as the attempts to define it. An exhaustive review of definitions can be found in (Franklin and Graesser, 1996), and it does not seem that the situation changed significantly since then. Among this variety of definitions some important features are most commonly pointed. The list provided below is based primarily on definitions given by Russell and Norvig (1995), Jennings and Woolridge (1998) and Jennings (2001). These are:

- autonomy: an ability to act without the direct intervention of other agents
- reactivity: an ability to perceive (sense) the environment and to act upon it
- pro-activeness: purposefulness, goal oriented operation
- social ability: interaction, communication and collaboration

The crucial role of autonomy is beyond any discussion. The agent's ability to react within its environment is essential as well, especially when animated agents are considered. Even in rare cases when animated characters do not interact with any elements of a virtual world they will at least act upon the stimuli generated by a human user. A good example may be a translation agent that interprets written or spoken sentences using sign (gesture) language (Francik and Fabian, 2002). Normally the agent behaviour should depend on what it perceives. Otherwise the contribution to automation of the animation process would be at least questionable. Animation based on this perceive–act paradigm is usually called *behavioural animation* (Reynolds, 1987; Tu and Terzopoulos, 1994). To achieve this, a general decisive structure should be enough. However, a much wider accepted approach, claimed by Evans (2002) an *orthodoxy* of agent technology, is the *Belief–Desire–Intention* architecture (Bratman, 1987). The *Beliefs* are data structures representing agent's knowledge about other objects, grounded in his own perception, the *Desires* represent goals that should be attempted, while *Intentions* may be interpreted as the actual plans how to achieve them. Technically, the intentions are calculated as functions of desires analyzed in relation to beliefs (perception). Some decisive structure is necessary; Evans (2002) calls them *Opinions*. Actions (reactions) are simply a consequence of realising intensions. An example of early implementation of this model is the BDI system (Rao and Georgeff, 1991), while more recent achievements may be represented by the IVA–Intelligent Virtual Agent solution (Monzani, Caicedo and Thalmann, 2001; Monzani, 2002). Another system called *Massive*, created purposely for *The Lord of the Rings* film production (Koeppel, 2002) controlled over 50,000 animated fighters in the movie's crucial battle scene of Helm's Deep. Similar systems controlled crowds of animated agents in such productions as DreamWork's *Ant Z* or Pixar's *Bug's Life*.

A simple autonomous reactive system, with a straightforward decisive engine, may be considered as an essential minimum for animated agents. Although helpful, such systems would not be counted among agent systems according to some stronger definitions (e.g. Jennings and Woolridge, 1998). An almost-indispensable feature of agency is the ability to behave proactively. It involves structured planning appropriate actions, and cannot be merely based on direct reacting to the environment stimuli, using a simple decisive structure. Distinction between reactive and proactive is not strict. Proactive is by some authors described as purposeful, or goal-oriented (e.g. Franklin and Graesser, 1996). It is difficult however to say that reactive style of behaviour is purposeless or not goal-oriented. Engaging the agent's own initiative is often stated as essential. Still it sometimes leads to difficulties. Shaking hands on meeting a friend is an own initiative of an agent or merely reaction for perceiving a stimulus? Unsurprisingly in this context, pro-active behaviour may be well implemented with the apparatus already mentioned above: the Belief–Desire–Intention architecture. The systems mentioned in the previous paragraph, BDI and IVA, are both capable of pro-active behaviour. On the other hand, the Massive system generally reveals reactive actions only; the authors do not have enough knowledge concerning DreamWorks and Pixar systems, but any manifestation of pro-active actions may be at most marginal. To summarise, pro-activity may magnificently enhance the animation, yet it is not essential, especially in crowd simulation. In the solution presented further on both reactive and pro-active approaches are implemented.

The role of social abilities is not so evident. To some extent an interaction occurs each time agents moving around in their virtual world avoid collisions between themselves. However this kind of trivial interaction does not go beyond a usual behaviour inherent to the reactivity feature; other agents are treated just as any objects existing in the environment. Still more advanced forms of interaction may be achieved on the perceiving-acting basis only, what we will show later. However useful, social ability does not seem to be essential for animated agents. On the other hand, communication skills may greatly enhance flexibility of an animation system. Examples of communicative animated agents may be found in (Allbeck and Badler, 2002; Caicedo, Monzani and Thalmann, 2001; Arafa et al., 2002). Our conception goes even further. As described in section 3.3, we use *smart objects*, first introduced by Kallman (2001). All unanimate objects in the virtual world are agents as well; hence whenever an agent initiates a reactive or pro-active behaviour, an inter-agent communication is inherently engaged.

Regardless of the features discussed above, autonomous, intelligent agents may have other attributes; the attendance or lack of them determines further classification, provided as a grid of diagonal subcategories. We will attempt to identify animated agents within these subcategories. An agent can exist in isolation (e.g. a data miner agent) or it can be *embedded* in some environment,

probably inhabited by other agents. Obviously animated agents are embedded. They are also *embodied*, i.e. they manifest in some kind of a body, in contrast to dealing directly and exclusively with abstract information. The subcategory of *embedded, embodied* agents incorporates also autonomous robots; hence we will distinguish a subcategory of *virtual agents* in contrary to the *real* ones. Last but not the least subdivision is into *believable* agents, i.e. providing the illusion of life, and *effective* agents, focused on effective fulfilment of their goals (e.g. robot simulators). Both subcategories may concern animated agents, however only the believable agents are in scope of this paper.

Animated agents are therefore: autonomous, reactive, preferably pro-active and social. They are also: embedded, embodied, virtual and believable. Some of them are learnable (adaptable): this ability may be treated as an ultimate form of being pro-active.

Two obvious applications for animated agents are games and film production (other possibilities are educational systems and simulation systems). For games, the technology offers excellent potential for foreground character steering; one of the leading examples is the *Black & White* game. The level of control and – first of all – believability that may be achieved is still highly insufficient to animate foreground characters. However the animated agents have a strong position in crowd animation, several examples of which have been mentioned above.

The theory and practice of crowd animation systems has been inspired by the revolutionary work of Reynolds (1987) on flocks of animals, in his case birds (or *"boids"*). His approach adapted some ideas from particle systems: each of his birds is a particle. The crucial for the whole domain of *flocking* systems is the aggregate behaviour. The autonomy of each individual is somewhat limited by the autonomy of a flock. Tu and Terzopoulos (1994) and Tu (1999) present a much more behavioural approach. An interesting solution is ViCrowd (Musse and Thalmann, 2001), in which individuals in a virtual crowd can have variable levels of autonomy, i.e. rule-based, scripted or guided interactively by the user.

3 Design Principles for Animated Agent Systems

Animating a virtual character is such a complex task that it should be decomposed into simpler elements. Multi-layered architectures are not new in this domain. In fact almost every practical implementation is divided into a number of parts or layers. We will analyze just two examples of such architectures.

Monzani, Caicedo and Thalmann (2001) propose an architecture that is generally divided in two parts: the body and the brain. This division is made primarily according to the abstraction criterion (low level: the body and high level: the brain). The authors claim however that low level does not mean in this case being simpler or easier. Both parts are almost equally complex. The body part is responsible not only for the geometric and kinematical (physical) aspects, but also incorporates low behavioural level, including actions like walking, grasping etc. In other words it extends to the domain of actions that - in case of humans - are made unknowingly, without a deliberate activity of the brain. The mind part is limited to "consciousness": deliberate planning and emotions. A characteristic feature of their approach is a very strict border between two parts. They communicate through a general-purpose network TCP/IP connection only and therefore they may be physically distributed on separate machines.

Bruce Blumberg and his team propose a multi-level brain for their C4 system of synthetic creatures (Russell and Blumberg, 1999; Isla et al., 2001). It is divided into distinct systems, which communicate through common access to an internal mental blackboard. The brain consists of several distinctive layers which are: sensory system, perception system, action system, navigation system and motor system, all with assistance of working (perception) memory and aforementioned blackboard. The C4 architecture incorporates direct support for multi-level direction.

However relevant is the argument of complexity management to the application of multi-layered architecture in animation systems, there are even more important arguments for the multi-layered, heterogeneous architecture. Winikoff (2001) notes that the inherent property of an agent is having many goals and many ways of fulfilling them. Our intention is to enable various ways for achievement of the same task. Which way will be chosen depends on various factors, like the actual context, availability of resources, working mode (e.g. real-time animation and off-line generation), not excluding pseudo-randomised factor. Our main design objectives are: open architecture, diversity of internal solutions, flexibility, interoperability and adaptability.

The architectures mentioned above, although multi-layered, contain fixed, dedicated structural parts. This type of internal structure does not facilitate the flexibility of the way, in which goals are performed. It tends to behave in a fixed way and special care is needed to maintain diversity and flexibility.

3.1 The FreeWill+ Framework

The proposed, new framework is called FreeWill+ (Francik and Trybicka-Francik, 2003) and it is an evolution of its predecessor system, FreeWill (Szarowicz et al., 2001). It provides general means for co-operation of various (heterogeneous) agents. The only condition for them to operate is the acceptance of common interfaces (or: protocols) for communication. This does not rule out the common use of some low level tools and mechanisms, however there are no (or almost no) fixed, dedicated parts in the higher levels.

The project's main design principle is a multilayered architecture dealing with an increasing level of abstraction in each part of the character's design (graphical representation of the body, motion description, intelligence). A similar architectural principle is also applied when modelling different aspects of the simulation (simulation engine, digital world, objects, characters, agents). Software components responsible for the body and mind parts may be diversified. However, on a more detailed level, the system consists almost entirely of loosely coupled modules, that arrange themselves into temporary, *ad hoc* created associations.

All high level functionality is entirely maintained by an open set of loosely coupled components which hold a common name of *actions*. Actions are software components providing common interfaces to allow communication in a uniform way. They are capable of fulfilling partial tasks and goals required by the agents. This is the outcome of a collective work of the action components on various levels of abstraction. The highest level involves both reactive and pro-active operation modes and is capable to make use of artificial intelligence to create and consolidate plans. The plans are subsequently decomposed into simpler tasks performed by the middle level. In the lowest level action components control the underlying virtual body in a straightforward, procedural way.

The agents in this structure should be considered simply as the entities that have – besides their virtual bodies – the access to a society of available actions. Agents are free to choose any of them, and on any level of organisation, so that to gain their goals [3]. It may be completed entirely by a single action component, but it is a rather rare case. Usually the task is decomposed into a sequence of simpler operations, which are redirected to lower-levels. Moreover, the situation, in which a single task may be successfully performed by many action components, is normal. Therefore, usually it is not straightforward to determine which one of them should be chosen for a given task; even then, there is still a question of which actions will be taken to perform sub-tasks after the

[3] Agents may manifest some preferences in this matter just to amplify their individuality.

decomposition. The whole structure is highly dynamic, with individual action components easy to interchange. There are no fixed hierarchies besides the ones created *ad hoc*, on temporary basis, just to fulfil a given task [4].

The intelligence of agents in FreeWill+ architecture is distributed among a society of action components. This resembles an architecture with agents and sub-agents; although some of such sub-agents are quite intelligent, we should notice that still the majority of actions do not meet the definition of agency: they happen to be very simple and show no autonomy at all. The presented design principles ensure flexible operation in heterogeneous environment, with components easy to interchange, which gives the solution the essential diversity of behaviour.

Future work involves maintenance of various levels of details (LoD) based on control of suitability factors. This work is inspired by the achievements of Sullivan et al. (2002) in obtaining different LoD on, among others, behavioural basis.

3.2 Action Components and their Instant Associations

The FreeWill+ only distinct set of fixed components is the low level body. It is significantly narrower than the Monzani's body. It incorporates kinematical (skeleton) model, mesh model for the final rendering and some tools for collision detection and path finding (A* component). Future extensions can also accommodate a component for dynamic physics-based modelling. An extra part of the solution is a real-time renderer and set of import-export plug-ins.

A question comes to mind: how action components are associated to establish a temporary hierarchy? To achieve this functionality, the action main interface (in our solution – *IAction*, Fig. 1) provides a query function that makes it possible to check – on a symbolic basis – if the action is capable to fulfil a given task, and if it is, how much suitable it is.

In Fig. 2 the collaboration between actions is shown. The request to *perform* a task comes from a higher level action (1); it may come from the agent itself as well. The calling level action identifies necessary sub-tasks and requests a dedicated component called *context* to *associate* lower level actions for the given task (2). The *context queries* actions against their suitability (3) and, finally, the most suitable components are chosen to *perform* the subtasks (4).

The capability of an action to perform the given task depends on a number of conditions. Obviously the most important is the inherent ability to perform

[4] To optimise, some hierarchies may be cached for later use.

```
interface IAction : IFreeObject
{
   // the context object
   [propget] HRESULT Context([out] IFreeContext**);
   [propput] HRESULT Context([in] IFreeContext*);

   // query functions
   HRESULT Query(
      [in] IFreeSymbol *pTask,     // task to be performed
      [in] ULONG nFlags,           // additional flags
      [out] ULONG *pnSuitability); // answer: suitablity factor

   // performing
   HRESULT Perform(
      [in] IAction *pInitiator,    // initiating action or agent
      [in] IFreeSymbol *pTask,     // task to perform
      [in] IFreeParams*);          // action per-task params

   // more functions not listed...
};
```

Fig. 1. IDL/ODL interface definition for an action component

the particular type of tasks; moreover the capability is a recursive function dependent on the capabilities of all the lower-level actions, as well as of other necessary resources. Still all capable actions may differ in the degree of suitability. More specialised actions are usually supposed to be more suitable than generic ones. There may be a number of other factors to have an influence; one of more important is if the mode of operation is off-line or on-line. There is a metric of suitability provided, that makes it possible to compare different actions and choose the best suited. Still, the task performance may fail; a next best action is then chosen to retry.

3.3 Communication, Interaction and Perceiving

A broad set of action components existing in the system gives the necessary means for maintaining the inter-agent communication in a uniform way. The overall communication is based on message-passing paradigm. Sending a message is reduced to requesting a specific action component to *perform* its task on behalf of the receiving agent. These requests are carried out indirectly: a dedicated *scheduler* is involved. Its responsibility is to locate the receiving agents and their particular actions, but also dealing with the message-passing

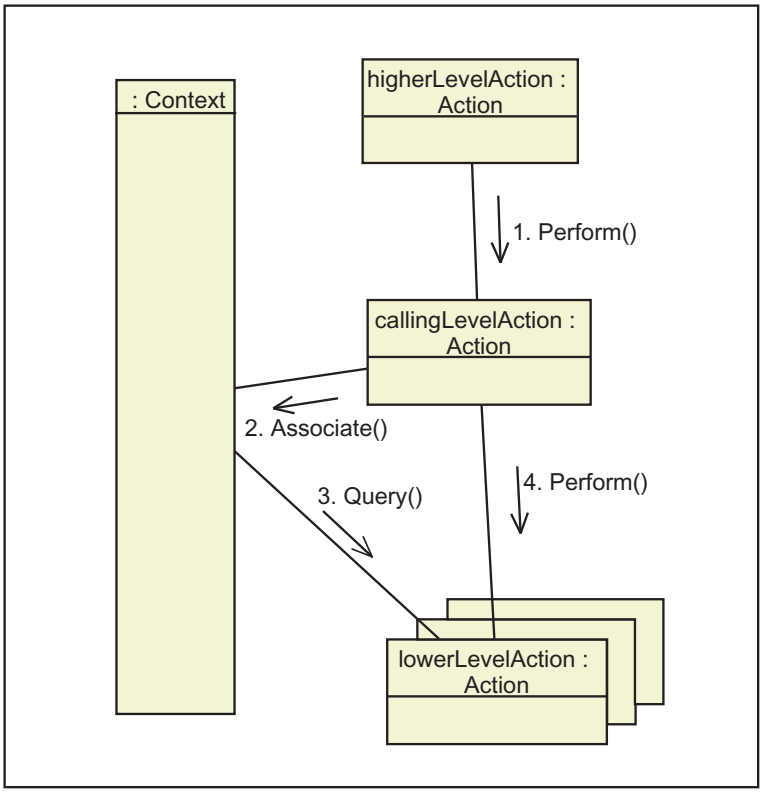

Fig. 2. Diagram of collaboration between action components.

in both temporal and spatial way (see Bandini, Manzoni and Simone, 2002, for spatial processing in agent systems). Message delivery may be individual, simply broadcast or broadcast to a limited area. Messages can be delayed or suspended, and triggered automatically on occurring an event or satisfying a precondition, for instance after completion a given task or entering a given state or a given area.

FreeWill+ uses the concept of *smart objects*, first introduced by Kallman (2001). The notion of smart objects refers to inanimate objects that are represented by agents, very much like virtual characters. Smart objects are usually not so "smart" as the animats, however they inherently introduce diversity into the virtual world. They can interactively instruct the "living" agents how to use them; therefore the latter require no knowledge on how to use objects in a smart way. Generally, every object in FreeWill+ is potentially smart (built with *agent-enabled* components), therefore the intelligence in that virtual world really corresponds to an *animated* environment. Almost all interactions of an animated agent with its surroundings are in fact of inter-agent nature. The only exception to that rule is collision detection - this kind of geometrically-based interaction is implemented as a low-level feature.

A distinctive feature of FreeWill+ is the lack of any special components dedicated to sensing/perceiving the surrounding world. Such components are quite

natural in robotic agents, for which separate sensing devices are essential, and are consequently adopted in a vast majority of animation systems. Unlike them, FreeWill+ perception takes place by the activation of one of its action components, exactly like any other high-level activity. They include "sensing statement actions" (*you can sense me*) as well as "sensing question actions" (*can I sense you?* or *which of you can I sense?*). The overall perception is reduced just to passing messages between such formulated action components. An important addition to this schema is the mechanism of filters, that limits sensing according to physical constraints (distance, orientation etc.).

As was shown in this section, the entire agent activity, including communication, interaction and sensing, is made on the basis of utilising the loosely layered structure of heterogeneous action components.

4 The Layered Structure of FreeWill+

The FreeWill+ system consists of several modules. Their structure and general relations are depicted in Fig. 3 as a simplified UML diagram. The essential agent architecture, as described in the previous Section, is entirely encapsulated in a single module called ACTION+. This module implements the whole actual agent behaviour; all the other modules provide the low level body (MESH+, KINE+, NAVI+), the renderer (RENDER+), and some vital, although rather marginal instrumentation utilised by the actions (FREE+).

The detailed description of the FreeWill+ modules is as follows:

- **MESH+** module encapsulates support for vertex meshes. It is a very low level component, operating close to the hardware abstraction level (HAL), and therefore highly dependent on the platform. The current implementation supports Microsoft DirectX version 9.0. The module contains vertex and face buffers, realised normally as strictly platform-dependent, and a more abstract mesh object that incorporates some higher-level operations. The latter supports vertex blending for single-mesh animation of multi-bone systems, resulting in flexible animation of deformable bodies. Both indexed and fixed vertex blending is implemented, with subdivision of larger meshes in case of insufficient number of bones supported by the hardware.
- **KINE+** module provides a bone (skeletal) system to drive motions of the embodied agents. It incorporates a general interface for 3D transformations, with implementation of both matrix and quaternions, as well as a set of interfaces for hierarchical system of 3D objects called bones. A single bone represents in fact just a transformation, capable to operate in hierarchy. KINE+ makes it easy to move and rotate various parts of the agent's body. Sophisticated agents, like virtual humans, may contain more than 50 bones

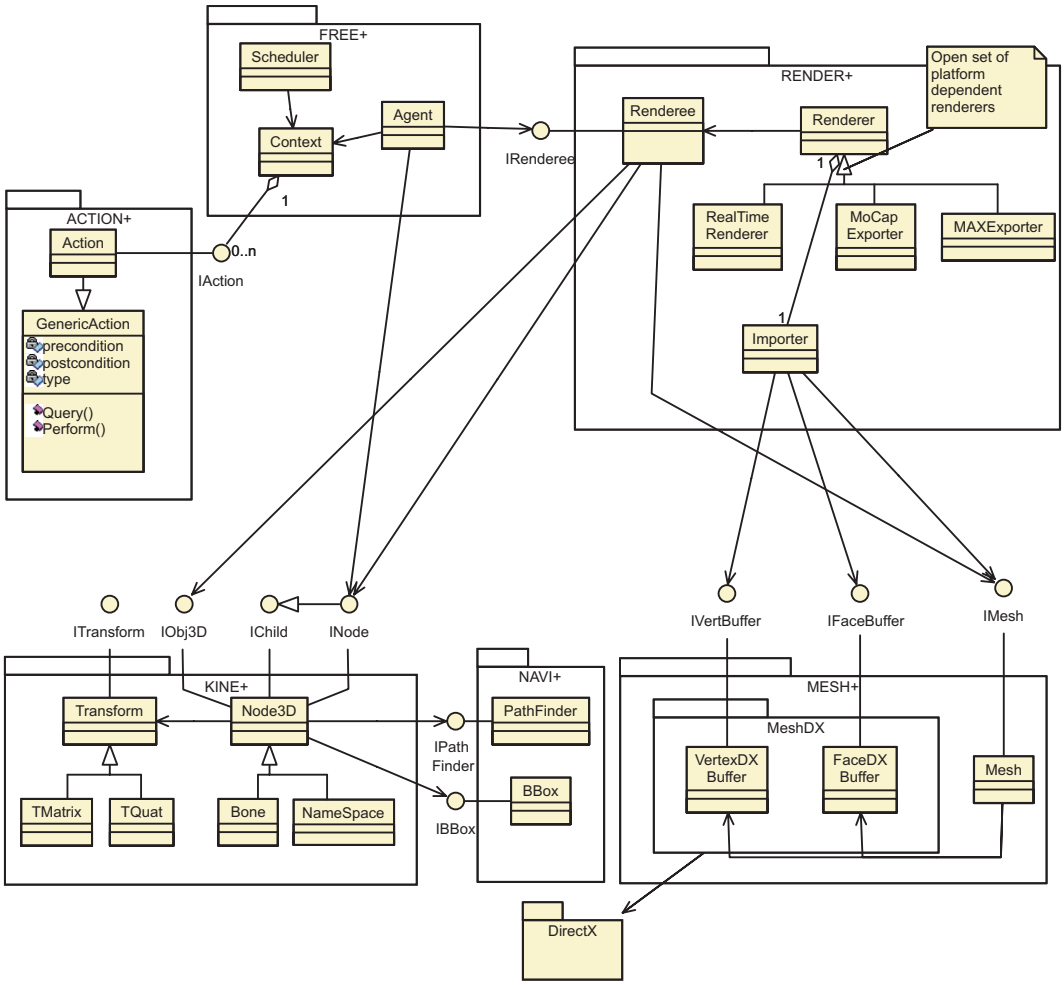

Fig. 3. An UML diagram of the FreeWill+ system construction.

in a highly hierarchical structure, while inanimate objects (represented by smart object agents) usually contain just single bones, which determine their position and orientation in space. The KINE+ module is platform-independent.

- **NAVI+** is a small component strictly co-operating with KINE+ models to support navigation in 3D space. It covers implementation of collision avoidance (based on OBB algorithm - *oriented bounding boxes*) as well as A* algorithm for path-finding.
- **RENDER+** module is another platform-dependent component that incorporates a mesh and a skeletal model into a single object enabled to render (*a renderee*), and thus constitutes a complete visual representation of an agent. Single mesh instances may be shared by numerous agents, while each agent should use its individual skeleton. Additional features like lighting and cameras are also supported. The module currently contains an implementation of an instant (real-time) renderer for DirectX compatible graphics accelerators. As an alternative, the animation may be streamed to the output file, which may be written in BIOVISION Motion Capture standard format ()

or 3DStudioMAX Script format (3ds max URL, 2004). The generated files are easy to be imported by a majority of commercially available 3D graphics packages for off-line rendering. The RENDER+ co-operates also with additional tools that make it possible to import 3D models from the external packages. Only plug-ins for the 3DStudioMAX are currently available.

- **FREE+** module is a surprisingly small component acting as a kernel for the whole FreeWill+ framework. It provides a cooperative environment and context for the agents and is capable of determining which agent is best suited for a given task. It also co-ordinates (schedules) the messages sent between them, including an event-like queuing and both individual and broadcasted messages. Messages are delivered in respect to temporal and spatial constrains (Bandini, Manzoni and Simone, 2002), and finally are subject to filtering. It is to ensure that the whole engine efficiently simulates natural perceiving processes. This is also the module where the *Agent* main class is defined, however in FreeWill+ almost all agent functionality is delegeted to the action components described below.
- **ACTION+** is the main, largest and most complicated part of the whole framework. Its general design principles - multi-layering and heterogeneity - have been discussed in the previous section. The module consists of an open set of highly autonomous components called actions, that are free to be used by agents, an that are capable to construct instant, hierarchical structures to fulfil their (or agents') goals. Majority of the system activity related to performing various animated tasks is supported by the actions. Although action layering is achieved ad-hoc during the run time, some levels still may be distinguished. These are:
  - Forward Kinematics Actions realising simple motions of the body's skeleton by directly setting the angles between bones in the joints.
  - Inverse Kinematics Actions that make it possible to move parts of the body just by imposing the target spatial position and orientation.
  - Behaviour Pattern Actions relate to behaviours that are well described algorithmically. Examples are walking (Boulic, Glardon and Thalmann, 2003) and grasping (Nebel, 2000; Kallman et al., 2003). They make use of forward and inverse kinematics actions.
  - Navigation Actions use mainly the walking behaviour pattern to roam around the digital surroundings, with extensive use of NAVI+ module for collision avoidance and path finding.
  - Intelligent Actions are a general term for high-level actions that usually apply AI techniques to perform. The first action of this category to be implemented was a Belief-Desire-Intention general model encapsulation.
  - Learnt Actions are distinguished subcategory of intelligent actions that apply machine learning algorithms, refer to section 5 for an example.

  The actual set of actions is open and currently under development.

FreeWill+ implementation is based on Microsoft COM (Component Object Model), with all the major modules and components, including the actions,

being COM in-process servers. This approach facilitates creation and use of uniform interfaces, gives also a standard implementation platform that may be easily extended by independent developers. Potentially COM-based architecture gives a chance to distribute the solution over a local network, however such a possibility has not been explored yet.

5 How Animated Agents can learn Actions

Machine Learning techniques offer interesting ways of constructing new actions to improve, automatically, the *repertoire* of agent behaviours. In particular, Reinforcement Learning methods (Sutton and Barto, 1998; Mitchell, 1997) appear most suitable for control of motion tasks. This is the case for those tasks that can be easily formulated as a sequence of movements necessary to fulfil a given goal. Examples might include approaching objects, grabbing and manipulating items or moving about in a virtual room. The current implementation of the FreeWill+ system (Szarowicz et al., 2003; Szarowicz, Mittmann and Francik, 2004) includes both the deterministic and non-deterministic *Q-learning* algorithms (Watkins, 1989; Mitchell, 1997). The objective of *Q-learning* is to find an optimal solution to a hard problem by learning how to act in a state space. Quality values (*Q-values*) are associated with each state-action pair. At present FreeWill+ has been tested on simple examples of actions in a discretized problem space. At each time step, an agent observes the state s_t, and takes action a. The choice of actions in early stages is usually chosen at random (any action may be selected from the possible actions set), while, as the agent becomes more *informed* about its surroundings, the choice becomes more guided/controlled and actions which give higher rewards are preferred. This human-like selection strategy forms what is usually called the policy of the agent, for a specific task, and the policy is a result of a controlled balance between exploitation and exploration. After executing an action the agent receives a reward r which depends on the new state s_{t+1}. The reward is usually discounted into the future, that is rewards received n time steps into the future are worth less by a factor $\gamma_n < 1$ than rewards received in the present. The deterministic algorithm uses the following update:

$$Q(s_t, a_t) \longleftarrow r_{t+1} + \gamma \max_a Q(s_{t+1}, a)$$

where $Q(s, a)$ is the quality table (*Q-table*) entry for state s and action a, s_t is the state visited at time t and s_{t+1} is the state resulting from executing action a in state s_t, r is the reward received by the agent and γ is a discount factor expressing the agent decreasing commitment on the outcome of future actions.

In case of a non deterministic solution the environment is subject to unexpected changes and both the actions and rewards of the agents may be non-deterministic. The algorithm uses the following update:

$$Q_n(s_t, a_t) = (1 - \alpha_n)Q(s_t, a_t) + \alpha_n[r_{t+1} + \gamma \max_a Q(s_{t+1}, a)]$$

where is α_n defined as:

$$\alpha_n = \frac{1}{1 + visits_n(s, a)}$$

s and a are respectively the state and action updated during n^{th} iteration and $visits_n(s, a)$ is the total number of times this state-action pair has been visited up to and including the n^{th} iteration. The non-deterministic update is very interesting because it gives an agent the possibility to learn even when the world is not entirely predictable: this makes the animated agent more realistic. Both its own actions and the world might not react as the agent would like to. For instance, a movement of an arm of an anthropomorphic agent might be slightly or grossly different from what the mind-body command wanted.

The training phase, during which an agent learns a policy, entails *rehearsing* a task or set of tasks in a number of iterations, also called epochs. Each iteration terminates when the agent reaches a goal state and receives a reward (negative or positive), or perhaps, when the agent gets completely lost in local minima for a long time (temporal thresholds can be devised to limit the training phase computations). Positive rewards are assigned to the agent when it successfully completes the task at hand. Negative rewards are assigned to the agent whenever it collides with an obstacle or outstretches a joint (hand moved beyond the legal boundaries defined by the state space, corresponding to a bio-mechanical physical model).

FreeWill+ embeds a number of tasks that can be learnt using the Q-learning technique. The teapot-lifting task is one of them (see Szarowicz et al. (2003); Szarowicz and Remagnino (2004) for another experiment results). The deterministic update was employed to learn the action for two control modes, forward (FK) and inverse kinematics (IK). The non-deterministic update was simulated using the inverse kinematics mode of control.

5.1 Learning using the deterministic algorithm

An example is shown in this section to introduce the idea of learning simple tasks. The task of the agent is to lift a teapot (z represents the co-ordinate of the teapot height that has to be increased). The elementary movements

available to the agent were selected from shown in Figure 4 and combined by example to solve the task. All available movements are graphically depicted in Figure 5. For the FK these were actions 1 – 5, and in this mode of control two experiments were conducted.

Forward kinematics control	**Inverse kinematics control**
1. Rotate arm up/down by Δalpha	1. Move palm by (Δx, Δy, Δz)
2. Rotate arm forward/backward Δalpha	
3. Rotate forearm by Δalpha	
4. Rotate hand along Z axis by Δalpha	
5. Perform the grabbing action	2. Perform the grabbing action

Fig. 4. Low-level actions used to train the avatar

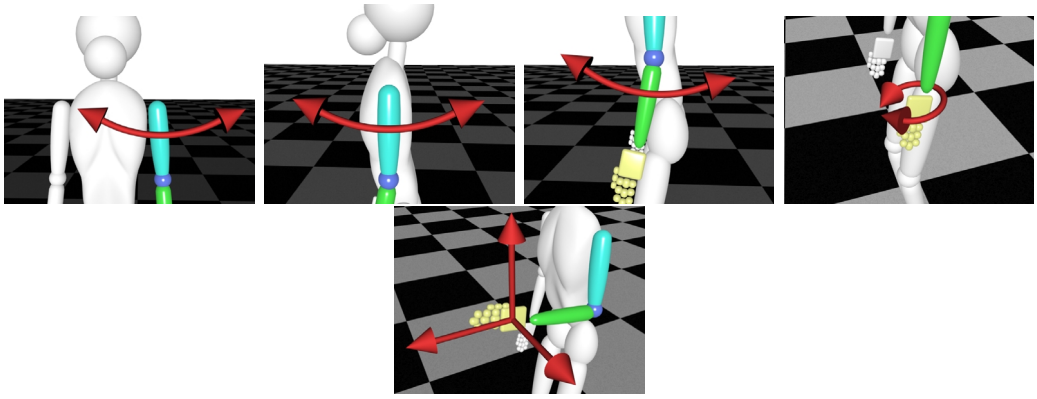

Fig. 5. Low-level actions used to train the avatar

The learning parameters include $\Delta\alpha$ set to 20 degrees in Experiment 1 and to 10 degrees in Experiment 2, while γ was set to 0.95. The difference between Experiment 1 and 2 is not in the size of the state-space but its sampling. The space axes were sampled more densely and thus the state space size of the second task results larger. Therefore, the state-space in the Experiment 1 consists of about 13000 states, while it is ten times larger in the Experiment 2 (121 000 states). The dimensionality of both tasks equals 5 - 2 degrees of freedom for the left arm, 1 for the left forearm, 1 for hand rotation and 1 for the state of the teapot. The respective size of these dimensions were 7, 12, 11, 7, 2 for the Experiment 1 and 12, 20, 18, 14, 2 for the Experiment 2. Ten simple actions were available to the agent at each time step (2 for each state space dimension, ie rotation or motion in 2 opposite directions for each degree of freedom). The minimum number of iterations for the Experiment 1 was about 4 000 and 20 500 for the second one. Thus, despite the tenfold growth of the state-space, the minimum number of iterations increases by a factor of about 5. The lengths of the best solutions are respectively 9 and 14.

	D			**ND**
Experiment (TeaPot)	FK-1	FK-2	IK	IK
State space	12936 (**5D**)	120960 (**5D**)	2240 (**4D**)	2240 (**4D**)
Actions per state	10	10	8	8
Min. no of iterations to find a solution	4000	20500	800	500
Approx. convergence	15000	80000	3000	800
Shortest sequence found	9	14	10	10

Fig. 6. Summary of the learning experiment (deterministic algorithm)

Similarly an experiment with biped control using inverse kinematics was also conducted. The simple movements available to the agent in this case are 1 and 2 shown in Fig. 4 (inverse kinematics column), and $\Delta x = \Delta y = \Delta z = 8cm$ for the motion of a hand, $\gamma = 0.95$. Therefore the state-space was 4-dimensional and the agent could choose from 8 simple actions - hand motion along 3 spatial axes in two opposite directions for each axis plus the grabbing/releasing action. The total size of state space was 2240 (8*14*10*2). The algorithm needs about 800 iterations to find a solution, and the best solution found is 10 actions long.

The number of states along 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 constraints: forearm can only rotate by about 180 degrees around the x-axis, arm 270 degrees 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 forearm rotation -20 degrees and 200 degrees would be the forbidden states). Finally in all experiments the *Q-table* was represented as a lookup table and the values were initialized to 0 before the simulation.

Results of the experiment are summarized in Fig. 6. Convergence graphs for the teapot simulation in each experiment are depicted in Fig. 7 and 8. They represent a sum of all Q-values as a function of epochs.

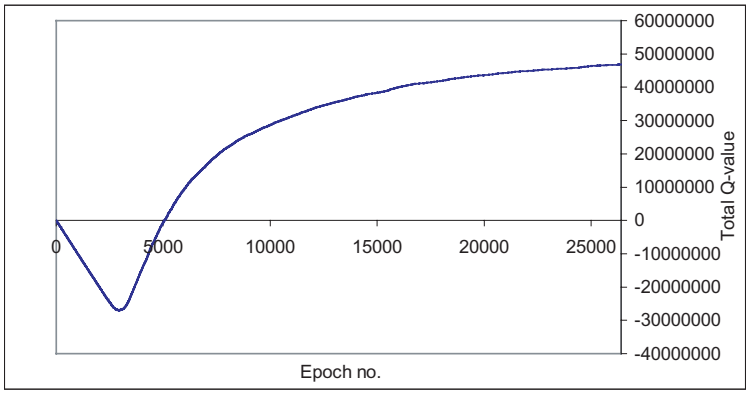

Fig. 7. Convergence for the FK teapot deterministic problem (experiment 1)

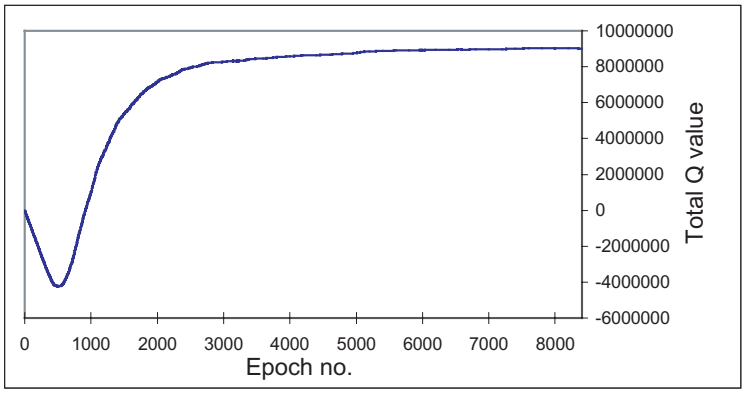

Fig. 8. Convergence for the IK teapot deterministic problem

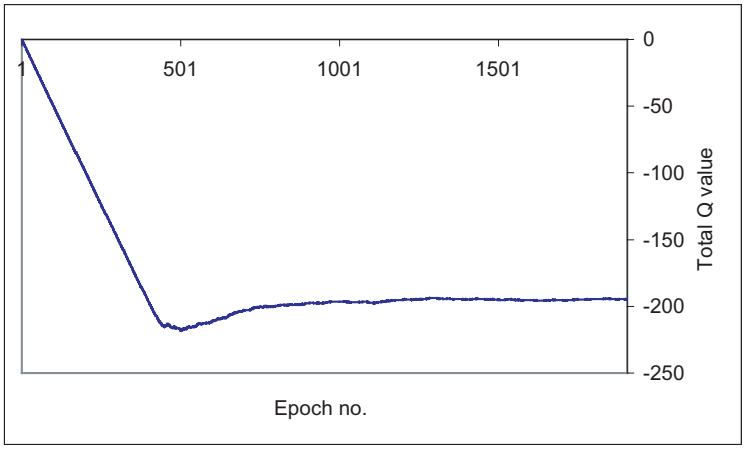

Fig. 9. Convergence for the IK teapot non-deterministic problem

5.2 Learning using the non-deterministic algorithm

The teapot task was also implemented using inverse kinematics control and the non-deterministic updating. The state space is the same as in the determinis-

tic implementation, and the length of the shortest solution is also 10 simple actions. The convergence was reached faster - in approximately 800 iterations as opposed to about 3000 in the deterministic case and the time necessary to reach the optimum solution was shorter as well - about 90 minutes on average (550 iterations). The convergence is also more stable (Fig. 9). This suggests that the non-deterministic version of the algorithm generates comparable results in a shorter amount of time. An example resulting animation sequence is presented in Fig. 10.

5.3 Learnt Actions in the FreeWill+ Framework

The presented results confirm that learning can be embedded and the outcome of the testing performed with the FreeWill+ architecture is a good example. In this specific example the agent learns incrementally how to use a set of relatively low-level movements. It is envisaged that a number of more advanced experiments can be attempted to utilise more complex actions, for instance to enter a building opening doors and move objects around.

The Reinforcement Learning Actions (RLA) are embedded into the society of FreeWill+ action components, and thus may be triggered to work in their instant associations, as described in Subsection 3.2. Queried by a higher-level action component about the suitability to perform a given sub-task, RLA components check if they are able to transform that task into the desired final state in a discretized problem space, and to define this problem space itself. The latter problem appears even harder in practice: it has to be kept small enough to be computable. We have also imposed severe limitations on the repertoire of the available low-level actions: currently we use the set shown in Fig. 4, optionally completed with simple walking forward and backwards. If, despite all these limitations, the component is able to give a positive answer to the query, the action may be performed. If possible, the results of previously executed learning passes are reused, so RLA components do not necessarily perform actual learning each time when triggered to use.

Application of RLA has several pros and cons. They significantly shorten a usual multi-stage chain of associated actions: they use directly the lowest-level components, namely the FK and IK action components that control the animated body and its bones without an intermediation of any other system components. The obvious gain is that learnt actions are generic: no previous knowledge is needed to perform any (possible to formulate) action. In non-predictable or hardly predictable environments (e.g. massive crowd simulations) they behave more realistically and thus prove that our embedded intelligence indeed can cope with uncertainty in the actions and changes in the environment. An important issue is also that agents behaviour may be

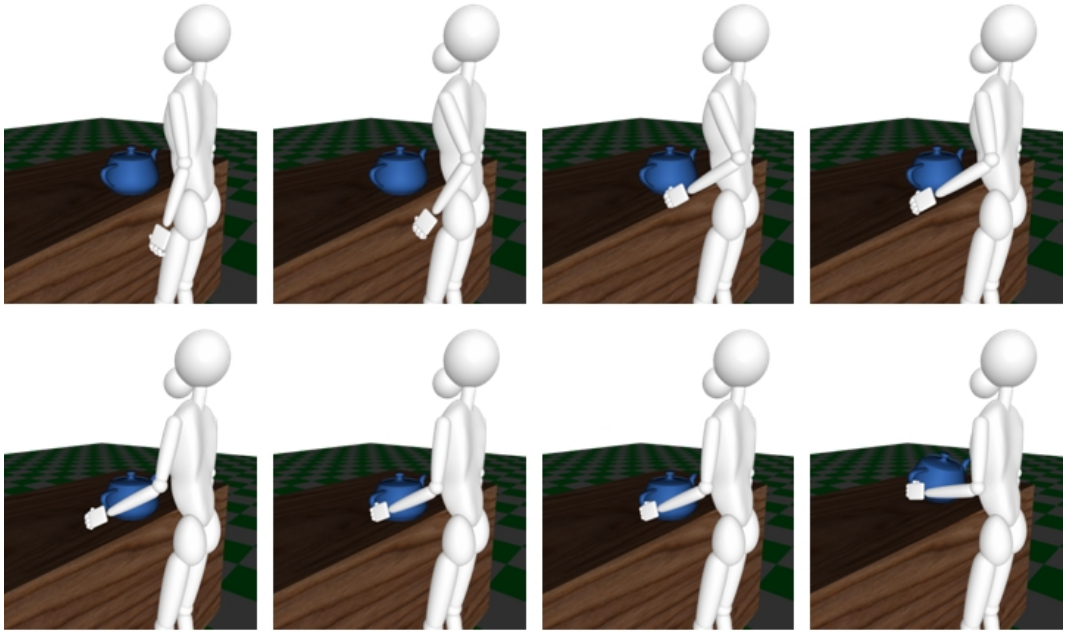

Fig. 10. A generated animation sequence

generated fully automatically.

The RLA components make it possible to successfully perform operations that are otherwise not known to the system intelligent control. Theoretically it could bring the solution's capabilities beyond any limitations. In practice it cannot go too far. The most important weakness is the lack of implementation of the physical rules; an example may be a task of jumping over an obstacle, which is impossible to learn with the RLA, at least until the component of physical dynamics is finished.

Another limitation is the efficiency. Learning takes a lot of time, therefore the RLA component is suitable to perform actions only in off-line animation mode.

6 Conclusion

Animators want their digital characters to be realistic, but also *alive*. Living creatures adapt themselves to a changing environment. The paper proposes an architecture for animated agents that makes possible the embedding of intelligence, ensures flexibility and diversity of behaviour and enables them to solve tasks and to learn. A principled design is put forward and an initial study on how learning can be embedded in animated agents.

Behaviour can be decoupled and made modular. Tasks requiring complex actions can be decomposed into simpler actions and can also be learnt. This makes a multi-layered structure of heterogeneous actions, with an essential additive of learning-capable components.

Learning techniques such as Reinforcement Learning lend themselves very well to the sought purpose: the automatic action acquisition with a minimal user intervention. Animations require an environment that can collaborate with the animated agents through its smart objects (we might call these *inanimats* as opposed to animats).The animated environment teaches FreeWill+ animats by rewarding positively or negatively according to how the task is or is not accomplished. These are the first steps towards autonomous animation engines that can learn how to act, behave and interact by following simple rules.

References

3DS Max® 6, product home page. Discreet®. http://www.discreet.com/3dsmax/

Allbeck, J. and N. Badler, Toward representing agent behaviors modified by personality and emotion. Workshop on Embodied Conversational Agents – Let's specify and evaluate them! at AAMAS 2002, Bologna, Italy.

Arafa, Y., K. Kamyab, E. Mamdani, S. Kshirsagar, N. Magnenat-Thalmann, A. Guye-Vuilleme and D. Thalmann, Two Approaches to Scripting Character Animation. Workshop on Embodied Conversational Agents – Let's specify and evaluate them! at AAMAS 2002, Bologna, Italy.

Bandini, S., S. Manzoni and C. Simone, Dealing with Space in Multi-Agent Systems: a model for Situated MAS. ACM AAMAS, Bologna, Italy 2002, pp. 1183-1193.

Boulic, R., P. Glardon and D. Thalmann, From Measurements to Model: the Walk Engine. Proc. of 6th Conf on Optical 3D Measurement Techniques. Zurich International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2003.

Bratman, M., Intention, Plans and Practical Reason. Harvard University Press, 1987.

Caicedo, A., J.-S. Monzani and D. Thalmann, Toward Life-Like Agents: Integrating Tasks, Verbal Communication And Behavioural Engines. The Virtual Reality Journal, Springer 2001.

Evans, R., Varieties of Learning in Steve Rabin (Ed.), AI Game Programming Wisdom, Charles River Media, Hingham, Ma, USA 2002, pp. 567-578.

Francik, J. and K. Trybicka-Francik, A Framework for Program Control of Animation of Human and Animal Characters. Studia Informatica, Vol. 24, No. 4 (56), 2003, pp. 55-65.

Francik, J. and P. Fabian, Animating Sign Language in the Real Time. 20th

IASTED International Multi-Conference Applied Informatics AI 2002, Innsbruck, Austria 2002, pp. 276-281.

Franklin, S. and A. Graesser, Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages. Budapest, Hungary, 1996, pp. 193-206.

Gleicher, Michael. Making Motion Capture Useful. SIGGRAPH'01 Course. Available on-line at http://online.cs.nps.navy.mil/DistanceEducation/ /online.siggraph.org/2001/Courses/cd2/courses/51, accessed on 14/02/2004.

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

Jennings, N. R., An agent-based approach for building complex software systems. Communications of the ACM, 44(4), 2001, pp. 35-41.

Jennings, N. R. and M. J. Woolridge, Applications of Intelligent Agents, in N. R. Jennings and M. J. Woolridge (eds.) Agent Technology - Foundations, Applications and Markets. Springer Verlag, 1998, pp. 3-28.

Kallmann, M., Object Interaction in Real-Time Virtual Environments. PhD Thesis, École Polytechnique Fédérale de Lausanne, Switzerland, 2001.

Kallmann, M., A. Aubel, T. Abaci and D. Thalmann, Planning Collision-Free Reaching Motions for Interactive Object Manipulation and Grasping. Eurographics 2003.

Nebel, J.-C., Realistic collision avoidance of upper limbs based on neuroscience models. Proceedings of Eurographics 2000, Vol. 19, issue 3, Interlaken, Switzerland, 2000.

Koeppel, D., Massive Attack. Popular Science (2002), accessed from Internet on 9/02/2004, http://www.popsci.com/popsci/science/article/0,12543,390918-1,00.html

Mitchell, T., Machine Learning, McGraw Hill, 1997.

Monzani, J.-S., A. Caicedo and D. Thalmann, Integrating Behavioural Animation Techniques. Proceedings of Eurographics 2001, vol. 20, issue 3, Manchester, UK, 2001.

Monzani, J.-S., An architecture for the Behavioural Animation of Virtual Humans. PhD Thesis, Ecole Polytechnique Fdrale de Lausanne, 2002.

O'Sullivan, C., J. Cassell, H. Vilhjalmsson, J. Dingliana, S. Dobbyn, B. MacNamee, C. Peters and T. Giang, Levels of Detail for Crowds and Groups. Computer Graphics Forum, 21(4) 2002 pp. 733-742.

Rick Parent. Computer Animation, Algorithmes and Techniques. Morgan Kaufmann Publishers, San Francisco 2002.

Raupp Musse, S. and D. Thalmann, Hierarchical Model for Real Time Simulation of Virtual Human Crowds. IEEE Transactions on Visualization and Computer Graphics, V. 7, N.2 2001 pp. 152-164.

Rao, A. S. and M. P. Georgeff, Modeling Rational Agents within a BDI-Architecture. Proceedings of the 2nd International Conference on Principles

of Knowledge Representation and Reasoning. Cambridge, MA, USA, 1991, pp. 473-484.

Reynolds, C. W., Flocks, herds, and schools: A distributed behavioral model. Computer Graphics, SIGGRAPH '87 Conference Proceedings, vol. 21(4), ACM SIGGRAPH 1987 pp. 25-34.

Russell, K. B. and B. M. Blumberg, Behaviour-Friendly Graphics. Computer Graphics International. 1999.

Russell, S. and P. Norvig, Artificial Intelligence. A Modern Approach. Prentice Hall, 1995.

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

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

Szarowicz, A., M. Mittmann, J. Francik and P. Remagnino, Automatic Acquisition of Actions for Animated Agents. Proc. of 4th International Conference on Intelligent Games and Simulation IJICS London 2003 pp. 170-174.

Szarowicz, A., M. Mittmann and J. Francik, Intelligent Action Acquisition for Animated Learning Agents, in: Lakhmi C. Jain (Ed.) Learning Coordination and Communication in MultiAgent Systems, Theory and Applications. World Scientific, to appear in 2004.

Szarowicz A. and Remagnino P., Avatars That Learn How to Behave. European Conference on Artificial Intelligence ECAI 2004, Springer, Valencia, Spain, 2004.

Tu, X. and D. Terzopoulos, Artificial Fishes: Physics, Locomotion, Perception and Behavior. Proc. of SIGGRAPH'94, Computer Graphics, Vol. 28, Annual Conf. Series 1994, pp. 43-50.

Tu, X., Artificial Animals for Computer Animation: Biomechanics, Locomotion, Perception and Behavior. LNCS 1635. Springer Verlag, Berlin 1999.

Watkins, C. J. C. H., Learning from Delayed Rewards. PhD thesis, University of Cambridge, Psychology Department, 1989.

Winikoff, M., L. Padgham, and J. Harland, Simplifying the Development of Intelligent Agents. Advances in Artificial Intelligence. 14th Australian Joint Conference on Artificial Intelligence AI2001, LNAI 2256, Adelaide, Australia 2001, pp. 557-568.