Artificial Intelligence for Animated Autonomous Agents

Adam Szarowicz

A thesis submitted in partial fulfillment of the
requirements of Kingston University
for the degree of Doctor of Philosophy

School of Computing and Information Systems
Kingston University

June, 2004

Acknowledgements

Finished at last! The viva was successful and all I have to do now is to write the acknowledgements.

I would like to thank my princess Ewa for all she did to help me. Special thanks are due to my Parents – Tadeusz and Janina Szarowicz and my sister Ewa. It would not have been possible without you! You were with me in times of trouble, in my crises and successes, always ready to pick up the phone. Big thank you and dziękuję.

I would like to acknowledge Dr Peter Forte who invited me to do this project and who was supervising me for two years. Peter has been the best supervisor and director one could imagine.

I am very grateful to my director of studies – Dr Paolo Remagnino, thank you for your support, advice, patience and sense of humour! It was not easy but I think I have eventually managed to convince you that one can defend a PhD in animation which, to make things worse, was written using UML!

Thanks go to my examiners prof. Norman Gough, prof. Wojciechowski and dr Sarah Barman and members of staff at our School – Petros Gelepithis, Souheil Khaddaj, James Orwell, Sergio Velastin, Ruediger Oehlmann and Graeme Jones who contributed to the successful completion of my degree. I am also grateful to prof. Stanislaw Kozielski from the Silesian University of Technology and prof. Jan Zabrodzki from Warsaw University of Technology. I am especially grateful to Jarek Francik for his most valuable comments, for his sleepless nights spent on writing these comments and proof-reading the draft. Warmest thanks go to his wife Kasia.

In a special way I would like to thank the members of our prayer group. Thank you Julius for your support and time, especially in moments of trouble, thank you Ela for feeding my ever-hungry stomach. Thank you Agnieszka for all you did to support me. Big thank you to all my English, French, German, Greek, Polish, Russian, Serb, Spanish, Srilankan, Swedish, ..., friends.

Abstract

Automatic creation of animated crowd scenes involving multiple interacting characters is currently a field of extensive research. This is because automatic generation of animation finds immediate applications in film post-production and special effects, computer games or event simulation in crowded areas.

The work presented here addresses the problem of inadequate application of AI techniques in current animation research. The thesis presents a survey of different industrial and academic approaches and a number of lacking features are identified. After extensive research in existing systems an agent-based system and an animation framework are chosen for extension and the cognitive architecture FreeWill is proposed. The architecture further extends its underlying principles and combines software agent solutions with typical animation elements. It also allows for easy integration with existing tools.

Another important contribution of FreeWill is a proposal of an algorithm for automatic generation of high-level actions using reinforcement learning. The chosen learning technique lends itself well to the animation task, as reinforcement learning allows for easy definition of the learning task – only the ultimate goal of the learning agent must be defined. The process of defining and conducting the learning task is explained in detail. The learning module allows for further automation of the process of animation generation, shortens the task of creating crowd scenes and also reduces the production costs. Newly learnt actions can be applied to increase the quality of the generated sequences.

The learning module can be used in both deterministic and non-deterministic environments. Experiments in both modes are presented, and conclusions are drawn. Two modes of control – inverse and forward kinematics are also compared and a number of experiments are demonstrated. Learning with inverse kinematics control was found to converge faster for the same task.

A working prototype of the architecture is presented and the learnt motion is compared with human motion. The results of the comparison demonstrate that the learning scheme could be used to imitate human motion in crowd scenes. Finally a number of metrics are defined which allow for easy selection of most relevant actions from the learnt set, again helping to automate the process.

The work concludes with pointing out further directions of research based on this work and suggests possible extensions and applications.

Table of Contents

List of Figures

List of Tables

Glossary

Agent – there is no agreement in the scientific community how to define the term. Researchers usually adopt definitions which are affected by the context of their work on agent systems. One of the most general formulations of the term was proposed by Russel and Norvig (1995) as "anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors". This definition does not imply that agents should be autonomous as suggested by Wooldridge and Jennings (1995), and they characterise an **autonomous agent** as a piece of hardware or (more usually) software-based computer system that enjoys 4 properties:

- autonomy: agents operate without the direct intervention of humans or other systems, and have some kind of control over their actions and internal state;
- reactivity: agents perceive their environment and respond in a timely fashion to changes that occur in it;
- pro-activeness: agents do not simply act in response to their environment, they are able to exhibit goal-directed behaviour by taking the initiative
- social ability: agents interact with other agents via some kind of agent-communication language;

A survey of agent definitions has been published by Franklin and Graesser (1996), who also proposed a more general definition of autonomous agents which we shall adopt in this thesis: "An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future."

According to this definition, the agents must be *reactive* (respond in a timely fashion to changes in the environment – sense and act), *autonomous* (exercise control over their actions), goal-oriented (*pro-active*), which means they can not simply act in response to the environment. Finally they must be represented as a continuously running process.

In the context of this work the agents will be visually represented as **autonomous characters** and they will persist in a virtual landscape. Reynolds (1999) calls such agents **virtual characters**, they "represent a character in a story or game and have some ability to improvise their actions". He also defines a **virtual agent** denoting a real agent in a virtual world. Thus his autonomous characters are situated (embedded in a world containing other entities), embodied (having some kind of a physical representation),

reactive and virtual.

Multi-agent system – is a system consisting of many independent agents, each of which can perceive and act autonomously. Additionally agents can affect each other and influence decisions made by others.

Animation – fast projection of a sequence of images (called frames of the animation), which gradually change over time. Because the changes in two consecutive frames are too small for the human eye to notice in a given amount of time, an illusion of a continuous, smooth motion is created (after Francik, 1999). In **computer animation** the generation of images is performed by specialised software and the motion of objects between two frames is often calculated according to a set of mathematical formulas. Hodgins *et al* (1999) define animation as

> "Animation is the production of consecutive images, which, when displayed, convey a feeling of motion. Animated images are almost magical in their ability to capture our imagination. By telling a compelling story, astounding with special effects, or mesmerizing with abstract motion, animation can infuse a sequence of inert images with the illusion of motion and life. Creating this illusion, either by hand or with the assistance of computer software, is not easy. Each individual image, or frame, in the animated sequence must blend seamlessly with the other images to create smooth and continuous motion that flows through time."

Architecture – is a fixed structure that realises a certain system (after Newell, 1990).

Articulated figure – it is a figure or character which consists of a number of objects called links which are connected by joints. Joints impose constraints on the way the two connected objects can move and rotate in relation to each other.

Avatar – in the context of computer animation avatar is a graphical representation of a character or human user in virtual environments. In this work avatar will be used interchangeably with character, depicting a visible virtual creature together with its agent aspect, which supplies the AI capabilities.

Collision detection and **collision avoidance** – collision detection is a process allowing to establish whether collision between objects in the scene is about to or has already

taken place. Collision avoidance is a set of techniques manipulating the motion or behaviour of objects and agents in the scene allowing them to prevent imminent collisions. The simplest form of collision avoidance would be stopping moving objects.

Crowd scene – is a virtual scene with multiple participating virtual characters.

Embedded agent – an agent which exists in some environment as opposed to an agent existing in isolation.

Embodied agent – is an agent which has some form of manifestation (a body), agents which are not embodied would be called abstract. Virtual characters are often seen as embodied autonomous agents.

Flocking and **schooling** – is a form of self-organising behaviour often exhibited by groups of animals (birds, fish, insects) whereby the members of the flock tend to exist in close proximity to other members of the group. Reynolds (1987) proposed the first computer model of flocking, where he proposed a number of simple rules constituting the flocking behaviour.

Keyframed animation – a way of creating animation in which only the main frames of the sequence are defined and software interpolates between them to create continuous motion.

Kinematics – the science of motion, where motion is treated without considering the forces which cause it. If motion of the end-effector is solved as a function of the joint angles we deal with Forward Kinematics (FK), if the pose of the end-effector is known and the motion is generated by calculating the joint angles we deal with Inverse Kinematics (IK), Figure 1. Current professional animation packages implement inverse kinematics algorithms allowing the artists to only define key poses of the animated figure (cf. Craig, 1986, Maestri, 1999, Johnson *et al*, 2000, Faloutsos *et al*, 2001, Francik and Fabian, 2002).

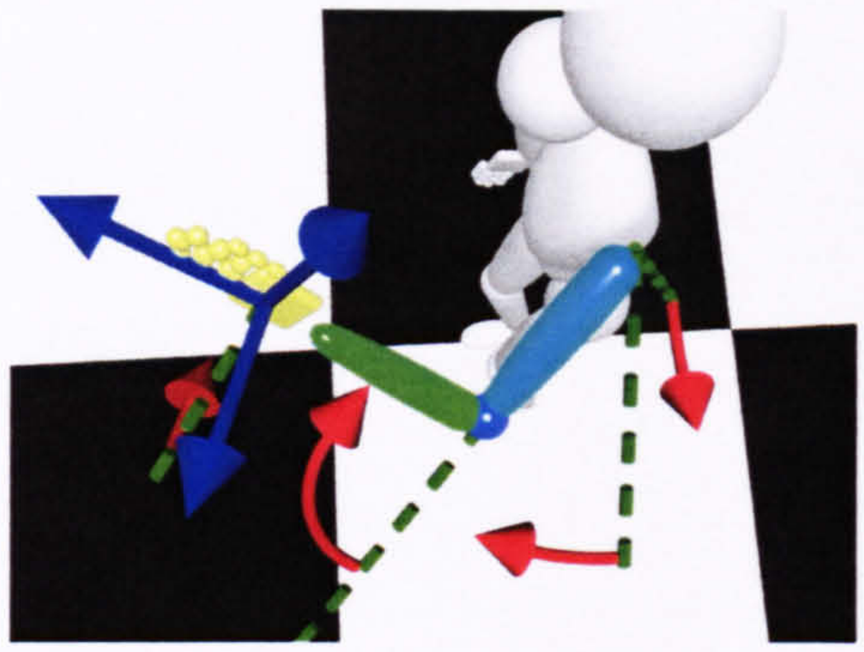


Figure 1 Position of the end effector can easily be calculated when all joint rotations are given (FK), the opposite task is the problem of IK

Machine learning – is a research field concentrating on building computer systems that can adapt to the environment with experience. The main branches of ML include Supervised Learning where a supervisor trains the system by first supplying correct answers to a number of tasks, **Reinforcement Learning** also known as learning by trial and error, where some form of verification is given upon finishing the task and Unsupervised Learning.

Self-awareness – in the context of this work self-awareness will be defined as ability of autonomous characters to perceive the surrounding environment and distinguish themselves from other characters and objects in the virtual scene.

UML – Unified Modelling Language, a graphical modelling tool which supports the design of complex object-oriented systems.

Chapter 1 – Introduction

Automatic creation of crowd scenes with many animated virtual characters is an ambitious challenge for the modern film postproduction industry. The current process, conducted by skilled animators and directors, is not only labour-intensive but also tedious and expensive. With the dynamic growth of computer-based techniques film creators and special effect designers are looking with increasing expectation towards the computer animation research community.

Figure 2 presents shots from two popular film productions – *"The Lord of the Rings"* and "*Gladiator*". In these and similar other films it was necessary to create scenes with many human characters performing simple actions in the background.

Figure 2 Shots from films where crowd scenes had been added digitally; left) Digital army (background riders) going into battle in The Return of the King [from Comingsoon, 2004]; right) background – soldiers and crowds in the ancient Rome in Gladiator [from Spielberg-dreamworks, 2003]

Even though the creation of those scenes has been extensively supported by modern computer equipment and sophisticated software packages, the most important part of the work still involved a substantial amount of human input. First of all in most cases the real human motion must either be captured and digitised or inputted manually using a technique called "keyframing" in which the animator defines all key poses of the animated objects. The first approach requires employing real actors and expensive hardware and software. Then the digitised motion must be labelled, edited and pasted into the digital scenes. In many cases the process of editing relies on frame-by-frame corrections and manual animation. Finally it is often necessary to reiterate the process, as the missing motion sequences cannot be generated by any other means. The

keyframing-based approach on the other hand is extremely time-consuming and can only be performed by skilled artists.

A much bigger progress might be made if such scenes could be created entirely by an intelligent system. For example, such a system would allow the animator to define a number of interacting characters and then supply a set of parameters and variables such as character goals, level of aggression or a list of desirable actions. The simulation would then be started, effortlessly generating the scene. Such an approach relying on moving the focus of animation to a higher level by changing the emphasis from animating to directing would significantly improve the capabilities of current animation systems.

Tools capable of delivering the functionality described above must obviously consist of many components. Figure 3 presents the most important elements of a package for automatic generation of crowd scenes.

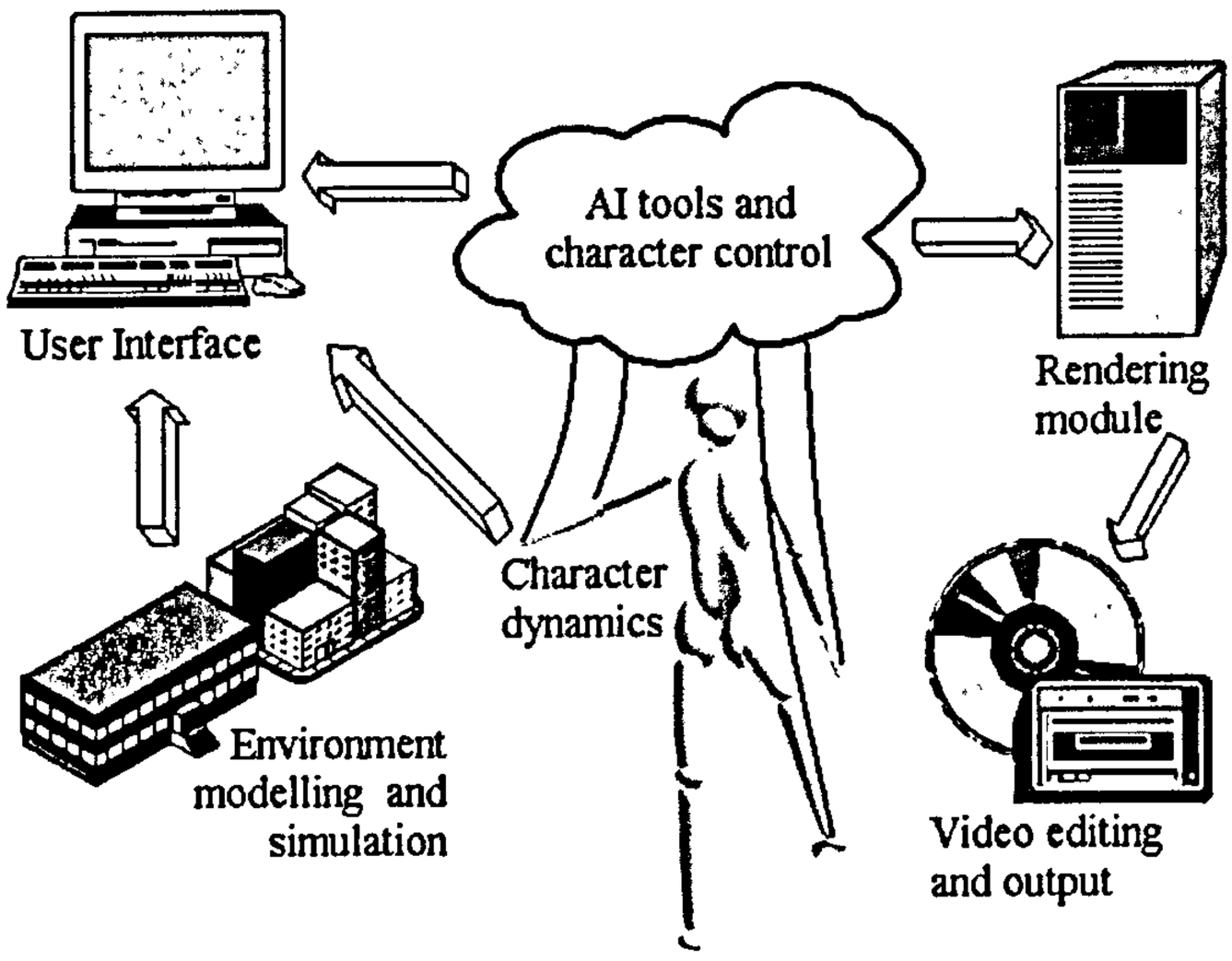


Figure 3 Elements of a crowd simulation tool

First of all, the scene itself must be modelled and simulated. This serves the purpose of providing a virtual environment for the artificial characters. Additionally, the characters, which populate the scene, must be represented in some graphical form; it would also be desirable to provide them with a set of physical constraints and some information on the dynamics of their motion. The most important part of a virtual character would be the delivery of navigation and behavioural modules, to allow the character to intelligently

move in the environment. It would also be necessary to endow it with other cognitive capabilities, that is capabilities pertaining to the character's mental capacities, including its internal world model, ability to learn and make decisions as to what actions to take. That functionality could be referred to as the character's intelligence and would ideally deliver sufficient complexity to limit the character creation process. The animation could thus be defined as a process of declaring the character goals, which are then achieved automatically using techniques borrowed from the domain of Artificial Intelligence (AI). These are the issues, which this work is trying to address.

The remaining components of the full system are a standard user interface and the rendering unit. The user interface allows the creator to work efficiently without the necessity to know the intricacies of the system's design. When the desired motion has been generated, a separate module must render the scene – that is create a sequence of images including the characters, scene objects, textured surfaces, lights, reflexes, shadows, weather artefacts such as rain and fog and other important elements. The last stage of the process is then the compositing and editing of the final video or film.

It is clear that a system capable of delivering all the required functionality would be a complex one. There is nonetheless a range of professional software tools, usually called professional 3D animation packages, which can deliver most of the elements depicted in Figure 3. However, whereas they can provide sophisticated graphics capabilities and complex user interfaces, they fall short on tasks related to Artificial Intelligence. Virtual characters lack self-awareness and autonomy, the concept of learning is not present in any of the industrial systems, and even such simple tasks as collision detection may pose problems.

This thesis addresses problems related to the building of the 'AI tools and character control' component as shown in Figure 3. We propose and implement an extendable cognitive architecture (Funge *et al*, 1999, Funge, 2000) designed to accommodate goals, actions, knowledge and beliefs, based on the latest developments in AI, in particular, the concept of autonomous agent (Jennings, 2001, Winikoff *et al*, 2001). This will, in turn, enhance the ability of animated characters to display autonomous intelligent behaviour. The following thesis is being formulated:

> Application of an intelligent architecture based on the existing techniques from Artificial Intelligence and using available professional 3D animation packages may significantly support automatic creation of animated sequences with many virtual characters.

The goals of the PhD can be formulated as follows:

- Review of existing intelligent architectures, especially regarding systems for automatic crowd scene generation and supporting creation of autonomous characters
- Proposal of an architecture capable of representing actions, goals, knowledge, plans, sensing and beliefs
- Adaptation of the proposed architecture to existing tools, namely to allow co-operation with professional 3D animation packages (such as 3D Studio Max and Maya) and to provide automatic collision avoidance which is a feature lacking in most of the current solutions
- Construction of an underpinning strong software engineering framework, designed and documented in a modern modelling language to facilitate ease of implementation, extendibility and clarity of the proposal
- A study on automatic acquisition of new character actions (such as raising objects) using machine learning techniques
- Implementation of the prototype and presentation of results

A more general aim of the work is a construction of a tool supporting the work of professionals thus allowing moving the focus of the work on crowd generation to a higher level by changing the emphasis from animating to directing by allowing them to parameterise the system behaviour.

The work presented in this thesis has been divided in two stages. The first part includes proposal and implementation of a general cognitive architecture supporting automatic crowd scene generation. The system is then extended using one of the Reinforcement Learning techniques (Mitchell 1997, Sutton and Barto, 1998, Watkins, 1989, Watkins and Dayan 1992) thus allowing to address the concept of rapid animation prototyping (Dontcheva *et al*, 2003, Liu and Popovic, 2002, Bregler *et al*, 2002). This reduces the need of employing motion capture-based techniques (Gleicher, 2001, Vicon, 2001) for character animation. In order for the proposed architecture to be sufficiently flexible and general it must be designed following modern software engineering guidelines. This is

why system design uses the Unified Modelling Language (UML) (Object Management Group, 2003, Booch *et al*, 1999), which is a cross-disciplinary modelling language and a good design tool for an object-oriented implementation of the system (Booch, 1994, Coleman *et al*, 1994).

1.1 Methodology

The methodology of the project is based on the iterative incremental development, as suggested by the Rational Unified Process – RUP (RUP, 2003), documented in the Unified Modelling Language (UML), which is a visual modelling tool supporting design and development of software systems. The architecture is gradually refined following standard design-implement-deploy cycles, progressively adding new features and testing new ideas. First, the basic problem of obstacle avoidance was selected, as this is a feature lacking in current animation packages. The next step is the action extension, which allows the characters to perform proactive actions (that is actions executed in order to achieve a high-level goal). The conceptual framework is also enriched building on the ideas from both the animation and intelligent agent fields. At this stage the additional actions are prepared manually in the form of script. The characters are able to generate motion sequences including both reactive and proactive actions. Having achieved this, the question whether actions could be added automatically is examined more closely. As a result, a learning algorithm for automatic acquisition of so-called high-level actions is proposed. This extension broadens the possible character's action repertoire thus making the created scenes interesting to professionals. The key technique here is Reinforcement Learning. Using this technique generation of motion using both forward and inverse kinematics control is investigated.

1.2 Beneficiaries

As mentioned earlier, automatic generation of character animation would find immediate applications in film post-production and special effects for films requiring creation of scenes with many interacting characters in the background. Examples might include digital extras in cities and shopping centres but also, and probably more importantly, digital battle scenes and combat sequences. Apart from simplifying the task of animation creators, film directors and post-production professionals, the proposed extensions can substantially increase the capability and appeal of computer games by

allowing the animators to create more advanced scenes, improve quality of group representation, and allow for automatic crowd generation. Possible applications may also cover generation of believable scenes for city planning purposes, crowd generation for virtual reality and surveillance and various commercial presentations. Finally, simulations of the safe evacuation of crowded areas in the event of fire or other disasters could also be created with the presented tool.

1.3 Structure of the Thesis

The remainder of this work is divided as follows.

Chapter 2 introduces the techniques currently used for creation of special effects involving human crowds. It starts with the presentation of three most advanced commercial animation packages and a custom-based system for character animation called Massive. This is followed by a brief discussion of the motion capture technique. The chapter concludes with a short characterisation of UML.

Chapter 3 is an extensive survey of the field of automatic animation generation and cognitive architectures. The review comprises an insight into general cognitive architectures and a survey of animation-dedicated frameworks is presented including cognitive, physics-based and crowd systems. Some of the more interesting agent-oriented systems are also discussed in more detail. Finally the Reinforcement Learning technique is presented together with example applications.

Chapter 4 starts with discussion of possible benefits from superseding object-oriented programming with the more recent concept of agent-orientation. The FreeWill prototype combining agent-oriented and animation-based concepts is then presented. This is followed by a detailed description of the architecture presented in UML.

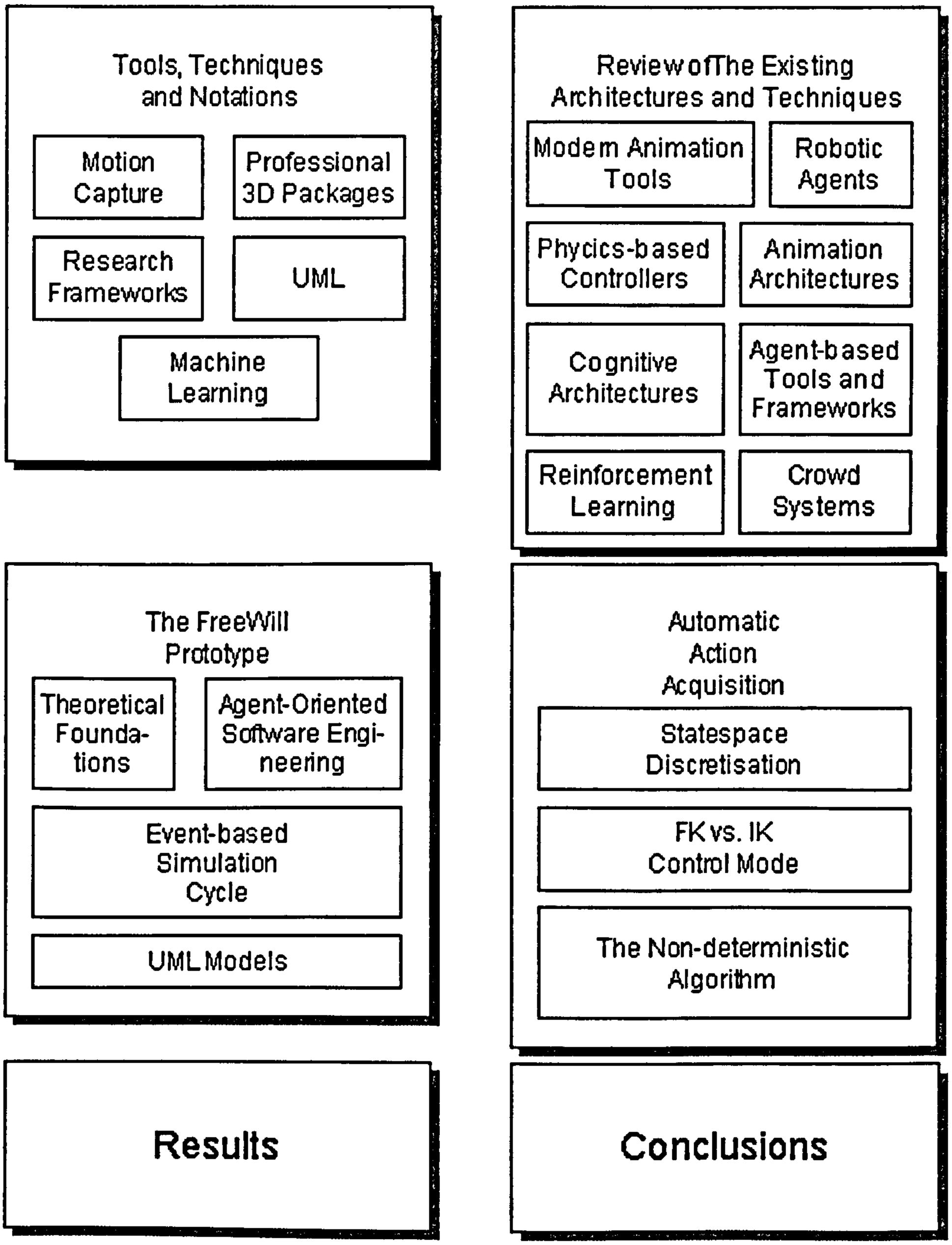
Chapter 5 complements the discussion of the architecture with a proposal of a learning module, which allows the avatars to acquire new actions through machine learning. Methods of motion control based on forward and inverse kinematics are also compared as well as two learning updates. Three metrics are proposed.

Chapter 6 presents results obtained from the system, including generation of animation and application of the learning techniques and metrics.

Chapter 7 concludes the work and presents possible future directions.

The structure of the thesis is graphically illustrated in Table 1

Table 1 Structure of the thesis



Chapter 2 – Tools and Techniques

Tools currently available on the market offer a set of features supporting automatic generation of motion for individual characters as well as crowds and flocks, thus trying to reduce the complexity of the task of animation generation (cf. Monzani *et al*, 2001). Examples of such tools may include the Character Studio plugin designed for 3D Studio Max or the Massive crowd simulation system described later. On the other hand there are already existing well-established techniques commonly employed by post-production studios. Those techniques rely mainly on motion capture systems and keyframed motion (Vicon, 2001, Kinetic Impulse, 2003). There is also a thriving branch of research dedicated to creation of intelligent creatures. This chapter overviews the current state of animation packages and industry standards.

2.4 Current Approaches To Character Animation

Professional 3D animation and modelling packages are currently used by production studios to create computer-generated special effects in film postproduction. This section presents capabilities and deficiencies of these packages, also including Massive, a custom built industrial system. In this review a special emphasis is placed on the ability of animation systems to create and animate intelligent characters, especially in the form of articulated bipeds, as all professional animation packages are able to animate swarms and flocks using particle systems (see Reynolds, 1987).

2.4.1 3D Studio Max, Maya and Softimage

Animation packages are tools used to create very complex and realistic images and animation sequences. They can be used to animate moving objects, fluids and are also often used to animate characters. One way of doing so is by manually creating virtual creatures modelled on top of a bone structure, dressing and skinning them and applying predefined meshes. The motion is then applied, which can be generated using one of the following techniques: motion capture and predefined motion, keyframed animation or particle systems. In the hands of an experienced user/animator these packages allow for creation of very impressive and indeed realistic scenes (see Figure 4). To extend the capabilities of the animation packages it is possible to create dedicated plug-ins which enhance functionality delivered by the package, for example it is possible to write

custom exporters, which output the created scenes in a format that can be utilised by game engines.

Figure 4 Images generated in a modern animation package (the figure presents five different aspects of creating computer animation)

The most popular and widely used animation packages are 3D Studio Max (Discreet, 2003), Maya (Alias, 2003) and SoftImage (SoftImage, 2003). Their functionality may be divided into three distinct areas (Figure 5). The first subsystem is a so-called modeller. This module is responsible for maintaining the internal data structures supporting all types of objects, scenes and characters. It also allows for visual and possibly scripted creation and modification of these elements by the user and displays them on the screen as geometrical objects. The second component is the animation creator – here the user can for example define motion of objects, adjust timing, add keyframes and preview the animation. Finally, when the scene is ready, it is passed onto a rendering engine which, using information such as lighting conditions, texture and transparency of objects, reflectivity and also desired resolution, number of frames etc., creates the final animation which can then be saved on a video tape or film. To extend the capabilities of the animation packages some companies produce dedicated plug-ins

which add functionality to the package such as extra lighting effects, realistic modelling of liquids, enhanced physics or character support.

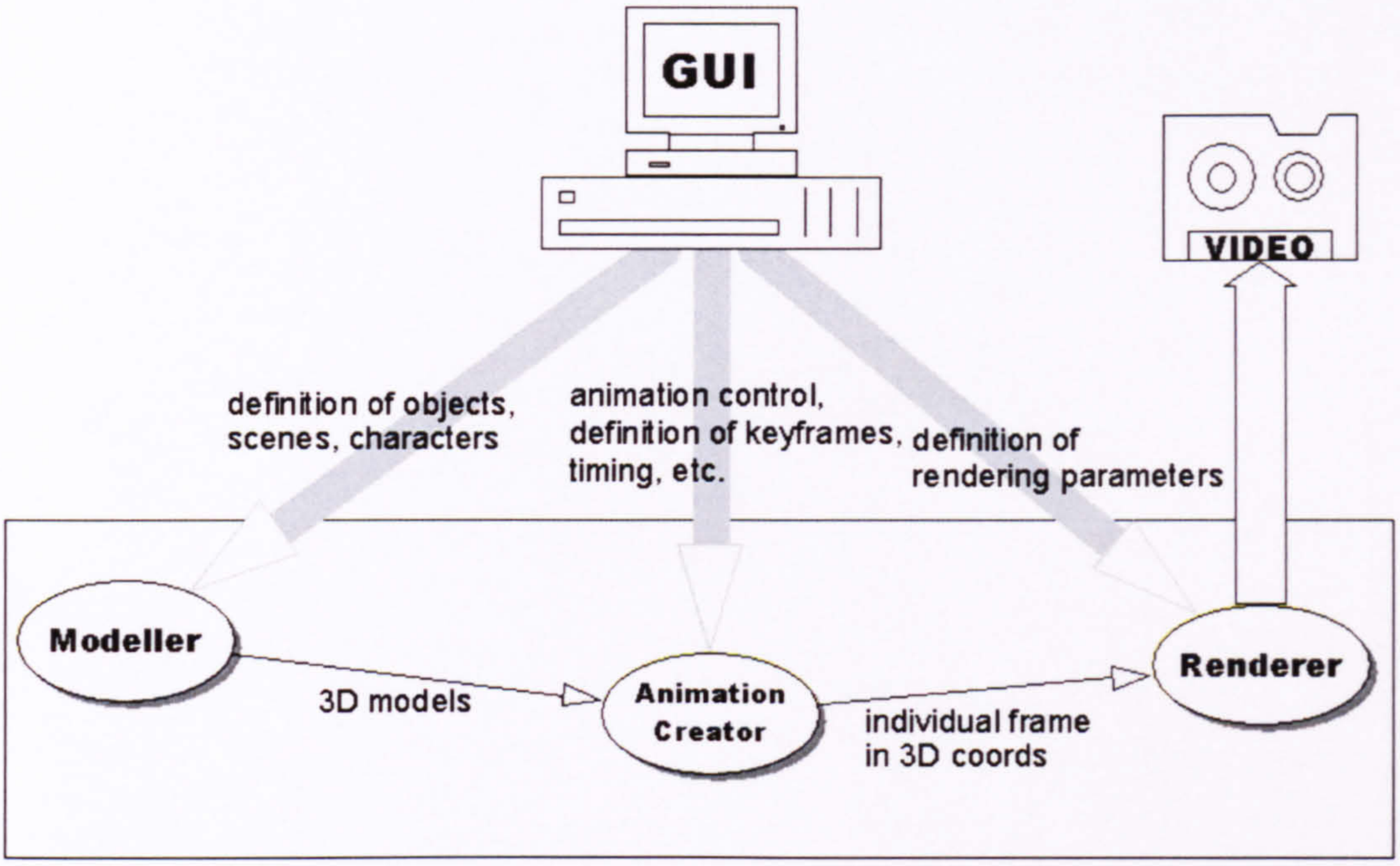


Figure 5 Components making an animation package

An example of the latter may be the "Character Studio" plug-in developed for 3DS Max. Character Studio extends the capabilities of the package by allowing for creation of bipeds (see Figure 6), equipped with most typical human joints. The package also implements a whole range of different control options (generation of human gait, run or jumps, bending the trajectory, freehand animation, etc.). Scenes can be saved into a file and imported into more complex virtual scenes. It is possible to import motion capture files.

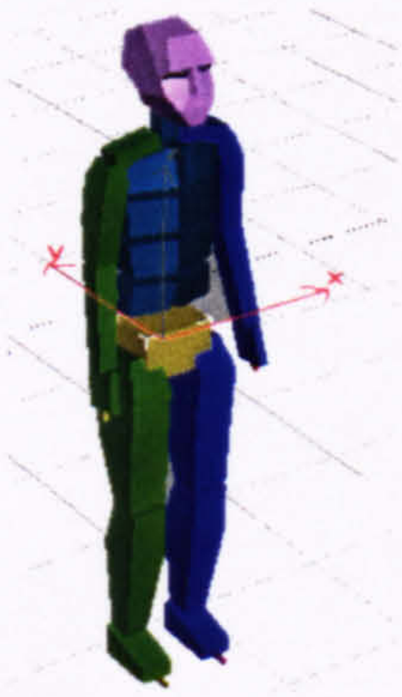


Figure 6 Character Studio biped

The biggest gain from using the plug-in is that it incorporates a reliable forward/inverse kinematics engine integrated into the virtual characters. It is not perfect (e.g. it is possible to drag one leg onto another) but frees the animator from many simple yet tiresome corrections. The characters may also form 'crowds', where characters (called delegates) will have a number of behaviours assigned to them, e.g. a 'repulsive' or 'follow' behaviour. The mechanism relies on a finite state machine underlying the character motion and allows for any type of objects to be assigned to the crowd controller. Like any other geometrical object, the bipeds can also be assigned meshes and textures.

However the use of bipeds created by the Character Studio plug-in has its limitations. The main disadvantage is that the characters do not have built-in collision detection (see Figure 7). More importantly, they are not autonomous, although there is a limited possibility to create crowd scenes with some sort of predefined intelligent motion (e.g. the aforementioned repulsive or following modes of behaviour). Nonetheless most of the job is still left to the animator and director and any enhancements to the avatar behaviour repertoire must usually be very specific with limited scope for extendibility or generalisation. Any apparently 'autonomous' behaviour must be hard-wired into the animation sequence. The characters do not have any self-awareness or scene recognition (identifying objects, avoiding collisions) and the artist has to make all the decisions and direct both main and secondary characters by adding all the necessary keyframes.

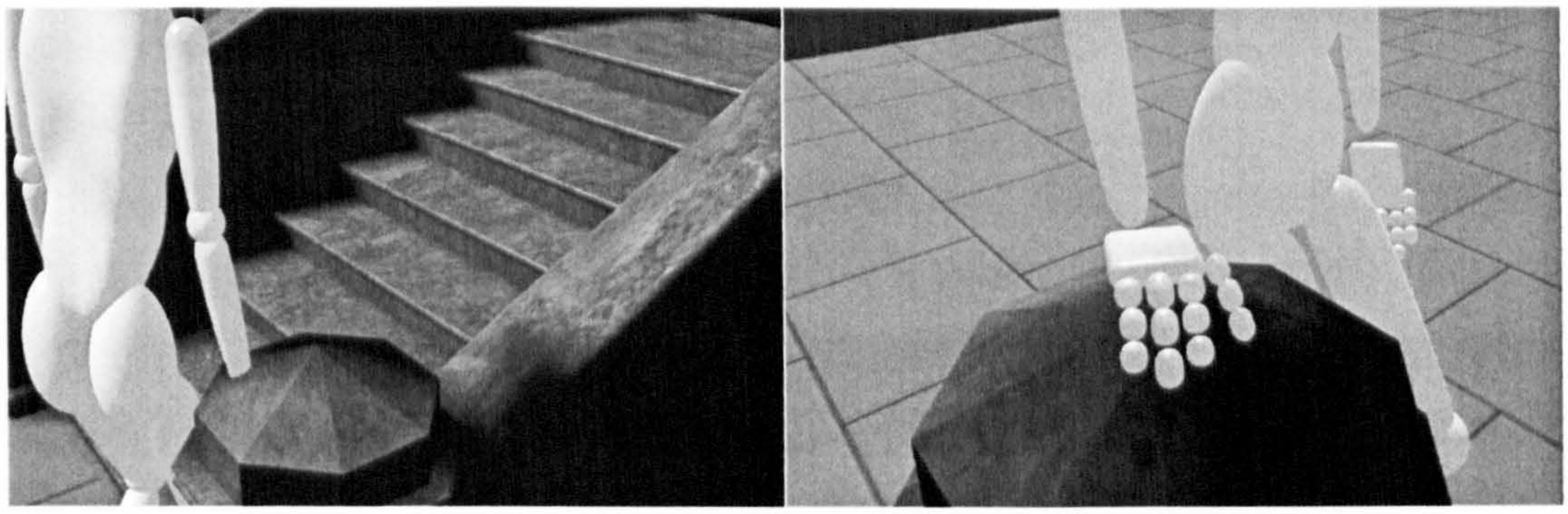


Figure 7 Lack of collision detection (left), manually corrected scene (right)

Solutions to some of those problems are addressed by Maya, one of the most widely used animation packages. Maya offers many highly sophisticated features, including support for cloth motion, built-in collision avoidance and a physics engine. It also supports creation of animated characters by supplying both the forward and inverse

kinematics, animation of objects with joints, built-in constraints and extensive use of motion capture animation but it does not provide a predefined character creation toolkit similar to the Character Studio plugin. Therefore the collision detection feature is only available on the object level. Additionally Maya's characters are completely 'unintelligent'. Despite this lack of functionality Maya remains a very powerful and popular platform and is more widely used in production studios (as opposed to 3DS Max, which is a common training platform). A crowd plugin for both Maya and 3D Studio Max is offered by AI.implant (AI.implant, 2003, AI.implant Animation White Paper, 2003).

Another important professional animation package is SOFTIMAGE|XSI and its recent extension called SOFTIMAGE|BEHAVIOR. The BEHAVIOR package was created to allow for fast production of scenes with multiple interacting objects, such as human beings, birds, insects but also for example blood cells (Softimage, 2003). The system's logic relies on a finite state machine, which can be designed using a set of graphical editing, scripting and debugging tools and a library of pre-defined behaviours. Each BEHAVIOR character may possess a separate 'brain' and can respond and interact with the environment and other avatars. The character engine comprises real-time inverse kinematics, automatic obstacle avoidance based on event-responses, dynamic path planning as well as other features. It is also possible to intertwine kinematics-based character animation with motion-captured scenes and predefined motion and additionally to randomise character movements and timing.

Despite the capabilities offered by modern packages animation is still a labour-intensive task. Using dedicated software, like that described above, animators can decrease this factor by specifying key frames and allowing the software to interpolate between the frames thus creating a continuous and fluent animation. The intelligent extensions support this process even further. Even so, generation of realistically looking sequences, especially when many characters are involved, demands a high number of corrections and manual assignments, making the process tedious and expensive, especially when keyframing is still required. Those problems become especially apparent in where it is necessary to generate scenes involving many interacting human figures. They can partially be overcome by approaches based on "motion capture" and duplication of the

characters. Although some of the packages also allow for more automatic scene generation (SOFTIMAGE|BEHAVIOR, the recent AI.Implant plugin and partially the Character Studio plugin) this comes at the cost of supplying a set of motion-captured animation sequences. It is also always necessary to manually create correct assignments and relations in the character's 'brain'. Even then much of the work in such scenes must be conducted manually by skilled professionals and for a very specific figure's behaviour it can take days of repetitive work to fine-tune the sequences. Similar problems arise in such domains as computer games and simulations of safe evacuation of crowded areas, for example tube stations or skyscrapers, in the event of fire or other disasters. Some recent work and discussion on issues concerning extensions of AI for computer games to make the characters more intelligent can also be found in Woodcock, 2000, Pottinger and Laird, 2000, Kaminka *et al*, 2002, van Lent *et al*, 1999, Amant and Young, 2001. For example van Waveren and Rothkrantz (2002) present a multilayered architecture for Quake III bots, which use automated path and route finding algorithms and Spronck *et al* (2003) applied neural networks to improve the intelligence of computer controlled opponents in a strategy game. It is obvious however that further research into applications of AI to the domain of character animation is still required.

2.4.2 Massive

Unlike the systems described in the previous section, which form parts of professional 3D animation packages, Massive (the Multiple Agent Simulation System In Virtual Environment) is a dedicated industrial module written specifically to support creation of combat scenes with tens of thousands of animated characters. The system was written to aid with creation of battle scenes in the Lord of the Rings trilogy. Thus the focus of the package is on creation of human-like figures animated using a range of different methods with primary character control using Artificial Intelligence. Massive equips each character, known as an 'agent', with an artificial brain and allows it to act on its own. Each character can select from a set of possible motions, each motion lasts about a second (Koeppel, 2002). The decisions a character makes during the course of the battle are predetermined by the construction of its brain – a net of so called logic nodes whose number may vary from a hundred to a few thousand. The logic nodes allow the character to perceive, interpret and respond to signals coming from the environment and

also control the character's behavioural patterns – e.g. aggression or the fighting style. The collections of nodes are further divided into more specialised modules, such as those responsible for navigation, targeting, turning or adopting to the terrain (Lord of the Rings, 2003). The creation of agent brains is carried out in a specifically designed graphical user interface. The result is a multidimensional web of connections and dependencies between various nodes. Additionally Massive employs fuzzy logic for the decision-making process. This is meant to introduce some unpredictability into the simulation, since the mechanism behind the agents' brain is a responsive, event-driven simulation. The decision making process is performed once per frame of animation and the agents are aware of other characters.

Although the characters are equipped with a relatively complex brain architecture the actual motion has been pre-processed using motion capture. However the agents' bodies do react to forces and collisions. A number of different movements and movement combinations have been captured and hard-wired into the characters' logic. Additionally some of the interactions between the agents were also modelled purely on the basis of motion captured sequences involving groups of actors. Thus, combining the captured motion and the logic delivered by the artificial brain, every character can exhibit a number of different activities – march, walk, jog, charge, run, flee, shoot, defend, die, demonstrate combat skills and perform a few other actions. The characters can also find, identify and engage enemies and exist in a number of emotional states. The agents in Massive come in two "breeds" – the more complex and clever Master Agents and more numerous simple agents. The difference is in the size of their brains (the number of nodes and their connectivity) and the number of different actions they can exhibit. An additional type of characters are so called 'Heroes', whose fights only rely on choreographed motion sequences and not on the Massive logic.

As demonstrated in the films, the system allowed for easy creation of battle scenes involving many human-like animated characters, which performed sophisticated actions, involving for example erecting siege ladders. However the main focus of the system was to assist the animators and directors in the creation of believable battle scenes, as opposed to the task of making a number of intelligent characters behave autonomously. An important and useful feature of Massive is a built-in collision

avoidance, but its creation was simplified to some extend by the predefined nature of the characters motions. Also the outcome of the battle was predefined and the system was only employed to create a realistically looking clash. The scenes were additionally broken into easily manageable smaller pieces and sometimes only specific agents were chosen for presentation from the simulated scenes to avoid discontinuities in motion and behaviour. Any changes in character's behaviour require direct intervention into the structure of its brain and it was even necessary to experiment with the placing of agents or to animate them directly to achieve the desired effect in terms of timing and layout. It is not possible within Massive, for example, to let the agents communicate in order to conduct a synchronised attack. Finally the system is a responsive one and the agents do not perform any action planning. In summary, despite obvious improvements with regard to the existing 3D animation packages, the creation of crowd scenes in Massive still requires a high degree of human intervention.

2.5 Motion Capture – the Industry Standard

Motion capture is a very popular technique widely used in the special effects industry. This approach usually involves attaching small reflexive sensors (although other methods are also used e.g. virtual gloves or other mechanical devices) to a human actor or dancer whose motion is then captured using dedicated cameras, digitised and converted into a motion capture file format (Figure 8, Figure 9). Such motion can later be edited, replicated and pasted into pre-created virtual scenes. Instead of animating each joint of the virtual figure, human motion is captured 'as a whole' and applied to the scene, which gives a very realistic effect. However there are many incompatible motion capture formats, different types of equipment – some of them difficult to wear and restraining the actor's moves (gloves). The process is also complex and involves a substantial amount of post-processing. The number of different behavioural patterns is hence typically limited by time and budget. For example in "Gladiator" (see Figure 2[right]) there were only a few different characters created for the Colosseum scene. In this technique the virtual characters are merely copies of real human actors and have no autonomy or awareness of the surrounding environment. The director or animator selects a suitable motion from a library of shots, applies necessary adjustments (cloths, shape of items held etc.) and inserts it into the scene. Thus the generated animation remains very rigid in the sense that it cannot be shaped beyond the existing clips and if

some of the motions have not been captured they must be recorded before applying them to the scene. Cassell *et al* (2001) point out that capturing human motion is an expensive method and suggests it only be used for foreground characters but not for crowds of extras. Arikan and Forsyth (2002) summarise the drawbacks of motion capture as follows:

> 1. Most motion capture systems are very expensive to use, because the process is time consuming for actors and technicians and motion data tends not to be re-used.
>
> 2. It is very hard to obtain motions that do exactly what the animator wants. Satisfying complex timed constraints is difficult and may involve many motion capture iterations. Examples include being at a particular position at a particular time accurately or synchronizing movement to a background action that had been shot before.

Despite these limitations motion capture is currently the most popular technique for the creation of artificial crowd scenes, and have been successfully used in such recent productions as "Titanic", "Gladiator", "The Mummy Returns", "Star Wars Episode 1 - the Phantom Menace", "The Patriot", "Enemy at the Gates", "Pearl Harbor" (Vicon, 2003). Apart from creation of crowd scenes motion capture is used to create virtual stunts and realistic foreground CG characters (e.g. "Lord of the Rings – Two Towers"), the technique is also used to generate synthetic cartoon characters ("Shrek", "Monsters Inc.").

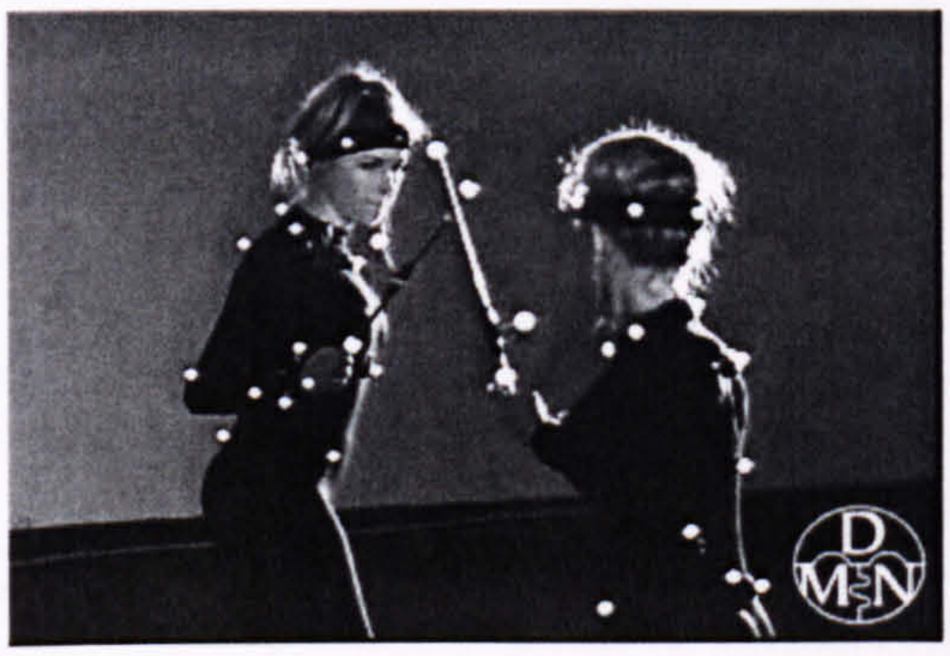


Figure 8 Capturing human motion (image from Vicon, 2001)

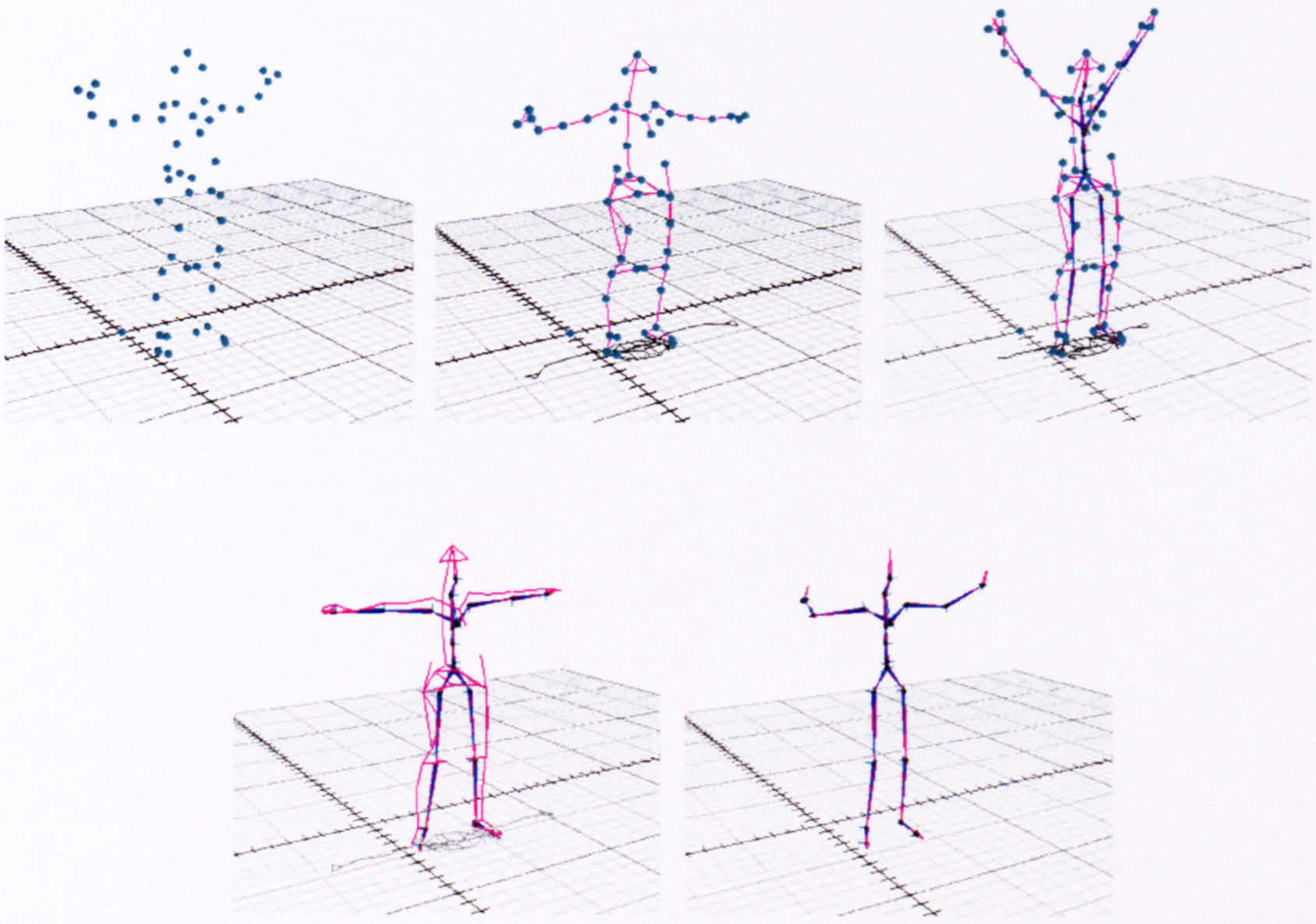


Figure 9 Digitising human motion using motion capture (images from Kinetic Impulse, 2003)

2.6 UML

The concept of modelling has been widely used in many areas of engineering to help visualise concepts and thus promote good design, early risk assessment, resource allocation, efficiency or plan completeness (see Sommerville, 2001). Therefore application of code modelling in the domain of computer programming and system development is a very natural idea, helping software engineers to create good programs. Transparent structure helps tackle complexity, it also promotes code reuse – well-documented independent modules can be shared over many different applications. Unified Modelling Language (UML, Object Management Group, 2003) is a visual modelling tool supporting design and development of software systems, it is also a means of agreeing on the system requirements. Projects designed in UML can be analysed from many perspectives, such as use case analysis, individual objects, flow of control, module interaction, system decomposition, concurrency issues, system deployment and physical layout, and more. UML is sufficiently flexible to allow design of both software and non-software projects, and although it was initially designed to support development of object-oriented programs it supports many different programming languages including procedural languages and database query languages.

There are numerous tools available on the market, many of which can generate code directly from the system specification given in UML. UML diagrams often serve documentation purposes too, given their wide popularity among software engineers, programmers and system analysts. Therefore UML seems to be the right medium for designing and documenting good concepts, including candidate architectures for Artificial Intelligence. Indeed, this is how good software engineering concepts, so called design patterns (Gamma *et al*, 1995), are documented. Moreover, the researchers investigating AI architectures have come up with solutions expanding the core UML notation, making it even more suitable for documenting research in agent solutions. The extended notation is known as Agent UML (AUML, 2003). Agent UML was first proposed by Bauer and Odell and their collaborators (Odell *et al*, 2000a, Bauer *et al*, 2001, Odell *et al*, 2000b, Odell *et al*, 2001). Similar concepts were also presented by Yim (Yim *et al*, 2000) and Bergenti and Poggi (Bergenti and Poggi, 2000). Since then the AUML standard has been consequently revised and updated (Bauer, 2001, Depke *et al* 2001, van Dyke Parunak and Odell, 2001, Wagner, 2002). The main disadvantage of AUML is currently the lack of tools supporting modelling in this language, which is the reason for rejecting AUML approach in this work. It should be expected however that as the technology matures, modelling tools catered for dedicated agent languages will become available.

2.7 Summary

This chapter has presented the most important animation techniques currently used by production studios. This includes creation of animation using motion capture and keyframing. A purpose-build crowd modelling system has also been described. The presented solutions offer limited character and crowd modelling support but further AI-based extensions are still desirable. Finally the UML language has been briefly introduced. The next chapter will now review the state of the art in the research techniques related to automatic generation of animated figures.

Chapter 3 – Cognitive Architectures and Animation Techniques

The previous chapter introduced the current techniques used in the industry. This chapter presents the main areas of scientific research supporting the creation of intelligent animation systems.

There are two dominant approaches taken by researchers working with intelligent characters. The first one, which might be called 'animation orientation' is represented by groups trying to build animation systems by first creating a small number of very sophisticated characters. The characters are usually equipped with a high degree of autonomy, they can learn and perform complex actions. An approach similar to that is the creation of characters with very complex motor skills using dynamic simulation. This category usually includes character animation, simulation of human motion, motion extraction and simulation, and group behaviour. The other approach concerns the generation of scenes with many characters following the current research in multi-agent systems. Although there have been few animation systems strictly following the agent principles (an example may be Flake *et al*, 2001), the techniques offered by agent-orientation have a great potential for modelling systems with many interacting characters.

This chapter presents the current state-of-the art in research falling in both categories and in the area of cognitive architectures. A few other research fields are also discussed. These include Reinforcement Learning techniques, some example robot architectures and motion generation techniques.

3.1 Cognitive Architectures

In the last two centuries we have been observing a huge increase into studies on theories of human and animal learning and cognition (Newell, 1990). The current literature published in the English language lists over 50 recognised theories relevant to human learning and cognitive aspects of knowledge and skill acquisition, and this is without including neuropsychology, learning disabilities or teaching strategies (Kearsley, 2003). Despite this variety and richness most of them have a rather limited use in the current

cognitive modelling research. This is because most of them are only crude approximations trying to explain the way human brain functions at the low level and while they manage to explain some narrow aspect of the learning and cognitive processes they lack understanding of the neural function of different brain elements. This section presents four of the most prominent and widely recognised of these theories in the context of building artificial intelligent systems. While the first two do not currently have other than historical importance they laid a foundation for the fourth one – Soar, which may be considered as a potential candidate architecture for implementation of an intelligent tool controlling behaviour of animated characters.

ACT*

ACT* (Anderson, 1983) advocates existence of three types of memory structure: declarative, procedural and working memory (see Figure 10). Declarative memory is depicted as a semantic net associatively linking propositions, images and sequences. Procedural memory represents information as productions (statements consisting of conditions and actions existing in declarative memory). It is also known as long-term memory. Productions have some degree of activation and the most highly activated part is referred to as working memory. All knowledge begins as declarative data and gives rise to procedural information learnt through inferences from existing facts. This framework supports three important types of learning: generalisation (broadening the applicability of productions), discrimination (opposite to generalisation) and strengthening (some productions are applied more often). New productions are formed from existing ones. One of the strengths of ACT* is that it explains sophisticated human cognitive skills such as geometry proofs, programming and language learning (Anderson, 1983, Anderson 1990). It also accommodates the use of goals and plans.

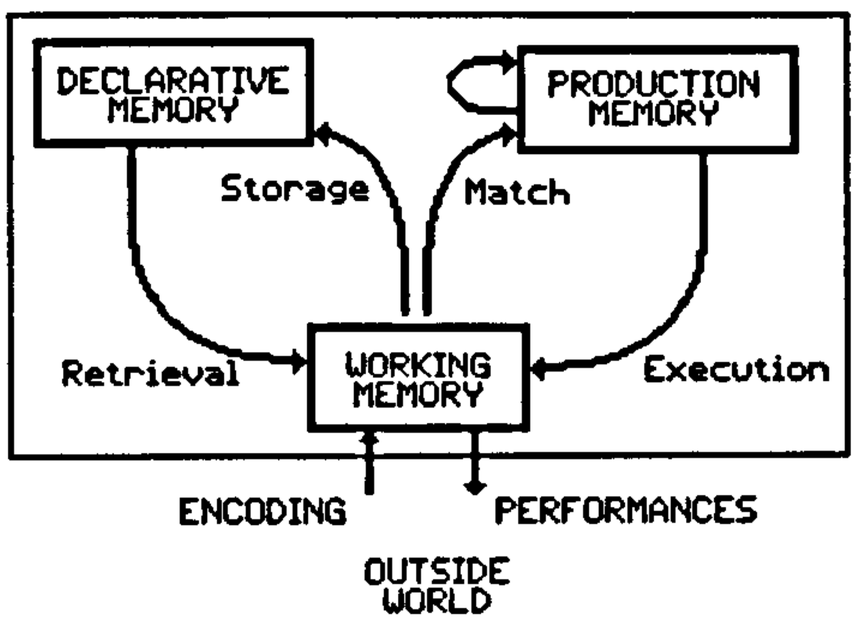


Figure 10 The ACT model (from Kearsley, 2003)

The GOMS Model

GOMS (Card *et al*, 1983) represents a theory of the cognitive skills necessary to perform human-computer tasks. It assumes there is a number of different types of memory (e.g., sensory store, working memory, long-term memory) for which there exist separate perceptual, motor, and cognitive processing.

In the GOMS model, cognitive structure consists of four components:

- set of goals
- set of operators
- set of methods for achieving the goals
- set of selection rules for choosing among competing methods

For a given task, it is possible to construct and use a particular model for prediction of the time required to complete the task. The model can also be used to identify and predict the effects of errors. The main application of the GOMS framework is to various computer tasks, it also serves as system design methodology for testing user interface designs (Kieras, 1988; Oray *et al*, 1993).

Gagne's Conditions of Learning

Gagne (Gagne, 1985) argues for the existence of several different types or levels of learning and that each type requires different types of instruction and different internal and external conditions. There are five main categories of learning: verbal information, intellectual skills, cognitive strategies, motor skills and attitudes. Therefore to learn cognitive strategies a learner must practise developing new solutions to problems, whereas when learning attitudes the learner must experience a credible role model or strong enough arguments. Learning tasks for intellectual skills can be organised in a hierarchy based on complexity: stimulus recognition, response generation, procedure following, use of terminology, discriminations, concept formation, rule application, problem solving.

The hierarchy helps to identify prerequisites that should be completed to ease learning at each level. The theory names nine instructional events and corresponding cognitive processes (Gagne, 1985):

(1) gaining attention (reception)

(2) informing learners of the objective (expectancy)

(3) stimulating recall of prior learning (retrieval)

(4) presenting the stimulus (selective perception)

(5) providing learning guidance (semantic encoding)

(6) eliciting performance (responding)

(7) providing feedback (reinforcement)

(8) assessing performance (retrieval)

(9) enhancing retention and transfer (generalisation).

Gagne's framework covers all aspects of learning but the focus of the theory is on intellectual skills.

Soar

Soar (Laird *et al*, 1987), developed by John Laird at the University of Michigan, Paul Rosenbloom at the Information Sciences Institute of the University of Southern California and Allen Newell at the Carnegie Mellon University, was meant to be "an architecture for general intelligence" (Newell, 1990). It was probably the first proposal to be both theoretically well justified and successfully implemented on a computer, allowing the creators to conduct 'computer-supported' research into the theories of cognition. Soar emerged from research in AI over many years and contains elements typically found in production systems (Figure 11) – working memory (also called global database, short-term memory or fact list) which is the system's representation of the current state in its world, long-term memory (production rules, knowledge base) comprising condition/action rules taking the system from one state to another and a control structure deciding which production rule(s) fire next (Franklin, 1995). A schematic representation of the co-dependencies of the above mentioned subsystems is depicted in Figure 12 and the complete Soar system equipped with perceptual and motor systems is shown in Figure 13.

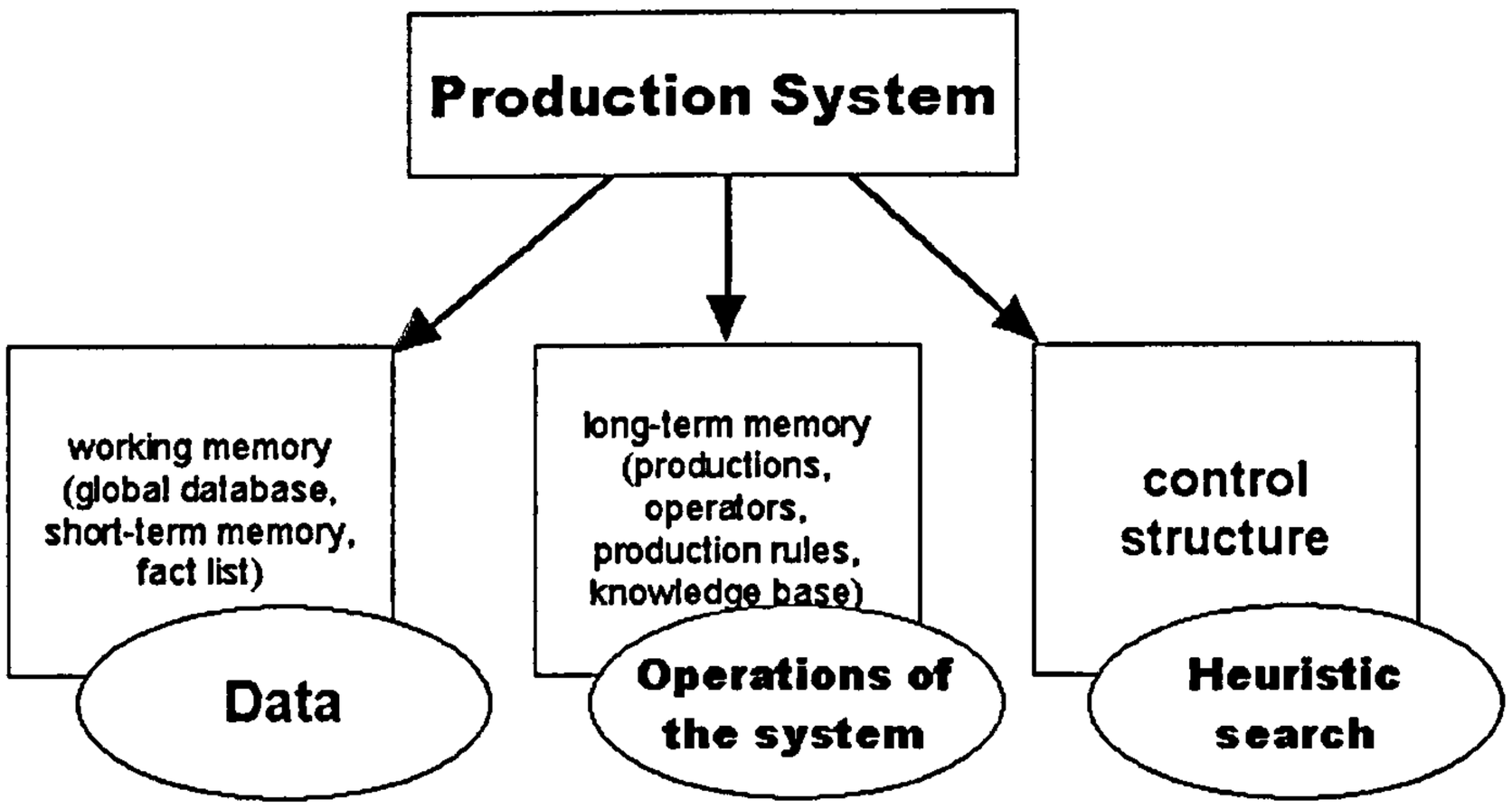


Figure 11 Components of a production system

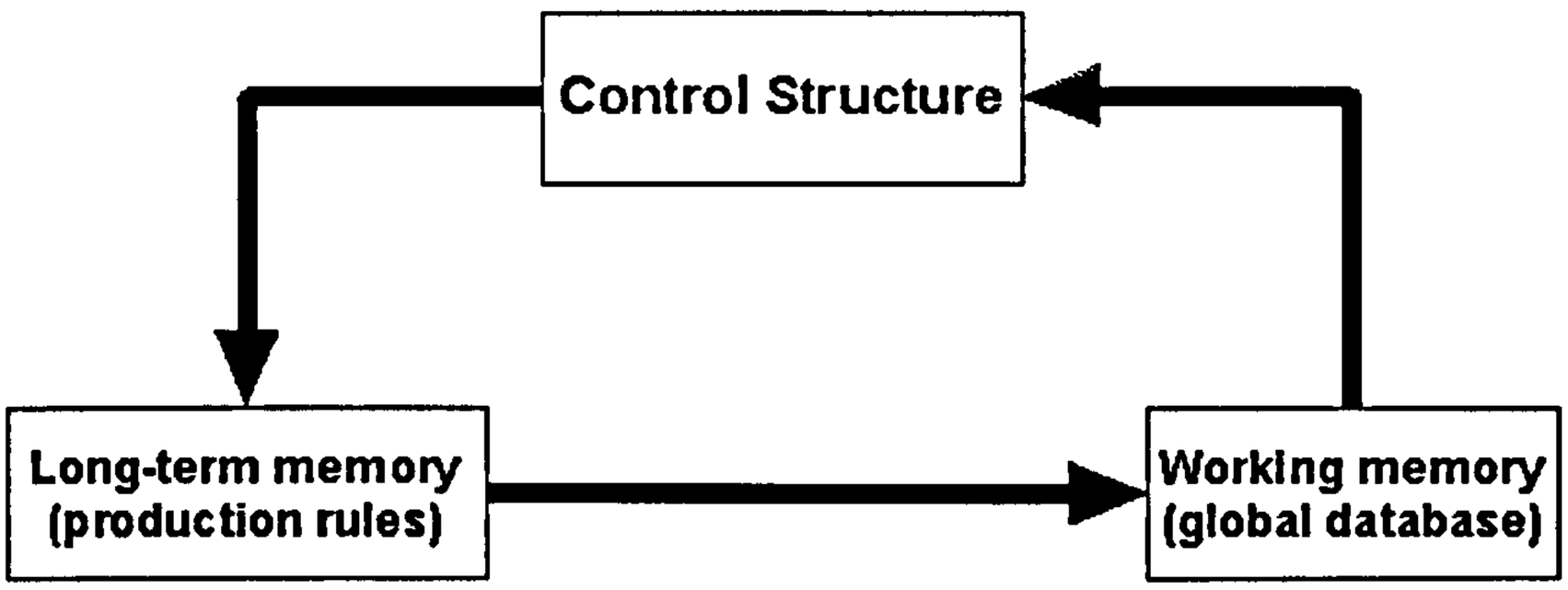


Figure 12 Dependencies in a production system (after Franklin, 1995)

Soar was built around the following well-defined principles as identified by Newell:

- All tasks are represented in problem spaces, therefore all cognitive actions occur within a search space. In each problem space there are available operators which can be applied to the current state. To operate in a problem space the system must know how to implement the operators and how to guide the search – this knowledge is kept in the long-term memory.
- Productions provide all long-term memory (search control, operators, declarative knowledge). Handling all Soar activities is done by this structure – a single production system. Newell argues that such uniformity of architecture is a good idea even though many AI researchers hold the view that declarative and procedural memory are different kinds of memories. Productions comprise sets of conditions which, if matched by elements in the working memory, fire actions entering new elements into the working memory. Therefore such a structure can well be described as a pattern matching system. Many productions may fire in a single cycle – there is

no conflict resolution strategy, which would choose only one, most suitable production.

- Attribute/value representation is used for all objects. Conditions, actions and working-memory elements are object-attribute-value triples, those triples are used for all representations within Soar.
- A preference-based procedure is used to make all decisions (accept/reject, better/indifferent/worse). Preferences are Soar's means of representing what actions should be taken. There are different types of preferences – a decision can be acceptable, rejected, indifferent, better, worse, best and worst. Only the preferences are considered by the decision procedure when it chooses the next step to take. A number of common-sense heuristics is applied when making the choice.
- Goals direct all behaviour (subgoals are created automatically from impasses). Sometimes the decision procedure does not give any obvious decisions to be made. In such cases the system goes into an impasse and therefore creates a subgoal to resolve it, therefore subgoals are only created when Soar does not know what to do next. There can be four kinds of impasses: a tie impasse – when all alternatives are acceptable, none are preferred, none can be rejected and they are not indifferent; no-change impasse – there are no choices at all; reject impasse – the only preference is one to reject a decision which has already been made; conflict impasse – when for example one operator is said to be better then some other operator and at the same time another preference says an opposite thing.
- Chunking of all impasse resolutions occurs continuously. Chunking is Soar's method of learning, which occurs during problem solving. Chunks are new productions created when an impasse is being resolved. Chunks can be used as soon as they are created. Chunking makes Soar's problem solving clearly faster with practice and allows for inter-task knowledge exchange.

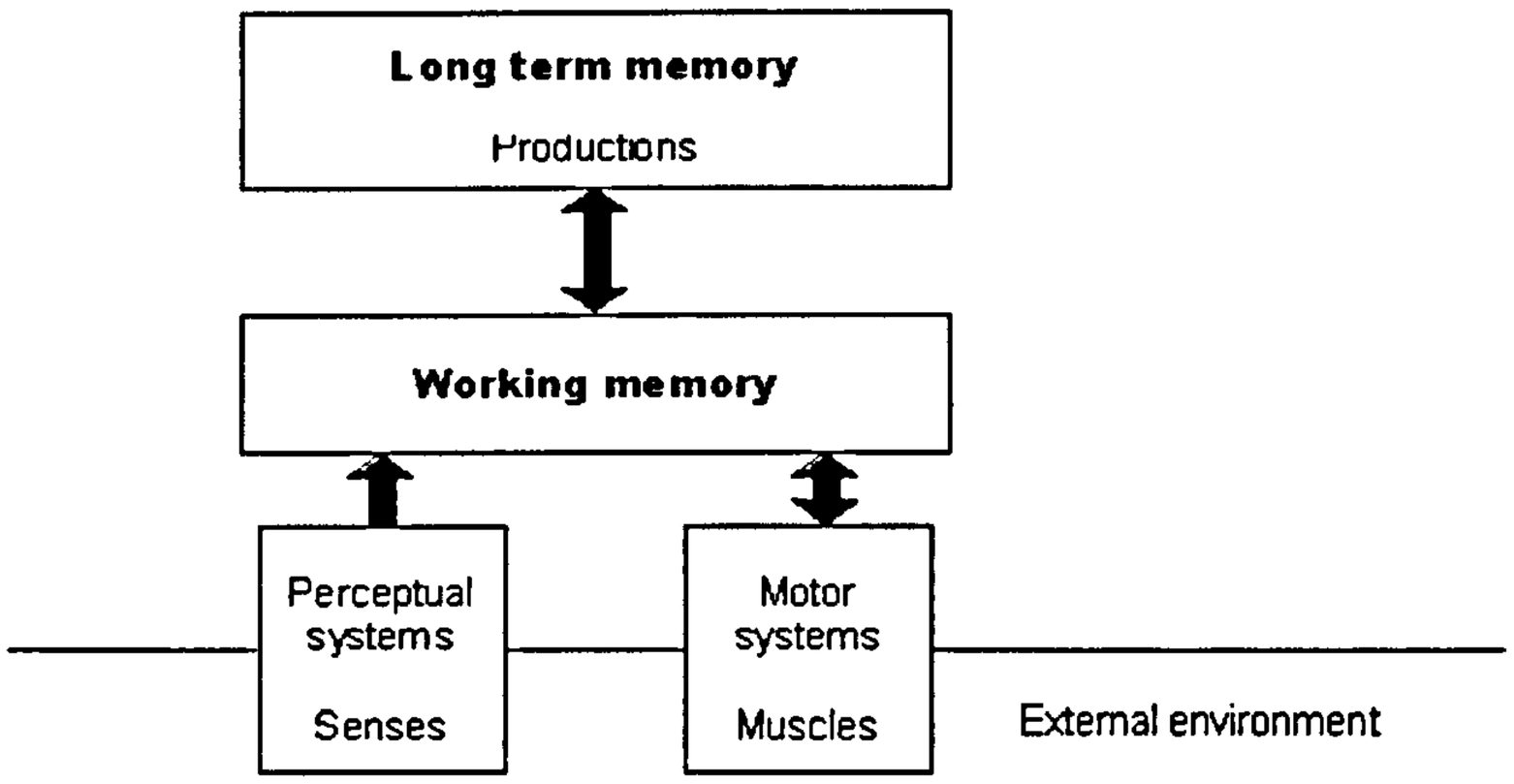


Figure 13 Soar's components (after Newell, 1990)

In summary – it should be pointed out that Soar is a very sophisticated finite-state machine without deliberative planning facility. There is no automatic task acquisition – all tasks given to Soar must be hand-coded, Soar learns monotonically and so it can not recover from learning errors.

Soar is a very popular and widely used tool allowing many researchers to investigate the working of human cognition, learn more about man-machine interaction and explore other AI-related fields. For instance NeuroSOAR was a connectionist attempt to implement Soar's production systems and decision procedure using neural networks (Cho *et al*, 1991). However due to uniformity of Soar's design all external knowledge must be first well understood and encoded into Soar production rules (cf. Wray 2002), which can be a tedious and demanding process. For example in the Soar-based real-time system directing computer simulated aircraft pilots (TicAir-Soar) created at the University of Michigan (Laird and Jones, 1998) there were approximately 5200 rules (and over 450 operators), many of them acquired after extensive consultations with experts in the field (military pilots). Such a rule-based knowledge must be complete and correct otherwise the system will not work properly. The process of acquiring it for serious applications will inevitably be expensive and require much iteration. For a system directing interacting characters this may mean that the developers would have to create hundreds of symbolic rules showing the Soar engine (or any similar one, cf. CLIPS, Giarratano and Riley, 1989 or JESS, Friedman-Hill 2003) how to handle many 'obvious' things such as biomechanics, inverse kinematics, or world geometry. Additionally Soar does not offer any way of handling multi-agent interactions (or

interactions between different instances of the Soar engine) other than through the environment. While this is sufficient to build a successful system the developer would again have to resort to creation of numerous detailed production rules. Therefore it appears that Soar may not be the best tool to create the proposed multi-agent animation system.

3.2 Animation Architectures

Most researchers who build intelligent characters do not attempt to mimic how the human brain works at the low level (by for example building complex neural structures). Instead new architectures are proposed, which may be informed by research into human or animal behaviour or can just be new proposals for creating virtual systems. The next sections give an overview of the most important systems designed to support creation of synthetic creatures. Special emphasis is placed on systems which include some form of Reinforcement Learning, that is unsupervised learning directed by trial and error.

3.2.1 Cognition-based animation systems

One of the dominant approaches taken by researchers trying to build intelligent characters is represented by research groups trying to build animation systems by creating a small number of relatively sophisticated characters, animated using kinematic techniques and/or predefined motion. The characters are usually equipped with a high degree of autonomy, can learn and perform complex actions.

A good example here is C4, an architecture proposed by a research group working at the MIT Media Lab (Isla *et al*, 2001, Burke *et al* 2000). C4 was designed to allow creation of autonomous and semi-autonomous creatures. The work is informed by the behaviour of living animals (in particular dogs). The main concept is based on a 'brain' divided into numerous systems communicating through an internal blackboard (Craig, 1995). The modules making C4 are: sensory and perception system, action system, navigation and motor system (which in turn is also composed of different layers e.g. Look At Layer, Emotion Layer), short-term memory and the mental blackboard (see Figure 14). Thus the whole system is highly modular which is its main architectural principle. The difference between the perception and sensory systems is that the perception system is

responsible for assigning meanings to the events in the world. Working memory is a repository for persistent objects constituting the creature's view of the world. An important point is that the world model not only maintains a list of creatures and objects present in the simulation but also serves as an event blackboard for posting and distribution of world events.

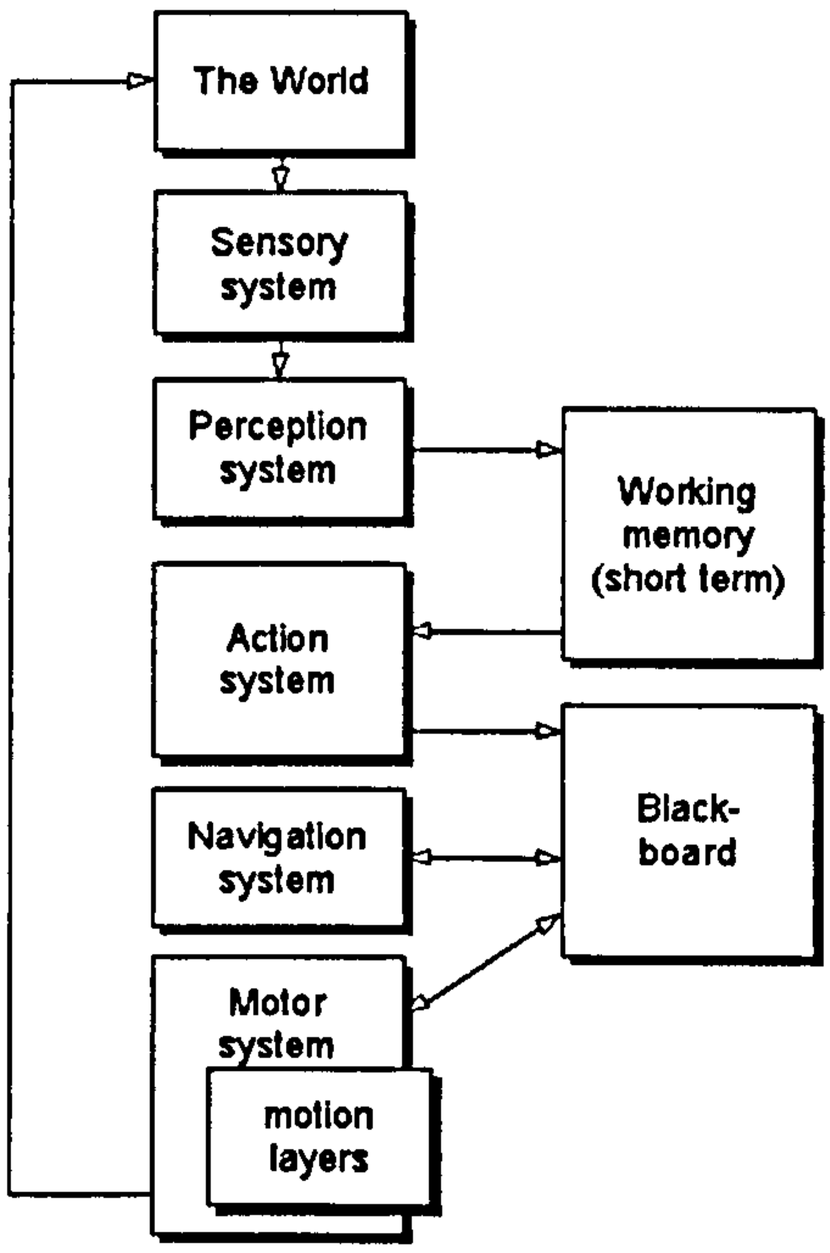


Figure 14 C4 components (after Isla et al, 2001)

The working of the system is supported by a robust version of a sense-learn-act loop with strong emphasis on the learning. C4 allows the AI components of the animated characters to perform (among others) the following capabilities:

- *reactive behaviour*
- *learning*
- *planning*

It is also defined as being highly extensible. The building bricks of C4 are carefully designed data structures, different for different subsystems, which allow for representation of many kinds of information present within the system and which have also great impact on the learning abilities. Examples can be DataRecords holding the sensory/perceptual information, or PerceptMemory objects residing in the working memory. Thus each creature acts according to the contents of its own memory.

The results of applying the system to create a film-like demonstration are admittedly impressing. While some manual modifications to the presented videos might still have been necessary, the system does allow for natural representation of both 'individualistic' characters (a dog) and semi-autonomous creatures such as a flock of sheep. It puts a strong emphasis on learning and defines a large hierarchy of stimuli from the environment (which is necessary in order to train the dog successfully). Different types of virtual creatures can also be represented.

C4's emphasis on learning, especially in the form of training a home pet (a dog), makes the system biased towards solutions allowing to handle this type of interaction. This includes dedicated data structures and a hierarchy of purposefully defined objects. It is not obvious how this design would fit into simulation of a human crowd (the only demonstrated type of 'crowd' was a flock of sheep) and whether the learning system is flexible enough to allow unsupervised learning of more advanced activities involving interaction with objects. This might include for example opening a door. Additionally the inter-character interaction is limited to flocking behaviour and issue of commands, the presented scenes do not contain more than one fully intelligent character. Therefore some serious changes to the system would probably be necessary in order to allow it to generate many intelligent human-like characters.

An extension of C4, which includes much greater learning capabilities, is described in Blumberg *et al* (2002). The applied learning algorithm is a modification of the Reinforcement Learning technique. However the task of the learning engine is not to learn the necessary motor skills but rather is defined on a higher level, "with respect to a motivational goal of moving in a certain way" and happens in real-time during the interaction with the system. Thus the authors define their learning system as being more abstract than the traditional approaches and additionally aim to use learning as a means of increasing online interactive capabilities of their characters and not as a design tool. The whole system was built to mimic a dog training technique called the "clicker training". The system uses a so-called pose-graph to generate motion, the nodes of which are derived from source animation amended by an interpolation technique (Downie, 2000). Thus the animation is realistic and transitions can be generated in real-

time but the actions must be prepared by an animator and pre-programmed into the system.

Another example of applying Reinforcement Learning (RL) to animation includes Yoon *et al* (2000), where RL techniques were used to create motivational and emotional states for a human character. This system incorporates such concepts as motivation driven learning (where the source of the reinforcement signal for learning was the creature's motivational module), organisational and concept learning but not motor learning. Similarly as before the learning occurs on a higher level and only affects the character's behaviour in an indirect way.

Another interesting work presented by the MIT Synthetic Character Group includes a study of user interaction with high-level control over synthetic wolves, whose emotional state is maintained by a computer (Tomlinson *et al* 2002), or a behaviour-based reactive autonomous cinematography system (Tomlinson *et al* 2000). In the former the intelligence was put not into a virtual character but rather into a virtual camera and lights in order to enhance the emotional expressiveness of the animation. Kline and Blumberg (2000) proposed an enhancement to the believability of virtual characters by endowing them with expectations. A summary of the systems created by the group can be found in Downie *et al* (2002). Additionally Russel and Blumberg (1999) identify elements and features which must be exhibited by a good real-time synthetic character architecture and present an example implemented solution. This discussion is complemented by Kline and Blumberg (1999) with a methodology for designing virtual characters exhibiting such characteristics as intentional behaviour and attempts to satisfy their desires.

The common feature of all systems created by the researchers from the MIT Synthetic Character Group is that the motor systems are always built upon animation material prepared by animators rather than creating character motion from scratch (Downie *et al* 2002).

Funge's Cognitive Architecture

Another example of a system dedicated to the creation of autonomous characters is John Funge's cognitive architecture (Funge, 1998, Funge, 1999, Funge *et al*, 1999, Funge 2000). Funge's research interests concern building cognitive systems dedicated for animated characters. He strongly supports the idea of adding autonomy to computer-generated creatures and he defines an autonomous character as "a character that, during the course of a computer game or animation, can decide how to behave on its own." (Funge, 1999). Funge argues that virtual characters should maintain a cognitive model that is an internal model of their world and additionally an explicit representation of some other knowledge about the character's world (Funge, 1998, Funge *et al*, 1999). Therefore he describes autonomous cognitive characters as characters having some domain knowledge – knowledge about the world's dynamics. At the same time he claims that generally the character should not be omniscient, which happens when the characters gain access to the 'true' world model. Funge formalises his ideas using the Situation Calculus (Levesque *et al*, 1997).

Funge emphasises that the cognitive layer is only one among many models underlying an animated character (Figure 11). He recognises that a developer trying to build a virtual scene will be faced with many problems apart from those of Artificial Intelligence. What he proposes in not only a cognitive architecture but also a whole process of building a virtual character.

Unfortunately the character interaction component seems to be omitted in the model. Funge concentrates on building a single character without addressing the problem of how to exchange information or co-ordinate its behaviour with other creatures. Also the process of building the whole system, especially how to merge all different aspects of the cognitive engine, is not addressed in detail and the learning aspect in only mentioned without deeper consideration. Some extensions introducing the missing components could make this architecture even more suitable for design and implementation of intelligent characters inhabiting virtual worlds.

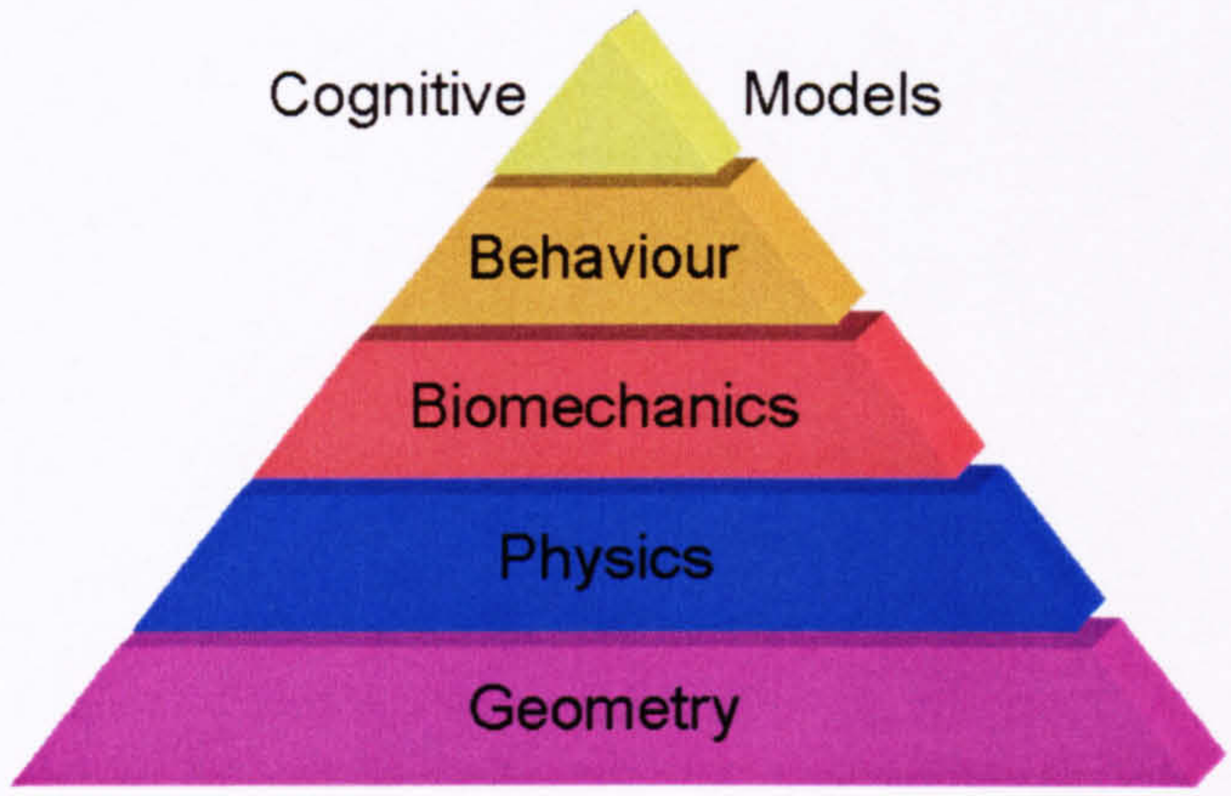


Figure 15 Pyramid depicting different models used to build a virtual character (after Funge, 1999)

Genetic-based Approach

Sims (Sims, 1994b) proposed a framework based on genetic algorithms which can create evolving creatures consisting of simple geometrical figures specialised to perform different activities, such as swimming or walking. The system could also simulate the laws of physics (friction, gravity, collisions) and make the evolving creatures compete for resources thus promoting further specialisation (Sims, 1994a). Another application of genetic algorithms to behaviour learning for animated characters was presented by Wan and Tang (2002) and also by Shim and Kim (2003) who presented a technique for evolving double-winged creatures, using genetic algorithms and neural networks. The flying creatures are able to fly in virtual landscapes without the need for complex modelling of flight dynamics. Gritz and Hahn (1997) applied genetic programming to construct simple controllers for character animation. One of the main problems with applying genetic techniques to generate animation is finding an appropriate fitness function, without making it too task-specific.

3.2.2 Physics-based controllers

An approach similar to the one described in the previous section is creation of (usually bipedal) characters with very complex motor skills using dynamic simulation. In this approach the emphasis is placed on the creation of one realistically modelled character simulated using the motion equations and physics-based controllers. Tu, Terzopoluos and Grzeszczuk (Tu and Terzopoulos, 1994, Grzeszczuk and Terzopoulos, 1995) present a system for animating dynamically simulated fish and snakes. However a similar approach applied to dynamic simulation of human figures requires that the

characters have many degrees of freedom thus making it computationally expensive. Despite that complication a lot of research is being conducted in this field. Hodgins *et al* (1995) propose controllers for three different athletic behaviours. Apart from dynamic simulation they also use state machines, techniques for reducing disturbances to the system introduced by idle limbs, and inverse kinematics. Van de Panne and others (van de Panne *et al*, 2000) propose a limit cycle algorithm for the animation of a walking biped and a dynamic motion planner for simplified characters (Acrobot, Luxo). Laszlo *et al* (1996) applied the limit control cycle technique to a 19 degree-of-freedom model of a human, similarly Anderson and Pandy (1999) investigated realistic simulation of human gait using a 23 degree-of-freedom model.

There have been few attempts to build dynamic controllers, which could control more than one specific motion. Examples of these are the ones proposed by Pandy and Anderson (1999) who tried to create a controller applicable to both jumping and walking behaviours and also the work presented by Faloutsos and his colleagues (Faloutsos *et al*, 2001a, Faloutsos *et al*, 2001b, Faloutsos, 2002), who combined several different controllers and additionally applied Support Vector Machines (see Christianini and Shawe-Taylor, 2000) to automatically learn preconditions of different dynamic actions as an off-line process. An alternative to physically-based simulation was proposed by Lee *et al* (2000) by implementing a system in which constraints imposed on motion of a character are calculated in a procedural way. Thus the calculations are faster and more stable and can easily be used in real-time applications.

3.2.3 Flocking systems and crowd simulation

A similar branch of research is automatic generation of animation sequences involving multiple computer characters, which is also concerned with the creation of intelligent characters but on a more massive scale. This is often referred to as crowd scenes. One of the first papers to be published in this area was Reynolds (1987) where an algorithm for simulating flocking behaviour was presented. Reynolds introduced the concept of "boids" (from bird-like, bird-oid), which is sometimes used to refer to agents, which are a part of a flock, herd, school or swarm and not necessarily birds. His ideas included separate vision systems for each simulated bird, collision avoidance and distributed behavioural modelling. The system was superior to the previously applied force fields

(Amkraut, 1987, after Reynolds, 1987) and particle systems (Reeves, 1983, after Reynolds, 1987). Furthermore Reynolds related his research to studies conducted by zoologists. More recently he extended his system by considering different types of behaviours which can be exhibited by flocking agents (Reynolds, 1999).

Another example of a complex animation system is that proposed by Terzopoulos and his collaborators (Terzopoluos *et al* 1994, Terzopoluos *et al* 1996), who extended ideas proposed by Reynolds. They created a marine world inhabited by realistically looking and behaving animals (mainly fish) to explore synthetic vision and navigation systems and applied flocking algorithms to fish equipped with a vision system (Tu and Terzopoulos, 1994). They also employed machine learning to acquire complex motor skills for the simulated fish. The virtual characters are able to learn low-level motions and also high-level behaviours. In their approach the researchers use physics-based simulation and create a dynamic model of the fish with muscles and springs. Such an approach to motion control however makes the simulation computationally demanding and so Grzeszczuk (Grzeszczuk, 1998, Grzeszczuk *et al*, 1998) proposed an application of neural networks to emulate dynamics. They claim that using this approach physically realistic animation can be generated one or two orders of magnitude faster than when using numerical simulation (Grzeszczuk *et al*, 1998). Still the system can only be applied to characters with relatively small number of degrees of freedom.

Anderson *et al* (2003) took further the ideas proposed by Reynolds and implemented a system generating flocking behaviours with constraints imposed by the user. The constraints can be defined for positions and timing for any number of characters, including the centre of mass for subsets of the flock and it is also possible to define desired shapes for the flock. The system calculates the motions in two steps – first an initial set of trajectories, which match the criteria is proposed, next the motion is improved (e.g. collision avoidance is added) while preserving the constraints. This two-state process introduces some problems however, as it may be difficult to quickly improve some trajectories (e.g. when many obstacles are present). The system is aimed at off-line production environments and can generate many candidate solutions.

Hodgins and collaborators worked on the modelling of motion of many agents with significant dynamics (Brogan and Hodgins 1997, Hodgins *et al* 1995) and also on adapting similar behavioural patterns to different creatures and environments (Hodgins and Pollard, 1997, Pollard and Hodgins, 1998). Metoyer and Hodgins (2000) presented a framework for rapid crowd motion prototyping, where simplified bipeds are playing American football. Additionally their agents can learn high level behaviours from real data using a memory-based learning algorithm.

Another important field of research within the crowd-modelling framework is simulation of human behaviour in urban environments. An example is the Informed Environment framework (Ferenc *et al*, 1999a) based on hierarchical decomposition of the scene with additional shift of part of the intelligence from the characters to scene objects (smart objects, Kallmann, 2001). In another paper (Ferenc *et al*, 1999b) this concept is utilised to perform simulations with virtual characters.

Tecchia *et al* (2001, 2002a) proposed a system for simulating agent behaviour in urban environments, the results they presented included up to 20 thousand agents, although it is claimed that the system can simulate up to 50 thousand figures. They also proposed a four-layered structure for agent control, which included 2D-grid inter-agent and agent-environment collision detection, based on collision maps (two-dimensional maps of objects) with a possibility of adding an elevation model. The system has been extended by adding rendering of shadows (Tecchia *et al*, 2002b), in that case however the presented results do not comprise inter-agent collision avoidance (Loscos *et al*, 2001, Loscos *et al*, 2003). Still (2000) modelled crowd dynamics to simulate emergency evacuation from stadiums and railway stations.

Two interesting recent architectures are ViCrowd (Raupp Musse and Thalmann, 2001) and ALOHA (O'Sullivan *et al* 2002). Both architectures support simulation of human crowds in real time. ViCrowd attributes intelligence to groups of characters rather than individual agents. This is in part implied by the required real-time performance of the framework. The system allows for control of the animation using scripted behaviours, external interaction or reactive rules and events, including general, local and emotional events. Crowds, groups and individual characters are equipped with intentions, beliefs

and knowledge, however the most 'intelligent' entities are the groups. Navigation is handled through precomputed paths, calculated for movements of each individual for the task of travelling between the current goal and the next one. A dedicated scripting language has also been created for control of the simulation. A short summary of the system is shown in Table 2.

Table 2 Characteristics of the ViCrowd system (after Raupp Musse and Thalmann, 2001)

Structure	Crowd, groups and agent levels
Participants	Many
Intelligence	Limited
Physics-based	No
Collision handling	Collision avoidance
Control	Pre-defined behaviour, rules and guided control

ALOHA puts greater emphasis on time efficiency of the simulation, thus optimising the geometry of the characters as well as motion and behaviours. The system can additionally be coupled to a voice generation module (Cassell *et al*, 2001).

Monzani (Monzani *et al*, 2001) presents a crowd simulation system based on agents driven by the BDI architecture. The creation of an agent is split into parts, including low-level and high-level tasks. The low-level layer comprises what they call *physical elements*, that is a 3D graphical representation of the avatar, animation and sound generation. Agents are able to interact with objects and other characters and communicate verbally. The high-level aspect of the architecture spans beliefs, emotions, goals and predefined plans. The authors of the architecture strongly emphasise the need for separating the motion generation and behavioural aspects of the agent's design. During the simulation each agent is controlled by a separate thread and awareness of other agents and their actions are implemented by some form of shared memory. Certain aspects of the agent design are modelled in UML, however the presented design is very general and mainly comprises multiple layers of inheritance necessary to construct the agent (see also Monzani, 2002). Animation is handled using a layered architecture supporting actions, tasks and task stacks. The system creates a scenario-driven simulation with inter-avatar task synchronisation through smart objects (Kallmann, 2001), which contain information about their functionality and guide the interaction.

Arafa *et al* (2002) describe two XML-based languages for scripting the animation of simulated characters, Character Mark-up Language for attribute definition and animation scripting and Avatar Mark-up Language meant to be used by human animators. Also recently Devillers and Donikian (2003) proposed a scenario language for controlling and directing semi-autonomous creatures. The language supports among other features character communication and scenario scheduling.

Ulicny and Thalmann (2001) proposed a crowd simulation system in which agent behaviour can be both scripted and autonomous. Each agent consists of three elements: attributes, high-level behaviours (implemented using finite-state machines build on top of low-level actions) and rules governing the selections of the behaviours. Low-level actions include pre-recorded animation sequences and walking. Additionally events provide the agents with a way of interacting with the environment or other agents. Global path planning is also used. The authors implicitly distinguish between visual representation of the character and the logic (called an agent). The simulations are conducted in real-time and allow for user intervention.

More recently Mac Namee and Cunningham (2003) proposed an architecture for creation of socially aware non player characters in computer games. Their work is influenced by the studies of personality models and they focus their attention on the modelling of emotion using neural networks. Their agents are also claimed to be proactive and persistent, meaning that they exist and act regardless of the course taken by the human player.

3.3 Agent-based Systems and Methodologies

This section presents examples of agent-oriented design techniques, which could potentially support the creation of multiple animated characters. First the BDI model and its descendant – the SAC concepts are discussed, this is followed by a presentation of two important agent-oriented design methodologies – Gaia and Tropos.

3.3.1 The BDI model

Together with the appearance of multi-agent systems into the main stream of AI research, many of the proposed frameworks are based on the notion of an intelligent

agent. An example is the BDI, Belief Desire Intention, architecture proposed by Rao and Georgeff in 1991 (Rao and Georgeff, 1991, Rao and Georgeff, 1993, Rao and Georgeff, 1995, Georgeff *et al* 1999). The BDI architecture has its foundations in philosophy but referenced to computer-based systems, and proposes a unified framework for the development of agents in situated planning systems, that is systems embedded in dynamic environments, which have to recognise and respond to occurring events. At the same time these systems must attempt to achieve their goals. Thus Rao and Georgeff conclude that situated systems must be both reactive and goal-oriented.

The model relies on five important concepts:

- beliefs, which in general represent the agent's knowledge base,
- desires describe a set of states the agent wants to "effectuate"; generic desires do not have to be consistent
- goals – a set of consistent desires (at a specific time),
- intentions depict the current goal and the means to realise it, in other words this is the set of currently assumed plans,
- plans, are predetermined action or goal sequences, which accomplish specific tasks.

Rao and Georgeff also propose a theoretical formalism to support their findings. In general however, despite numerous applications of this architecture, it remains rather theoretical and complex in implementation (Winikoff, 2001).

3.3.2 SAC

Recent proposals by Winikoff (Winikoff *et al*, 2001, Harland and Winikoff, 2001) extend the BDI architecture to make it more accessible as a software engineering development framework (see Figure 16). Winikoff's objective is to propose "a simplified model which retains the power (and efficiency) of the BDI model but allows more people to develop intelligent agent systems" – hence the name SAC (Simplified Agent Concepts). Winikoff's concepts are much more 'implementable' than the original BDI architecture and he puts more emphasis on goals, events and construction of plans.

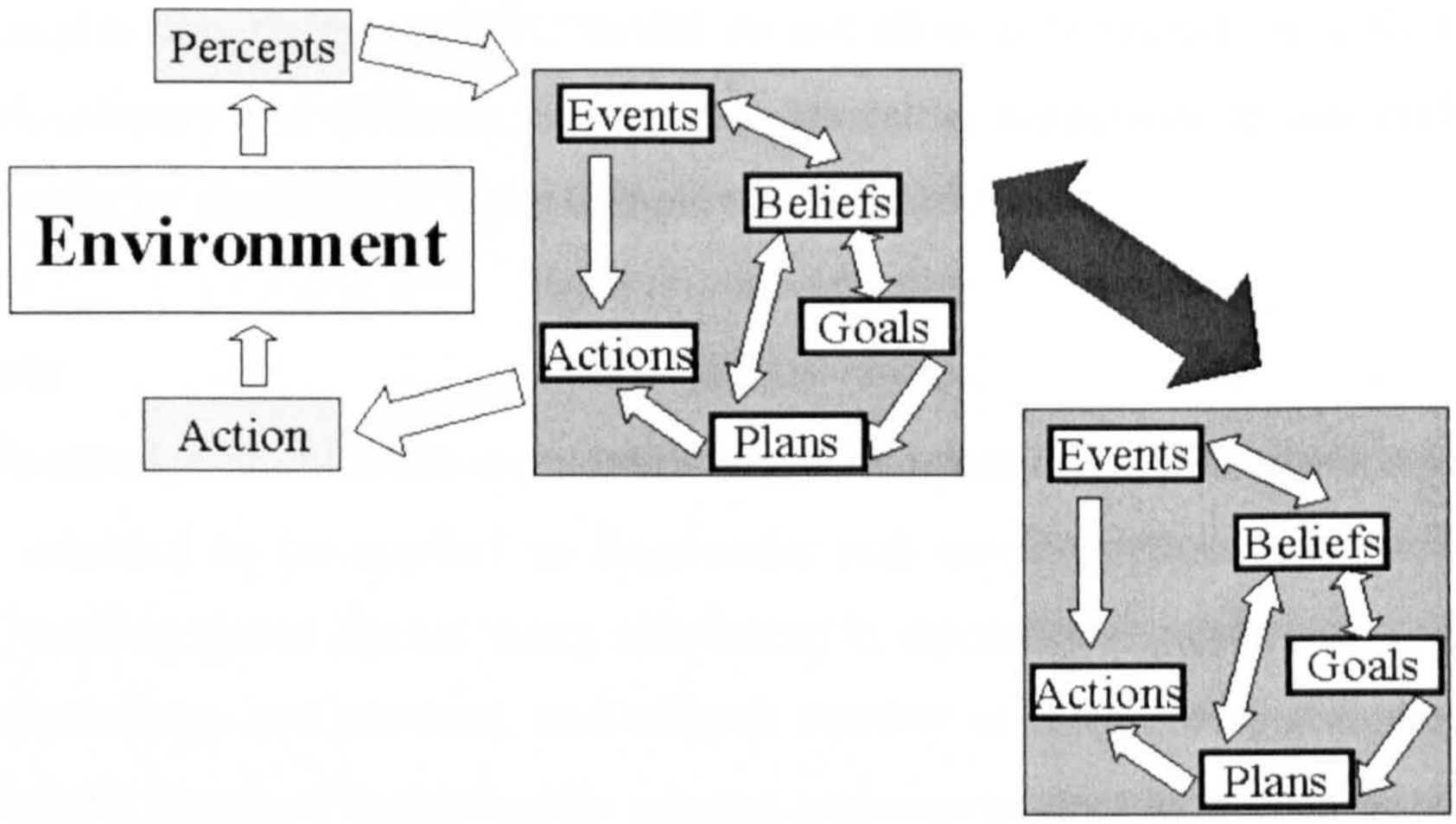


Figure 16 Components of the SAC architecture (depicting two agents, after Winikoff, 2001)

SAC agent's execution model can be described in 7 steps:

- percepts are interpreted and give rise to events (which are internal agent's events and not events coming from the environment, examples can include appearance of a new object in vicinity but also a tick of an internal clock)
- agent's beliefs are in turn updated
- events can create reflexive actions and possibly new goals
- goals are updated
- plan is chosen (if necessary)
- chosen plan is expanded
- an action is chosen, scheduled and performed

SAC agents have separate reactive and proactive execution cycles, with the reflexive actions triggered by events. Events in the context of this architecture are internal – they are defined as interpreted percepts of some significance to the agent and can also be generated inside the agent (for example by the agent's clock). Winikoff also allows the environment to be implemented as an agent, although it is not necessary. The developer is responsible for the identification of all agent's goals – "it is important for the developer to identify the top level goals of the agent as well as subsidiary goals which are used in achieving main goals." Additionally the agent may be equipped with a plan library from which it can select appropriate plans when necessary.

The principles underlying the SAC model do not allow it to include specific needs of system developers and different agent types. Therefore extensions to this architecture will generally be necessary in more complex applications.

3.3.3 Gaia

Gaia (Wooldridge *et al*, 2000) is a methodology for agent-oriented analysis and design. Gaia is intended to be applied to large-scale real world applications, with coarse-grained, heterogeneous agents using substantial computational resources, static inter-agent relationships and services, and a small number of different agents. Apart from addressing the issues of designing the internal structure of an agent, Gaia also presents ways of dealing with the societal aspects of multi-agent systems design. Gaia is meant to be neutral with regard to specific target domains and agent architectures. Gaia divides the process of modelling a multi-agent system into three stages: requirement capture, analysis, and design, inputs and outputs from the last two stages are presented as models (Figure 17).

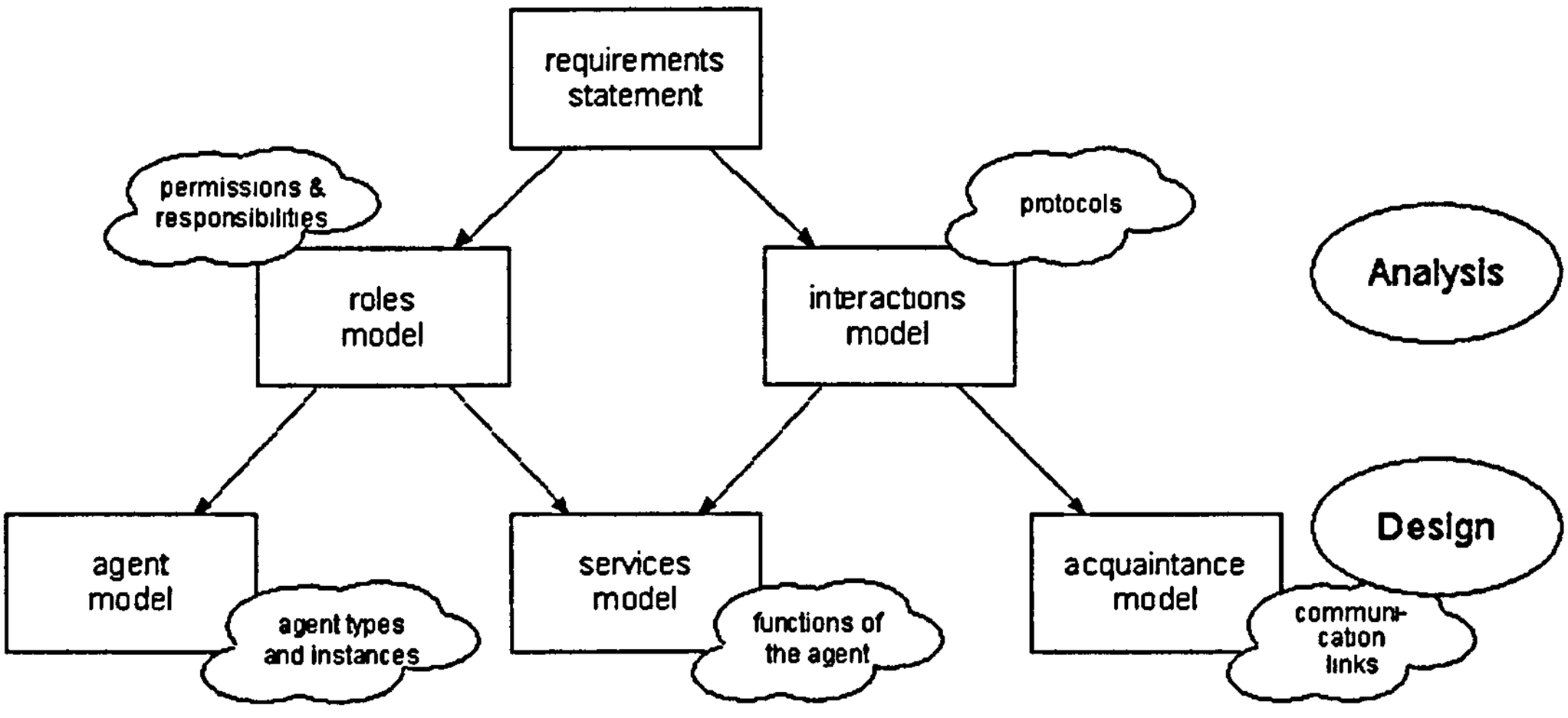


Figure 17 Gaia Models (after Wooldridge et al, 2001)

Although the Gaia methodology seems very elegant, this comes at the cost of a lack of detail. Most of the models only capture more formally what every agent designer would have done if faced with a task of designing such a system. Additionally from the point of view of applying Gaia to design a breed of animated agents Gaia falls short in this task as it is aimed at designing heterogenous agents, which share common goals. Additionally Gaia does not explicitly address the problem of situatedness, even though it is one of the key agent concepts.

3.3.4 Tropos

Tropos (Mylopoulos *et al*, 2001, Castro *et al*, 2001, Giunchiglia *et al*, 2001) is another popular software development methodology aimed at creating agent-oriented solutions. Tropos spans the whole design cycle of the system, capturing early and late requirements, analysis and architectural and detailed design. This is in contrast to Gaia, which only models the two intermediate stages. Tropos also proposes extensions to UML diagrams, which accommodate extensions required by multi-agent systems. The framework is generally designed to support development of multiplatform, 'componentised' software systems, such as e-business applications. Because of this however, the generic Tropos diagrams tend to contain large numbers of elements, which makes them complex and difficult to follow. The development procedure lacks more detailed transitions and consists mainly of a set of milestone models.

Other similar methodologies include MaSE (Wood and DeLoach, 2001, Raphael and DeLoach, 2000), which uses UML diagrams as part of the process, CASSIOPEIA (Collinot *et al*, 1996) oriented towards the design of a robotic soccer team and Prometheus (Padgham and Winikoff 2002) evolved in an industrial environment. This list is not exhaustive, most of the agent-oriented design methodologies however are still too biased towards specific solutions and immature to be applied to a problem of generating intelligent animated agents. For a survey of such methodologies see Iglesias *et al*, 1999 and also Kinny *et al*, 1996, a good comparison of several of these methodologies can also be found in Dam and Winikoff, 2003.

3.4 AuRA and Other Selected Robotics Architectures

Some of the issues tackled by the research in robotics are closely related to problems encountered when trying to create intelligent animated characters. While much of the work into construction of autonomous robots stills tries to address mechanical aspects of robot creation and problems with imperfect sensing and actuating, there is a substantial amount of overlap in both of these fields. Thus some of the results of research into robotic systems are interesting when building an architecture suitable for animation of autonomous avatars.

One such example is AuRA (Autonomous Robot Architecture) proposed by Arkin (Arkin, 1992, Arkin *et al*, 1993, Arkin 1998). The architecture consists of two parts: a reactive component built upon motor schema behaviours, which are analogous to animal behaviours, and a deliberative hierarchical planner. Each motor schema (see Figure 18) comprises a number of perceptual schemas, which fire when necessary stimuli are present in the perceived environment. Thus a particular motor schema contributes to the robot's behaviour when a set of conditions associated with it is supplied by the robot's surroundings. Examples of motor schemas may include move-ahead, move-to-goal avoid-static-obstacle, escape, follow-the-leader and other behaviours. Eventually all contributing motor schemas are translated into corresponding action vectors (comprising orientation and magnitude), normalised and added using vector summation and the result is passed to the motors controlling the robot.

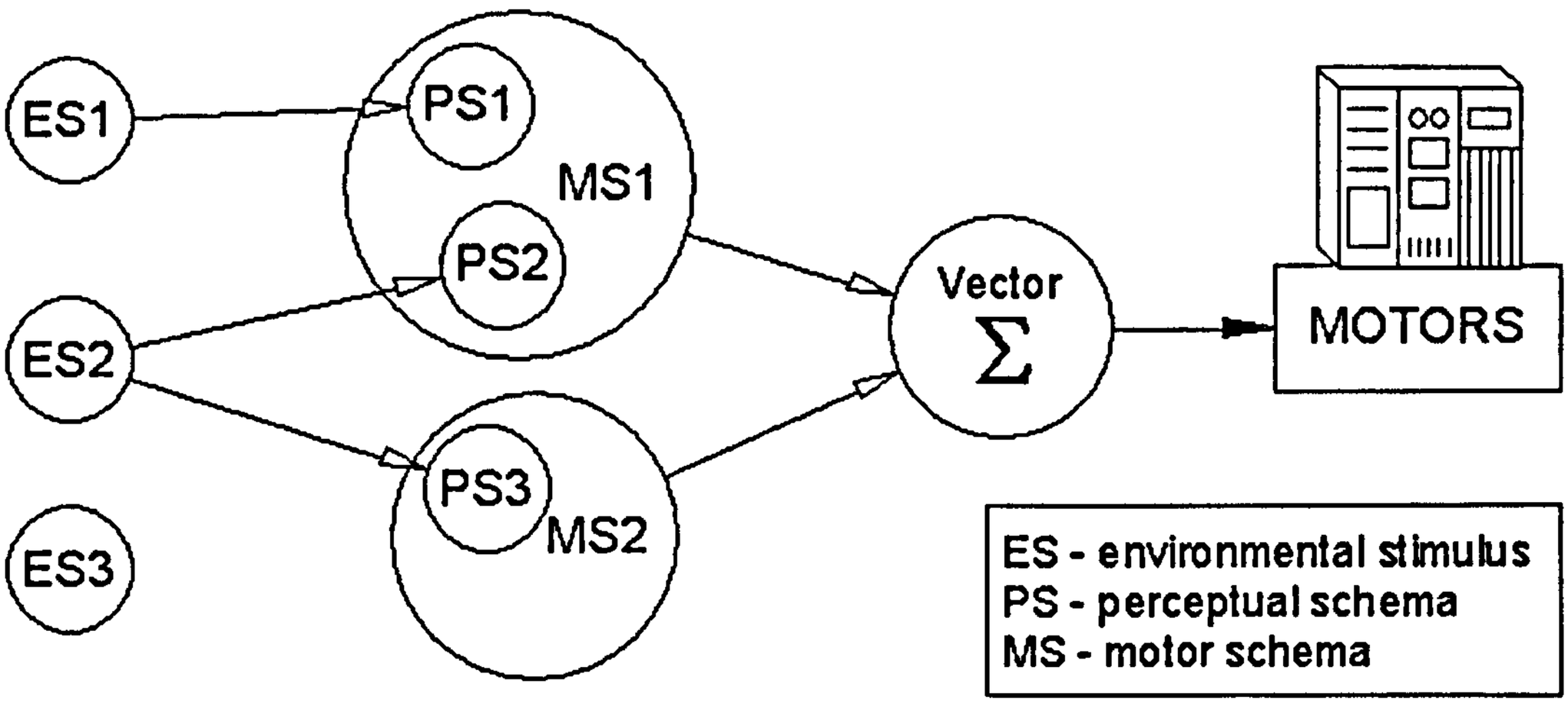


Figure 18 AuRA Reactive Framework (after Arkin, 1998)

The deliberative part of the AuRA architecture is a hierarchical planner consisting of a mission planner responsible for managing the high level goals, spatial sequencer constructing navigational paths from cartographic information stored in the long-term memory and plan sequencer which translates paths into motor behaviours. The planner communicates with the schema controller (middle part of Figure 18) and intervenes only when the reactive module fails to cope with the task. The architecture has also been equipped with two types of learning capabilities. The first one is online adjustment of schema gains and parameters controlling the schemas. These modifications control the strength with which each schema can contribute to the general behaviour of the system.

The second learning framework based on genetic algorithms allows building families of robots with varying fitness functions, thus optimising performance in regard to speed, safety or time constraints.

The AuRA architecture demonstrates that even relatively simple framework based on several reactive behaviours can generate a realistically behaving intelligent system. Extra units such as a planner help resolve critical situations and manage goal-directed behaviour and the addition of learning capabilities allows the designer to easily adjust the working parameters of the system and also to impose more complex constraints on the overall performance of the system.

Another useful approach to designing robotic systems was presented by Ishiguro and his colleagues (Ishiguro *et al*, 1999). Although the proposal is also a hybrid robotic architecture, they focus their attention on presenting a development process applicable to the design of robotic systems. Only then do they move onto creating a functioning robot. The approach is based on situated modules, that is condition-action pairs encoded in a procedural language. Kumar (Kumar, 1998) on the other hand explored the possibility of extending the BDI concepts in the context of robotic agents and proposed a much more theoretical framework for reasoning, acting and rule acquisition for reactive behaviours.

3.5 Reinforcement Learning

When performing fully automated acquisition of high-level animation actions it is desired that a user only define a goal for the learning task without intervening in the way the action is performed. Reinforcement Learning (RL) methods (Sutton and Barto, 1998, Mitchell, 1997) fulfil these criteria. RL is a machine learning technique in which an agent learns by trial and error which action to perform by interacting with the environment. Models of the agent or environment are not required. 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. 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, Watkins and Dayan, 1992). 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 favours 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. Thus the cumulative discounted reward is given by

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

where $\gamma \in [0,1)$. The Q-value is updated at each step using the update equation (1) for a deterministic Markov Decision Process (MDP) as follows

$$Q\ (s_t, a_t) \longleftarrow r_{t+1} + \gamma \max_a Q\ (s_{t+1}, a) \quad (2)$$

A sequence of actions ending in a terminal state (known as an absorbing state) is called an epoch. 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 and Saher, 1999).

Reinforcement Learning has been applied to create successful board games implementations (Schraudolph *et al*, 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), Sutton (1996) succesfully applied RL to various optimisation tasks including control of the Acrobot – a two-link robot actuated only at the second joint and Boone (1997) compared Q-learning with other control methods for the Acrobot task including the A* search. Recently Tedrake and Seung presented a Reinforcement Learning technique for expanding a controller for the planar one-legged hopping robot (Tedrake and Seung, 2002). Q-learning-based solutions have also been modified and adapted. Examples include ant systems (Gambardella and Dorigo, 1995, Monekosso and Remagnino, 2001, Monekosso *et al*, 2002) or reward shaping (Ng *et al*, 1999) a technique in which additional rewards are used to guide the learning. Recent applications of RL to character animation have been presented before. A survey of Reinforcement Learning techniques can be found in Kaelbling *et al* (1996),

an excellent tutorial on Reinforcement Learning techniques was published by Harmon and Harmon (1996) and Touzet (1999) describes techniques for combining Q-learning and neural networks in the context of robotics.

3.6 Rapid Prototyping and Motion Capture-based Techniques

Although not directly related to the creation of intelligent characters, motion capture-based techniques allow for creation of realistic animation without the need for keyframing or behaviour simulation through application of techniques which allow for generation of new motion from a database of clips. This can also serve as an interesting technique for visualisation of behaviours generated using intelligent techniques, in cases where low-level motion generation in impractical. Similarly rapid prototyping techniques offer an opportunity to quickly sketch an animation sequence without the need for simulated motion. Some of the motion learning techniques can also be included in this category. Therefore this section presents the most recent advances in both domains.

Fang and Pollard (2003) proposed a system for fast generation of motions for characters having from 7 to 22 degrees of freedom using physical simulation. Another recent system for creating and editing of character animation was presented by Dontcheva *et al* (2003). The system is capable of rapid prototyping of expressive motion and can work in real time. It is based on a motion capture framework with immediate feedback displayed on a large screen. Similarly the system presented by Lee and his colleagues (Lee *et al*, 2002) allows the user to combine clips from a database of mocaped data by identifying possible transitions between motion segments. The system works in real-time and can additionally be controlled by sketching required motions or by acting them in front of a camera. The generated results are comparable to recorded human motion.

Zordan and van Der Horst (2003) presented an new solution for mapping motion capture using optical motion capture systems to joint trajectories for a fixed limb-length skeleton based on virtual springs. This allowed them to generate smooth and uniform motion applied to virtual avatars.

Neff and Fiume (2003) focused on improving existing animation to make it more aesthetically pleasing, they proposed three tools for varying timing and shape of motion

and Kovar and Gleicher (2003) proposed a novel technique for motion blending – a technique which allows to create new motions by combining multiple clips according to some criteria.

Li with his colleagues (Li *et al*, 2002) described a system for synthesis of complex human motion (dancing) from motion captured data. The system learns so called motion textons (repetitive patterns in complex motion) and their distributions and can synthesise new motion. A similar concept was introduced by Liu and Popovic (2002). They presented a system for rapid prototyping of realistic (highly dynamic) character motion from a simple animation provided by an animator. The system learns an estimator for predicting transition poses from examples taken from a database of motion captured motions. Similarly Pullen and Bregler (2002) proposed a system generating motion from a motion capture library based on a small number of keyframes sketched by the animator. The concept is based on the observation that joint movements are often correlated and thus the missing data can be derived from real motion.

Another real-time motion synthesis framework was created by Arikan and Forsyth (2002). The system allows to generate motion from mocap by pasting sequences. Although the system works in real time it requires precomputation of the clips (they quote 5 hours for a library of 60-80 short motions). Another interesting system is that presented by Bregler *et al* (2002), where they propose a 'cartoon capture and retargeting' technique which allows to capture expressive motion from traditional 2D cartoons and apply it to 3D models and 2D drawings.

All of the above systems allow for fast creation of realistically moving animated characters without the need for sophisticated AI-based techniques. However the task of directing the motion is still left to the animator and the systems cannot cope with animations with inter-character interaction, or where it is necessary to animate avatars in a complex environment.

3.7 Summary

This chapter presented main areas of research necessary to consult when building systems able to control intelligent characters. The most important part of the review

included animation frameworks and architectures, which can be divided into three categories. Some of the presented solutions focus on creating a small number of very intelligent characters (C4 and its extensions, Funge). Such architectures are usually geared towards embracing a user in a compelling interactive experience, such as training a pet or playing against a clever computer controlled opponent in games. This usually means that the underlying system is very complex and specialised and characters are difficult to duplicate. Second group of animation architectures includes physics-based systems, which are capable of generating characters with very complex motor skills. Such systems generate realistic motion using physical simulation, however currently the level of intelligent behaviour generated by such systems is low. Additionally physical simulation tends to be computationally demanding and controllers offering multiple behaviours are not as yet widely popular. Finally there is a number of crowd systems which allow to generate numerous scene participants, usually at a cost of partially sacrificing autonomy (predefined scenarios), motion realism and/or creation time (mocapped sequences, keyframed motion) or character intelligence (state machines, reactive behaviours). Other systems discussed in this chapter included cognitive systems trying to mimic the function of human brain, autonomous agents, Reinforcement Learning and rapid animation prototyping. All approaches presented in this chapter can contribute to creation of an architecture which can automatically generate complex character animation. Although many animation architectures present a great potential, they are usually too specific to generate multiple crowd participants able to interact with each other and the environment. The next chapter will now move on to propose an intelligent system building on some of the presented solutions and extending them.

Chapter 4 – The FreeWill Prototype

The previous chapters presented an overview of the current research in the field of digital character animation, intelligent architectures, agent-oriented design methodologies and also described existing industrial systems and 3D animation packages. The survey demonstrates that industrial solutions offer high-quality capabilities for the creation, texturing and rendering of virtual scenes, and also offer specialised procedures for incorporating digitised human motion into the animation sequences. This is an important factor contributing to improve the realism of the digital scenes. The drawback of the existing professional animation systems is that they rely on manual creation of animation, keyframing techniques or motion acquired from real actors. They fail to deliver tools for automatic prototyping and creation of scenes comprising high numbers of avatars. Any automation of the process employed to create animated crowd scenes is usually limited to the creation of kinematics-based motion and in some cases to the use of state machines to speed up the creation of different behavioural schemas. In such cases the assignment of states and related motion must always be conducted manually, characters are not self-aware, autonomy is very limited and even collision detection is not always present.

In contrast to this, systems for character animation created by the research community offer a number of complex AI techniques suitable to improve the way character animation is made. This for example includes specialised architectures supporting automatic control of avatars performing simple reflexive and complex pre-planned actions. Learning and emotions are other important features. Hence the scientific frameworks are able to create highly autonomous, interactive characters, which can explore their environment, perform complex actions and even learn new ones (see Chapter 3). The characters are capable of maintaining their own representation of the world, they can interact with each other and the external user, have complex motor and navigation capabilities. In most cases however, research methods focus on building custom systems and their characters are animals or simplified human-like creatures, because such representation is easier to animate and render. Integration with existing tools is rarely provided and motion is usually pre-generated. Easy generation of crowd scenes with anthropomorphic beings is not always possible and again dedicated systems

must be written to create animation sequences with characters endowed with additional mental, emotional or motor skills. Additionally, very few authors of these systems take into consideration the fact that an architecture able to accommodate complex human-like look and behaviour will always be a challenging engineering task. Thus even though used research architectures are complex and achieve their goals, they are very hard to emulate and extend, and usually remain rather closed systems. Application of modern software engineering principles and design methodologies would certainly improve the readability, reuse and extendibility of such architectures and possibly offer a more modular design and easier implementation.

It is therefore clear that extensions to the existing animation packages made by incorporating ideas from industrial animation systems, research proposals, and concepts from software engineering would allow to build a far more powerful animation architecture. The AI fields, which appear particularly useful for this purpose, are intelligent architectures, autonomous agents and machine learning, since they allow for enhancement of the automation level. On the other hand, software engineering concepts and good software design practises would certainly allow for creation of a system which is more robust, easy to implement, reuse and extend. Finally, if the proposed system allowed for creation and rendering of a virtual scene within a professional animation system, the high quality of the resulting animation and flat learning curve for existing users would be assured. To provide this, the process of creating the animation sequences involving animated characters might be conducted in an external module feeding the Artificial Intelligence and returning to the animation package a set of necessary motion commands for each avatar (Figure 19). Thus the proposed extensions could be incorporated into existing animation packages by adding an additional module on top of the existing components and allowing it to communicate with the animation engine. Figure 19 proposes the extensions, which could be made to the existing animation packages depicted in Figure 5 to address the need for intelligent behaviour.

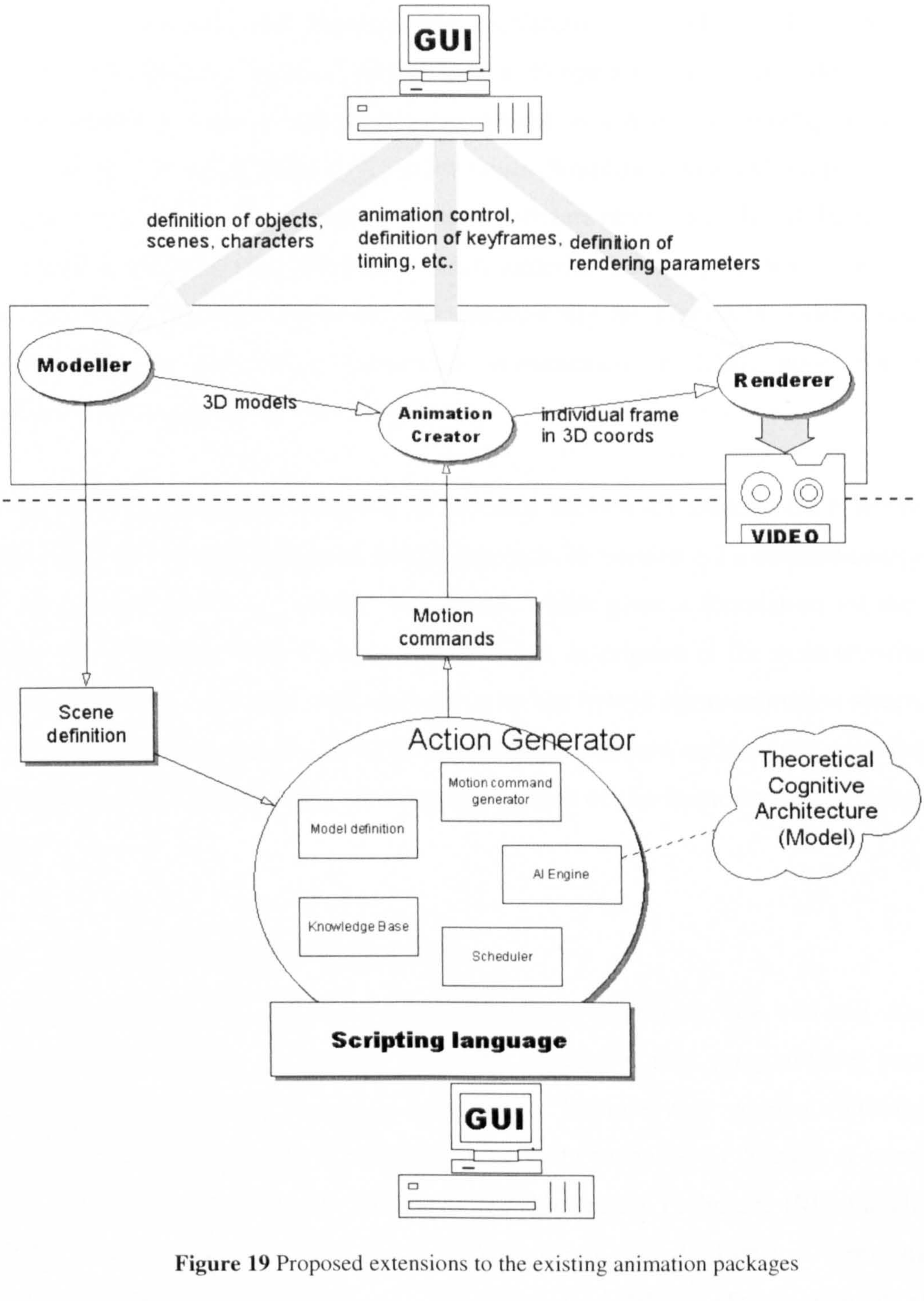


Figure 19 Proposed extensions to the existing animation packages

This chapter presents an intelligent architecture, FreeWill, which combines concepts from Artificial Intelligence with a strong software framework, that extends the capabilities of existing animation packages. FreeWill has been designed to automate the process of creating scenes involving many interacting human-like characters and incorporates goals, plans and actions thus allowing for collision-detection, self-

awareness, autonomy, and learning. The architecture, is built based on two of the systems presented in Chapter 3. These are the Funge's Cognitive Architecture (FCA) which addresses many problems closely related to creation of intelligent animated creatures and the agent-based SAC architecture (Simplified Agent Concepts) proposed to make agent notions more approachable to software developers. It will be shown that by merging and extending the ideas, which underlie these approaches, a more robust and flexible architecture can be created. Additionally the process is underpinned by a strong software engineering framework documented in UML upon which the implemented prototype has also been based.

The chapter is divided into a number of sections. Section 4.1 presents the benefits from applications of the agent-oriented design approach. In Section 4.2 a detailed comparison of the FCA and SAC concepts is introduced, which gives a foundation for the main ideas underlying FreeWill. Then, in Section 4.3, a description of the main constructs of the architecture is provided, with main focus on the hybrid agent-animation structure of the system. These proposals are underpinned by a software architecture modelled and presented using UML. Finally all main subsystems of the framework are presented in detail.

4.1 Benefits of Agent Technology

Only about 10-15 years ago the dominant technique in software labs was still structural programming. In the late eighties and early nineties a new programming paradigm called object-orientation was born and quickly became very popular (although the technology was actually first developed in the early seventies by Xerox). A number of techniques and methodologies, such as Object Modelling Technique (Rumbaugh *et al*, 1991), Responsibility Driven Design (Wirfs-Brock *et al*, 1990), Object-Oriented Software Engineering (Objectory – Jacobson *et al* 1992), Object-oriented Design (Booch, 1991), the Fusion Method (Coleman *et al*, 1994) were created, thus building strong foundations for the software engineering techniques and implementation of new, better programming languages. Object-orientation introduced or extended a number of very useful and interesting ideas such as inheritance, polymorphism, encapsulation, abstraction, easier code reusability or component-based architectures. It is currently the main technique for code design and implementation, and is supported by a number of

tools, methodologies, visual modelling applications and most importantly modern programming languages. Object-orientation (OO) was a natural step towards enhancing a single programmer's creativity and the ability to control a growing amount of lines of code.

However the object oriented programming did not fulfil all its promises. Jennings (Jennings, 2001) enumerates the following shortcomings of the OO approach:

- objects are passive in principle and can only be invoked by other objects (or actors); they have no initiative, this means that objects do not provide behaviour activation encapsulation – so once a method is called it does what it has been programmed to do
- objects do not provide a sufficient level of abstraction for complex systems – the level of granularity is too high and the interactions between entities must be pre-defined in a detailed way
- the OO approach does not provide sufficient support for specification and management of dynamic organisational relationships (the only such OO mechanism is inheritance hierarchy).

Therefore with the fast growth of the processing power and the complexity of software, some software engineers tried to introduce a yet higher level of abstraction for developing complex system, a novel, more intelligent and versatile type of applications assisting the users in many every day routine tasks. The entities of that approach were called agents and the technology is known as agent orientation. The concept was coined by Shoham in the early nineties (so in fact his idea was parallel to the growth of OO). In his most widely known paper (Shoham, 1993), Shoham introduced *agent-oriented programming* (AOP). A more theoretical view on the problem was also presented by Rao and Georgeff (Rao, and Georgeff, 1992) and agent-orientation is currently one of the most rapidly growing fields of research in the domain of Artificial Intelligence and software engineering. Despite the fact that all early applications of AOP were implemented in object-oriented languages, the underlying concepts of this technique are very different. Agent orientation further removes coupling of software components, introduces action encapsulation (as opposed to data encapsulation in OO), presents interaction as an active process (interaction relationships are fixed in OO) and promotes

flexible behaviour of agents (by adding both reactive and proactive actions) (Jennings, 2001).

Jennings argues that an agent oriented approach (Shoham, 1993, Burmeister, 1996, Wooldridge, 1997, Wooldridge and Jennings, 1998, Nwana and Ndumu, 1998, Jennings, 1999, Jennings, 2000, Jennings 2001, Wooldridge and Ciancarini, 2001) is much more suitable for the development of systems which can handle the necessary complexity. The main advantages of AO according to Jennings include:

- reduction of coupling of components – each agent can be a self-contained entity and will exchange messages with other agents only sporadically (when considered on the software level)
- no need to manage control – the agents are active all the time, hence they can respond to signals from the environment as they arise
- strong support for high-level interaction which additionally can be initiated by the agent
- flexible behaviour – agents support both reactive (responding promptly to external events) and proactive (making decisions and deciding what to do to achieve the goal) behaviour.

Winikoff (Winikof, 2001) also notes that the inherent property of an agent is having many goals and many ways of fulfilling them and that agents make their decisions according to the actual feedback from the environment.

Agent orientation has been used successfully to deliver applications in different domains, such as traffic and transportation (Burmeister *et al*, 1997), real time tracking (Horling *et al*, 2001), or network resource configuration (Hayzelden and Bigham, 1998), to name but a few. More examples can be found in Mondal and Jain, 2001, Jennings and Wooldridge, 1998, Burmeister *et al*, 1998, Burmeister *et al*, 1997, Georgeff and Rao, 1996. Some of the most prominent agent design methodologies have been presented earlier.

However, despite there being a few implementations of agent languages and platforms (Rao, 1996, Mayfield *et al*, 1996, JACK, 2003, T-Tool, 2003, FIPA, 2003, AgentTalk –

Winikoff, 2003), there is still no widely recognised standard. Additionally new ones are constantly being contributed. Even assuming that one could create agent architectures still based on object languages, even an agent design methodology has not been agreed upon. Finally, apart from the lack of methodology, the agent community is still working on the design notation. Agent UML (AUML, 2003) seems to be the main candidate, but the definition of this modelling language is still far from completion.

Given the complexity and intrinsic structure of the problem presented in this work (many autonomous avatars, complex external environment, interaction with other packages) it appears that the agent-oriented approach is an appropriate one for the development of the FreeWill project. However due to the lack of mature tools supporting such approach a mere application of the existing techniques might not be sufficient to conduct full design and implementation. Given the maturity of the object-oriented techniques, including programming languages and modelling tools an object-oriented approach has been chosen for the implementation and the agent concept has only been used to support the high-level design of the system. Hence FreeWill tries to merge useful concepts from the agent-oriented approach with a sound software engineering framework, exploiting ideas from both agent orientation (design principles, structure of the system) and object orientation (implementation). The next section presents the hybrid agent and animation concepts underlying the system's design.

4.2 Comparison of the FCA and SAC Architectures

In this section specific features of the FCA architecture proposed by Funge and the agent-based SAC concept are compared before discussing how a synthesis and extension of the two might be achieved.

As suggested in Chapter 3, the FCA architecture is strongly biased towards building an intelligent character. It addresses the low-level concepts of character creation including physical representation, motion and navigation as well as the cognitive features. The SAC framework is more concerned with creation of multiple autonomous agents and uses some concepts from the BDI approach. Both architectures stress the importance of goals, reactive and proactive behaviour, and the need for some representation of the character's knowledge (beliefs). They are also built based on the sense-think-act cycle –

a standard approach to building systems which have to communicate with the environment. But whereas in FCA the sensing process always updates the character's world model in SAC it first gives rise to internal events, which may in turn update the agent's beliefs. FCA is a dedicated animation architecture, SAC is a general-purpose model for designing and implementing situated agents. What follows from this is that only FCA includes animation concepts such as geometry or inverse kinematics. FCA also addresses the problem of learning in the form that it may be necessary for some characters to be able to learn new behaviour. Following the BDI's statement that agents should in general maintain a list of predefined plans SAC includes the concept of plan libraries. On the other hand FCA allows to pre-define behaviour (in the form of reactive behaviour rules or state machines) but for the goal-directed behaviour recommends doing searches in (perhaps pruned) situation trees. Similarly the interaction aspect of SAC remains in strong contrast with FCA which does not provide any explicit mechanism to allow interaction between the animated characters (a possibility of using a communication protocol is only mentioned).

The table below provides a summary of these and other important features of both systems.

Table 3 Comparison of the architectures

Architecture name	**FCA**	**SAC**
Design paradigm	Character oriented	Agent oriented
Recognition for the animation aspect	Yes	No
Representation of the character's knowledge	Internal world model + domain knowledge	Beliefs (a knowledge base)
Autonomy	Yes (based on action search, planning and reasoning)	Yes (based on planning and reasoning)
Learning	Recognition for learning	No explicit emphasis on learning
Reactive *and* pro-active behaviour	Yes	Yes
Interaction with other characters	Implicit (perceived clues)	Explicit
Main implementation cycle	Sense-think-act	Sense-think-act
Sensing/perceiving	Updates the world model	Gives rise to (internal) events
Planning	Search on atomic actions	Plan libraries
Ease of implementation	Not obvious	Easy to follow development model
Goals	Yes	Yes
Formalism	Situation calculus	Linear logic

4.3 The FreeWill Architecture

The main goal of FreeWill is to propose an architecture suitable to create intelligent and realistic animation in the form of crowd scenes. Therefore it contains elements found in both animation-driven systems and distributed (multiagent) solutions. The system incorporates the concept of agents and plan libraries from the SAC model and it also addresses the need for representing the avatar by giving it a physical body, ability to move and to maintain an internal world model. Additionally, these concepts are extended by separating the agent from the environment and introducing external events as a means of providing a uniform way of interacting with other objects and agents.

Each avatar participating in the animation consists of an intelligent agent implemented as a modified SAC agent together with a body layer, which is responsible for handling the visible part of the agent (see Figure 20).

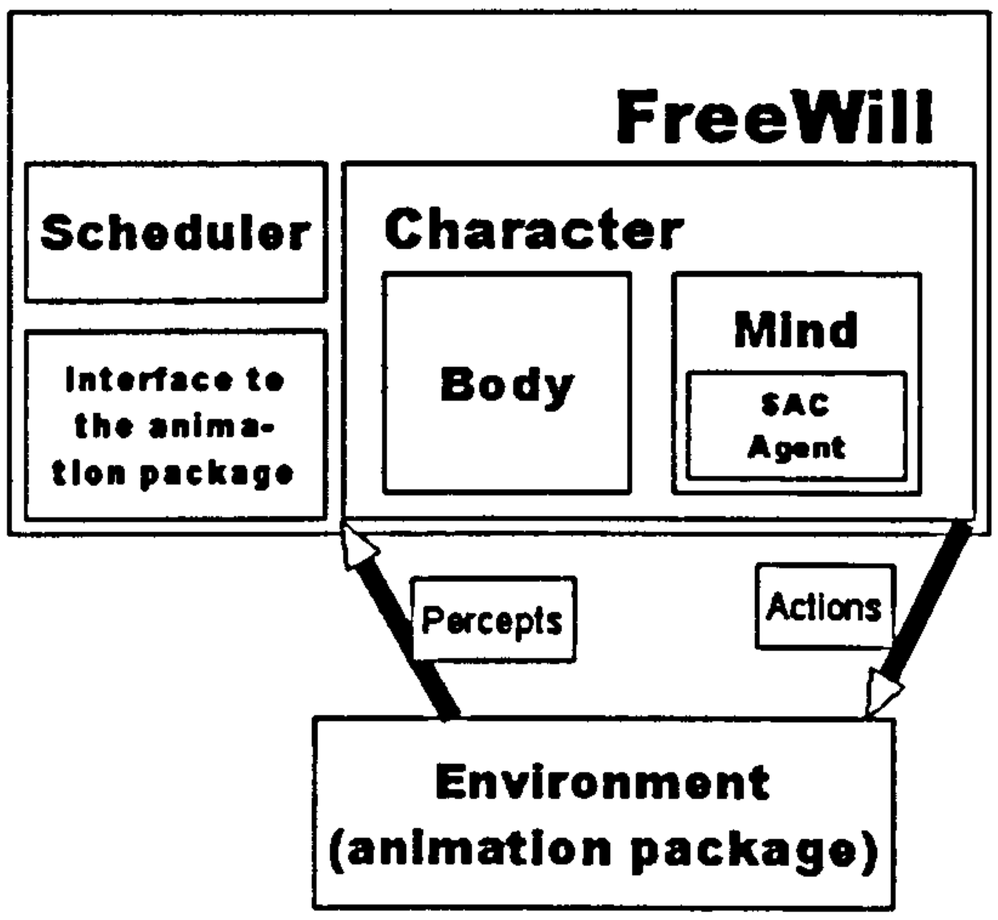


Figure 20 The FreeWill framework

The geometry, physics (kinematics) and behavioural layers are incorporated from the FCA architecture. Each avatar is ultimately modelled as a set of simple geometrical shapes controlled by the forward and inverse kinematics (implemented by the existing animation package and only visible after rendering). All characters follow simple patterns of behaviour, which allow them to be identified as anthropomorphic creatures. Additionally, in accordance with Funge's view, a substantial part of the character's knowledge is the internal world model, which is updated by sensing. On the other hand agent concepts allow to build a more specialised 'mind' – the agents store plan libraries

instead of creating plans from scratch every time a new plan is needed, new plans are constructed only if there is no existing template, and characters must interact. The interaction includes actions executed by two different avatars and as such is fully modelled by reaction to events, which are created during the sensing process. This is similar to the way humans interact with each other (using hand-eye co-ordination).

The agent components are presented in Figure 21. The agent consists of a knowledge base, a planning unit and a stack of actions currently scheduled for execution. The knowledge base maintains the internal world model and all the agent's goals (primary and secondary). Primary goals represent an agent's ultimate aim of the simulation and would normally be assigned to it by the animator. This may include end positions (get to the end of a sidewalk) or resulting states of the animation (kill your enemies). Secondary goals on the other hand are goals created by the agent during the course of the simulation. Example secondary goals would be 'shake hands with a friend' or 'look at a watch while waiting for a bus'. The planner handles the task of pre-processing the desired course of action and also manages the plan library – a list of high-level, complex actions, which the agent can perform. An example of such an action may be a handshake. In each processing cycle the plan is reviewed and if necessary the action stack may be updated, then the last action from the plan is chosen and executed.

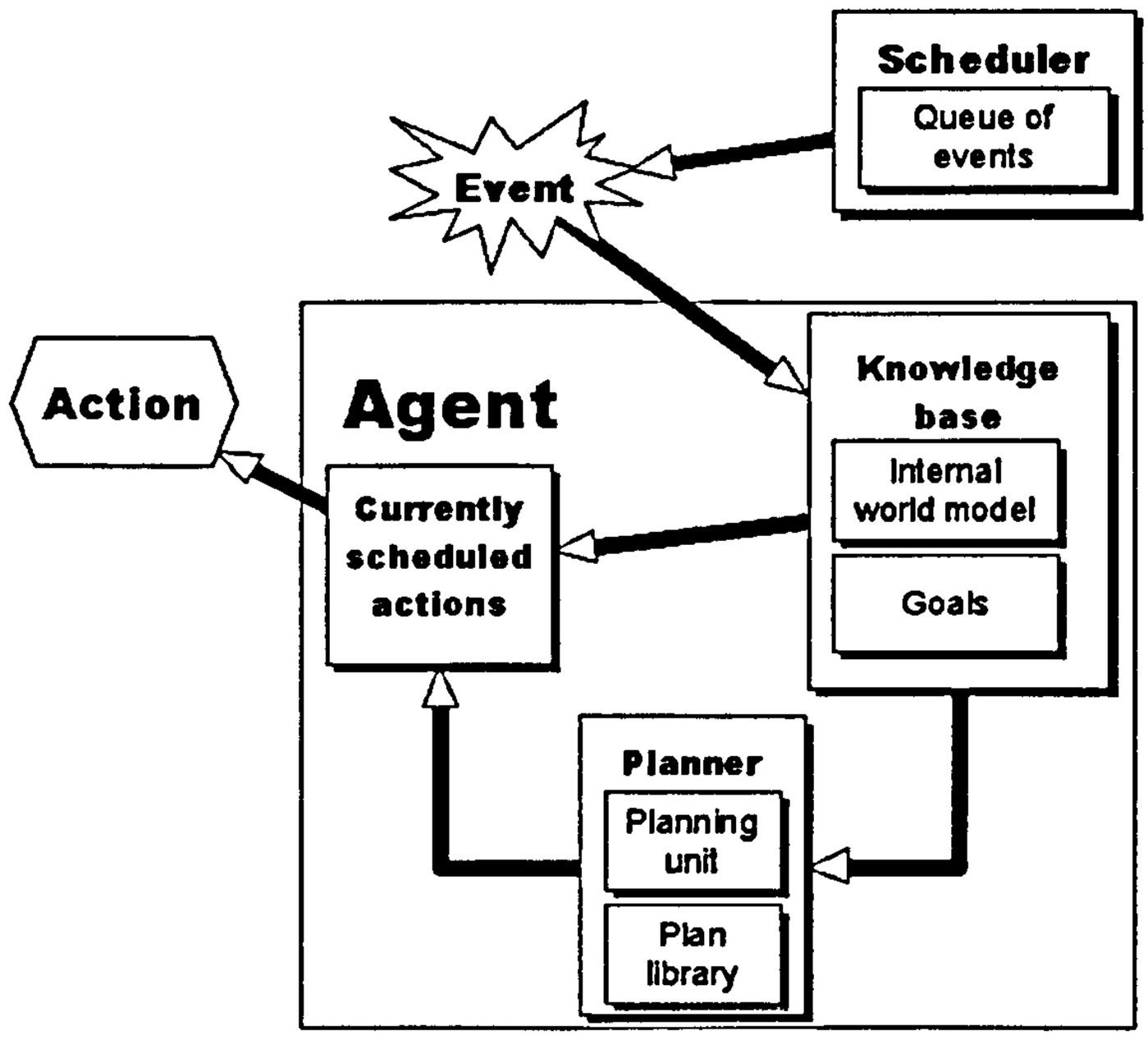


Figure 21 The agent component of FreeWill

The main difference between the SAC model and FreeWill is that events to which the agents respond are not created inside the agent but are external – come from the scheduler. They are however submitted to the scheduler queue by the agent when it decides to update its knowledge base or when an action must be performed. External events allow for a greater uniformity of the architecture – both acting and sensing events can be modelled in this way and similarly scene objects can be handled following this scheme. It is also in line with the principle of easy implementation. Event-based systems with external events are popular and straightforward to implement.

The scheduler serves as an event dispatcher and sequencer. All events are managed globally, scheduled and then distributed to appropriate avatars. This allows for greater flexibility (an avatar may or may not wish to 'see' the results of its actions) and efficiency (simple data access collision resolving) and also provides a uniform way of both maintaining the avatar's awareness and communicating with other avatars and possibly world objects too. The FreeWill execution cycle proceeds as follows:

1. A sensing action is executed by the avatar
2. Agent's beliefs are updated
3. Current plan is evaluated, if there is a need to perform a reflexive action the current action queue is cancelled and a new action gets submitted
4. Goals are updated if necessary
5. Plan is updated if necessary
6. Last action is chosen and submitted to the scheduler, when the avatar is later asked to execute the action it also submits a new sensing action to be put in the event queue (Figure 21)

Point 3 indicates that FreeWill agents have common reactive and proactive execution cycles unlike the SAC agents. The main features of the proposed architecture are gathered in Table 4:

Table 4 Attributes of the FreeWill architecture:

Architecture name	**FreeWill**	**FCA**	**SAC**
Design paradigm	Character oriented	Character oriented	Agent oriented
Recognition for the animation aspect	Yes	Yes	No
Representation of the character's knowledge	Internal world model + knowledge base	Internal world model + domain knowledge	Beliefs (a knowledge base)
Autonomy	Yes (based on action search and planning)	Yes (based on action search, planning and reasoning)	Yes (based on planning and reasoning)
Learning	Off-line learning of new actions	Recognition for learning	No explicit emphasis on learning
Reactive and pro-active behaviour	Yes	Yes	Yes
Interaction with other characters	Implicit (perceived clues)	Implicit (perceived clues)	Explicit
Main implementation cycle	Sense-think-act with external events	Sense-think-act	Sense-think-act
Sensing/perceiving	Updates the world model	Updates the world model	Gives rise to (internal) events
Planning	Plan libraries + construction of plans	Search on atomic actions	Plan libraries + constr. of plans
Ease of implementation	Easy to follow (UML documentation exists)	Not obvious	Easy to follow development model
Goals	Yes	Yes	Yes
Formalism	None	Situation calculus	Linear logic

4.4 The FreeWill Framework

An animated sequence consists of characters (avatars) interacting within a graphically defined setting. As explained earlier, avatars are modelled as agents with a geometrically implemented body (all physical objects are represented as 3D geometrical shapes). The setting is a virtual environment, for instance a city street populated by avatars walking in either direction. The basic requirement is for the avatar to be able to walk towards a set destination (goal-oriented behaviour) while avoiding collisions. The primary goal must always be assigned to each avatar, if any additional goal is used, they must also be explicitly declared by the user. The activities which the avatars exhibit in order to fulfil their goals are called actions and can include simple movements such as joint rotation but also complex high-level sequences, for example a handshake. Characters performing actions which require explicit interaction between two avatars (handshake) shall be called participants or "friends".

Subject to fulfilling its goals, an avatar's behaviour is otherwise autonomous. Characters deliberate the manner in which the primary goal is fulfilled and the animation would look different if generated multiple times. The action is simulated by the FreeWill engine, and the result is passed onto the animation package. The interaction with the animation package can take the form of a script – Figure 22 (fragment of the handshake script, more examples can be found in Appendix C) or stepfiles (in the case of 3D Studio Max, see Amiguet Vercher, 2000) or as direct communication via the COM interface. In the example script shown in Figure 22 two new keyframes for Avatar1 are created – first two lines define the current position as the starting keyframe (time 0), next the time is advanced by 10 units and the avatar right arm is rotated by 30 degrees along the local z-axis. A new keyframe is then defined, time is advanced again and the right forearm is rotated by 80 degrees (bending the elbow). Again a keyframe is defined. After the scripts have been executed, the animation package renders each frame and produces a video of the simulated interaction of the avatars. A scene from one such video is shown in Figure 23 (characters avoiding each other on a sidewalk).

```
biped.AddNewKey LarmCont1 0
biped.AddNewKey RarmCont1 0
      sliderTime = 10
rotate RForearm1 30 [-1,0,0]
biped.AddNewKey LarmCont1 10
biped.AddNewKey RarmCont1 10
      sliderTime = 20
rotate RForearm1 80 [0,0,-1]
biped.AddNewKey LarmCont1 20
biped.AddNewKey RarmCont1 20
```

Figure 22 Sample script for generating avatar behaviour

For the purpose of modelling the presented architecture, a number of classes and collaborations have been identified.

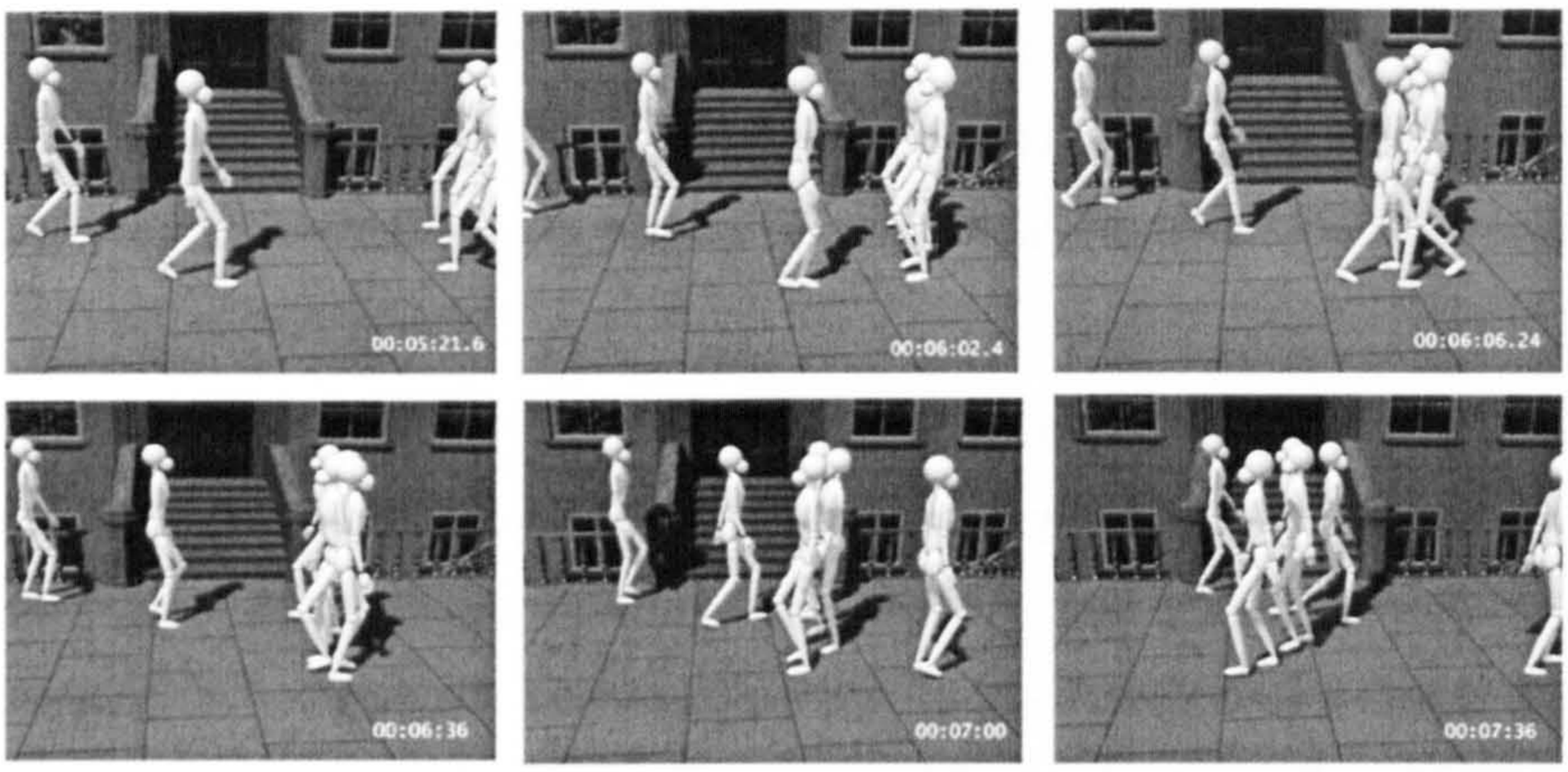


Figure 23 Avatar interaction

The class structure underpinning FreeWill is depicted in Figure 24, which presents a UML model of the system. As shown in Figure 24 the principal elements of the system are:

- *World* comprising all physical *objects*, including avatars, participating in the scene. Details stored for each object include a complete description of shape, dimensions, colour, texture, current position etc, sufficient to render the object.
- *Avatar*, which consists of a physical *body* together with a *mind*, instantiated as a separate object (on-board "brain") for each avatar, this structure corresponds with the Character component from Figure 20. The body provides the mind with all necessary sensing and actuator services, while the mind itself is responsible for perception (interpretation of information) and the issue of appropriate motion commands based on goal planning. As a subsystem, the mind engine is built of an *action planner*, a *motion controller*, and a *knowledge base* storing *goals* and *facts*, and the avatar's *private world model* (which represents the fragment of the virtual world currently seen and remembered by the avatar and is implemented as a subset of the world objects). The knowledge base is currently implemented as a collection of goal, fact and world objects, which in turn consist of methods and attributes. For goals this includes the type of goal and its parameters (e.g. end position) and facts are lists of items (e.g. avatar's friends). The world objects stored in the knowledge base can be used to access object information such as colour or position. Such structure could however be easily expanded to contain information in a form of logic clauses. This would facilitate the use of logical inference engines to make decisions concerning large numbers of goals and facts.

- A *Scheduler* based on discrete event simulation and a queue handler enabling the autonomous behaviour to unfold within the virtual world by passing control to appropriate world objects (including avatars) according to the *event* which is currently being processed.
- There is also one external component used to generate the final animation – the animation package or more generally *visualisation engine* – this part of the system is responsible for displaying the world model and the interacting avatars. This can be for example performed by the package 3D Studio Max as described above. The system could also interface other products and other formats, *e.g.* those using motion capture files. The visualisation engine must also allow for rendering the scenes and for saving the final animation.

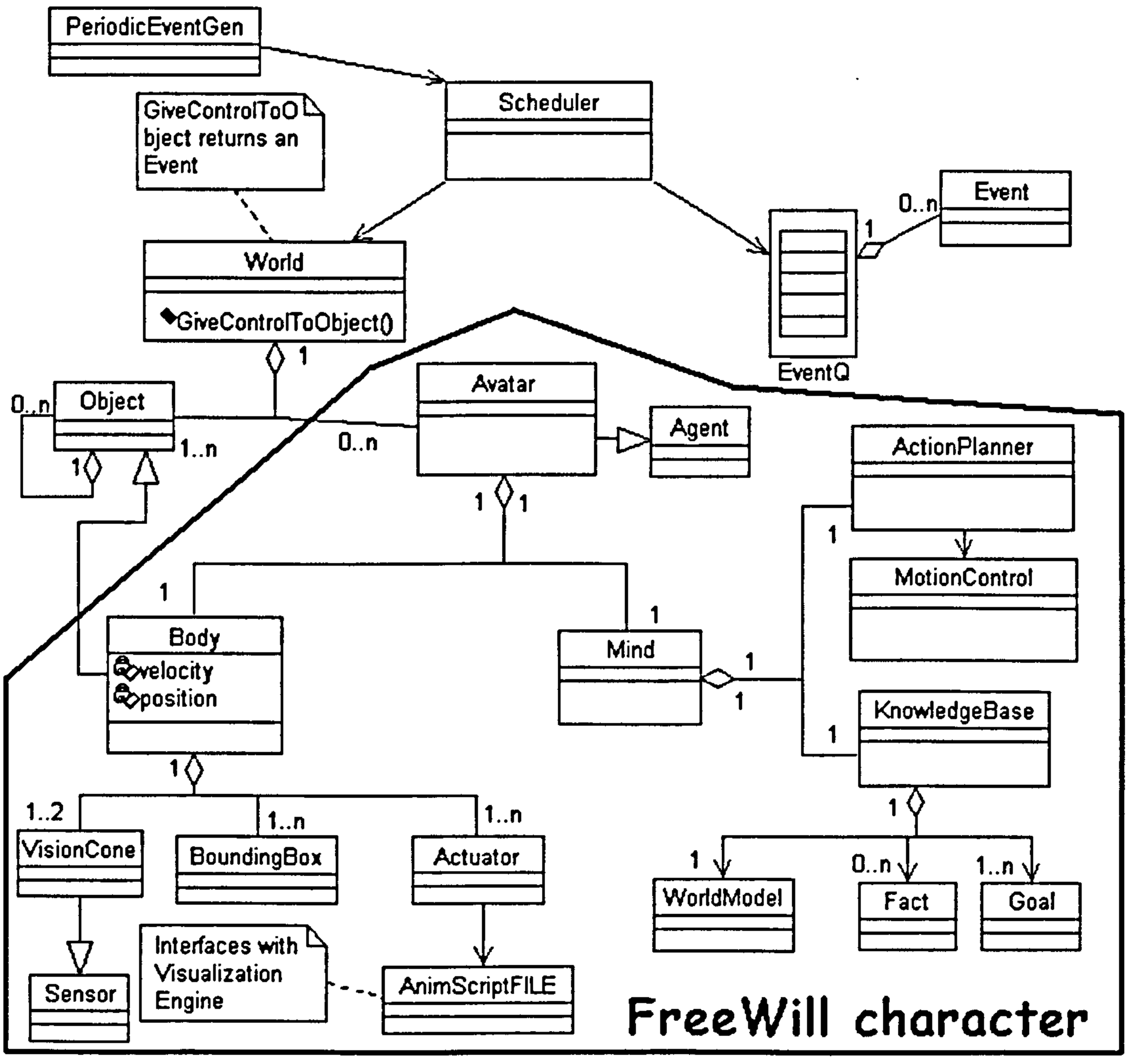


Figure 24 UML model of the system

4.5 Controlling Avatar Behaviours

One of the key elements of the knowledge base is the internal world model. Each time an avatar performs an action, the process is initiated by first updating the avatar's world model, which means a new sensing event must be inserted into the queue. Thus the avatar has a possibility to assess the results of its actions. The avatar senses the world via a vision cone, through which it gains awareness of immediate objects in its path (see Figure 25). The information obtained from the vision cone is then used to modify the avatar's plan and perform the next action. This happens when a sensing event is selected from the queue – the avatar then updates its world model and reconsiders the chosen course of action. Finally a new acting event is generated and next time the avatar is given control the last action from the queue is executed.

Figure 25 Scene as seen by an avatar

In general the avatars are able to perform two types of actions – simple, built-in actions such as walking and more complex actions including collision avoidance, handshakes or interaction with objects. The second group of actions is constructed in three ways. Firstly a prescripted action can be added to the avatar's plan library, an example of such action present in the implemented system is the handshake. The avatar can also synthesise a more complex action from the simple ones during the run-time – collision avoidance is one example. Finally new actions can be learnt offline, this technique will be described in the next chapter. With regard to interaction there also exist three categories. The system discriminates actions performed by a single avatar (waving,

looking around), avatar-object interactions (door opening) or avatar-avatar interactions (handshake, collision avoidance). Actions performed by exactly two avatars and involving explicit interaction (handshake) are the most complex ones. Figure 26 illustrates all different types of actions present in the FreeWill system.

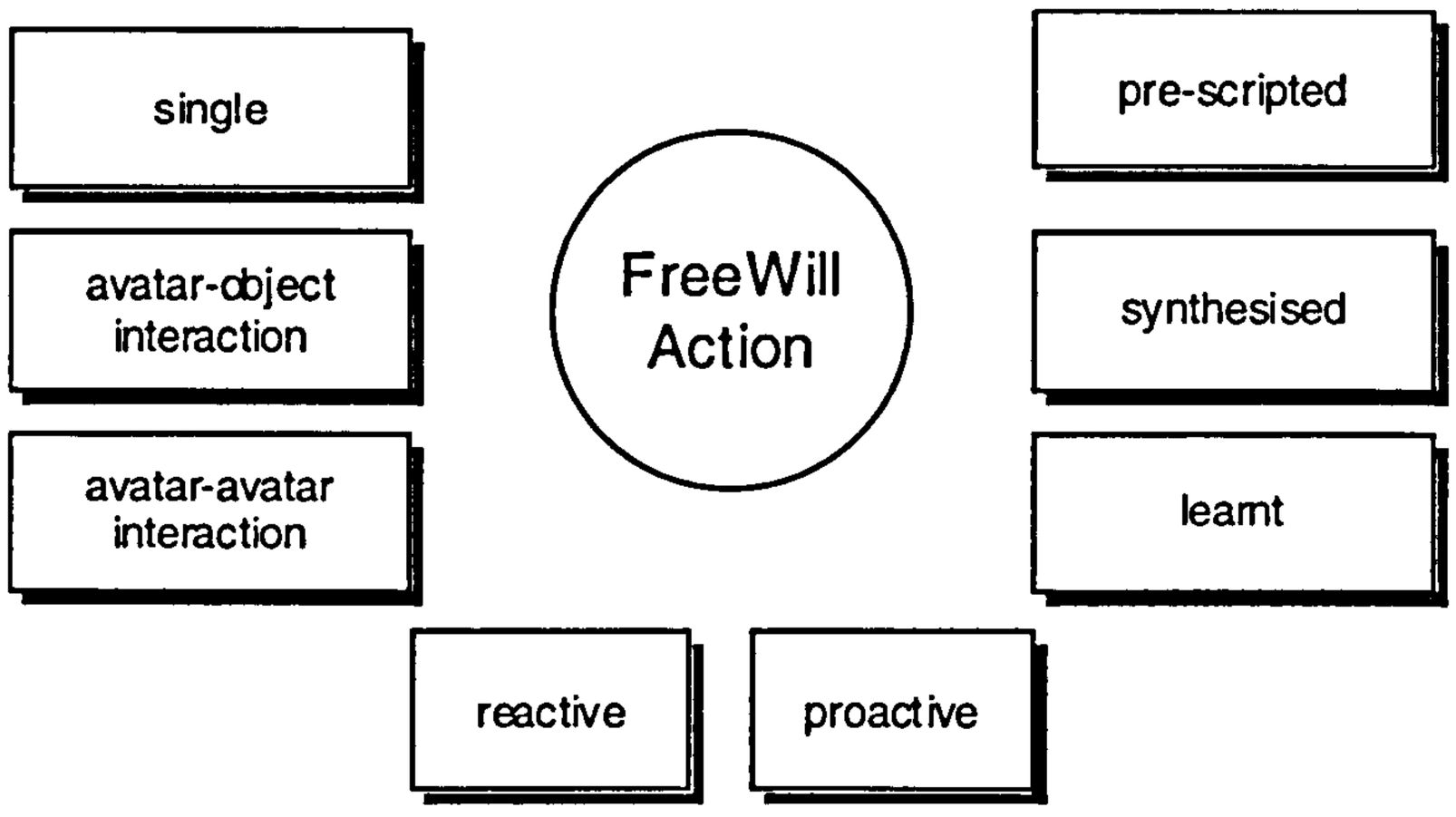


Figure 26 Different types of actions supported by FreeWill

The main simulation loop is located within the Scheduler class, which consecutively picks events from the event queue, the events are placed in the queue in the order they are submitted by the avatars (FIFO). Control is then passed to the appropriate world object to which the event refers (which will usually be an avatar) and necessary actions are taken. These can be:

- an 'act' action – such as move a hand or make step. The action is rolled out (the avatar's state variables are updated) and a new line is added to the script file. This action returns a new sensing event related to this avatar to be inserted in the event queue
- a 'sense' action – which means that the avatar should compare the perceived fragment of the world with its own internal model. Then the avatar has a chance to rethink its plan and possibly update goals and the planned set of future actions. This action returns a new acting event.

The returned actions are inserted in the event queue and the time is advanced so that the next event can be selected. A PeriodicEventGenerator class has been introduced to generate cyclic sensing events for each avatar so that even a temporarily passive avatar has its internal world model updated.

Avatar behaviours are goal directed. The primary goal is provided by the user and represents the aim of the simulation for that avatar. An example might be 'get to the end of the sidewalk'. However the fulfilment of this goal may be enacted with accomplishment of secondary goals which are set and assessed by the avatar. Examples are 'avoid collisions' and 'shake hands with friends'. Such goals are a part of the avatar's knowledge. When to give such goals priority can be inferred from the current world state. The knowledge base provides information about static world objects and other avatars (e.g. a list of friends). The avatar can be given two types of actions by the scheduler – if the action is an 'acting' action then the motion (a step, arm rotation etc.) is immediately executed. For sensing actions however the avatar needs to sense its environment and process this information. It may happen that as a result of the new state of the world the avatar has to modify its plan. In this case a new plan is generated. This algorithm is depicted below, an expanded version of it, including participating objects is attached in Appendix B.

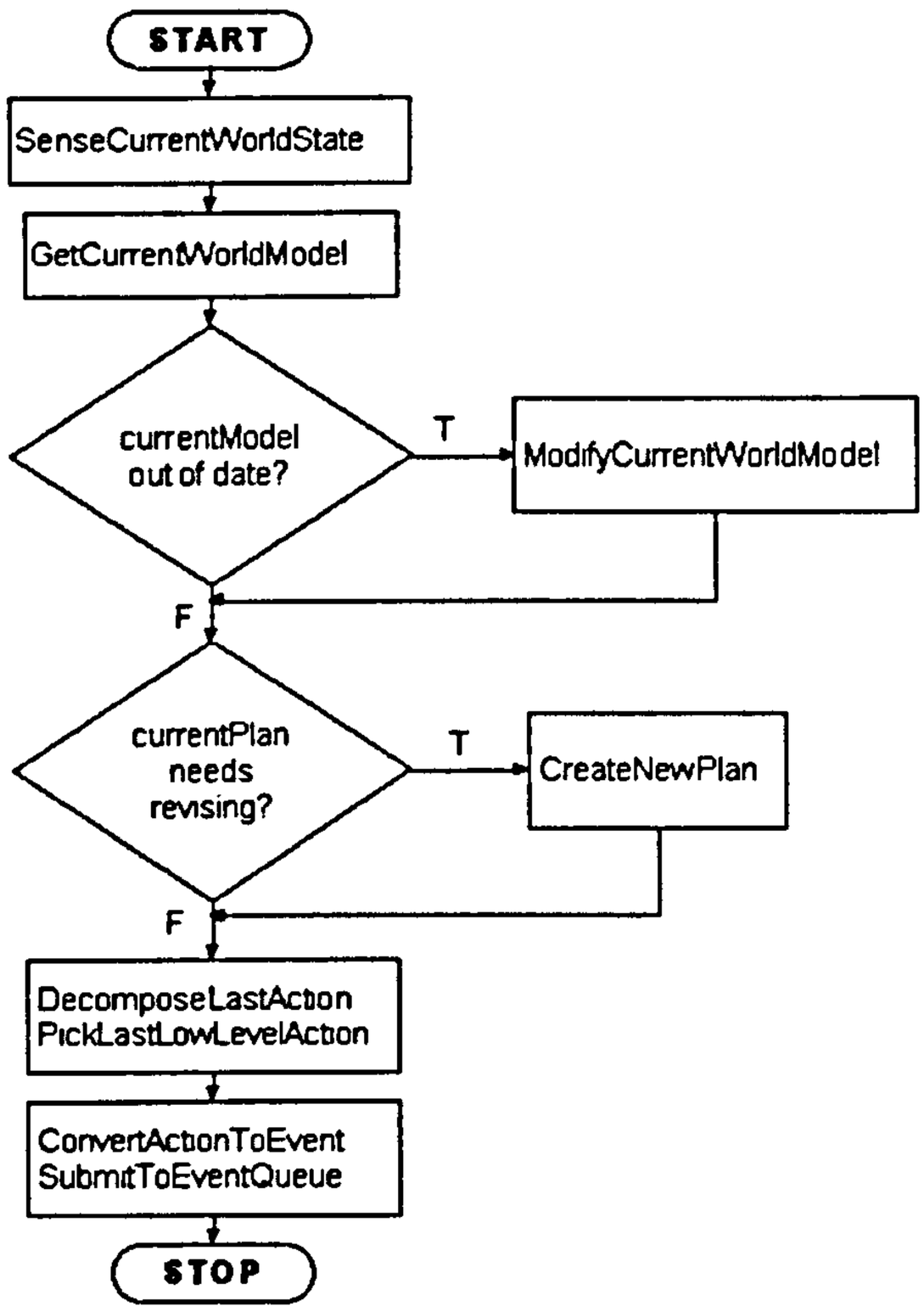


Figure 27 Algorithm controlling an avatar's behaviour

The goal-planning algorithm constructs plans using the notion of an action as a generic planning unit. An action can be defined at various levels of specialisation – from very

general ones (e.g. 'get to the end of the sidewalk') to fairly detailed activities ('do the handshake'). The most detailed actions (microactions) are said to be at level 0 (Figure 28). Examples of such actions would include atomic joint rotations and actions which can not be further decomposed, for example predefined motions supplied by the animation package (mocapped actions, predefined animated sequences etc.). Level 0 actions correspond to action events in the event queue and also to script file entries. In general every action is implemented by an avatar's member function, which will perform the action and update the state of objects affected by it. These objects can be world objects or parts of the avatar's body. The planning unit (ActionPlanner) operates on actions from level N to 1 – creating general plans and then refining them. The ActionPlanner maintains the chosen plan from which the last action is submitted to the MotionControl unit. It is then decomposed into a set of level 0 microactions (e.g. handshake consists of a set of arm and hand movements) which can be executed one by one. Any change in the plan may cause the list of microactions to be dropped and new ones to be generated.

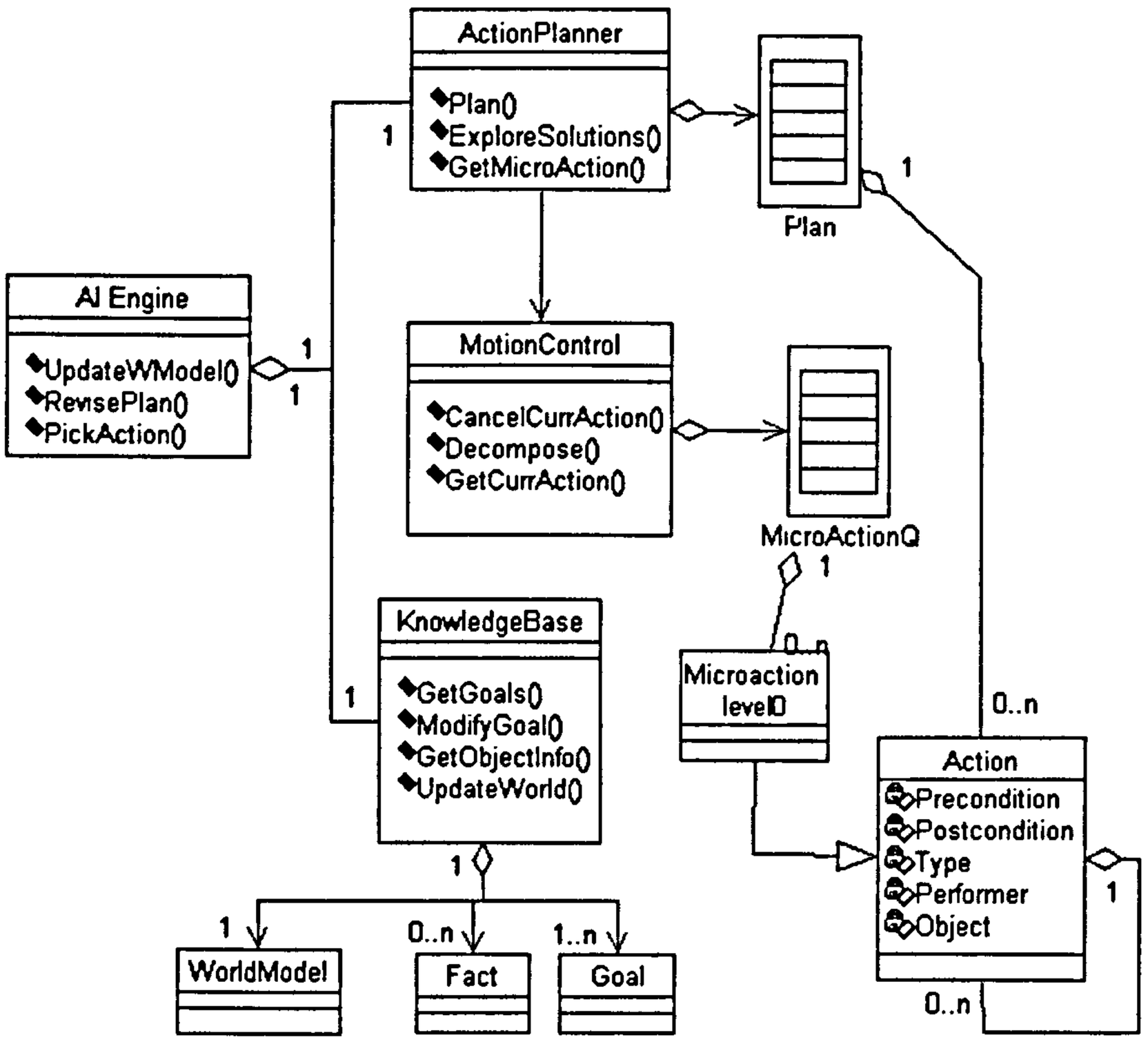


Figure 28 Action hierarchy and other AI components

The general planning collaboration is described in Figure 29. The collaboration is initiated by the Avatar attempting to update its world model (because some changes were detected in the perceived world state). The request is passed onto the Mind object which in turn updates the WorldModel stored in the character's KnowledgeBase. At this stage the Avatar submits another request to revise the current plan and the request is passed to the ActionPlanner object. If the ActionPlanner decides to update the current plan, it communicates with the KnowledgeBase to retrieve the goals and the current world model. Once a new sequence of high-level actions has been created the first action is then passed to the MotionControl object and decomposed into atomic actions. When the Avatar requests the next action the first action from the MotionControl queue is returned, it is next wrapped into the Event structure and submitted to the EventQueue (this is not shown in the diagram).

The scheduler then selects the next action to be executed. If the action event is a sensing event the collaboration starts again for a different avatar, if an action event is pulled from the queue then the action is executed by the appropriate avatar (or object), meaning all the necessary properties of the participating world objects are updated. At the same time the actuators pass the information of that movement to the interface with the animation package so as to update the state of the world that will be displayed in the animation. For UML diagrams illustrating the full sensing and acting collaborations see Appendix D.

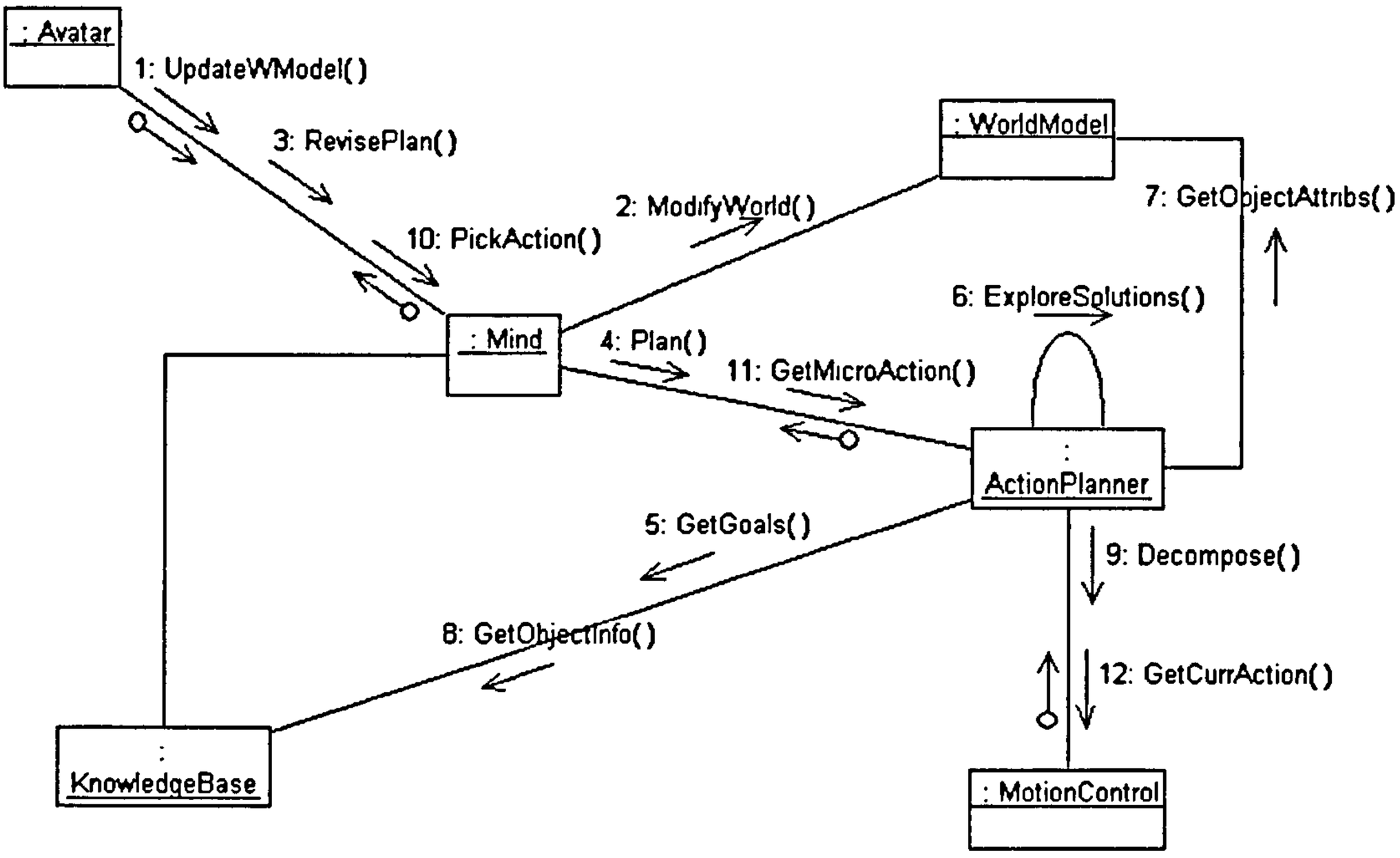


Figure 29 Objects participating in the planning collaboration

4.6 The Interaction Algorithm

For an autonomous animated character to be believable it must be able to perform actions involving at least two avatars. Such actions, to look natural, must be synchronised in time and aligned in space. The spatial aspect of the interaction is handled by defining the required distance between the avatars for each of the collaborative actions. The orientation of interacting characters must also be set and here they are assumed to be always facing each other. Another important aspect of such actions is the timing, since for such actions as a handshake even a small difference in timing will cause the sequence to look unrealistic. FreeWill does not use dummy objects to co-ordinate the collaborative actions (cf. Monzani, 2002), neither an explicit communication language as this is usually not the way humans synchronise their actions. The solution proposed is based on finite state machines (see Cremer *et al*, 1995), which take the interaction through a number of stages and allow for spatial alignment (initial stages) and temporal synchronisation (late stages). Such an approach allows for easy substitution of actions in the plan library without the need to modify the synchronisation scheme, thus allowing for quick creation of different animations. The details of the synchronisation algorithm for actions involving two avatars, are presented in Figure 30, and the different stages of the interaction process are illustrated in Figure 31. The algorithm distinguishes three separate states for the action duration – the close-

up, preparation and execution stages. During the first phase the avatars notice the other participant and start walking towards it (the time of noticing the other character does not have to be the same). For some actions this will only happen if the action has not previously been executed with the same participant (e.g. handshakes). However if the action can be executed the current goal will be replaced with a newly created intermediate goal and the avatar will pursue the new goal by approaching the other character. When the distance between the two characters decreases to some predefined value, one of the participants (because of the chosen event scheduling scheme this will initially happen to one of the characters) will stop walking and enter the preparation phase. The preparation phase involves initiating the action (e.g. raising a hand). Upon finishing the preparation phase the avatars execute the main part of the action (shake hands) and resume the primary goal. If necessary the knowledge base is also updated with a new fact stating that the two avatars have executed the action. A pseudocode version of this algorithm can be found in Appendix B.

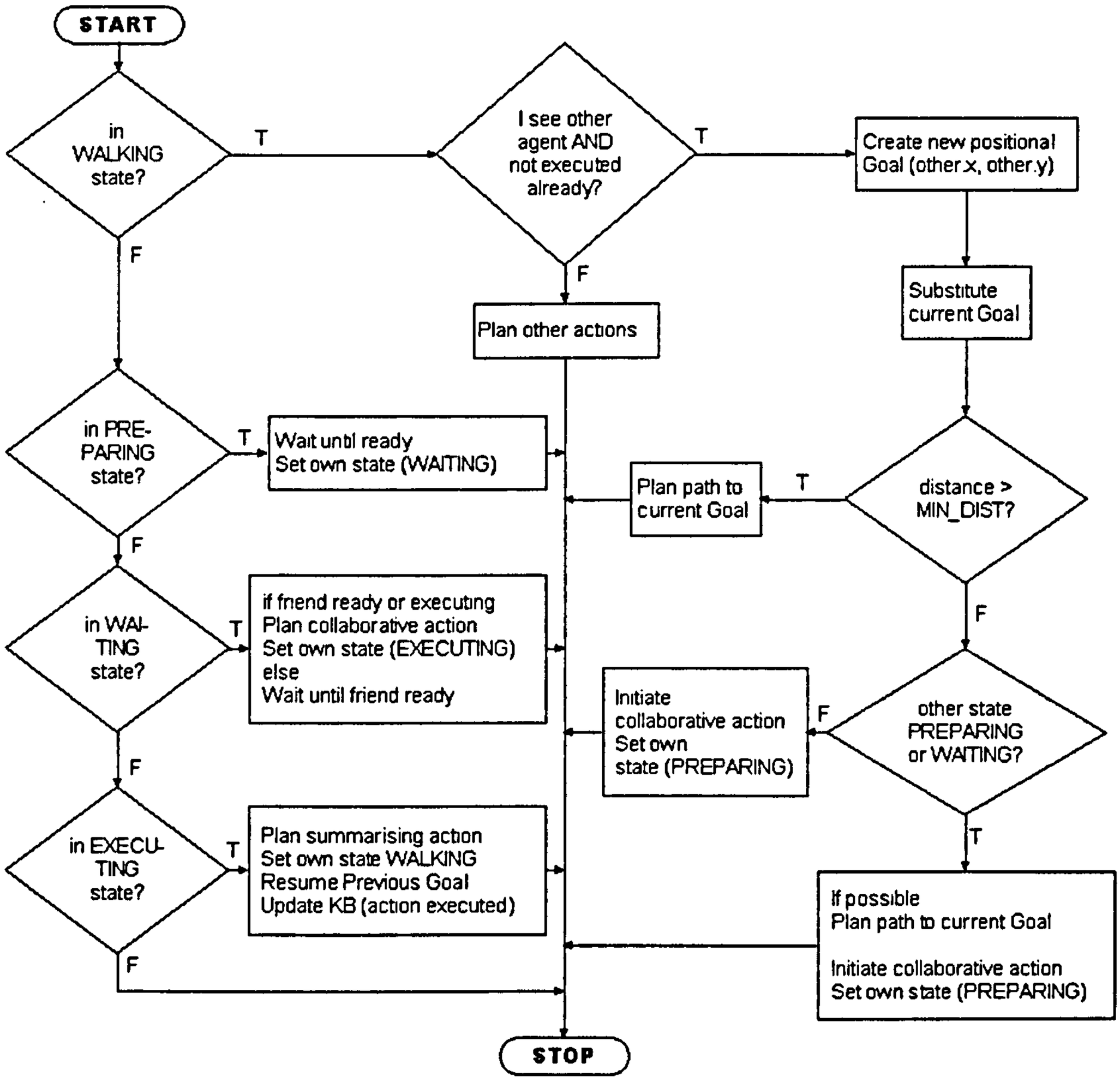


Figure 30 The synchronisation algorithm (handshake)

Although for action synchronisation it might be useful to implement a dedicated communication language, the only use of such a language would be in actions involving direct collaborative interaction between avatars (e.g. a group task including gathering a number of avatars in order to lift a heavy object). Therefore high-level communication is not currently included in the FreeWill interaction scheme.

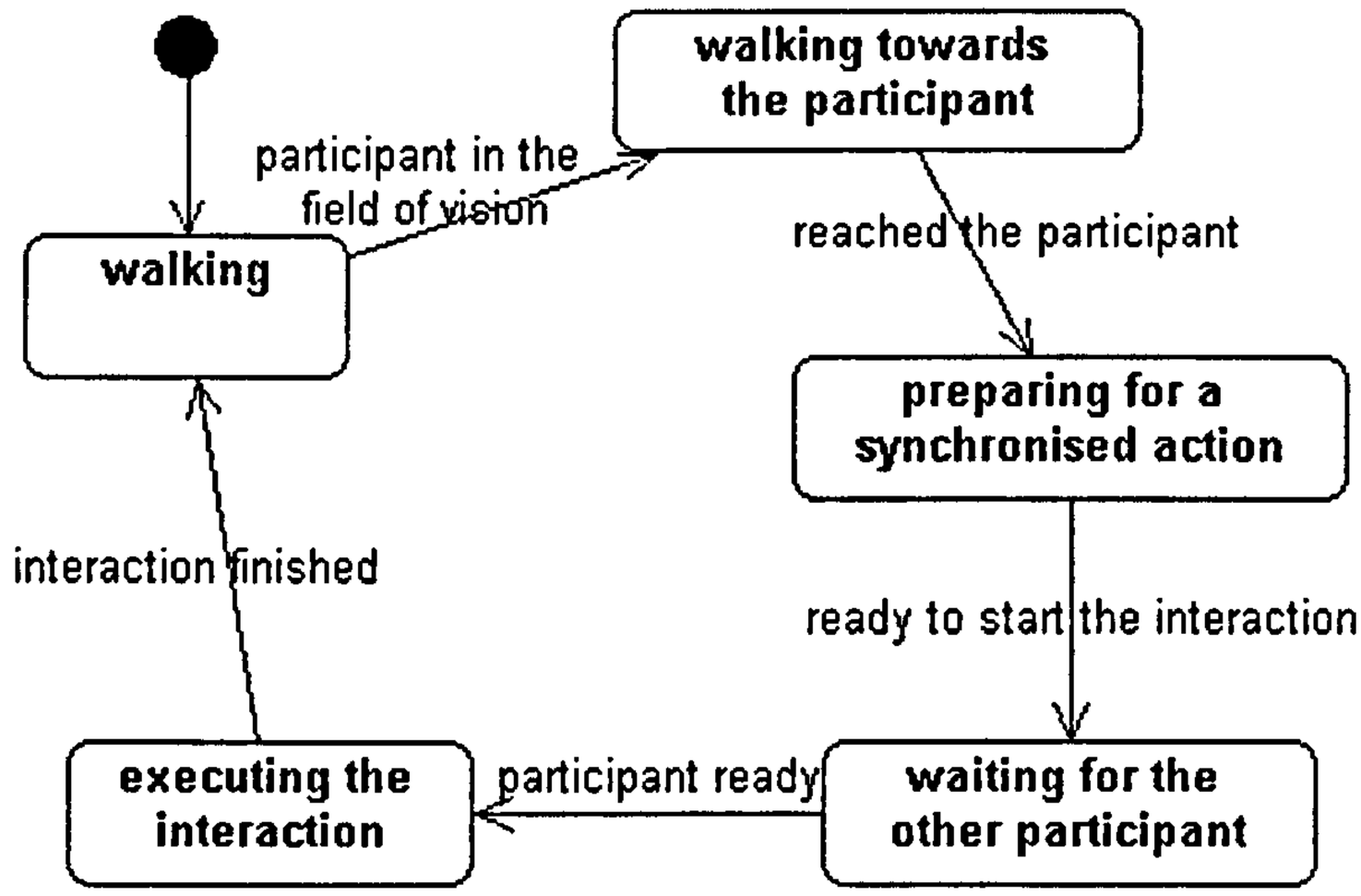


Figure 31 Synchronisation of inter-avatar interaction

4.7 Details of the AI Module

A detailed view of the AI subsystem is presented in Figure 32. It consists of 3 general parts – the main class - Mind, the planning cluster and a knowledge base. Mind, apart from serving the purpose of a module's container, is also the subsystem's interface, through which other units request services and update information about the environment. The planning classes do both the actual planning and maintain a list of planned actions, which can be executed in turn. The knowledge base stores all other information. The figure also presents some of the most important methods and properties of the implemented prototype and detailed relationships between different objects.

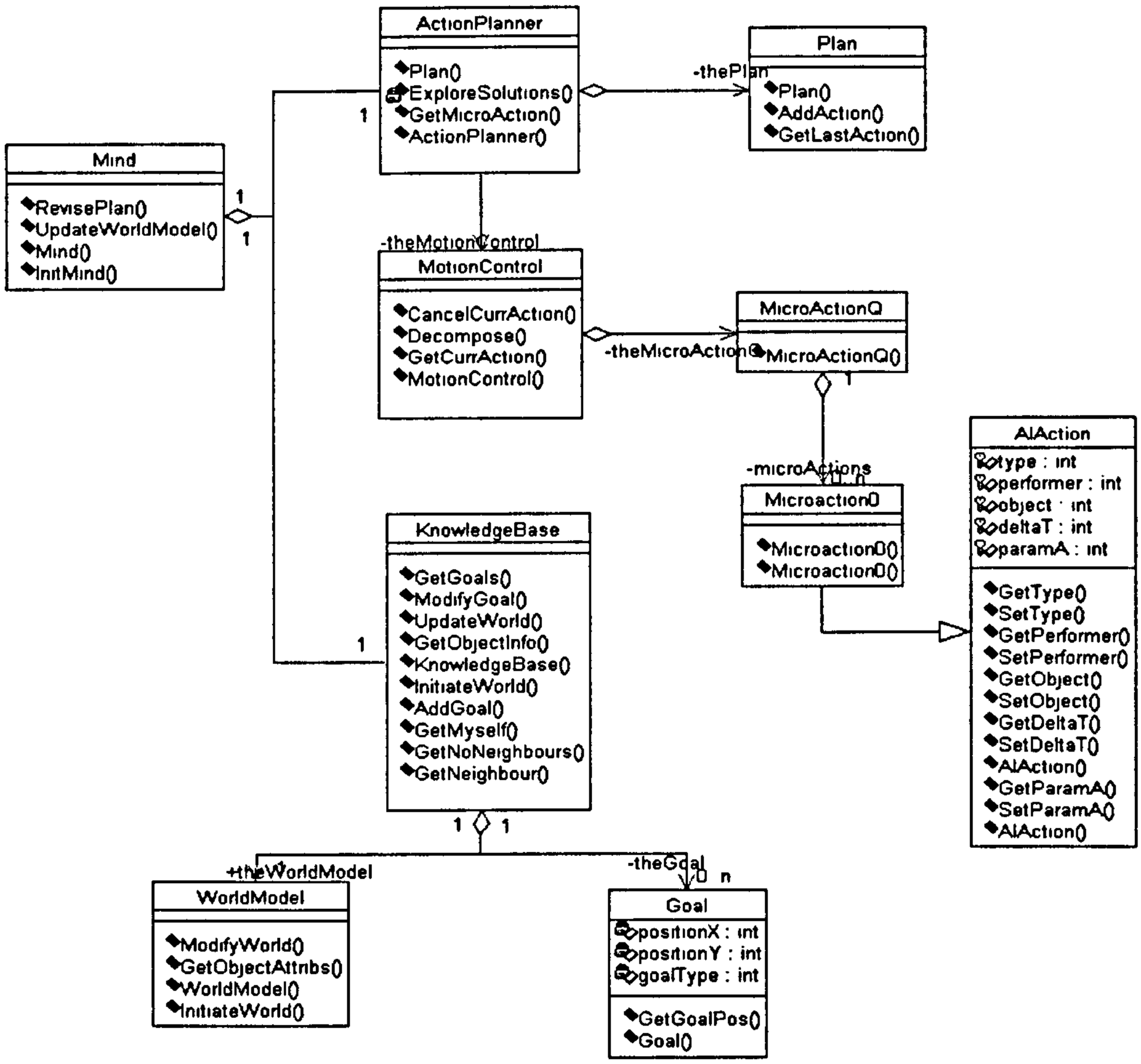


Figure 32 The AI subsystem

4.8 Communication with 3DS Max

In the current implementation, the presented prototype communicates with the animation package 3DS Max using an off-line file-based communication mode (stepfiles+scripts). The generation of the animation is executed as an external process and an animation file is generated. The file is then imported into the animation engine for rendering. A similar mode of communication could be used to retrieve the scene information to be used to construct the virtual world. In the learning unit presented in Chapter 5 another type of communication is presented which allows for real-time control of the avatars and direct export of scene objects into the FreeWill system using 3DS plugin capabilities. Avatar collision detection in the current implementation is based on detecting objects intersecting with the directional vector of the agent.

4.9 Summary of the Prototype

This chapter presented the FreeWill prototype aimed at supporting creation of 3D animated autonomous characters. The system proposals have been presented as a theoretical cognitive architecture derived from two existing systems, comprising goals, actions, plans, sensing and addressing the need for a geometrical representation. The architecture was then modelled using a software engineering notation and a number of issues such a planning collaborations and action synchronisation have been considered. The main missing element at this stage is automatic addition of actions into the system. This issue will be addressed in the next chapter. Results obtained by implementing the proposed system will be presented in Chapter 6.

Chapter 5 – Automatic Action Acquisition

The architecture presented in the previous chapter relies heavily on the number of different high-level actions, which the avatars can perform. So far the only way of adding new actions to the avatars' behaviour repertoire has been by manually scripting them. Ideally, however, an animation generation framework, apart from importing predefined actions in a form of a script or motion capture file should also allow for automatic generation of new actions, at least for the purpose of animation prototyping. This chapter presents a technique for automatic acquisition of new high-level actions using machine learning.

5.1 The Deterministic Algorithm

The initial proposal of the learning mechanism is based on the deterministic version of the Q-learning algorithm, with the update equation presented before:

$$Q(s_t, a_t) \longleftarrow r_{t+1} + \gamma \max_a Q(s_{t+1}, a) \quad (3)$$

The simulation is conducted within a fully controllable, virtual world, hence both the rewards and the results of agent's actions are fully deterministic. This assumption allows building the learning module around this simpler version of the algorithm. As described in Chapter 2 the Q-learning algorithm requires that the state-action space and the goals of the agent be defined explicitly. Therefore the development of the learning module has been conducted in a few general steps:

a) The goals of the learning process have been identified as any complex (high-level) actions consisting of a number of smaller hand and body movements.

b) The agent has been assigned a number of simple actions, which include arm, forearm and hand motion along a number of orthogonal axes, and also walking and grabbing (this includes only the actual motion of the fingers). The selection of actions depends on the mode of control. Two modes of control have been proposed to acquire the motion – in the first one the agent is controlled using forward kinematics, in the second one inverse kinematics may be used. All simple actions used by the agent are grouped in Table 5 and graphically depicted in Colourplate 1.

c) The state space has been defined. Each mode of control (forward and inverse kinematics) corresponds to a different state space, but in both cases the state space is

a discretisation of a continuous space defined as a number of degrees of freedom for the joints necessary to be used to participate in the task. For the forward kinematics mode of control the degrees of freedom of the arm are defined as rotations around spatial axes, with discretisation defined over the angles of rotation. Additionally each of the angles is limited to a specific range (larger however that the biologically feasible motion to enable detection of violation of the biomechanical constraints). In the inverse kinematics case the state-space discretisation is applied over the spatial location of the end effector (hand) of the agent. In both cases walking along one dimension gives an additional action, descretised to include several steps around the agent's starting position. The only represented states are those of the agent. The states of the external objects (door, teapot) can be represented as additional variables, thus reducing the state-space of the learning task and allowing for a more precise control and rendering of those objects. Additionally, because the actual definition of the state space depends on the type of problem, the state-space may be defined on a per-task basis. In some cases, for example, the definition of the learning task precludes the use of particular actions such as walking. This allows for the size of that space for a particular task to be defined as a subset of the full state space available to the agent.

d) The action space has been defined. For each possible state space dimension there are always two possible actions, which allow moving the body part (arm, forearm, hand etc.) along the specified dimension in the opposite directions. Examples here might be simple walking forward and backwards or hand movement along the vertical axis resulting in lowering or raising a hand.

e) The exploration policy has been defined. This is presented in the section below.

Table 5 Low-level actions used to train the avatar:

Forward kinematics control	**Inverse kinematics control**
1. Rotate arm up/down by $\Delta\alpha$	1. Move palm by Δx
2. Rotate arm forward/backward by $\Delta\alpha$	2. Move palm by Δy
3. Rotate forearm by $\Delta\alpha$	3. Move palm by Δz
4. Rotate hand along Z axis by $\Delta\alpha$	
5. Rotate shoulder along Z by $\Delta\alpha$	
6. Perform the grabbing action	4. Perform the grabbing action
7. Move forward/backward by Δx	5. Move forward/backward by Δx

5.1.1 The exploration policy

According to the Q-learning algorithm the Q-values are initially set to random values and the agent starts exploring the environment. Upon reaching a goal the agent is then randomly placed in a new state and begins a new iteration. It is apparent that in those initial stages the exploration policy followed by the agent should not rely on the current Q-values. This is because the values are highly unreliable and many of the states have not been explored at all yet. Only after a number of iterations, when the agent increases its belief about the quality of the Q-values can the action selection policy be changed to one relying more on the current Q-values (this is called the exploitation phase). The exploration/exploitation strategy applied by the action learning module uses the Boltzmann distribution (see Michell 1997). For equal Q-values the action is chosen at random, otherwise actions with higher Q-values are preferred. The equation is given below:

$$P(a_i \mid s) = \frac{k^{\hat{Q}(s,a_i)}}{\sum_j k^{\hat{Q}(s,a_j)}} \quad (4)$$

where $P(a_i \mid s)$ is the probability of selecting action a_i given that the agent is in state s, $k > 0$ is a constant which determines how strongly the selection favours actions with high $\hat{Q}$-values, $\hat{Q}(s,a)$ is the current approximation of the Q-function. For the action acquisition tasks presented in this thesis, k = 2. Such representation of the exploration/exploitation policy in early stages assigns similar probabilities to state transitions not visited yet and changes bias as the agent learns more about the results of its actions.

An additional problem when learning a motion sequence for an animated agent is that not all of the state space is interesting from the animation generation point of view. Although the Q-learning algorithm does not give preference to any part of the state-space, in animation tasks there is always a number a privileged states which are more important to the task definition. This includes mainly the starting state of the animation, since the task is run to generate a sequence from a rigidly defined starting pose. Therefore the exploration policy utilised by the learning module was additionally reinforced to proceed more frequently around the initial motion states. This has been implemented by finishing

each learning task with a number of iterations in which the agent is always placed in the same starting state. The number of these special iterations has been set to span about 5-10% of the total number of epochs, thus optimising the path through the state space for a particular problem. Additionally 0.5% of all actions selected by the agent are random, regardless of equation (4) (Sutton and Barto, 1998) and an upper limit was imposed on the length of the exploration phase – if the agent did not encounter an absorbing state after several hundred transitions the iteration was restarted.

5.1.2 Positive and negative rewards

Each learning task must include at least one state in which the agent receives a positive reward. This is the goal state and upon visiting it the agent finishes the current epoch and starts a new one. An example of reward state may be one in which the agent finds itself in a particular position, or when a state of another object changes (e.g. a teapot is lifted above a certain height). Such positive rewards are used by the learning algorithm and contribute to finding the optimal path through the state space.

Additionally, it is also desirable that the agent avoids certain states while learning the optimal solution. An example might include a state in which the agent collides with an object from its surrounding environment or when the position of one of the agent's limbs exceeds the limits imposed by biological constraints (e.g. overstretching a joint). In such cases it is necessary to inform the agent that the constraints have been broken. While it is possible to limit the state space so that it does not include the wrong positional states (by limiting the angles by which an arm can be rotated thus directly imposing the biological constraints), such solution can not be applied to the problem of collision avoidance. Neither can it be adopted when the inverse kinematics is used, because the IK algorithm imposes its own constraints on rotational angles resulting from the underlying calculations. A technique unifying the problem of state space limitation has therefore been applied which uses negative rewards to label out undesired motions. Thus the algorithm remains fairly general, and the tagging of the state-space happens during the learning process without a need for additional knowledge about the environment prior to the simulation. Additionally the negative rewards are not propagated to the surrounding states but are only used to locate the undesired state transitions.

The collision detection approach presented in Chapter 4 is not sufficient to detect collisions between the agent and the small objects with which the agent must interact during the learning. Therefore a quick and efficient way of detecting collisions was necessary. The desired collision detection algorithm should avoid testing all the polygonal faces in both objects for overlap as this is a very expensive operation. More efficient solutions are based on spatial volumes (spheres) and bounding boxes but for these algorithms tightness of fit between an object and its bounding volume is crucial for the precision of the collision test. In a situation requiring the collision detection to be performed with a high level of precision this approach is insufficient or it requires strong spatial subdivision techniques applied in a hierarchical manner. Therefore the collision detection used by the learning module is based on the OBB (oriented bounding boxes) approach (Gottschalk, 1996). The OBB method allows for tight approximation of the surrounded object and it is not very computationally intensive. For details of the method and its implementation to character animation see system designed by Francik (2003).

5.1.3 Integration with the FreeWill framework

The presented learning module can at this stage be easily integrated with the main FreeWill framework. The most important part of the integration includes transition of the generated motion sequences, as the learning process is performed off-line, in isolation from the crowd simulation engine. However the scripts generated in the course of learning can be directly ported into the FreeWill's plan library. The next step is the modification of the planning module to take into account the newly added actions. This may for example include definition of pre- and postconditions for each new action. It may also be possible to utilise the newly learnt sequences with avatars with different body textures (if the bone structure is the same), to add variety to the motion (by adding a number of similar actions), or to replace bad hand-animated sequences. Parameterisation of the learnt sequences could also be attempted to allow reuse of new motion for characters with different body sizes. Figure 33 illustrates integration of the learning module with the FreeWill system.

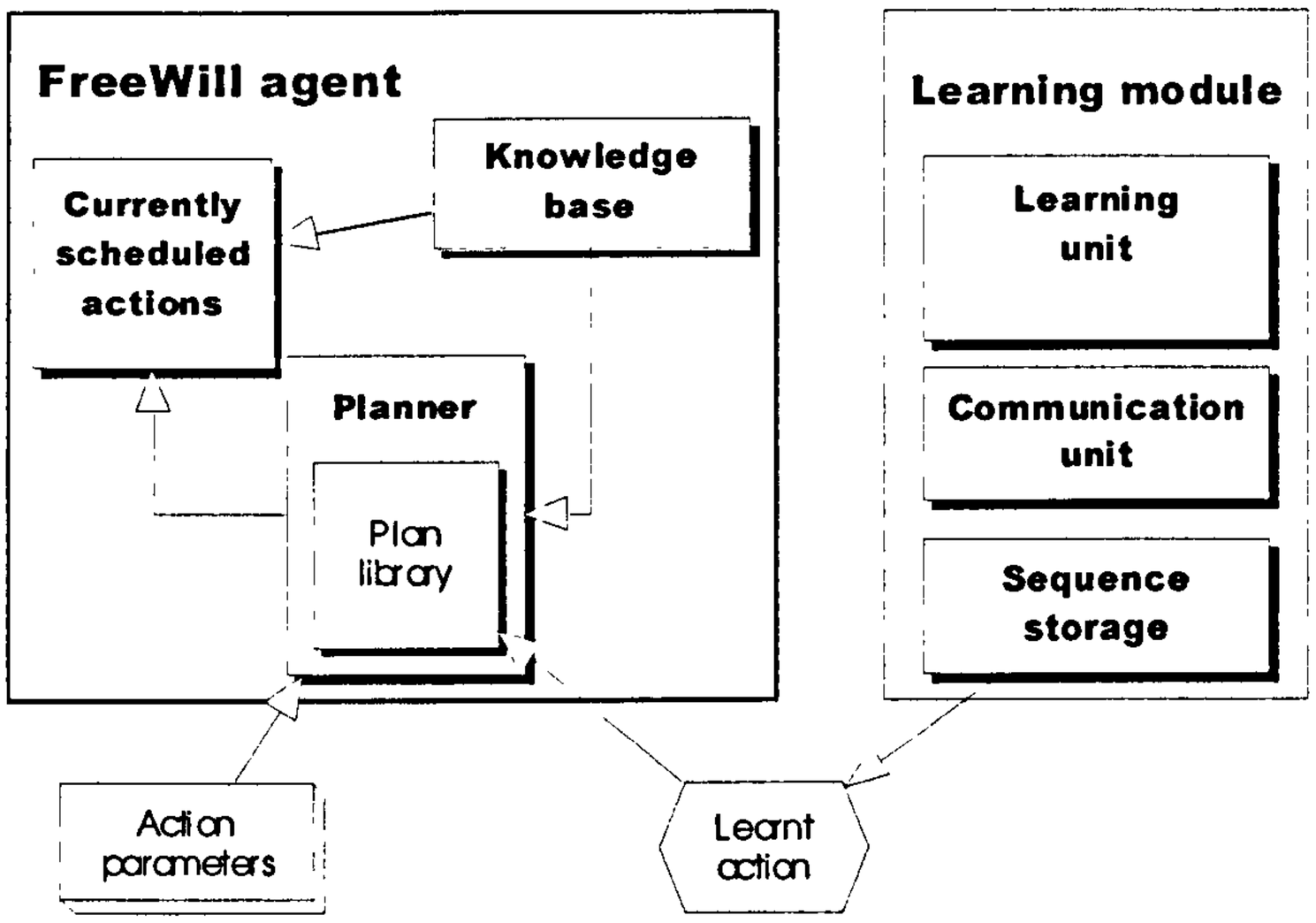


Figure 33 Integration of the learning module with FreeWill

5.1.4 Communication with the animation package

The learning extension presented here requires real time communication capabilities with the animation package. Therefore an extension of the communication mechanism described in the previous chapter was necessary. The learning module uses the Microsoft COM technology to invoke functions written in Max Script, which in turn control the behaviour of objects inside the animation package. In the same way the information about the objects is returned to the learning unit. Additionally the collision detection algorithm has been implemented as a plug-in integrated with the 3DS Max animation framework.

The simulation proceeds in a number of iterations. When it is terminated the shortest sequences of actions fulfilling the goal of the learning task are saved into a file in a form of script. A typical learning cycle will generate up to several thousands different solutions, the final number depending on the length of the simulation and complexity of the task. The resulting scripts can then be verified in the animation package, imported to FreeWill and used to generate the final animation. The framework is depicted in Figure 34.

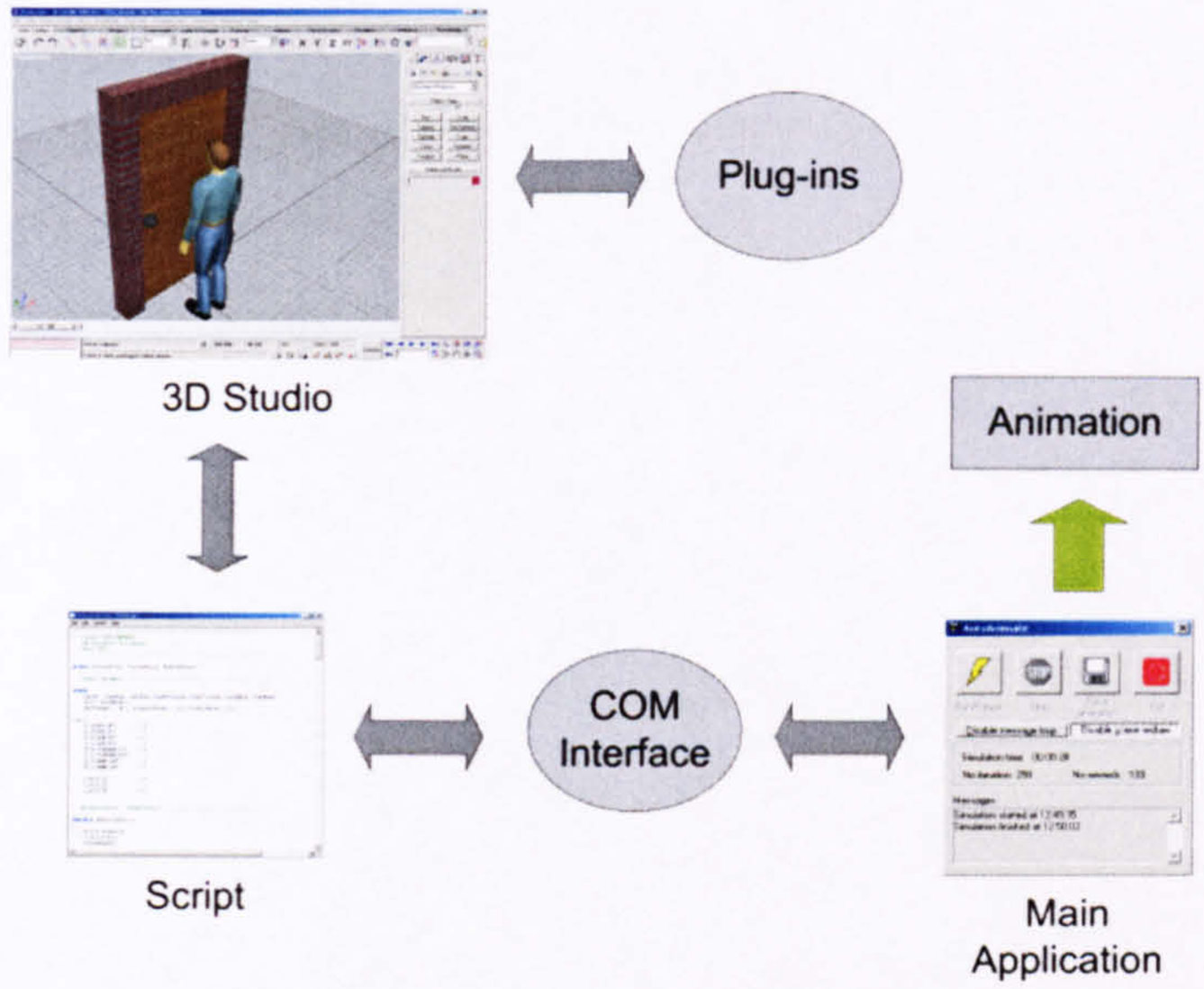


Figure 34 Communication with the animation package

5.2 The Non-deterministic Algorithm

Depending on the task, it is also possible to implement the action acquisition mechanism using the non-deterministic version of the Q-learning algorithm. This version of the algorithm generates more robust solutions with respect to uncertain action selection mechanisms and non-deterministic rewards.

The non-deterministic update equation is:

$$\hat{Q}\ (s_t, a_t) = (1-\alpha_n)\hat{Q}\ (s_t, a_t) + \alpha_n[r_{t+1} + \gamma \max_a \hat{Q}(s_{t+1}, a)] \quad (5)$$

where α_n is defined as follows: $\alpha_n = \frac{1}{1+visits_n(s,a)}$, $\gamma \in [0,1)$

The variables *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 (Mitchell, 1997).

The definition of the state space and the action space remain the same. So do the exploration policy and the conditions under which the agent receives rewards.

The non-deterministic version of the learning has been applied to the inverse kinematics mode of control in a non-deterministic environement. Results obtained from both approaches have been summarised and compared in the next chapter.

5.3 Evaluation Metrics

The learning task defined in this chapter is in fact an optimisation problem formulated over the number of actions necessary to fulfil the goal. This means that apart from trying to find a set of transition states from the starting position to the goal state, the agent will also minimise the number of actions necessary to achieve its goal. Additionally, as mentioned before, during the process the agent may in fact find more than one sequence, all of which lead to the desired solution in a minimum number of steps. If the number of such solutions is small, they can all be reviewed manually according to some predefined criteria (the chosen criteria can be perceived realism of the sequence, a particular sequence of actions, way of approaching external objects including limb orientation in space, movement direction etc.). However, in the case of more than several different solutions this approach is highly unfeasible. This section presents three metrics which try to select a small number of action sequences from the full set of generated solutions. One or more of these metrics would normally be used by the user to limit the number of final animation sequences which will be added to the plan library or directly inserted into the animated scene.

5.3.1 Local distance metric

The local distance metric (LDM) selects a pair of animation sequences and generates a distance value depicting the similarity of these sequences – the higher the distance value, the more dissimilar the two sequences are. The distance, which is commutative, is calculated on the basis of individual actions in the sequence, located on corresponding positions within the string. Each two such actions are compared and if they appear to be different, the total distance for the sequence is increased by one, otherwise it remains the same. The resulting set of distance values, for all permutations of the action sequence pairs in the input solution set, is then searched for the maximum and the resulting subset can be generated as pairs of sequences for which the distance is equal to the maximum. It is also possible to add other pairs with the distance value lower than the maximum within a given margin, thus allowing the user to influence the size of the presented subset. The formal definition of the metric is given below.

Let us consider two learnt sequences of the same length M, A and B, consisting of a number of low-level composite actions each:

$$A = a_1 a_2 a_3 a_4 \cdots a_M$$
$$B = b_1 b_2 b_3 b_4 \cdots b_M \quad (6)$$

The length of each sequence, that is their cardinality, is thus given as $M = card(A) = card(B)$. A distance between two sequences A and B can now be defined – similar to the Hamming distance (Cover and Thomas, 1991) – as:

$$d_{AB} = d_{BA} = \sum_{i=1}^{M} | \, sign(a_i - b_i) \, | \quad (7)$$

The local distance metric finds all pairs of sequences for which the distance is equal to the maximum distance found between any two sequences in a given set:

$$d_{\max} = \max\{d_{AB}\}, \underset{A \in S}{\forall}, \underset{B \in S}{\forall}, A \neq B \quad (8)$$

where S is a set of all sequences found for a given task.

When applying the metric it is possible to release the constraint put on the maximum distance between the two sequences by setting it to a value lower than $d_{\max}$, thus increasing the number of different pairs in the resulting set.

Example

A = 0 0 0 0 4 4 4 4 4 6 1 1 8 1
B = 0 6 0 0 0 4 4 4 4 1 4 1 8 4
C = 0 6 1 0 0 4 4 4 4 1 4 1 8 7
$d_{AB} = 5$ $d_{AC} = 6$ $d_{BC} = 2$, $d_{max} = 6$

5.3.2 Global distance metric

The local distance metric from the previous section finds sequences, which substantially differ from each other, but it does not guarantee that the resulting set will be well sampled. This is because one sequence very different from all other ones will affect the maximum distance and cause this particular sequence to be paired with a number of other solutions which may nevertheless be mutually similar. The global distance metric (GDC) relies on

finding an average sequence, from which the distance of individual sequences can be assessed following the model defined by the local distance metric. The average sequence is created by finding, for each consecutive position in the sequence, an action which appears most frequently in this position, considering all the action sequences from the input set. The average sequence is thus defined as:

$$Avg = \{\arg\max_i(h_1(O_i)),\cdots,\arg\max_i(h_M(O_i))\} \quad (9)$$

where the function $h_j(O_i)$ indicates the histogram of occurrences of actions on the j-th position in sequence O_i. The average sequence is built by using the *argmax* – over all possible sequences ($\forall O_i$). Each *argmax* operator returns the action a in the set S of available actions, yielding the maximum number of occurrences in the histogram. If there are several maxima, then one of such actions is chosen at random.

Example

A = 0 0 0 0 4 4 4 4 4 6 1 1 8 1
B = 0 6 0 0 0 4 4 4 4 1 4 1 8 4
C = 0 6 1 0 0 4 4 4 4 1 4 1 8 7

O = 0 6 0 0 0 4 4 4 4 1 4 1 8 1
$d_{AO} = 4$ $d_{BO} = 1$ $d_{CO} = 2,$ $d_{max} = 4$

5.3.3 Action similarity metric

This metric is similar to the local distance metric in that it considers pairs of action sequences. The numbers of occurrences of each composite action within both sequences are counted and compared between the two sequences, the resulting differences in occurrences of each composite action are added up. The result is a scalar, which is higher if the sequences consist of a larger number of different composite actions. Thus the similarity metric for two action sequences A and B is defined as:

$$k_{AB} = k_{BA} = \sum_{j=1}^{L} | g_{a_j}(A) - g_{a_j}(B) | \quad (10)$$

where the function $g_{a_j}(S)$ indicates the a_j bin of the histogram of occurrences over all actions in sequence S and L is the number of all different low-level actions used to train the agent.

Example

A = 0 0 0 0 4 4 4 4 4 6 1 1 8 1
B = 0 6 0 0 0 4 4 4 4 1 4 1 8 4

$$k_{AB} = |4 - 4| + |3 - 2| + |0 - 0| + |0 - 0| + |5 - 6| + |0 - 0| + |1 - 1| + |0 - 0| + |1 - 1| = 2$$

5.4 Summary

This chapter presented an adaptation of both Q-learning updates to the task of automatic acquisition of actions for an animated bipedal avatar. The details of the state-action spaces for this problem have been presented as well as the exploration strategy employed. The integration with the main FreeWill engine has also been explained, and the modified real-time communication with the animation package has been illustrated. Finally, based on the conclusion that the algorithm may generate multiple optimisation solutions for the given tasks, a number of metrics have been proposed for selecting a subset of solutions. The reason for doing this is to further automate the process of adding new actions by presenting the human user with a small number of varied action sequences.

Chapter 6 – Results and Evaluation

The proposals presented in Chapters 4 and 5 have been implemented as a prototype version of the FreeWill architecture (Amiguet-Vercher *et al*, 2001, Szarowicz *et al*, 2001a, 2001b, 2002, 2003, 2004, 2005, Forte and Szarowicz, 2002, Szarowicz and Forte, 2003, Szarowicz and Remagnino, 2004, Szarowicz, 2004). Results obtained from this implementation are presented in this chapter. The chapter is divided in two parts. The first part presents results obtained from the FreeWill crowd system prototype (Sections 6.1, 6.2) and includes a comparative study and evaluation against three other animation frameworks. The second part presents results generated by the learning module described in Chapter 5 and compares action acquisition for the two modes of control used – forward and inverse kinematics.

6.1 Crowd Scenes

The experiments run using the FreeWill framework involved a number of avatars participating in a crowd scene. The example settings are presented below.

In the first setting the characters were placed on a sidewalk. Each avatar was given a primary goal of getting to the opposite end of the sidewalk, while at the same time avoiding collisions. Additionally a couple of avatars were defined as 'friends', which resulted in a handshake being performed before completing the primary goal. Colourplate 2 presents the change of goal-directed behaviour exhibited by two avatars changing their current goals in order to shake hands with a friend. Before starting the simulation the avatars were placed at a considerable distance from each other, so that they could not see each other. They were also given primary goals ('get to the position (X,Y)', which in this case is the end of a sidewalk, Colourplate 2, shots 1-2), which they try to pursue immediately after the simulation is started. Hence they start walking in the opposite directions and as a result the distance between them decreases (Colourplate 2, shots 3-4), eventually they recognise each other as friends and decide to replace the main goal with an intermediate one ('shake hands with a friend'). Their plans are updated and they start the synchronisation sequence preparing for the handshake (shot 5). Next if there is no unexpected problems the handshake is rolled out (shots 6-8) and the avatars return to their original goal – they try to continue their journey to the desired location. However at this point they detect an avatar colliding with their chosen path so a reflexive action is submitted to the action queue

('avoid'), meaning all other actions are cancelled (shots 9-11). Eventually the avatars re-plan a new path to the goal and decide to pursue it (shot 12).

The implemented system was able to successfully generate a scene with 20 avatars walking on a sidewalk, the avatars performed both reactive (collision avoidance) and proactive (the only implemented one was handshake) actions. Colourplate 3 presents a few shots from a completed animation, Colourplate 4 presents the same scene as seen by a single avatar.

A different scene has also been generated. In this scene the avatars fight a medieval battle using shields and swords. This setup proved the generality of the architecture – apart from redefining the avatars' goals and adding a new action to the plan library ('attack and defend') no other changes were necessary in order to generate the scene. Colourplate 5 presents a few shots from the second sequence.

The evaluation criteria for the generated sequences were based on asserting lack of collisions in the scene, the fulfilment of avatar goals and on qualitative impression of believable motion. All characters achieved their goals, while managing to avoid collisions and perform additional actions.

Table 6 summarises the two experiments, detailed evaluation of the system is presented in the following section.

Table 6 Evaluation of the generated scenes

Scene	No. of avatars	Additional actions performed	Goals fulfilled	Quality of generated motion
Sidewalk	20	Yes	Yes	Good
Castle	13	No (not defined)	Yes	Acceptable

6.2 Qualitative Comparison of the FreeWill System

FreeWill is an architecture designed to conduct simulation of multiple articulated human figures participating in crowd scenes. In this respect it is similar to a few other architectures presented in Chapters 2 and 3. In this section the main features of FreeWill are evaluated by comparing it to three most prominent crowd modelling systems, chosen as being representative to different approaches to the problem of crowd modelling. The first one is the ViCrowd framework which represents a recent animation generation tool proposed by

the research community, the second one is the industrial Massive crowd system constructed and successfully employed to simulate battle scenes in the "Lord of the Rings" trilogy. Finally, since the two systems do not exhibit any learning capabilities the C4 system is also included in the comparison.

The FreeWill system, similarly to ViCrowd, is a research framework designed to simulate crowd scenes with many interacting avatars and is additionally aimed at being used as a plugin to existing professional animation packages. Although the maximum number of agents is smaller than in the Massive and ViCrowd systems this is imposed by the limitations of the animation package and can be negotiated by migrating to a better 3D engine. FreeWill comprises a cognitive architecture incorporating goals, knowledge (facts), beliefs (which are the sensed state of the world) and intentions (currently executed plan). FreeWill's actions include both synchronised inter-avatar actions based on finite state machines as well as object interaction and the primary simulation entity are individual agents. The reactive control is based on external events and motion control relies primarily on scripted high-level actions. The simulation task is fully parameterised, thus promoting modularity, and so for example an explicit scripting language can easily be added. FreeWill's main focus is on autonomous behaviour and as yet it does not allow for execution of scripted scenarios. The system includes a learning engine which allows for automatic learning of new motor actions based on simple definition of the task (only the goal needs to be provided), the agents do not possess any emotional states. Depending on the type of simulation FreeWill implements collision prediction and avoidance based on changing the velocity vector and also supplies full collision detection during the learning process. Finally the FreeWill architecture is underpinned with a software engineering model, documented in a modern graphical modelling language. This guarantees ease of implementation, expandability and system verification, promoting further growth of the architecture.

None of the above architectures includes a formal agent communication language typical to other multi-agent systems. Similarly motion is not generated using physics-based simulation, but keyframing, kinematics and scripting are used instead. The table below summarises the most important features of each of these three architectures.

Table 7 Different animation architectures

Architecture name	**ViCrowd**	**Massive**	**C4**	**FreeWill**
Purpose	Academic research	Industrial application	Academic research	Academic research
Real-time animation	Yes	No	Yes	No
Applicability	Standalone	Standalone	Standalone	Plugin
Number of agents	Tens to hundreds	Thousands	One to few	Tens (limited by the animation system capabilities)
Intelligence	Reactive architecture with group focus	State machines and fuzzy logic	Cognitive architecture with goals, plans and complex data structures	Cognitive architecture with goals and plans
Autonomy	Yes	Limited, relying on condition-action rules	Yes (high)	Yes
Behaviour control	Scripted, reactive (rules) and autonomous behaviour, user intervention possible	Rules and guided control	Autonomous behaviour and learnt motions, reactions, user interaction possible	Autonomous behaviour based on planning and reactive responses, learnt actions
Localisation of intelligence	Groups	Individual agents	Individual agents	Individual agents
Interaction	Based on smart objects and reactive 'dialogues'	Based on mocap-ed motions and reactive events	Fully simulated or learnt	Fully simulated or learnt
Execution loop	Sense-think-act	Responsive, event-driven	Sense-learn-act	Sense-think-act
Action planning	Path planning	None (system is fully responsive)	Action planning	Limited path and action planning
Reactive control	External events and 'reactions'	Event-driven	Event-driven	External events
Information categorisation	Knowledge, beliefs and intentions	N/A	Hierarchy of objects and concepts	Knowledge, beliefs and intentions (represented for individual agents only)
Modifications of character behaviour	Easy, scripting language exists, can be done in real-time	Requires direct modification of the brain structure, off-line	Requires direct modification	Modification of the parameters, off-line

Communication to perform synchronised actions	Smart objects	Not possible	Indirect (visual and audio) communication	Indirect (visual) communication (for actions involving two avatars)
Motion control	Scripted and keyframed motion	Short motion captured sequences	Short keyframed motions	Scripted motion
Scenario-ed or choreographed motion	Yes	Yes	No	No
Learning	No	No	Automatic action acquisition	Automatic action acquisition
Emotions	Yes	Yes	Yes	No
Collision handling	Collision prediction and avoidance (by changing the velocity vector) based on the state of the world	Built-in collision avoidance simplified by the predefined nature of character motions	Built-in collision avoidance and navigation	Collision prediction and avoidance (by changing the velocity vector) based on the current state and full collision detection in learning
Physics-based	No	No	No	No
Intergration with existing tools	Limited	Easy	Hard	Easy
Underpinning software model (ease of implementation)	Not documented	Not documented	Not documented	UML documentation exists
Formal communication language	None	None	None	None

In summary, FreeWill exhibits many features typically found in recent crowd simulation systems, such as limited autonomy, reactive events, or collision prediction. The main missing features compared to other similar frameworks is the lack of support for user-defined scenarios and emotions. However FreeWill extends the crowd simulation framework by adding action learning capabilities, thus removing the need for maintaining expensive libraries with motion captured or keyframed actions. Better collision avoidance, compared to existing systems, has also been added. An important aspect of the architecture is the existing UML documentation, as the design of other crowd simulation systems has not been documented in any formal way. Existing documentation allows to easily implement a similar system and also to immediately include a set of modules affirmed as necessary components of such an animation framework. Furthermore a well-defined design and development process can also be followed. A number of extensions could still be made to make FreeWill more reliable and flexible, these will be discussed in Chapter 7.

6.3 The Learning Tasks

Two learning tasks have been defined and executed using the learning module presented in the previous chapter. The first task required the avatar to learn how to open and pass through a locked door. In the second task the character had to lift a stationary object. Each task was conducted using both the forward and inverse kinematics control modes. Table 8 illustrates all experiments, details are provided below.

Table 8 The learning experiments

	Forward Kinematics	Inverse Kinematics
Door Opening Task	Experiment 1.1 Experiment 1.2	Experiment 1.3
Teapot Task	Experiment 2.1 Experiment 2.2	Experiment 2.3

The door opening task

For this task the goal of the agent is to get through a locked door. The door would be unlocked upon touching the door handle (although this could easily be extended by adding a simple animation in which the agent actually rotates the handle, as demonstrated in the second experiment). The avatar would then have to push the door and pass through. The agent is rewarded whenever its position is behind the door. The simple actions available to the agent were selected from Table 5, for the FK these were actions 1,2,3,7 (Experiment

1.1) and 1,2,3,4,5,7 (Experiment 1.2), $\Delta\alpha$ was set to *20 degrees*, step size (Δx in action 7) was *35 cm*, in all experiments $\gamma = 0.95$. Experiment 1.2 differed from Experiment 1.1 in that two additional degrees of freedom for the arm motion were added. Therefore the state space for the first experiment consisted of approximately 12000 states distributed across 4 dimensions (2 degrees of freedom for the left arm, 1 for the left forearm and 1 for backward/forward walk, the sizes of these dimensions were 12, 16, 11 and 6 respectively). Eight simple actions were available to the agent at each time step. These were three rotations – two for the arm and one for the forearm – in two opposite directions and walk along one axis (2*3+2). The two additional degrees of freedom of the hand (hand and arm rotation around the z-axis, 13 different positions for each) added for the second experiment made the total number of states of over 2 million and 12 actions per state. In the first experiment the solution is usually found in only about 250 iterations, the second experiment requires at least 1500 iterations. The lengths of the shortest solutions in both experiments were 5 simple actions.

The task of experiment 1.3 is the same as for the previous ones but the mode of control and the state and action spaces have been changed. The simple actions available to the agent are 1, 2, 3 and 5 (Table 5, inverse kinematics column), $\Delta x = 35$ cm for walk (the size of a single step) and $\Delta x = \Delta y = \Delta z = 5$ cm for the motion of a hand, $\gamma = 0.95$. Thus a 3-dimensional cube of x,y,z positions around the initial position of the avatar's hand has been defined, the last dimension was walk along one axis. Therefore the agent could choose from 8 simple actions – hand motion along 3 spatial axes in two opposite directions for each axis plus walk (2*3+2). The total simulated state space was 1 296 000, however this initial number was highly redundant and could be reduced to 6720 (8 states for the x-direction, 14 for y and 10 for z, plus 6 positions for the walk). This reduction will be demonstrated in the second task (see below). Similarly as before the solution is usually found in a few hundred iterations.

The teapot lifting task

The goal here is to lift a teapot (z co-ordinate of the teapot position has to increase). Therefore the agent is rewarded whenever the end position of the teapot is higher than the start position. The simple actions available to the agent are again selected from Table 5, for the FK these are actions 1,2,3,4,6 (experiment 2.1 and 2.2). Unlike in the previous task, the

agent was assigned an additional action (action 6 – grab an object) which improved the resulting animation and served as a means of representing the state of the teapot (grabbed/not grabbed). The learning parameters were set as follows: *Δalpha* was set to *20 degrees* in experiment 2.1 and to *10 degrees* in 2.2 and $\gamma = 0.95$. The difference between experiments 2.1 and 2.2 is therefore not in the size of the state-space but in the sampling of it – the space axes are sampled more densely and thus the state space size of the second task is larger. The state-space in the first experiment consists of about 13000 states, and for the second one is about ten times bigger (121 000 states). The dimensionality of both tasks is 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 are 7, 12, 11, 7, 2 for experiment 2.1 and 12, 20, 18, 14, 2 for experiment 2.2. Ten simple actions are available to the agent at each time step (2 for each state-space dimension, as described earlier). The minimum number of iterations for the first experiment is about 4 000 and 20 500 for the second one. Thus, despite the tenfold growth of the state-space, the minimum number of iterations increased by a factor of about 5. The lengths of the best solutions found are 9 and 14 respectively (differences resulting from different definitions of the state space).

Similarly an experiment with biped control using inverse kinematics control mode when attempting to lift the teapot has also been conducted (Experiment 2.3). The simple actions available to the agent in this case are actions 1, 2, 3 and 4 from Table 5 (inverse kinematics column), and $\Delta x = \Delta y = \Delta z = 8$ cm for the motion of a hand, $\gamma = 0.95$. Therefore the state-space is 4-dimensional and the agent can choose from 8 simple actions – hand motion along 3 spatial axes in two opposite directions for each axis plus the grabbing action. The total size of state space is only 2240 (8*14*10*2). The algorithm needs about 800 iterations to find a solution, and the best solution found was 10 actions long.

For both tasks the 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 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). For walking only the route through the door was represented as walking to the door could easily be achieved within the FreeWill system. Finally in all experiments the Q-table was represented as a lookup table and the values were initialised to 0 before the simulation.

Results of all experiments are summarised in Table 9 and Table 10, the number of iterations required to find a stable optimum solution would normally be 1.5-2 times higher than these presented in Table 10. Example sequences (Experiment 2.3) are depicted in Colourplate 6.

Table 9 Resulting action sequences for the teapot problem

	Example action sequence	**Length of the shortest sequence found**
Door task:		
FK control (1.1)	1 1 6 6 6	5
FK control (1.2)	1 4 6 6 6	5
IK control (1.3)	2 4 6 6 6	5
Teapot task:		
FK control (2.1)	3 4 0 4 4 2 1 8 7	9
FK control (2.2)	4 0 0 0 0 2 2 6 1 2 4 1 8 7	14
IK control (2.3)	1 1 2 4 4 2 2 0 6 0	10

Table 10 Summary of the learning experiments (*simulated state space, **sufficient state space)

	FK			IK		
	State space (dimensio-nality)	Actions/ state	Min. no. iterations to find a solution	State space (dimensio-nality)	Actions/ state	Min. no. iterations to find a solution
Door	12672 (*4D*)	8	200	1 296 000* (6720)** (*4D*)	8	700
	2 141 568 (*6D*)	12	1500			
TeaPot	12936 (*5D*)	10	4000	2240 (*4D*)	8	800
	120 960 (*5D*)	10	20500			

Convergence and average reward graphs for the teapot simulation are depicted in Figure 35–Figure 40. Figure 35, Figure 37 and Figure 39 present convergence graphs depicting the total Q-value (sum of Q-values from the entire Q-table) as a function of the number of epochs. The fastest and most distinct convergence can be observed for Experiment 2.3 (IK control) and the slowest is for Experiment 2.2 (FK with larger state-space). Similarly Figure 36, Figure 38 and Figure 40 present a sum of rewards received by an

agent at the end of an epoch as a function of the number of epochs, calculated as a sum of all Q-values and rewards collected by the agent in the states visited between the start state and the absorbing state. In the first and last figure the curve is relatively flat for most of the simulation. This is because the agent reaches the predefined limit on the number of visited states without receiving a reward relatively often. In such a case the agent is placed in a new random state and his reward sum is set to zero. When the constraint is released (toward the end of the simulation) the longer spikes represent epochs in which the agent was stuck in local minima for a long period of time and managed to accumulate high rewards. When this constraint is not imposed at all (Figure 38) the sum of rewards grows during the initial exploration phase and eventually stabilizes for the rest of the simulation. The resulting animation solution is identical to the previous case, only the way of exploring the state space proceeds in a different manner. Sequences of shots from resulting animations are presented in Colourplates 6–10.

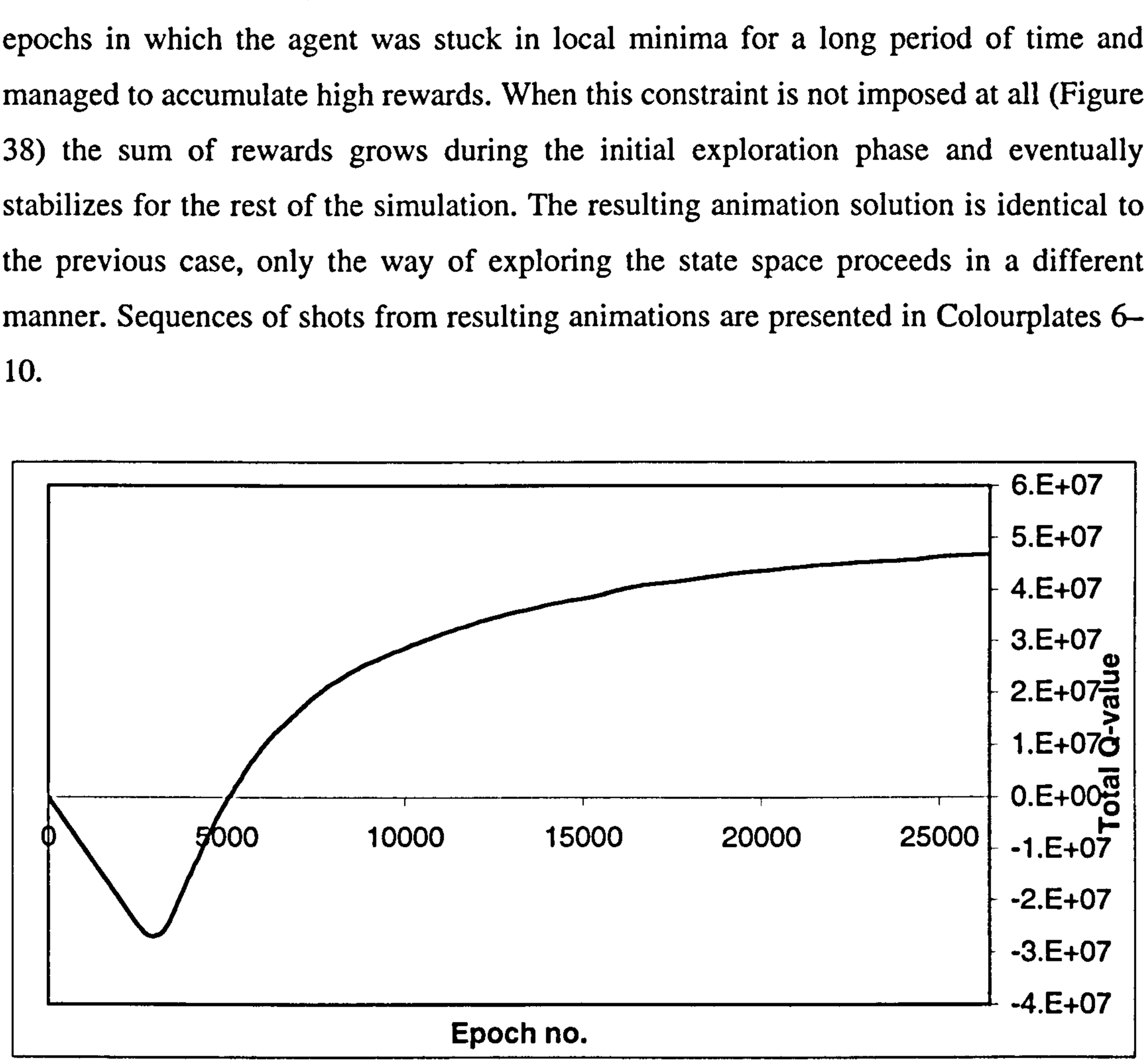


Figure 35 Convergence graph for the FK teapot problem (Experiment 2.1)

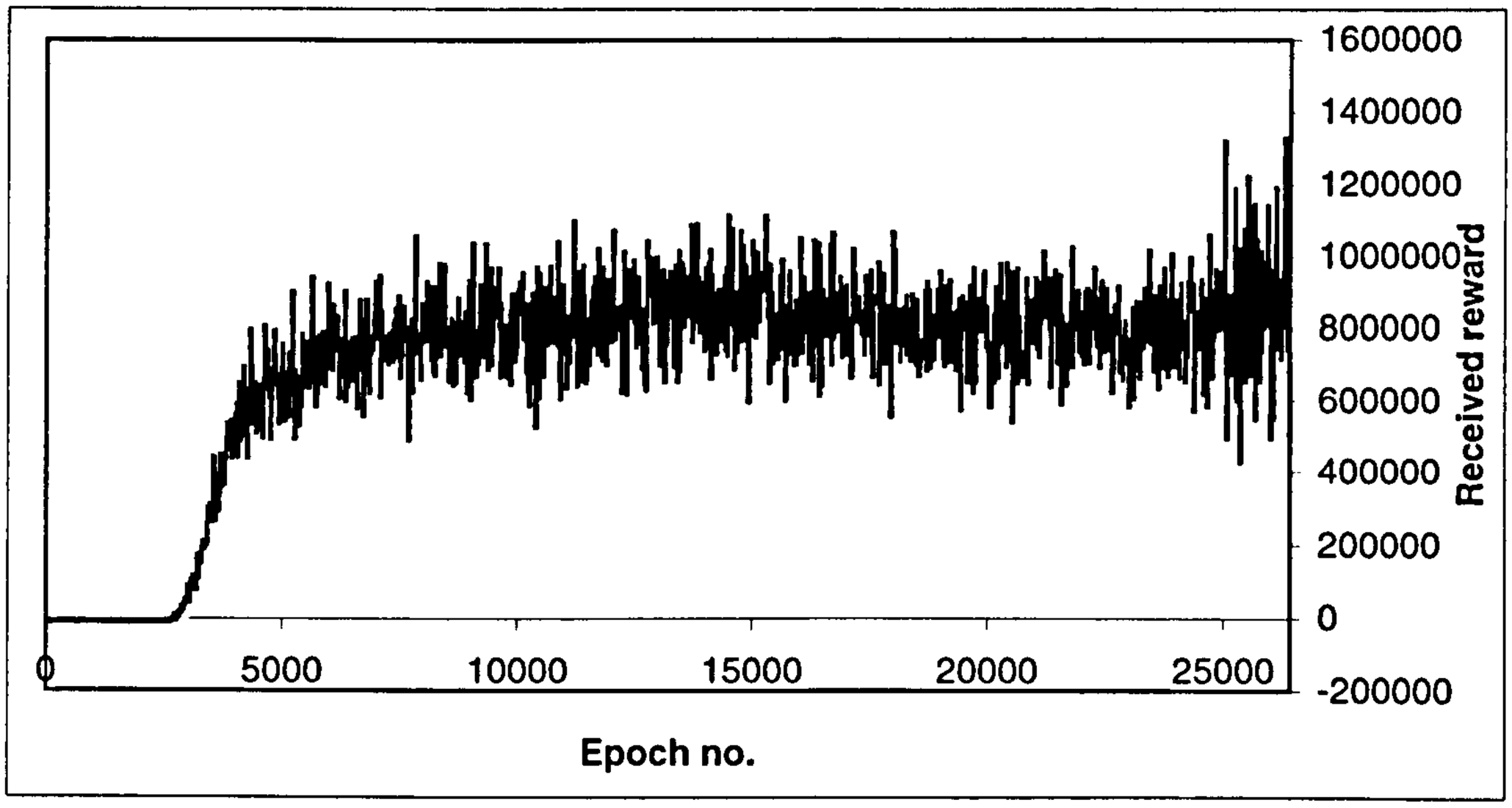


Figure 36 Reward received by an agent (the peaks at the end are caused by removing the limit on maximum number of random actions, averaged over samples of 20 values), Experiment 2.1

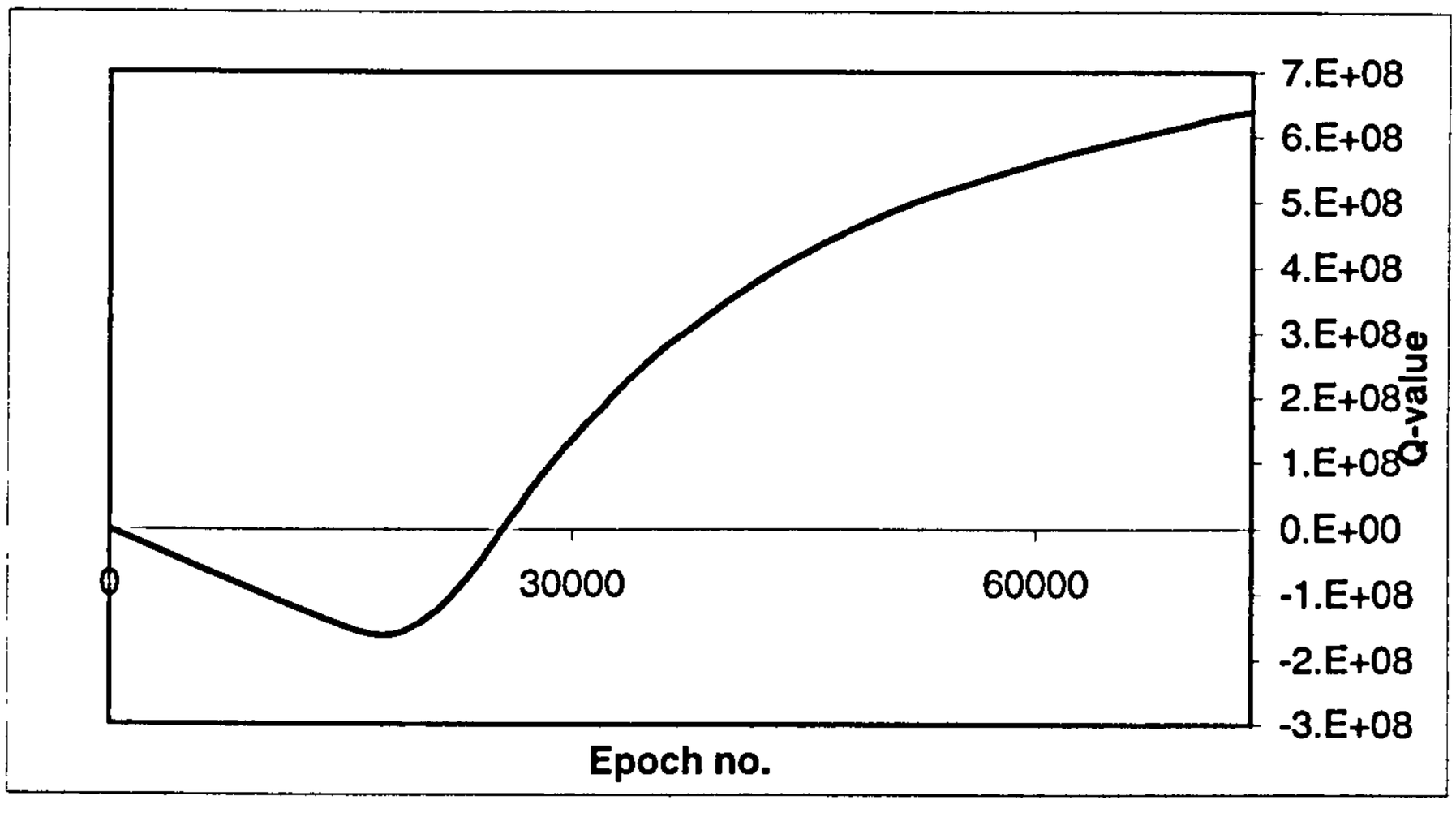


Figure 37 Convergence graph for the FK teapot problem (Experiment 2.2)

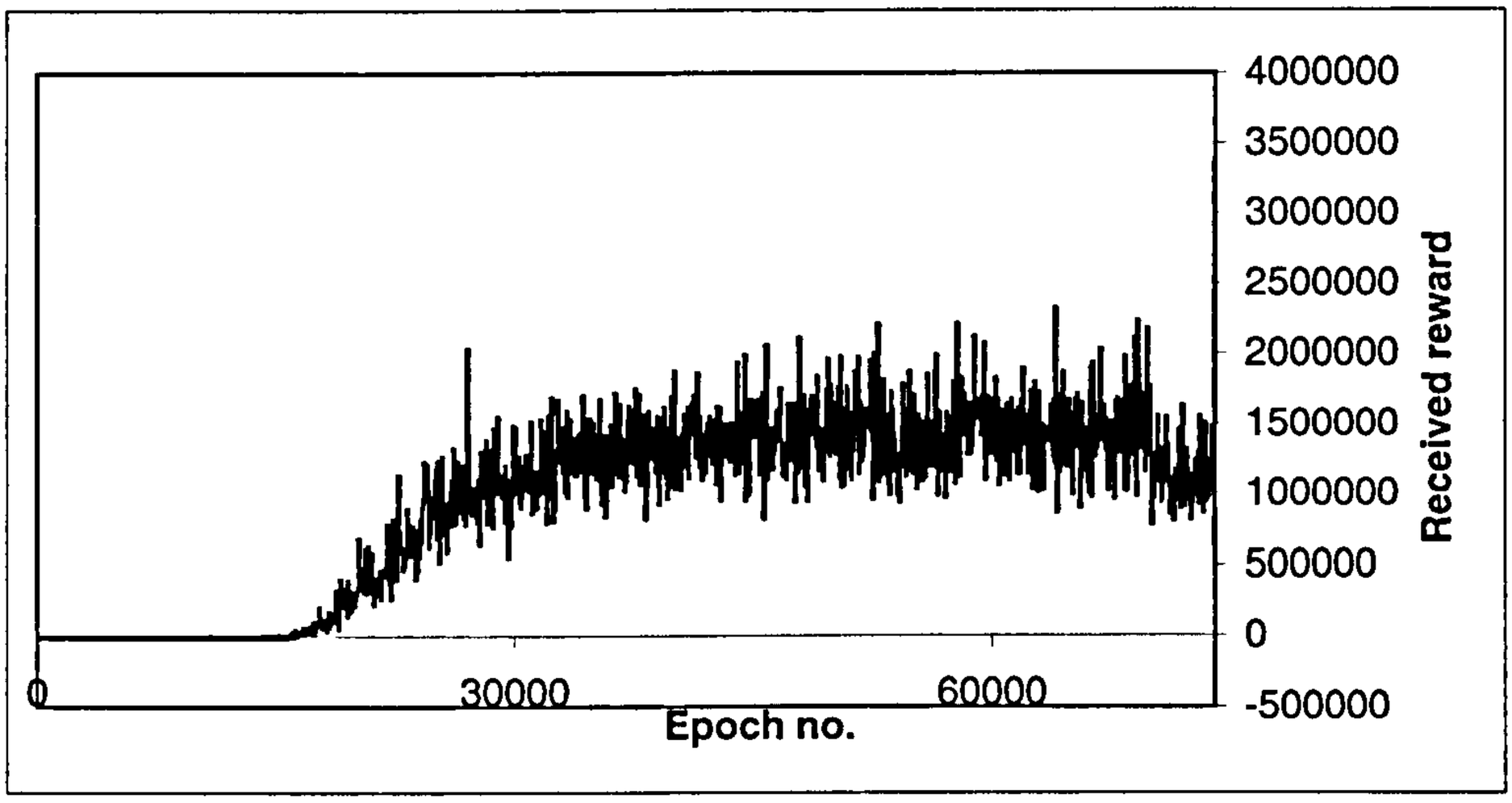


Figure 38 Reward received by an agent (the limit on maximum number of visited states during the exploration phase was increased ten times in this experiment, compared to Experiment 2.1, so it almost did not affect the curve), averaged over samples of 20 values, Experiment 2.2

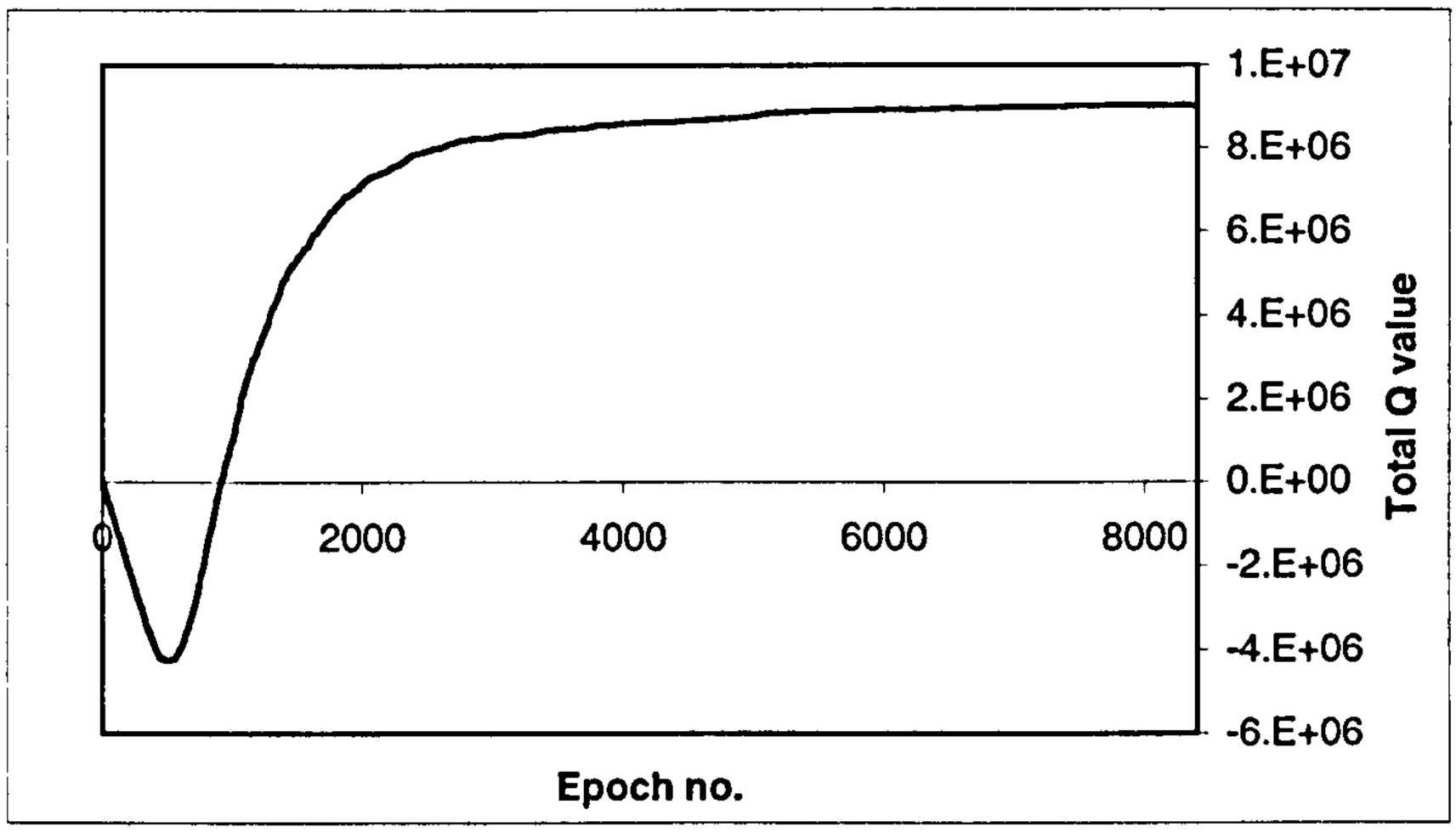


Figure 39 Convergence graph for the IK teapot problem (Experiment 2.3)

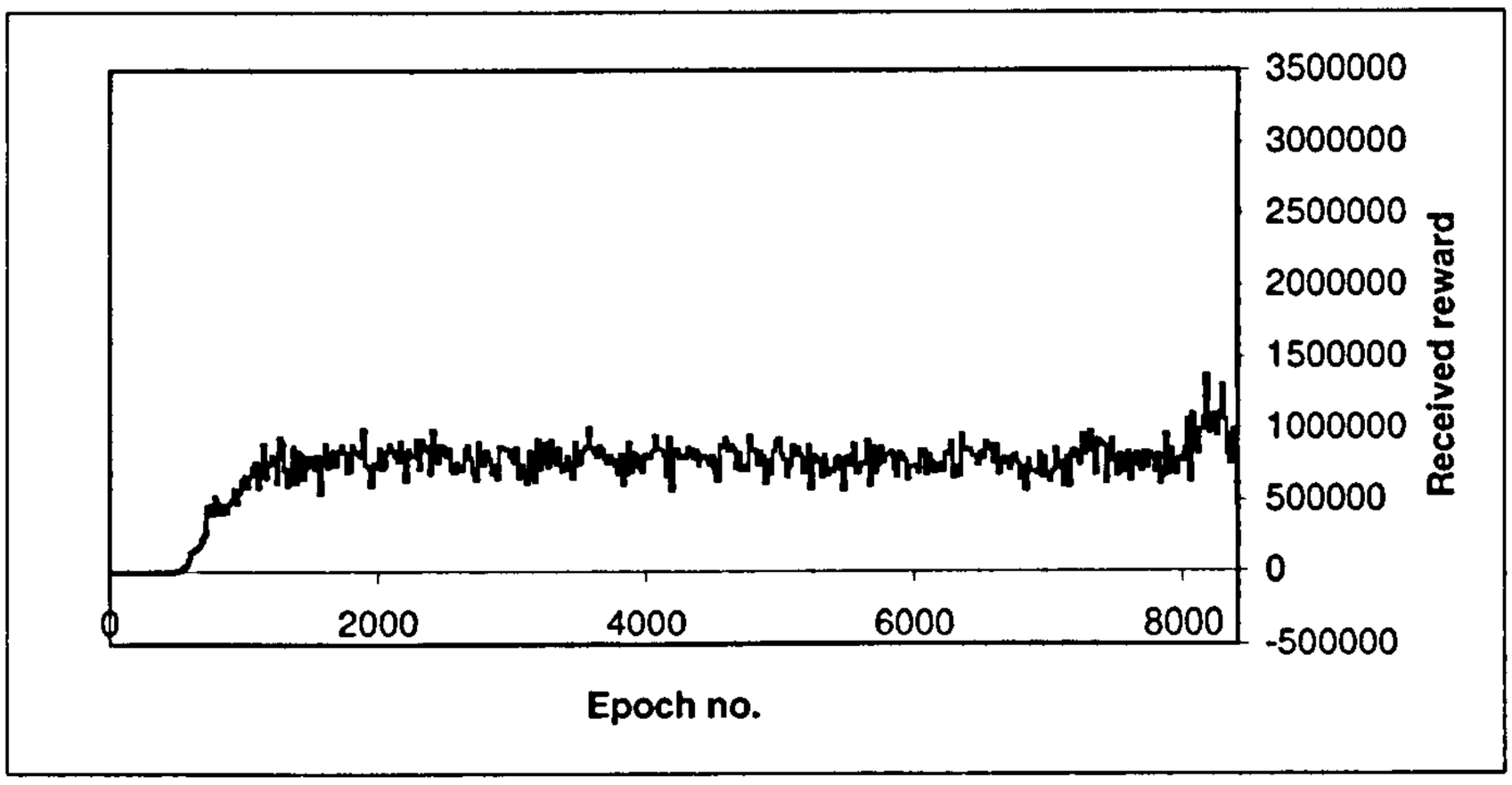


Figure 40 Reward received by an agent (the peak at the end is again caused by removing the limit on maximum number of steps per epoch in the exploration phase), averaged over samples of 20 values, Experiment 2.3

6.4 Time Requirements of the Deterministic Algorithm

This section compares the execution time of the algorithm on a computer with Pentium II 400 MHz processor and 384MB RAM running version 3.1 of the 3D Studio Max package and Character Studio version 3.0. The execution times presented here are intended only to allow for relative comparisons as the actual timing may be influenced by many factors such as processor speed and workload, communication speed between the COM interfaces, the quality of the random number generator (e.g. if the same, already performed action, is repeated a number of times) and also by the working speed of the animation package. Additionally it is difficult to assess the overheads introduced by the inter-application interfaces, plugin communication and execution and collision detection routines. Therefore the purpose of presenting them here is to allow the comparison between the Q-learning implementation and an exhaustive search performed in similar conditions on the same task state space.

The times obtained differ considerably for each of the simulations presented above. For example, despite defining a relatively large state space for the door opening task, the actual part of the state space necessary to solve the problem is only a small fraction of it. Hence the algorithm produces the results very quickly (and in a small number of iterations) – 3 minutes for Experiment 1.1, 8 minutes for 1.2 and 9 minutes for 1.3. For comparison, it takes only 30 seconds (6 times faster) for experiment 1.1 to run on 1,5 GHz processor with 0,5 GB RAM. The results of experiment 2 were as follows: over

5.5 hours for experiment 2.1, 3-4 days for experiments 2.2 and 130 minutes for 2.3. The timing however strongly depends on the actual number of iterations performed and increases quickly as the agent explores the state space, finds a solution and then gets stuck in local minima. Times for all experiments are shown in Table 11. The table presents average numbers of iterations necessary to reach any correct solution for all three experiments. Similarly, numbers of iterations necessary to reach the best solution for a given task (as found at the end of simulation, after convergence had been reached), and times necessary to accomplish these given numbers of iterations using the experimental setup defined before are also presented.

Table 11 Simulation times for the Q-learning implementation

Experiment	*Avg. no iterations to reach a solution*	*Avg. no iterations to reach the best solution*	*Avg. time for first solution found*	*Avg. time for the best solution*
FK (2.1)	3800	5000	5.5 hours	6 hours
FK (2.2)	20500	45000	18 hours	4 days
IK (2.3)	800	1500	80 minutes	130 minutes

For comparison, assuming that the avatar control system performs 12 actions per second for the FK algorithm and about 9 for the IK task (which are the numbers of actions per second obtained during the learning task) the times necessary for exhaustive search of the state space would be as presented in Table 12. The length of the sequence was assumed to be shorter by two actions compared to the best solutions found by the Q-learning algorithm to allow for possible better solutions, this is 12 and 8 actions for FK and IK respectively (please refer to Table 9). Additionally the size of the action set available in each state was reduced by 2 for the forward kinematics case (only the more complex task was considered) to 8 actions and by 1 in the case of the inverse kinematics control, which means 7 actions were considered in each state. These reductions approximate the pruning of the search tree as a result of collisions and other forbidden movements. The time estimates ($\hat{t}$) were calculated according to this equation:

$$\hat{t} = \frac{N_A{}^M}{N_S} \quad (11)$$

where N_A is the number of actions which the agent can choose from in each state, M is the assumed length of the optimum sequence and N_S is the number of actions per second which can be executed by the software when the agent is continuously performing the actions.

Table 12 Simulation times of an exhaustive search

Control method	*Actions per state*	*Sequence length*	*No. actions per second*	*Time [seconds]*
FK	8	12	12	$5.7 * 10^9$ (≅181 years)
IK	7	8	9	$6.4 * 10^5$ (≅ 7.5 days)

The results depicted in Table 12 show that an exhaustive search performed on the FK state space would require over 180 years to complete (assuming that the optimum sequence is shorter by two actions from the best Q-learning solution found). The result for the IK is only about 7.5 days, this is nevertheless still substantially longer then the results obtained using Q-learning. The difference for the estimated search times between the FK and IK modes is caused by the fact that the IK state space is noticeably smaller, however the FK results demonstrate that as the space size increases the search quickly becomes unfeasible, even when it is still possible to find a solution using the Q-learning method.

6.5 Learning Using the Non-deterministic Algorithm

This section presents results obtained when applying the non-deterministic update of the Q-learning algorithm to the task of action acquisition. The task implemented using this technique is the IK-controlled teapot problem. The state space is the same as in the deterministic implementation, and the length of the shortest solution is also 10 simple actions (Table 13). The convergence is reached faster – in approximately 800 interactions as opposed to about 3000 in the deterministic case and the time necessary to reach the optimum solution is shorter as well – about 90 minutes on average (550 iterations). The convergence is also more pronounced (Figure 41). This suggests that the non-deterministic version of the algorithm generates comparable results in a shorter amount of time. Figure 42 presents the stabilisation of the total sum of rewards with the number of epochs.

Table 13 Resulting action sequences for the teapot problem

Teapot task:	State space	Actions per state	Example action sequence	Length of the shortest sequence found
IK control	2240 (*4D*)	8	2 1 4 4 1 2 2 0 6 4	10

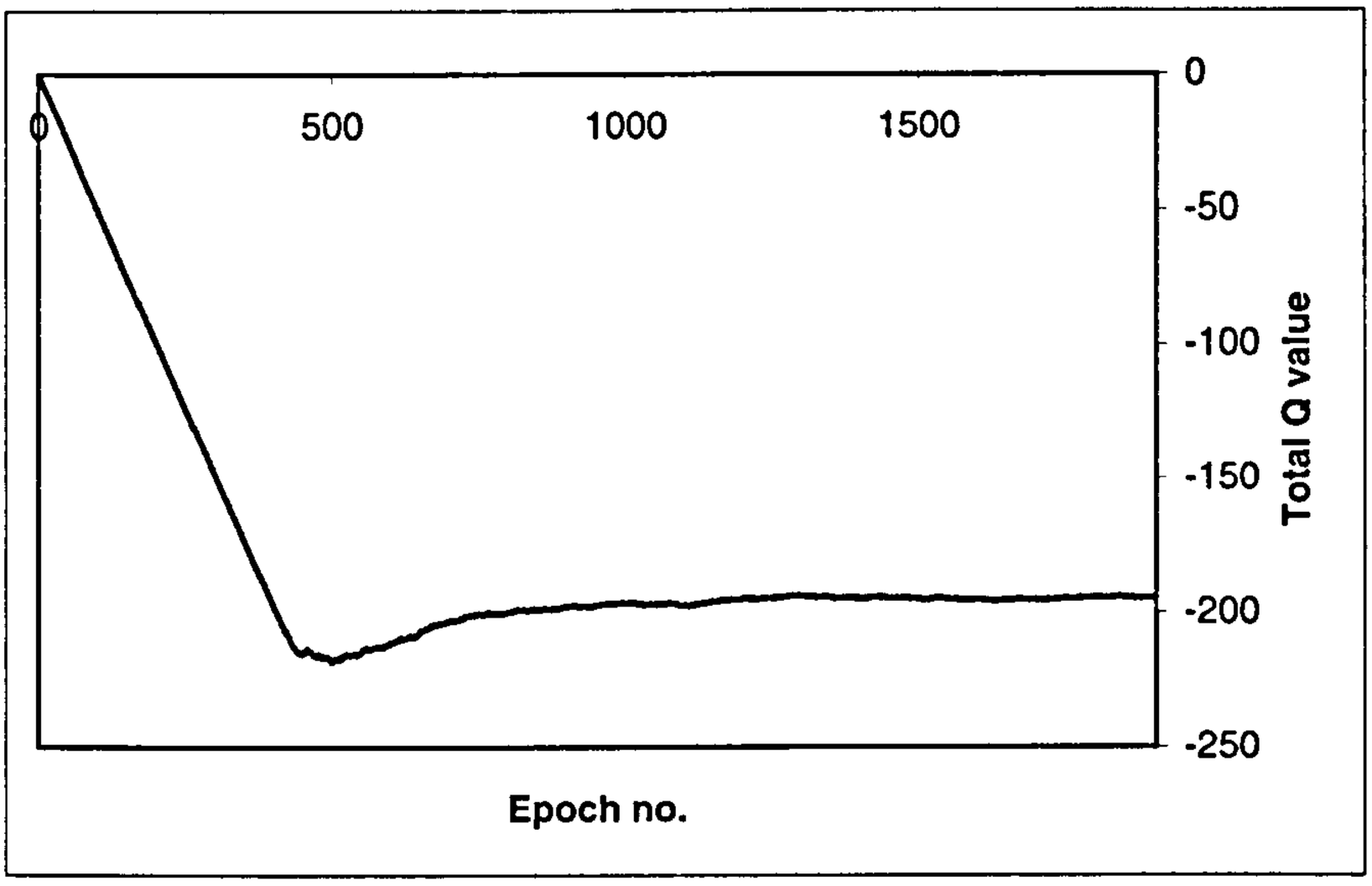


Figure 41 Convergence graph for the IK teapot non-deterministic problem

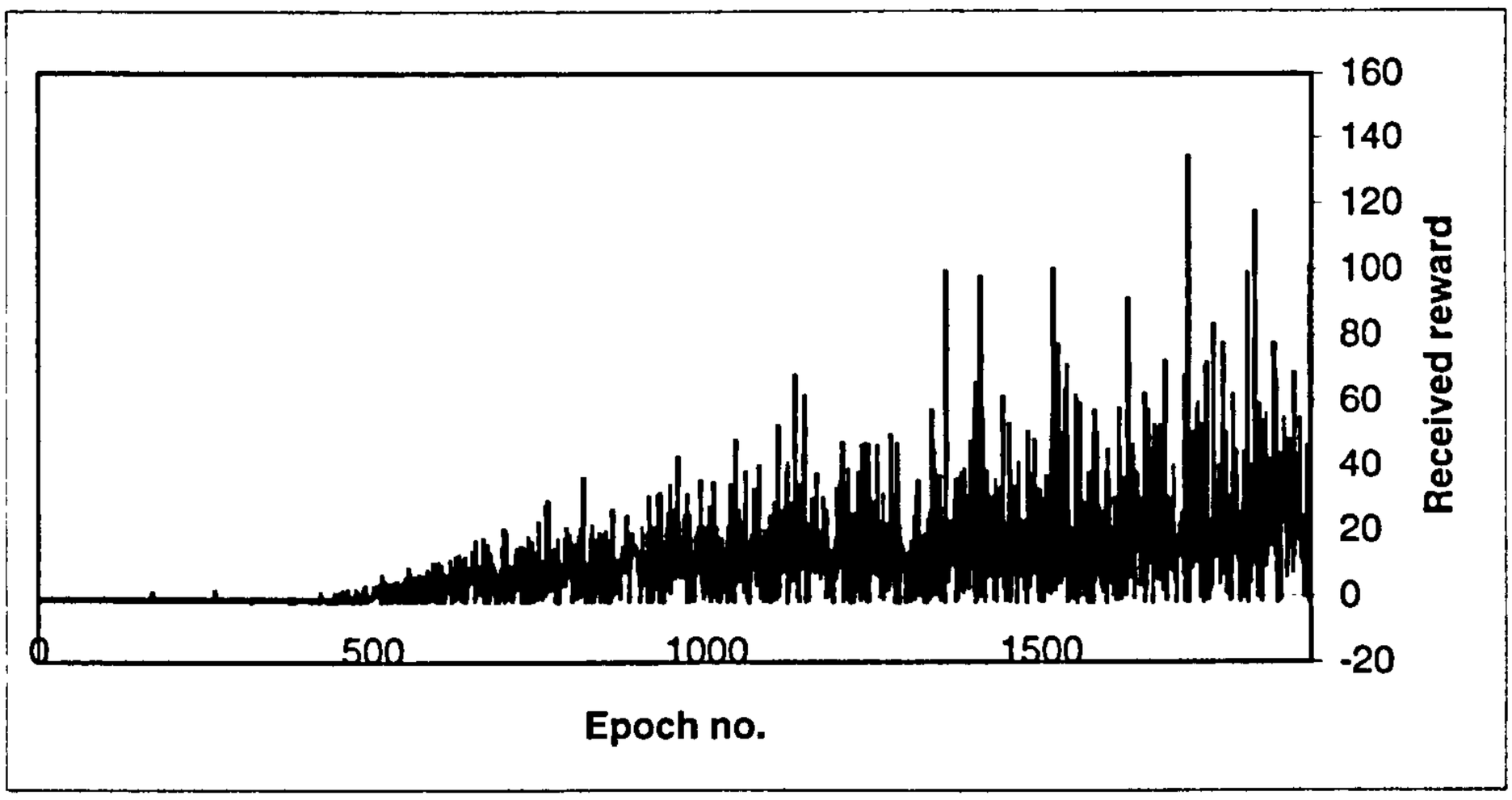


Figure 42 Reward received by an agent in the IK teapot non-deterministic problem

6.6 Learning with Non-deterministic Action Selection

An additional simulation with the non-deterministic update has also been executed, in which the outcome of the action selection was randomised in some percentage of cases. The action selected by the agent according to its Q-table was replaced with a random

action with some predefined probability. The results of that simulation for different levels of action randomisation are presented in Figure 43.

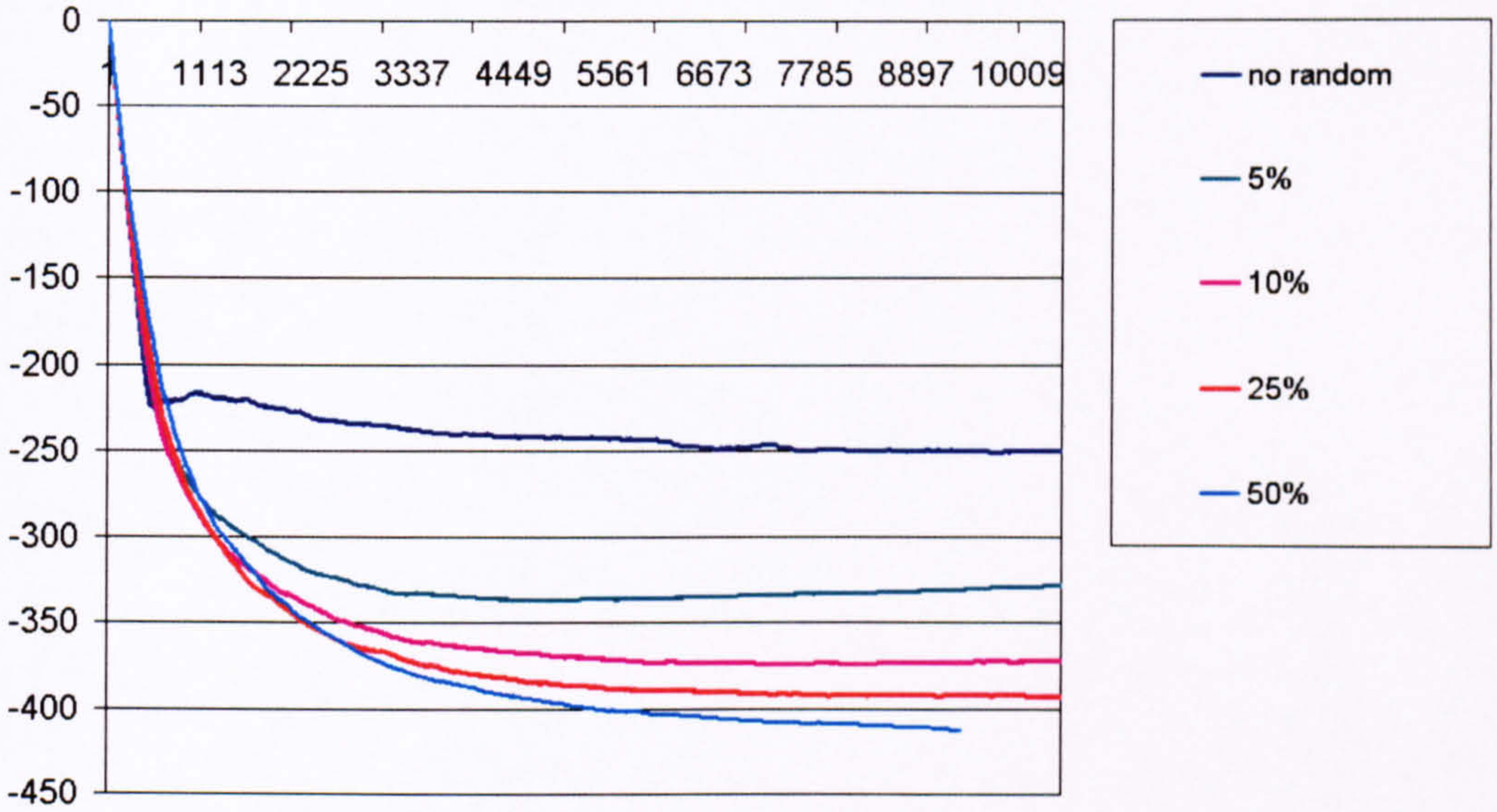


Figure 43 Convergence for randomised action selection updates

As presented in Figure 43 the speed of convergence is decreased with the growth of the uncertainty of the action selection mechanism. However, convergence is still reached even for relatively high uncertainty levels. Although the results of this experiment do not have much significance in a fully predictable animated landscape, they suggest possible utilisation of the action acquisition scheme for robotic environments.

6.7 Evaluation of the Learning Results

The results presented in the previous sections indicate that the IK learning mode is faster and easier to implement. The convergence is reached in a smaller number of iterations, compared to the FK case, and is more pronounced (the curve is flatter). However the ultimate assessment can only be made upon analysing the resulting animations. Because of the relative simplicity of the door opening tasks, the results obtained for this task are not very different (see Colourplate 11 for comparison with real data). However noticeable differences exist in the second learning task. In experiment 2.1 (rarely sampled forward kinematics control) most of the results are of insufficient quality – the motions are too jerky and inaccurate. This has been improved substantially by increasing the sampling rate of the state space – results obtained from experiment 2.2

are of sufficient visual quality. Some artefacts still remain however – this mainly concerns unnecessary motions and especially a zigzag-like way of approaching the teapot which is present in some animations, but the result is resembling human motion with a sufficient detail as demonstrated in Colourplate 7 and 8. Indeed the way of executing the action achieved using the FK mode of control matches the way of executing the same action by a human actor without giving her additional guidelines prior to performing the task.

Results yielded by Experiment 2.3 are also interesting. First of all the state space is substantially smaller than for the FK experiments and therefore the solutions are found in fewer iterations. The resulting motion looks realistic, despite the fact that the human actor did not initially perform the action in the way suggested by the IK solution. This does not mean humans cannot perform the lifting task in this way as demonstrated in Colourplate 9 and 10, and the reason for this way being less natural is only in the fact that the table was relatively high. Reducing the height of the table changes the way of performing the task by humans (the hand does not have to be moved around the table). Moreover, the generated motion still looks natural, and contains fewer unnecessary artefacts compared to the FK solution, because IK control implicitly rejects some of the unnecessary moves. The IK state space can be represented in a more compact way (only three values need to be stored regardless of the hand position). This however causes problems when more expressive motion or combination of different modes of control are required (for the door opening task it was necessary to combine IK hand control and walking), as the representation of the state space for such extensions is more uniform when using the FK approach. The main problem with FK approach is its extensibility – additional degrees of freedom very quickly expand the state space and substantially increase the number of iterations required to find a solution. Therefore tasks for which more than 6-7 degrees of freedom is necessary may have to be simulated using the more compact IK control.

It also appears that the non-deterministic algorithm generates the solution faster than the deterministic one, maintaining the same quality of the results. Future implementations therefore should rely on this version of the Q-learning technique.

The number of iterations required to reach convergence differs from about 700 to about 20 000 (and amounts to 100 000 only for the most complex problem), with about 100-130 state transitions on average per iteration. These are relatively small values, for example Laurent and Piat (2001) used 2 million steps for their enhanced Q-learning algorithm and Aljibury *et al* (2002) report 650 million Q-table updates during a 10-day simulation. The reason for this discrepancy is a rather small state space for the IK-controlled problems and also the modified exploration strategy in the final stages of the algorithm execution (bias towards one starting state). The result is comparable to other work for the most complex FK task (over 10 million updates).

In summary the learning technique presented here generated satisfactory results when applied to a non-trivial task. Comparison of the generated motion to the motion of a human actor indicates that the sequence is sufficiently realistic to be applied in a crowd scene. Although the technique appears to be difficult to scale up, some extensions, especially using the IK control, will be possible to it, allowing to add a few additional degrees of freedom to simulate a task requiring the use of both hands or the head motion by the simulated biped. Additionally results obtained with a better hardware configuration suggest that a modern computer will improve the learning times by at least one order of magnitude.

6.8 Metrics Applied to the Learnt Sets

Based on two result sets generated by the learning algorithm the Local Distance Metric and the Global Distance Metric have been calculated. The first result set (Set 1) contained 651 sequences each consisting of 14 actions obtained from experiment 2.2, the second set contained 72 sequences (Set 2) with 11 actions each, generated by experiment 2.3. The reduced sets generated by the metrics are presented below.

LDC applied to Set 1 generated 120 sequences with the maximum distance of 12 (all pairs have been included in Appendix E).

LDC applied to Set 2 generated 72 pairs of sequences with the maximum distance of 8. An example of such a pair has been presented in Colourplate 6.

GDC applied to Set 1 generated 40 sequences most distant from the averaged sequence, the maximum distance was 8.

GDC applied to Set 2 generated 4 sequences most distant from the averaged sequence, the maximum distance was 6. The four resulting sequences were:

2 4 1 1 4 2 2 0 4 6 0 (seq 25, Colourplate 12)
2 4 1 4 1 2 2 0 4 6 0 (seq 29, Colourplate 12)
4 4 1 1 2 2 2 0 4 6 0 (seq 60)
4 4 1 2 1 2 2 0 4 6 0 (seq 65)

Although these sequences are the most distant from the average, they are relatively similar (see Colourplate 12) and therefore an extension of this approach has been proposed. The resulting set proposed for addition to the plan library should include sequences for which the GDC generates the highest and the smallest values thus guaranteeing to include both most average and most different sequences. Rerun of the GDC algorithm with a distance of 0 resulted in further 4 sequences being proposed for the action library:

1 1 4 4 4 2 2 2 0 6 0 (seq 8, Colourplate 13)
4 1 1 4 4 2 2 2 0 6 0 (seq 41, Colourplate 13)
4 1 4 1 4 2 2 2 0 6 0 (seq 46)
4 1 4 4 1 2 2 2 0 6 0 (seq 51)

This extension of the algorithm applied to Set 1 generated additional 10 sequences.

The results indicate that for large sets of sequences the sets of sequence pairs generated by the LCD are still too large to be analysed individually. The GDC metric reduces the amount of resulting sequences, however the most distant solutions appear to be very similar. The proposed extension of generating both most distant and most average solutions and selecting a few examples from each set overcomes this problem.

6.9 Other Applications of the Learning Technique

A filled Q-table contains optimum transitions from any state to the goal state. Therefore the results of the learning tasks demonstrated in this chapter can be used to create animations from any starting position of the agent thus adding more than one specific action to the plan library at a time. It may even be possible to store the learnt Q-tables rather than single action sequences and use them whenever the agent needs to perform an action starting from a state present in the Q-table and ending in a goal state defined

for the specific learning task for which the Q-table was generated. For an example, see Colourplate 14.

Similarly the Q-learning technique may be used to learn the same action for a different configuration of objects (for example the size of the table or teapot may be different) and encode this different configuration as a set of varying preconditions for the same action. Equally the size of the avatar itself can be changed and a new solution for a different character may be generated (see Pollard and Hodgins, 1998 for a different solution to this problem).

Another benefit of using the learning technique for action acquisition is that the resulting sequence comes in a form of a script, which can then easily be manipulated and parameterised and also incorporated into other animation tools. This is in strong opposition to the keyframing and motion capture based approaches which, although easily portable and capable of delivering realistic motion, are nevertheless difficult to modify using automatic methods. This advantage is also utilised when calculating the metrics presented in the previous sections, as it is otherwise difficult to calculate metrics for motion sequences without underlying semantic representation.

Finally the technique can be applied to other research domains such as robotics, provided the robot can already perform basic actions such as walking.

Chapter 7 – Conclusions and Future Work

The aim of this thesis was to create an architecture capable of automatic generation of crowd scenes with many interacting human-like characters. Research was conducted in two stages – first of all, a general architecture for crowd simulation was proposed and an initial prototype was built. Second stage was to enhance the action library by adding algorithms for fully automatic acquisition of actions. This chapter discusses the main achievements of the project, highlighting the strengths and weaknesses of the proposed solutions.

7.1 Analysis of the Industrial and Research Systems

As discussed in Chapter 2 some of those packages offer limited character and crowd extensions, but the main identified problems included lack of AI-based frameworks, collision detection and focus on animating rather than directing the crowd sequences. The Massive system, for example, partially overcomes some of the problems but it remains a specialised engine, which also heavily relies on a large database of motion captured clips, which are usually difficult to obtain and often must be recorded for specific tasks.

Although many academic animation architectures present a great potential, they are usually too specific to generate multiple crowd participants able to interact with each other and the environment. Existing crowd systems, on the other hand, do not offer any action acquisition mechanisms, and often rely mainly on pre-recorded scenarios without sufficient scope for character autonomy. Physics-based systems at the moment do not offer any intelligent capabilities and are usually computationally very expensive. Agent systems address the problems of agent communication, social behaviour and knowledge representation but they are rather generic, without explicitly addressing the problem of implementing animated agents. Additionally none of the reviewed frameworks offered a way of integrating with existing professional tools, the design process has not been documented nor a uniform design and documentation notation selected.

The importance of a better crowd modelling tool has been pointed out in this work. Such an architecture should comply with the recent advances in software engineering and should also easily integrate with existing animation systems.

7.2 Main Features of the Proposed Framework

FreeWill extends both the SAC and FCA architectures, using some of their underlying concepts. It recognises the need for multilayered character structure (necessary to represent the creature's body), simple motion patterns, and cognitive capabilities. It also uses the agent-derived concepts of beliefs, goals and intentions and maintains plan libraries. FreeWill represents the view that events should be represented as entities external to the agent. This is based on the assumption that any process not comprising part of the agent should not be included in its design, including sensory events and action execution. Such an approach simplifies the design of the system, as the sense-think-act loop only relies on one type of event. FreeWill addresses the problem of collision detection, missing in many of the professional packages, collision detection is handled by each character separately. The motion layer of the FreeWill agents is managed by an external animation engine and relies on kinematics motion. Agents maintain a list of currently executed actions and goals, they can plan ahead and sense the environment. Full UML documentation of the system is provided and several different communication techniques with the animation package are proposed. However FreeWill does not address the problem of representing character emotions and scenario prototyping is not possible within the framework. Also action acquisition remains a problem, as all new actions have to be manually scripted. Therefore an important extension to the FreeWill architecture has been developed that allows for automatic acquisition of actions. The underlying algorithm relies on one of the machine learning techniques, namely Q-learning.

The learning requires a discretisation of the state space and also an explicitly defined set of simple actions. At this stage the only required input is the goal of the learning task, and the action sequence fulfilling the goal is found automatically. Two control mechanisms can be used to learn the new action – the forward and inverse kinematics, and additionally both the deterministic and non-deterministic Q-learning algorithms are proposed for investigation. Such a definition of the learning task however generates

multiple solutions, posing a problem of how to select a few most suitable. In the case when only a small number of such solutions is found, these can be presented to the human user for selection or all of the solutions can directly be added to the action library. However in most cases the number of solutions found exceeds a hundred. To allow for automatic selection of learnt sequences a number of metrics is proposed. The aim is to select a small number of varying action sequences.

7.3 Outcomes

Upon implementing the FreeWill architecture, the observed results confirm that the current set of AI tools (such as planning, machine learning, finite state machines and agent-oriented approach) generates a crowd simulation framework which is significantly more versatile than existing state-of-the-art approaches, providing additional functionalities. The architecture generates believable animation sequences (Colourplates 2-5) and it is easy to redefine character actions and goals. The number of participating characters is limited only by the capabilities of the animation packages used, and the speed of the microprocessor. In many respects the architecture outperforms existing approaches by applying the learning mechanism, providing better collision detection and easy integration with existing tools. The learning results for the simulated tasks are comparable with real human motion (see Colourplates 7-11), can easily be parameterised and extended. The results indicate slight superiority of the non-deterministic Q-learning algorithm, and also show that the IK control method promises greater scope for extendibility. The IK-controlled, non-deterministic algorithm is also the fastest converging one. Although the learning system is far from real-time performance, the pre-processing must only be done once, and a sufficiently rich action library will allow for generation of scenes with avatars presenting extensive behaviour "repertoires". The metrics also fulfil their role (Colourplates 12, 13), although the application of the global distance metric had to be modified as a result of the experiments conducted. The proposed set of actions should consist of both maximum and minimum distance actions.

7.4 Main Contribution of the Project

This section again shortly summarises the main contribution of the thesis.

State of the art

This work presented an extensive overview of the field of character animation including industrial and research systems. The review included such domains as animation architectures, motion capture, crowd simulation and rapid animation prototyping. The suitability of different agent design methodologies to character creation has also been considered. It has been established that there is a potential for such applications to be made in automatic crowd modelling systems. Additionally Reinforcement Learning in the context of character animation has been discussed.

A new architecture

Based on the conclusions from the literature review a new hybrid animation architecture called FreeWill has been proposed. It combines the multiagent as well as the animation concepts. The architecture defines all main components of a crowd simulation architecture and a design framework documented in UML. The main distinguishing aspects of the architecture are uniform representation of actions and external events. Other important features are also goals, plan libraries and collision avoidance. The system allows modelling both inter-agent and agent-object actions. A prototype version of the proposed architecture has been implemented and evaluated. Three types of possible communication with professional animation packages have been identified and applied to the architecture.

Action acquisition

The proposed architecture has been extended to accommodate automatic action acquisition based on machine learning. The chosen algorithm was Q-learning and both the deterministic and the non-deterministic versions of the algorithm have been applied to the task of motor learning. The results are complex actions with quality comparable to real human motion. Two modes of control – forward kinematics and inverse kinematics were used in conjunction with the learning algorithm. This allowed for comparison of the two techniques along such dimensions as size of the state space, learning time and quality of the results.

Animation metrics

A number of metrics have been proposed to select a representative sample of solutions from the results generated by the learning algorithms.

7.5 General Assessment of the System

The proposed architecture has been qualitatively compared with three other important architectures – two representative to academic research and another one applied by the industry to generate special effects in film postproduction. As a result the FreeWill architecture appears to offer a similar set of features regarding knowledge representation, behavioural modelling, reactive control, and autonomy. A number of properties have also been identified in which the new architecture outperforms the existing frameworks. This includes learning, collision avoidance algorithms applied and the software engineering approach. Also the learning algorithm generates realistic results in a relatively small number of iterations and allows for motion prototyping or even complete replacement of other expensive techniques for crowds where fine detail is not a crucial factor. The architecture integrates easily with existing professional packages what was one of its design principles.

It is therefore believed that the problem of proposing a flexible character animation framework based on current advances in Artificial Intelligence has been solved.

7.6 Possible Extensions

A number of extensions can be made to the presented architecture. Some most obvious would involve implementing the features missing in the current prototype. This might include replacement of the current ad hoc planning routines with a complete planning algorithm based on one of the inference engines. A method of automatic import of predefined 3D scenes into the system and labelling the scene elements would also enhance the capabilities of the existing implementation.

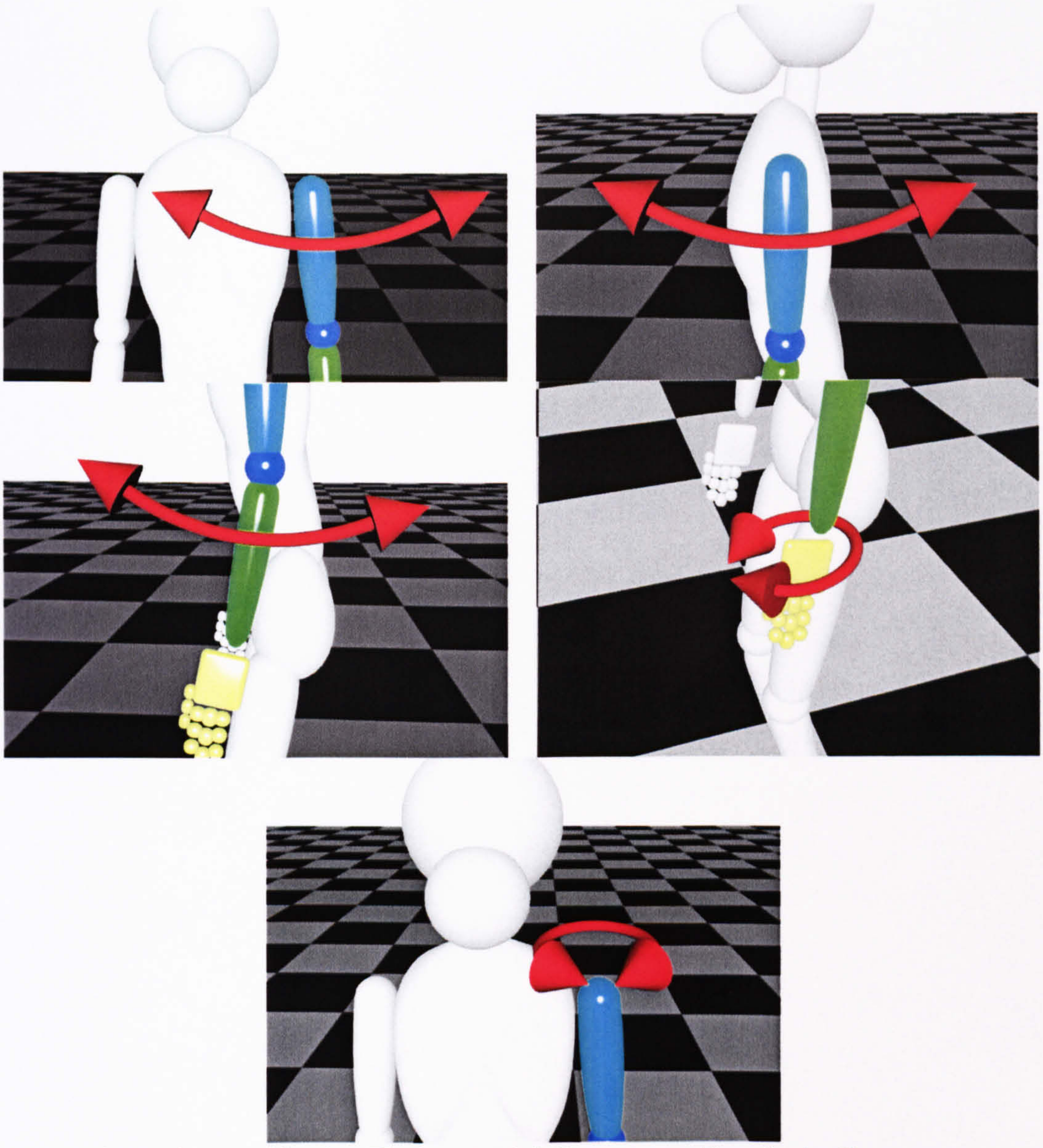
It would also be useful to expand the architecture by adding some features present in other systems, as identified in Chapter 6. Adding scenarios and other motion control approaches such as import of motion capture files or keyframed sequences should not

be too difficult, especially if the rendering is done in a professional animation system. An increase in the sampling of the IK state space may also be necessary. Although the assumed sampling generates good results, a better sensitivity would be necessary in task relying on finer object control. The state space itself should be extended to allow for more complex tasks. This might include defining additional degrees of freedom to accommodate other limbs (e.g. the right hand), spine and head. Current results suggest that 3-4 additional degrees of freedom in the IK experiments should be a feasible extension. A useful approach might be to start learning with a coarsely defined state space and gradually increase the sampling rate in the regions of the state space visited when generating the approximate solution. Some experiments involving the learning parameters γ and α may also be attempted to identify values yielding highest convergence speed. Finally modelling of character emotions could be proposed, which would however require relatively large modifications of the architecture (for example the characters should be able to vary their motion to allow them to display emotions, some emotion representation scheme would also be necessary, events influencing character emotions must be identified).

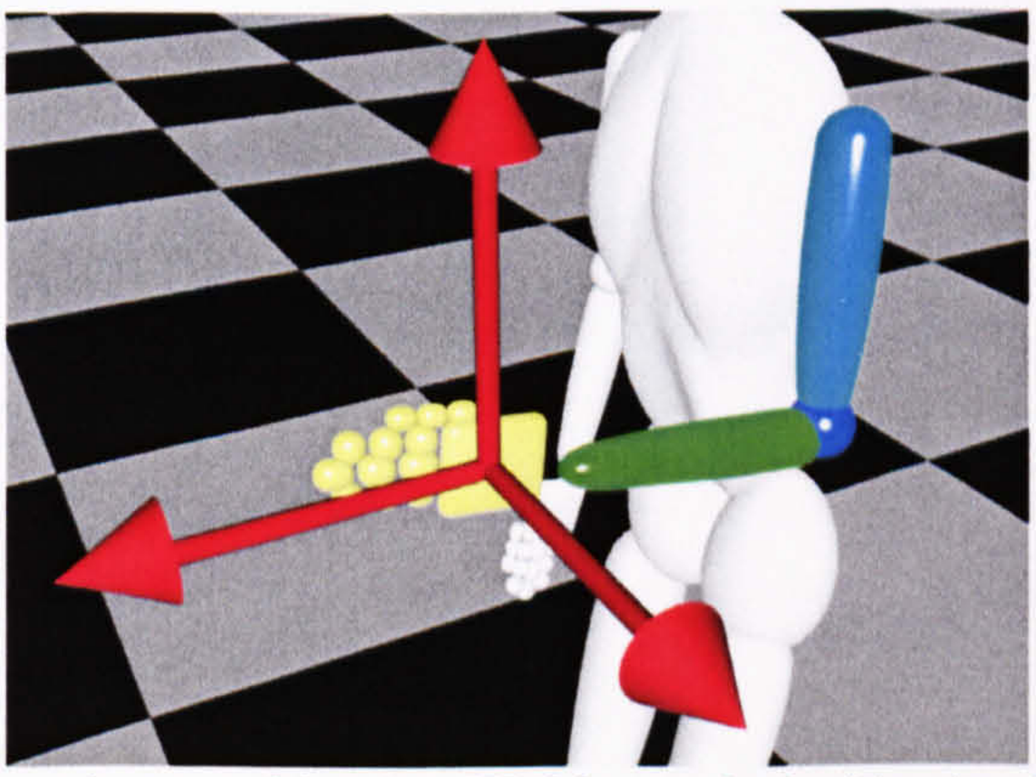
Future research might concentrate on further optimisation of the learning task (modification of the non-deterministic IK-based learning scheme seems to be a good starting candidate). It would also be interesting to explore other learning algorithms to find one which can learn the best action sequence in the shortest amount of time. Such an attempt has already been made (see Lach 2003) and the attempted technique was genetic programming (Koza, 1992). Also genetic algorithms could be used for this task. A comparison of two or more different learning approaches applied to the same task might then be attempted. A way of tackling the state explosion identified by the learning routines might also be to use neural networks (Haykin and Saher, 1999).

Another application of the presented system would be an on-line learning crowd system. The starting point would be a FreeWill system with very few actions present in the plan library. In the course of the simulation the agents could apply the learning framework to tackle a particular problem, such as using a lift. This would allow for greater flexibility compared to other systems as the application of the learning scheme allows to adjust the same action to the needs of a specific agent (position, body orientation, size of the

avatar etc.). Thus the characters would populate the action library with new actions learnt 'on demand'. It is expected that a varied and interesting scene would be generated with this approach. It might also be possible to reapply the learning scheme to the newly created high-level actions, thus creating composite actions and effectively building a hierarchical reinforcement learning system.

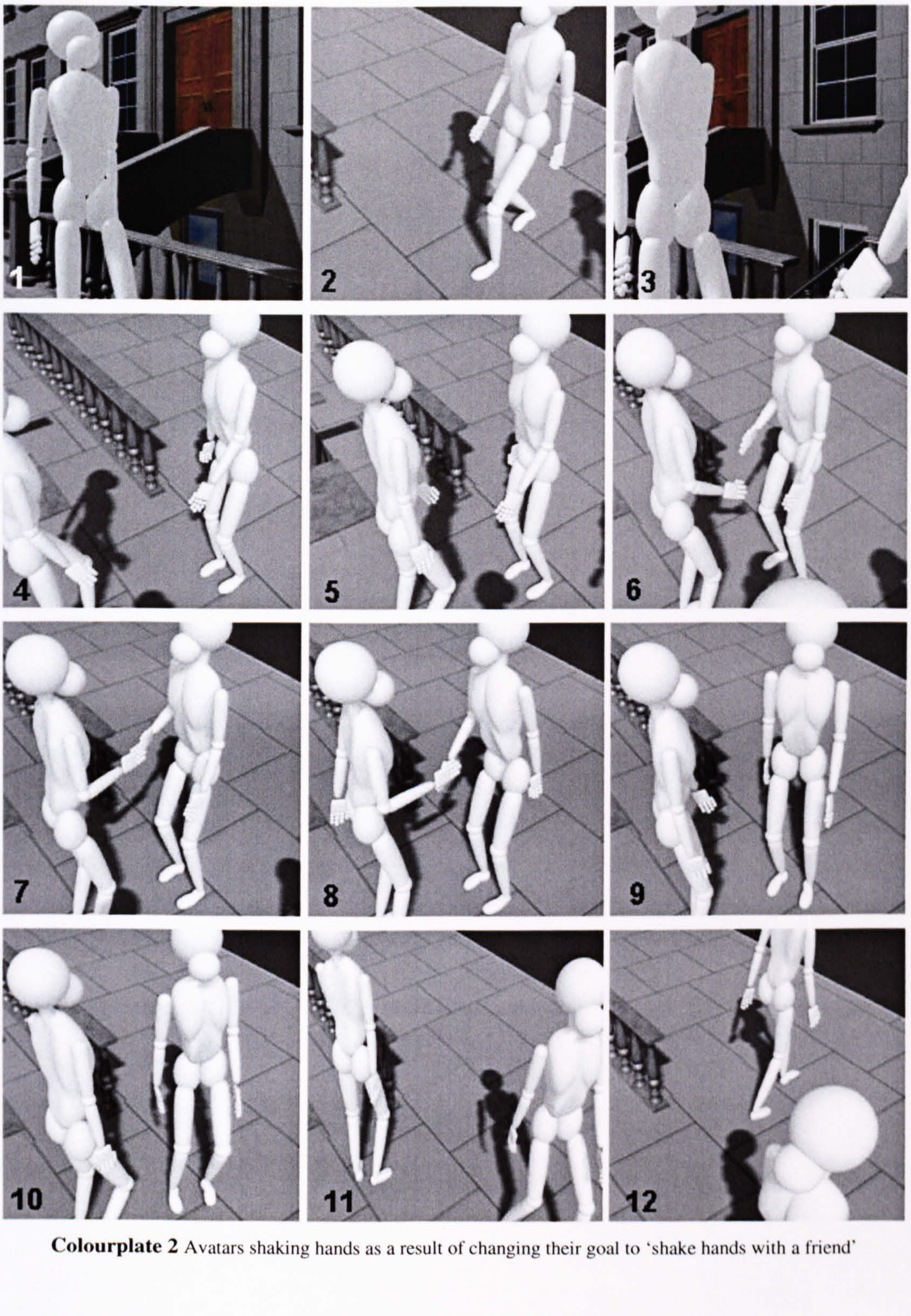


Avatar degrees of freedom FK (from top-left: Rotate arm up/down by $\Delta\alpha$, Rotate arm forward/backward by $\Delta\alpha$, Rotate forearm by $\Delta\alpha$, Rotate hand along Z axis by $\Delta\alpha$, Rotate shoulder along Z by $\Delta\alpha$)

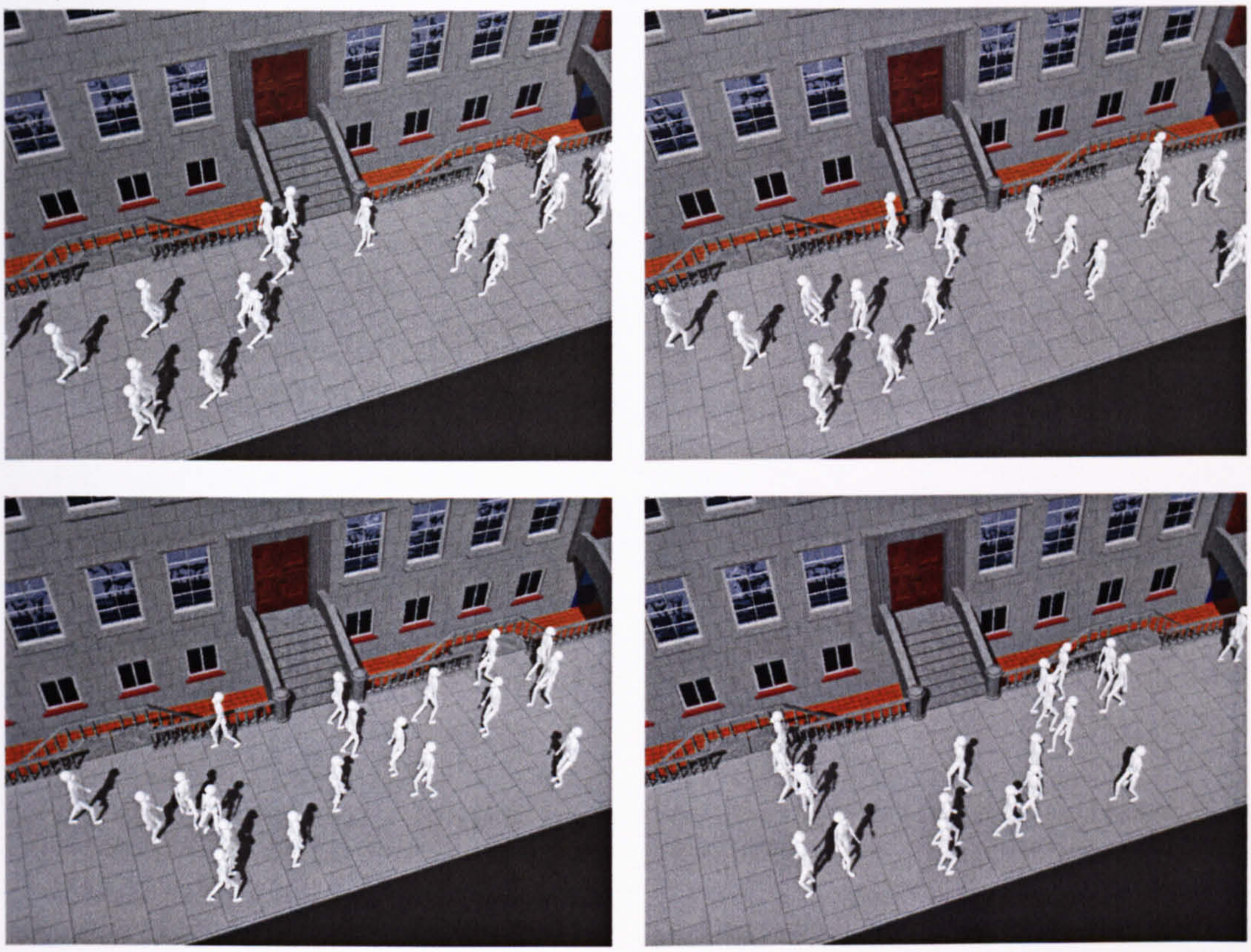


Avatar degrees of freedom IK (Move palm by $[\Delta x, \Delta y, \Delta z]$)

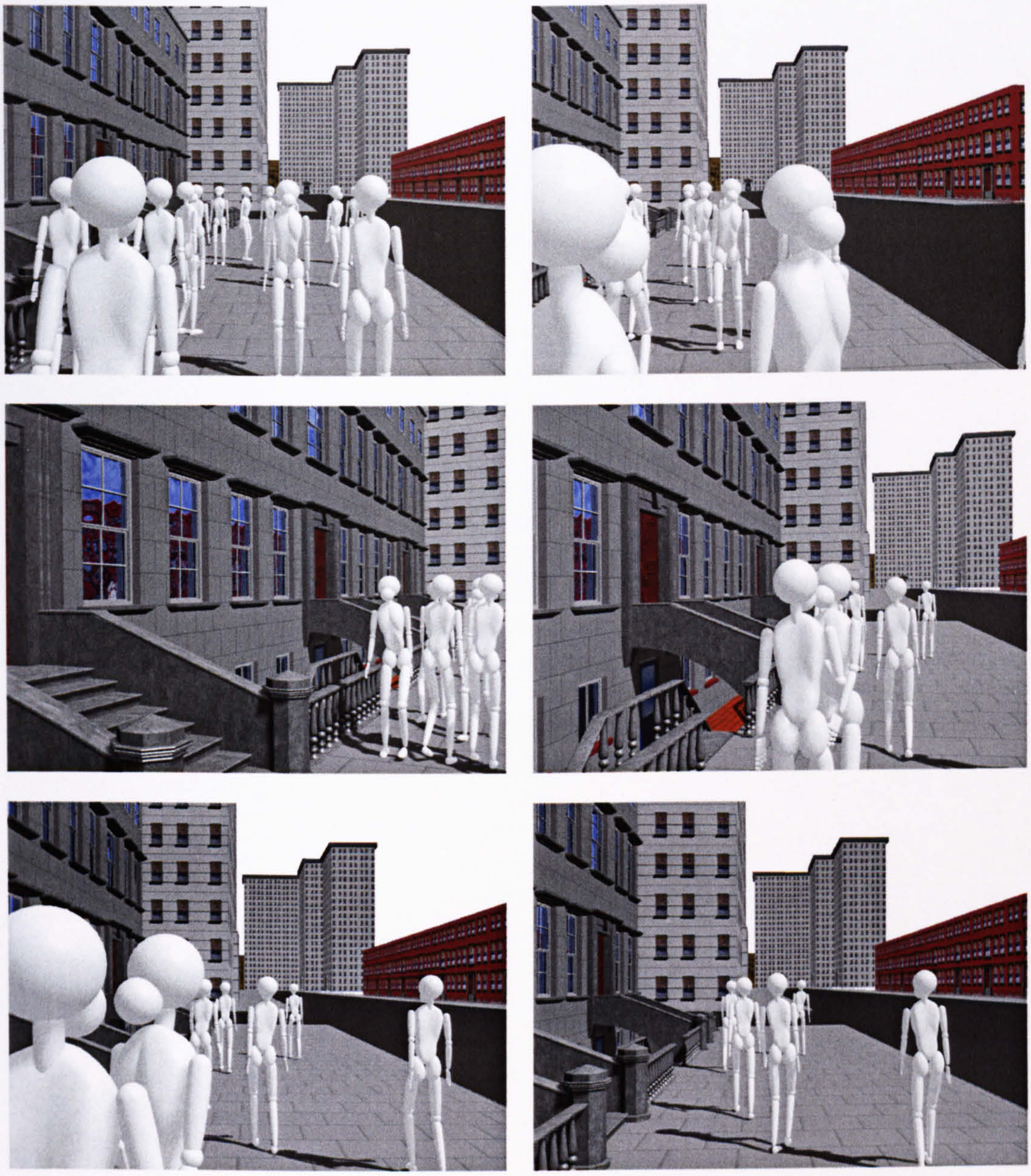
Colourplate 1 Avatar degrees of freedom



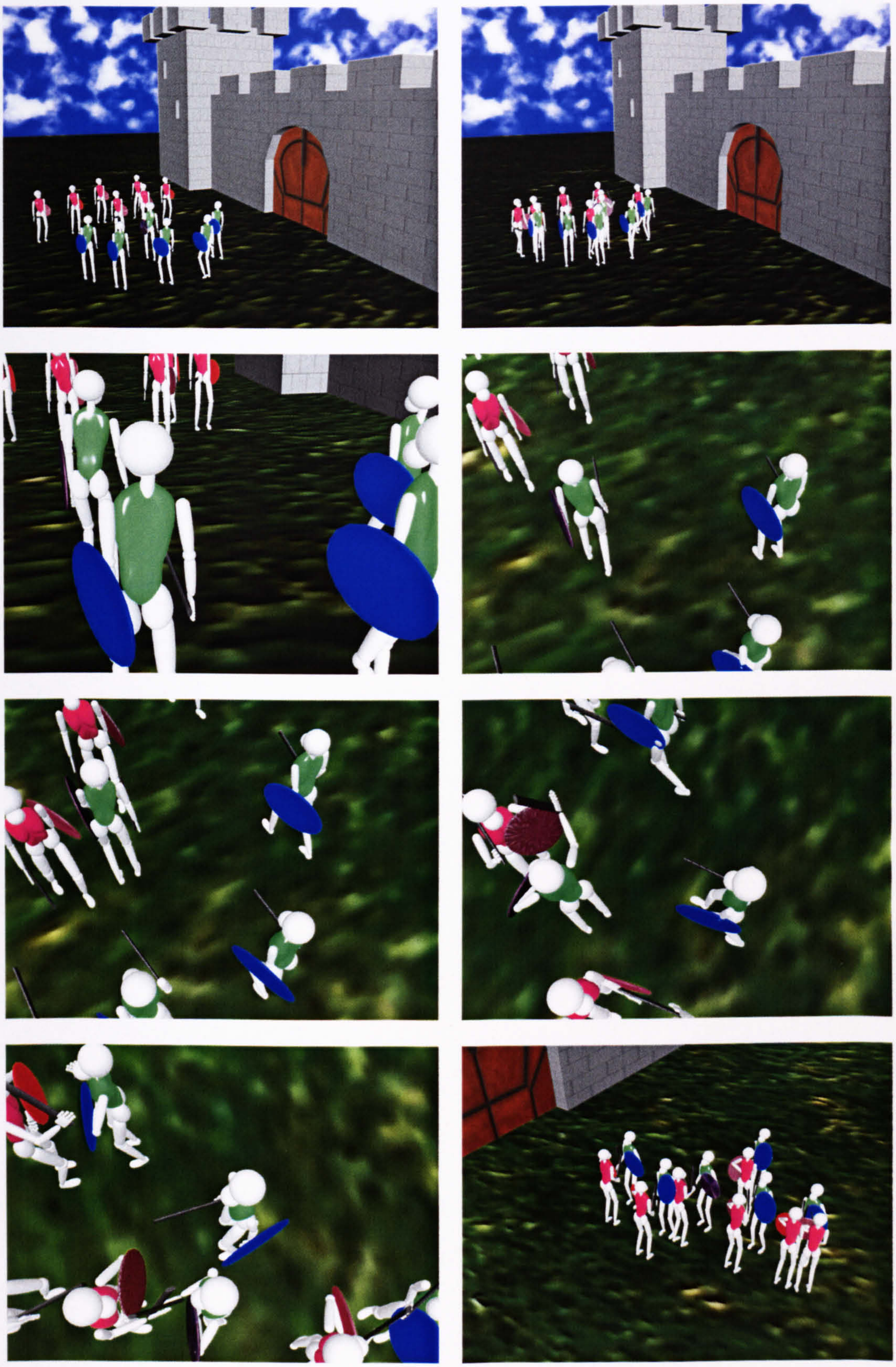
Colourplate 2 Avatars shaking hands as a result of changing their goal to 'shake hands with a friend'



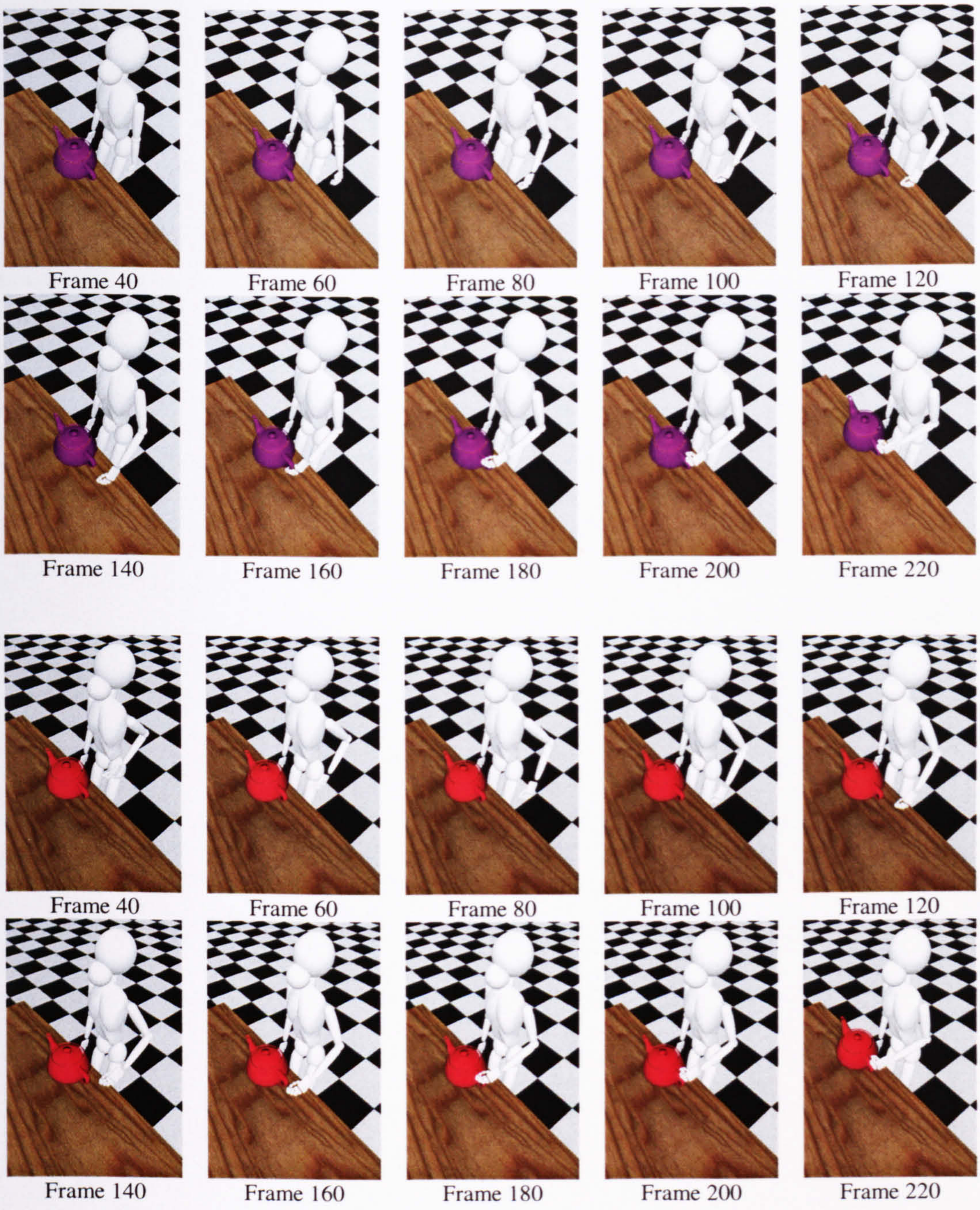
Colourplate 3 Avatars mingling on a sidewalk



Colourplate 4 Scene as seen by a single avatar



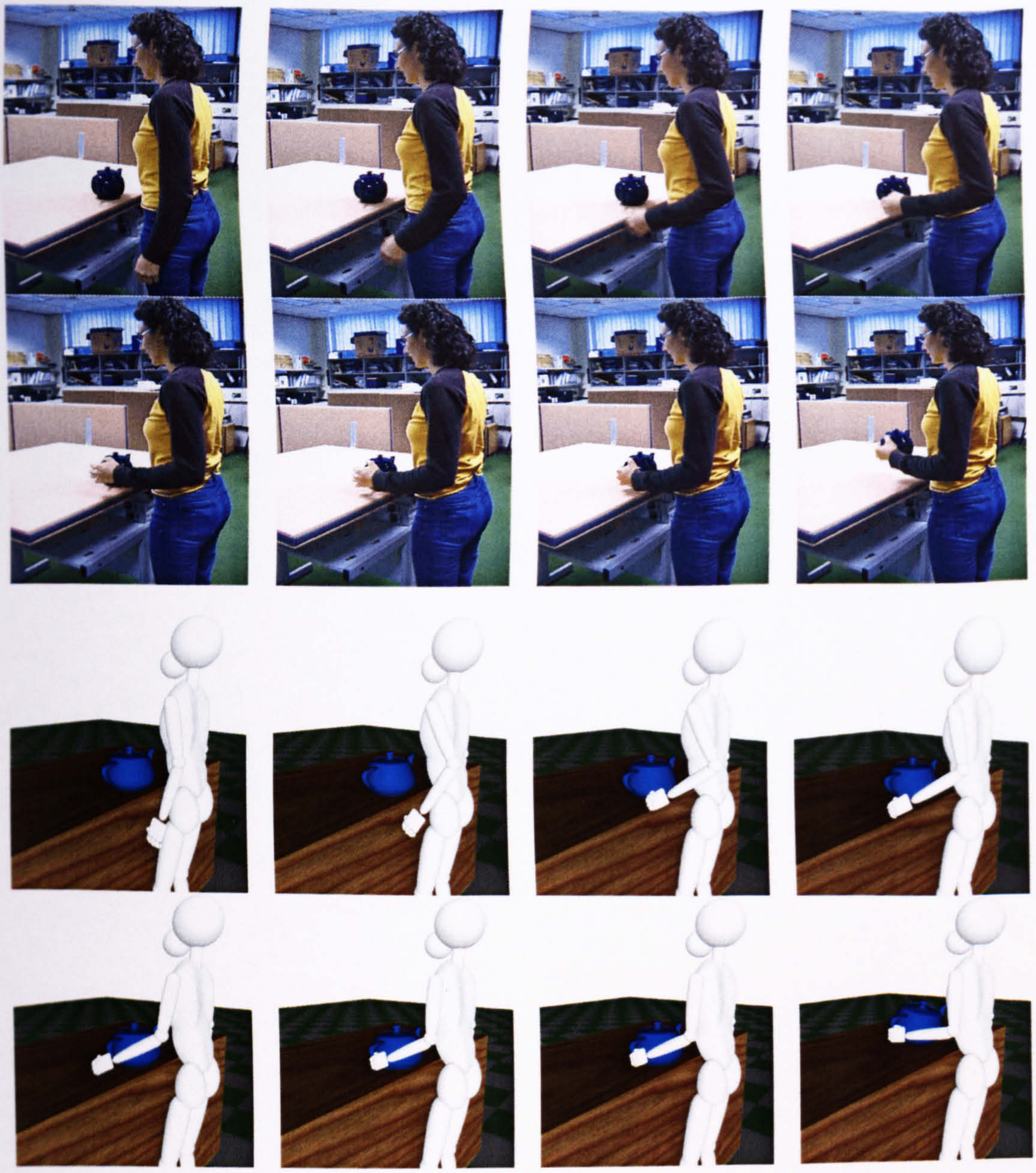
Colourplate 5 A battle scene simulated in FreeWill



Colourplate 6 Two different action sequences selected using the Local Distance Metric (IK set)

Colourplate 7 – FK controlled motion compared to motion of a human actor

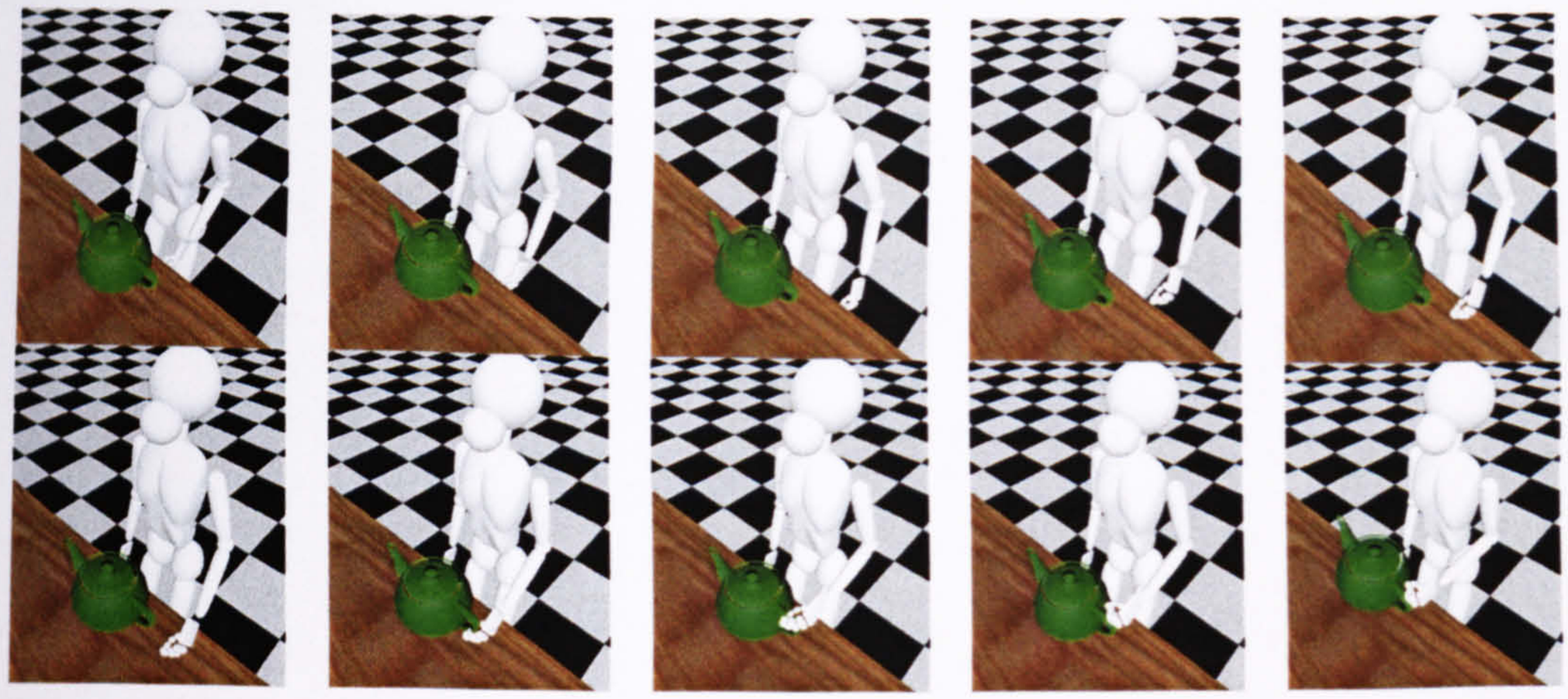
Colourplate 8 – FK controlled motion compared to motion of a human actor



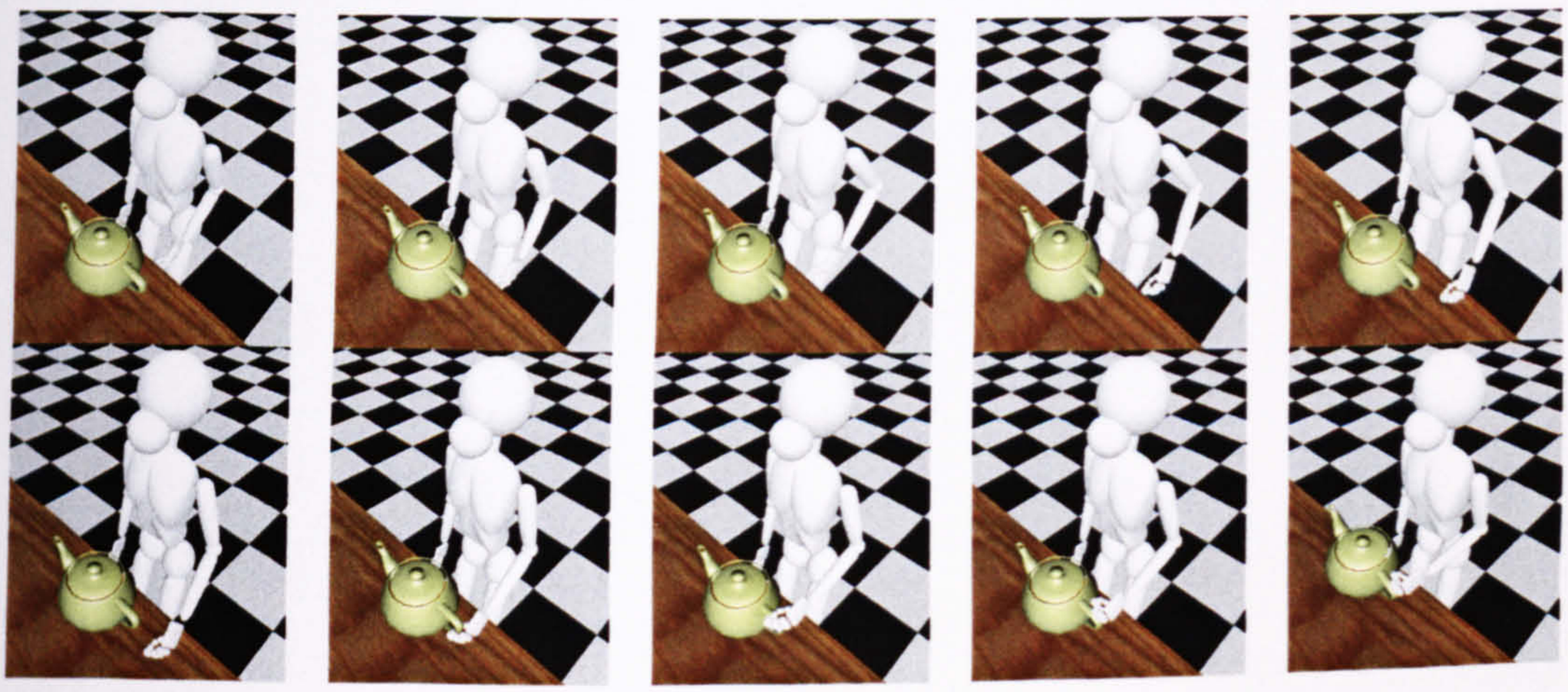
Colourplate 9 – IK controlled motion compared to motion of a human actor

Colourplate 10 – IK controlled motion compared to motion of a human actor

Colourplate 11 – FK and IK controlled motion compared to motion of a human actor



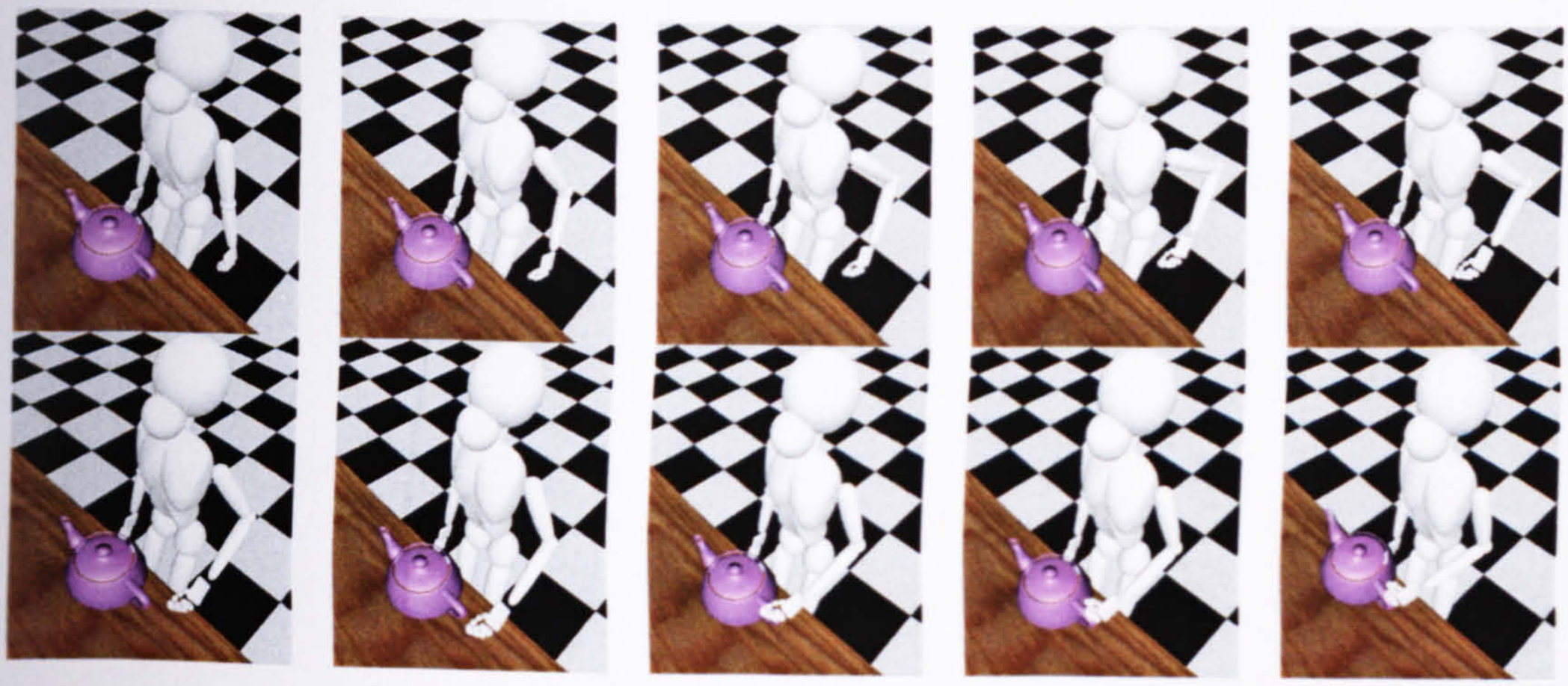
Sequence no 25 with maximum distance from average sequence



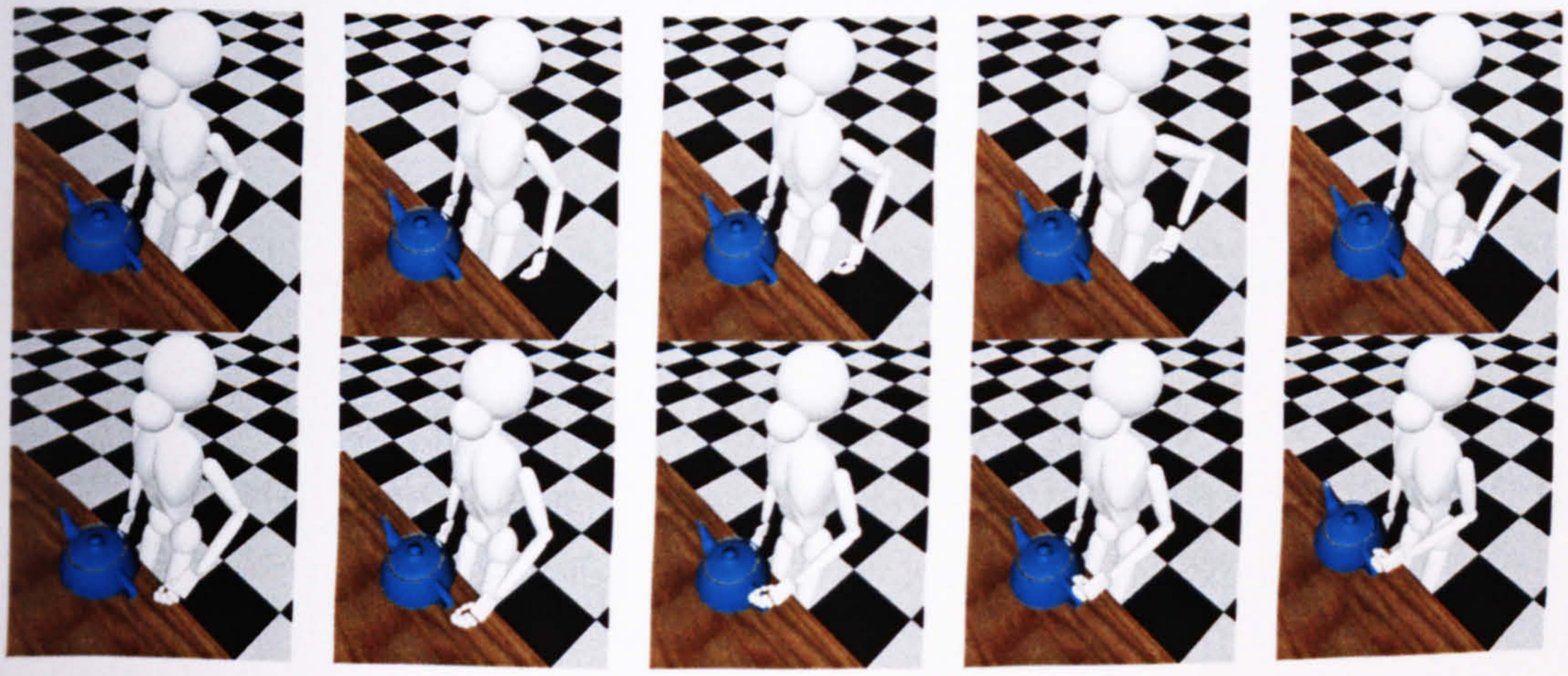
Sequence no 29 with maximum distance from average sequence

Frames presented are frames no.: 40 60 80 100 120 140 160 180 200 220

Colourplate 12 Sequences generated by the Global Distance Metric



Sequence no 8 with minimum distance from average sequence



Sequence no 41 with minimum distance from average sequence

Frames presented are frames no.: 40 60 80 100 120 140 160 180 200 220

Colourplate 13 Sequences generated by the Global Distance Metric

Colourplate 14 Avatar starting from a different position and achieving a goal using the same Q-table.

References

[1] AI.implant, http://www.ai-implant.com, accessed on 31 Oct 2003

[2] AI.implant Animation White Paper, accessed from http://www.ai-implant.com/pdf/AI.implant-Anim-White-Paper.pdf on 3 Nov 2003

[3] Alias website, www.aliaswavefront.com, accessed on 24 Oct. 2003

[4] Aljibury H., Arroyo A., Nechyba M., Using Locally Weighted Regression to Enhance Q-Learning in Proc. FCRAR 2002, Miami, May 2002

[5] Amant R. St. and Young M. R., Artificial Intelligence and Interactive Entertainment, Intelligence, Vol. 12, Issue 2, pp 17-19, Summer 2001

[6] Amkraut S., personal communication, January 8, 1987

[7] Amiguet Vercher J., Automatic crowd scene generation in SPIE Journal of Electronic Imaging, International technical group newsletter 11(1) December, pp. Front Page, 2000

[8] Amiguet Vercher J., Szarowicz A., Forte P., Synchronised multiagent simulations for automated crowd scene generation, Workshop on Spatial and Temporal Reasoning with Agents Focus - International Joint Conference on Artificial Intelligence IJCAI-01, August 4-10, Seattle, USA, 2001

[9] Anderson J., The Architecture of Cognition, Cambridge, MA, Harvard University Press, 1983

[10] Anderson J., The Adaptive Character of Thought, Hillsdale, NJ, Erlbaum Associates, 1990

[11] Anderson F. C. and Pandy M. G., Three-Dimensional Computer Simulation Of Gait, Bioengineering Conference Big Sky, Montana, June 16-20, 1999

[12] Anderson M., McDaniel E., Chenney S., Constrained animation of flocks, Proceedings of the ACM SIGGRAPH San Diego, California, USA, 2003

[13] Arafa Y., Kamyab B., Mamdani E., Kshirsagar S., Magnenat-Thalmann N., Guye-Vuilleme A., Thalmann D., Two Approaches to Scripting Character Animation, Proc. Workshop Embodied Conversational Agents - let's specify and evaluate them! at AAMAS 2002, Bologna, Italy, 2002

[14] Arikan O. and Forsyth D.A. , Interactive Motion Generation From Examples, Proceedings of the 2002 ACM SIGGRAPH, San Antonio, Texas, USA, 21-26 July 2002

[15] Arkin, R.C., Modeling Neural Function at the Schema Level: Implications and Results for Robotic Control", chapter in Biological Neural Networks in Invertebrate Neuroethology and Robotics, ed. R. Beer, R. Ritzmann, and T. McKenna, Academic Press, pp. 383-410, 1992

[16] Arkin, R. C., Behavior-Based Robotics, The MIT Press, 1998

[17] Arkin R. C., Carter W. M., MacKenzie D. C., Active Avoidance: Escape and Dodging Behaviors for Reactive Control. International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI), Volume 7 (1), World Scientific Publishing, pp.175-192, 1993

[18] AUML, Agent UML website, www.auml.org, accessed on 19 May 2003

[19] Bauer B., UML Class Diagrams: Revisited in the Context of Agent-Based Systems, in Proceedings of Agent-Oriented Software Engineering (AOSE) 2001, Agents 2001, Montreal, 2001

[20] Bauer B., Müller J. P., Odell J., Agent UML: A Formalism for Specifying Multiagent Software Systems, International Journal of Software Engineering and Knowledge Engineering, Vol. 11, No. 3, pp1-24, 2001

[21] Bergenti F. and Poggi A., Exploiting UML in the Design of Multi-Agent Systems, ESAW Workshop at ECAI 2000

[22] Bertsekas D.P. and Tsitsiklis J.N., Neuro-Dynamic Programming, Athena Scientific, 1996

[23] Blumberg B., Downie M., Ivanov Y., Berlin M., Johnson M. P., Tomlinson B., Integrated learning for interactive synthetic characters, ACM Transactions on Graphics, Vol. 21, Iss. 3, pp417-426, July 2002

[24] Booch G., Object-Oriented Design with Applications, Benjamin/Cummings, Redwood City, CA, 1991

[25] Booch G., Object-Oriented Analysis and Design with Applications (second edition). Addison-Wesley: Santa Clara, California, 1994

[26] Booch G., Rumbaugh J., Jacobson I., The Unified Modeling Language User Guide, Addison-Wesley, 1999

[27] Boone G., Efficient reinforcement learning: Modelbased acrobot control, Proceedings ICRA 1997

[28] Bregler C., Loeb L., Chuang E., Deshpande H., Turning to the Masters: Motion Capturing Cartoons, Proceedings of the 2002 ACM SIGGRAPH, San Antonio, Texas, USA, 21-26 July 2002

[29] Brogan, D.C. and Hodgins, J. K. Group Behaviors for Systems with Significant Dynamics, Autonomous Robots 4(1), pp. 137-153, 1997

[30] Burke R., Isla D., Downie M., Ivanov Y., Blumberg B., Creature Smarts: The Art and Architecture of a Virtual Brain, Proceedings of the Game Developers Conference 2000

[31] Burmeister B., Models And Methodologies For Agent-Oriented Analysis And Design, in Klaus Fischer, editor, Working Notes of the KI'96 Workshop on Agent-Oriented Programming and Distributed Systems. 1996

[32] Burmeister B., Haddadi A., Motylis G., Application of Multi-agent Systems in Traffic and Transportation, IEE Proceedings Software Engineering, Vol. 144, No. 1, February 1997

[33] Burmeister B., Bussmann S., Haddadi A., Sundermeyer K., Agent-Oriented Techniques for Traffic and Manufacturing Applications: Progress Report, in N. Jennings and M. Wooldridge, editors, Agent Technology: Foundations, Applications, and Markets. Springer-Verlag, 1998

[34] Card S., Moran T. Newell A., The Psychology of Human-Computer Interaction, Hillsdale, NJ, Erlbaum, 1983

[35] Cassell J., Vilhjálmsson H.H., Bickmore T., BEAT: The Behavior Expression Animation Toolkit, ACM SIGGRAPH 2001 Los Angeles, CA, USA, 12-17 August 2001

[36] Castro J., Kolp M., Mylopoulos J., A Requirements-Driven Development Methodology, in Proceedings of the 13th International Conference on Advanced Information Systems Engineering (CAiSE'01), Interlaken, Switzerland, pp. 108-123, June 4-8, 2001

[37] Cho B., Rosenbloom P. S., Dolan C. P. Neuro-SOAR: A neural network architecture for goal-oriented behavior, Proceedings of Thirteenth Annual Conference of the Cognitive Science Society, Lawrence Erlbaum Associates, Chicago, IL, 1991

[38] Christianini N. and Shawe-Taylor J., Support vector machines and other kernel-based learning methods. Cambridge University Press, 2000

[39] Coleman D., Arnold P., Bodoff S., Dollin C., Gilchrist H., Hayes F., and Jeremaes P., Object-Oriented Development: The FUSION Method, Prentice Hall International, 1994

[40] Collinot A., Drogoul A., Benhamou P., Agent oriented design of a soccer robot team, in Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS-96), Kyoto, Japan, 1996

[41] Comingsoon website, http://www.comingsoon.net, accessed on 26 January 2004

[42] Cover T.M. and Thomas J.A., Elements of Information Theory, Wiley Interscience, 1991

[43] Craig J., Introduction to Robotics: Mechanics and Control, Addison-Wesley, 1986

[44] Craig J., Blackboard Systems, Ablex, 1995

[45] Cremer, J., Kearney, J., and Papelis, Y., HCSM: Framework for Behavior and Scenario Control in Virtual Environments, ACM Transactions on Modeling and Computer Simulation, 5(3):242-267, 1995

[46] Dam K. H., Winikoff M., Comparing Agent-Oriented Methodologies, Proceedings of the Fifth International Bi-Conference Workshop on Agent-Oriented Information Systems, Melbourne, July 2003

[47] Davison D. E. and Bortoff S. A., Acrobot software and hardware guide, Technical Report Number 9406, Systems Control Group, University of Toronto, Toronto, Ontario M5S 1A4, Canada, June 1994

[48] Depke R., Heckel R., Küster J., Improving the Agent-Oriented Modeling Process by Roles, Proc. of Fifth International Conference of Autonomous Agents (Agents 2001), pp. 640-647, Montreal, 2001

[49] Devillers F. and Donikian S., A scenario language to orchestrate virtual world evolution, Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, San Diego, California, USA, pp265 - 275, 27-31 July, 2003

[50] Discreet website, www.discreet.com, accessed on 24 Oct 2003

[51] Dontcheva M., Yngve G., Popovic Z., Layered Acting for Character Animation, Proceedings of the ACM SIGGRAPH San Diego, California, USA, 2003

[52] Downie M., Behavior, Animation and Music: The Music and Movement of Synthetic Characters, M.Sc. Thesis, The Media Lab. MIT, 2000

[53] Downie M., Tomlinson B., Blumberg B., Developing an aesthetic: character-based interactive installations, Computer Graphics Vol. 36, Issue 2, May 2002

[54] Faloutsos P., Composable Controllers for Physics-Based Character Animation, Ph.D. Thesis, Department of Computer Science, University of Toronto, 2002

[55] Faloutsos P., van de Panne M., Terzopoulos D., Composable Controllers for Physics-Based Character Animation, in Proceedings of SIGGRAPH 2001, Los Angeles, USA, August 2001

[56] Faloutsos P., van de Panne M., Terzopoulos D., The virtual stuntman: dynamic characters with a repertoire of autonomous motor skills, Computers and Graphics, Volume 25, Issue 6, pp. 933-953, December, 2001

[57] Fang A. C. and Pollard N. S., Efficient Synthesis of Physically Valid Human Motion, ACM Transactions on Graphics 22(3) pp417-426, SIGGRAPH 2003 Proceedings, 2003

[58] Farenc N., Boulic R., Thalmann D., An Informed Environment Dedicated to the Simulation of Virtual Humans in Urban Context, Proc. Eurographics '99, Milano, Italy, pp.309-318, 1999

[59] Farenc N., Raupp Musse S., Schweiss E., Kallmann M., Aune O., Boulic R., Thalmann D., A Paradigm for Controlling Virtual Humans in Urban Environment Simulations, Applied Artificial Intelligence Journal, 1999

[60] FIPA, from: http://www.fipa.org, accessed on 12 Nov 2003

[61] Flake S., Geiger C., Küster J. M., Towards UML-based Analysis and Design of Multi-Agent Systems, in Proceedings of International NAISO Symposium on Information Science Innovations in Engineering of Natural and Artificial Intelligent Systems (ENAIS'2001), Dubai, March 2001

[62] Forte P., Szarowicz A., The Application of AI Techniques for Automatic Generation of Crowd Scenes, The Eleventh International Symposium on Intelligent Information Systems, Sopot, Poland, June 3-6, 2002

[63] Francik J., Data Flow Tracing for Algorithm Animation, PhD thesis (in Polish), Silesian University of Technology, Gliwice, Universite de Lille, 1999

[64] Francik J., A Framework for Program Control of Animation of Human and Animal Characters, Studia Informatica, Vol. 24, No. 4 (56), pp55-65, 2003

[65] Francik J. and Fabian P., Animating Sign Language in the Real Time. 20th IASTED International Multi-Conference Applied Informatics AI 2002, pp. 276-281, Innsbruck, Austria 2002

[66] Franklin S., Artificial Minds, The MIT Press, 1995

[67] Franklin S. and Graesser A., 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, Springer-Verlag, 1996

[68] Friedman-Hill E., Jess in Action Java Rule-based Systems, Manning Publications, 2003

[69] Funge J. D., Making Them Behave: Cognitive Models for Computer Animation, PhD thesis, Department of Computer Science, University of Toronto, 1998

[70] Funge J. D., AI for Games and Animation. A Cognitive Modeling Approach, A K Peters Natick, Massachusetts, 1999

[71] Funge J. D., Tu X., Terzopoulos D., Cognitive Modeling: Knowledge, reasoning and planning for intelligent characters, Computer Graphics Proceedings: SIGGRAPH 99, Aug 1999

[72] Funge J., Cognitive Modeling for Games and Animation, Communications of the ACM, Volume 43, Number 7, July 2000

[73] Gagne, R., The Conditions of Learning (4th ed.). New York: Holt, Rinehart & Winston, 1985

[74] Gambardella L.M. and Dorigo M., Ant-Q: A Reinforcement Learning approach to the travelling salesman problem, Proceedings of ML-95, Twelfth Intern. Conf. on Machine Learning, Morgan Kaufmann, pp.252–260, 1995

[75] Gamma E., Helm R., Johnson R., Vlissides J., Design patterns: elements of reusable object-oriented software, Addison-Wesley, 1995

[76] Georgeff M. P. and Rao A. S., A profile of the Australian AI Institute, IEEE Expert, 11(6), pp89–92, December 1996

[77] Georgeff M., Pell B., Pollack M., Tambe M., Wooldridge M., The Belief-Desire-Intention Model of Agency, Proceedings of the 5th International Workshop on Intelligent Agents V : Agent Theories, Architectures, and Languages ATAL, Paris, 1999

[78] Giarratano J. and Riley G., Expert Systems, PWS-KENT, 1989

[79] Giunchiglia F., Mylopoulos J., Perini A., The Tropos Software Development Methodology: Processes, Models and Diagrams, Technical Report No. 0111-20, ITC - IRST, Nov 2001

[80] Gleicher M., Making Motion Capture Useful, Siggraph 01 Courses, Los Angeles, USA, August 2001

[81] Gottschalk S., Lin M.C., Manocha D., OBB-Tree: a hierarchical structure for rapid interference detection, Proc. of ACM SIGGRAPH, 1996

[82] Gritz L. and Hahn J., Genetic Programming Evolution of Controllers for 3-D Character Animation, Proceedings of the Genetic Programming '97 Conference, July 1997

[83] Grzeszczuk R. and Terzopoulos D., Automated Learning of Muscle-Actuated Locomotion Through Control Abstraction, Proceedings of SIGGRAPH 95 ACM SIGGRAPH, pp. 63--70, 1995

[84] Grzeszczuk R., PhD Thesis, NeuroAnimator: Fast Neural Network Emulation and Control of Physics-Based Models, Dept. of Computer Science, University of Toronto, May 1998

[85] Grzeszczuk R., Terzopoulos D., Hinton G., NeuroAnimator: Fast Neural Network Emulation and Control of Physics-Based Models, proceedings of SIGGRAPH 98, Computer Graphics Proceedings, Annual Conference Series, pp. 9-20, Orlando, Florida, 1998

[86] Harland J. and Winikoff M., Agents via Mixed-mode Computation in Linear Logic: A Proposal, Proceedings of the ICLP'01 Workshop on Computational Logic in Multi-Agent Systems (CLIMA-01), Paphos, December, 2001

[87] Harmon M., Harmon S., Reinforcement learning: a tutorial, accessed from http://www.nbu.bg/cogs/events/2000/Readings/Petrov/rltutorial.pdf, on 08 Oct 2002, published December 1996

[88] Haykin S. S. and Saher S., Neural networks: a comprehensive foundation. - 2nd ed. - Upper Saddle River, N.J.: Prentice Hall, 1999

[89] Hayzelden A. L. G. and Bigham J., Heterogeneous Multi-Agent Architecture for ATM Virtual Path Network Resource Configuration, Proceedings of the Second International Workshop on Intelligent Agents for Telecommunication IATA'98, 1998

[90] Hodgins, J. K., Wooten, W. L., Brogan, D. C., O'Brien, J. F., Animating Human Athletics, Proceedings of Siggraph '95, In Computer Graphics, pp 71-78, 1995

[91] Hodgins, J. K. and Pollard, N. S., Adapting Simulated Behaviors For New Characters, SIGGRAPH 97, Los Angeles, CA, 1997

[92] Hodgins J. K., O'Brien J. F., Bodenheimer R. E., Computer Animation, In the Wiley Encyclopedia of Electrical and Electronics Engineering, John G. Webster, ed., v. 3, pp.686-690, 1999

[93] Horling B., Vincent R., Mailler R., Shen J., Becker R., Rawlins K., Lesser V., Distributed sensor network for real time tracking. In Proceedings of the fifth international conference on Autonomous agents, pages 417-424, ACM Press, 2001

[94] Iglesias C. A., Garijo M., Gonzalez J. C., A survey of agent-oriented methodologies, in Mueller J. P., Singh M. P., Rao A. S. (eds), Intelligent Agents V — Proceedings of the Fifth International Workshop on Agent Theories, Architectures, and Languages (ATAL-98), Lecture Notes in Artificial Intelligence. Springer-Verlag, Heidelberg, 1999

[95] Ishiguro H., Kanda T., Kimoto K., Ishida T., A Robot Architecture Based on Situated Modules, International Conference on Intelligent Robots and Systems, pp. 1617-1623, 1999

[96] Isla D., Burke R., Downie M., Blumberg B., A Layered Brain Architecture for Synthetic Creatures, pp. 1051-1058, in Proceedings of Seventeenth Joint Conference on Artificial Conference IJCAI-01, 4-10 August, Seattle, USA, 2001

[97] JACK, from: http://www.agent-software.com/shared/products/index.html, accessed on 12 Nov 2003

[98] Jacobson I., Christerson M., Jonsson P., Övergaard G., Object-Oriented Software Engineering. A Use Case Driven Approach, ACM Press/Addison-Wesley, 1992

[99] Jennings N. R., Agent-Oriented Software Engineering, in Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World: Multi-Agent System Engineering (MAAMAW-99), LNAI 1647, pp1-7, Valencia, Spain, June 30-July 2, 1999

[100] Jennings N. R., On agent-based software engineering, Artificial Intelligence, vol. 117, no 2, pp277-296, 2000

[101] Jennings N. R., An agent-based approach for building complex software systems, Communications of the ACM, 44(4), pp35-41, 2001

[102] Jennings N. R. and Wooldridge M. J., Applications of Intelligent Agents, in Jennings N. R. and Wooldridge M. J. (eds) Agent Technology: Foundations, Applications, and Markets, pp3-28, Springer-Verlag: Heidelberg, Germany, 1998

[103] Johnson W.L., Rickel J.W., Lester J.C., Animated Pedagogical Agents: Face-to-Face Interaction in Interactive Learning Environments, International Journal of Artificial Intelligence in Education 11, pp47-78, 2000

[104] Kaelbling L.P., Littman M.L., Moore A.W., Reinforcement learning: A survey, Journal of Artificial Intelligence Research, vol.4 pp.237-285, 1996

[105] Kallmann M., Object Interaction in Real-Time Virtual Environments, PhD thesis, École Polytechnique Fédérale de Lausanne, 2001

[106] Kaminka G. A., Veloso M. M., Schaffer S., Sollitto C., Adobbati R., Marshall A. N., Scholer A., Tejada S., GameBots: a flexible test bed for multiagent team research, Communications of the ACM, Vol. 45 , Issue 1, pp43-45, January 2002

[107] Kearsley G., Explorations in Learning & Instruction: The Theory Into Practice Database, accessed from: http://tip.psychology.org/ on 20 Mar 2003

[108] Kieras D.E., Towards a practical GOMS model methodology for user interface design, in M. Helander (Ed.), Handbook of Human-Computer Interaction, Amsterdam, Elsevier/North Holland, 1988

[109] Kinetic Impulse website, http://www.kinetic-impulse.com, accessed on 16 Oct 2003

[110] Kinny D., Georgeff M., Rao A., A Methodology and Modelling Technique for Systems of BDI Agents, in Van de Velde W. and Perram J. W. (eds), Agents Breaking Away: Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World, (LNAI Volume 1038), pp56–71. Springer-Verlag: Berlin, Germany, 1996

[111] Kline C. and Blumberg B., The Art and Science of Synthetic Character Design, Proceedings of the AISB 1999 Symposium on AI and Creativity in Entertainment and Visual Art, Edinburgh, Scotland, 1999

[112] Kline C. and Blumberg B., Observation-based expectation generation and response for believable reactive agents, Proceedings of the Fourth International Conference on Autonomous Agents, ACM Press New York, pp.46-47, Barcelona, Spain, 2000

[113] Koeppel D., Massive Attack, Popular Science 2002, accessed from http://www.popsci.com/popsci/science/article/0,12543,390918-1,00.html, on 11 Jan 2003

[114] Kovar L. and Gleicher M., Flexible automatic motion blending with registration curves, Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, San Diego, California, USA, pp214 - 224, 27-31 July, 2003

[115] Koza J.R., On the Programming of Computers by Means of Natural Selection Artificial System, MIT Press, 1992

[116] Kumar D., A Hybrid Connectionist and BDI Architecture for Modeling Embedded Rational Agents (1998), Cognitive Robotics: Papers from the 1998 AAAI Fall Symposium, Technical Report FS-98-02, 1998

[117] Lach E., Genetic Programming in the Animation of Human Avatars, 3rd Int. PhD Students' Workshop on Control and Information Technology IWCIT'03, Gliwice, Poland, 2003

[118] Laird J. E., Newell A., Rosenbloom P. S., Soar: An architecture for general intelligence, Artificial Intelligence, vol. 33 (1), pp1-64, 1987

[119] Laird, J. E. and Jones R. M., Building advanced autonomous AI systems for large scale real time simulations. Computer Games Development Conference. Long Beach, CA, 1998

[120] Laszlo J., van de Panne M., Fiume E., Limit Cycle Control and its Application to the Animation of Balancing and Walking, Proceedings of SIGGRAPH 1996, (New Orleans, LA, August 4-9, 1996), in Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, pp.155-162, 1996

[121] Laurent G. and Piat E., Parallel Q-Learning for a block-pushing problem, Proceedings of the International Conference on Intelligent Robots and Systems, IROS 2001, pp286-291, Maui, Hawaii, USA, 29 Oct-3 Nov 2001

[122] Lee J. W., Baek N., Kim D., Hahn J. K., A Procedural Approach to Solving Constraints of Articulated Bodies, Eurographics 2000 Short Presentations Programme, Interlaken, Switzerland, August 20-25, 2000

[123] Lee J., Chai J., Reitsma P. S. A., Hodgins J.K., Pollard N.S., Interactive Contol of Avatars Animated With Human Motion Data, Proceedings of the 2002 ACM SIGGRAPH, San Antonio, Texas, USA, 21-26 July 2002

[124] Levesque H., Reiter R., Lesperance Y., Lin F., Scherl R.. Golog: A logic programming language for dynamic domains, Journal of Logic Programming, Special issue on Reasoning about Action and Change, 31:59–84, 1997

[125] Li Y., Wang T., Shum H.-Y., Motion Textures: A Two-Level Statistical Model for Character Motion Synthesis, Proceedings of the 2002 ACM SIGGRAPH, San Antonio, Texas, USA, 21-26 July 2002

[126] Liu K.C. and Popovic Z., Synthesis of Complex Dynamic Character Motion From Simple Animations, Proceedings of the 2002 ACM SIGGRAPH, San Antonio, Texas, USA, 21-26 July 2002

[127] Lord of the Rings website, http://www.lordoftherings.net, accessed from http://www.lordoftherings.net/effects/index.html on 18 Sep 2003

[128] Loscos C., Tecchia F., Chrysanthou Y., Real-time shadows for animated crowds in virtual cities, Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST) '01, pp85-92, Banff, Alberta, Canada, November 2001

[129] Loscos C., Tecchia F., Chrysanthou Y., Real Time Shadows for Animated Crowds in Virtual Cities, project website, accessed from: http://www.cs.ucl.ac.uk/research/vr/Projects/Crowds/VRST01, on 03 Jul 2003

[130] Mac Namee B. and Cunningham P., Creating Socially Interactive Non Player Characters: The μ-SIC System, International Journal of Intelligent Games and Simulation, Vol. 2 No. 1, February 2003

[131] Maestri J., Digital Character Animation, New Riders Publishing, 1999

[132] Mayfield J., Labrou Y., Finin T. Evaluation of KQML as an Agent Communication Language, Proceedings on the IJCAI Workshop on Intelligent Agents II: Agent Theories, Architectures, and Languages, vol. 1037, Springer-Verlag, pp. 347-360, 1996

[133] Metoyer, R. A., Hodgins, J. K., Animating Athletic Motion Planning By Example. Proceedings of Graphics Interface 2000, pp. 61-68, Montreal, Quebec, Canada, May 15-17, 2000

[134] Mitchell T.M., Machine Learning, McGraw Hill, 1997

[135] Mondal A. S., Jain A. K., A Multi-Agent System For Sales Order Processing, Intelligence, Vol. 12 , Issue 3 pp 32-42, Fall 2001

[136] Monekosso N.D. and Remagnino P., Q-Learning augmented with synthetic pheromones, Proceeding of 2nd International Workshop of Central and Eastern Europe on Multi-Agent Systems, CEEMAS'01, Cracow, Poland, Lecture Notes in Artificial Intelligence Springer Verlag, 26-29 September 2001

[137] Monekosso N.D., Remagnino P., Szarowicz A., An Improved Q-Learning Algorithm Using Synthetic Pheromones in From Theory to Practice in Multi-Agent Systems, Lecture Notes in Computer Science, vol. 2296 Edited by Dunin-Keplicz, B. and Nawarecki, E., Springer-Verlag, pp. 197, March, 2002

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

[139] Monzani J-S., Caicedo A., Thalmann D., Integrating Behavioural Animation Techniques, Proc. Eurographics 2001, vol. 20, issue 3, Manchester, UK, 2001

[140] Mylopoulos J., Kolp M., Castro J., UML for Agent-Oriented Software Development: The Tropos Proposal, in Proceedings of the Fourth International Conference on the Unified Modeling Language, Toronto, Canada, October 2001

[141] Neff M., Fiume E., Aesthetic edits for character animation, Proceeding of the 2003 ACM SIGGRAPT/Eurographics Symposium on Computer Animation, pp239-244, San Diego, California, USA, 2003

[142] Newell A., Unified Theories of Cognition, Harvard Press, 1990

[143] Ng A. Y., Harada D., Russell S., Policy invariance under reward transformations: Theory and application to reward shaping, Proceedings ICML-99, Bled, Slovenia, 1999

[144] Nwana H. S. and Ndumu D. T., A Brief Introduction to Software Agent Technology, in Jennings N. R. and Wooldridge M. J. (eds) Agent Technology: Foundations, Applications, and Markets, pp29-47, Springer-Verlag: Heidelberg, Germany, 1998

[145] Object Management Group, OMG Unified Modeling Language Specification v.1.5, January 2003, See http://www.omg.org or http://www.uml.org

[146] Odell J., Van Dyke Parunak H., Bauer B., Extending UML for Agents, in Proceedings of the Agent-Oriented Information Systems Workshop at the 17th National Conference on Artificial Intelligence, pp3-17, AOIS Worshop at AAAI 2000

[147] Odell J., Van Dyke Parunak H., Bauer B., Representing Agent Interaction Protocols in UML, in proc. ICSE Workshop on Agent-oriented Software Engineering, Limerick, Ireland, 2000

[148] Odell J., Van Dyke Parunak H., Bauer B., Representing Agent Interaction Protocols in UML, Agent-Oriented Software Engineering, Paolo Ciancarini and Michael Wooldridge eds., Springer, Berlin, pp. 121-140, 2001

[149] Oray W.D., John B., Atwood M., Project Ernestine: Validating GOMS for predicting and explaining real-world task performance, Human Computer Interaction, 8, pp112-120. 1993

[150] O'Sullivan C., Cassell J., Vilhjálmsson H., Dingliana J., Dobbyn S., McNamee B., Peters C., Giang T., Levels of Detail for Crowds and Groups, Computer Graphics Forum, Vol. 21(4) pp 733-742, November 2002

[151] Padgham L., and Winikoff M., Prometheus: A Methodology for Developing Intelligent Agents, Proceedings of the Third International Workshop on Agent-Oriented Software Engineering, at AAMAS'02, Bologna, Italy, 2002

[152] Pandy M. G. and Anderson F. C., Three-Dimensional Computer Simulation Of Jumping and Walking Using the Same Model, in Proceedings of the VIIth International Symposium on Computer Simulation in Biomechanics, August 1999

[153] Pollard, N. S. and Hodgins, J. K.. Adapting Behaviors to New Environments, Characters, and Tasks . Yale Workshop on Adaptive and Learning Systems, 1998

[154] Pottinger D. C. and Laird J. E., Game AI: The State of the Industry, Part Two, Game Developer, August, 2000

[155] Pullen K. and Bregler C., Motion Capture assisted Animation: Texturing and Synthesis, Proceedings of the 2002 ACM SIGGRAPH, San Antonio, Texas, USA, 21-26 July 2002

[156] Rao A., AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language, Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Eindhoven, The Netherlands, 1996

[157] Rao A. S. and Georgeff M. P., Modeling Rational Agents within a BDI-Architecture, Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning pp 473--484, Cambridge, MA, USA, April 1991. Morgan Kaufmann Publishers, 1991

[158] Rao A. S. and Georgeff M. P., An Abstract Architecture for Rational Agents' in B. Nebel, C. Rich, W. Swartout (eds.): Proc. International Conference on Principles of Knowledge Representation and Reasoning (KR-92), Morgan Kaufmann, San Mateo, 1992

[159] Rao A. S. and Georgeff M. P., Intentions and Rational Commitment, 1993, revised, from: http://citeseer.nj.nec.com/rao93intentions.html, accessed on 11 Feb 2002

[160] Rao A. S. and Georgeff M. O., BDI Agents: From Theory to Practice, in Proceedings of the First International Conference on Multiagent Systems ICMAS95, June 12 - 14, 1995

[161] Raphael M. J. and DeLoach S. A., A Knowledge Base for Knowledge-Based Multiagent System Construction, Proceedings of the National Aerospace and Electronics Conference (NAECON), Dayton, OH, October, 2000

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

[163] Reeves W. T., Particle Systems – A Technique for Modeling a Class of Fuzzy Objects, ACM Transactions on Graphics, Vol 2 No. 2, April 1983

[164] Reynolds, C. W., Flocks, herds, and schools: A distributed behavioral model, Computer Graphics, SIGGRAPH '87 Conference Proceedings, vol. 21(4) pp25-34, ACM SIGGRAPH 1987

[165] Reynolds, C. W., Steering Behaviors For Autonomous Characters, in the proceedings of Game Developers Conference 1999 held in San Jose, California. Miller Freeman Game Group, San Francisco, California, pp763-782, 1999

[166] Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorenson W., Object-Oriented Modelling and Design, Prentice Hall, Englewood Cliffs, 1991.

[167] Russell S. and Norvig P., Artificial Intelligence: A Modern Approach, Englewood Cliffs, NJ: Prentice Hall, 1995

[168] RUP website http://www.rational.com/products/rup, accessed on 11 Sep 2003

[169] Russell K.B. and Blumberg B., Behavior-Friendly Graphics, in Computer Graphics International, pp44, 1999

[170] Schaal S. and Atkeson C., Robot juggling: An implementation of memory-based learning. Control Systems Magazine, 14, 1994

[171] Schraudolph N. N., Dayan P., Sejnowski T. J, Temporal difference learning of position evaluation in the game of Go. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, Advances in Neural Information Processing Systems 6, pp. 817-824, Morgan Kaufmann, San Mateo, CA, 1994

[172] Shim, Y.-S. and Kim C.-H., Generating flying creatures using body-brain co-evolution, Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, San Diego, California, USA, pp276 - 285, 27-31 July, 2003

[173] Shoham Y.. Agent-oriented programming, Artificial Intelligence, 60(1), pp51–92, 1993

[174] Sims K., Evolving 3D Morphology and Behavior by Competition, Artificial Life IV Proceedings, ed.by Brooks & Maes, pp.28-39, MIT Press, 1994

[175] Sims K., Evolving Virtual Creatures, Computer Graphics (Siggraph '94 Proceedings), pp.15-22, July 1994

[176] Softimage news website, www.softimage.com/home/press/pressreleases, accessed on 15 Sep 2003

[177] Sommerville I. Software engineering, Addison-Wesley, 6th ed 2001

[178] Spielberg-dreamworks website, www.spielberg-dreamworks.com, accessed from www.spielberg-dreamworks.com/gladiator/Image_Gallery_Two.htm on 05 May 2003

[179] Spronck P., Sprinkhuizen-Kuyper I., Postma E., Improving Opponent Intelligence Through Offline Evolutionary Learning, International Journal of Intelligent Games and Simulation, Vol. 2 No. 1, pp20-27, February 2003

[180] Still G. K., Crowd Dynamics, PhD thesis, Warwick University, 2000

[181] Sutton, R. S., Generalization in reinforcement learning: Successful examples using sparse coarse coding, In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, Advances in Neural Information Processing Systems: Proceedings of the 1995 Conference, pp.1038-1044, Cambridge, MA, MIT Press, 1996

[182] Sutton R.S. and Barto A.G., Reinforcement Learning: an introduction, MIT Press, 1998

[183] Szarowicz A., Reinforcement Learning Techniques for Action Generation Using Inverse Kinematics, Postgraduate Research Conference PREP 2004, 5-7 April, Hatfield, United Kingdom, 2004

[184] Szarowicz A., Amiguet-Vercher J., Forte P., Briggs J., Gelepithis P., Remagnino P., The Application of AI to Automatically Generated Animation, Australian Joint Conference on Artificial Intelligence, AI'01, Adelaide, Dec 10-14, 2001

[185] Szarowicz A., Amiguet Vercher J., Forte P., Multiagent interaction for crowd scene generation, Workshop on Autonomy Delegation and Control Interacting with Autonomous Agents - International Joint Conference on Artificial Intelligence IJCAI-01, August 4-10, Seattle, USA, 2001

[186] Szarowicz A., Forte P., Amiguet-Vercher J., Gelepithis P., Application of Autonomous Agents for Crowd Scene Simulation, 2nd Hellenic Conference on Artificial Intelligence, SETN-02, Thessaloniki, Greece, April 11-12, 2002

[187] Szarowicz A., Forte P., Combining Intelligent Agents and Animation, AIxIA 2003 - Eighth National Congress on AI, Lecture Notes in Artificial Intelligence, vol 2829 Springer-Verlag, September 22-26, Pisa, Italy, 2003

[188] Szarowicz A., Mittmann M., Remagnino P., Francik J., Automatic Acquisition of Actions for Animated Agents, 4th Annual European GAME-ON Conference, November 19-21, London, United Kingdom, 2003

[189] Szarowicz A., Mittmann M., Francik J., Chapter Intelligent Action Acquisition for Animated Learning Agents in Learning Coordination and Communication in MultiAgent Systems, Theory and Applications Edited by Lakhmi C. Jain, World Scientific, to appear, 2004

[190] Szarowicz A., Remagnino P., Avatars That Learn How to behave, European Conference on Artificial Intelligence ECAI 2004, Springer, Valencia, Spain, 2004

[191] T-Tool, from: http://sra.itc.it/tools/t-tool, accessed on 9 Sep 2003

[192] Tecchia F., Loscos C., Conroy R., Chrysanthou Y., Agent Behaviour Simulator (ABS): A Platform for Urban Behaviour Development, presented at the ACM/EG Games Technology Conference, January 2001

[193] Tecchia F., Loscos C., Chrysanthou Y., Image-Based Crowd Rendering, IEEE Computer Graphics and Applications, vol. 22, number 2, March-April 2002

[194] Tecchia F., Loscos C., Chrysanthou Y., Visualizing Crowds in Real-Time, Computer Graphics forum, vol. 21, Number 4, pp 753-765, December 2002

[195] Tedrake R. and Seung H. S., Improved Dynamic Stability using Reinforcement Learning, Proceedings of the International Conference on Climbing and Walking Robots (CLAWAR'02), 2002

[196] Terzopoulos D., Tu X., Grzeszczuk R., Artificial Fishes: Autonomous Locomotion, Perception, Behavior, and Learning in a Simulated Physical World, Artificial Life, 1(4):327--351, 1994

[197] Terzopoulos D., Rabie T., Grzeszczuk R., Perception and Learning in Artificial Animals, Artificial Life V: Proc. 5th Inter. Conf. on the Synthesis and Simulation of Living Systems, Nara, Japan, 1996

[198] Tesauro G., TD-Gammon, a self-teaching backgammon program achieves master-level play. Neural Computation, 6(2) pp.215-219, 1994

[199] Thrun S., Learning to play the game of chess. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, Advances in Neural Information Processing Systems 7, MIT Press Cambridge, MA, 1995

[200] Tomlinson B., Blumberg B., Nain D., Expressive autonomous cinematography for interactive virtual environments, Proceedings of the Fourth International Conference on Autonomous Agents, Barcelona, Spain, pp.317–324, 2000

[201] Tomlinson B., Downie M., Berlin M., Gray J., Lyons D., Cochran J., Blumberg B., Leashing the AlphaWolves: mixing user direction with autonomous emotion in a pack of semi-autonomous virtual characters, Proceedings of the ACM SIGGRAPH symposium on Computer Animation, San Antonio, Texas, 2002

[202] Touzet C. F., Neural Networks and Q-Learning for Robotics, IJCNN '99 Tutorial, 1999 International Joint Conference on Neural Networks, Washington, DC - July 10-16, 1999

[203] Ulicny B. and Thalmann D., Crowd simulation for interactive virtual environments and VRtraining systems, Proc. Eurographics Workshop on Animation and Simulation, pp. 163-170, Springer-Verlag, 2001

[204] van de Panne M., Laszlo J., Huang P., Faloutsos P., Dynamic Human Simulation: Towards Agile Animated Characters, Proceedings of the IEEE International Conference on Robotics and Automation 2000, pp. 682-687, San Francisco, CA, 2000

[205] van Dyke Parunak H. and Odell J., Represening Social Structures in UML, From Proc. of Agent-Oriented Software Engineering (AOSE) 2001, pp. 17-31, Agents 2001, Montreal, 2001

[206] van Lent, M., Laird, J., Buckman, J., Hartford, J., Houchard, S., Steinkraus, K., Tedrake, R. (1999) Intelligent Agents in Computer Games, in Proceedings of the National Conference on Artificial Intelligence, Orlando, July 1999

[207] van Waveren J. and Rothkrantz L., Artificial Player for Quake III Arena, International Journal of Intelligent Games & Simulation, Vol. 1 No. 1, pp. 25-32, March 2002

[208] Vicon 8 Motion Capture System – Demonstration CD, presented at 3December Exhibition, London, 2001

[209] Vicon website, http://www.vicon.com/entertainment/applications/film.shtml on 09 May 2003

[210] Wagner G., A UML Profile for External AOR Models, Proc. of Agent-Oriented Software Engineering (AOSE), Workshop at AAMAS 2002, pp.99-110, Bologna, Italy, 2002

[211] Wan T. R. and Tang W., Simulating Virtual Character's Learning Behaviour as An Evolutionary Process Using Genetic Algorithms, Journal of WSCG, Volume 10, Number 3, 2002

[212] Watkins, C.J.C.H., Learning from delayed rewards, PhD thesis, University of Cambridge, Psychology Department, 1989

[213] Watkins, C.J.C.H. and Dayan, P., Technical Note: Q-Learning. Machine Learning 8: 279-292, 1992

[214] Winikoff M., RMIT Department of Computer Science Seminar, Simplifying Agent Concepts - presentation, 5 June 2001

[215] Winikoff M., AgentTalk, from: http://goanna.cs.rmit.edu.au/~winikoff/agenttalk/, accessed on 10 Nov 2003

[216] Winikoff M., Padgham L., and Harland J., Simplifying the Development of Intelligent Agents, In AI2001: Advances in Artificial Intelligence. 14th Australian Joint Conference on Artificial Intelligence, LNAI 2256, pages 557-568, Adelaide, December 2001

[217] Wirfs-Brock R., Wilkerson B., Wiener L., Designing Object-Oriented Software, Prentice-Hall, Englewood Cliffs, 1990

[218] Wood M. and DeLoach S. A., An Overview of the Multiagent Systems Engineering Methodology, in Agent-Oriented Software Engineering. P. Ciancarini, M. Wooldridge, (Eds.) LNAI Vol. 1957, Springer Verlag, Berlin, January 2001

[219] Woodcock S., Game AI: The State of the Industry, Game Developer, August, 2000

[220] Wooldridge M., Agent-based software engineering, IEE Proc Software Engineering 144, pp26-37, 1997

[221] Wooldridge M. and Jennings N. R., Agent Theories, Architectures, and Languages: a Survey, in Wooldridge and Jennings Eds., Intelligent Agents, Berlin: Springer-Verlag, 1-22, 1995

[222] Wooldridge M. and Jennings N. R., Pitfalls of agent-oriented development, in Proceedings of the Second International Conference on Autonomous Agents (Agents 98), pp385–391, Minneapolis/St Paul, MN, May 1998

[223] Wooldridge M., Jennings N. R., David Kinny, The Gaia Methodology for Agent-Oriented Analysis and Design, Autonomous Agents and Multi-Agent Systems, Vol. 3, No. 3, pp285-312, 2000

[224] Wooldridge M. and Ciancarini P., Agent-Oriented Software Engineering: The State of the Art, In P. Ciancarini and M. Wooldridge, editors, Agent-Oriented Software Engineering. Springer-Verlag Lecture Notes in AI Volume 1957, January 2001

[225] Wray, Soar: A Functional Approach to General Intelligence, accessed from: http://ai.eecs.umich.edu/~soar/docs.html, 2002

[226] Yim H., Cho K., Kim J., Park S., Architecture-Centric Object-Oriented Design Method for Multi-Agent Systems, ICMAS 2000

[227] Yoon S.Y., Blumberg B. M., Schneider G. E., Motivation driven learning for interactive synthetic characters. In Proceedings of Autonomous Agents 2000

[228] Zordan V.B. and Van Der Horst N.C., Mapping optical motion capture data to skeletal motion using a physical model, Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, San Diego, California, USA, pp245 - 250, 27-31 July, 2003

Appendix A – A List of Publications

Books and Chapters

[1] A. Szarowicz, M. Mittmann, J. Francik, Chapter "Intelligent Action Acquisition for Animated Learning Agents" in 'Learning Coordination and Communication in MultiAgent Systems', Theory and Applications Edited by Lakhmi C. Jain, World Scientific, to appear, (2004)

[2] N.D. Monekosso, P. Remagnino, A. Szarowicz, Chapter "An Improved Q-Learning Algorithm Using Synthetic Pheromones" in 'From Theory to Practice in Multi-Agent Systems', Lecture Notes in Computer Science. VOL. 2296 Edited by Dunin-Keplicz, B. and Nawarecki, E., Springer-Verlag, March, pp. 197. (2002)

[3] M. Czaja, A. Szarowicz, Chapter "Foundations of Digital Technology" in 'Construction and Design of Computers', vol. 2211 Edited by Grzywak A., Silesian Technical University Press, pp. (in Polish). ISBN/ISSN 0434-0825-098 (2000)

Journal Papers

[4] A. Szarowicz, J. Francik, M. Mittmann, P. Remagnino, "Layering and Heterogeneity as Design Principles for Animated Embedded Agents" in 'International Journal of Information Sciences, to appear', Elsevier, (2005)

Conferences

[5] A. Szarowicz, P. Remagnino, "Avatars That Learn How to behave", European Conference on Artificial Intelligence ECAI 2004, Springer, Valencia, Spain, (2004)

[6] A. Szarowicz, "Reinforcement Learning Techniques for Action Generation Using Inverse Kinematics", Postgraduate Research Conference PREP 2004, 5-7 April, Hatfield, United Kingdom, (2004)

[7] A. Szarowicz, M. Mittmann, P. Remagnino, J. Francik, "Automatic Acquisition of Actions for Animated Agents", 4th Annual European GAME-ON Conference, November 19-21, London, United Kingdom, (2003)

[8] A. Szarowicz, P. Forte, "Combining Intelligent Agents and Animation", AIxIA 2003 - Eighth National Congress on AI, Lecture Notes in Artificial Intelligence, vol 2829 Springer-Verlag, September 22-26, Pisa, Italy, (2003)

[9] P. Forte, A. Szarowicz, "The Application of AI Techniques for Automatic Generation of Crowd Scenes", The Eleventh International Symposium on Intelligent Information Systems, Advances in Soft Computing Physica-Verlag, June 3-6, Sopot, Poland, pp. 209-216. ISBN/ISSN 3-7908-1509-8/1615-3871 (2002)

[10] A. Szarowicz, P. Forte, J. Amiguet Vercher, P. Gelepithis, "Application of Autonomous Agents for Crowd Scene Generation", 2nd Hellenic Conference on Artificial Intelligence SETN-02, vol. 2 April 11-12, Thessaloniki, Greece, (2002)

[11] J. Amiguet Vercher, A. Szarowicz, P. Forte, "Synchronised multiagent simulations for automated crowd scene generation", Workshop on Spatial and Temporal Reasoning with Agents Focus - International Joint Conference on Artificial Intelligence IJCAI-01, August 4-10, Seattle, USA, (2001)

[12] A. Szarowicz, J. Amiguet Vercher, P. Forte, "Multiagent interaction for crowd scene simulation", Workshop on Autonomy Delegation and Control Interacting with Autonomous Agents - International Joint Conference on Artificial Intelligence IJCAI-01, August 4-10, Seattle, USA, (2001)

[13] A. Szarowicz, J. Amiguet Vercher, P. Forte, J.H. Briggs, P. Gelepithis, P. Remagnino, "The Application of AI to Automatically Generated Animation", 14th Australian Joint Conference on Artificial Intelligence AI01, LNAI 2256: Advances in Artificial Intelligence Springer-Verlag, Dec 10-14, Adelaide, Australia, pp. 487-494. (2001)

Appendix B – FreeWill Algorithms

A1. Algorithm controlling an avatar's behaviour (pseudocode, see Section 4.5)

```
DoSensing()
{
     image = Body.Sense()
     {
            return VisionCone.GetImage()
     }
     Mind.UpdateWorldModel(image)
     {
            KnowledgeBase.ModifyWorld(image)
            {
                   WorldModel.ModifyWorld(image)
            }
     }
     Mind.RevisePlan()
     {
            ActionPlanner.Plan()
            {
                   KnowledgeBase.GetGoals()
                   ExploreSolutions()
                   KnowledgeBase.GetObjectInfo()
                   {
                          WorldModel.GetObjectAttribs()
                   }
                   CreatePlan()
                   lastAction = SelectLastPlannedAction()
                   MotionControl.Decompose(lastAction)
            }
     }
     action = Mind.PickAction()
     {
                   microAction = ActionPlanner.GetMicroAction()
                   {
                          return MotionControl.GetCurrentAction()
                   }
            return microAction
     }
     return ConvertActionToEvent(action)
}
```

A2. The synchronisation algorithm (pseudocode, see Section 4.6)

```
if (self.state == STATE_WALKING)
{
      if (friend = SeeFriend() && !ShakenHandsYet(friend))
      {
            if (plan empty)
            {
                  aGoal = new Goal(friend.x, friend.y);
                  knowledgeBase.SubstituteCurrentGoal(aGoal);
            }
            if (GetDistToFriend() > 2*STEP_SIZE)
            {
                  add to plan (turn to goal) // goal == friend
                  add to plan (make step)
            }
            else
            {
                  if (friend.State == STATE_PREPARING)
                   || (friend.State == STATE_WAITING))
                  {
                        if ((DistanceToFriend - STEP_LENGTH) >
                              FRIEND_HANDSHAKE_DISTANCE)
                        {
                              add to plan (turn to goal) // goal == friend
                              add to plan (make step)
                        }
```

```
                    else
                    {
                          add to plan (turn to goal) // goal == friend

                          stepSize = DistanceToFriend -
                                FRIEND_HANDSHAKE_DISTANCE - STEP_LENGTH
                          add to plan (make step, stepSize)

                          add to plan (step in place)
                          add to plan (raise hand)
                          self.SetState(STATE_PREPARING);
                    }
              }
              else
              {
                          add to plan (step in place)
                          add to plan (raise hand)
                          self.SetState(STATE_PREPARING);
                    }
              }
              else
                    execute last action from the plan
        }
        else
        {
              plan other actions
        }
}
else if (self.state == STATE_PREPARING)
{
      if (self.ready())
      self.SetState(STATE_WAITING);
      if (plan empty)
      add to plan (no action)           // wait until finished
}
else if (self.state == STATE_WAITING)
{
      if (friend.ready()) || (friend.executing())
      {
            add to plan (shake hand)
            self.SetState(STATE_EXECUTING);
      }
      else                        // wait for the friend
      {
            if (plan empty)
            add to plan (no action)
      }
}
else if (self.state == STATE_EXECUTING) // when shakehand executed - finish
{
      add to plan (lower hand)
      add to plan (step in place)
      add to plan (step in place)

      self.SetState(STATE_WALKING);
      knowledgeBase.ResumePrimaryGoal
      SetHasShakenHands(friend);
}
```

Appendix C – Example 3DS Max Scripts

B1. The handshake script

The script generates a handshake action for two avatars

```
--seting control params for the biped
RarmCont3 = (biped.getNode currentBip 2).transform.controller
LarmCont3 = (biped.getNode currentBip 1).transform.controller
RArm3 = biped.getNode currentBip 2 link:2
LArm3 = biped.getNode currentBip 1 link:2
RForearm3 = biped.getNode currentBip 2 link:3
LForearm3 = biped.getNode currentBip 1 link:3
RHand3 = biped.getNode currentBip 2 link:4
LHand3 = biped.getNode currentBip 1 link:4


--animation sequence now interlaced
animButtonState=on

--bip01 then bip02
biped.AddNewKey LarmCont3 sliderTime
biped.AddNewKey RarmCont3 sliderTime


sliderTime = sliderTime+20

rotate RForearm3 30 [-1,0,0]
biped.AddNewKey LarmCont3 sliderTime
biped.AddNewKey RarmCont3 sliderTime


sliderTime = sliderTime+20

rotate RForearm3 80 [0,0,-1]
biped.AddNewKey LarmCont3 sliderTime
biped.AddNewKey RarmCont3 sliderTime

sliderTime = sliderTime+22
rotate RHand3 10 [0,0,-1]
biped.AddNewKey LarmCont3 sliderTime
biped.AddNewKey RarmCont3 sliderTime


-- moving down both hands
sliderTime = sliderTime+14
rotate RForearm3 10 [0,0,1]
biped.AddNewKey LarmCont3 sliderTime
biped.AddNewKey RarmCont3 sliderTime

-- compensating with the hand

rotate RHand3 10 [0,1,-1]
biped.AddNewKey LarmCont3 sliderTime
biped.AddNewKey RarmCont3 sliderTime

-- moving up both hands
sliderTime = sliderTime+6
rotate RForearm3 20 [0,0,-1]
biped.AddNewKey LarmCont3 sliderTime
```

```
biped.AddNewKey RarmCont3 sliderTime

-- compensating with the hand
rotate RHand3 10 [0,-1,1]
biped.AddNewKey LarmCont3 sliderTime
biped.AddNewKey RarmCont3 sliderTime

-- moving down both hands again (2)
sliderTime = sliderTime+10
rotate RForearm3 20 [0,0,1]
biped.AddNewKey LarmCont3 sliderTime
biped.AddNewKey RarmCont3 sliderTime

-- compensating with the hand
rotate RHand3 10 [0,1,-1]
biped.AddNewKey LarmCont3 sliderTime
biped.AddNewKey RarmCont3 sliderTime

-- moving up both hands again (2)
sliderTime = sliderTime+10
rotate RForearm3 20 [0,0,-1]
biped.AddNewKey LarmCont3 sliderTime
biped.AddNewKey RarmCont3 sliderTime

-- compensating with the hand
rotate RHand3 10 [0,-1,1]
biped.AddNewKey LarmCont3 sliderTime
biped.AddNewKey RarmCont3 sliderTime

sliderTime = sliderTime+28
rotate RForearm3 100 [0,0,1]
biped.AddNewKey LarmCont3 sliderTime
biped.AddNewKey RarmCont3 sliderTime

rotate RForearm3 30 [1,0,0]
biped.AddNewKey LarmCont3 sliderTime
biped.AddNewKey RarmCont3 sliderTime

animButtonState=off
--end of animation sequence
```

B2. Example teapot lifting script

This script was generated as a result of applying the learning algorithm to the task of lifting a teapot (Chapter 5)

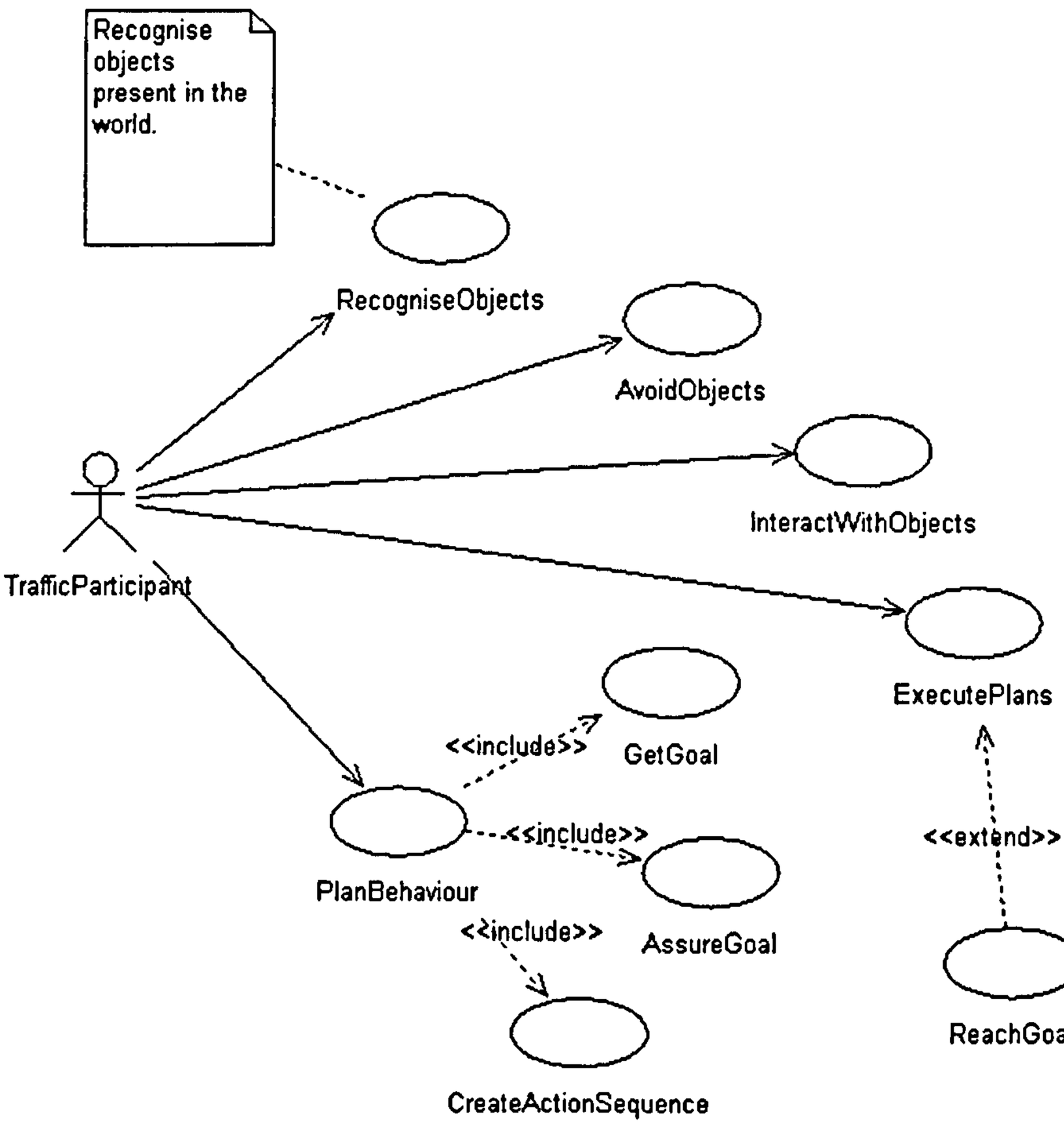
```
animationRange = interval 0 400

CreateScene()
CreateBiped ()

bipLClavicleCtrl = (biped.getNode bipObj  1).transform.controller
teaCtrl = tea.controller
tea2 = Teapot radius:15 smooth:on segs:4 body:on handle:on spout:off
lid:on mapCoords:off pos:[15,35,108] isSelected:on
tea2Ctrl = tea2.transform.controller

animate on
```

```
(
	sliderTime = 0
	addNewKey tea2Ctrl 0
	biped.AddNewKey bipLClavicleCtrl 0

	sliderTime = 20

	BipedMoveLHandTo 0.000000 8.000000 103.298172
	addNewKey tea2Ctrl 20
	biped.AddNewKey bipLClavicleCtrl 20
	sliderTime = 40

	BipedMoveLHandTo -8.000000 8.000000 103.298172
	addNewKey tea2Ctrl 40
	biped.AddNewKey bipLClavicleCtrl 40
	sliderTime = 60

	BipedMoveLHandTo -8.000000 8.000000 111.298172
	addNewKey tea2Ctrl 60
	biped.AddNewKey bipLClavicleCtrl 60
	sliderTime = 80

	BipedMoveLHandTo -16.000000 8.000000 111.298172
	addNewKey tea2Ctrl 80
	biped.AddNewKey bipLClavicleCtrl 80
	sliderTime = 100

	BipedMoveLHandTo -16.000000 8.000000 119.298172
	addNewKey tea2Ctrl 100
	biped.AddNewKey bipLClavicleCtrl 100
	sliderTime = 120

	BipedMoveLHandTo -16.000000 8.000000 127.298172
	addNewKey tea2Ctrl 120
	biped.AddNewKey bipLClavicleCtrl 120
	sliderTime = 140

	BipedMoveLHandTo -16.000000 16.000000 127.298172
	addNewKey tea2Ctrl 140
	biped.AddNewKey bipLClavicleCtrl 140
	sliderTime = 160

	BipedMoveLHandTo -16.000000 24.000000 127.298172
	addNewKey tea2Ctrl 160
	biped.AddNewKey bipLClavicleCtrl 160
	sliderTime = 180

	BipedMoveLHandTo -8.000000 24.000000 127.298172
	addNewKey tea2Ctrl 180
	biped.AddNewKey bipLClavicleCtrl 180

	GrabTeaPot()
	sliderTime = 200

	BipedMoveLHandTo -8.000000 24.000000 127.298172

	ReleaseTeaPot()

	tea2.rotation = tea.rotation
	tea2.pos = tea.pos
```

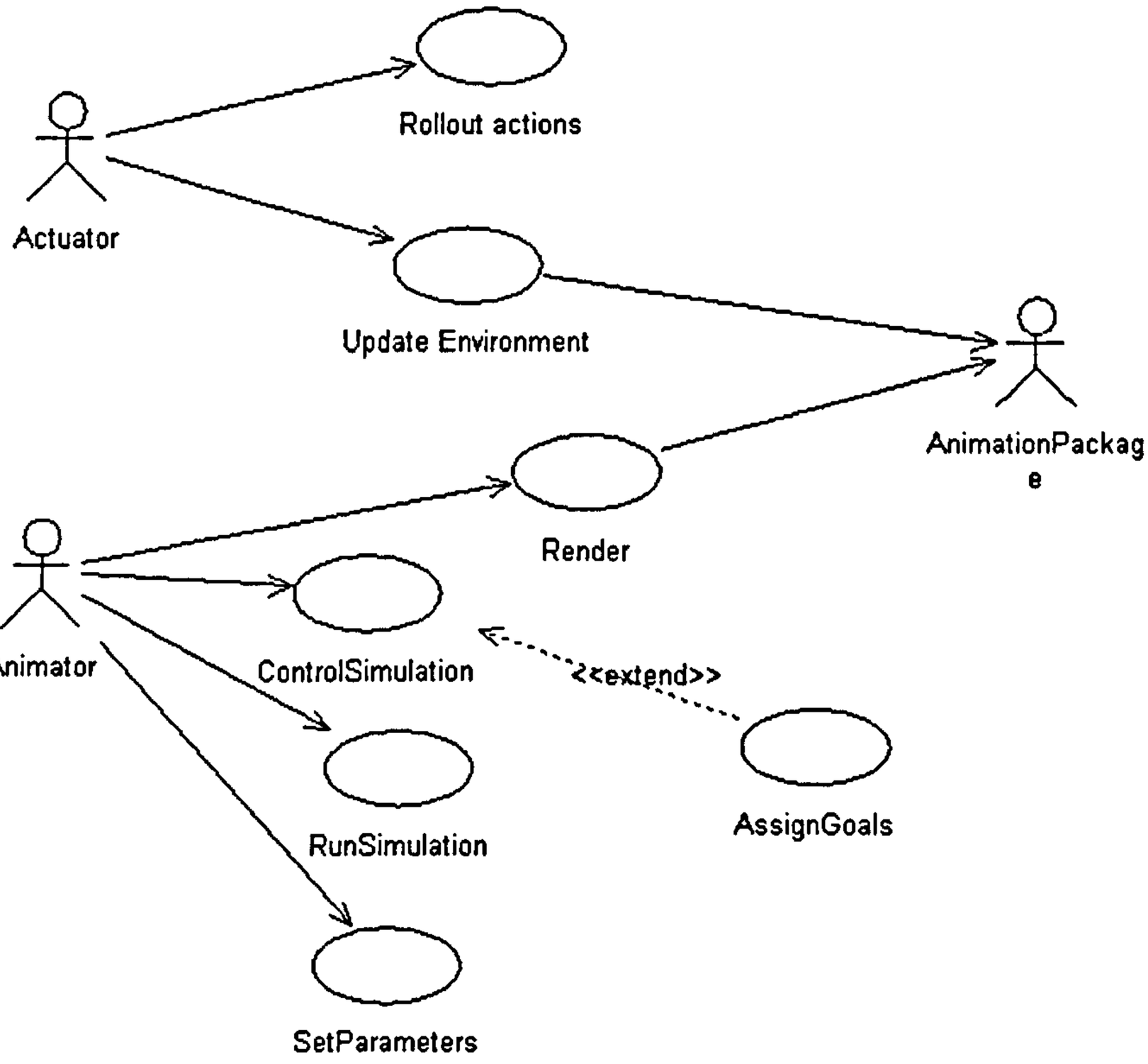
```
addNewKey tea2Ctrl 200
biped.AddNewKey bipLClavicleCtrl 200

GrabTeaPot()
sliderTime = 220

BipedMoveLHandTo 0.000000 24.000000 127.298172

ReleaseTeaPot()

tea2.rotation = tea.rotation
tea2.pos = tea.pos
        addNewKey tea2Ctrl 220
        biped.AddNewKey bipLClavicleCtrl 220
)
delete tea
delete boxHandle
```

Appendix D – Example UML Diagrams

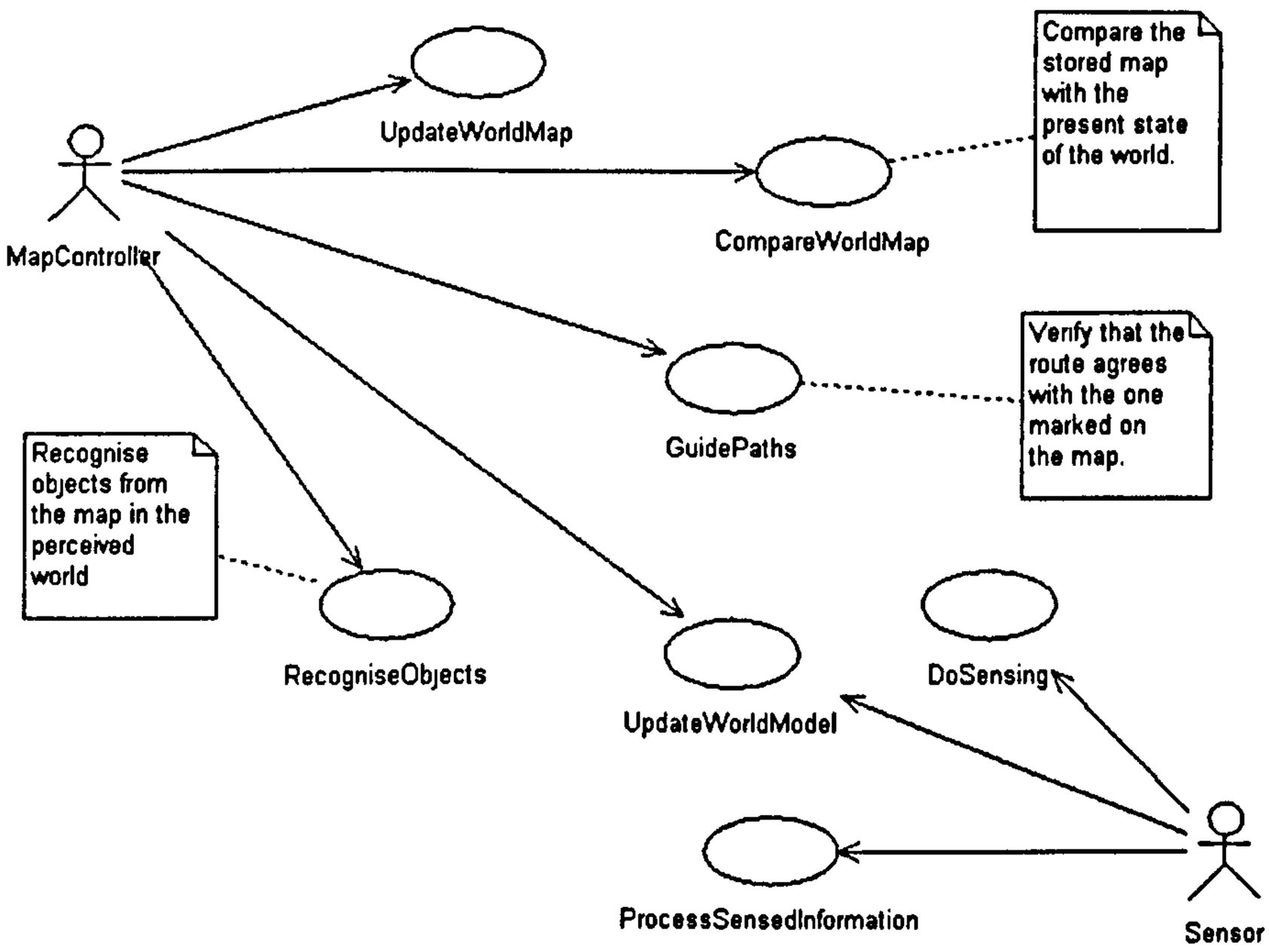
As described in Chapter 4 the FreeWill system has been designed and documented using UML. Some of the UML diagrams are presented below, see also Chapter 4.



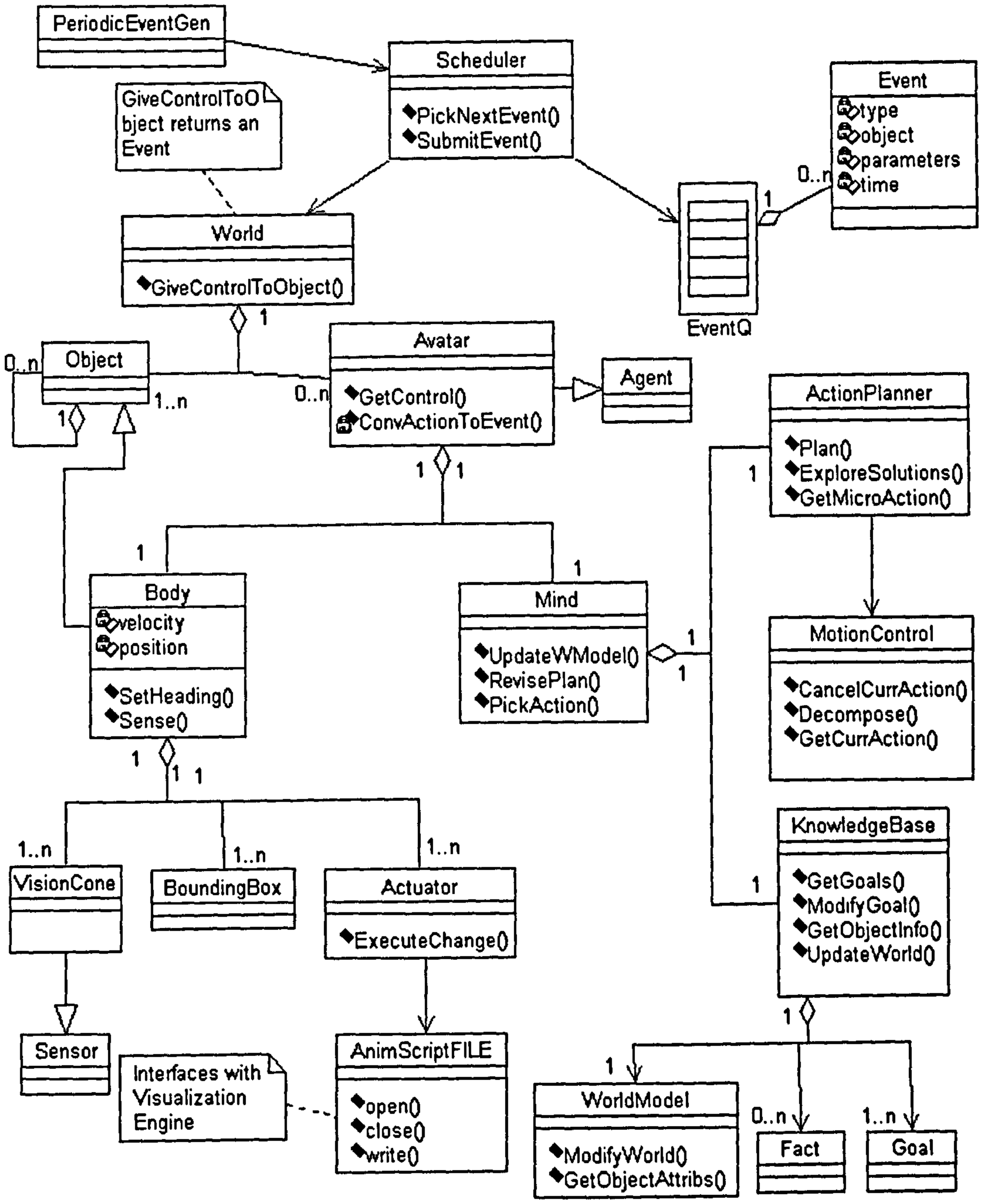
Initial use case model of the system (1)



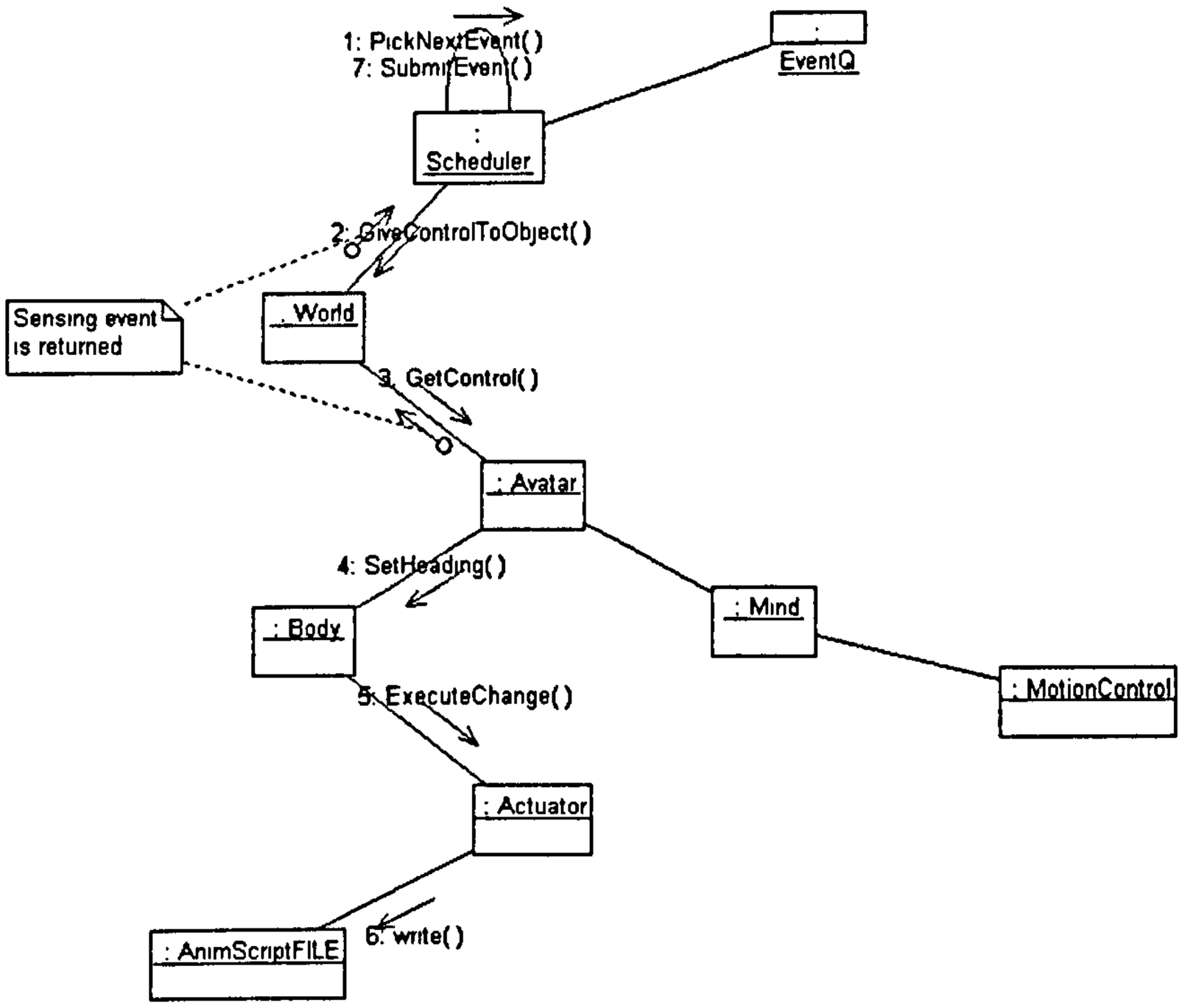
Initial use case model of the system (2)



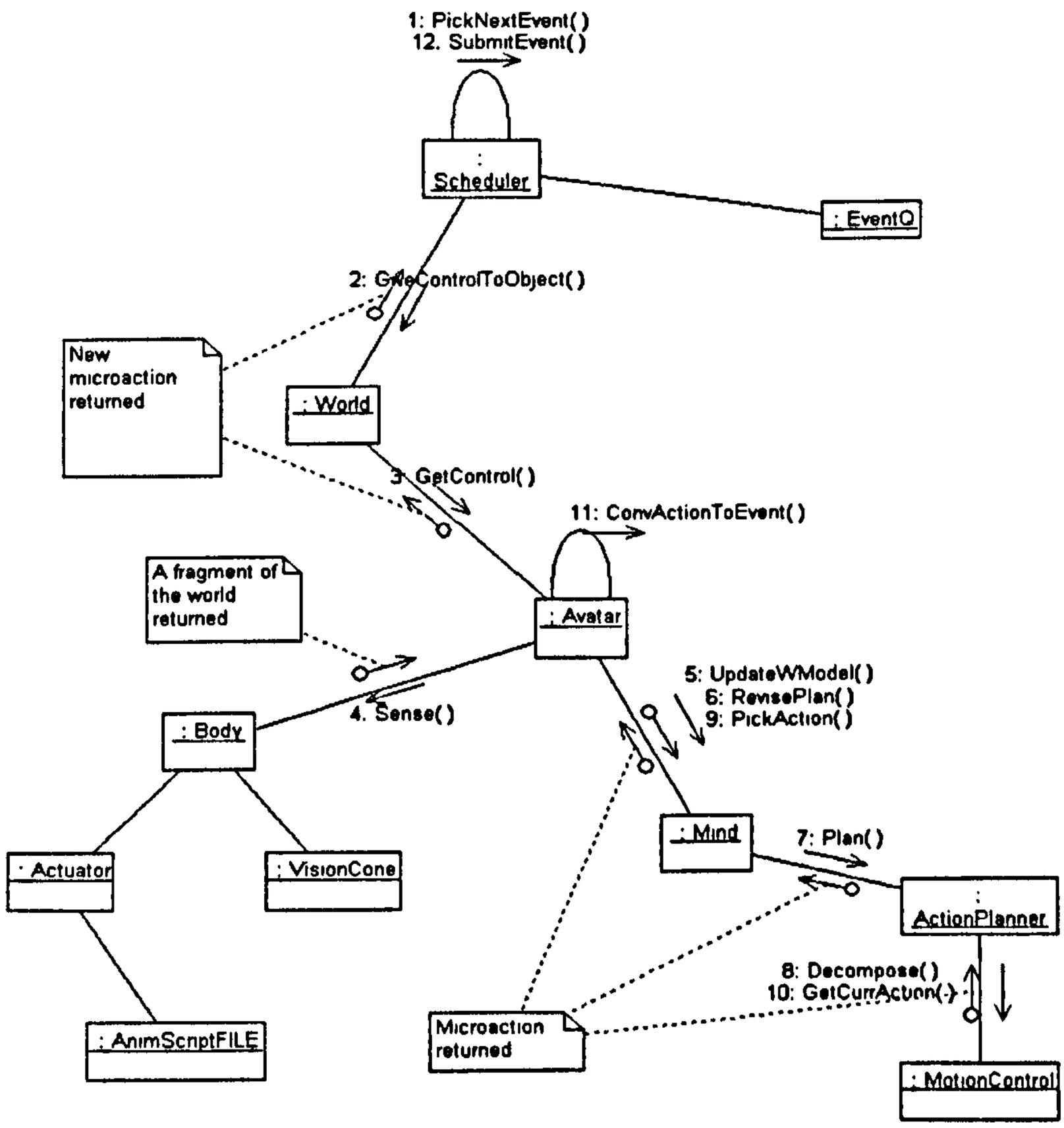
Initial use case model of the system (3)



Class diagram of the system



Message sequence for an acting event



A sensing collaboration

Appendix E – Metrics Results

Three metrics have been proposed as a way of limiting the number of different action sequences generated by the learning algorithm. Definitions of these metrics and discussion of results are presented in Chapters 5 and 6, this appendix presents pairs generated by the Local Distance Metric on a set of 651 sequences obtained for the FK teapot experiment and a set of 72 sequences obtained for the IK teapot experiment. The third part of this appendix presents the result of applying the Global Distance Metric to the first set of sequences (for discussion see Section 6.8).

6, 457	40, 605	108, 589	233, 397
6, 458	41, 450	108, 590	233, 543
6, 610	41, 451	109, 435	233, 544
6, 611	41, 603	109, 437	237, 400
7, 456	41, 604	109, 588	237, 401
7, 458	45, 454	109, 590	237, 547
7, 609	45, 455	110, 435	237, 548
7, 611	45, 607	110, 436	238, 399
8, 456	45, 608	110, 588	238, 401
8, 457	46, 453	110, 589	238, 546
8, 609	46, 455	219, 469	238, 548
8, 610	46, 606	219, 470	239, 399
12, 460	46, 608	220, 468	239, 400
12, 461	47, 453	220, 470	239, 546
12, 613	47, 454	221, 468	239, 547
12, 614	47, 606	221, 469	306, 469
13, 459	47, 607	222, 475	306, 470
13, 461	102, 433	222, 476	307, 468
13, 612	102, 434	223, 474	307, 470
13, 614	102, 586	223, 476	308, 468
14, 459	102, 587	224, 474	308, 469
14, 460	103, 432	224, 475	309, 475
14, 612	103, 434	231, 397	309, 476
14, 613	103, 585	231, 398	310, 474
39, 451	103, 587	231, 544	310, 476
39, 452	104, 432	231, 545	311, 474
39, 604	104, 433	232, 396	311, 475
39, 605	104, 585	232, 398	no of pairs: 120
40, 450	104, 586	232, 543	
40, 452	108, 436	232, 545	
40, 603	108, 437	233, 396	

LDM applied to the FK set

0, 31	4, 28	9, 39	13, 36
0, 32	4, 54	9, 40	13, 62
0, 58	4, 55	9, 67	13, 63
0, 59	4, 67	9, 68	15, 25
0, 62	4, 68	11, 29	15, 33
0, 63	6, 25	11, 37	15, 60
2, 29	6, 52	11, 65	16, 25
2, 56	6, 65	12, 29	16, 33
2, 60	7, 25	12, 37	16, 60
3, 29	7, 52	12, 65	17, 58
3, 56	7, 65	13, 27	17, 59
3, 60	9, 31	13, 28	17, 67
4, 27	9, 32	13, 35	17, 68

19, 56	21, 62	25, 49	31, 42
19, 65	21, 63	25, 50	32, 42
20, 56	23, 52	27, 47	no of pairs: 72
20, 65	23, 60	28, 47	
21, 54	24, 52	29, 44	
21, 55	24, 60	29, 45	

LDM applied to the IK set

most	433	548	most similar: (dist==2)
different:	434	586	0
(max ==8)	436	587	21
220	437	589	33
221	451	590	60
223	452	604	96
224	454	605	147
307	455	607	225
308	457	608	318
310	458	610	462
311	460	611	621
397	461	613	no of seqs: 10
398	544	614	
400	545	no of seqs: 40	
401	547		

GDM applied to the FK set