

Virtual actors that can perform scripts and improvise roles

Peter Wavish

Philips Research Laboratories
Redhill, Surrey, RH1 5HA, United Kingdom
wavish@prl.research.philips.com

David Connah

Consultant to the Institute for Perception Research (IPO)
PO Box 513, 5600 MB Eindhoven, The Netherlands
david@agents.demon.co.uk

Abstract

This paper describes work in progress on the Humanoid 2 ESPRIT project, in which we are integrating the human figure animation software described in (Magnenat Thalmann & Thalmann 1995) with the agent technology described in (Wavish & Graham 1996) in order to produce virtual actors for films, games and virtual reality worlds. In this paper we focus on how the actors perform scripts and improvise role-related behaviour. Our approach to creating believable characters is modelled on how characters are portrayed by human actors during a theatrical performance. The character played by the actor is originally specified in the script. The actor's purpose is to perform the script as skilfully as it can in order to portray the character. Our actors are constructed internally as societies of simple situated agents, where each agent corresponds to a skill or to a role. Skills are task oriented behaviours of an individual actor, whereas roles provide a means for the actor to engage in routine social interactions.

1. Introduction

Computer graphics technology has reached the stage where human figures can now be modelled, animated and rendered with close to photo-realistic results (Magnenat Thalmann & Thalmann 1995). At the same time, computer games companies are putting sophisticated 3-D graphics hardware platforms into the home. In contrast, the technology for making simulated human figures *behave* realistically, so that they can engage in lifelike social interactions with each other and with human users, is in its infancy. Simulated humans have potential applications in a range of entertainment and business products such as films, computer games, training simulations, home shopping, shared virtual spaces and interactive story worlds. The technical factor that determines the commercial acceptability of these applications is the degree to which the simulated humans are able to sustain believable behaviour.

It is not at present possible to simulate human behaviour perfectly, because that would involve creating a human mind, which is well beyond the reach of current technology. Some degree of deception is therefore necessary to make the human observer (or user, or interactor) believe that the human figure he or she is looking at is acting in a deep, cognitive, manner when in fact it is not. Deception of this kind is commonplace in the arts; it is what authors, actors and film directors achieve when they

create characters. A character, whether in a cartoon or in a serious novel, is essentially a convincing depiction of an individual who has no actual existence.

The central issue therefore in animating human figures is not how to make the figures (or strictly speaking, the agents controlling them) more intelligent as such, but to make them more capable of portraying characters convincingly. The natural model to turn to, to see how this might be done, is the theatre, in which human actors are highly skilled at portraying characters while suppressing their own personal states of mind. The approach to creating characters described in this paper is therefore to automate the skills associated with acting; in other words, to produce virtual actors.

This paper describes work in progress on designing virtual actors being carried out within the Humanoid 2 ESPRIT project. This European project brings together research on agent-controlled human figure animation (Magnenat Thalmann & Thalmann 1995) and research on agent-based characters (Wavish & Graham 1996). The objective is to provide autonomous actors that can interact with each other and with human users to a limited extent. There are immediate applications for this technology in film production and in computer games. The human figure animation work has already been extensively documented, so in the remainder of the paper we will confine our attention to the design of the actor agents that control the simulated human figures.

It is important to emphasise the practical criteria that have guided us in this work. We have deliberately chosen to define a rather simple architecture in order to be able to implement a system that can be used in the short term for real applications. This simplicity is to some extent deceptive, however, and we believe that systems built in this way can be very powerful in their intended field of application.

Other, related, work includes the work at CMU (Bates, Loyall & Reilley 1991), New York University (Perlin 1995), MIT (Maes, Darrel & Pentland 1995), and Stanford University (Hayes-Roth & Feigenbaum 1994). Whilst most of these groups are addressing a similar set of issues and problems, their intended applications and their solutions to the problems raised are very diverse and are all different from the approach described here.

2. Issues in the design of virtual actors

2.1 Believable characters

The direct approach to creating believable characters is to simulate those aspects of mind that are important for creating the impression that the character is real. These aspects typically include goal-oriented behaviour, coherent emotional responses, and language generation and understanding. Considerable progress has been made in creating characters which can be given

different personalities by setting parameters or by additional programming. By carefully limiting the depth to which aspects of mind are simulated, it is possible to control the computational costs, and this approach supports detailed interactions between users and characters within the context of a particular fictional setting (Bates, Loyall & Reilley 1991). However, there are current applications, such as computer games or virtual reality worlds, where such detailed interactions are unnecessary, and where much shallower characters can be tolerated. Practical considerations such as the amount of CPU time or memory space required for portraying personality, and the testability of the software, are typically of more importance to developers than the degree of interaction that can be obtained. Computer game developers in particular are accustomed to working within the constraints of a given technology, and to exploiting its limits to the full.

2.2 The theatre metaphor

The metaphor of computer as theatre is well known (Laurel 1991). However it is not usually applied coherently to the creation of believable characters. In the theatre, characters are specified in a script, and the actors portray the characters by improvising actions that are consistent with what the author has conveyed about the character via the script.

A practical advantage of creating characters in virtual worlds in this way is that the virtual actors require no internal model of mind. What they need are the skills associated with portraying the state of mind of a character. The problem of automating particular human skills is technically much easier than the more general problem of simulating human minds.

Whilst some applications, such as virtual theatre (Hayes-Roth & Feigenbaum 1994), demand a rather complete application of this metaphor, other applications requiring believable characters do not. Nevertheless the strategy of building characters as actors performing scripts and improvising roles is applicable even to characters in computer games (Wavish & Graham 1996).

2.3 Foreground and background

Performing a scripted direction typically requires considerable improvisation on the part of the actor. This is because the actor is embedded in a situation consisting of its immediate physical environment, the other actors, and the audience, as well as the fictional world of the work being performed. Actions must be performed in a way that is appropriate to this complex situation. So virtual actors must be able to improvise their actions to some extent even while they are following a script.

Typically a script specifies the action of only one actor at a time, and actors not directly involved in the foreground action may receive little or no guidance from the script for extended periods of time. Thus as well as performing scripted foreground action, virtual actors must also be capable of improvising unscripted background action that is consistent with their roles. The script acts as a vehicle for communicating what these roles are, even though it does not directly specify the actor's actions.

Given that actors, most of the time, are required to improvise suitable role-related background behaviour, but from time to time must perform scripted foreground action, a particular problem for the design of virtual actors is to manage the transition between improvised background and scripted foreground behaviour gracefully. In our system the performance of the script results from the collaborative activity of the actors; there is no central control. Individual actors are therefore free to choose between improvising background action and performing scripted actions. The script is regarded, not as a set of commands, but as yet another aspect of the actor's situation which constrains the

improvisation of actions. In this system scripts are analogous to the role of plans as discussed by (Suchman 1987); they are a resource for action.

2.4 Scripted action

The script, then, is the primary way that authors can describe characters and their actions. Unlike real actors, our virtual actors have no real cognition and hence are incapable of reading and understanding scripts in any depth. They do however need to have something equivalent to a written script which specifies their roles and their actions in a way that they can easily interpret. A practical approach which we are pursuing is for the author to write the script in a highly simplified restricted form of English, and to translate this formal script into an even simpler form that the actors can understand.

At this point it is necessary to anticipate a little and explain what limits the ability of the actors to understand a script. Actors are implemented as situated agents that communicate by means of deictic representation, so that objects are referred to by pointing to them rather than by naming them. Objects that are referred to in discourse between actors must be physically present in the actor's environment. The script therefore needs to be represented to the agents in a deictic form that they can understand.

At the formal level, the script specifies roles and actions in simple sentences such as:

Betty goes to the chair by the window.
Betty sits on it.

Even for such simple sentences there are problems of disambiguation (which window?) and anaphora (what is 'it?') which must be solved. Generally, it is the responsibility of the script translator, which generates the runtime representation from the formal script, to do this. Referents must be found, and pointed to, so that the actors can have direct access to the other actors or objects being referred to. Since referents do not exist until the start of a performance, and in some cases not even then (the chair may be placed by the window by another actor), the process of finding referents must continue during performance. In some cases finding referents may involve the activity of the actor, where it is clear that the object being referred to is connected in some way with the actor's current focus of attention.

A further problem is that of concurrency. A strictly sequential script would prohibit actors from performing scripted actions concurrently with other actors. The device used to overcome this problem is to specify explicitly that an activity is starting or stopping. So, for instance, in this example:

Arnold starts walking to the door.
Betty turns towards the window.

the two actions are performed concurrently.

Furthermore, some activities may persist throughout the performance, typically those associated with portraying some aspect of a character's personality. So scripts may contain sentences such as:

Arnold starts being a male.
Betty starts being a female.
Arnold starts being a friend of Betty.

and so on. These sentences specify roles that the actors adopt and portray, typically by opportunistic, situation dependent behaviour. Thus it is the social behaviour associated with being male, not the sex of the character, that is being specified here.

A character may be playing many roles simultaneously. This does not necessarily mean that the amount of activity for a given actor is correspondingly increased since, as mentioned above, role-related action is opportunistic and is only required when the

role is relevant to the current situation. What it does mean is that the actor has a much richer repertoire of behaviour as a result of the multiple roles which it is capable of enacting.

2.5 Improvised action

Roles govern the way in which actors interact with one another in social situations. They are performed by complementary sets of skills which relate and constrain the behaviours of the actors taking part so as to allow larger scale social scenarios to take place. Examples of this are the roles of giver and receiver when something is being passed between actors or the roles of buyer and seller when a purchase is being made. Once the interaction is entered the sequence of actions and responses of each actor is both expected and required (although not literally forced on them). There is also a sense in which these expectations and fulfilments are a way in which actors can 'understand' the actions of other actors.

The capability for improvising role-related behaviour is central to the virtual actors described here. As already mentioned, for much of the time, even in a scripted scenario, individual actors must improvise unscripted background activity, and their repertoire of roles allows them to engage in complex social interactions, such as buying a drink, without detailed scripting of action. Simulations of routine social interactions are designed in a top-down manner, by designing *organisations* in terms of their constituent roles, and then implementing the set of skills needed by the actors to perform the required roles. At run time, instances of organisations self-assemble as sets of interacting actors playing complementary roles.

Actors, as well as being able to interact with other actors, must be able to interact with objects in the physical world. This interaction includes activities such as moving about from place to place avoiding obstacles, performing actions such as sitting on a chair, and more obviously skilful activities such as pouring drinks. Skills of this kind can be specified directly in the script, or perhaps more often, are required (as in the bartender example above) in order to perform particular roles. Actors also need skills which are specifically to do with acting. The most obvious skills required are those of reading the script, identifying and accepting cues, and making proper use of the stage or set.

More importantly, the actor is concerned with portraying the personality of the character in which it has been cast. This involves making choices about how best to do this. For instance, anger could be portrayed by facial expression or by violent action, such as thumping the table. For an actor, the choice of what to do depends on the circumstances of the performance; for instance, if the actor is not currently facing the audience, facial expression will not be effective and the actor must either turn or else express anger in some other way.

These acting skills are not different in kind from the skills required for playing roles or for interacting with the physical world; in all three cases they can be regarded as a means of producing actions that are appropriate to the various situations in which the skills are deployed.

3. Communicating deictic agents (CDAs)

The basic building blocks from which the system, including the set, the cast and the script, is built are simple autonomous agents. To distinguish these finer grain agents from the actor agents, we will refer to them as communicating deictic agents (CDAs). A general discussion of deixis can be found in (Cremers 1996). The individual CDAs used in this system vary in capability, but are closely related in concept to the Pengi and Sonja video game players (Agre & Chapman 1987, Chapman 1991) and have their

theoretical justification in the notion of situated action (Suchman 1987).

Each CDA is a (simulated) asynchronous digital logic circuit, consisting of registers, logic elements and time delays, which is coupled to its environment by means of markers. Markers provide a way of referring to objects (including other CDAs) deictically. That is, instead of objects being referred to by name, they are referred to by their position in space relative to the CDA, typically by pointing at them. A CDA that has a marker on another CDA can access directly the components of the other CDA's internal state, including the states of its registers and the positions of its markers.

CDAs are intermediate in power between purely reactive agents, which have no internal representations, and classical artificial intelligence agents, which have explicit symbolic models of their environment. CDAs represent their environment by means of the boolean variables (registers) embedded in their logic circuits and by the positions of their markers. Part of the skilled activity of a CDA is concerned with positioning its markers so that they correspond to its focus of attention and current interests and so allow the CDA to represent efficiently just those parts of its environment with which it is currently concerned.

Two CDAs can communicate by updating their own internal state appropriately and by observing each other's state, and can communicate *about* things, such as other CDAs, by observing the positioning of each other's markers. This means that the designer must make at least some of a CDA's internal state meaningful to other CDAs. In practice, this tends to occur in any case, because representations of the CDA's own activity are needed for its own internal decision making processes. Some of the activity of the CDA is normally concerned with maintaining these internal representations consistent with its actual behaviour.

This communication mechanism, based on deixis, is quite different in style from conventional message passing, in which messages sent by one agent are automatically received by another named agent. In the case of CDAs, communication must begin by one CDA attracting the attention of another, so that the second CDA comes to place a marker on the first one. Methods for doing that resemble everyday techniques such as saying a person's name, waving, or catching a person's eye. The equivalent of the last case is that the first CDA knows that it has the attention of the second if it can see that the second CDA's attention marker is positioned on it. Subsequent communication resembles non-verbal human communication by means of facial expression, gesture, posture, pointing actions and direction of gaze; that is, it depends on the skill and attention of the listener in extracting the content of the communication using the shared situation as a resource.

Another departure from normal practice is that privacy is not enforced. One CDA can directly affect another CDA's internal state. However, CDAs are typically designed to be 'polite'; one CDA can affect the actions of another by exploiting the other's own dynamics. Adopting a particular role, so provoking another CDA into playing the complementary role, is one way to do this.

CDAs can be seen to conform with the weak notion of agency described by (Wooldridge & Jennings 1995):

- autonomy: each CDA in a system has its own inherent dynamics resulting from the structure of its digital logic circuit.
- social ability: as already described, CDAs communicate by having mutual access to (subsets of) their internal state and can self-assemble into organisations in which each CDA plays a predefined role.
- reactivity: changes of state induced by external events propagate asynchronously and concurrently across the

asynchronous digital logic circuit from sensors to effectors.

- pro-activeness: the asynchronous digital logic circuit can easily be designed so that it determines what actions to perform on the basis of its own internal state and the state of other CDAs that are currently being marked.

Although digital logic circuits can be designed to correspond to logical formalisations of naive psychology concepts such as goals and intentions (Rosenschein and Kaebbling 1986), so conforming to the stronger notion of agency also described by Wooldridge & Jennings, the CDAs used in this system are not designed on that basis. The approach taken to their design is that of behaviour-based AI [Brooks 1986, Maes 1993], in which an agent is conceptualised in terms of its behaviour rather than in terms of mentalistic notions such as beliefs, desires and intentions. This does not however preclude CDAs from having beliefs, desires and intentions ascribed to them (Dennett 1987), or from having them in the stronger sense that intentional concepts can be related to the behaviour of complex dynamical systems (Kiss 1991).

A stronger characterisation of individual CDAs is that they provide a way of capturing human skills by building them as sets of task achieving behaviours. An example of this is the tetris player described in (Wavish & Graham 1995).

One of the most important aspects of the CDAs used as the basis of the architecture of this system is that they scale. The simulated asynchronous digital logic circuits are change driven, hence the CPU time used is proportional to the amount of activity occurring, rather than to the absolute size of the digital logic circuit. Most of the time, most of the circuit does nothing and consumes no power. This makes it feasible to consider very large individual CDAs and very large numbers of CDAs in a system without incurring significant computational costs. The system currently being built contains hundreds of CDAs, and could easily be expanded to contain thousands.

CDAs can readily be implemented by encoding them in the RTA programming language (Wavish 1990, Graham & Wavish 1991, Wavish & Graham 1995). Their behaviour is written in terms of time-annotated situation-action rules, and these are compiled into the digital logic circuit representation. Unlike earlier approaches such as (Agre & Chapman 1987), the programmer does not need to conceptualise the CDA design in hardware terms. Support for the marker based communication mechanism is built directly into RTA, which supports unlimited numbers of CDAs executing concurrently along one or more different timelines. In RTA the basic CDA model is augmented by a range of datatypes whose values are updated by means of constraints implemented in C, so providing a general purpose computational facility as well as a convenient means of embedding CDAs in the host software system.

4. System architecture

4.1 Scripts, actors and physical objects

The system can be considered as having three different levels:

- The script, which exists in two forms, a formal written form, and a runtime form in which instructions are specified deictically.
- The agents controlling the virtual actors which perform the script and improvise background activity.
- The simulated physical world in which the actors are embedded, consisting of simulated physical objects (including the animated bodies of the actors) modelled using standard 3-D computer graphics techniques.

This neat categorisation in terms of layers does not mean that

each layer can be treated in an implementation independent manner. On the contrary, because the actor agents are essentially situated agents, they rely on the world in which they are embedded to minimise the work of representing it. Situated agents engage in loops of activity threading through each other and through physical objects in their environment. To produce a coherent system, therefore, it is necessary to consider not only how agents produce action, but also how agents appear to each other and (in the case of agents embedded in simulated worlds) how physical objects can best be represented.

A way of simplifying the system design is therefore to model all entities in this world (including the script) using CDAs. This provides a uniform means of communication, so that actors can easily read their script, and can interact with inanimate physical objects as easily as they can interact with each other.

4.2 The Script

The formal script is translated by software into a runtime deictic representation consisting of 'fragments' and 'refexes' (referential expressions) where the fragments are keyed to the main verb of each sentence. Fragments and refexes are implemented as CDAs. Fragments are linked to each other and to their refexes by markers. Once the script has been translated, the fragments do not change, but the refexes actively seek to find referents by examining the contents of the simulated world in which the script is being performed. Refexes can examine objects to match their properties against the referential expression. Depending on the type of referential expression, they can also examine the internal state of the actor in order to identify referents that are related in some way to the actor's current interests.

The searching activity of a refex continues until a referent has been found, which in normal circumstances is before the time at which the referent is needed by the actor performing the fragment with which it is associated. Once a refex finds its reference, it indicates this by placing a marker on it. If the referent subsequently ceases to match the referential expression, the refex can uncouple itself and resume its search for another referent.

This method of determining referents has two consequences. One is that once a referent has been determined, there is no ambiguity about what it is, which might be the case if it were left to the actors to determine referents. Thus collaborative action does not need to cope with breakdowns caused by differences of interpretation. The other is that the system is tolerant to changes in the simulated world in which the script is performed. The layer of refexes effectively decouples the script from the world, so that the same script can be enacted in different virtual worlds (because the refexes automatically search for appropriate referents), and modifications can be made to a world even while the performance is in progress.

4.3 Actor agents

The actor agents are constructed as societies of CDAs. Each CDA implements a role or skill that the actor can perform, either by itself, or in collaboration with other CDAs. There is little distinction between roles and skills at this level. In fact CDAs corresponding to roles inherit all of the default behaviour of skill CDAs, including a repertoire of behaviour used for communicating with other CDAs. In addition they have markers which can be placed on other actors performing the other complementary roles, so providing the means by which the actor is able to interact with other actors in a role-determined manner.

The actor agent architecture superficially resembles some existing multi-agent models such as Minsky's society of mind (Minsky 1986), Maes' behaviour nets (Maes 1991) and various DAI models. However its aim is much less ambitious. Although

CDAs in the actor can act concurrently, they normally don't. At any one time only a small subset of skills and roles are active. Thus objectives such as making actions emerge from the collective activity of a large number of CDAs are not essential to the design of actors. On the contrary, the development of the actor depends on partitioning its overall capability into a large number of relatively independent skills which are simple enough to be easily programmed. Skills and roles can invoke each other in a top-down manner, and can compete for resources, but only in particular cases do they operate collaboratively.

An advantage of structuring the actor agent in this way is that a one-to-one mapping can be set up between fragment types in the runtime representation of the script and CDAs in the actor agent, so that the vocabulary of roles and actions available to the script writer can easily be extended by adding corresponding roles, skills and behaviours to the actor. Because, for each fragment type, there is a matching type of CDA in the actor, specific information about the specified role or action can be held in the fragment and be understood by the corresponding CDA without affecting the overall architecture of the actor.

Actors must be able to perceive what is going on around them in order to be able to interact with their physical environment and with other actors. Because the world in which they exist is simulated, there is a choice to be made as to how this is done. Considering vision (although actors need to be able to hear and touch as well as to see), actors could simply be presented with images of the scene as viewed from their own position, and then carry out image understanding in order to get the information they need. At the opposite extreme, actors could have direct access to the internal states of other actors and objects and hence to symbolic representations of their behaviour, thus bypassing perception almost completely.

In fact a hybrid strategy is used. Following the 'synthetic vision' approach described in (Renault et al. 1990) the 3-D scene rendered from the actor's viewpoint is encoded so that each point in the image is labelled with an identification of the object that is seen at that position. Actors can therefore see where objects are, relative to their own position, and can find out further information about the objects by placing markers on them and so accessing their internal states. Actors can also gain access by this means to objects that they are not currently marking that may nevertheless (because of their proximity) be relevant to their activity.

Synthetic vision is also used to support behaviours that rely on unrepresented spatial relations between objects in the world. For instance, to identify 'the book on the table' it is only necessary to look at the book and check that the table is beneath it in the field of view. Obviously this is a simplistic account of spatial relations (Herskovits 1986), but simplification is inevitable in a practical project such as this.

Providing synthetic vision based on rendered images carries with it the obligation to decide where to look. CDAs are capable of a simple form of active vision. An example of this is the tetris player described in (Wavish & Graham 1995), in which three collaborating CDAs control the viewing position, choose the best aiming point, and fit the falling block skilfully into position.

The actor agents are situated agents that are not capable of planning their actions in the classical AI sense. However in the context in which they are used, planning is unnecessary because activity is either scripted or else is routine.

Similarly, actors are not capable of natural language generation or understanding, but are restricted to performing predefined utterances (typically by playing prerecorded audio). These utterances can be explicitly scripted, or may be produced as a routine and role-specific accompaniment to improvised background activity.

4.4 Modelling the physical world

Objects in the physical world, including the bodies of the actors, are modelled by conventional 3-D graphics techniques, allowing scenes to be animated by directly updating data structures. For instance, the walking motion of an actor is created by updating the joint angles of the actor's skeleton (Magnenat Thalmann & Thalmann 1995).

Each object in the world is also modelled as a CDA in order to provide a uniform environment for the CDAs comprising the actor agents. It is natural therefore to exploit the capabilities of the CDA for representing properties of inanimate objects. For instance, a CDA can use a marker to represent the object that it is currently on, and apply a constraint between the two objects so that both move together. The grasping of an object by an actor can be represented in this way. When the actor lets go, the object can simulate falling under gravity by updating its own velocity vector. Similarly the level of water in a glass can be made to react to the drinking behaviour of an actor.

Markers can also be used to model ownership. Ownership is important at the scripting language level so that possessive pronouns can be used to refer to objects, and is an integral part of such everyday activities as drinking from one's own glass, or giving a present to someone. Each physical object therefore has a marker indicating who owns it, allowing actors (or reflexes in the script) to tell who an object belongs to just by looking at it. Obviously this is an unrealistic model of how ownership works in the real world, but it has the merit of being extremely simple and costing very little in the way of computational resources.

4.5 Implementations

So far we have constructed two implementations of subsets of the architecture described, and are currently building a complete implementation which is due to be finished by mid 1997.

The first implementation focused on two actors performing a script written in a simplified version of the scripting language. Actions included moving about the simulated three-dimensional space, picking objects up and putting them down, and giving and receiving.

The second demonstrator focused on improvising roles. It involves three actors playing the roles of drinkers and bartender in a bar. A simple form of active vision is implemented. The drinkers go to the bar when they notice that their drinks are finished and are served by the bartender.

The complete system that we are currently building includes all of the aspects of the scripting language and the actors that have been discussed. The setting is a bar, again, but this time there are four actors playing the different roles of wife, lover, husband and bartender in a scripted scenario.

The translation of the script into the runtime deictic representation is complete and working, and the framework of the actors is in place. Current efforts are directed at populating the set of roles and skills needed to perform the script and integrating the actor agents with the human figure animation software. We are also investigating the usefulness of bottom up invocation of CDAs by a spreading activation mechanism (Maes 1991, Rhodes 1995).

5. Conclusions

In this paper we have focused on the overall architecture of our system and have glossed over a number of aspects of the system which deserve to be treated in more depth. These include the design of the scripting language, the way that skill modules within an actor communicate and collaborate and how their activation

dynamics is controlled, and the design of the RTA programming language in which this system is implemented. We plan to address these topics in future publications.

We have introduced a number of features into our architecture which we believe to be particularly significant for future developments. These include the combination of scripted foreground behaviour with improvised background behaviour, actors that can perform multiple roles concurrently, the top-down design of simulated social activity based on roles, and the use of deixis as a means for situated agents to communicate with one another.

At the time of writing our system has not been completed, so its effectiveness cannot yet be assessed. A central issue is how believable the characters will be, and this raises the questions of how to assess and predict believability. These questions are being studied as another part of the Humanoid 2 ESPRIT project.

The system is designed to be extensible in order to improve believability and ease of use. New skills or roles can be added to the actors, and their behaviour can be refined by adding new rules to their existing CDAs. There are also possibilities for increasing the expressiveness of the scripting language, by encoding more sophisticated search strategies into the reflexes.

The shallowness of the actors and their dependence on scripts prohibits the use of the system as it stands for applications where actors are required to interact verbally with the user. However developments of this system could be considered in which the script is generated at runtime (Kelso et al. 1992) rather than being pre-written, in which case the ability of an actor to portray a character convincingly would depend strongly on the capabilities of the automated script writer.

The ultimate test of this work is the acceptance of it by users in commercial products and services such as computer games and shared virtual worlds. This work forms part of a programme of research aimed at creating new opportunities for products and services in the field of interactive entertainment.

Acknowledgements

This research was carried out as part of the ESPRIT project Humanoid 2 and was funded in part by the European Commission. Mike Graham and David Eves contributed to the preparation of this paper.

References

- Agre, P. & Chapman, D., 1987. 'PENGI: An implementation of a theory of activity', Proceedings of the Sixth National Conference on Artificial Intelligence AAAI-87, pages 268-272, Seattle, WA, 1987.
- Bates, J., Loyall, A.B. & Reilly, W.S., 1991. 'Broad Agents', SIGART Bulletin, Vol. 2, no. 4, August 1991.
- Brooks, R.A., 1986 'A Robust Layered Control System for a Mobile Robot', IEEE Journal of Robotics and Automation, RA-2, April 1986.
- Chapman, D., 1991. 'Vision, Instruction and Action', 1991, MIT Press, Cambridge, MA and London, England, 1991.
- Cremers, A, 1996. 'Reference to objects: an empirically based study of task-oriented dialogues', Proefschrift, The Technical University of Eindhoven, The Netherlands, 1996.
- Dennett, D.C., 1987. 'The Intentional Stance', Bradford Books, MIT Press, Cambridge MA, 1987.
- Graham, M. & Wavish, P.R., 1991. 'Simulating and Implementing Agents and Multi-Agent Systems', Proc. 1991 European Simulation Multi-Conference, Copenhagen, June 1991.
- Hayes-Roth, B. & Feigenbaum, E.A., 1994. 'Proposal for a

- Computer-Animated Improvisational Theater for Children', AAAI Workshop on AI and Entertainment, Seattle, 1994.
- Herskovits, A., 1986. 'Language and Spatial Cognition', in the series 'Studies in Natural Language Processing', Cambridge University Press, 1986.
- Kelso, M.T., Weyhrauch, P. & Bates, J., 1992. 'Dramatic Presence', Technical Report CMU-CS-92-195, School of Computer Science, CMU, Pittsburgh, PA, December 1992.
- Kiss, G, 1991 'Autonomous Agents, AI and Chaos Theory'. In 'From animals to animats', Proceedings of the First International Conference on Simulation of Adaptive Behavior, eds. Jean-Arcady Meyer and Stewart W. Wilson, Bradford Books, MIT Press, Cambridge MA, 1991.
- Laurel, B., 1991. 'Computers as Theatre', Addison-Wesley, 1991.
- Maes, P., 1991. 'A bottom-up mechanism for behavior selection in an artificial creature', in 'From animals to animats', Proceedings of the First International Conference on Simulation of Adaptive Behavior, eds. Jean-Arcady Meyer and Stewart W. Wilson, Bradford Books, MIT Press, Cambridge MA, 1991.
- Maes, P., 1993. 'Behavior-Based Artificial Intelligence', Proceedings of the Second Conference on Adaptive Behavior, MIT Press, 1993.
- Maes, P., Darrell, T., Blumberg, B. & Pentland, A., 1995. 'The ALIVE system: Full-body Interaction with Autonomous Agents', Proceedings of the Computer Animation '95 Conference, Geneva, Switzerland, IEEE Press, April 1995.
- Magenat Thalmann, N. & Thalmann, D., 1995. 'Digital Actors for Interactive Television', Proceedings of the IEEE, Vol. 83, no.7, July 1995.
- Minsky, M., 1986. 'The Society of Mind', Simon and Schuster, New York, 1986.
- Perlin, K., 1995. 'Real Time Responsive Animation with Personality' Trans. IEEE on Visualization and Computer Graphics, Vol. 1 no. 1, pp. 5-15, March, 1995.
- Renault, O., Magenat Thalmann, N. & Thalmann, D., 1990. 'A vision-based approach to behavioural animation', The Journal of Visualisation and Computer Animation 1 (1), pp 18-21, 1990.
- Rhodes, B., 1995. 'Pronomes in Behavior Nets', MIT Media Lab. Technical Report no. 95-01, January 1995.
- Rosenschein, S. & Kaelbling, L.P., 1986. 'The synthesis of digital machines with provable epistemic properties', in Proceedings of the 1986 Conference on Theoretical Aspects of reasoning about knowledge, pp. 83-98, ed. J.Y. Halpern, Morgan Kaufmann Publishers: San Mateo, CA, 1986.
- Suchman, L.A., 1987. 'Plans and situated actions: The problem of human-machine communication', Cambridge University Press, 1987.
- Wavish, P.R., 1990. 'Real Time ABLE' in 'Review of 1990', Philips Research Laboratories, Redhill.
- Wavish, P.R. & Graham, M., 1995. 'Roles, Skills and Behaviour: A Situated Action Approach to Organising Systems of Interacting Agents', in 'Intelligent Agents' ed. M.J. Wooldridge and N.R. Jennings, Springer-Verlag, 1995.
- Wavish, P.R. & Graham, M., 1996. 'Situated Action Approach to Implementing Characters in Computer Games', Applied A.I. Journal, January 1996.
- Wooldridge, M.J. & Jennings, N.R., 1995. 'Agent Theories, Architectures and Languages: A Survey' in 'Intelligent Agents' ed. M.J. Wooldridge and N.R. Jennings, Springer-Verlag, 1995.