Reactive Agents in Behavioral Animation

MÔNICA COSTA[1]
BRUNO FEIJÓ[1]
DANIEL SCHWABE[1]

[1]ICAD - Laboratório de CAD Inteligente
Departamento de Informática, PUC-Rio
Rua Marquês de São Vicente, 225
22453-900 Rio de Janeiro, RJ, Brasil
monica@icad.puc-rio.br

Abstract. This paper presents actors in behavioral animation as being reactive autonomous agents in a virtual environment. The principles underlying the proposed model are cognition, emergence, situatedness, recursion and cooperation. Also the model is based on a cognitive architecture where both controlled and automatic procedures coexist. An animated sequence of a navigation scene is generated by a prototype.

Introduction

In behavioral animation characters have personality and talent. In this case, the performance of a character emerges from its own beliefs, intentions, humor, fears, feelings and interactions with other actors. The animator becomes a director rather than a keyframe designer. Strictly speaking the animator becomes a meta-animator as pointed out by Craig Reynolds [Reynolds, 1987].

The notion of behavioral animation can be traced back to Zeltzer's provocative paper on knowledge-based animation [Zeltzer, 1983] and reconstructed from a number of concepts, such as: task-oriented animation [Zeltzer, 1985], distributed behavioral models [Reynolds, 1987], synthetic actors [Magnenat-Thalmann and Thalmann, 1991], stimulus-response models [Wilhelms, 1990], synthetic vision for behavioral actors [Renault et al., 1990], ecosystem simulation [Tu, 1994] and virtual creatures generated by genetic algorithms [Sims, 1994]. Generally speaking, the authors believe that behavioral animation shares its problems with those found in artificial intelligence, robotics and artificial life [Maes, 1990] [Levy, 1992] [Langton, 1994].

One of the basic assumptions in this paper is that performing before an audience is, in essence, a question of planning. However, traditional AI-based planning programs are not reactive systems and, consequently, cannot cope with the continuity, surprises, interactions and ongoing history of the real world. This paper presents an innovative architecture for behavioral animation systems based on reactive autonomous agents that perform in a virtual environment. Furthermore, the proposed architecture is inspired in the mental models found in the cognitive science, where both controlled and automatic procedures coexist. In this paper, automatic procedures represent reactive processes and break with AI traditional paradigms.

In this paper, agents represent any actor, such as characters, decorative objects, cameras, lights or even more abstract entities such as the story. The feasibility of the proposed architecture is illustrated by a prototype that is able of creating simple navigation scenes.

Agents

Agent technology has being applied in distributed AI [Bond and Gasser, 1988], in groupware [Baecker, 1993], in virtual environments [Bates et al, 1992] and in robotics [Brooks, 1990]. Also agent-oriented programming has been proposed as a post-object paradigm [Shoham, 1993]. Agent theory is not mature yet and leads to several definitions of agents and their properties. A complete survey on agent theories, architectures and languages can be found elsewhere [Wooldridge and Jennings, 1994].

In this work, the authors define agents as active objects described by the intentional stance. Indeed the notion of agency is bound to that of action. Therefore, agents are active objects, because they originate actions that affect their environment. This is an important aspect for behavioral animation, because actors are characters that act without the animator's intervention. To ascribe the intentional stance to agents means that they possess beliefs and desires. Also this is a significant aspect in dealing with behavioral animation, because the actions performed by the characters result from their intentions.

Intention can be formally defined in terms of non-classical logic, such as the multi-modal logic proposed by Cohen and Levesque for their rational agents [Cohen and Levesque, 1990]. However, this is not within the scope of the present work.

There are three approaches to build agent-based computer systems: deliberative, reactive and hybrid architectures. The deliberative architecture is based on the classical symbolic AI paradigm. Examples of this approach can be found in [Wood, 1993] and [Vere and Bickmore, 1990]. In this case, the symbolic model of the world is explicitly represented and the agents act via explicit logical reasoning. Usually, in this approach, an AI planning system is the central component of the agent. This architecture, however, has several drawbacks: (1) the frame problem renders the knowledge difficult to represent in practice; (2) it is computationally inefficient; (3) it cannot cope with unpredictable events such as the actions of other agents; (4) it always requires that plans be too detailed, although one generally acknowledges that no system could produce completely detailed plans in domains of realistic complexity [Agre and Chapman, 1989]. Therefore, alternative approaches to agent architectures have been proposed.

The reactive architecture is an alternative approach that breaks with the traditional symbolic AI paradigm. This sort of architecture is strongly advocated by Rodney Brooks who claims that intelligence can emerge without having explicit manipulable internal representation or explicit reasoning systems [Brooks, 1991]. This architecture is based on reactive agents that must respond dynamically to changes in their environment.

The hybrid architecture attempts to harmonize the classical architecture with the reactive approach [Arkin, 1990] [Georgeff, Lansky and Schoppers, 1987]. The present work proposes a hybrid architecture for autonomous agents in a virtual environment. However, the emphasis of this work is on the reactive side of the hybrid architecture.

The authors believe that a special agent language for behavioral animation should be developed. However, this is an issue not yet fully investigated by the authors.

The Proposed Agency Principles

The principles underlying the proposed reactive agent model are: cognition, emergence, situatedness, recursion and cooperation. Most of theses principles are inspired from Rodney Brooks' work [Brooks, 1991].

By adopting cognition as one of the cornerstones of the model, the authors state that any agent architecture should be influenced by human models of mind. This is important not only because some animation characters are human-like (with emotion, desire and beliefs), but also because this is a new tendency in computer architecture.

The principle of emergence states that the intelligence of the agent system emerges from the interaction of agents among themselves and with their environment [Steels, 1990]. As pointed by Rodney Brooks: "It is hard to identify the seat of intelligence within any system, as intelligence is produced by the interactions of many components. Intelligence can only be determined by the total behavior of the system and how that behavior appears in relation to the environment. The key idea from emergence is: *Intelligence is in the eye of the observer*" [Brooks, 1991, p.16]. This principle can also be identified in the work by Minsky where he proposes that intelligence emerges from a society of mindless agents [Minsky, 1987].

Situatedness is an idea proposed by Rodney Brooks [Brooks, 1991] who claims that the agent's intelligence is situated in the world and not in any formal model of the world built in the agent. Therefore, an agent uses its perception of the world rather than deductions based on a symbolic representation of this world (such as those found in theorem provers or expert systems). This is a dramatic change from the traditional AI paradigm. However, the authors believe that this change is of the utmost importance for behavioral animation. Maintaining the traditional AI paradigm means that behavioral actors will always have access to direct and perfect perceptions/actions and, consequently, no external world will really exist with its continuity, surprises and ongoing history.

Recursive structure and agent cooperation are principles attached to the object nature of the agents. In this view, an agent is built by sub-agents that have the same structure. Furthermore, agents work cooperatively distributing tasks, interchanging messages and sharing databases.

The Proposed Reactive Agent Architecture

In the proposed architecture, actors are reactive agents with the structure presented in Fig. 1. Actually, actors are motors which are themselves actors with the same structure. This focus on motors is the essence of animation, where characters are constantly performing actions and changing their environment.

The Sensory Centre has two kinds of basic functions: (1) functions to send and receive messages; (2) sensory perception functions. An agent is activated by a message sent by its parent-agent. Most of the time, this message is passed to the agents' motors by the cognition centre, in order to distribute tasks. A motor always reports success or failure to the agent that called it. This interchanging of messages is the basis for the cooperative work. The other kind of basic function, i.e. the sensory perception function, detects events in the virtual environment associated to vision, hearing and touch.

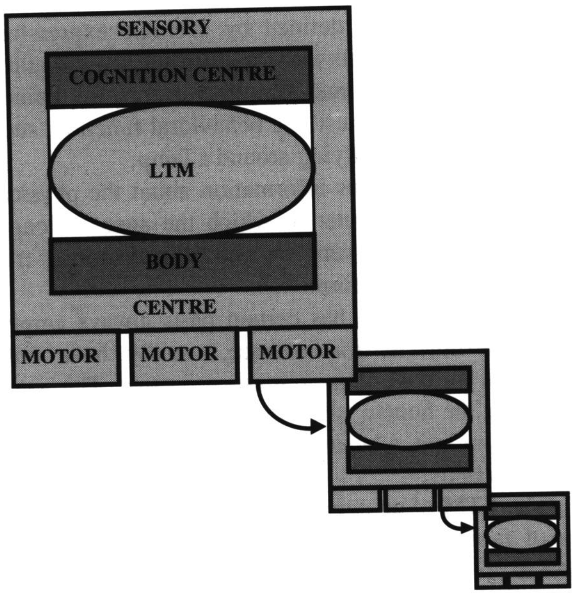

Fig.1 The actor structure

A simple example of a vision function for a navigation scene can be illustrated by Fig. 2, where **fdir** points to the direction of the movement and α is the vision angle. An object is detected if it enters the view volume and is not occluded by another object. A very simple detection method can be implemented by calculating the angle β of the vector **tobj** (obtained by the inner product). A more complex method is the intersection calculation between the view volume and the 3D object. Visibility also depends on the module of **tobj**. Stereoscopic vision is not required because the distances between objects can be calculated straightforwardly. Also no pattern recognition is required because the list of the objects is available to the character. These two latter facilities clearly illustrate the advantages of working with virtual characters instead of real robots.

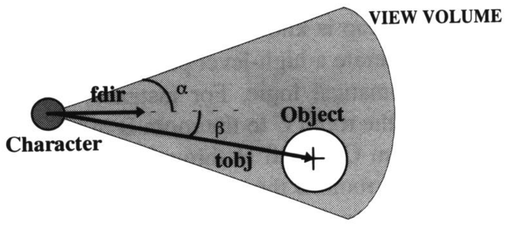

Fig.2 Simple vision function

The architecture for the Cognition Centre and the LTM (Large Term Memory) is inspired in the results of the cognitive science [Stillings et al, 1987]. The LTM is a declarative memory with facts specified by the animator and facts perceived by the character during its existence in the virtual environment. These facts are, however, inert structures that should be operated by processes in the Cognition Centre. Processes are procedural knowledge of two types: controlled and automatic procedures (Fig.3). Controlled procedures require conscious attention like an interpreter. Automatic procedures are like compiled programs automatically triggered by events or goals. The Logical Procedures in Fig.3 are sentences in mathematical logic. They are used in situations where deductive thought is required in specific domains. General path-planning with low degree of details is usually done by logical procedures. A previous work done by the first two authors [Feijó and Costa, 1993] is an example of how to use logic in behavioral animation.

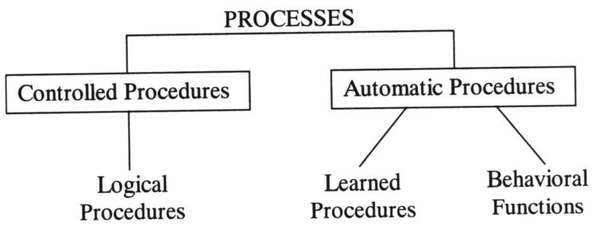

Fig.3 Processes in the Cognition Centre

Learned Procedures are reactive plans encoded as compiled programs. In this context, there is no distinction between planner and executor. These plans are continuously revised and, consequently, can adapt themselves to unexpected events. The name Learned Procedure comes from the fact that these procedures represent learned skills with no need for conscious attention.

In a navigation example, the learned procedure **MoveTo** is capable of taking a character from its position **fpos** to a specific location **tpos**, avoiding obstacles if any. Fig.4 shows a character making a detour to avoid an obstacle by first calling the sub-agent **FACE** to point the character to a new direction and then calling the sub-agent **MOVE** to move it along a straight line. This procedure requires recursive calls to **MoveTo**, because new obstacles can appear any time in the way. The vector **tpos** is in the LTM.

The piece of C code in Fig.5 illustrates the learned procedure **MoveTo**.

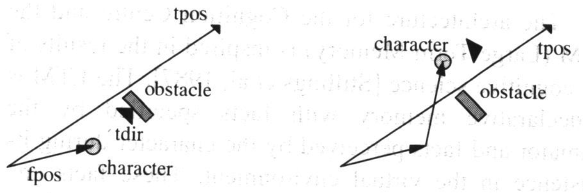

Fig.4 Character making a detour around an obstacle

```
answer_message_ MoveTo (move_to_ *move_to)
/* return if agent reaches the target                    */
{ if(fpos(move_to) == tpos(move_to))
  { answer_message.type = SUCCESS;
    return(answer_message);
  }
/* compute direction of the movement                     */
  tdir = tpos(move_to) - fpos(move_to);
/*  set up messages and call face                        */
  send_message.type = FACE;
  send_message.data = tdir;
  answer_message = *(answer_message(sensory_centre(
                        face(move_to)))) (face(move_to),
                        send_message);
/* in case of success set up messages and call move      */
  if(answer_message.type == SUCCESS)
  { send_message.type = MOVE;
    send_message.data = tpos(move_to);
    answer_message = *(answer_message(sensory_centre(
                        move(move_to)))) (move(move_to),
                        send_message);
/* in case of fiding obstacles, reports failure          */
    if(answer_message.type == FAIL)
    { aux_tpos = tpos(move_to)
      obj_pos = answer_message.data     /* obstacle pos */
      tpos(move_to) = calc_new_tpos(fpos(move_to),
                        obj_pos);
/* recursive call to MoveTo                              */
      answer_message = MoveTo(move_to);
      if(answer_message.type == SUCCESS)
      { tpos(move_to) = aux_tpos;
        answer_message = MoveTo(move_to)
      }
    }
  }
  return (answer_message);
}
```

Fig.5 Procedure MoveTo

Learned procedures are not based on traditional AI techniques. They are compiled programs that are very efficient. Also there is no need for explanation-based reasoning. Learned procedures implement the reactive behavior of the characters in the same spirit proposed by [Georgeff, Lansky and Schoppers, 1987] and [Brooks, 1991]. The agents are driven by their learned procedures and the intelligence they exhibit is a result of the interactions that occur within the virtual environment. The principles of emergence and situatedness are satisfied by the learned procedures.

Behavior functions are primitive forms of automatic procedures defined by a single expression. They are used by agents that should react in a stimulus-response basis. Sometimes simple creatures are defined by a single agent and just one behavioral function, such as a very small insect flying around a lamp.

The Body contains information about the physical structure of the character to which the agent belongs. Only the agents in the very low end of the hierarchy tree contains this sort of information.

Usually an agent has certain parts always empty. Decoration objects, for instance, usually have only bodies and no cognition or sensory centres. The agent controlling the human gait only has a sensory centre (to receive stimulus), a behavioral function (to perform the gait) and a body.

The LTM of an agent is in fact a window to a vast area of factual data, reminding to a certain extent the classical blackboard technique in AI [Hayes-Roth, 1985]. Sometimes there are facts that are common to more than one agent. Only a especial agent called the Universal Agent has the consciousness of the entire factual data base. The Universal agent is on the top of the hierarchy and is in contact with the user.

The reactive capacity of the virtual environment is defined by a time step called the system clock. After each clock the system has the opportunity of checking if any event happened and then it takes the necessary steps. Similarly to the case of time stepping in numerical integration, large clocks degenerate the reactive response of the system. Different agents with different accelerations have the same clock. In this case, the difference relies on the amount of displacement steps or rotation steps performed by the agent during the clock.

Example of Moving in a House

A character that is capable of moving itself in a house can be built by the agents presented in Fig.6.

The agent **go_to** moves the character around a house, once the map is known. This agent has a logical procedure to generate a high-level plan (with no details) based on mathematical logic. For instance, a possible plan to go from the room C to the room B in Fig.7 could be "go from room **C** to hall **H** through door **c** and then go from hall **H** to room **B** through door **b**".

The agent **move_to** takes the character to a specific position, avoiding collision with obstacles if necessary.

The agent **face** points the character to a new direction in order to avoid a collision. The agent **move** takes the character to a certain position along a straight line. This agent has a vision function in its sensory centre in order to detect obstacles.

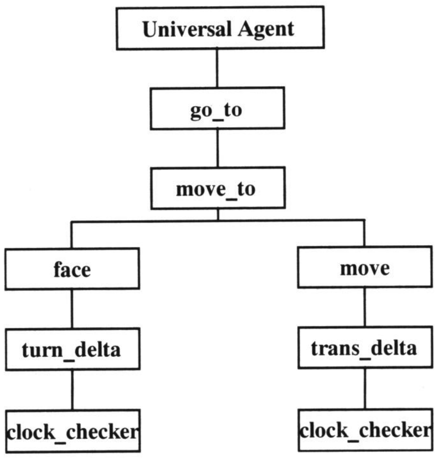

Fig.6 An agent hierarchy for navigation

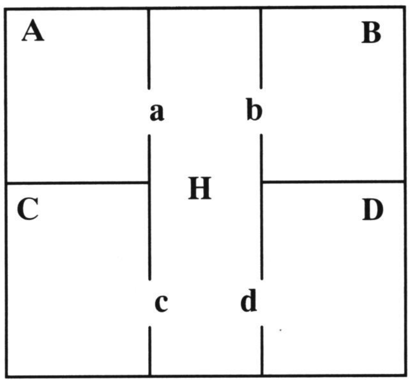

Fig.7 A simple map for navigation

The agents **turn_delta** and **trans_delta** perform a single rotation step and a single translation step during one system clock several times. Learned procedures determine the size of these primitive movements. In the example implemented by the authors, the size of the primitive movements is constant and arbitrary. After each clock, in a more complex case, these learned procedures could read the positions of their agents from the dynamic analysis performed by a physics-based model associated with the character's body.

After each clock, the agent **clock_checker** looks for any event occurrence in the virtual environment. This agent will stop and send messages back to its parent if an event threatens the plan. These messages go up in the hierarchy, from agent to agent, until the appropriate agent takes the necessary steps.

A prototype was built by the authors and the example of a character moving in a house was tested. Figs 8, 9 and 10 show a kid avoiding a toy resting on the floor and executing the plan of going to the farthest room in the house. The animated sequence was rendered using the script language available in 3D Studio.

Conclusions

The authors have demonstrated an innovative architecture for behavioral animation systems based on reactive autonomous agents living in a virtual environment. This component-oriented approach revealed an efficient and easy way of creating cooperative work among agents and satisfying the principles of emergence and situatedness.

However, the prototype should be improved towards more complex animation and actors. Also there are several theoretical topics for future work. The animator could use a library of agents to set up a scene or even still the entire story. He/she could also use the library to create a new character. These ambitious steps require, however, to solve problems of reusing components. Furthermore, an agent language should be developed.

A formal method to support the learned procedures is also required. In this context, the concept of procedural logic by [Georgeff et al., 1985] might be a good start. A more expressive LTM should also be investigated with levels of activation attached to factual data. Furthermore, an emotion generator should be incorporated to the cognition centre.

Acknowledgments

The authors would like to thank the CNPq for the financial support and Hélio Magalhães for the technical support in the rendering part.

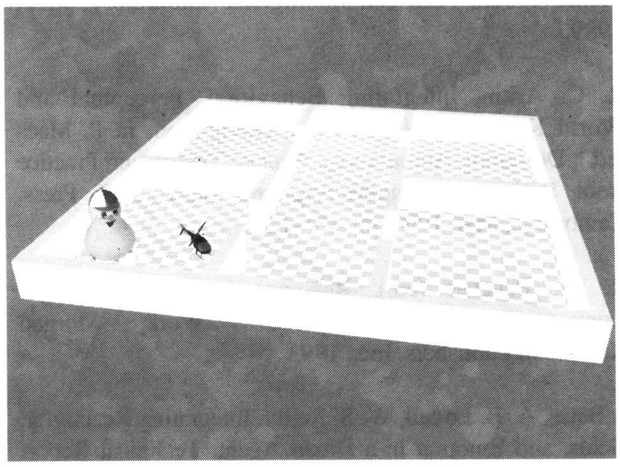

Fig.8

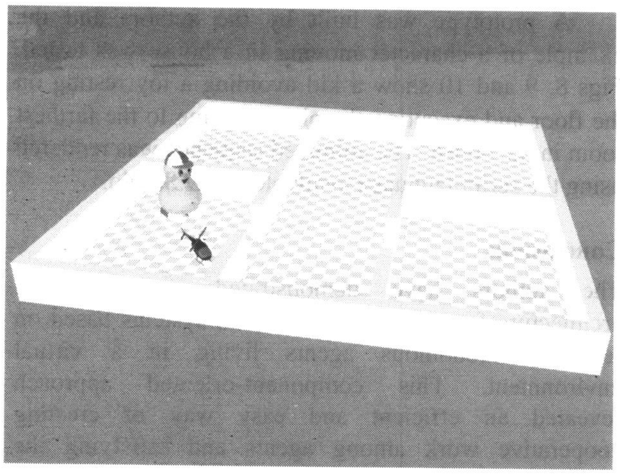

Fig.9

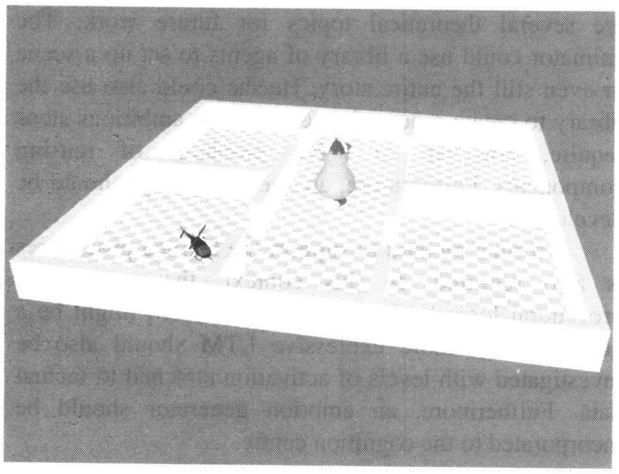

Fig.10

References

P. E. Agre, D. Chapman, What are plans for?, A.I. Memo 1050a, Artificial Intelligence Laboratory, MIT, 1989.

R. C. Arkin, Integrating Behavioral, Perceptual and World Knowledge in Reactive Navigation, in P. Maes (ed.), Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back, MIT Press, Cambridge, MA, 1990, pp. 105-122.

R. M. Baecker (ed.), Readings in Groupware and Computer-Supported Cooperative Work, Morgan Kaufmann Publishers, Inc., 1993.

J. Bates, A. B. Loyall, W. S. Reilly, Integrating Reactivity, Goals, and Emotion in a Broad Agent, Technical Report CMU-CS-92-142, School of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, 1992.

A. H. Bond, L. Gasser (eds.), Readings in Distributed Artificial Intelligence, Morgan Kaufmann Publishers, Inc., 1988.

R. A. Brooks, Elephants don't play chess, in P. Maes (ed.), Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back, MIT Press, Cambridge, MA, 1990, pp. 3-15.

R. A. Brooks, Intelligence Without Reason, A.I. Memo No. 1293, Artificial Intelligence Laboratory, MIT, 1991.

P. R. Cohen, H. J. Levesque, Intention is choice with commitment, Artificial Intelligence, **42**, 1990, pp. 213-261.

B. Feijó, M. Costa, Animação Comportamental Baseada em Lógica, Proc SIBGRAPI'93, 1993, pp. 117-122.

M. P. Georgeff, A. L. Lansky, P. Bessiere, A Procedural Logic, in Proc Ninth Internatinal Joint Conference on Artificial Intelligence, Los Angeles, CA, 1985.

M. P. Georgeff, A. L. Lansky, M. J. Schoppers, Reasoning and Planning in Dynamic Domains: An Experiment with a Mobile Robot, Technical Report 380, Artificial Intelligence Center, SRI International, Menlo Park, CA, 1987.

B. Hayes-Roth, A blackboard architecture for control, Artificial Intelligence, **26**(3), 1985, pp. 251-321.

C. Langton (ed.), Artificial Life III, Addison-Wesley, 1994.

S. Levy, Artificial Life, Vintage Books, NY, 1992.

P. Maes (ed.), Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back, MIT Press, Cambridge, MA, 1990.

N. Magnenat-Thalmann, D. Thalmann, Complex Models for Animating Synthetic Actors, IEEE Computer Graphics and Applications, 1991, pp. 32-44.

M. Minsky, The Society of Mind, Pan Books, London, 1987.

O. Renault, N. Magnenat-Thalmann, D. Thalmann, A vision-based approach to behavioral animation, The Journal of Visualization and Computer Animation, **1**(1), 1990, pp. 18-21.

C. Reynolds, Flocks, Herds and Schools: A Distributed Behavioral Model, Proc SIGGRAPH'87, Computer Graphics, **21**(4), 1987, pp. 25-34.

Y. Shoham, Agent-oriented programming, Artificial Intelligence, **60**(1), 1993, pp. 51-92.

K. Sims, Evolving Virtual Creatures, Proc SIGGRAPH'94, Computer Graphics, 1994, pp. 14-21.

L. Steels, Towards a Theory of Emergent Functionality, in Proc. First Int. Conf. on Simulation of Adaptive Behavior, MIT Press, Cambridge, MA, 1990, pp. 451-461.

N. A. Stillings, M. H. Feinstein, J. L. Garfield, E. L. Rissland, D. A. Rosenbaum, S. E. Weisler, L. Baker-Ward, Cognitive Science: an Introduction, MIT Press, Cambridge, MA, 1987.

X. Tu, D. Terzopoulos, Artificial fishes: Physics, Locomotion, Perception, Behavior, Proc SIGGRAPH'94, Computer Graphics, 1994, pp. 43-50.

S. Vere, T. Bickmore, A basic agent, Computational Intelligence, **6**, 1990, pp. 41-60.

J. Wilhelms, R. Skinner, A "Notion" for Interactive Behavioral Animation Control, IEEE Computer Graphics and Applications, 1990, pp. 14-12.

S. Wood, Planning and Decision Making an Dynamic Domains, Ellis Horwood Ltd., 1993.

M. J. Wooldridge, N. R. Jennings, Agent Theories, Architectrures, and Languages: A Survey, Proc ECAI94 Workshop on Agent Theories, Architectures and Languages, Amsterdam, The Netherlands, 1994, pp. 1-32.

D. Zeltzer, Knowledge-based Animation, Proc SIGGRAPH/SIGART Workshop on Motion, 1983, pp. 187-192.

D. Zeltzer, Towards an integrated view of 3D computer animation, The Visual Computer, **1**(4), 1985, pp. 249-259.