Chapter 7

The Message Concept

7.1 Introduction

In the animated world environment the agent develops autonomous behaviour through the use of instructions. The instruction concept presents a degree of adaptability by "sensing" the environment and developing a compatible sequence of actions to be performed by the agent. Such a feature in itself is very adequate for developing an autonomous behaviour, however, it is not sufficient to cover a wider range of behaviours that the agents have the potential to perform. A large class of behaviours to explore are those that require an agent to interact with another agent, for example, an *customer* agent ordering a drink from a barman or calling a waiter for service. A natural way to make such interactions possible is to provide the agents with capabilities of exchanging messages.

The idea behind the message feature implemented in this work is fairly simple, however, it provides an extra dimension to the animation in which the agents are not necessarily restricted to individual activities. Through the mechanism of the message an agent can recognise others intelligent entities with the capability of co-operating. Its use permits the animator to extend the capability of the instructions by including the message concept as part of an agent's plans thereby including other agents as part of its resources. Furthermore, the animator can establish a kind of "division of work" in the animation environment where agents are assigned to different behaviours and thus they may perform specialised activities.

7.2 A Brief Background

The use of the message as an interaction mechanism between intelligent agents as described above is quite unusual in the related areas of computer animation and robotics. In order to situate our approach to the message, we firstly explore the extent to which interactive capabilities of agents have been used in computer animation. Secondly, in the field of robotics and AI we examine, though very superficially, some aspects of the co-ordination of autonomous agents involving exchange of messages.

7.2.1 Applications in Computer Animation

In Chapter 2 we have reviewed work on computer animation. Among those systems which were identified as behavioural animation none has exhibited any degree of communication. One aspect resembling some form of communication is the stimulus-response between an agent and other agents or an agent and the environment in general. The most that has been reported is the reaction of agents towards somebody else's presence. That is, the agents were somehow capable of perceiving patterns of stimuli or that the thresholds limits have been passed, so that the corresponding responses were triggered. Therefore the behaviour is in response to stimuli rather than through communication between individuals. For example, in Sun's ball room [Sun93] and Tu's artificial fishes [Tu94] actions were executed individually though simultaneously causing an effect of synchronisation.

In the specific case of Reynolds' *boids* (a kind of bird agents) [Reyn87] the communication between agents does occur through the use of messages. His approach to implement the agents' behaviour is similar to Object-Oriented Programming where each agent is an object with encapsulated behaviours. These behaviours are implemented by methods, that is, program procedures. The method implementing a behaviour may differ from one object to another. However, objects having similar behaviours can be activated by similar messages sent to them. Therefore, an agent that wishes to affect the behaviour of another can do so by sending a message, or equivalently, by invoking the method associated with that behaviour in the object. This message sending is in fact a procedure call from one object to another, affecting the receiver's behaviour.

7.2.2 Robotics

The development of robots exhibiting a varying degree of autonomy has been established as a long term goal in robotics research [Ande88]. Such kinds of robots are intended to operate on specific tasks and in environments in which human access is difficult. In this kind of environment autonomy of the robots is an important factor for achieving the goal with a minimum of human intervention. The robots operate through the use of sensing mechanisms and some visual capabilities. Their behaviour can also be guided by an operator, but no mechanism of communication between robots has been reported.

7.2.3 AI

A recent field in AI called Distributed AI (DAI) is concerned with co-operative solution of problems by a decentralised group of agents [Huhn87]. As these agents are distinct entities, commonly geographically distributed computational entities, communication becomes an essential factor in co-ordinating their efforts.

Martial [Mart92] has presented interesting research in DAI which explores the co-operative (or coordinated) work between intelligent agents. These agents develop their own plans of activities which can be restructured to avoid conflicts (i.e., conflicts of resources or incompatible actions) with other agents or also to include beneficial relationships (favours) among them. Parts of the agents' plans can be coordinated by negotiating their tasks and modifying plans which may require the anticipation, postponing, spreading, or reducing the task execution time. Such a negotiation may require a lengthy process of communication until the involved parts have their plans coordinated. Although his work gives some insights on communication and favour relationship, it is still too complex for the purpose of animating multiple agents.

7.3 The Message Exchange Scheme

The capability of an agent to request an action from another agent is made through the use of the message exchange scheme. The message is an entity that behaves like a carrier that embeds an instruction. Such an instruction, which is called a *requested instruction,* thus specifies the desired action which the target agent is expected to perform. The target agent is scheduled with the *requested instruction* for evaluation and, if possible, its fulfilment. In Figure 7-1 this idea is illustrated in a very simplistic manner. On the left side of the equivalence sign, a full plan is depicted as a process that is developed and executed entirely by the agent A. On the right side of the equivalence sign, the same plan is shown as equivalent to two partial plans. Thus, the overall process can be understood as being undertaken by two distinct agents where the main partial plan, which commands the major developments of the process, is performed by the agent A which is the *requester*. While the minor partial plan, which represents the *requested instruction*, is developed and executed by the agent B which is the *receiver* of the message. When the *requested instruction* is completed then the control is resumed in the major partial plan developing the rest of the agent A's plan.

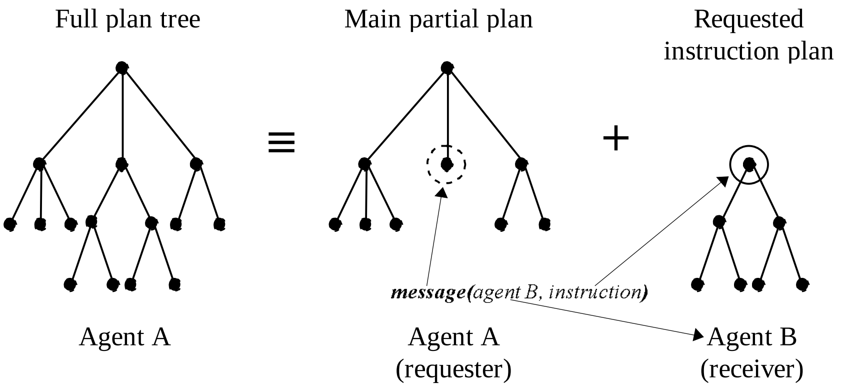

Figure 7-1: Equivalence of plan to two partial plans with message.

There is, however, a counterpart to the *request message,* that is the *acknowledgement message* which is sent in the opposite direction, that is, from the *receiver* to the original *sender* of the message. In both cases instructions are embedded in the message and sent to their correspondent receivers, though with some differences to be discussed. The *request message* is a request for action where the requesting agent sends the instruction to the *receiver* to perform it. The *receiver* in turn attempts to undertake that instruction and at the conclusion it is expected to communicate the result of the requested task by sending an *acknowledgement message.* This message comprises the information (success or failure) as an essential part and a gesture (or signal) as an optional part. Additionally, the *requester* may optionally send an acknowledgement back in response to the *receiver*'s acknowledgement without including any vital information. In fact any gestural acknowledgement, called signal, is optional and the choice of a gesture is basically determined by the availability of resources[3]. Gestures are aimed only at giving a "visual effect" to the viewer. The proposed message scheme is not concerned at establishing a kind of "verbal conversation" between agents and such display of intelligence is still beyond the reach of cognitive science. Figure 7-2 presents a schematic view of this mechanism.

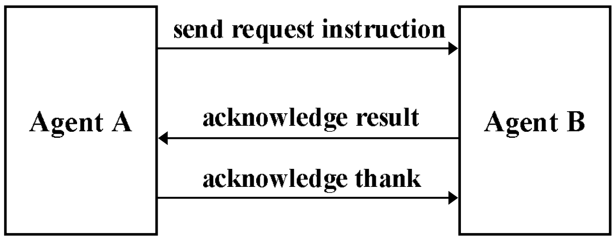

Figure 7-2: General message exchange scheme.

7.4 Operation of the Message Mechanism

The message entity is the component of the problem solving scheme with the special purpose of communicating, or instigating, actions between agents. Because these entities are part of the planning process and they connect the related actions, the original process and the instigated action, they are also parameterised entities. That is, it has at least one parameter which is the *receiver* of the message. Figure 7-3 presents the frames of the two existing types of messages: the *request message* and the *acknowledgement message*.

[3] Issue on resources is discussed in the next chapters.

```
frame msg_req;
  default state       is start and
  default name        is request and
  default type        is message and
  default request     is instruction and
  default reply       is reply_success and
  default reply_success  is thank_i  and
  default reply_failure   is never_mind_i and
  default root        is nothing and
  default template   is {receiver} and
  default receiver    is person .

frame msg_ack ;
  default state        is start and
  default name         is acknowledgment and
  default type         is message and
  default message   is ok and
  default signal       is signal_done_i and
  default root         is nothing and
  default template   is {receiver} and
  default receiver    is person .
```

Figure 7-3: Message frame.

The Message KS, shown in Figure 7-4, is invoked to deal with both types of message nodes found during the traversal of a planning tree. Basically this control is concerned with the initialisation and the finalisation of message instances. In the case of the initialisation of a *request message*, there is an embedded instruction in the *request* slot which is called *request instruction*. The *request instruction* is initialised as a new process carrying information about the involved agents and it is then scheduled to execute (the scheduling scheme is presented in Chapter 9). The initialisation stage includes operations such as the instantiation of the instruction, the initialisation of its parameters through the binding of information from the message instance, and the instantiation of a new root to identify the new process. Once the initialisation operations have been performed, the original process, which is represented by *Root A*, is momentarily suspended until the newly instigated process (*Root B*) has executed and returns the result of its action. The *request message* node thus has its focus of attention restored and, depending of the incoming result, its control may go to the *Stop* state or to the *End* state. In the case of the *Stop* state, it is understood that a problem has occurred during the execution of the *requested instruction* (*Root B*). In the case of the *End* state, the *requested instruction* has been fulfilled. In both cases a gesture expression from the agent A, in the form of signal, is scheduled by the Message KS to execute. That is, the agent A will acknowledge the result appropriately using the gestural instructions given in the slots *reply_success* and *reply_failure*. Actually such acknowledgements are not part of the process A's plan but help to complement behaviour.

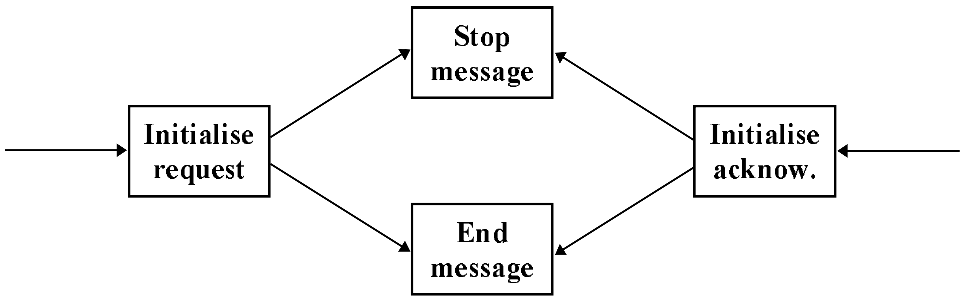

Figure 7-4: Message control.

In contrast, the *requested* instruction, here identified by *Root B*, has its execution carried out normally as any other instruction. Eventually the execution of the instruction might encounter an *acknowledgement message* node as part of the *requested instruction* plan, as shown in the lower part of Figure 7-5. The operation of the *acknowledgement message* is similar to the *request message*, but simpler. This message has the purpose of instigating a simple and fast instruction such as that which makes the agent A look at the agent B. Apparently this is a "weak request for attention" which does not demand for a result. In such a circumstance the requested activity effected by the agent B are considered complete and he carries on with other activities immediately.

In terms of process control, it can be observed that the *requested instruction* instance returns both the result of its operation and the focus of attention back to the main instruction plan, in the same way that an instruction instance would do to its parent instance at the conclusion of its partial plan. Therefore, the overall process proceeds as if it were a single process.

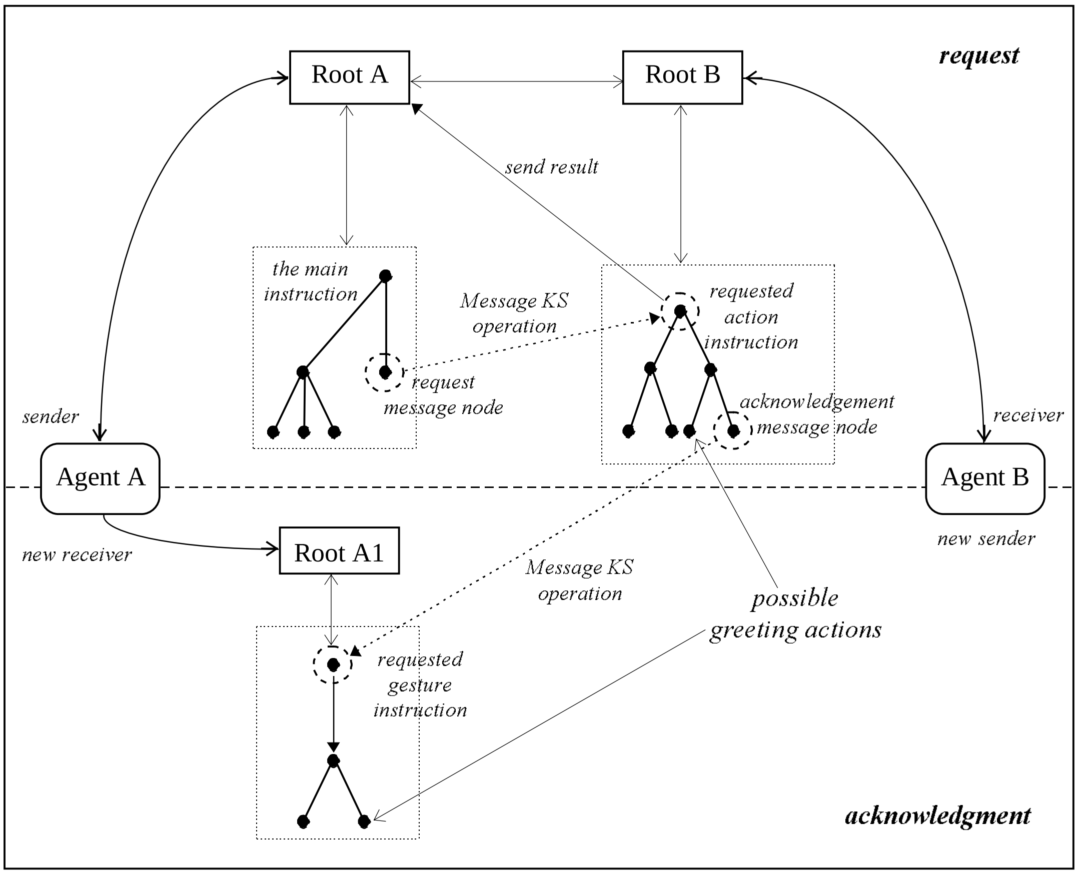

Figure 7-5: Message communication process.

7.5 Chaining Two Messages

As we have seen, the use of the message concept in planning permits a co-ordinated action involving two agents with features sufficient to cover most of the kinds of behaviour. However, there are situations that might happen involving a third agent. Such cases of chaining multiple plans work in the same way as the single message scheme. For example, suppose that in our bar scenario a *customer* agent, John, wishes to have a drink. His initial plan is to approach the *bar_counter* and to order a drink from the *barman* agent. The *barman* attempts to deliver a drink to the *customer*, provided he has some in his reach. In the case that no drink is available he has to order from the *supplier*. Upon receiving the request, the *supplier* in his turn will bring some drinks to the *exchange_counter* which is used for services internal to the bar. Then the *supplier* sends his acknowledgement of the delivery to the *barman* and carries on with his normal activities. Finally the barman reaches the drink in the *exchange_counter* and delivers the drink to the customer at the *bar_counter*. The description of this chain of actions is given in Figure 7-6. Obviously it depicts only the relevant actions related to John's action of having a drink. Other activities that John might be doing while he is waiting for the drink or other activities the barman might be doing while his request is being serviced are not included. Discussions of simultaneous actions are given in Chapter 9.

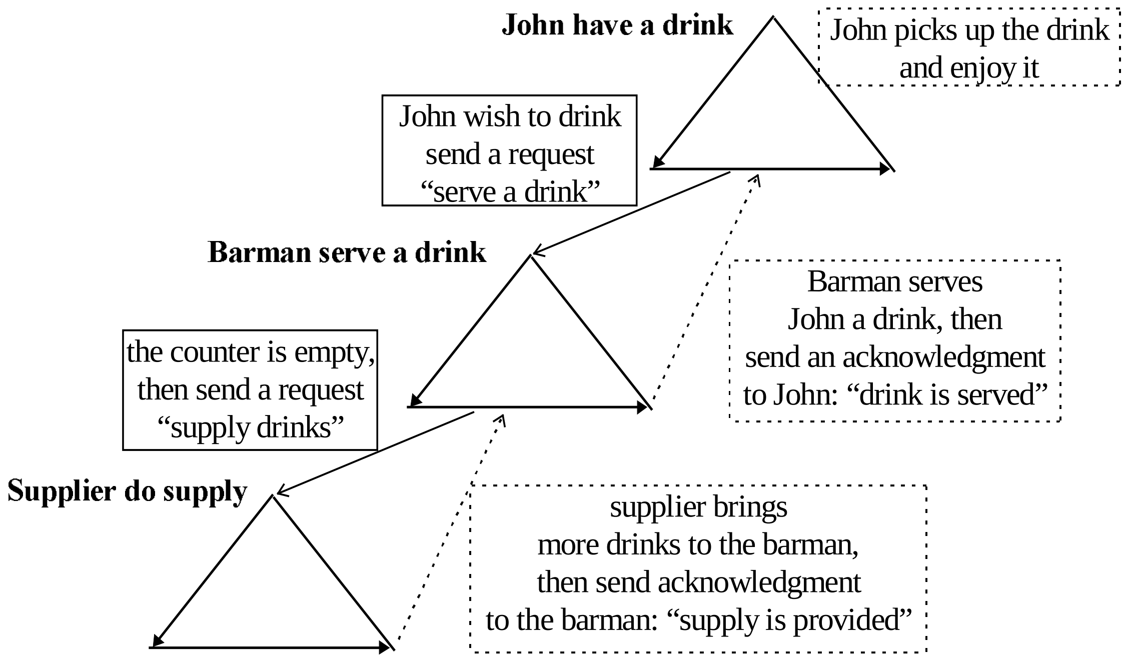
John have a drink
John picks up the drink and enjoy it
John wish to drink send a request "serve a drink"
Barman serve a drink
Barman serves John a drink, then send an acknowledgment to John: "drink is served"
the counter is empty, then send a request "supply drinks"
Supplier do supply
supplier brings more drinks to the barman, then send acknowledgment to the barman: "supply is provided"

Figure 7-6: Multiple message exchange.

7.6 Summary

A basic structure for controlling message exchange between agents has been presented. The inclusion of a message node as part of the instructions' plans permits the activities of two distinct agents to be connected as a co-ordinated action. That is, use of messages permit the realisation of a co-operative work between agents. The scheme is simple and feasible, but it is quite effective for animation purposes.

CHAPTER 7 THE MESSAGE CONCEPT **116**

7.1 INTRODUCTION 116

7.2 A BRIEF BACKGROUND 117

7.2.1 Applications in Computer Animation *117*

7.2.2 Robotics *118*

7.2.3 AI *118*

7.3 THE MESSAGE EXCHANGE SCHEME 118

7.4 OPERATION OF THE MESSAGE MECHANISM 120

7.5 CHAINING TWO MESSAGES 123

7.6 SUMMARY 124

FIGURE 7-1: EQUIVALENCE OF PLAN TO TWO PARTIAL PLANS WITH MESSAGE. 119

FIGURE 7-2: GENERAL MESSAGE EXCHANGE SCHEME. 120

FIGURE 7-3: MESSAGE FRAME. 121

FIGURE 7-4: MESSAGE CONTROL. 122

FIGURE 7-5: MESSAGE COMMUNICATION PROCESS. 123

FIGURE 7-6: MULTIPLE MESSAGE EXCHANGE. 124