Chapter 5

The Animation Framework

5.1 Introduction

It was observed in Chapter 2 that recent developments in computer animation systems have been towards motion autonomy. The main attention has been focused on mechanisms that enable the figures to realise motions with minimum assistance from the human animator. In extreme cases, some systems tend to simulate natural behaviour rather than perform directed animation. This is the case with Tu's *artificial fishes* [Tu94] in which the sensor inputs of the fishes trigger a repertoire of behaviour. However, there are other systems in which the animated entities do not act solely on their sensor inputs and are required to achieve some goals defined by an animator. Goal seeking has been advocated in earlier systems to perform high-level tasks [Badl85b, Zelt85].

In our work the focus is on providing a mechanism whereby an animator can record knowledge about skills and enable the figures to fetch this knowledge as conditions require. This is thus the application of a knowledge-based system for motion control. The experiences reported by Ridsdale [Rids87] and Calvert [Calv91] have pointed out that the application of typical expert system techniques is not suitable for multiple moving figures. Later, Calvert [Calv94] suggested the blackboard model as an appropriate adequate AI tool and describes a blackboard system to control the behaviour of multiple entities.

In the current work a blackboard approach to motion control is presented. The motions of the figures can be seen as processes that are created, suspended, interrupted, and

completed asynchronously. As the blackboard model deals with problems typical of an environment with multiple processes, it was adopted as a problem-solving paradigm in this framework. The framework is thus intended to combine the requirements of an entity to exercise intelligent behaviour such as reacting to events in an environment, to interacting with other entities, and being able to control its own motion.

In the first part of this chapter we present an overview of the implemented system and the details will be examined in the subsequent chapters. In the second part of the chapter we describe the entities handled by the animation system, namely, static and dynamic figures.

5.2 The Animation System

The animation system presented here satisfies two requirements. Firstly, it is required to endow the figures with basic motive skills, such as walking, as described in Chapter 4. Secondly, it should implement mechanisms that permit the agents to exhibit autonomous behaviour, that is, to provide a mechanism for automatic planning of their actions and interaction with the environment. A proposed system satisfying both requirements is composed of two modules as shown in Figure 5-1.

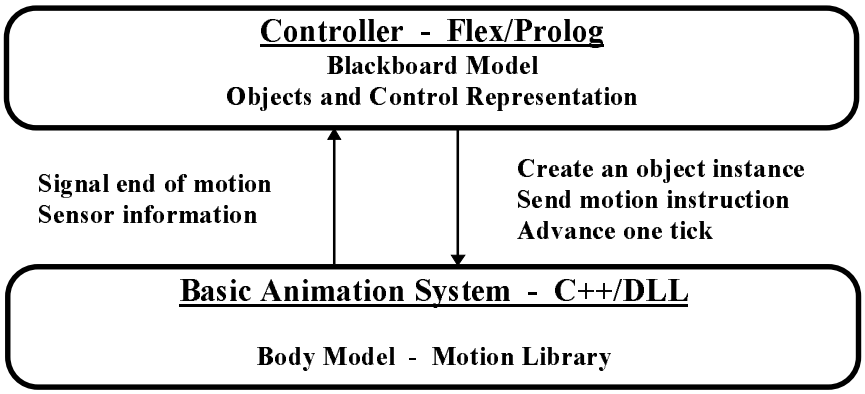


Figure 5-1: Animation framework composed by two interacting blocks.

The controller resides at the higher level and implements the blackboard model. It manages multiple process threads and reacts to changes in the environment. The underlying language is Prolog with Flex providing expert system facilities to represent frames, rules, and explicit knowledge representation of the reasoning mechanisms

[Vase92, Ring88]. This component of the animation system deals with symbolic information controlling the agents' behaviour. It is called the *animation controller*, or simply *controller*.

In the lower component there are the facilities of a typical animation system such as the camera, objects and their motions, and also "sensing" functions such as *the ability of an arm to reach an object*. This part is written in the C++ language and compiled into Windows DLL. The communication between this component and the higher component is through messages. There is a master/slave relationship, with the controller sending messages to the lower component to create objects, to perform motions, to advance one frame of the animation sequence, and to enquire about information of the animated objects. The lower component returns information to the controller about the state of the objects and transmits signals indicating that motions have finished.

5.3 The Animation Controller

As outlined in the chapter 3, the blackboard model has three main components: the blackboard data structure (or simply the blackboard), the knowledge sources and the control. The precise implementation of these components depends on the requirements of the problem and how it is to be solved. Some of the interesting features of motion control in animation systems are:

- multiplicity of objects that may have similar, or different, behaviour;
- agents that perform actions in response to different sources of stimuli in the environment;
- the execution of actions that cause changes in the environment which can be perceived by the agents;
- the course of events in the animation that can be influenced by the animator, either directly, by using scripted directions; or indirectly, by modifying the layout of the environment, by re-positioning the static objects, or by modifying the initial attributes or facts of the animated objects;
- the motion of human-like figures that should exhibit some degree of realism characteristic of human behaviour, with the possibility of including secondary movements such as swinging the arms while walking;

- the actions of an animated entity may be purposeful in nature, being goal-oriented and planned according to the agent's specific situation;
- when a figure has several potential actions to perform, a criteria scheme should allow the choice of the most appropriate or an applicable one.

Given these features, the animation activities can be organised into four co-ordinated types of control in the form of levels of abstraction:

- *scheduling*. Actions are scheduled or interrupted as the context requires.
- *instruction*. The organisation of actions as a control structure permits the planning and execution of activities in the pursuit of goals.
- *task*. The execution of primitive motions which require the allocation of resources.
- *message*. Animated figures can communicate by exchanging messages.

Each one of the activities above is handled by separate sets of knowledge called *knowledge sources* (KSs), namely, the Scheduling KS, the Instruction KS, the Task KS, and Message KS. This leads to an efficient overall control structure with a representation of the knowledge which aids understanding. Figure 5-2 describes the general organisation of the *controller*. Arrows within the blackboard indicate the transition of control from one problem solving stage to another.

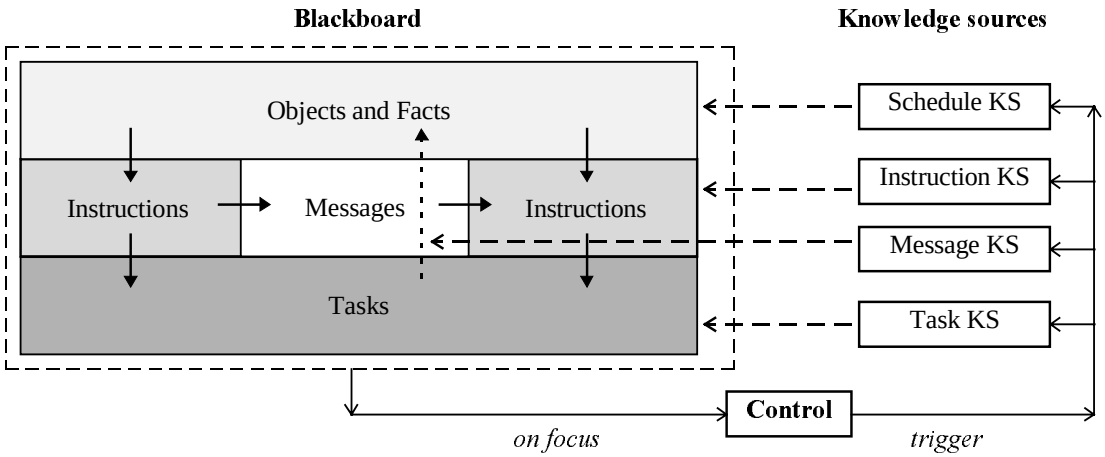


Figure 5-2: Organisation of the animation controller.

The several KSs will be discussed in more detail in the following chapters. The blackboard data structure reflects the operation of the KSs and, consequently, this will be described along with the correspondent KSs. In the following sections the blackboard

control and blackboard data are presented with a brief overview of the KSs. Furthermore, the controller handles the data representation of the animated objects which will be described in this chapter.

5.3.1 The Blackboard Data Structure

The blackboard data structure is a workspace that holds a variety of data instances such as the animated entities and the process control structures. These data are stored and manipulated by the knowledge sources in the pursuit of a solution. A *process* is created to solve a motion control problem and its data structure is implicitly developed into a hierarchy as shown in Figure 5-2 and Figure 5-3. That is, the analysis of a solution evolves dynamically from one level to another in the hierarchy.

As the animated environment contains a multiplicity of animated agents, there are at least as many processes being pursued simultaneously in the blackboard data structure, each of which being in a different state of analysis and having its solution tree spanning across the blackboard levels. Additionally, the animation entities have internal information, such as posture and location, which are also taken into account in the solution. The representation of the animation entities will presented in this chapter.

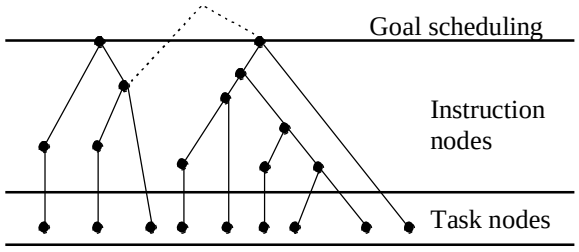


Figure 5-3: The three control stages of process.

5.3.2 The Knowledge Sources

Because the problem solving is partitioned into solution spaces, each partition has a corresponding KS that acts as a expert in operating on that specific type of data or activity. The four KSs are briefly described as follows.

The Scheduling KS undertakes the job of assigning agents with actions. Some actions are scripted by the animator and others result from the natural developments of an animation. The Scheduling KS has also the important role of keeping the agents busy

when no relevant activity is being performed. For example, it is better to have agents in the background perform activities appropriate to their role rather than staying still.

The Instruction KS monitors the progress of the actions being carried out by the animated agents. As discussed previously, an action is an activity that can be of a varying degree of complexity. The control of this complexity is represented by the concept of *instruction*. The term instruction is very convenient in the sense that it encloses within a computational entity activities related to a verb of action**.** For example, the activity of *picking up of an object* is achieved by the action verb *pick up*. As will be discussed in more detail in the next chapter, an instruction is a verb of action that can be represented by sequences of other instructions. All the information related to an instruction is contained in a frame data structure. The Instruction KS will thus develop the agents instructions and store results of its operation in the corresponding structure. That is, it plans how the instructions will be carried out, decomposing the instructions into less abstract instructions that accomplish the goal, recover from failures, etc. An instruction is developed as a tree where the terminal nodes, or *tasks nodes*, control the physical motion of the agent.

The *task* nodes are handled by the Task KS. The Task KS monitors a specialised skill which is also defined by a frame data structure. Each task acts as an interface between the Controller and the basic animation system. It allocates the necessary resources to that particular task and sends simple motion commands to the animation system. When the motion is completed the task decides the next step of the motion.

The Message KS is a special type of KS that deals with the communication between agents. In fact it behaves as a bridge forming an intermediary between two processes in a co-operative action.

5.3.3 Control

The role of the control component is to identify patterns of data that require the attention of one of the existing KSs. As the processes develop, items of data are added to the blackboard for further development. These items remain there as long as the processes are active. However, only one part of each process is considered at a time by the

control, this part having the *focus of attention*. The remaining parts are put into the "out of focus" group.

The control component of the animation controller is basically a loop that continuously checks for processes requiring the focus of attention. The control does not need to be complex because all the necessary processing is performed within the knowledge sources. A specific knowledge source is activated when at least one item of data of that type of source is in focus and remains active until none of that type of data is requesting attention. That is, the data of a region of the blackboard are processed in their entirety in one visit. However, the action of a knowledge source on the processes causes modification in the blackboard which may cause other types of data to request attention from the knowledge sources. When a region has been completely served, the control re-examines the hierarchy from the top region (scheduling) down to the lowest region (the task) to determine if any attention is required. The strategy is thus to bring the operation of all existing processes simultaneously to the task level.

When all the processes requiring attention have been served, that is, no more processes on the blackboard are on focus, the controller sends a message to the animation system to advance the animation frame by one step. That means all the motions scheduled in the animation system advance a little further. When the requested motions have been completed a signal is sent back to the corresponding task control node. This task node is thus "re-activated" or put "on focus" so that it can receive the attention of the Task KS to proceed with further operations or to pass the control to the *instruction* instances which controls it. To summarise, between one animation frame and the next, the control evaluates the solution state of the current actions.

5.4 The Example of Animation Scenario: Bar

In order to explore the performance of the system, a relatively complex scenario of a bar has been created with the following characters: *barman*, *waiter*, *supplier*, and *customers*. The scenario also includes some static objects such as *chair*, *glass*, *counter*, *shelf*. There are also two rooms or areas of action: the *bar area* and the *restaurant area* (Figure 5-4). In the *bar area* there are several counters with different functions: the *bar counter* is where the *barman* waits upon *customers*; the *exchange counter* where the

barman puts used glasses for collection and also receives provision of new glasses; the *supply counter* with a stock of new glasses; and a *waste counter* where used glasses are placed. In the *restaurant area* there are two tables with corresponding chairs. The *bar counter* is a special one where up to four potential places are available for *customers*. Obviously only one person at a time can occupy one of these places or one of the chairs. There are also some special areas known as the *door area* where the characters appear and disappear in the scenario.

The person is a generic character which provides the basis for any of the characters of the scenario. It is capable of walking, sitting, waving to somebody, picking up at most two glasses at a time, looking around at somebody, etc.

The barman has the role of serving customers at the *bar counter*. When he is not busy he removes the used glasses, if any, from the *bar counter* to the *exchange counter*. The provision of a new set of full glasses is also made through the *exchange counter*. If the *barman* realises that a new provision of glasses is required, he makes a corresponding request to the *supplier*. If a *customer* orders a drink, the *barman* searches the *bar counter* first and if none is found he seeks one from the *exchange counter* as a second alternative. If both fail he makes a request to the *supplier*.

The supplier supplies the *barman* with fresh glasses at the *exchange counter*. However, if the *supplier* is idle he might check whether new glasses should be supplied and also removes any used ones.

The waiter serves *customers* in the *restaurant area*. If a *customer* calls a the *waiter*, the *waiter* goes to the *customer* and receives an order to bring one or two glasses from the *exchange counter*. The *waiter* searches for new glasses at the *exchange counter* or, if necessary, at the *supply counter*. When the waiter is idle it collects glasses that are no longer used and puts them on the *bar counter* or on *exchange counter*.

The customers are the main characters of the scenario. They go to the bar to take a drink or if there are no places at the counter they go to a free table. Otherwise, they just wander around. At the table they order a drink from the *waiter*. They enjoy drinking and between sips they look at the people around them.

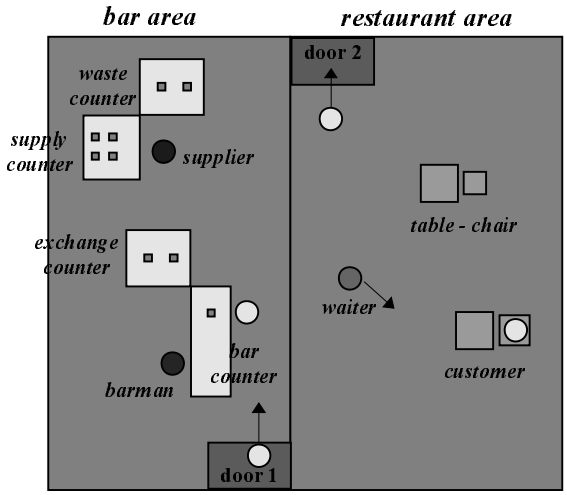


Figure 5-4: A view of the bar scenario from the top.

5.5 The Animation Entities

In Chapter 4 the animation entities were shown to be high-level computing objects encompassing both the geometric model and motional skills. Now, in the context of the *controller,* these same entities further incorporate symbolic information which represents knowledge or concepts about them, typically used in the reasoning process. That is, in our simulated world both types of object, dynamic and static, can have inherent properties explicitly described and manipulated. Such information is organised in a frame structure and is taken into account when planning the actions of the entities. For example, an agent can sit on a chair if there is information in the frame for that chair indicating that it is currently unoccupied, otherwise the frame could report the identity of the current occupant. In another example, an agent can pick up a glass if one of his arms is free and provided there is glass at a particular place. In any case the actions can be achieved by an agent if the parts involved exist and are in the specified conditions.

Figure 5-5 presents a hierarchy of conceptual entities conceived for this animation scenario. Each box is a frame which represents an aggregation of attributes that characterise a kind of entity. The topmost frames, *entity* and *surface*, are generic concepts from which more specific concepts are derived. The *entity* frame defines attributes common to individual objects such as position, orientation, colour, type, name, etc. The *surface* frame defines the attributes possessed by some of the objects such as the table and counter. That is, it specifies areas and information regarding the list of

objects contained in that surface. The rounded boxes are instances of objects that are actually handled by the controller. The hierarchical structure is typical of the frame representation. Although the objects in the animation are not required to be organised this way, the representation greatly assists in the task of maintaining the database by avoiding the repetitions of attributes.

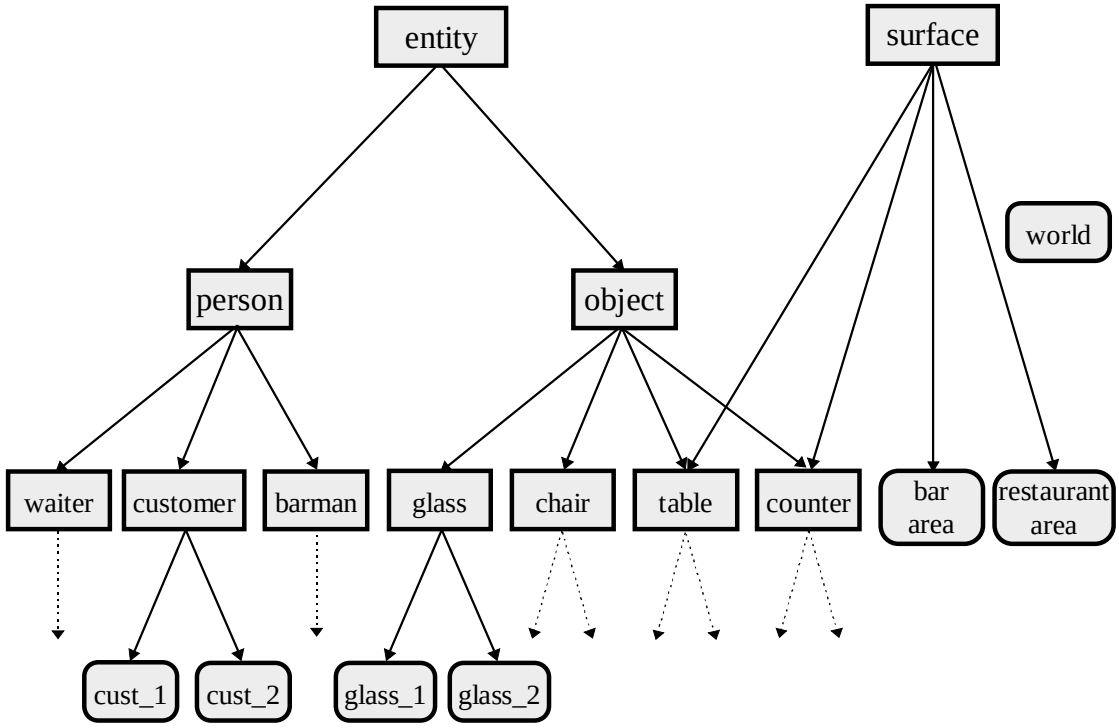


Figure 5-5: Example of hierarchy of an animation cast.

Apart from the individual information, these entities hold a variety of relationships with other entities. Such facts are useful for drawing conclusions and accessing other objects. Two examples of typical relationships are *containment* and *placement*. Examples of containment are:

> bar_area's objects_list contains {bar_counter, tall_glass, wine_glass}, and
> bar_counter`s objects_list contains {tall_glass, wine_glass}.

Such information, for example, allows an agent to find a counter in the *bar_area* or to pick up whatever glass is on the *bar_counter*. The containment relationship is the opposite of placement, for example:

> bar_counter's place is bar_area, and

tall_glass's place is bar_counter.

5.6 The Agent Entity

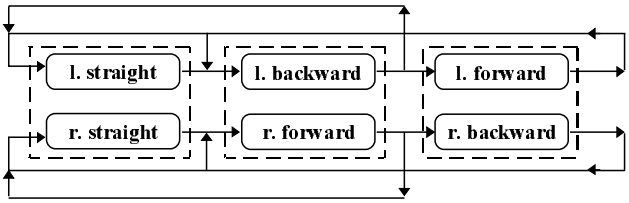
As discussed in Chapter 4 our human figure is composed of 19 links or limbs and these limbs may have one, two, or up to three degrees of rotational freedom at the joints. Nevertheless, for motion purposes a single degree of freedom or a single limb is rarely handled exclusively even in the most subtle motion, such as the turning of the head which involves not only the head but also the neck. For more practical motion control, the body is logically divided into six major limbs rather than into DOFs: left_leg, right_leg, left_arm, right_arm, head, and torso. These parts are, therefore, treated as resources which must be allocated before they can be operated. That is, tasks are entities that control motor actions and a task only becomes effectively active when all the necessary parts can be allocated for its control (see Chapter 7 for details).

```
frame person ;
  default type is person and
  default behaviour is customer and                    % Prototypical character of the person
  default activity is do_leisure_i and                 % Default activity when idle.
  default sitting_place is nothing and                 % Sitting on which object.
  default resource_list is {l_arm, r_arm, l_leg, r_leg, head, torso} and
  default l_arm is st(straight, free) and              % Posture and name of the task.
  default r_arm is st(straight, free) and
  default l_leg is st(straight, free) and
  default r_leg is st(straight, free) and
  default head  is st(straight, free) and              % Posture of the head.
  default torso is st(straight, free) and              % Posture of the torso.
  default l_hold is nothing and                        % Hold what object in the left hand.
  default r_hold is nothing and
  default support is both and                          % Standing on which legs.
  default posture is stand_up and                      % General posture of the body.
  default lock_resources is nothing and                % List of limbs used in special operations.
  default resources_used is nothing                    % Overall resources allocated.
  default main_instance is nothing and                 % The main instruction currently in action.
  default l_r_handed is right and                      % Have a preference for which arm.
  default person_attention is look_at_i  and           % Reaction to an approching individual.
  default task_list is nothing and                     % List of tasks instances.
  default instruction_list is nothing and              % List of instructions instances.
  default suspended_list is nothing and                % List of amin instruction suspended.
 .....
```

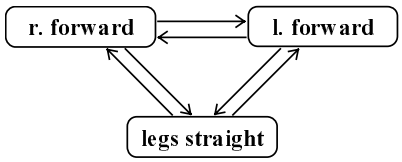
Figure 5-6: The person frame.

In Figure 5-6 the *person* frame has a slot for each of its limbs. Each limb slot is characterised by a pair of data: the current state (or position) of the limb and the name of the task that is controlling the limb. The state of the limbs and their motions is usually represented in the form of a state machine [Zelt82, Mora90, Badl94]. A limb can be in a

number of known positions. The action of a motion changes the configuration of the limb and therefore changes its state to a new one. For example, using the state machine in Figure 5-7, supposes that a "walking person" currently has the right leg *forward,* that is, the right leg is at the node state named *right forward* and the next step of walking makes the left leg swing forward. Thus, this sets the right leg to be in the *backward* state position. In our specific case, between one state and another the walking motion step is comprised of three stages as discussed in section 4.3.8.1, others [Zelt84] do the motion in up to seven stages for more realism. The motion of one leg is obviously synchronised with the other (Figure 5-7a) and this is simply implemented as production rules as for any other state based motion. These rules can be represented as a simple state machine (Figure 5-7b) as they identify certain limb configurations and update the figure's database to new configurations when a step of motion completes.



(a) Example of a state machine representing the motion of both legs.



(b) A simplified state machine.

Figure 5-7: Example of state machine for the walk motion.

5.7 Conclusions

The existing animation (Chapter 4) can be greatly improved by adding an intelligent control layer on top of it. This control scheme is composed of control elements of varying complexity and this will be discussed in the following chapters. An example of a complex animation scenario, a public bar, has been presented. Each participant in the scenario is characterised by appropriate frames which contain information manageable by the associated control components.