Chapter 2

A Review of Computer Animation

2.1 Introduction

This chapter presents a general overview of previous work on computer animation. It begins with a brief characterisation of traditional animation, followed by work on conventional 3-D computer animation and ends with recent work on behavioural animation. A broad overview of computer animation can be found elsewhere [Meal92, Vinc92]

2.2 The Traditional Animation

A long time before the advent of the electronic computer, Walt Disney and Hanna-Barbera produced the well known animated cartoons which have had great commercial success [Magn85c]. Although the production process is now considerably assisted by computers, many of the original techniques remain in use [Calv83]. The computer facilitates the job of the animator in producing sequences of still pictures called *frames*. It involves tasks like editing, drawing key frames, composition of a picture, colouring, inclusion of dialogue and sounds, etc. In the specific task of generating animation sequences, the animator draws the key frames of the story and fills the gaps with several intermediate frames to make a smooth transition. However, the production process in its entirety is much more complex. A commercial production team includes large numbers of people with specialist skills in each step in the development process from the initial planning of the story until the final production of the film [John90].

Catmull [Catm78], Lasseter [Lass87], and John [John90] have summarised the history of traditional computer animation and explain the film production process with the conventional and computerised schemes such as storyboarding, inbetweenings, post production, etc. They also present fundamental principles of traditional animation that give expressiveness to computer animation [Mari84]. For example, bending, twisting, squashing and stretching give the idea of mass and rigidity of an object by distorting the shape during an action. Exaggeration is another fundamental element of expressiveness since true realism is hard to achieve. An example of a 3-D cartoon that successfully incorporates such fundamentals is the Lasseter's "Luxo Junior" in the film *Tin Toy* [Lass87].

Despite the high cost of their production, the continuing release of new cartoons is a proof that the traditional methods are still more effective than modern computer animation techniques for the control of the characters. In the cartoon animation, the fine control of the character's motion and their mood is paramount in the visual effect.

2.3 2-D Computer Animation

2-D computer animation is usually regarded as a computerised version of traditional animation. The animated objects are assumed to be in 2-D form, and there is no explicit 3-D model manipulation by the computer. The process of inbetweening in 2-D computer animation is called *image* or *shape interpolation* [Fore86]. If the objects to be animated are originally 3-D, the interpolation of the projected images of the objects in 2-D is very difficult because of the lack of an explicit 3-D model in the computer from which to infer the correct result [Catm78, Thal89].

2.4 Overview of 3-D Computer Animation

Nowadays computer animation is mainly concerned with viewing 3-D models as two-dimensional images. There are significant differences between 2-D and 3-D animation. Firstly, being in 3-D space, objects can have physical properties, that is, attributes such as mass and volume can be simulated. Secondly, camera position control can be

incorporated in the animation because there is a concept of depth. According to Pueyo and Tost [Puey88, Tost88], a complete computer animation process can be divided into the following phases: pre-processing (synopsis storyboarding), scene editing and object modelling, the animation itself, image rendering, post-processing (shooting, synchronisation, etc.), and analysis of the results. In the present work we are mainly concerned with the animation phase. Accordingly, in the following sections we present a classification of motion control for animation.

2.4.1 Animation Motion Control

The control technique is a key issue in animating 3-D figures. Most authors distinguish between two types of motion control: key-frame animation and algorithmic animation [Magn85, Fore86, Puey88]. However, a hybrid combining both control modes is often used in practice.

Key-frame Animation

In this the animator has detailed control of the animation. Through the assistance of the animation system interface, the animator creates a number of key positions for the animated objects and intermediate positions are determined through an interpolation process know as *inbetweening* [Tost88, Fore86, Stek85].

Algorithmic Animation

In algorithmic or procedural animation the motions (e.g., translations and rotations) are described by programming in an animation language [Thal89, Gree88]. Procedural specification of the animation permits complex movements to be generated systematically by changing parameters such as speed, location, colour, etc. [Reyn82]. The procedures may include equations that describe a law of motion or simulate the physical properties of the animated object. Use can be made in the animation process of languages constructs such as loops, data types, etc.

The problem with algorithmic animation is that the animator may not have much concept of the result of the overall animation until it can be run. Thus, the animator must make a

great deal of use of his imagination and may find it necessary to make many iterations before a satisfactory result is achieved.

2.4.2 Zeltzer's Control Levels

According to Zeltzer [Zelt85, Zelt91] motions in any animation systems are characterised by two components: *specification* and *control*. These motions have a varying degree of these components which can be organised as a Cartesian space as shown in Figure 2-1. In the *interaction* axis, the control of the motion varies between *guiding* and *programming*. While in the *abstraction* axis, the level of motion specification varies between *machine level* and *task level*. He describes the three levels of interest:

- Guiding is equivalent to keyframe animation control. In this level, the animator specifies the objects and their motions directly on the screen.
- Programming is the equivalent to programmed control. In this level the behaviour of the objects are specified algorithmically using programming languages.
- Task Level, in this the control of the animation is left to the system as much as possible. The behaviour of the objects is specified in terms of task or goals rather than detailed actions. The higher level of task abstraction implies there is a greater degree of autonomy.

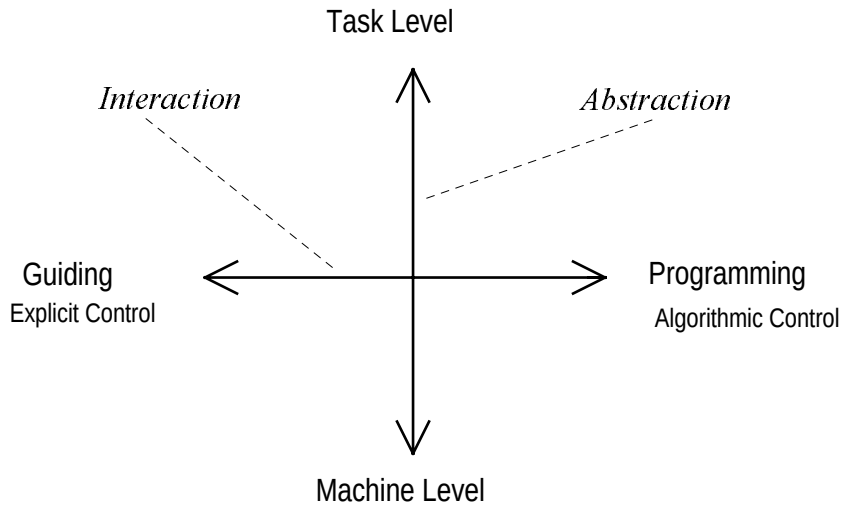


Figure 2-1: Interaction and Abstraction.

2.4.3 Animation Methods

There has been controversy concerning whether the motion of objects in an animation should be controlled dynamically or kinematically [Arms85]. This dichotomy, similar to the keyframing and procedural motion specification, has been dismissed by Wilhelms [Wilh86] and Boulic [Boul92]. It is now generally accepted that both kinds of motion control are necessary to achieve high quality animation, that is, in different situations one method may be better than the other. Furthermore, motion can be achieved in two ways, the "forward" process or an "inverse" process. In the case of a "forward" process, the object is caused to move without *a riori* planning and constant testing will be required to verify if the specified goal has been satisfied. For example, walking on an uneven surface needs constant testing to avoid making a leg penetrate the floor. Whereas in the "inverse" process, steps are determined by returning the object from the final planned position back to its current position.

The Dynamic Method

In this method objects are modelled as masses connected by joints under the influences of torques and forces. The interdependence of the body limbs are taken into account in the equations of the dynamics. Forsey and Wilhelms [Fors88] point out that dynamic analysis is a more appropriate way to deal with articulated bodies than the kinematic approach because of the many degrees of freedom involved. Once the articulated body is modelled by dynamics, it may react very realistically within the environment. An example is a stone in a free fall down a slope. It rolls down the slope until it reaches a resting condition.

Forsey and Wilhelms further point out the drawback with this approach:

> "The difficulties involved in using dynamic analysis for animation are threefold. First, dynamic analysis is computationally expensive even using a linear, recursive formulation. Second, the dynamics equations are solved by using numerical methods, and when many degrees of freedom are involved numerical instability problems can arise. Third, and most serious, precisely controlling bodies using dynamics is complex because the largely kinematic world view of the animator must be translated to the internal dynamic world

view of the system - where on the body should you pull, how hard, and for how long."

The Kinematic Method

This approach has been employed in most animation systems. Here, physics plays a small or zero role in the animation process which relies purely on the visual aspect. Although the kinematic method is often simple and intuitive, it lacks the integrity of the dynamic approach. The advantage of kinematic animation is that it allows animators to think naturally in terms of positional changes rather than in terms of forces, which is also the way traditional animators work [Lass87]. More realistic results can be generated by using equations to fake the effects of the mass in the motion of the object [John90, Magn85]. That is, the motions can be described, or guided, by equations describing curves. Another interesting technique is to copy live motion using electro-goniometers attached to the limbs of a human body. This method is called rotoscoping in which the path described by actual motions can be recorded and then applied to the animated figures [Calv80].

2.4.4 World Modelling

The stage preceding the creation of an animation sequence is the modelling of objects in the animation environment. The choice of the geometric representation usually depends on whether it is required to generate fast draft-like shapes or highly detailed surfaces. Some of the most important geometric representations for animated objects are presented in this section, however, it is not intended to make an extensive review of existing representation models. Further details on other types of surface representation and geometric modelling can be found in [Fole90].

Particle Systems

Particles are small entities with attributes such as colour, size, position, and speed. Visual effects such as the natural phenomena: fog, smoke, wind, grass, etc. can be simulated by a very large number of particles using stochastic methods [Reev83].

Stick Models

The stick model is the fastest but also the most ambiguous representation for articulated bodies. This representation has been used in several animation systems as one of the alternative display modes [Mari85, Calv91, Brud93, Owen94]. The visual quality is very poor because it does not give enough cues about the 3-D form of the object and it may cause difficulties in realising what is happening with the figure. Nevertheless, it may be useful for representing different bodies geographically distributed in an environment. It is also useful for representing postures selectable from a menu panel.

Wire Frames

This is the simplest method of building 3-D models. Edges of the object are drawn with straight segments forming polygons which give a rough shape of the object. It is widely used for previewing an animation sequence before proceeding to a final production. It gives some clues about the actions of an object despite possible ambiguity [Gira87], and the visual quality can be improved if hidden lines are removed [Badl85b]. Bruderlin and Calvert have made extensive use of a type of ellipsoid wireframe for human figures [Brud93]. It is particularly useful for choreography because it gives a reasonable visual perception of the motion of the body.

Polyellipsoids Display

Herbison-Evans [Herb82] made use of the graphics system called NUDES (acronym for Numerical Utility Displaying Ellipsoid Solids) to draw figures with multiple ellipsoids with hidden lines omitted. This is an alternative to the stick and wire frame representation. It is claimed that some of the simplicity and speed of stick figures is retained, however, problems in determining the concave parts of the anatomy have been pointed out.

Constructive Solid Geometry (CSG)

A set of 3-D geometric forms such as box, sphere, cylinder, and cone is used as the primitive set for building graphical objects. The ability to add and subtract volumes

makes it useful for designing mechanical components in CAD. The CSG model may incorporate surface information such as colour, reflection for the visualised geometric object [Doi88].

2.5 Previous Work on Autonomous Motion Control

In the last decade, several approaches were developed focusing on the aspects of motion control of multiple moving objects in dynamically complex environment. Because of the multiplicity of animated objects and their interactions with each other in the environment, there are many difficulties for the animator in planning the animation. Whereas conventional control modes are quite sufficient to deal with the motion of an individual figure in a static environment, they are inadequate for a typical dynamic animation environment which requires more comprehensive motion control to allow the figures to navigate in the surroundings and to react accordingly.

In behavioural animation, attempts have been made to shift the bulk of the control to the system, by providing the figures with greater autonomy to "control their own motion". That is, the expertise of the animator is captured and a knowledge-based control mechanism is used to develop the figures' motions. Sun [Sun93] has distinguished three categories of motion control of animated objects in behavioural animation: *the sensor-effector*, *predefined environment*, and *behaviour rules*. A brief description of these control methods is given followed by some relevant examples.

2.5.1 Sensor-effector Approach

In this approach the behaviour of the animated figures is formed by three components: sensors, effectors, and a neural network. The sensors provide the inputs to the behaviour neural network. Certain combinations of sensed data are recognised in the neural network and if a computed value is within a specified threshold value then the associated effectors are activated. These elements actually emulate the response of biological systems in response to events in the environment [Wilh90, Tu94, Reic94, Zelt84].

The Artificial Fishes of Tu

Tu [Tu94] has simulated the behaviour of different types of fishes. The fishes inhabit a virtual marine world in the presence of simulated underwater currents, aquatic plants and other types of fishes. The artificial fish is a graphical entity comprising a motor system, perception system, and behaviour system. The motor system comprises the model that simulates the dynamics of the fish by using the actuators (fins) and a set of motor controllers. With this "mechanical" model, the fish can effect motor function such as "swim forward" or "turn left". The perception system gathers visual and temperature information from the environment. The behaviour system decides on its intention from the fish's habits (schooling, likes warmth, etc.), mental state (hunger, libido and fear), and incoming sensory information, at each time step. There is a generic intention generator which selects a behaviour to perform, and each type of fish has a specialised version of the generic behaviour. The triggering of one behaviour or another is a function of selected variables.

This system does not employ a neural network as such. However, the way that the intention generator combines the different sensory information, uses the three mental states and habits, and triggers behaviour based on threshold values is similar to the operation of a neural network.

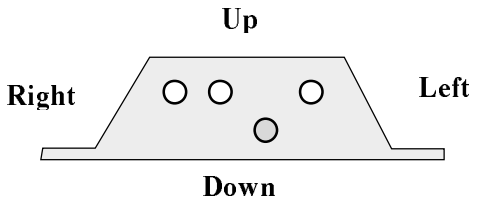
2.5.2 The Predefined Environment Approach

This approach is interesting when the distribution of objects in an environment is largely static. Then a number of collision-free paths can be determined before the animated entities even start their motion. The choices of path from an initial location to the goal location is planned following constraints or criteria. A number of systems [Rids86, Rids90, Cher89] have been proposed using pre-computed paths as a way of avoiding the cost of computing them at run time.

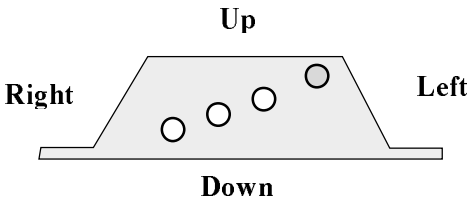
The Director's Apprentice System

Ridsdale makes use of an expert system for planning the motion of actors on a stage [Rids86]. A set of staging rules capturing the facts and relations of the directing

principles are stored in a database in the form of "if-then-else" rules. These rules map the scene attributes such as the motivations of the character and their interaction with the environment, and also guide the motion of subsidiary characters relative to those of the main character across a set of pre-computed paths. There are many categories of rules, for example, the focus of attention of the public is directed to the principal actor by placing him strategically in relation to the rest of the cast on the stage. The actor may be positioned ahead or in the middle of the stage, and because it is the main actor at that moment of the play, rules are triggered to ensure a formation that highlights him. An example of formation and rule is illustrated in Figure 2-2.



a) Focus by Position.



b) Actual Line Focus.

Here, the principal actor (the next to speak) has been given focus by having the others actors stand further upstage.

IF next-to-speak is actor$_i$
AND actor$_j$ is-downstage-of actor$_i$
THEN moves-upstage-of $_i$,

Here, the principal actor (the next to speak) has been given focus by having actors form a line which points towards him.

IF next-to-speak is actor$_i$
AND actor$_j$ is-not-aligned-to actor$_i$,
AND actor$_j$ is-closest-to actor$_i$,
THEN align-actor $_{j,i}$,
AND next-to-be-aligned-is $_j$

c) Rules to handle the actual line of focus.

Figure 2-2: Planning the motion of actors on a stage.

2.5.3 The Behaviour Rule Approach

The behaviour approach is similar to the sensor-based one except that the control of the object's behaviour is implemented by behaviour rules rather than a neural network. These rules are typically expert systems rules, "*if condition then action*", where only one rule is selected at a time from among the candidate rules. Then, the action part is done if the condition part is satisfied. The chaining of successive rules which identify a variable

number of conditions in the environment will eventually trigger the effector actions that produces a behavioural action.

The Paradise System

Maruichi et al. [Maru87] implemented a behavioural simulation of a school of barracuda and herring. The herring object has variables such as position, direction, and velocity that represent its state. Instances of herrings are created with associated classes such as the sensor class which is needed by the herrings to probe the environment. Each message has a transmissive area and a life span. For example, the smell message can be sensed within a certain radius and after a lifetime it is removed from the environment. Each character communicates with others indirectly by sending and receiving messages to/from the environment. Three rules identify the possible situation a herring could be in:

- A herring is normally swimming forward when there is nothing around it.
- A herring will join a school with other herrings when there is no barracuda nearby.
- When a barracuda is near a herring will try to escape and in its panic will ignore other herrings.

Despite some similarity with Tu's Artificial Fish, this system employs behavioural rules rather than a neural network to make decisions.

The PetWorld System

Petworld [Code88] is a system for modelling aspects of animal behaviour intuitively. It is a world of pets, rocks, and trees in a limited two-dimensional Cartesian plane. In Petworld the pets have a body orientation, a limited field of view, can carry one rock at a time, eat trees, etc. Pets are assumed to be antagonistic and can attack each other. Each pet has a limited set of internal states such as *hunger, fear,* and *injury*, on a scale of 0 to 100. In Figure 2-3 several sets of rules define the behaviour pattern of the pets. These rules, when conditions allow, chain into an hierarchical tree as shown in Figure 2-4. The hierarchy controls the flow of decisions which is similar to the operation of some computer games. Some of the possible actions are MOVE-TOWARDS, TURN, LIFT, DROP, EAT and ATTACK.

BRAIN

1. **If both HUNGER and FEAR are high, effect a tradeoff between COMBAT and FORAGE.**
2. **If FEAR is high, COMBAT.**
3. **If HUNGER is high, FORAGE.**
4. **If FORAGE is recommending that there is a food element immediately available, then FORAGE.**
5. **If NEST has some non-trivial action to perform, then NEST.**
6. **Otherwise, EXPLORE.**

COMBAT

- **If you have an available attack, and your damage is low, then recommend a tradeoff of attacking and running away.**
- **Otherwise, recommend running away from any visible pets.**

ATTACK

- **Construct a ranking of things you can attack standing right where you are.**

Figure 2-3: Pet's behaviour patterns defined by rules.

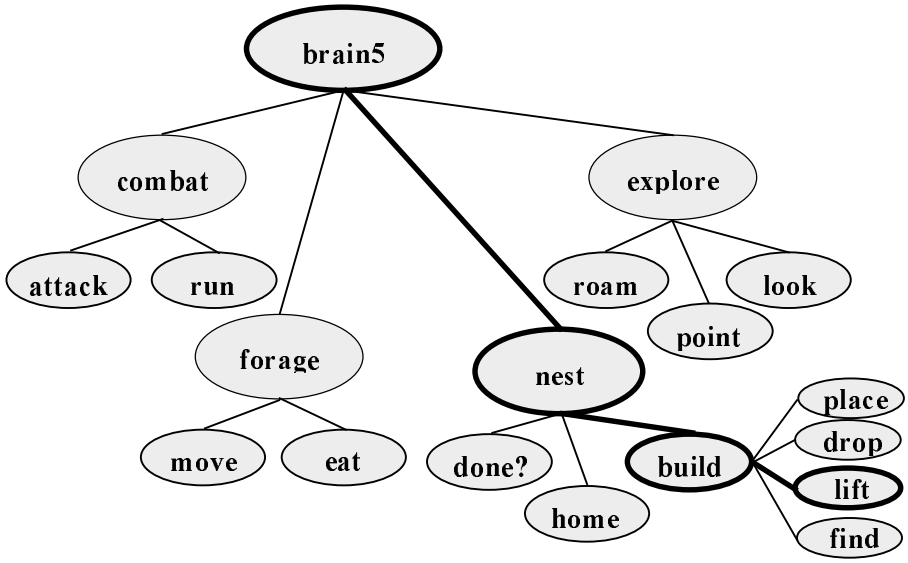


Figure 2-4: Hierarchy of a decision tree. Bolder lines indicate traversal of the hierarchy through selected branches of the tree until the lift action is selected.

The Relation Approach

Sun [Sun93], has discussed the problems of animating multiple moving objects and has proposed the *relational* approach for solving the problem of specifying multiple interactions between moving objects. In the example of a group of dancers, a set of pre-defined relations provides a pattern for the coordinated motional behaviour of the dancing couples (Figure 2-5). The environment is comprised of static objects and dancers which may stimulate or constrain the motion of a particular dancing couple. For example, a moving object avoids colliding with another moving object, or one will back away from a disliked object. At a particular moment a collection of influences may be affecting a moving object and an appropriate behaviour is instigated in response. Each of these influences is specified by a relation that maps from the object that causes the movement to the one that performs it. A relation is, thus, a unit of behaviour which has three main parts as listed in Figure 2-5: a *source* which is the object that causes the motion to occur; a *responder* which is the object that performs the motion, and a *response* which specifies how the responder responds to the *source*. There are also some structuring mechanisms to select and control the global motions of the objects:

- relation state - a relation can be in a *potential*, *active*, *suspended*, or *terminated* state. The state is switched from one to another depending on the dynamic changes in the environment during a motion. For example, when a relation selected by the environment is in the *potential* state and senses its enabling condition then it changes to the *active* state and at the end of its response it returns to the *potential* state.

- interaction control - an active relation determines the state of other relations. For example, if a relation detects danger then it suspends the relations currently controlling the object movements and activates others that make the object go to a safe distance.

- pattern control - is based on collecting all the relations that control a particular behaviour pattern. There are two pattern structures, time patterning and relation patterning. One or more relations are collected together and if the referenced time is

reached or a given relation becomes active then the collected relations are switched to the *potential* state.

- sequence control - if more than one behaviour occurs they are compressed or stretched in time to use the available time, that is, bridging the time gap if any.

relation	(source responder): enabling condition?	response behaviour
inroom	(alls dancer): too close?	turn a a from the all
a a block	(blocks dancers): too close?	turn a a from the block
dislikeblock	(blocks dancer_): colour red?	quick reverse turn

Figure 2-5: Example of relations.

Finally, relations permit the specification of small units of motion. Whenever possible the structuring mechanism abstracts the atomic relations into a larger behaviour. Sun's relational approach is, thus, a hierarchy that is built in a bottom-up fashion.

NSAIL: Behavioural Animation using Constraint-Based Reasoning

Mah [Mah94] built a behavioural animation based on a constraint-based expert system, ECHIDNA. In his 2-D animation of sailing boats, the motions of the boats are planned under physical influences: winds, boat heading, and collision avoidance. Small knowledge units called *morsels* encode knowledge that relates the animated objects, the variables in other *morsel*, the goal, the state of the object, etc. Upon occurrence of a goal *morsel*, new *morsels* that satisfy the current morsels of the animation are successively considered, as the constraints attached to them are satisfied, until eventually the goal *morsel* is reached. The sequence of *morsels*, thus obtained in this process, describes a plan with actions that achieve that goal. The information of the environment delimits the space of possible *morsels* for the formulation of the boat's navigation plans. All the reasoning eventually leads to a plan which includes a series of adjustment of the boat's heading and sail angle.

2.5.4 Other Related Work

Several approaches to animation have focused on the realisation of autonomous motion. Such systems have some points in common with research being carried out in robotics and artificial intelligence [Calv91]. The most evident aspect in the approaches described below is the need for the automatic planning of motion and the embedding of behaviour into the figure's autonomous response.

Actor Systems

Historically, the actor concept was introduced in AI by Hewett [Hewi73, Agha86] and it was brought later into the realm of computer animation by Reynolds in the ASAS language [Reyn82] and by Thalmanns in the MIRA system [Magn83]. The actor concept was conceived as a computational entity that followed directions given in a script. According to Reynolds:

> "Most basically an *actor* is a 'chunk' of code which will be executed once each frame. Usually an *actor* (or a team of them) is responsible for one visible element in an animation sequence, hence it contains all values and computations which relate to that object. In this sense an *actor* serves to modularize and localize the code related to one aspect, isolating it from unrelated code..."

Therefore for a particular type of actor the functions that model certain kinds of behaviour can be clustered into a single module. When its role is required, an instance of the actor type is created. It has a finite lifetime and it is deleted when no longer needed. This is similar to methods and messages in the Object-Oriented Programming approach [Byte89]. The actor performs some predefined tasks through two special message operators, ***send*** and ***receive***, with which the actor can exchange messages with other entities. The ***send*** operator places a message in the receiver's (the second actor) "mailbox" requesting an action. The ***receive*** operator picks up the message from the mailbox and develops the requested actions. That is, the message type is matched against the recipient cases such as *speedup* and *slowdown* (shown in Figure 2-6), if a match succeeds then the associated action is executed. Figure 2-6 is the original example from [Reyn82] that illustrates the message operators.

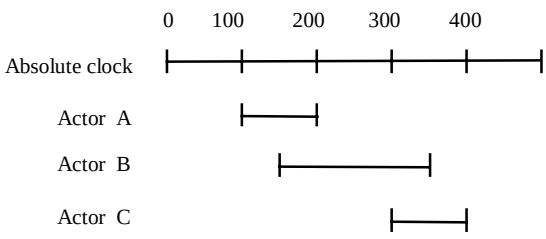
```
(send bouncer speedup  1.10)
_____________________________________

(receive ((speedup f)    define speed (times speed f)))
         ((slowdown f)  define speed (quo speed f)));
         ( any          (print 'What?)));
```

Figure 2-6: Actor message operators

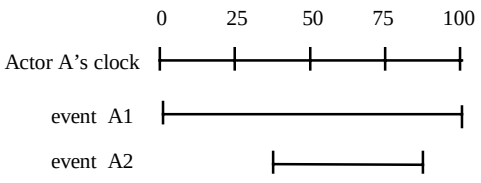
In another work, Reynolds [Reyn87] models the flocking behaviour of birds. Each bird entity, called *boid*, is an independent actor that keeps flocking with its companions, observing behaviours such as: collision avoidance, speed, and geometric formation with the flock. These principles can also be applied to groups of other animals.

Animation of Multiple Actors

Bergeron [Berg83], presents a structured motion specification approach to coordinate the performance of the actors in an animation. He uses a metaphor of choreography where an imaginary director directs a cast of actors involved in a play. The director only sends the starting and stopping messages to the actors. The actor is a computerised entity that embodies a collection of activities pertaining to a person or an object. He can be a lighting technician, a camera-man, or even a virtual camera that is used by the display software. Figure 2-7a shows a global list of events where a director schedules the cast to act in the specified time and their existences are limited to periods of time. Each actor, in turn, has a predefined list of activities within its lifetime. For example, in Figure 2-7b the events A1 and A2 are part of the actor A activities and performed within its allotted time. Similar work has been reported by Fortin [Fort86].



The actors' lives relatively to the animation time.



Actor A's actions relatively to its lifetime.

Figure 2-7: Schematic presentation of actors's performance and events.

Knowledge-Based Systems

Zeltzer [Zelt86] proposed a frame based approach to goal-directed animation. He discussed the requirements for the automatic motion synthesis of articulated bodies. A goal such as the STAND_UP skill is implemented by a frame that is potentially achieved by one of the three actions, S1, S2, and S3. Depending on which condition is satisfied the corresponding skill is trigged. The movement frame, such as WALK, also establishes a hierarchical connection with the other movements frame, STAND_UP. Each skill S is a specific motion that the figure performs and is implemented by procedures called *motor programs*. Figure 2-8 exemplifies Zeltzer's proposal. Drewery and Tsotsos [Drew86] have also proposed the use of frames.

Frame:	**WALK**
Speed:	**"normal" (default)**
Direction:	**"forward" (default)**
Preconditions:	**standing, feet on ground**
Triggers:	**if not standing, call STAND_UP**
Frame:	**STAND_UP**
Speed:	**"normal" (default)**
Direction:	**"ahead" (default)**
Preconditions:	**feet on ground near center of gravity**
Triggers:	**if current posture is sitting, S1**
	if current posture is lying, S2
	if current posture is kneeling, S3
Frame:	**S1 (prepare to stand from a sitting position)**
Speed:	**"normal" (default)**
Direction:	**"ahead" (default)**
Preconditions:	**none**
Procedure:	**move_legs**

Figure 2-8: Zeltzer's frame example.

The Instruction Approach

Badler et al [Badl91b] has discussed the belief that computer animation in the form of *narrated animated simulations* is an engaging medium for instructing agents in the performance of tasks. They justify this by:

> "The only way to create the kind of *flexible* narrated animations needed to instruct agents of varying capabilities to perform tasks with varying demands in work places of varying layout is to drive *both* animation and narration

from a *common representation* that embodies the same conceptualization of tasks and actions as natural language itself."

Example of a sequence of instructions simulating two agents in front of a control panel [Badl91b]:

John, look at switch twf-1.
John, turn twf-1 to state 4.
Jane, look at twf-3.
Jane, look at tg1J-1.
Jane, turn tg1J-1 on.

They also discuss that in a more elaborate situation, an instruction, as a plan, can delineate an action at several levels of detail or in several ways. For example, the instruction for "filling holes in plaster where the lath, as well as the plaster, has disintegrated" is given by:

"Clear away loose plaster. Make a new lath backing with metal lath, hardware cloth, or, for small holes, screen. Cut the mesh in a rectangle or square larger than the hole. Thread a 4- to 5-inch length of heavy twine through the center of the mesh. Knot the ends together. Slip the new lath patch into the hole...".

They have proposed a system structured as a pipeline with activities distributed in several stages: Natural Language Processor, Incremental Planner, Semantic Mapper, Simulator, Motion Generators, Display Process, and Narrative Planner and Generator. The detailed discussion of the proposed system is described in their paper [Badl91b]. It represents a big advance over the previous works by Zeltzer [Zeltzer86], and Drewery and Tsotsos [Drew86], specially regarding the narrated animation where a great deal of information is implicit, however, some difficult problems in linguistics have to be overcome.

A Blackboard Approach

Calvert [Calv94] has argued that the methods of motion control, such as dynamics and kinematics used in current animation systems, only account for a limited control of specific motion tasks of the humanoid. In order manage a major human behaviour, explicit knowledge should be incorporated into the system in the form of an expert system. The expert system deduces from this knowledge and subsumes the existing

motion control methods. The addition of an expert system to the existing animation system as a top layer is a natural approach to handling a higher level of control. However, Calvert has pointed out that simply chaining production rules in the expert system engine does not adequately synchronise the animation events to produce a correct response in a complex animation. Thus he suggested the use of a blackboard concept [Jaga89, Enge88, Haye93] as a way of integrating the reasoning with the animation algorithms. This also ensures a reasonable response time. The components of the blackboard model are presented in the next chapter.

2.6 ESPLANADE

ESPLANADE, an acronym for Expert System for PLANning Animation, Design and Editing, is a knowledge-based animation presentation planner which makes use of filmmaking techniques [Karp93]. Although the presentation, or viewing, of the animation is not directly related to the motion control, it has the job of communicating the actions to the viewer more effectively. ESPLANADE is particularly interesting because of the approach to building the sequence of viewing shots. It accepts as input a script with a plan of actions and the contextual information about actions and objects. It also produces as output a presentation plan ready for viewing. Internally a film is organised into a structure of levels of hierarchy as shown in Figure 2-9. The actual activity is developed in the *Shot Level* where the centre of interest and duration are specified. The remaining levels have different scopes that coordinate the sequences of shots.

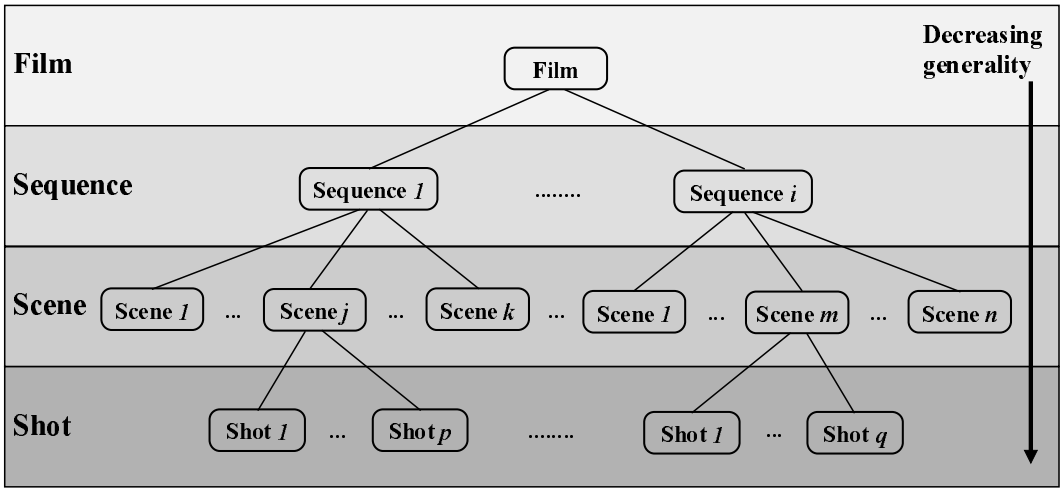


Figure 2-9: Karp and Feiner's film structure.

2.7 Summary

The characterisation of 3-D computer animation in terms of models of control helps in the understanding of the complexities involved in co-ordinating the motions of the animated figures in a dynamic environment. However, this separation does not exist in an actual system. That is, the animator interactively composes some keyframes and invokes motions among others by procedures. As pointed out by most of the workers in the field [Calv94, Zelt91] the most important aspect of motion control is goal orientation. The current trend of computer animation is to consider each animated figure as an individual capable of autonomous behaviour leading to so called behavioural animation.