PERCEIVED ANIMATE MOTION BY SIMPLE DETERMINISTIC RULES OF INTER-OBJECT BEHAVIOUR

by

Timothy Christian Lethbridge
B.SC.(CS) University of New Brunswick, 1985

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Master of Science in Computer Science

in the School

of

Computer Science

This thesis is accepted.

.....

Dean of Graduate Studies and Research

THE UNIVERSITY OF NEW BRUNSWICK

April, 1987

ABSTRACT

This thesis describes the use of behaviour functions to create systems of graphic objects that appear to be behaving purposefully. These functions determine the motion of an object based on the positions and velocities of all the objects in its environment.

A mathematical framework is presented that relates classes of rules to the type of behaviour the objects are perceived to take on. Functions using just the positions of objects at the previous time interval can model pushing, pulling, approaching, retreating and tendency to maintain distance. When the second preceding time-step is used, several other important fundamental types of response can be achieved. Sophisticated behaviour such as chasing, pouncing and fish schooling can be created by combining the more simple responses.

A software system is described that, in real-time, controls the movement of a number of objects on the screen according to behaviour functions described in terms of a simple language.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Colin Ware who suggested the topic, and through whose kind assistance I was able to bring this thesis to completion.

CONTENTS

LIST OF FIGURES

Chapter 1.

INTRODUCTION

The word animation has two main meanings; it can mean a motion picture produced from a series of images, each representing a small advancement in the action, or it can refer to the general quality of vivacity or liveliness. This thesis is concerned with both senses: A technique of animation is described that is useful and applicable to the creation of all types of animated motion picture, however special emphasis is placed on those animations that move with intentionality or life-like actions.

In effective animation, viewers receive the psychological impression of animacy when they are in fact seeing the simple movement of geometric shapes on a screen. We may perceive, for example, one object 'kicking' another around, even though both shape and motion may be quite simple. In conventional animation, this perceived intentionality is achieved through the skill and intuition of the animator who must achieve the behaviour by specifying the positions of all parts of the scene in a succession of time frames. The computer can aid many aspects of the animation process, but the animator's judgement usually remains an essential element.

The behaviour-function technique presented in this thesis enables the animator to tell objects how to behave, rather than just where they should be, thus allowing him to concentrate his creative energy at a higher level and be potentially far more productive. The concept of behaviour functions is that objects move by responding continuously to the various objects in the environment -- the primary objective of this thesis is to find behaviour functions that would be useful to animators.

In the remainder of this chapter, a look is first taken at animation in general; then several research studies with relevance to behaviour function animation are summarized. During this discussion, comments are made about the relevance of the research to behaviour function animation and the implementation created for this thesis, PAM (Perceived Animate Motion). In subsequent chapters, the concept of behaviour functions and the current implementation is described in detail, after which findings regarding classes of simple behaviours are outlined.

1.1 AN OVERVIEW OF TRADITIONAL ANIMATION

Motion picture animation had its beginnings in the 19th century with the invention of several mechanical devices that

displayed rapid sequences of images.[1] Animated films were first produced starting in 1906 and techniques were continually refined over the subsequent decades. It was in the 1930's, however, when animation became big business by the work of Walt Disney -- the first full-length commercial animated cartoon being Snow White and the Seven Dwarfs.

Today, animation is everywhere. We see it most frequently in entertainment both on television and in the movies -- it is used for entire films or just for special effects. Industry and government make extensive use of animation for marketing, mass communication and education. It is also useful for research where it is applied to simulation.[2]

The complete process of producing a traditional animated film is too complex to describe in detail, but some fundamental elements are worth mentioning. The story is first successively refined in a hierarchical series of documents[3]. After the sketches of the characters and background setting have been decided upon and the sound-track has been recorded, the animation-process proper begins. A master animator draws key frames -- significant points in

---

1. Magnenat-Thalmann and Thalmann 1985 p. 11
2. Ibid. p. 12
3. Ibid p. 4-5.

the action. Assistant animators draw some of the in between images and the remaining frames are created by so-called 'in-betweeners'. Once the drawings are complete, the images are painted ready for shooting. The shooting process can be quite complex, with series of images being moved over various levels of background. The final stage is postproduction where sequences are assembled, edited and synchronized with sound before release prints are made.

1.2 THE APPLICATION OF COMPUTERS TO ANIMATION

The computer has been used as an aid in virtually all aspects of the animation process.[4] Numerous systems have been developed with the animation artist in mind. Also, languages have been developed that allow the full power of computers to be exploited for various special effects. There is a debate between advocates of interactive animation systems and advocates of languages -- the issue being ease and naturalness of use versus flexibility and power.

Magnenat-Thalmann and Thalmann describe two fundamental classes of system: computer-assisted (key-frame) animation and modelled animation. Computer-assisted animation has two levels of complexity: level 1 systems are merely graphics editors for drawing and painting key-frames; level 2 systems

---

4. Ibid p. 13-17

compute the in-between images, moving objects along defined paths between two key-frames. Modelled animation systems allow for the, "manipulation of more general representations."[5] Modelled systems are conceptually broken down into three levels above computer-assisted systems. Level 3 systems provide operations such as scaling, rotation and virtual camera movements, level 4 systems provide actors that move on their own based on constraints or rules, and level 5 systems, as of yet undeveloped, would incorporate learning and intelligence into the animation process.

Level 4 systems are at the forefront of research; the PAM system designed for this thesis is such a system although it is only experimental and lacks many features typical of a modelled system. Magnenat-Thalmann and Thalmann discuss several modelled systems[6] termed object-oriented or actor systems. An object-oriented animation system is one where the programming emphasis is on the movements and interactions of individual objects rather than the environment as a whole. Many of the higher-level animation systems are written in the Smalltalk language as this treats all entities as objects which interact by sending messages. The concept of 'actors' was developed by Hewitt[7] who, along with

---

5. Ibid p. 14

6. Ibid. pp. 135-164

7. Hewitt et al. 1973

Greif[8], defined 'behaviours' in terms of the actions taken in response to transmissions from other objects. The concept of behaviour function animation developed in this thesis is somewhat different, as is the concept of actors. In the work of this thesis, actors conceptually respond continuously -- all movements are calculated responses -- whereas in previous object-oriented systems, patterns of movement are changed at irregular time intervals when messages are received. Our behaviours are continuous patterns of action that may develop and change over time as opposed to actions taken in response to a particular event. Actors in previous object-oriented systems might be imagined as blind, guided around by what they can learn by talking to other objects; our actors, on the other hand, can see!

Another class of systems that could be classed as level 4 systems are constraint-oriented systems. An example of such a system although not specifically intended for animation, is Thinglab[9]. Such systems operate based on a set of rules or equations that must be satisfied at all times. Like behaviour-function systems they must check their environment at every time step. They must iterate towards some 'solution' to the constraints, therefore exhibiting some behaviour. The rules controlling objects in a

---

8. Greif and Hewitt, 1975

9. Borning 1986

behaviour-function system are not constraints, allowing a much wider set of available behaviour.

Most of the level 4 systems currently implemented were designed as experimental prototypes of systems that could be used for production. Most of the differences among these systems are in the ways they allow for the description of higher level object and actions. For example, such a system may be able to define hierarchical objects, actor behaviours, camera movements and light sources into procedures with parameters and other variables. This thesis is only intended to study behaviour functions in a low, detailed level; it is not intended to create a production-type system.

Several other technological considerations are also important in the categorization of computer animation systems. One consideration is whether the system is designed to produce animation in real time or frame-by-frame; a second consideration is the realism that the system is capable of producing. These two considerations conflict with each other since a real-time animation system (such as a video-game), typically single-user and microprocessor based, does not have the computational power to produce great realism. Magnenat-Thalmann and Thalmann[10] outline many realism issues

---

10. Magnenat-Thalmann and Thalmann 1985 p. 79-130

and techniques such as hidden-surface elimination, reflectance, shading, transparency, texture, shadows, anti-aliasing and human modeling. The PAM system is not concerned with any of these things. It is designed for real-time or near-real-time operation and attempts to present realism of movement, not image realism.

1.3 MICHOTTE'S WORK ON PERCEIVED CAUSALITY

When a behaviour pattern is displayed on a screen, what is important is its perception on the part of the observer, not the abstract movements themselves. When attempting to synthesize behaviour, it is therefore desirable to know the classes of simple motions and interactions to which humans would attach higher-level meanings at a subconscious level, and the variations and limits on these classes of motion that would still yield such impressions. Armed with findings about fundamental effects, we should be able to generate more complex effects.

Among the phenomenological effects discussed in the literature are causality, intentionality and animacy. The impression of causality is the impression that one action caused another -- the actors could appear totally inanimate. Intentionality refers to the impression that an object purposefully performed some action. Animacy is less clearly defined -- the impression of animacy has much in common with

the impression of intentionality, but might be regarded as a broader term encompassing any impression of life-like activity ('animation' in its sense of vivacity or liveliness).

A major work in the perception of causality is by Michotte.[11] Michotte studies two psychological effects in depth: 'launching' and 'entraining'. Both of these effects are described in terms of the 'ampliation' of movement, which is the "extension of a movement from one object onto a second, in such a way that it remains the movement of the first object while bringing about the 'displacement' of the second."[12]

The launching effect occurs when one object impacts another and sets the second object in motion, the whole motion being regarded as that of the first object. As with most of the effects Michotte defined, an optimum range of speeds was found in which subjects reported this effect -- if the first object moves too fast or too slowly, the effect is weaker or nonexistent. A time-lag between impact and the start of movement of the second object was also found to be significant. As the halt-time increases the effect changes from an impression of continuous movement to an impression an object is 'caught' on something, to an impression of a movement in

---

11. Michotte 1963

12. Ibid.

two stages, to an impression of two separate movements. Another example of a modification in this effect occurs with a change in the ratio of the speeds of the two objects: if the second object moves off much faster than the first object approached, then a 'triggering' effect is reported where the ampliation is cut short.

Michotte's second major effect, entraining, involves the impact of one object into a second after which both objects move off at the same velocity. Like launching, Michotte identified many modifications to this effect given special conditions.

One particularly interesting result from Michotte is his finding regarding the impression of animal (as opposed to animate) movement. If one perceives animal movement, one perceives that an object is moving under its own power. Michotte reports that for such an impression, it is necessary that a change in shape be followed by a forward movement -- by way of ampliation, the change in shape becomes the forward movement. This, in fact, seems quite intuitive since it would be hard to imagine an unchanging object moving with a steady speed and direction as having any animal-like quality unless its shape were changing (e.g. the contraction and extension of a caterpillar). Although it would be possible to build change of shape into a generalized behaviour function animation system, the inves-

tigations in this thesis and the PAM implementation only involve the motions of unchanging objects. Due to this, when the movements of animals are simulated, the emphasis is on interaction of several actors as opposed to the locomotion of a single actor. The objective is to obtain impressions of 'animacy' from these interactions. If change of shape were introduced to add Michotte's 'animal movement,' the impression of animacy might be enhanced.

1.4 NECESSARY INGREDIENTS FOR THE IMPRESSION OF ANIMACY

Marion, Fleischer and Vickers[13] outline five requirements they feel necessary for an animation system to effectively elicit impressions of animacy in observers. They derive their conclusions from work in psychology (including Michotte), animation, dance and related areas. Certain concepts are not well structured by Marion et al: For example one of their five requirements for animacy, intentionality, is an impression that has its own sub-requirements, while the other four requirements are attributes of the motion. It would seem reasonable to handle the sub-impression of intentionality, a property of the perceiver, separately from the properties of the stimuli and actors, but this is not done in Marion et al. Also, they fail to define animacy, the very concept for which they are attempting to classify

---

13. Marion, Fleischer and Vickers, 1984

requirements. Nevertheless, the work of Marion et al. is the only broadly-encompassing research in the field and is therefore discussed here.

Intentionality is the perception that Marion, Fleischer and Vickers identify as a requirement for the perception of animacy. They admit that detailed work is needed to determine the precise circumstances which evoke feelings that an object has intentions in its actions, but do indicate several circumstances that appear to heighten this impression. In particular they mention apparent violation of Newton's laws, systems too complex to understand and systems where goals are modelled. Behaviour function animation certainly allows for the first two circumstances: for objects to violate Newton's laws is more the norm than the exception in any animation system unless constraints are applied. Complexity is easily achieved because the number of possible slight differences in interactions mounts rapidly as additional freely moving objects are added to a behaviour-driven system. The modelling of goals, however, is not inherently part of the behaviour function concept since, in general, objects would be expected to move around with fixed behaviours ad_infinitum. Goals in behaviour function animation would only be achievable if 'traps' were set that would prevent particular behaviours once a particular circumstance has occurred.

Dependent and independent motion is the second animacy-inducing factor identified by Marion et al. For this it is necessary for objects to influence each other in varying degrees. This factor is unavoidable in behaviour- function animation since movements are defined in terms of inter-object influence.

Marion, Fleischer and Vickers' third requirement for animacy is temporal phrasing. This refers to the possibility of breaking down a larger behaviour into distinguishable units. Two types of temporal phrasing can be considered, sequential and hierarchical. In the sequential type, one behaviour changes to another at a certain point in time, presumably due to some change in stimulus. In hierarchical temporal phrasing a higher-level ongoing behaviour, say walking along a street, can be seen to be composed of lower-level behaviours such as movements of arms or legs. In behaviour function animation, sequential temporal phrasing would be naturally present due to changes in behaviour as objects move among stimuli. Hierarchical temporal phrasing, although achievable with behaviour-function animation, is most often attached to hierarchically described objects. Since only simple objects are to be considered in this thesis, little attempt is made to obtain the latter kind of temporal phrasing.

Another important quality found to yield animacy is exaggeration, which is achieved by increasing the contrast between two objects' behaviours. Exaggeration itself gives rise to impressions of humour or enthusiasm which are only attributable to animate objects.

The final factor considered to be prerequisite for animacy is lack of repetition. Marion, Fleischer and Vickers suggest that the injection of randomness into a system is necessary to achieve this objective. However it is desirable to avoid random variables because they add complexity and make mathematical models less easy to define, and it is possible that they may not even be necessary. From the mathematics of chaos we know that for a system of even minimal complexity, the process of feeding back outputs into inputs results in sequences that in general do not repeat; as it happens, this is exactly what is going on in behaviour function animation when the status of objects as a result of previous behaviours is used in the computation of current behaviour.

In addition to concerning themselves with the qualities of interaction and movement that might yield animacy, Marion, Fleischer and Vickers turn the question around and propose four features necessary in a software system that is to display animate objects. Three of these features -- logical relationships between objects, separation of motion and

object representation, and parameterization of the motion description -- are naturally part of the behaviour function concept. The fourth feature, hierarchy of motion description, can be achieved explicitly by the animator by defining behaviours such that a given object responds primarily to an object considered to be at the next higher level in a hierarchy.

1.5 BRAITENBERG'S VEHICLES

This concept of feedback producing animate behaviour is made use of by Braitenberg[14] whose work was recently discussed in Scientific American [15]. Braitenberg calls his concept 'synthetic psychology.' He works with simple 'vehicles' that are composed of rectangular blocks with light sensors connected to wheel motors. Several methods of interconnection are described, but they are all very simple. The vehicles are released on a plain whose monotony is broken only by occasional light bulbs. The vehicles move around on the plain, however different vehicles exhibit different behaviours depending on how the light sensors are connected to the motors. In one case the effect of turning towards a light bulb might lead to a stronger attraction to the light bulb. As this process continues, the object might accelerate

---

14. Braitenberg 1984

15. Dewdney 1987

towards the bulb, eventually destroying it. Instead of this 'aggressive' behaviour, the simple wiring of other objects might lead to 'love', 'timidity' or a wide variety of other animate behaviours.

Braitenberg's approach is, in essence, very similar to the one developed here, except that using computer graphics and simulation as opposed to mechanical devices is far more flexible and allows more freedom for exploration.

Chapter 2.

FRAMEWORK FOR BEHAVIOUR FUNCTION ANIMATION

This chapter presents the formal specifications for behaviour functions. Several additional concepts are defined that have been found useful for the refinement of the behaviour function concept; these are partial responses and sub-responses. Factors are also introduced that allow for the classification of behaviour functions and responses.

2.1 BEHAVIOUR FUNCTIONS - THEORETICAL BACKGROUND

The micro world which is considered in this thesis is a graphical environment with n distinct objects. Several characteristics of each object are defined; of particular importance is the position vector, $\underline{p}_{i,t}$ of a given object i at time t.

We define a behaviour function B_i as a function which determines the velocity vector (change in position) of an object over a discrete time interval. At time t, the behaviour returned by function B_i is velocity $\underline{v}_{i,t}$. The new position of object i is calculated as:

$$\underline{p}_{i,t} = \underline{p}_{i,t-1} + \underline{v}_{i,t} \qquad \text{(Eq. 2.1)}$$

The set of behaviour functions we are concerned with are those for which the velocity of an object i at time t is a function of the positions of all the objects in the environment, including itself, at times t-1, t-2 ... t-k for some finite number, k, of previous intervals. Thus,

$$\underline{v}_{i,t} = B_i \left(\underline{p}_{1,t-1}, \underline{p}_{1,t-2}, \ldots, \underline{p}_{1,t-k}, \underline{p}_{2,t-1}, \underline{p}_{2,t-2}, \ldots, \underline{p}_{2,t-k}, \ldots, \underline{p}_{n,t-1}, \underline{p}_{n,t-2}, \ldots, \underline{p}_{n,t-k} \right) \quad \text{(Eq. 2.2)}$$

In the investigation reported here we have restricted ourselves to behaviour functions where k is at most 2. In other words, the positions of objects at times t-1 and t-2 are the only variable inputs to the functions. From a practical point of view these functions are the most interesting in that the overhead of storing the previous states of the system is minimized. A major goal of this research is to discover whether a rich and interesting variety of behaviour is available within this limitation.

2.2 PARTIAL RESPONSE FUNCTIONS

Every object in an environment exists both as a stimulus, provoking the behaviours of the other objects and as an actor, behaving in response to its environment. For conceptual simplicity we found it necessary to define a set of 'partial response functions' $R_{i,j}$, which determine the partial response of each actor to each stimulus. We have used the term partial response rather than behaviour because the partial response is only a tendency to behave; partial responses to other objects must be combined at a higher level to determine behaviour. We can express this set of partial response functions as a matrix in which each object appears twice, once as an actor a_i, and once as a stimulus s_j.

$$
\begin{array}{c}
\quad s_1 \quad s_2 \quad \cdots \quad s_n \\
\begin{array}{c} a_1 \\ a_2 \\ \cdot \\ \cdot \\ a_n \end{array}
\begin{bmatrix}
R_{1,1} & R_{1,2} & \cdots & R_{1,n} \\
R_{2,1} & R_{2,2} & \cdots & R_{2,n} \\
\cdot & \cdot & & \cdot \\
\cdot & \cdot & & \cdot \\
R_{n,1} & R_{n,2} & \cdots & R_{n,n}
\end{bmatrix}
\end{array}
$$

The labels on the columns and rows of the matrix identify stimulus and actor objects respectively. The elements of the rows are the partial responses of a given actor to each of the stimuli in its environment -- the entire row represents the elements of the actor's behaviour function. In

practice, many elements of this matrix may be zero, meaning that an actor never responds to a given stimulus -- where an entire row is zero, the object would be static and act as a stimulus to other objects only. The fact that the behaviour matrix is square means that the computational complexity of behaviour-function animation is inherently $O(n^2)$ -- in practice this worst-case is reduced by the presence of zeros making the matrix sparse.

At a higher level the organism must select among the various stimuli in its environment and decide which are to be responded to at a given instant. In principle the function combining partial responses to form the complete behaviour could be of arbitrary complexity. We chose, for the sake of simplicity, to sum the values returned by the partial response functions. Thus, the subset of functions we have studied most extensively considers the complete behaviour as the sum of the independent responses, $R_{i,j}$, to each object. The behaviour function B_i in this subset can therefore be expressed as:

$$\underline{v}_{i,t} = \sum_{j=1}^{n} R_{i,j}\left(\underline{p}_{i,t-1}, \underline{p}_{i,t-2}, \underline{p}_{j,t-1}, \underline{p}_{j,t-2}\right) \qquad \text{(Eq. 2.3)}$$

where the object i is the actor and the objects j are the stimuli.

2.3 SUB-RESPONSES

Thus far we have introduced a two-level hierarchy where a behaviour function is made up of a number of partial response functions, designating the response of an object to each other object in its environment. In fact we found it necessary to subdivide even further so that partial responses were themselves composite functions consisting of 'sub-responses'.

The idea of a sub-response can easily be given by an example: Curiosity might cause an animal to approach an unknown object, and fear of the unknown might cause it to withdraw. Curiosity tends to dominate while the object is distant, fear tends to dominate with closer proximity. Very often radically different behavioural forces exist at different distances, and together they make up the partial response of an actor to an object. These components of partial responses are what we call sub-responses. Specifically the partial response functions we have investigated are of the following form:

$$\underset{i,j,t}{\underline{v}} = \sum_{r=1}^{s} (M_{i,j,t,r} + c_{i,j,r})\, b_{i,j,r}\, \underset{i,j,t,r}{\underline{D}} \qquad \text{(Eq. 2.4)}$$

where each of the expressions (subscripted by r) being summed is a sub-response. The M are magnitudes and the $\underline{D}$

are direction unit vectors; all are computed using $\underline{p}_{i,t-1}$, $\underline{p}_{i,t-2}$, $\underline{p}_{j,t-1}$, $\underline{p}_{j,t-2}$. The finite list of time-dependent magnitudes and directions is discussed in the next section. The c and b are variables that allow the fine-tuning of the partial response. It has been found in general that where some generic effect can be achieved by using different combinations of magnitudes and directions, a specific effect may be generated by appropriate values for c and b. b is composed of two parts -- a response-distinguishing constant m_r and a weighting value u_r:

$$b_{i,j,r} = m_{i,j,r} \; u_{i,j,r} \qquad \text{(Eq. 2.5)}$$

The $u_{j,r}$ of all sub-responses in a behaviour function are guaranteed to sum to one, i.e. for a behaviour function B_i,

$$\sum_{j=1}^{n} \sum_{r=1}^{n} u_{i,j,r} = 1 \qquad \text{(Eq. 2.6)}$$

The weighting factor allows some sub-responses to be powerful and others to have only negligible effect on the resulting behaviour. As is seen in the next chapter, the weight can be based on a function of some time-dependent magnitude from the same set as M came from. A special case is where the weight is conceptually set to zero if the magnitude is

outside some range -- this is implemented using conditional expressions to determine which sub-responses is executed.

2.4 CLASSIFICATION OF BEHAVIOUR FUNCTIONS AND RESPONSES

There are several ways that behaviour functions could be classified. One fundamental method is based on the maximum number of time-steps involved in the calculations. Thus, we call a behaviour function which determines behaviour based only on the positions of objects at time t-1 a T1 behaviour function. A behaviour function in which both t-1 and t-2 environments are taken into account is a T2 behaviour, and so on. As mentioned above, we have only investigated T1 and T2 behaviours. The T1-T2 classification can be expanded by considering which magnitude and direction components go into behaviour functions -- the components are discussed below and much of the discussion in chapter 5 uses this method of classification.

From the observer's point-of-view, behaviours can be classified in order of increasing animacy. At the low end of the scale are the purely mechanical behaviours that are necessary so that objects obey fundamental physical laws. At the high end of the scale are behaviours that appear increasingly complex and purposeful. Note that those behaviours that 'appear' to be increasingly complex need not be based on complex behaviour functions. We have created such

things as the appearance of 'teasing' with functions which are relatively simple.

2.5 TIME-DEPENDENT MAGNITUDES AND DIRECTIONS

In the T1 case, the only quantities available for computation of responses are the distance and the inter-object direction. In the T2 case, on the other hand, a much wider variety of time-dependent quantities exist; these include the velocity and direction of the actor and stimulus, the relative motion and the change in distance. When studying T2 responses, many different combinations of magnitudes and directions were considered.

The velocity vector resulting from a behaviour is a function of the magnitudes and directions used as input to the constituent responses. For an individual sub-response, the direction of the response vector is the same as the direction input (if the magnitude is negative, the direction appears reversed though); it is only the magnitude of the output vector that is affected by the additive and multiplicative parameters as well as the input magnitude. When several sub-responses are summed however, both the direction and magnitude of the output vector are dependent on all factors: input magnitudes, input directions, additive and multiplicative parameters and weights.

T	Keyword	Method of Calculation
		FUNDAMENTAL QUANTITIES
1	distance	Euclidean distance between perimeters
2	velocity	Velocity of actor
2	othvelocity	Velocity of stimulus (other's velocity)
2	changedist	Change in distance
2	relvelocity	Relative velocity
		COMPOUND QUANTITIES
2	timetoreach	Time-to-reach; distance/changedist
2	appfixedness	Apparent fixedness; distance/othvelocity
2	stillness	Stillness; distance/relvelocity
		INVERSE-SQUARE QUANTITIES
1	idistance	Inverse-square distance between centres
2	ichangedist	Change in inverse-square distance
2	itimetoreach	Inverse-square distance/changedist

Figure 2.1 -- Time-Dependent Magnitudes

Figure 2.1 lists the magnitudes used in sub-responses for the work of this thesis, and figure 2.2 lists the directions. The first column of each table shows whether the item is T1 or T2; the classification of a sub-response is based on the highest classification of its components, hence a sub-response can only be classed as T1 if both its magnitude and direction are T1.

The second column of the tables shows the symbolic keyword chosen to represent each quantity -- these are used in the language which is described in the next chapter.

The third column describes the method of calculation of the quantities. Some quantities, termed fundamental, are calculated directly from positions or from subtraction of other fundamental quantities. The compound quantities are ratios of two fundamental quantities. The inverse-square magnitudes are closely related to other magnitudes -- they involve inverse-square distance instead of Euclidean distance.

In figure 2.2, the pairs towards-away, forwards-backwards and with-against are inverses of each other. In one sense it is redundant to allow both directions in each pair because the second member of the pair could always be generated from the first by specifying a negative multiplicative parameter; however, away, backwards and against are simple intuitive concepts and it was decided to allow them for the convenience of the user.

Most fundamental quantities come in natural magnitude-direction pairs: distance-towards, velocity-forwards and relvelocity-moverelative are examples. Each pair can be combined to form a vector that is meaningful in the physical world (the vector of absolute movement, the vector between two objects, and the vector of relative movement, respectively). The compound quantities do not, in general, come in pairs.

T	Keyword	Method of Calculation
		FUNDAMENTAL QUANTITIES
1	towards	Direction between nearest points
1	away	Negative towards
2	forwards	Direction of movement
2	backwards	Negative forwards
2	with	Direction of movement of stimulus
2	against	Negative with
2	moverelative	Direction of relative movement
		COMPOUND QUANTITIES
2	reflection	Equal to angle of incidence
2	orbital	Parallel to normal
2	intercept	Direction to optimize interception

Figure 2.2 -- Time-Dependent Directions

Complete details of the algorithms for computing all the quantities in figures 2.1 and 2.2 can be found in the technical documentation for PAM -- the general philosophy is also discussed in chapter 4.

Chapter 3.

THE ANIMATION DESCRIPTION LANGUAGE

A simple language was developed to describe the environment and behaviours. This may be called the 'assembly language' of behaviour function animation, since there is a logical correspondence between the statements and the equations of chapter 2.

The following illustrative example is the complete description of an environment using this language:

```
wraparound                                          | SCALARS

object 1                                            |
    circrad 40                                      |
    xcent 100 ycent 200                             |
    dx 12 dy 11                                     |
obend                                               |
                                                    | OBJECTS
object 2                                            |
    xhalfdim 35.5 yhalfdim 35.5                     |
    xcent 900 ycent 600                             |
    dx -13 dy -10                                   |
obend                                               |

partresp 1                                          |
    if (timetoreach < .5)                           |
        return(velocity reflection)                 | PARTIAL
    else                                            | RESPONSES
        return(velocity forwards)                   |
prend                                               |

bmatrix                                             |
    0 1                                             | BEHAVIOUR
    1 0                                             | MATRIX
```

The language statements are on the left and the main sections of the language are indicated on the right. The scalars section defines the general mode of operation. The objects section describes the shape, colour and initial movement of the objects. The partial responses section defines various types of action that may be taken by an actor in response to a stimulus; and the behaviour matrix section relates each actor/stimulus pair to a partial response. If the above example were executed, the user would see a square and a circle moving around and bouncing off each other.

The following description shows the concrete syntax of the language in BNF along with some descriptive material and examples; more complete details may be found in the PAM user guide and technical description documents.[1] In the BNF definitions, <fval> means a floating-point value, <ival> means an integer value and <sval> means a character string value. All of the keywords are unique to three characters.

There are four main sections in the description of every environment:

```
<environment> ::=
   <scalars>
   <objects>
   <partial_responses>
   <behaviour_matrix>
```

---

1. Lethbridge T., 1987

These sections are indicated on the right hand side of the example given above.

3.1 SCALARS SECTION

The scalars section allows for definition of general attributes of the environment and has the following syntax:

```
<scalars> ::=
      <null>
    | <scalar_item>
    | <scalar_item> <scalars>

<scalar_item> ::=
      halt
    | speed <ival>
    | protection
    | cyclimit <ival>
    | mouseactive
    | nextenv <sval>
    | wraparound
```

Each scalar item keyword may only be specified once in an environment. The first three keywords only affect the user interface. Cyclimit and nextenv allow for the chaining of environments and the showing of short sequences. Mouseactive allows one object to be under the control of the user and wraparound creates a situation where if an object passes through one side of the screen it comes back in through the other side.

3.2 OBJECTS SECTION

The objects section allows for the definition of objects which may later be made to interact with each other:

```
<objects> ::=
      <object_definition>
    | <object_definition> <objects>

<object_definition> ::=
    object <ival>
    <object_description>
    obend

<object_description> ::=
      <object_parameter>
    | <object_parameter> <object_description>

<object_parameter> ::=
      colour <ival> | color <ival>
    | circradius <fval>
    | xhalfdim <fval> | yhalfdim <fval>
    | xcentre <fval> | ycentre <fval>
    | dx <fval> | dy <fval> | static
```

Each object parameter keyword may only be specified once for each object and where a parameter is absent a default is assumed. The colour parameter allows the choice of a colour from 1 to 15 with 15 being invisible. The radius and half parameters define the shape; Figure 3.1 shows the effect of the presence or absence of these parameters. In all cases, the shape is symmetrical about both x and y axes, is convex and its perimeter is composed only of horizontal and vertical lines and quarter-arcs of circles. The shape parameters may be negative, indicating that the object is a 'hole' meaning that another object would be inside its mass if it were outside its apparent perimeter.

Value Present			Resulting Shape
x	y	circ	
N	N	N	Point
N	N	Y	Circle
N	Y	N	Horizontal line
N	Y	Y	Horizontal bar with semicircular ends
Y	N	N	Vertical Line
Y	N	Y	Vertical bar with semicircular ends
Y	Y	N	Rectangle
Y	Y	Y	Rectangle with rounded corners

Figure 3.1 -- Effect of Shape Parameters

An example of an objects section might be:

```
object 1
    colour 2
    circrad 30
    xcent 50 ycent 50
    dx 1 dy 1
obend

object 2
    colour 3
    xhalfdim 100 yhalfdim 12
    xcent 512 ycent 600
    static
obend
```

This would define a red circle in the bottom left-hand corner of the screen with an initial velocity towards the top-right, as well as a green horizontal bar near the top of the screen that never moves.

3.3 PARTIAL RESPONSES SECTION

The partial responses section is used to create a series of partial responses that can be used to relate any actor-stimulus pair. The syntax is as follows[2]:

```
<partial_responses> ::=
      <p_response_definition>
    | <p_response_definition> <partial_responses>

<p_response_definition> ::=
    partresp <ival>
    <p_response_description>
    prend

<p_response_description> ::=
      return(<sub_response>)
    | return(<sub_response>)
      <p_response_description>
    | if <condition> return(<sub_response>)
    | if <condition> return(<sub_response>)
      <p_response_description>
    | if <condition> return(<sub_response>)
      else <p_response_description>

<sub_response> ::=
    <magnitude_section>
    <direction_section>
    <weight_section>

<magnitude_section> ::=
      <magnitude>
    | <magnitude> * <fval>
    | <magnitude> + <fval>
    | (<magnitude> + <fval>) * <fval>
```

---

2. The syntax for this section is actually slightly different from that which PAM uses. Since the work of this thesis does not involve compiler-writing, the PAM syntax is made somewhat easier to parse but remains functionally identical. Appendix A contains an inter-language mapping.

```
<magnitude> ::=
      distance
    | velocity
    | othvelocity
    | changedist
    | relvelocity
    | timetoreach
    | appfixedness
    | stillness
    | idistance
    | ichangedist
    | itimetoreach

<direction_section> ::=
      <null>
    | forwards
    | backwards
    | with
    | against
    | towards
    | awayfrom
    | moverelative
    | reflection
    | orbital
    | interception

<weight_section> ::=
      <null>
    | weight <fval>

<condition> ::=
      <magnitude> > <fval>
    | <magnitude> < <fval>
    | <magnitude> > <fval> & < <fval>
```

Note that the magnitudes and directions are the same as those found in the tables in the previous chapter. The weights given in sub-responses need not add to one -- this is done internally.

An example of a partial response description might be:

```
if timetoreach < .5
    return(velocity * .9 reflection weight 100)
else
    if distance < 100
        return(velocity * 1.01 away)
    else
        if distance < 300
            return(velocity * 1.01 towards)
```

In English sentences, this is saying that if the actor object is about to collide with the stimulus it is very important that it bounce off. Otherwise, if the distance is less than 100 units, the actor should be repelled from the stimulus. In the final case, if the distance is less than 300 units, the actor should be attracted. Visually, we see the actor mind its own business until the stimulus comes within range, then the actor starts to accelerate towards the stimulus. At a distance of 100 units it begins to decelerate, and ultimately accelerates away again. If the actor is moving fast, however, we may see inadequate 'braking' followed by a collision. On an on-going basis we may see the actor moving alternately towards and away from the stimulus. The overall impression depends, though, on exactly what the stimulus is doing.

3.4 BEHAVIOUR MATRIX SECTION

The behaviour matrix section occurs once at the end of every environment; it has the following syntax:

```
<behaviour_matrix> ::=
      <sub_format>
    | <full_format>

<sub_format> ::=
    bsubmatrix
    <dimensions>
    <column_labels>
    <labelled_rows>

<dimensions> ::=
    <ival> <ival>

<column_labels> ::=
      <ival>
    | <ival> <column_labels>

<labelled_rows> ::=
      <ival> <behaviour_function>
    | <ival> <behaviour_function> <labelled_rows>

<behaviour_function> ::=
      <ival>
    | <ival> <behaviour_function>

<full_format> ::=
    bmatrix
    <unlabelled_rows>

<unlabelled-rows> ::=
      <behaviour_function>
    | <behaviour_function> <unlabelled_rows>
```

The integers in the <behaviour_function> definition refer to partial response numbers which relate an actor to a stimulus. The row and column labels in the sub-format are the numbers of objects to be related. The dimensions in the sub-format indicate the number of objects acting as actors and stimuli.

The following example of a behaviour matrix section shows two actor objects (5 and 8) reacting to three stimulus

objects (1, 5 and 7). There are four types of partial response involved (2, 3, 4 and 5). Both actors respond to stimulus 1 with partial response 2, and both respond to stimulus 7 with response 5. Object 5 responds to itself with response 3 and object 8 responds to stimulus 5 with response 4.

```
bsubmatrix 2 3

           1  5  7

    5      2  3  5
    8      2  4  5
```

Chapter 4.

THE SOFTWARE

This chapter is an attempt to present the implementation of a behaviour-function oriented animation system in a general way. The discussion is primarily about the PAM (Perceived Animate Motion) implementation made for this thesis, with an eye to concepts necessary for implementations in general. It should be noted that a full-scale commercial implementation would most likely combine features of other animation systems along with behaviour functions -- this would significantly complicate the issues and hence the discussion here is limited to a 'pure' behaviour-function implementation.

The PAM implementation was written in C with library calls for graphics output and other device interface requirements. The hardware used was a Silicon Graphics IRIS 2400 with a floating point accelerator. PAM is dealt with in much more detail in the technical description document[1].

4.1 THE FUNDAMENTAL STRUCTURE

At the core of any behaviour function system must be a set of routines that take the status of objects, apply the

---

1. Lethbridge T., PAM Technical Documentation, 1987

behaviour functions and output the new status. Indeed this is all that is needed for a bare minimum system since the behaviour-functions could be hard-coded and the displaying of objects at the output positions could be left to other software or hardware. The minimal data structure needed would be the positions of objects at the previous two time steps (assuming that only T1 and T2 behaviours are involved). We return to this environment updating module after looking at the broader aspects of implementation.

For a fully functional system there must be two other modules: Routines are needed to load behaviour functions from some external description and possibly modify the behaviour functions at intermittent intervals, since if behaviour functions were hard-coded the system would be highly restricted. Also, routines are needed to process the output of the environment update module -- either to display the objects on a screen or record them somewhere else for later display.

Figure 4.1 shows the relationship of the three main modules in terms of the flow of control. The load/modify module need only be executed when the program first starts although occasional execution might be needed later. The environment update module and the display/record module must logically be executed repeatedly and alternately, one cycle for every time-step or frame. These two modules can there-

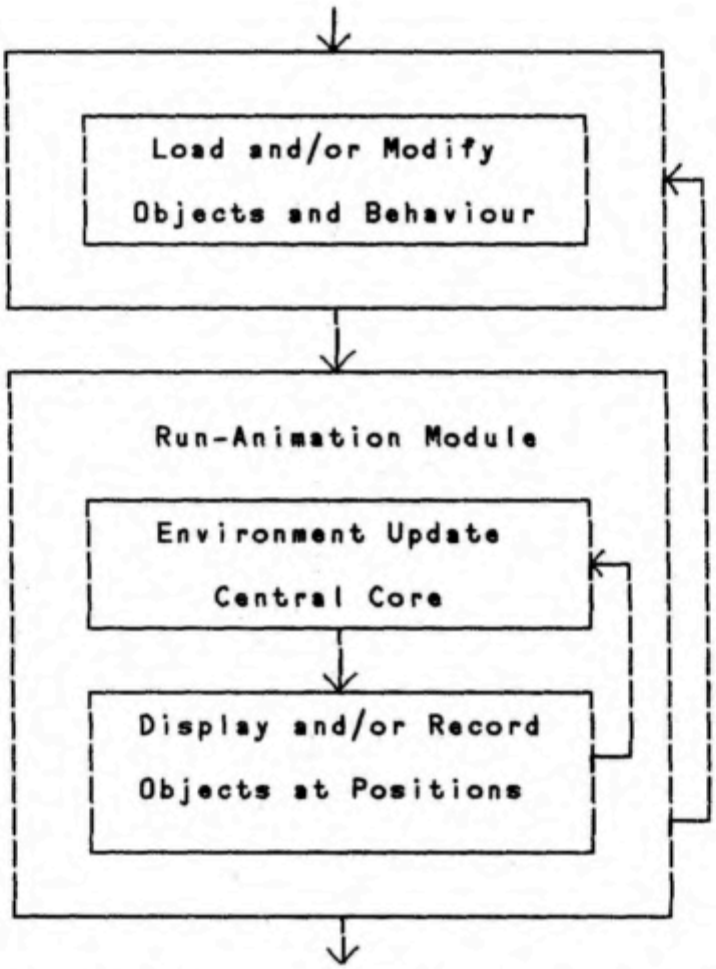

Figure 4.1 -- Modules in Behaviour-Function Animation

fore be conceptually combined into a single run-animation module.

Behaviour function animation is computationally intensive due to the necessity to compute large numbers of frames. This is exacerbated by the fact that, at its worst, it is an n^2 process. As a result efficiency is a paramount concern, especially if the frames are to be displayed in real time. For real-time animation, a fresh display must be created at least every 1/15 of a second for the human brain

to have the illusion of continuity. Since monitors have a refresh rate of 1/33 of a second the design must strive to allow iterations of the run-animation module at this frequency.

In the PAM system it was found necessary to do a maximum of pre-processing in the load/modify module and optimize the operation of the run-animation module to achieve operation as close to real-time as possible. The discussion in subsequent sections deals with each of these modules in turn.

It should be mentioned that in addition to the modules described above there is additional initialization and termination work required in PAM, primarily for the graphics system and the user interface. The complete PAM system can therefore be best described by figure 4.2.

4.2 THE LOAD/MODIFY MODULE

This module could be arbitrarily complex; if behaviour-function animation were to be combined with other methods of animation, this would be where the extra complexity would lie. For the PAM system it was decided to simply make this module read a series of environment files containing the animation description language described in the previous chapter, and create data structures optimized for the run-

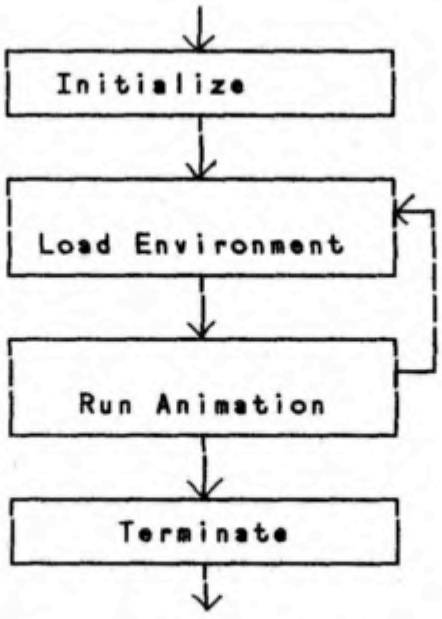

Figure 4.2 -- Top-Level Diagram of PAM

animation module. No facilities were incorporated to allow interactive modification of the environment although such a facility would be a logical and useful enhancement. To update an environment using PAM, the user must edit the language description (using vi[2] or some other UNIX text editor) and then re-load it without restarting the program -- PAM provides a convenient one-button reload capability.

Figure 4.3 shows an intermediate-level breakdown of the environment loading module. This method of structuring has been found to provide the best combination of structured design and efficiency. Although this structure specifically represents PAM, it seems applicable to any behaviour-

---

2. Joy W. and Horton M., 1984

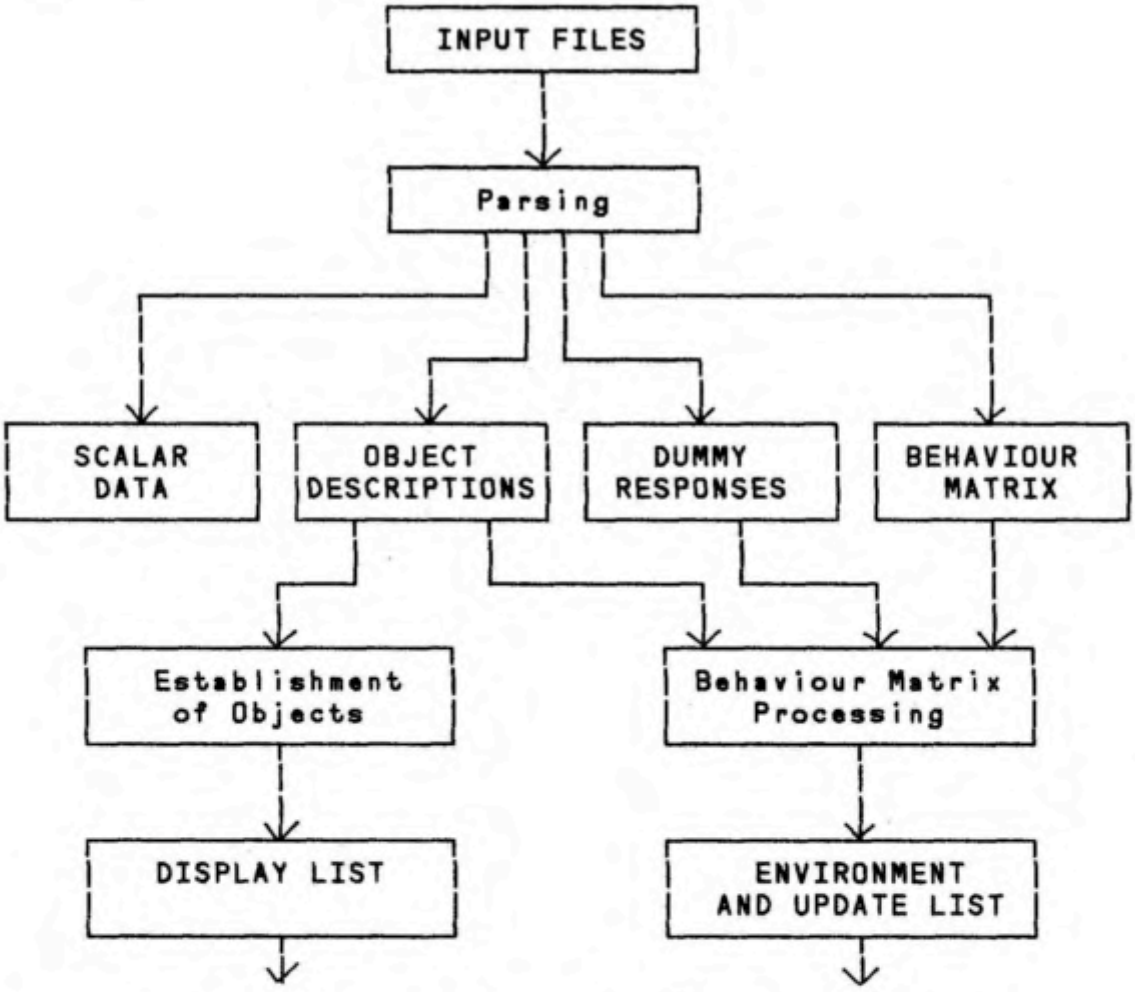

Figure 4.3 -- The Environment-Loading Module

function animation system that does not allow changes to the environment after loading.

The first process is to parse the environment files and generate data structures representing each of the four sections of the language: scalars, objects, partial responses and the behaviour matrix. This intermediate-level data is in a format similar to that in the environment files. The partial responses are termed 'dummy' because they

are merely templates -- they have not yet been applied to any specific actor-stimulus pair. In PAM, the behaviour matrix is not actually stored at the intermediate level because the process of parsing is tied directly to the next process -- producing the structures required for the run-animation module.

Two separate structures are built that allow the system to run as fast as possible in the run-animation module. The display list is a list of graphics commands stored for most efficient display. The update list is a program to optimally update the environment and associated data. Two options were considered for this, a series of data items processed by an interpreter or a routine especially compiled for the environment. For the PAM implementation, the interpreting option was chosen for simplicity; the update list is integrated with the environment data and is composed primarily of pointers to data that must be updated and pointers to functions that do the updating. The environment data consists of the positions and velocity vectors of moving objects plus storage space for whatever other quantities may need to be calculated (magnitudes, directions etc.) along with flags indicating whether they have been updated or not at a given time-step

Many things can be done to optimize the update list and environment data. An example is the pre-multiplying of weights by multiplicative factors.

4.3 THE RUN-ANIMATION MODULE

This module runs in a loop performing the main operations -- environment update and display of objects -- described earlier. Additionally, this is where any user-interface routines would be placed; PAM, for example checks once every time step to see if any button is pressed and, if so, performs one of the tasks outlined in the user documentation such as allowing the mouse to control an object or allowing objects to trace out paths so still photographs may be taken.

4.3.1 The Environment Update Module

The environment update module, as mentioned earlier, is the central core. It can be divided into two phases: the first phase involves proceeding down the update list processing each partial response, and the second phase involves the computation of finalized behaviours and position vectors for each actor from partial responses. Figure 4.4 summarizes the environment update module and the following paragraphs expand on some details.

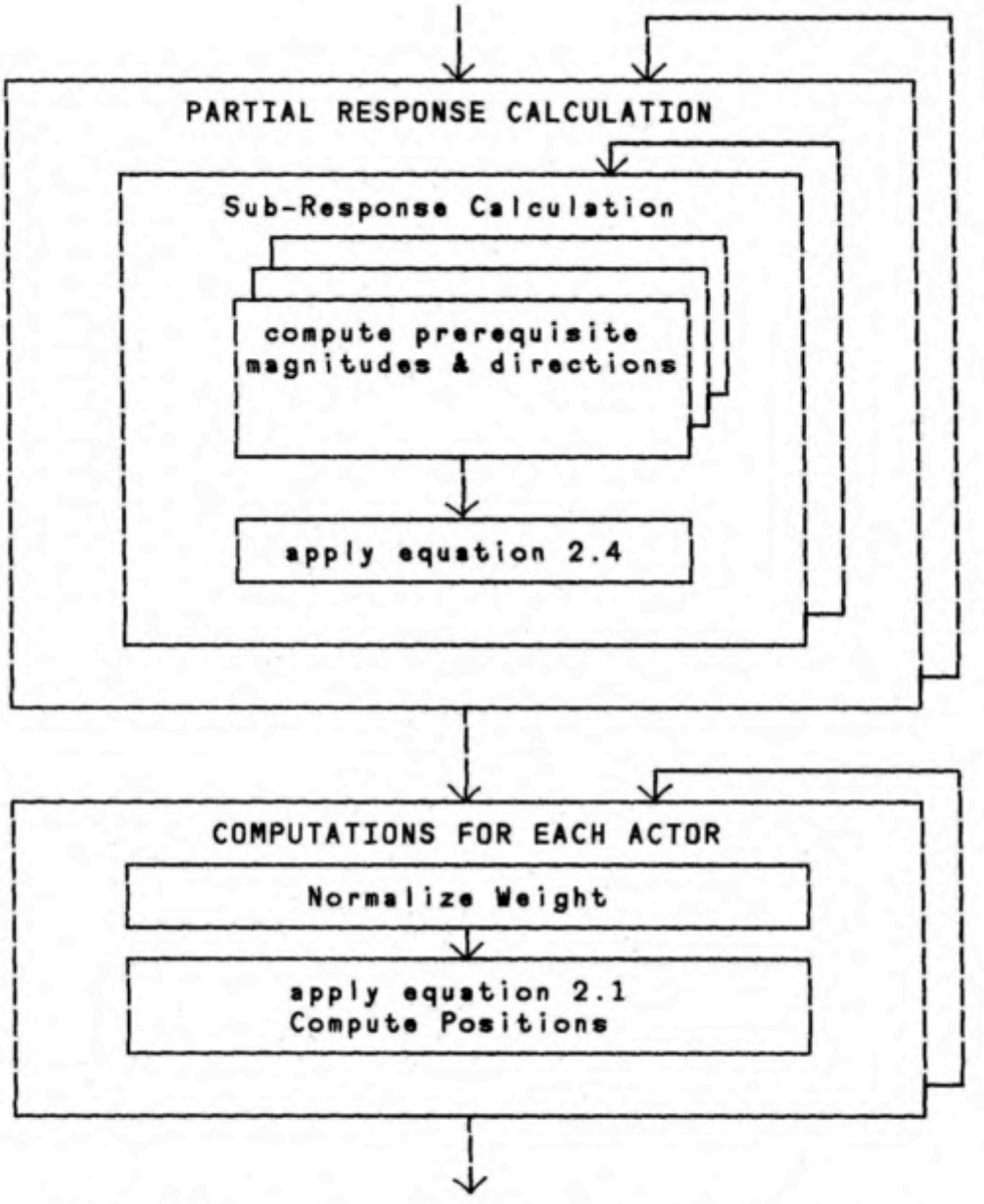

Figure 4.4 -- The Environment Update Module

All the sub-responses are processed in sequence by following the update list. The update list is actually broken down into sublists, one for each partial response; the elements of the sublists are the sub-responses. The basic computation for a sub-response is very simple, it is a mat-

ter of applying equation 4 from chapter 2. Before this can be done though, the time-dependent magnitudes and directions required for the response must be calculated -- this turns out to be the most time-consuming part of the whole system.

Often a given magnitude or direction will have already been found by a previous sub-response in the same partial response or by a sub-response in the inverse partial response (if the actor-stimulus roles are reversed many quantities are the same or the negative -- distance remains the same, while the towards direction is the negative). Much redundant computation is saved by flagging every value once it has been updated for a given time-step.

If a magnitude or direction has not been found, then it must be updated using an appropriate algorithm. In PAM, the update list contains pointers to a series of functions to perform these algorithms. Most of the algorithms are very simple; the most important and one of the more complex algorithms is that which computes the Euclidean distance between perimeters and the towards direction at the same time -- full details about this are in the technical documentation.

All of the algorithms have prerequisite data. The only prerequisite data that is guaranteed to be up-to-date are the positions of objects at the previous two time steps and the velocity vector at the previous time step. (All that

strictly need to be saved for T2 behaviours are the two sets of positions. Any value that is computed at the previous time step and may be needed at the current time step is saved. Since velocity is always computed, it is always available.) A given algorithm may have to call other algorithms to find other prerequisites, and these may in turn call further algorithms until finally all data is available. A network graph can be drawn showing the relationships of all algorithms and the prerequisites they require back to the position vectors. Since the whole network is somewhat complex, a sub-network only is shown in figure 4.5. This shows the prerequisites required to find time-to-reach at the current time step (t): Before time-to-reach can be found, the distance and the change in distance must be available; these in turn have their own prerequisite quantities.

As described in chapter two, when the responses are calculated, the result velocities are summed for each actor to yield a velocity vector for this time step. The most heavily weighted responses contribute more to the sum than the less-weighted responses. In the model all the weights must sum to one; this is best handled by summing the weights for each actor as the responses are calculated, and then normalizing the velocity vector by dividing by the weight sum. The user, therefore, need not be concerned that weights they supply sum to one.

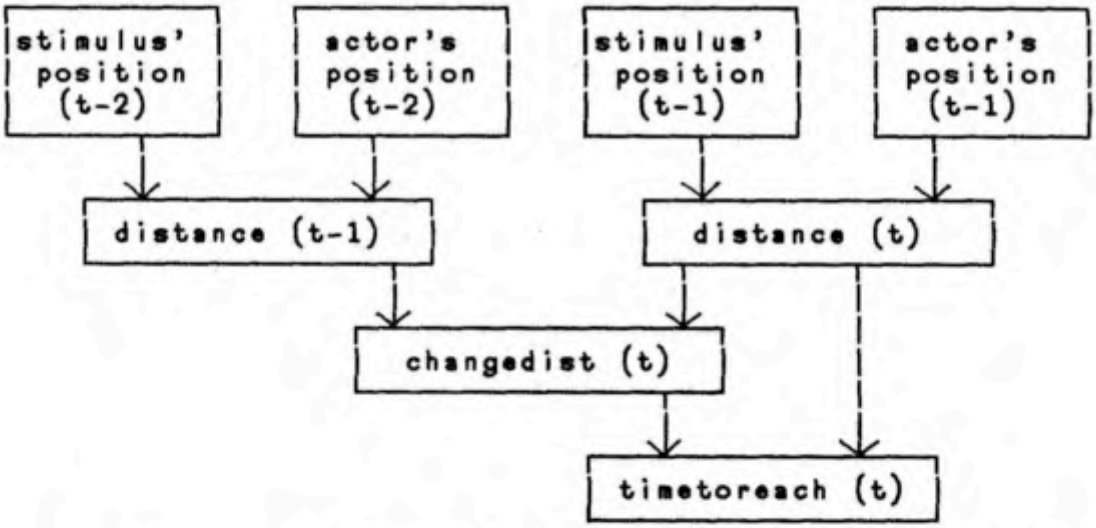

Figure 4.5 -- Prerequisite Network for Time to Reach

The final step in the environment update module is to apply equation 2.1 in order to determine the final positions of each actor. This simply involves adding the velocity to the previous position. (Note that this does not involve the adding of incompatible units because the velocity is the change in position for a single time step.)

4.3.2 The Display/Record Module

This module has two alternative but simple tasks. If it is to display objects, it must update the display list to reflect the new positions of the objects and then must cause the display list to display graphic output. If the module is to record the positions of objects, it must allocate space in memory for a record or write a record to a a file. PAM always displays data although it can be made to do this less

frequently than once a time-step. PAM is also capable of recording positions both in memory or in a file at the request of the user.

Since many objects in an environment may only have the role of stimulus, there is no need to update the positions of these static objects in the display list. To improve efficiency, static and non-static objects are represented differently in the display list. As in most graphics environments, an object in PAM is represented by a series of function calls. Static objects are centered on the screen at their absolute locations by these function calls. Non-static objects on the other hand are centered at the origin and a single translation is done before each object is displayed -- it is this translation that is modified at each time step to cause the object to move.

Chapter 5.

FINDINGS: THE GENERATION OF BEHAVIOUR

In this chapter we look at a wide range of simple categories of response from both a mathematical and perceptual point of view. A study is made of the categories of simple sub-response discussed in chapter 2 as well as interesting combinations of these. A look is also taken at some of the problems that can occur when dealing with behaviour functions, and how to deal with these problems.

The findings reported here were obtained by a combination of trial-and-error experimentation using the PAM implementation, and mathematical deduction. The general procedure was to start with a very basic version of the implementation and add features when it was determined they might serve a useful purpose. When patterns were observed or hypothesized, an attempt was made to formalize these mathematically.

Some of the results are of an objective nature and others are somewhat subjective. The subjective observations are the author's impressions and may be interpreted slightly differently by others. The objective results, which form the bulk of the findings, have been repeatedly tested to determine their rigorousness, but are nevertheless due to

preliminary analysis on the part of the author. As research progresses, the relative importance of various findings is likely to alter and useful findings may come out of areas of investigation that appeared barren to the author.

Figure 5.1 shows how the fundamental and complex time-dependent magnitudes and directions can be combined to form a matrix of 56 different types of sub-response. Some of these have been found to be broadly useful, and others less so. Reverse directions and inverse-square magnitudes have been left out for simplicity. It should be remembered that this table is by no means an exhaustive list of sub-response types because although the fundamental quantities form an exhaustive list (if only two time steps are considered), there are many more possible complex quantities. The numbers in the boxes in figure 5.1 show how useful the various combinations were found to be on a scale of increasing usefulness from 1 to 10. The numbers by the magnitudes indicate their usefulness as range parameters in conditions.

5.1 T1 RESPONSES

Of the 56 table entries in figure 5.1, only the top left element is T1; the other 55 are T2 (see chapter 2 for the basis of the T1/T2 classification). In general T1 responses can model situations where one object gravitates to a certain distance from another. More specifically, we have used

usefulness: 10 = essential 5 = sometimes useful 0 = useless DIRECTION MAGNITUDE		towards	forwards	with	moverelative	reflection	orbital	intercept
distance	10	10	9	7	2	2	6	3
velocity	10	8	10	8	2	9	7	8
othvelocity	8	8	6	7	1	1	5	2
changedist	6	3	4	4	1	0	4	4
relvelocity	1	2	1	1	1	0	1	1
timetoreach	9	8	8	6	1	3	3	7
appfixedness	7	2	3	3	0	0	2	1
stillness	1	1	1	0	1	0	0	0

Figure 5.1 -- Categorization of Sub-Responses

them for clinging, pushing, pulling, chasing, attraction, escaping and repulsion.

The only magnitudes that can be used in a T1 response are distances, since these are all that can be calculated from the positions of objects at the current time step only. (Referring to equation 2.3, note that in the T1 case there are only two, not four, position inputs to response functions.) For now only Euclidean distance is considered;

inverse-square distance has comparable properties and is discussed later. The only T1 direction is the direction between the actor and stimulus; the towards and away directions are both manifestations of the same thing and can be interchanged by negating the multiplicative parameter. The towards direction seems more intuitive to use for inter-object direction, but the away direction has the nice property that when used with a positive magnitude it causes an increase in distance and when used with a negative magnitude the distance decreases.

Since the only input to equation 2.4 is distance, in a simple two-object T1 system the actor's velocity towards or away from the stimulus must be a linear function of distance. In order to design a T1 partial response we must first decide what the velocity should be at various distances. Figure 5.2 shows a typical graph with distance on the x-axis and the desired away velocity on the y-axis (towards velocities are represented as negative). Linear segments with slopes m1, m2 and m3 have been drawn between distance-velocity points. This graph indicates that at distances d0, d1, d2 and d3 the velocities should be v0, v1, v2 and v3 respectively (v1 and v3 coincide at the x-axis). In this typical graph, the actor's away velocity would decrease as distance increased from d0 to d1 with v1 being zero. From d1 to d2, the velocity would increase in the towards direction, and then decrease again from d2 to d3. In reality, the

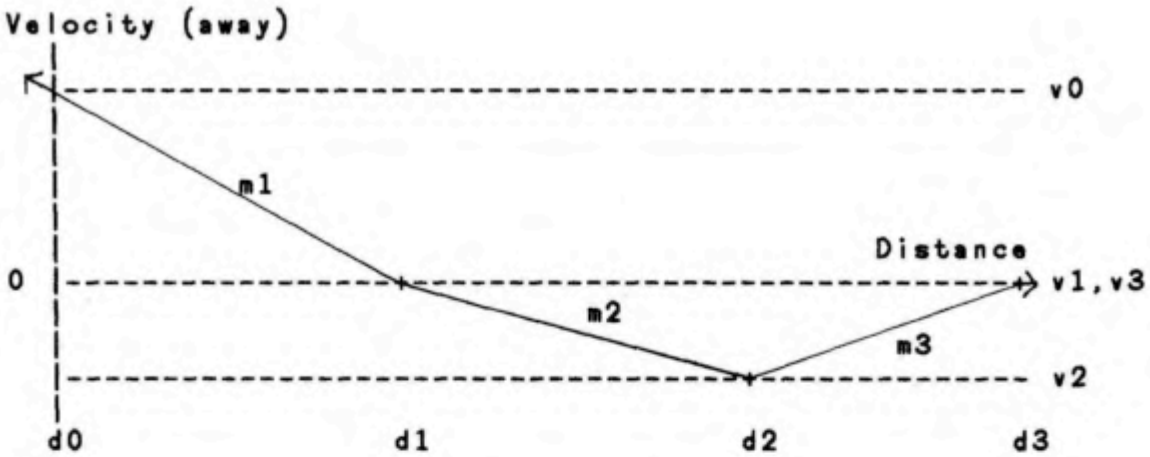

Figure 5.2 -- Velocity vs. Distance in a T1 Partial Response

graph extends beyond d0 and d3 to negative infinity and infinity -- when designing a T1 partial response, these extensions must also be considered.

A plan similar to figure 5.2 is essential if a T1 partial response is to yield reasonable behaviour. The following guidelines should be followed when setting up such a response:

1) Any number of linear segments may be used, but the curve should generally be continuous to prevent unnatural movements or oscillation -- these problems and others are discussed later in the chapter. (Small discontinuities may cause unnatural or jerky movements that are desirable because they can enhance the impression of intentionality, but the high risk of oscillation from such discontinuities makes

it more practical to cause abrupt movements by juxtaposition of segments of shallow and steeper slopes). It should be noted that the graph need not necessarily have segments both above and below the x-axis.

2) The left and right extensions must always have a negative or zero slope. Additionally, the velocity in the left extension must at some point be non-negative and the velocity in the right extension must at some point be non-positive. These guidelines are essential in order to keep the response from going out of control or oscillating. Typically the left extension, into the object, is a continuation of the first segment, and the right extension has a slope and velocity of zero (the response has no influence in this range).

3) The maximum velocity generated at any distance where the actor might exist should not be too fast; at the same time, the mean velocity over distances where the actor might exist should be synchronized with the rest of the environment. If the actor is not expected to ever exist inside the stimulus, then the velocity in that part of the graph could be made arbitrarily high. Likewise, if the actor is never expected to get very far from the stimulus (a condition that normally cannot be relied on) the graph velocity could be allowed to become high at great distance.

Each of the segments of the graph can be expressed as a linear equation with an additive parameter c and a multiplicative parameter m. These translate directly into subresponses of the general form:

$$\text{if distance} < d_r$$
$$\text{return}((\text{distance} - c_r) * m_r \text{ away})$$

The subscript r indicates the particular segment or subresponse. The multiplicative parameter is the slope of the curve segment and can be computed as follows:

$$m_i = (v_r - v_{r-1}) / (d_r - d_{r-1}) \qquad \text{(Eq. 5.1)}$$

It controls the responsiveness and speed of the actor. The additive parameter is the negative x-intercept of the curve segment and can be computed as follows:

$$c_r = v_r / m_r - d_r \qquad \text{(Eq. 5.2)}$$

Both parameters used together control whether the actor is repelled or attracted to the stimulus and in what ranges.

Figure 5.3 shows values for a typical T1 partial response. Note that the above guidelines have been followed;

r	d	v	m	c
0	0	12		
1	80	0	-0.15	-80
2	200	-10	-0.083	-80
3	500	0	0.033	-500

Figure 5.3 -- Typical Calculations for a T1 Response

in particular, the rightmost velocity is zero and the leftmost velocity is positive. There is no sub-response 0, but the distance and velocity 0 are needed for the calculations. The underlined numbers, the m and c parameter columns, were found by solving the equations; if the equations are solved for successively increasing distances, continuity of the segment-curve is guaranteed.

The complete partial response translated from the computations in figure 5.3 is as follows:

```
if distance < 80
    return((distance - 80) * -0.15 away)
else
    if distance < 200
        return((distance - 80) * -0.083 away)
    else
        if distance < 500
            return((distance - 500) * 0.033 away)
```

Since this partial response contains no sub-response effective above 500 units of distance, the velocity in that range will be zero (ignoring the effects of other partial re-

sponses). Also, since the range of the first sub-response encompasses all distances from 80 down to negative infinity, the first segment is extended to distances inside the stimulus. Both of these facts conform with the guidelines.

The above response is decreasingly repulsive to 80 units of distance, increasingly attractive to 200 units and then decreasingly attractive to 500 units. For T1 responses in general, the actor will gravitate to the distance at which the velocity segment-curve crosses the x-axis; in this example that would be 80 units from the stimulus.

The overall effect of a simple T1 response depends on what the stimulus is doing. One of the simplest behaviour patterns that can be created using two objects is persuit. If two actors respond to each other with T1 functions that have differing stable distances, the actor with the response whose stable distance is smaller will become the persuer and the other actor (the stimulus to the first) will become the quarry. This type of persuit alone does not appear animate, in fact the two objects will maintain equilibria of distance, velocity and direction if there are no further influences. For simple T1 systems, equilibrium distance and velocity can be computed from the following formulas:

$$d_{eq} = -(c_p m_p + c_q m_q) / (m_p + m_q) \qquad \text{(Eq. 5.3)}$$

$$v_{eq} = (c_{qpq} m_{qpq} - c_{ppq} m_{ppq}) / (m_p + m_q) \quad \text{(Eq. 5.4)}$$

where the c and m values are the additive and multiplicative coefficients of sub-responses that are active at the equilibrium state: q is the quarry's response at distances just less than its stable distance, and p is the persuer's response at distances just greater than its stable distance. The following environment contains two T1 responses. Object 1 would be the persuer and object 2 would be the quarry; the responses active in the equilibrium state are underlined.

```
wraparound

object 1
    circrad 20 xcent 100 ycent 200
obend

object 2
    circrad 20 xcent 924 ycent 668
obend

partresp 1
    if distance < 65
        return((distance -65) * -.385 away)
    else
        if distance < 280
            return((distance -65) * -.056 away)
        else
            if distance < 295
                return((distance -295) *.8 away)
prend
```

```
partresp 2
    if distance < 150
        return((distance -150) * -.1 away)
    else
        if distance < 280
            return((distance -150) * -.192 away)
        else
            if distance < 500
                return((distance -500) *.114 away)
prend

    bmatrix
            0 1
            2 0
```

The equilibrium distance in the above example would be 119.5 units and the equilibrium velocity would be 3.05 units. The direction of motion would be the initial inter-object direction.

A simple T1 response becomes more interesting as the stimulus' behaviour becomes more interesting. If the stimulus moves in an arbitrary manner, it will seem to push or pull the actor at a distance, as if by a long rubber pole. The length of the 'pole' is the equilibrium distance and the apparent stiffness is determined by the slopes of the segment-curve. As the slopes (multiplicative parameters) approach -1 the pole will appear very stiff -- the actor will spring to the equilibrium distance in about 1 time unit. If the slopes are that high, the potential for excessive velocity is also high -- this type of response is only useful if there is nothing that could rapidly move the actor

away from the equilibrium distance. An away slope less than -1 causes oscillation and is never useful.

If the invisible pole becomes arbitrarily short, by using an equilibrium distance of zero or less, then the observed behaviour will be pure attraction. If a slope near -1 is combined with a zero equilibrium distance then the stimulus appears to be pushing or pulling the actor and the actor appears to be clinging to the stimulus. The following very simple partial response causes this behaviour if used alone:

return(distance towards)

Note that a multiplicative parameter of +1 was used with the towards direction instead of the equivalent -1 away. This type of T1 behaviour has a very realistic feel about it. For example, if the actor is being pushed and the two meeting surfaces are not perpendicular to the direction of movement, the actor will tend to slide sideways, swing round to the rear of the stimulus and end up being pulled. The only problem with this response is in cases where the object being pushed becomes 'caught' due to an overriding response and then jumps back with excessive velocity.

T1 responses can be constructed in numerous ways to create many interesting effects. Say, for instance, a re-

r	d	v	m	c
0	-40	0		
1	-20	-20	-1	40
2	40	0	0.333	-40

Figure 5.4 -- Calculations for a Be-Captured T1 Response

sponse was wanted that simulated 'being captured'. The first step would be to set up a table like figure 5.4. Again the underlined unknowns are found by solving the equations given earlier; this time however, the first segment has a known slope (-1 to keep the actor captured at a fixed distance) but an unknown distance.

The following would be how these calculations would appear as a partial response:

```
if distance < -20
    return((distance + 40) * -1 away weight 500)
else
    if distance < 40
        return((distance - 40) * 0.333 away)
```

The response has no effect if the actor is further than 40 units from the stimulus. A quite strong attraction is felt if the actor comes within 40 units, the closer the stronger. The attraction sucks the actor towards the stimulus as if it were being captured. The very strong weight on the first sub-response ensures that no other response can cause it to

escape. There is one drawback to this function though, if the stimulus moves faster than 20 units after the capture has taken place it will release the actor. This may sometimes be useful, but normally the condition in the above example must be modified to use apparent fixedness -- this is discussed later. To make the capture visually effective, the actor should be defined before the stimulus because objects defined later can cover up objects defined earlier.

In the examples above, some behaviour on the part of the stimulus was needed in addition to the T1 responses to achieve an interesting system. A system composed entirely of T1 responses will settle down into an equilibrium state with all actors hovering at fixed distances from their stimuli (assuming the responses are constructed properly so the system does not go out of control or oscillate). The only movement could be the monotonous type of equilibrium persuit. Thus while many useful elements of a system can be T1, T2 elements are required to obtain animate behaviours.

5.2 T2 RESPONSES

In T2 responses, calculations are based on the positions of objects in the previous two time intervals. The magnitudes and directions that can be used to build T2 sub-responses are various expressions of the concept of 'change in position'. The most important magnitude is velocity and the most

important direction is the forwards direction (the direction of movement).

5.2.1 Self-Oriented (Reflexive) Responses

One of the most interesting and useful T2 responses is that of an object responding to its own previous velocity and forwards direction. The simplest self-oriented response is:

```
return(velocity forwards)
```

which has the effect of at least trying to maintain the current movement (it could be overruled by more heavily weighted partial responses).

Responses involving just velocity and forwards are the only ones where the actor is not concerned with an external stimulus; the stimulus is the actor itself. The most logical place to enter such partial responses in a behaviour matrix is on the diagonal where the row and column refer to the same object. In fact, the only partial responses that make sense on the diagonal are self-oriented ones (PAM will allow any partial response on the diagonal, but the output velocity from one that is not self-oriented is a meaningless constant). Since, however, self-oriented partial responses are only concerned with the actor, it is possible to place them in any element of a behaviour matrix row with identical effect.

Self-oriented responses can be used to make an object have a characteristic velocity to which it will return in the absence of other influences. This characteristic, or homeostatic, velocity might be very slow in the case of, for example, a grazing animal, or relatively fast in the case of a hungry predator. One must, though, bear the user's ability to perceive the scene in mind when deciding on a characteristic velocity -- this issue is discussed later in this chapter. Self-oriented responses can also be used to do such things as limit the top speed of an actor so that while it may accelerate rapidly to some speed while engaged in persuit or avoidance, it requires an increasingly powerful response to its stimulus to continue accelerating as it approaches its maximum speed.

The following is an example of a useful self-oriented partial response:

```
if velocity < 1
  return(velocity + 0.5 forwards weight 0.1)
else
  if velocity < 5
    return(velocity * 1.05 forwards weight 0.01)
  else
    if velocity < 8
      return(velocity forwards weight 0.001)
    else
      if velocity < 15
        return(velocity * .95 forwards weight 0.01)
      else
        return(velocity * .9 forwards weight 2)
```

The aim of this response is to cause the actor to maintain a speed of between 5 and 8 units per time frame. In that speed range, the output is the maintenance of the current velocity, however the low weight means that some other partial response could easily override this one. If the actor deviates slightly from the optimal velocity range, either above or below, then the velocity will be decreased or increased slightly at each step in order to bring it back to normal. The weights are higher for this in order to make it slightly harder for the correction to be overruled. At the extremes -- velocity below 1 unit or above 15 units -- even more corrective action is needed. At very low velocity, an additive parameter is used instead of a multiplicative one because if the velocity dropped to zero, no amount of multiplication could ever increase it. At excessively high velocities the weight of the self-oriented response jumps dramatically -- this provides the actor with a maximum speed.

The parameters in the above example could be altered somewhat, but the function performed would always be the same. Unlike T1 responses there are no computations involved to set up a self-oriented response. One important thing to bear in mind is that the multiplicative parameters should not deviate far from one, otherwise there is the risk of the behaviour going out of control or oscillating wildly.

5.2.2 Combinations of T1 and Self-Oriented Elements

The distance-towards (T1) and velocity-forwards (self-oriented) types of responses together provide a firm foundation for many types of behaviour. The variety of behaviour available from just these responses can be further expanded by combining their elements in different ways.

In general, any response that uses distance as its variable magnitude is constructed in a way similar to T1 responses. The user must choose velocities the actor should assume at the desired distances and then solve the equations for the parameters to a series of sub-responses.

Responses that use velocity as their variable magnitude are usually constructed like self-oriented responses in order to achieve control of acceleration in a certain direction.

Responses that use the forwards direction do not have any influence over the direction of movement whereas responses that use the towards direction are concerned with movements towards or away from the stimulus.

An example of an interesting combination are distance-forwards responses. Unlike with T1 functions, it is usually only advisable to have positive velocities on the velocity vs. distance graph since negative velocities would result in

backwards movement and oscillations would ensue unless masked by some other response. Additionally, some of guidelines for T1 responses can be ignored, such as ensuring the velocity at the outermost distance range is zero, because sudden velocity changes are not compounded by sudden direction changes. Distance-forwards responses are useful to make an actor speed up or slow down in the vicinity of a stimulus; this can be helpful in the building of impressions such as those of 'fear' but since no change of direction occurs, other elements are needed.

The following partial response contains several combinations of the elements described above. Distance-forwards elements in three ranges cause an actor to slow down as it comes within an initial range of the stimulus, but then rapidly speed up if it gets too close; the calculations for this part of the sub-response are in figure 5.5. Velocity-towards based elements cause the actor to move towards and away from the stimulus at different ranges. In the intermediate range there is a self-oriented sub-response; it is also assumed that this partial response would be combined with a low-weighted self-oriented response that would take effect outside the range.

r	d	v	m	c
0	0	15		
1	80	6.5	-0.1063	-141.176
2	120	1	-0.1375	-127.273
3	180	6.5	.0917	-109.117

Figure 5.5 -- Calculations for a Distance-Forwards Response

```
if distance < 80
    return ((distance-141.176)* -.1063 forwards)
    return (velocity * .6 away)
else
    if distance < 120
        return ((distance-127.273)* -.1375 forwards)
        return (velocity forwards)
    else
        if distance < 180
            return ((distance -109.117)
                * .0917 forwards)
            return ((velocity + -.2) * .4 towards)
```

This response is the first discussed in this chapter that had some semblance of intentionality, especially when the stimulus is also moving. Figure 5.6 shows a trace of the path of an actor with the above partial response towards the stimulus in the centre. In this photograph it is easy to see that a pattern is actually being formed. When no trace is being made, the pattern is not so easily detected. If the stimulus moves even slightly, the pattern is completely disrupted. One thing that the photograph cannot convey are the changes in velocity that the actor undergoes.

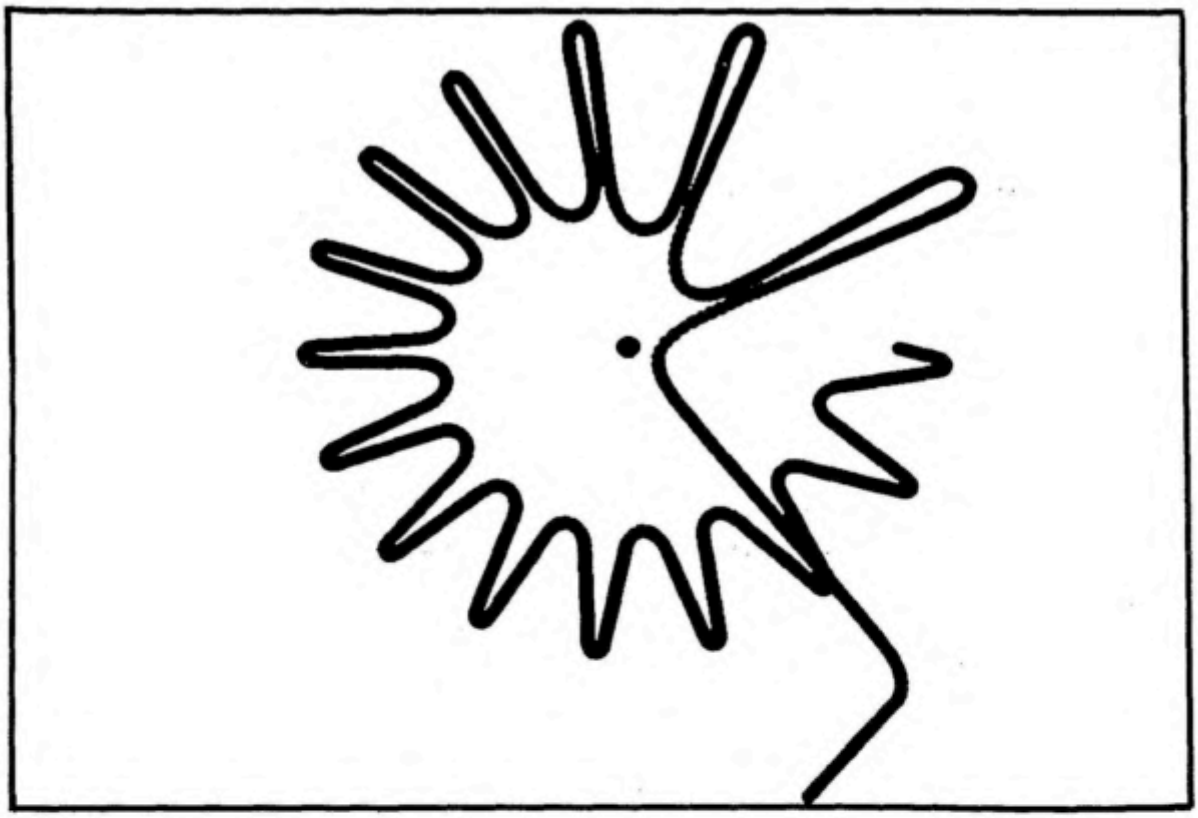

Figure 5.6 -- Path of an Actor with Intentional Behaviour

With this response, the actor will approach and retreat from the stimulus, sometimes making small hops and other times making much larger hops. The behaviour can be compared to a dog rounding up sheep, except that this dog sometimes loses interest if it gets too far away. To complete the scene, of course, a 'sheep' behaviour is needed on the part of the stimulus.

5.2.3 'Othvelocity' and 'With' Direction

The othvelocity magnitude and the with direction allow an actor to respond to the stimulus' movements instead of its distance and direction. One drawback of these quantities is that the actor responds in the same manner no matter what the distance.

The response:

```
return(othvelocity with)
```

causes the actor to move identically to the stimulus. This can be useful in behaviour such as fish schooling where groups of objects should move together, however a similar and often superior effect can be obtained with T1 responses. The fundamental difference is that othvelocity-with responses fix the direction at which the actor stays with respect to the stimulus -- this often looks unnatural.

One useful combination involving these elements is:

```
return(velocity * .9 forwards)
return(othvelocity * .1 with)
```

This has the somewhat unexpected effect of ensuring that the actor always stays at the same relative position with respect to the stimulus, but lags behind in catching up to

this required position. The lag-time can be adjusted using the multiplicative parameters. Unfortunately, this effect easily breaks down if other responses are present. Another problem is that this does not give the correct impression of 'following' because the equilibrium direction is fixed as well as distance. Changing 'with' to 'towards' solves this second problem somewhat -- the resulting response is to move towards only if the stimulus is moving; the two objects will collide though.

5.2.4 Change in Distance

Change in distance (changedist) can be used in similar ways to velocity.

Distance-based responses have the problem that the behaviour of the actor is symmetrical when it is approaching and retreating from a stimulus. By using change in distance in conjunction with other elements, an object can be made to move in one manner on approach and in another manner as it moves away. To do this, one set of response could be active if the change in distance were negative (moving towards) and another active if it were positive.

5.2.5 Time to Reach

Many responses that uses the distance magnitude can be converted to use time to reach instead. Time to reach, which is distance divided by the change in distance, allows an inter-

action to be made independent of velocity. A slow object might be made to exhibit the same behaviour at a closer distance that a faster object might exhibit at a greater distance.

One set of cases where it is essential to prevent the velocity of an actor from having an effect on the response are physical responses such as collision. A fast-moving actor must bounce off, just as readily as a slow-moving actor. Theoretically, distance could be used as the criteria to detect when a collision has occurred, but for reasons of discretization time to reach is the only magnitude that does the job. An example which uses 'timetoreach' is the simple bouncing response at the start of chapter 3.

5.2.6 Apparent Fixedness

Apparent fixedness (appfixedness) is another magnitude that is a ratio with distance as the numerator. The denominator this time is the velocity of the stimulus.

One of the most useful responses involving apparent fixedness is a capture response that allows no escape:

```
if appfixedness < 1
    return(distance + c towards weight 500)
```

where c should be twice the sum of the minimum radii (or half dimensions) of the objects. This would cause the stimulus to apparently 'snap up' the actor if the distance became less than the velocity (change in distance at one time step) of the stimulus. In a discrete system, capture responses which only involve distance as the range magnitude cause the actor to be left behind if the stimulus moves too fast. This cannot happen with apparent fixedness because any movement of the stimulus causes an equal or less increase in distance, maintaining apparent fixedness less than 1. The capturing object should be defined after the captured one and should be larger in order to create a proper effect. If the radii are similar and the stimulus moves fast, the 'eaten' actor may appear to bulge out of the stimulus due to a one-time-unit time delay.

5.2.7 Quantities Involving Relative Movement

When the actor is stationary, the relative velocity is the velocity of the stimulus; when the stimulus is stationary the relative velocity is that of the actor. This property seems promising, but in fact almost any response that can be created using relative velocity can be created by summing velocity and othvelocity (along with forwards and with directions). Despite several attempts, no good use has yet been found for relative velocity and the direction of relative movement, because most functions that use them have a tendency to oscillate.

A related quantity, stillness (distance divided by relative velocity), has been found to be equally useless for similar reasons.

5.2.8 The Use of Compound Directions

The three compound directions, reflection, orbital and interception were created for specific purposes. They allow the response designer to direct an object to go in a direction that has a physical meaning but which cannot be directly obtained using simpler directions and magnitudes.

The reflection direction is primarily used to allow bouncing. The example at the beginning of chapter 3 shows it in use. This direction only has meaning when an actor is about to collide with a stimulus.

The orbital direction is useful in object avoidance. It allows the actor to move in a direction orthogonal to the inter-object (towards) direction, and generally should be combined with other responses to create a slow avoidance curve. The orbital direction can be used to simulate the orbiting of planets, but if used alone the orbiting object will tend to spiral out -- an element of towards direction must be added. Orbiting can be achieved in a more simple way by combining elements of distance-towards (gravity) and velocity-forwards (momentum). This simpler method is closer to what happens in the real universe, but as in the universe

the orbiter may go out of orbit or plunge towards the stimulus if balancing is not correct.

The interception direction can replace the towards direction in any persuit behaviour. It adds an element of 'intelligent planning' by causing the actor to move towards the path of the stimulus, instead of directly towards it, causing a more rapid capture.

5.2.9 Experience with Inverse-Square Magnitudes

For the purposes of supplying fixed ranges for responses, inverse-square distance is identical to Euclidean distance since there is a one-to-one correspondence between them. The only advantage of inverse-square distance is in its non-linear nature; it can, for example, be used to make a system of orbiting planets behave physically correctly. A T1 response using inverse-square distance is created in a similar way to a normal T1 response, except that more complex computations must be done to ensure the segment-curve is continuous. Each segment in the curve is non-linear, but since velocity changes in normal T1 responses are generally of low slope, and the curves in inverse-square T1 responses have low curvature, the resulting behaviour is very similar.

Inverse-square change in distance and inverse-square time to reach have been used to generate various behaviours, but no new classes of behaviour have been found with these.

5.3 COMPLEX BEHAVIOUR SYSTEMS

Behaving systems can be put together in an infinite variety of ways. The most complex environment we have created so far simulates the schooling of fish. We have nine 'fish' objects in the school and one 'predator fish' object, all with several types of response.

Each school fish has a weak self-oriented response, three different sub-responses to the other school fish, and two strong sub-responses to the predator. The self-oriented response causes the fish to move forward, tending to a natural speed. One response to the other fish is to move slowly towards them. A more powerful inter-fish response keeps them from coming too close to each other. The third inter-fish response is even more powerful and is of a physical nature, preventing fish from occupying the same space. The primary response of fish to the predator is to move away -- it is very powerful but can be overridden by the physical response if fish collide. If a fish should become intercepted by the predator, its overriding response becomes to stay with the predator (it has been eaten). Eating is simulated by the simple occlusion of the prey by the predator.

The predator also has a homeostasis type response, which becomes overridden by a chasing response when fish come close. When the predator actually intercepts its prey

and captures it, a self-oriented homeostasis response again takes over.

The initial conditions are a random collection of fish in one part of the screen and a predator in another part. Visually, we see both types of object minding their own business. The school of fish move forward, gradually coalescing into a semi-ordered school, there is some jostling around but a general unity of movement. We see the fish and predator swim off one side and onto the other side of the screen in a wrap-around manner. After a short while, the predator makes a sudden movement and heads towards the centre of the school. As the predator approaches, the school splits apart, scattering fish in all directions. The predator catches up with one of the fish which disappears. Finally, the school reforms as the predator heads away.

The change in predator behaviour which occurs after it has captured its prey is interesting. This permanent change has occurred because the predator is now a composite object, it is carrying the prey with it at zero distance and so in this state it can be given different behavioural properties.

5.4 PROBLEMS WITH BEHAVIOUR FUNCTIONS AND THEIR SOLUTIONS

Although new and interesting behaviours can easily be created by varying the response functions, it has been found

somewhat difficult to create some behaviours given a preconceived notion of what they should look like. This has been particularly the case when a more animate appearance is desired and when the interactions become more complex. Problems encountered when designing behaviour functions can be grouped into two classes, those of a perceptual nature and those resulting purely from the mathematics of the system.

5.4.1 Problems of a Perceptual Nature

Behaviours we have classed as having perceptual problems may be quite rich and interesting but are unacceptable to the viewer for one reason or another.

The primary perceptual problem is lack of balance, especially of velocity. Velocities must not be too fast or the viewer's ability to follow action is inadequate, and a too-slow velocity loses all appearance of animacy. By experimentation acceptable velocities can be found; but even if objects keep some mean velocity, transitions must be considered.

5.4.1.1 Acceptable Velocities

Smaller objects allow slower movements to have the same effect as a larger object moving faster, but when an object becomes too small its behaviour becomes less noticeable. Objects with a radius of 1 to 2 per cent of the screen width (10 to 20 units in PAM) have been found to be effective, and

a velocity on the order of 0.5 per cent of the screen width per time-step (5 units in PAM) seems to be a reasonable average. For a reasonable system it has been found desirable to let objects move in the range of velocities from 1 to 20 units, with the faster velocities reserved for situations when some active behaviour is taking place.

5.4.1.2 Handling Screen Edges

At a mean velocity of 5 units, screen edges become a serious problem since an object can move from one edge to another in six seconds or less. Three methods have been used to handle screen edges, two of which are special features built in to PAM.

The first method is to allow the system to follow objects as they move outside screen boundaries by performing translate and scale operations to cause the 'virtual camera' to pan and zoom. In a full-scale production animation system, this would be the most attractive solution and could be expanded to allow multiple camera angles in three dimensions; it does have a few drawbacks though: One drawback is the necessity for some kind of background so the observer receives the appropriate sense of lateral movement -- without panning or zooming this is not necessary as the physical boundaries of the screen act as a reference. A second drawback to panning or zooming is a side-effect of the behaviour function animation technique; the objects have independent

movements and when not constrained they will tend to move apart. In order to observe the behaviours of all objects it is necessary to zoom back continuously causing the objects to become apparently smaller and thus more difficult to notice. The PAM system allows for an optional background grid to act as a reference and a zooming capability to keep all objects in view. Figure 5.7 shows the effect of this as objects move apart. In practice, this zooming has only had limited usefulness -- future work might enhance this to allow for key objects to be identified and for the system to pan, keeping only the most important objects in view.

The second way of handling screen edges, also implemented as a special feature of PAM, is to allow the positions of objects to be calculated using modulo arithmetic. Figure 5.8 shows that with this scheme, objects that move off one side of the screen move back on the other side; this works in both x and y directions. This 'wraparound' technique is commonly used in video games. Although this method ensures that all the behaviour one is interested in stays within screen boundaries, the impression of a particular behaviour is severely reduced as it goes through the boundary. It can be particularly disconcerting to watch a chase scene where one object has already passed through the boundary and its persuer is still just approaching the boundary -- although the persuer follows the quarry through the boundary there is a period of time when the persuer appears

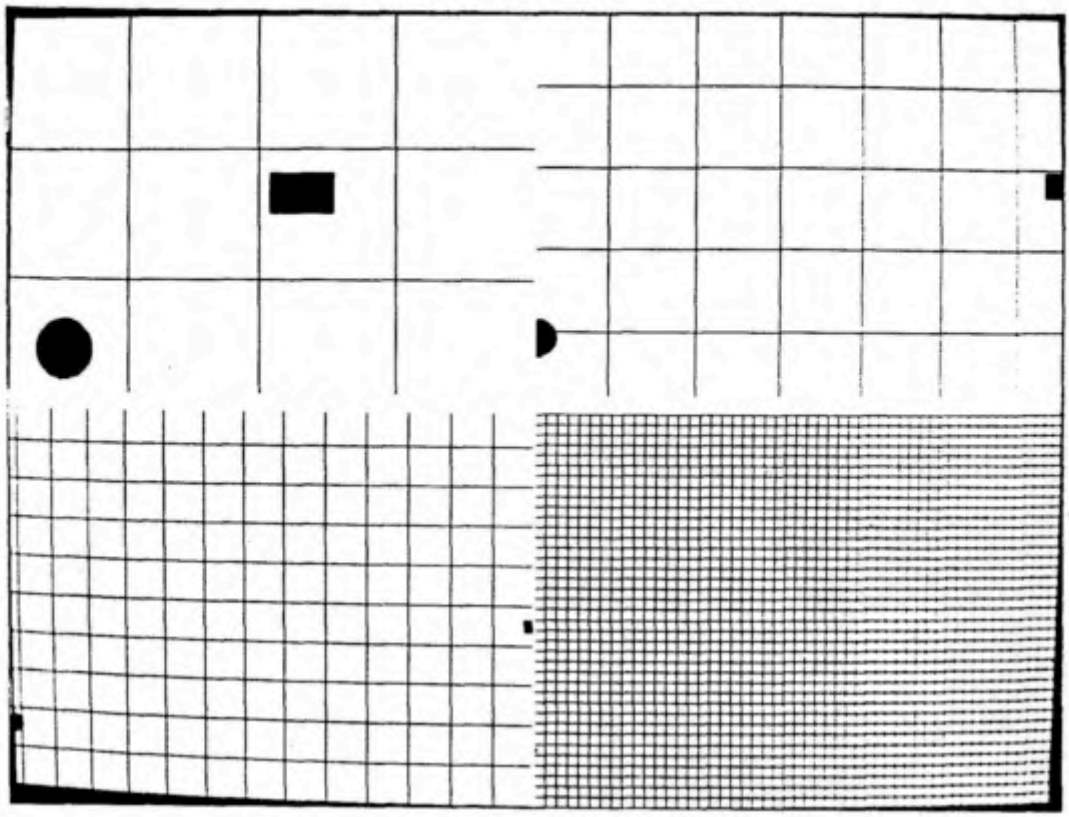

Figure 5.7 -- Using Zooming to Keep Objects in View

to be moving away from the quarry at a distance of almost one screen width.

Another problem with wraparound is that as objects separate by more than half the screen the towards direction suddenly reverses causing strange effects in some behaviours.

The third technique for handling boundary conditions does not require special features of the implementation. This involves the setting up of a large object that fills

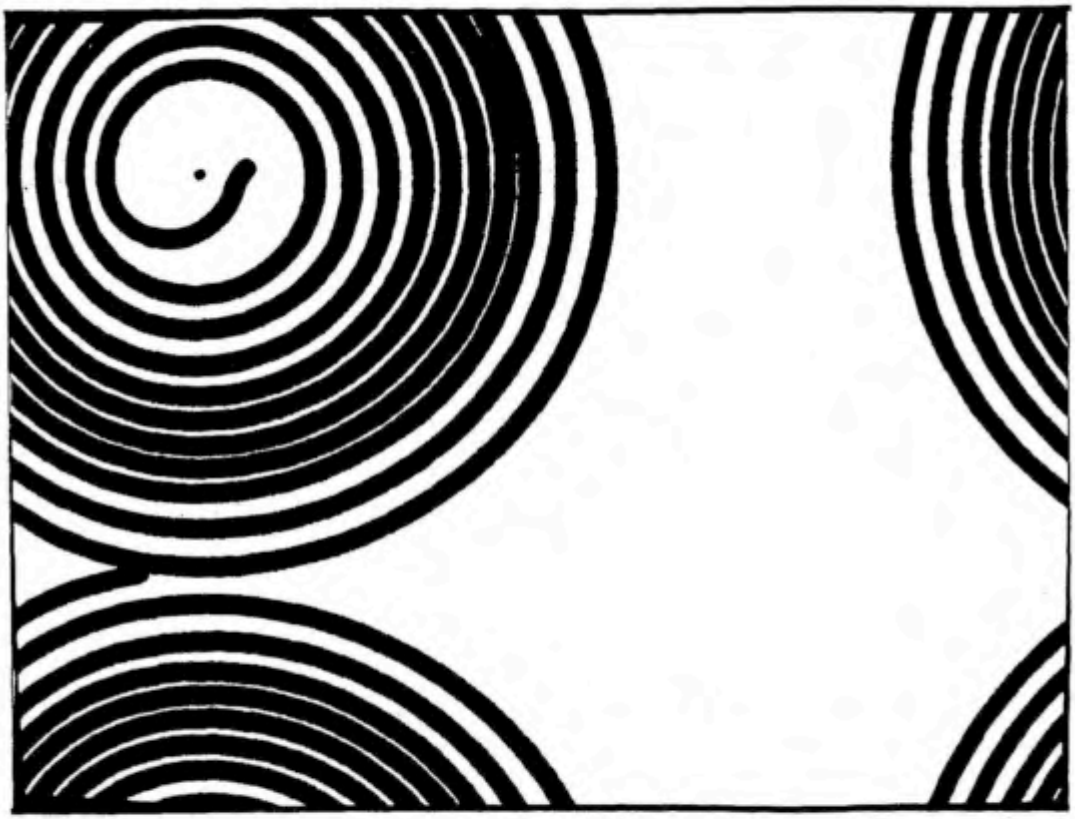

Figure 5.8 -- Using Wraparound to Keep Objects in View

the entire screen and has the role of 'walls'. The other objects are constrained by behaviour functions to remain within the boundary -- when any object is about to reach the wall a heavily weighted partial response would cause a bouncing or similar action. The advantage of this method is that the observer can continuously follow the movement of an object without it decreasing in apparent size or going through boundaries; unfortunately though, the behaviour of interest may be completely altered by the partial response to the walls.

	METHOD OF HANDLING SCREEN EDGES		
ADVANTAGE	Pan and Zoom	Wrap-around	Wall Object
No special implementation required			*
Does not cause perception problems			*
No size change		*	*
Objects never diverge		*	*
No background required		*	*
Compatible with trace-making for taking still photographs of paths		*	*
Behaviour never altered	*		
Observer can follow movement continuously	*		*
No disruption due to overriding walls responses	*	*	

Figure 5.9 -- Methods of Handling Screen Edges

Figure 5.9 summarizes the advantages of the methods of handling screen edges. Either of the first two methods may be combined with the third method; the first two methods are mutually exclusive. Most of the work was done using wall objects because, although behaviours were modified, it was found to be the most pleasing situation to observe and it was still possible to observe meaningful interactions in the

time spans between wall collisions. The least desirable option has been found to be zooming.

5.4.1.3 Unnatural Movements

The worst types of unnatural movements are those where position changes abruptly. Since the behaviour function model we are using is based on the determination of velocities at each time step, discontinuities of position will not occur unless the velocity gets out of control (this situation occasionally occurs due to mathematically based problems discussed later). Discontinuities of velocity or acceleration are other circumstances to be considered; we will look only at velocity since acceleration jumps do not seem to cause perceptual problems.

A commonly-seen situation might be as follows: an object is moving around at some mean velocity and then suddenly speeds up or slows down and may also make a sharp change in direction. It reaches a threshold beyond which a different response or set of responses influence it, probably due to the approach of another object.

Abrupt changes in velocity at transition points are often a problem but can also enhance the animation by adding an element of intentionality. It is as though the brain, unable to attribute the movement to some physical cause, attributes it instead to a deliberate action. It should be

the objective in the design of an effective behaviour function to ensure intentionality-inducing transitions are present and annoying ones are absent.

Experience has found that two factors can be manipulated to deal with transitions: the mean velocity and the weight. If the velocities produced by responses on both sides of a transition threshold are close, no transition effect is seen. As a general rule, responses should be individually designed so the velocities they yield are comparable with the velocities of other responses in the same environment. The problem is that as objects move, the spheres of influence of various responses move too, with the result that an object will at different times be under the control of differing numbers of responses. In the behaviour function model we are using, responses are summed, so when an actor comes under two influences the effect may be doubled or they may cancel each other out. The goal must be to prevent sudden changes in the sums of responses.

There are three ways to combat changes in response sums: increasing the velocity gradually within the range of a response, weighting responses so only one has effect, or using minimum-distance or a similar computation to select only one response.

The first solution requires a response that, at the boundary of its influence, results in a velocity of zero. Since these boundaries are normally distances (although this need not be the case -- an actor could respond to a provoker when the provoker's velocity or time-to-reach is within a certain range), the linear equation for T1 functions can be solved to ensure the velocity is zero at the influence boundary and increases slowly within the boundary. Complex T2 magnitudes containing elements of distance can also be used.

Manipulating weights is probably the easiest way to handle response summing. The essence of this technique is to rank responses in terms of overall importance and assign to each rank weights that differ in several orders of magnitude. Although a large number of different ranks can be used, it has been found especially useful to assign weights to the three ranks shown in Figure 5.10. Using these weights, a physical constraint response (e.g. bouncing off a wall) will take preference over all else. Where no physical constraint need be applied, but where the actor is within the area of influence of a stimulus, the primary response of interest will be effective. Finally, if there is no stimulus within range, the homeostasis type response will control the actor's behaviour.

TYPE OF RESPONSE	WEIGHT
Self-Oriented/Homeostasis -- effective outside all other responses	.01- .05
Primary Response -- effective within range of stimulus	1 - 5
Physical Constraints -- effective near stimulus boundaries	100 - 500

Figure 5.10 -- Weights for Responses of Increasing Importance

There are two disadvantages of controlling response summing using weights: It is not effective where responses are of equal weight -- with the weighting in figure 5.10, the summing of the primary responses could still be a problem. The second disadvantage is the converse, it renders responses of lower rank ineffectual within the ranges of higher-ranked responses. For example, it is often desirable to have a self-oriented response override the primary response when the velocity becomes too high or too low. This can be combatted by making the weight from a partial response higher at the extremes: A series of different subresponses could operate within different ranges, or the system could be told to vary weight continuously -- in PAM a negative user-supplied weight specifies that the weight is actually to be the magnitude of the weight given, multiplied by the output velocity.

A third way of combatting summing of responses is to have the system select a particular response. In PAM it is possible to have an actor only respond to the stimulus whose distance (or other magnitude) is the minimum. This solution has not been studied in detail and should be the subject of future research. This method is similar to weighting in that only a certain set of responses has effect at one time, the difference is that with minimum selection, stimuli of equal importance can be made to dominate each other whereas the hierarchy of dominance is always fixed when weighting is used. Minimum selection is also not effective when one of the responses in the selection is self-oriented -- for example you cannot compare the 'distance to yourself' with the distance to another object.

Figure 5.11 summarizes the advantages of the different solutions to the response-summation problem. In a complex environment the three methods can be combined to obtain a desired mix of advantages.

5.4.2 Problems of a Mathematical Nature

The mathematical class of problems tend to cause behaviours that are uninteresting if left unchecked; they tend, however, to be easier to solve than the perceptual class of problems. There are five such problems in total. Two, namely positive and negative feedback, are inherently part of the behaviour-function concept and are generally useful,

	METHOD OF HANDLING PROBLEMATIC RESPONSE SUMMATION		
ADVANTAGE	Gradual Velocity Change	Weighting Selection	Minimum Selection
Stimuli of equal importance can be made to dominate each other			*
Hierarchy always enforced		*	
Easy to describe in language		*	*
Responses never override others	*		
Mean velocities of responses matter less	*		
Effective when identical partial responses interact	*		*
Works with self-oriented responses	*	*	
Does not require extra computations	*	*	

Figure 5.11 -- Methods of Handling Response Summation

only becoming problems when they get out of control. The third problem, oscillation, is related to the feedback problems but is never useful. Discretization, the fourth problem, is a problem caused by practical limits in implementation. The final problem, exact repetition, occurs occasionally due to pure bad luck.

5.4.2.1 Positive Feedback

A response with positive feedback is one whose effect is to change some condition such that subsequent changes become greater. This leads to a lack of stability. It might be a change in position or a change in velocity that increases; both are problematic if unchecked but increasing change in velocity (increasing acceleration) can be far more spectacular -- an actor's velocity can exceed the speed of light in a couple of time steps and can exceed the largest representable floating-point number in less than a second!

It is essential to have some positive feedback in a system because it is the only way of achieving smoothly increasing velocity. There must always be some kind of limiting factor, however, to prevent the system from going out of control. The most reliable way of preventing excessive positive feedback is to provide each response with a condition that specifies a range in which it can operate. Once the magnitude of concern is no longer in the range, positive feedback is cut off. Another way to control positive feedback is to provide a second, more heavily weighted, partial response that takes control within some range. This is effectively what walls are supposed to do -- their objective is to prevent objects from getting too far apart. Walls can suffer from other problems though, including discretization and oscillation. Modular arithmetic (wraparound mode) can

control excessive distance, but it cannot control excessive velocity.

It has been found that for most types of response, particular combinations of multiplicative and additive parameters will cause positive or negative feedback, oscillation or combinations of the three. Conversely, it is usually possible to eliminate these conditions by choosing appropriate parameters, although this may mean desired behaviours cannot be achieved. When responses become complex though, it can be difficult to predict undesirable feedback so restricting responses to ranges may be essential.

The subsequent paragraphs identify the values of parameters that are problematic for two of the simplest types of response.

The following T1 partial response increases velocity steadily due to increasing change in distance if it is the only response affecting an actor:

return(distance away)

The velocity here will double at every iteration. Any T1 response will run into problems unless it follows the format described earlier. The away direction with a positive multi-

plicative parameter or the towards direction with a negative parameter will cause a loss of control due to positive feedback, unless there is some limiting factor.

The self-oriented response shown below is also problematic:

return(velocity mult 1.1 forwards)

In general, any time the response is based on velocity with a multiplicative parameter greater than 1, positive feedback will occur. With such a multiplicative factor the result is increasing acceleration; one should beware if the factor is substantially above one. A positive additive factor also causes positive feedback, but only a velocity increase which is more benign.

5.4.2.2 Negative Feedback

A response with negative feedback is one whose effect is to change some condition such that the next change is smaller. The result is a damping down of movement; if uncontrolled the whole animation may grind to a halt.

Some negative feedback is necessary to control positive feedback and to generally make behaviour interesting. The primary methods of control are the same as with positive feedback -- restrict response functions to a range so that

if velocity is too low the negative feedback is cut off, or provide another response that can override the one causing negative feedback. Since much lower velocities and distances are involved, the second solution is much more effective for negative feedback than it is for positive feedback.

If an object slows down to a halt, the culprit might not be a negative-feedback response. The response-summation problem mentioned earlier may be what is to blame; responses may be cancelling each other out.

The simplest cases of negative feedback are similar but opposite to the examples given for positive feedback. The following T1 response causes an actor to rapidly decrease in velocity as it approaches its stimulus:

```
return(distance * .1 towards)
```

In general any T1 response which has a multiplicative parameter between 0 and 1 exclusive and a towards direction (or a parameter between -1 and 0 and an away direction) will exhibit negative feedback. The following T2 response will decrease the speed to nill:

```
return(velocity * .9 forwards)
```

The conditions for those self-oriented responses that cause negative feedback are almost the same as the conditions on T1 responses: any self-oriented response whose multiplicative parameter is between 0 and 1 exclusive and whose direction is forward (or -1, 0 and backwards) will slow down.

5.4.2.3 Oscillation

Oscillation is a frequently observed phenomenon in behaviour function animation. It occurs when a vicious circle is set up; the effect of a response at one time step is counteracted by a response at the next time step. This type of cycle results in the actor rapidly vibrating between two positions at a frequency of half the time-step update rate.

There are many circumstances that cause oscillation. They can be divided into two categories: those caused by a single sub-response and those caused by two or more sub-responses interacting. The solution is always to remove the cause of the oscillation. In the case of single-response oscillations it is sufficient to alter parameters; to remove oscillation from a multi-response situation it may be necessary to do substantial tinkering with parameters, weights and the responses themselves.

Single-response oscillations can be usually dealt with if found. The following shows a typical T1 response that

will cause its actor to oscillate between two points that might be quite distant:

```
return(distance * 1.1 towards)
```

In general, any unconstrainted T1 response with a multiplicative parameter greater than 1 and a towards direction (or a parameter less than -1 with an away direction) will oscillate since the actor will always jump back and forth over the stimulus.

Badly designed self-oriented responses like the following also have oscillation problems:

```
return(velocity * -1 forwards)
```

Any such response with a negative multiplicative parameter and a forward direction (or a positive parameter and a backwards direction) will oscillate.

All responses involving relative movement have a tendency to oscillate -- it is for this reason that these responses have been found not very useful.

The following partial response illustrates a typical multi-response oscillation problem:

```
if timetoreach < .5
    return(velocity reflection)
else
    return(velocity towards)
```

In this example the actor will initially move towards the stimulus. When it is about to collide with the stimulus, the actor will bounce off. At the next time step, it will again approach and the approach-retreat cycle will repeat forever at a very rapid rate.

There are many other possible combinations of sub-responses or sub-response groups that counteract each other at alternate time steps. Any response that contains directions that can oppose each other should be used with caution. If for example a velocity-forwards and a velocity-towards response were used together they will either counteract or complement each other depending on the actor's forwards direction.

5.4.2.4 Discretization

Discretization results from the fact that although the illusion of continuity is maintained in behaviour-function animation, calculations are actually made at finite intervals. As a result a whole time-step may pass before behaviour functions can respond to a situation.

Discretization can be a problem in many instances. One example is the unintentional passing of an actor through a stimulus when it is in fact supposed to bounce off. This would be likely if the velocity of the actor were greater than the sum of the radii of the actor and stimulus, and the following were the response that is supposed to cause the bouncing:

```
if timetoreach < 0
    return(velocity reflection)
```

The trouble with this is that the time to reach may not be checked until the actor is over half way through the barrier, at which time the reflection direction would be forwards. One solution is to make the response bounce earlier: if the time to reach is less than 1. That works in most cases except when the actor is rapidly increasing in speed. There is a drawback to this solution though, if the actor is moving rapidly, it will appear to bounce before reaching the stimulus. When there is no danger of the actor passing through the stimulus it is preferable to have the bouncing occur when the time to reach is less than .5 because then the average distance at which it will actually bounce is the exact boundary of the stimulus.

5.4.2.5 Exact Repetition of a Sequence

In a pseudo-random number generator, it is expected that cycles of numbers will appear. A good generator will attempt to make a non-repeating sequence as long as possible, but eventual repetition is inevitable. Behaviour function animation works on the same principle as pseudo-random number generators -- outputs are fed back into inputs. The main difference between pseudo-random number generators and behaviour function animation in terms of calculations, is that there is more than one input and more than one output in the animation process. Nevertheless, if a given status recurs in the animation, a cycle will be formed.

The more complex the environment, the less likely that the repetition will occur. Exact repetition has only been observed with a maximum of two objects and two simple partial responses. Since cycles that repeat tend to be uninteresting, they should be prevented. There seems to be no way to predict them, but making minuscule adjustments to one of the parameters of a partial response solves this problem.

Sometimes an apparently-repeating sequence will crop up that is in fact not exact repetition. An example of this might be where an actor has 'cornered' a stimulus and swings back and forth near the corner. If precise quantities are noted, it will be found that this is merely a behaviour that follows a close but not identically repeating cycle. If this

behaviour is regarded as a problem, the behaviour functions could be modified to allow the cornered object to escape.

Chapter 6.

FUTURE RESEARCH

The work of this thesis has involved the investigation of some fundamental principles of behaviour function animation. The general behaviour of simple classes of responses and some problems involved in creating behaviour have been identified. There is no question that this technique of animation has promise, but much more work needs to be done.

Most of the future work outlined in this chapter would require enhancements to the PAM implementation. The technical documentation gives details about how some of these changes may be made.

6.1 LANGUAGE ENHANCEMENTS

One area of future research is the generalization of the animation description language, especially the partial response section.

One improvement would be to convert the keyword syntax described in appendix A into the formal syntax of chapter 2. If necessary, this could be done using a pre-processor.

A second, and probably more important enhancement would be to remove some of the idiosyncrasies that are built in for run-time optimization reasons. Some of the easily-identifiable idiosyncrasies are the lack of multiple conditions on a sub-response, the restriction to a single multiplicative and additive parameter, the fact that addition is done before multiplication and the fact that else conditions encompass all subsequent rules. If generalization were done, it would most likely be necessary to actually compile the partial responses in order to maintain real-time operation.

A major enhancement to make the system more usable would be to allow response functions to be used in the same way as functions in a conventional programming language -- with arguments and formal parameters. Other programming-language-like features such as symbolic constants and local variables with expression evaluation would also be useful. If these enhancements were made, research would be far easier, especially into T1 or distance type functions where the programmer must now solve systems of equations. High-level functions could be programmed and saved in libraries; these might have definitions like the following:

```
predator_avoid(notice_criteria, wary_action,
               danger_criteria, evasive_action,
               succumb_criteria, captured_action,
               release_criteria, escape_action)
```

with the arguments being numbers or symbols.

6.2 BEHAVIOUR FUNCTION RESEARCH

Much work could be done to increase the body of knowledge about behaviour functions. Firstly, since there are an almost infinite variety of behaviour functions possible, interesting and useful combinations are certain to be uncovered by continued 'tinkering' with the PAM system as it now stands. Mathematical properties of T1 responses have been described in detail in this thesis, further work could involve finding useful equations for T2 responses and simple combinations of responses.

There are several magnitudes and directions that could be added. In this thesis a few ratios of fundamental magnitudes are looked at. An extension would be to try all possible ratios as well as products (sums and differences can be done now due to the summing of sub-responses).

Another area of possible research is into the way various responses can be combined to create behaviour. The minimum-magnitude capability of PAM needs to be investigated, and this could be expanded to allow for the selection of minimum and maximum partial responses.

In order to enhance animacy, a useful feature would be to maintain a facing-direction for each object (the direction the object is looking as opposed to the way it is moving). Other useful features would be the ability for behaviour functions to manipulate the shape (so animal movement could be created[1]) and colour of objects, or other aspects of the environment. This would imply a quite fundamental change in the system to accommodate more than one output from all types of response.

At an even higher level, it would be useful to allow behaviour functions to work on complex objects, perhaps defined hierarchically. The main problem here would be finding perimeter distances. If complex objects were built from components of the type used in this chapter the distance would be that of the closest pair of components, however this might be an excessively inefficient way to proceed.

Yet another research path might be to investigate T3 or higher classes of response, or responses that have some other mode of memory -- perhaps saving data to be used as input at some future time-step.

---

1. Michotte 1963

6.3 GENERAL CAPABILITIES OF THE IMPLEMENTATION

There are several areas where improvements could be made over the PAM implementation.

A future PAM2 system might allow for panning after important objects to solve some of the problems outlined in the last chapter. A way of concatenating sections of behaviour that is more effective than PAM's simple 'load after time-out' could be implemented to allow the production of films. A related enhancement would be the ability to display complex backgrounds.

The whole system could be enhanced to operate in three dimensions. This could be relatively easily added to the current PAM system; the only difficulties would be creating depth-perception and updating the algorithms (especially the Euclidean distance between perimeters algorithm), ensuring they are still efficient enough for real-time display. If 3-D were implemented, camera movements could be defined; the virtual camera might actually be an actor responding with behaviour to the objects it is 'filming'.

One feature that was originally planned for PAM was the ability to interactively modify the environment. At its simplest level, such an enhancement would involve manipulating constant values in functions using some valuator device such as the mouse. At a more complex level, this might involve

changing whole functions. This would greatly enhance research productivity since fine tuning responses by rereading data files can be very time-consuming. Dummy interfaces are already provided in PAM for access to updating routines and to routines that would save the internally-updated data back into a file in the manner of a disassembler.

A final possibility for future work is to make a portable behaviour-function animation system -- the non-machine-dependent elements of PAM could be isolated, and the amount of code in the remainder could be minimized. This would facilitate research using other systems.

Chapter 7.

SUMMARY AND CONCLUSIONS

Behaviour function animation has been found to be an effective and promising technique. We have discovered that it is possible to create simple functions which cause apparently complex behaviours to be perceived.

The following paragraphs summarize some of the elements of the thesis:

1) In the universe, a relatively small number of physical laws are sufficient to produce all the behaviour we see in every day life. Likewise, in behaviour function animation, a small number of simple responses can create a vast array of behaviour. The concept of the chaotic system is the driving force in both cases.

2) It is useful to consider behaviour in terms of the relationships between each pair of objects in the environment. Each object can take the role of both actor and stimulus. The reaction of an actor to a stimulus is called a partial response, and can be further broken down into a series of sub-responses. The combination of partial responses for a particular actor forms the behaviour. In this thesis,

behaviour is generated by summing responses, but there is potential in other methods of combination.

3) Because in the worst case, every object reacts independently to every other object in the environment, and there is the necessity to calculate the state of the environment repeatedly, behaviour function animation is fundamentally an $O(n^2)$ process. Actual environments tend to be better than this due to the fact that many actor-stimulus relationships are null, usually because the actor is a static object like a wall.

4) The only output of behaviour functions, in the model investigated in this thesis, is the velocity vector of an object. Outputting positions is not useful because desirable behaviours involve smooth movements of objects.

5) To create behaviour functions, it is sufficient to consider only the positions of objects at the two preceding time-steps. When just the one previous time step is used, the behaviours and responses are called T1. Responses considering a second previous time step are called T2.

6) It is most effective to combine the basic input positions in fundamental ways to form series of separate magnitudes and directions. The type of behaviour obtained

depends most strongly on the particular combination of magnitudes and directions used.

7) T1 functions use distance and inter-object direction. Systems with only these elements can be easily defined in terms of simple linear equations. They allow for the modelling of many types of interactions between objects. The type of distance found most useful is distance between perimeters, as opposed to distance between centres -- it is only with the former that such effects as bouncing can be achieved.

8) The most important type of T2 functions are those with which an object responds to itself. This self-oriented behaviour is used to maintain momentum in a system or to prevent a system from going out of control. The elements which generate this type of behaviour are velocity magnitude and direction of movement.

9) Although most of the complexity of a system can be achieved by using just T1 and self-oriented elements, it is necessary to use certain other magnitudes and directions for particular effects. Of particular interest are:

-- the velocity and direction of movement of the stimulus;

-- the change in distance, which allows an interaction to be different on approach and retreat;

-- the time-to-reach (ratio of distance to change in distance), which allows an inter-object interaction to be independent of velocity;

-- the ratio of distance to velocity of the stimulus, which allows for effective capture behaviour; and

-- several compound directions (reflection, orbital and interception) which allows objects to move in various physically meaningful ways.

10) There is little use for certain magnitudes and directions, because the behaviour generated by functions in which they are used is either unstable or can be more easily generated in other ways. Quantities in this category are those involving the direction and magnitude of relative motion and inverse-square distance.

11) It is useful to divide responses into levels of importance with more weight given to the most important. Highly weighted responses prevent physically impossible situations, allowing for bouncing off walls or constraining maximum speed. Intermediate weight responses create the behaviour that is of interest. Low weight responses create background behaviour that maintains movement in the absence of other responses. The high and low weight responses have been found extremely easy to create.

12) Useful responses tend to be found by chance or by means of a considerable amount of trial-and-error experimentation. With a low-level language it is not easy to deliberately design a function which will cause an object to move in a particular animate manner. However, once they have been discovered, response functions tend to be robust in the sense that their parameters can be changed through a wide range of values with similar behaviour still appearing. Thus, for example, 'pushing' might be changed to 'pushing hard' due to a change in parameter values. For behaviour functions to be a useful animation tool it would be desirable for the animator to have a library of functions which generate standard behaviours, and for each of these functions to have well designed control parameters.

13) There are several problems that must be borne in mind when working with behaviour functions. On the psychological side are maintaining acceptable velocity, handling behaviour that would cross screen edges, and preventing unnatural movement by dealing with boundaries of response summation. Problems of a mathematical nature are positive and negative feedback, oscillation, discretization and exact repetition.

14) Animation with behaviour functions can be done in real time. Using an IRIS 2400 with a floating-point board, it is possible to animate over ten objects at once. Consid-

erable restriction must me placed on the way partial responses are specified to allow for optimization; it is a concern that if these restrictions were removed, less could be done in real time.

15) It is possible to create behaviours with the criteria outlined in the psychological and animation literature and to observe such impressions as causality, animacy and intentionality.

16) Only a relatively superficial look at behaviour function animation is presented in this thesis. It is the author's opinion that this methodology has great potential and that further research will be highly rewarding.

Appendix A.

MAPPINGS BETWEEN LANGUAGE VERSIONS

The animation description language described in chapter 3 was designed for ease of understanding on the part of the user. The language actually used by the PAM implementation, on the other hand, was designed for ease of parsing -- it was not an objective of this thesis to be a compiler-writing exercise. The PAM language follows a strictly keyword-parameter format with the parser expecting a keyword followed, possibly, by one or more required parameters. Input in the PAM language is strictly free-form with the any number of spaces, tabs or new-lines separating each the keywords and parameters. Additionally, any of the keywords can be abbreviated to a minimum of three characters.

The only difference between the two language versions is in the way partial responses are specified. The two languages are functionally identical and there is a one-to-one mapping between them. The following shows the mapping; for each language element that differs, the formal version is shown first and the PAM version follows.

The following is the formal language:

```
<p_response_description> ::=
      return(<sub_response>)
    | return(<sub_response>)
      <p_response_description>
    | if <condition> return(<sub_response>)
    | if <condition> return(<sub_response>)
      <p_response_description>
    | if <condition> return(<sub_response>)
      else <p_response_description>
```

The following is the exactly corresponding PAM language:

```
<PAM_p_response_description> ::=
      <PAM_sub_response>
    | <PAM_sub_response>
      <PAM_p_response_description>
    | <PAM_condition> !<PAM_sub_response>
    | <PAM_condition> !<PAM_sub_response>
      <PAM_p_response_description>
    | <PAM_condition> !<PAM_sub_response> only
      <PAM_p_response_description>
```

Formal language:

```
<sub_response> ::=
   <magnitude_section>
   <direction_section>
   <weight_section>
```

PAM version:

```
<PAM_sub_response> ::=
   <PAM_magnitude_section>
   <direction_section>
   <weight_section>
```

Formal language:

```
<magnitude_section> ::=
      <magnitude>
    | <magnitude> * <fval>
    | <magnitude> + <fval>
    | (<magnitude> + <fval>) * <fval>
```

PAM version:

```
<PAM_magnitude_section> ::=
      <magnitude>
    | <magnitude> mult <fval>
    | <magnitude> plus <fval>
    | <magnitude> plus <fval> mult <fval>
```

Formal language:

```
<condition> ::=
      <magnitude> > <fval>
    | <magnitude> < <fval>
    | <magnitude> > <fval> & < <fval>
```

PAM version:

```
<PAM_condition> ::=
      <magnitude> gt <fval>
    | <magnitude> lt <fval>
    | <magnitude> gt <fval> lt <fval>
```

It should be noted that definitions not shown remain identical (e.g. <magnitude> <direction_section> <weight_section>). Another point worth noting is that PAM has greater flexibility in specifying some items than shown here. For example, when a <PAM_magnitude_section> is preceded by a <PAM_condition> with the same magnitude, the second magnitude with its preceding exclamation mark may be omitted. Also, the ordering of keywords within sub-responses is not enforced, although for consistency and readability the syntax shown above should be followed.

BIBLIOGRAPHY

Borning, A. H. The Programming Language Aspects of Thinglab, A Constraint-Oriented Simulation Laboratory. ACM Transactions on Programming Languages and Systems 3(4):353-387, October 1981.

Braitenberg V. Vehicles: Experiments in Synthetic Psychology The MIT Press, 1984

Dewdney, A.K. Computer Recreations. Scientific American March 1987, 16-24

Greif I, Hewitt C. Actor Semantics of PLANNER-73. Proceedings ACM SIGPLAN SIGACT Conference. 1975, 67-77

Hewitt C., Bishop P., Steiger R. A universal modular actor formalism for artificial intelligence. Proceedings International Joint Conference on Artificial Intelligence. 1973, 235-245

Joy W. and Horton M. An Introduction to Display Editing with Vi. May, 1984

Lethbridge T. PAM Technical Documentation UNB School of Computer Science, 1987.

Lethbridge T. PAM User's Guide UNB School of Computer Science, 1987.

Magnenat-Thalmann, N., Thalmann, D. Computer Animation. Theory and Practice. Springer-Verlag, Tokyo, 1985.

Marion A., Fleischer K., Vickers M. Towards Expressive Animation for Interactive Characters. Proceedings. Graphics Interface 84. p.p. 17-20.

Michotte, A. The Perception of Causality. Methuen, 1963.

INDEX

VITA

Candidate's full name:	Timothy Christian Lethbridge
Place and date of birth:	London, England July 7, 1963
Permanent address:	725 Murray Avenue Bathurst, New Brunswick
Schools attended:	Bathurst High School Bathurst, New Brunswick 1978-1981
Universities attended:	University of New Brunswick Fredericton, New Brunswick B.Sc.(CS) 1985 Mount Allison University Sackville, New Brunswick French Immersion, 1981