Practical Behavioural Animation Based On Vision And Attention

Mark Fyfe Pietro Gillies

University of Cambridge Computer Laboratory
Technical Report TR522

September 2001

To My Parents

Declaration

All of the research described in this dissertation has been entirely my own work with the exception of some of that described in section 6.2.1 and appendix A which were done in collaboration with Tony Polichroniadis.

This dissertation does not exceed the word limit of 60 000 words.

Abstract

The animation of human like characters is a vital aspect of computer animation. Most animations rely heavily on characters of some sort or other. This means that one important aspect of computer animation research is to improve the animation of these characters both by making it easier to produce animations and by improving the quality of animation produced. One approach to animating characters is to produce a simulation of the behaviour of the characters which will automatically animate the character.

The dissertation investigates the simulation of behaviour in practical applications. In particular it focuses on models of visual perception for use in simulating human behaviour. A simulation of perception is vital for any character that interacts with its surroundings. two main aspects of the simulation of perception are investigated:

- The use of psychology for designing visual algorithms
- The simulation of attention in order to produce both behaviour and gaze patterns.

Psychological theories are a useful starting point for designing algorithms for simulating visual perception. The dissertation investigates their use and presents some algorithms based on psychological theories.

Attention is the focusing of a person's perception on a particular object. The dissertation presents a simulation of what a character is attending to (looking at). This is used to simulate behaviour and for animating eye movements.

The algorithms for the simulation of vision and attention are applied to two tasks in the simulation of behaviour. The first is a method for designing generic behaviour patterns from simple pieces of motion. The second is a behaviour pattern for navigating a cluttered environment. The simulation of vision and attention gives advantages over existing work on both problems. The approaches to the simulation of perception will be evaluated in the context of these examples.

Acknowledgements

I would like to thank my supervisor Dr Neil Dodgson for all his help and advice while I was working on my PhD. I would also like to thank all the members of the Rainbow Group for countless discussions and impromptu help, and in particular Tony Polichroniadis for the use of his Gepetto system and for always being around to talk over new ideas.

I would also like to thank the EPSRC for funding this research and the Cambridge University Computer Laboratory for providing the resources that I needed.

Finally I would like to thank my parents for their support and advice over the years and especially for reading the thing; my girlfriend Debs and all my friends for just being around.

Contents

Chapter 1

Introduction

Recent years have seen a huge increase in the use of 3D computer animation. At the high end this is due to high quality renderers running on large clusters of computers now having the power to produce images that are of a high enough quality to be blended seamlessly with live action film images. At the same time low cost hardware allows the average user access to real time interactive 3D animation.

Both of these applications require the animation of human or anthropomorphic characters to make them more compelling. The success of the animation as a whole often rests on the quality of animation of the characters. Computer animation allows the possibility of these characters being controlled by the computer and having some degree of autonomy from a human animator. This is desirable for a number of reasons, as described in chapter 2. In order for the computer to control the character it must be programmed with some sort of simulation of human behaviour. For this simulation to be effective it must include the characters' interaction with their environment and to do this it must simulate the characters' perception of the environment. This dissertation looks at two aspects of the simulation of perception, focusing in particular on visual perception.

The first aspect is the use of results from human psychology research in designing visual algorithms. This aims to produce efficient algorithms that generate realistic behaviour.

The second aspect is the simulation of attention, the focusing of a person's perception onto a particular object or group of objects. This is important for two reasons. Firstly it is an important aspect of perception and is vital to the correct simulation of behaviour. Secondly it enables the computer to animate the characters' eye movements, enhancing their realism and expressivity.

These two aspects are discussed in the context of previous work in the field. Solutions to the problems involved are suggested and implementations of these solutions are described. These implementations are applied to two examples, a general method for designing simple behaviour patterns and a method by which a character can navigate a cluttered environment. The ideas presented are evaluated based on their use in the examples.

1.1 Structure of the dissertation

The dissertation is divided into 6 further chapters. The approach is described in chapters 3 and 4 and it is applied to two examples in chapters 5 and 6. The content of each chapter is described below.

Chapter 2 gives a general overview of the animation of human-like characters. It describes the main applications and previous work. It gives general motivation for the work described in the rest of the dissertation and outlines some general approaches to the field. It does not, however, contain any new work.

Chapter 3 discusses the simulation of vision for computer animation. It describes the approaches used by other researchers and presents my approach. It describes two new algorithms developed with this approach.

Chapter 4 discusses the simulation of attention. After discussing previous simulations it describes a new method and its implementation.

Chapter 5 describes a method by which a user without knowledge of programming can design a simple behaviour pattern for a character. It describes how using the vision and attention algorithms of the last two chapters a new approach to the problem can be used.

Chapter 6 shows how the simulation of vision and attention can provide a new solution to a common problem in computer animation: how to navigate a cluttered environment without colliding with obstacles.

Chapter 7 evaluates the work described and suggest further developments.

1.2 Scope and contribution

This work aims at producing partial simualations of behaviour that can be used in conjunction with human intervention, rather than a complete simualation. The work looks at the behaviour and motion of a character, but not its graphical appearance. The work is presented as a system that produces animations that are available for subjective evaluation. More formal evaluation, though desireable is not considered to be part of the current work.

There are two main contributions of this work. Firstly an investigation of the use of psychology for designing animation algorithms. This is shown to be useful but problematic. In particular a provably correct psychologically based simulation of behaviour is unlikely to be possible. The second contribution is a demonstration of the usefulness of a simualtion of attention for behavioural animation. Part of this is a presentation of a method for doing this simulation.

Chapter 2

Virtual Actors

The first part of this chapter introduces the general area of research into virtual actors. It describes their current uses in the entertainment and other industries and the constraints and demands of the various uses. Current research problems are then described together with suggested solutions. The second part of the chapter focuses on the area with which this dissertation is mostly concerned: the simulation of behaviour, how it has been done in the past and how it can be applied to current applications of virtual actors.

Definitions The term *virtual actor* is used to mean an (often humanoid) character in a piece of computer animation. A virtual actor will normally have a graphical 'body', a method for controlling motion and possibly some degree of autonomous behaviour. Equivalent terms are *synthetic actor* used by Magnenat-Thalmann and Thalmann[MTT87] and *synthespian*. Virtual actors are often referred to merely as actors and this thesis will use this convention. If a non-virtual actor is intended he or she will be referred to as a *real* or *human* actor. The term *autonomous actor* or *autonomous agent* (e.g. see Maes[Mae95]) is often used for a virtual actor which is entirely controlled by the computer using a behavioural simulation without the need for intervention by an animator, the actors described in this thesis do not generally fall into this category.

2.1 Applications of Virtual Actors

Virtual actors are used in two main applications both connected to but not exclusive to the entertainment industry. The first is film and the second is interactive virtual environments. Each has a different set of problems that need to be solved.

2.1.1 Film

Virtual actors are increasingly being used in animated and live action films. They are used to produce characters for animated films which are entirely animated on a computer such as "Toy Story" but also for live action films where a character's appearance is too inhuman to be acted by a real actor or where the character performs actions which would not be possible if performed by a real person, for example Jar-Jar Binks in "Star Wars Episode I: The Phantom Menace". There is no need for interaction[1] so the actors do not need to be able to react autonomously to events and they do not need to be able to execute in real time. However it is very important for motion and behaviour to be realistic, particularly in live action film where the virtual character needs to blend seamlessly with the real ones. Also it is very important for the characters to express personality and emotion in their actions. In film the actor's actions are closely scripted by an animator for a particular scene and so the actor requires very little autonomy. The general approach is to use *motion capture*. This is the process of recording the motion of a real actor using a magnetic or visual tracker. This motion is then post-processed, largely by hand and applied to the character. This process is extremely time consuming and expensive as not only must each action of the character be captured on an expensive motion-capture system but the process of post-processing and applying the motion to the character requires considerable time from a skilled animator. Ways of speeding up the process by reducing the amount of motion capture needed and the amount of post-processing would greatly reduce the cost of the technology and make it available to a much wider range of film makers.

2.1.2 Virtual Environments

An *interactive virtual environment*[2] is a, generally 3D, computer generated space containing objects and characters. The user can move around this space and interact with it. There are many uses of virtual environments. Currently the most popular is computer games though they are being more and more used for architectural and engineering and are used to some extent in education where a virtual environment can allow people to see computer simulations of historical events and far off places. With the growing use and bandwidth of the internet, virtual environments are moving on from being individual experiences to being a space where people can interact with each other over a network, becoming so called *collaborative virtual environments*. Multi-player computer games are one of the most important early users of the technology but it is predicted to spread to many other areas such as on-line teaching, social areas, virtual meeting, virtual museums and internet retail environments.

There are two main type of virtual actor in a virtual environment. There is normally a representation of the user in the virtual environment (called an *avatar*) and also computer controlled characters with which the user can interact.

[1]It might in fact be useful, in films where real and virtual actors appear concurrently, for the real actors to be able to interact directly with the virtual actor while acting scenes. However, the problems involved are as much to do with finding a way of embodying a virtual actor in the real world as with making the actor interact with the real actors. As solutions to the former problem currently seem far off it is too early to think about interaction problems in this application.

[2]I will use the term to mean both immersive and non-immersive environments

Avatars

In a single user environment the user will often have some sort of graphical representation. The requirements of this representation are low as there is no one there to observe its actions. However, the avatar is still important. Slater and Usoh[SU93] have shown that using an avatar to represent a user increases that user's sense of immersion. Barfield, Zeltzer, Sheridan and Slater[BSZS95] note that it is important for the user to identify with the motions of the avatar shown on the display. For example, by having the avatar mimic the motion of the user.

In a collaborative environment, however, the users' avatars become very important as they must communicate a user's actions to the other users. Studies such as McBreen and Jack's[MJ00] and Garau, Slater, Bee and Sasse[GSBS01] show that avatars are important but they are only useful if their behaviour is believable and suitable for the context. The avatars are much less autonomous than the computer controlled characters as their actions are controlled by the user. However, they must be easy to control in real time, allowing the user to interact effectively with the environment. It is also useful if they are expressive, allowing the user to display a personality and mood within the environment. This not only allows each user more expression but also increases the impression that the avatar represents a real person rather than a robotic figure, the sense of co-presence. Finally, if the avatar is part of a networked environment it is important that the information needed to control the avatar is small enough to travel in real time on low bandwidth systems so that the avatars appear to be reacting in real time on all the machines on the network.

The range of techniques for controlling avatars is great as Mine's survey shows[Min95]. One extreme is simple keyboard and mouse interaction where the computer has to animate the character based on very limited input. At the other end are systems which record the user's motion and facial expressions and map them onto an avatar, what Slater and Usoh[SU94] call *Body Centred Interaction.* An extreme case is the ALIVE system[MDBP95] where real-time video of the user interacts with virtual objects. Full body systems provide a far stronger sense of immersion and communication and so are a much better way of interacting with an environment. However, there are a number of reasons why they will not be universally used, at least for a long time. They are expensive and require a lot of space in order to give the user freedom of movement so they are not suitable for people who wish to use *virtual environments* from a home computer. It is also not always desirable to have the user's own body language and actions mapped on an avatar, the user might want to present a different image in the environment than in real life and this would require not just a different graphical appearance but also different motion and behaviour.

Computer controlled characters

These are characters placed in the environment by the designer to fulfil some specific role. They are controlled by the computer and must be able to interact with the users of the environment autonomously, without any human intervention. In many ways the features that are desirable in the behaviour of an avatar are also desirable for computer controlled characters. However, there is no requirement to map the user's desired behaviour onto the character. This has the negative aspect that the behaviour of the character must be generated without any user input. As such there is a great requirement for realistic algorithms to control their behaviour. Currently these algorithms are primitive and in general the character's actions are limited. Almost all *autonomous actor* research falls into this category. Two interesting examples are Blumberg and Galyean's[BG95] dog that interacts with users in the ALIVE system and Thórisson's talking head characters that hold a conversation with the user[Thó98].

2.2 Problems

In this section I will outline some of the main research problems connected with virtual actors and previous work that has tackled these problems. My work is mostly concerned with behaviour; I will describe this in more detail in section 2.3.

2.2.1 Low-level Motion

This is the problem of how to represent and control the parts of an actor's body so as to produce realistic motion. The main approach is to represent the actor's limbs as *articulations* composed of a series of rigid links connected by joints with one or more degrees of freedom. These are normally controlled by *inverse kinematics*, using the position of the end of an articulation to control the joints(for example, Korein and Badler[KB82]). This has been applied extensively to simulate walking (e.g. Girard and Maciejewski[GM85]). Inverse kinematics has been extended to hierarchies of articulations representing whole body (e.g. Zeltzer[Zel82] or Badler *et al.*[BMW87]). There have also been attempts to use dynamic rather than kinematic control, for example Wilhelms[Wil87] or Raibert and Hodgins[RH91].

The above techniques are useful for humans and animals with limbs but other techniques are needed for other types of actor, for example Miller uses a spring-mass system to represent snakes and worms[Mil88] and Tu and Terzopoulos use similar methods for fish[TT94].

A very common technique for producing motion is *motion capture.* This consists of using magnetic or visual sensors to record the motion of various parts of the body. If this sort of system is applied to an human actor performing an action that action can be recorded and then applied to a virtual actor. Recently there has been great interest in transforming pieces of motion captured motion so that they can be used in more than just the simple context they were originally recorded for. Gleicher[GL96, Gle97] developed an interactive motion editing system based on *Space-Time Constraints.* Space-Time Constraints, developed by Witkin and Kass[WK88] are constraints on motion which are solved for not merely spatially in a single frame but temporally across all the frames of the animation. Gleicher[Gle99] later used Space-Time Constraints to retarget motion from one character to another character of a different body shape. Rose et al[RGBC96] used Space-Time Constraints to automatically create smooth blends between pieces of motion.

Polichroniadis[PD99] uses a classifier system to automatically find appropriate start and end points for such blends. Popović and Witkin[PW99] extend Space-time Constraints by solving them in a way that produces dynamically feasible motion. They also added the ability to add dynamic constraints and change the dynamic properties of the motion.

Witkin and Popović[WP95] propose a general scheme for warping pieces of motion. It consists of a warp of the spatial properties of the motion of the form:

$$\theta'(t) = a(t)\theta(t) + b(t)$$

where θ is the original motion, θ' is the new motion and a and b are a time varying scale and offset. There is also a time warp which is a mapping from the timing of the original to a new timing. Bruderlin and Williams[BW95] use multiresolution signal processing techniques to transform motion. Lee and Shin[LS99] use hierarchical B-splines to interactively warp and edit motion.

2.2.2 Appearance

Modelling an actor as an articulation of rigid links is normally good enough for producing motion but does not produce a realistic outward appearance. There have been various approaches to simulating soft bodies which will move with the underlying articulation. Magnenat-Thalmann and Thalmann[MTT87] use a form of operator called a joint-dependent local deformation to map individual sections of flexible skin onto a skeleton. Van Overveld[vO90] deforms a skin by weighted interpolation of the displacements of points on the skeleton. Chadwick, Haumann and Parent[CHP89] use free-form deformations to simulate muscle deformations for an articulated character. Gascuel and Verroust[GV91] provide *collision detection* for soft tissue. There is a lot of interesting research to be done in this area but it is rather peripheral to the subject of this dissertation.

2.2.3 Behaviour

Behavioural Animation aims to endow a virtual actor with some degree of intelligence and ability to interact with its environment so that it can produce its own behaviour independent of, or loosely scripted by, an animator. The actor is normally controlled by a series of specific *sensori-motor skills* which I will call *behaviours.* Reynolds was one of the first innovators in this field with his flocking 'boids'[Rey87]. Other notable work in this area includes McKenna, Peiper and Zeltzer's virtual roach[MPZ90] and Tu and Terzopoulos' artificial fishes[TT94]. As this is the area in which I am working I will discuss it at greater length in section 2.3.

Designing Behaviours

A major problem with behavioural animation is that designing behaviours is a difficult programming task but that ideally behaviours should be designed by animators who generally do not have very extensive programming skills. Wilhelms and Skinner[WS90] attempt a solution based on nodes representing simple behaviours which can be built into complex networks. Haumann and Parent[HP88] suggest a library of simple behaviours which can be combined to produce more sophisticated behaviours. Del Bimbo and Vicario[DBV95] suggest a different approach, they propose that the animator explicitly controls one actor in the virtual environment. This character demonstrates a set of motions which the other actors learn. Ydrén and Scerri[YS99] use a map on which arrows are drawn to design soccer playing strategies for robots. This is analogous to the way that soccer coaches demonstrate strategies to real players. This area of research is discussed in more detail in chapter 5.

2.2.4 Social behaviour, conversation and gesture

There is one aspect of behaviour that has been studied very fruitfully but is beyond the scope of the current work. This is the simulation of social behaviour, conversation, body language and gesture. Cassell *et al.* have done extensive work in this area[CPB$^+$94]. For examples Vilhjálmsson and Cassell[VC98] designed and actor which has appropriate eye movement during conversation. Thórisson's agent holds a conversation with the user while producing gestures[Thó98]. Cassell, Vilhjálmsson and Bickmore[CVB01] analasise text in order to produce suggestions for appropriate body language and gesture for an actor speaking the text. This area will be discussed somewhat in chapters 4 and 7.

2.2.5 Direction and Control

If virtual actors and behavioural animation are to be useful in animation there must be some way in which an animator can direct the actions of the actor. This can be direct control taking over from a behaviour or higher level control interacting with the behaviour. Haumann and Parent[HP88] propose using scripts which change actor state in the same way as behaviours and so override behaviours. This means that scripts can act transparently to other behaviours and actors. McKenna, Pieper and Zeltzer's roach [MPZ90] can be controlled at various different levels: using ASCII scripts; interfacing with other programs using function calls; using graphical sliders to affect internal state, and using a 'data glove' which is an input device roach reacts to as if it were an object in the environment. Blumberg and Galyean's cartoon animals[BG95] also uses a multi-level system, the main means of control are: changing the values of internal variables (e.g. hunger, tiredness); changing aspects of a behaviour (e.g. its target or the probability of it being triggered); starting a behaviour irrespective of the state of an actor, or providing imaginary sensory input.

2.3 Behavioural Animation

As explained earlier Behavioural Animation aims to simulate the behaviour patterns of actors in an attempt to automatically generate part of an animation. The most common approach is to divide the control of the actor into a set of interacting simple behaviour patterns, called *behaviours*. Each of these is given some form of input from the environment and the output of all the behaviours is combined to give the actions of the actor. This paradigm is strongly influenced by *agent based artificial intelligence* (which is discussed in section 2.3.1) and theories from *ethology* and psychology which claim the behaviour of animals and humans consists of a series of interacting primitive sensori-motor skills (see for example Paillard [Pai80] on humans and Arbib and Liaw [AL95] on frogs). This is not the only approach, for example Mah, Calvert and Havens use constraint based reasoning as a basis for behavioural animation [MCH94], but most of the work I will describe below follows this general scheme. Reynolds was one of the first innovators of behavioural animation, he simulated the flocking behaviour of birds using a set of three simple rules:

‘1. Collision Avoidance: avoid collisions with nearby flockmates.

2. Velocity Matching: attempt to match velocity with nearby flockmates.

3. Flock Centring: attempt to stay close to nearby flockmates.’[Rey87, p28]

The outputs from these rules are combined and the final output enables the simulated flock to produce convincing flocking behaviour. There has been extensive work on behavioural animation since then. Haumann and Parent [HP88] produced a test-bed aimed at producing behavioural animation but used it mostly for physics based simulation. McKenna, Peiper and Zeltzer [MPZ90] produced an ethologically based simulation of a cockroach. The controller is divided into a sensori-motor level which deals with simple reflexes and the kinematics of motion and a reactive level which converts higher level stimuli from the environment into instructions for the sensori-motor level about how to move. Each leg of the roach was driven by an independent oscillator and a set of reflexes. Only the frequency of the oscillators is controlled centrally. The oscillators are coupled in such a way that different gaits are produced at different walking frequencies, as in a real cockroach. Renault, Magnenat-Thalmann and Thalmann [RMTT90] have an actor that uses maps and planning to navigate a virtual environment. Zeltzer and Johnson’s [ZJ91] actor consists of a distributed set of processes which can plan motor behaviour without explicit maps or reasoning. Tu and Terzopoulos’ simulated fish [TT94] had motivations as well as simple reactive behaviour. These motivation are represented as numerical variables (such as hunger and libido). They initiate behaviour patterns which can be interrupted by reactions to more pressing environmental stimuli (such as avoiding a predator). Blumberg and Galyean [BG95] created a complex ethologically based actor with multi-level behaviour. Each behaviour is given a priority based on the internal motivation of the actor and the external stimulus. The behaviour with the highest priority becomes dominant and gains control over the actor. However, other behaviours can still influence the actions of the actor.

2.3.1 Agent Based AI and Robotics

Work on virtual actors and behavioural animation has been greatly influenced by *agent based artificial intelligence* (see Maes[Mae94] or Meyer[Mey98] for a longer introduction).

This is a new approach to the problems of AI as applied to robotics. Traditionally these problems have been divided into a set of functional units such as a vision module, a world model, a planner and an execution module. There has been an attempt to make general modules for these sub-tasks which can be combined together to solve a whole range of problems. Rather than try to build such a general functional unit, in isolation from other units and the target problem, the Agent Based approach aims to find simple solution to a single problem. These solutions combine all parts of the problem, vision, planning and action in such a way that the constraints of the problem allow short-cuts and simplifications to the solution. This is similar to what happens in real animals, for example frogs do not have general reasoning systems in their brain, rather than analysing the images on the retina to recognise flies they snap as a reflex reaction to a small moving patch on their retina (which will often be a fly)[AL95]. Brooks, in his work on robotics, originated many of the ideas of agent based AI. His *subsumption architecture*[Bro90] enables simple behaviours to be combined to produce more complex agents. The problem of building a high level behaviour can be simplified by assuming the existence of lower level behaviours which can be invoked when needed or can override the high level competence if needed. For example, in [Bro85] the existence of a behaviour which avoids environmental obstacles makes it easier to build a robot which can navigate around an office style environment. Using the agent based techniques many of the traditional AI modules become unnecessary, for example directly sensing the environment whenever necessary removes the need for an explicit representation of the world[Bro91].

The agent based approach has been the dominant approach to the simulation of behaviour for computer animation. Blumberg and Galyean is a good example[BG95]. The work described in this dissertation is implemented within Gepetto, an agent based virtual actor system (see appendix A for more details) and a subsumption style architecture is used for the navigation agents of chapter 6.

Many of these ideas can be transfered from robots in a real environment to actors in a virtual environment and the general methodology can be useful for creating actors. However, chapter 3 mentions some reasons why the two cases are different and different techniques are needed.

2.4 Simulating human behaviour in real applications

One of the most notable facts about behavioural animation in general is that it is prohibitively difficult, involving as it does a complete simulation of human behaviour which amounts to a solution of the hard AI problem. What is more, even if a perfect simulation of human behaviour were possible in an animation context it is not clear that it would be useful. A behavioural simulation implies a character with a “mind of its own” acting entirely in accordance with the results of its simulation. This is radically different from the current conception of a character in film or virtual environments (or any medium) where the action is directed by a human, an artist. It is likely that the human capacity for artistic creation will remain far superior to that of a computer for a long time if not for ever and so it is highly undesirable for a computer to take over the task of directing the behaviour of characters in a piece to any degree that will interfere with the desires of the animator/director.

The main problem with behavioural animation is that human behaviour is very complex and therefore difficult to simulate. This inevitably means that the behaviour produced will tend to be simplistic and unrealistic. This is fine for the simulations of insect or fish behaviour that are common in the behavioural animation literature but not very useful when trying to simulate humans. It can reasonably be assumed that a complete simulation of human behaviour is not possible, at least for the forseeable future. This means that the task of research into behvioural animation should attempt to find ways of making it into a useful tool without it being a complete simulation.

Another problem with behavioural animation, as noted by Reynolds[Rey87], is that it can be difficult to get the actors to do what the animator wants, they create their own behaviour but do not know what behaviour is desired of them. The actors can have a very accurate simulation which produces correct results based on the actors' "desires" and "personality" but this is no use if the result is not correct in the context of the plot of the film. There should therefore be some mechanism for making the simulation follow the needs of the animator or director.

Fortunately, directorial control can be made to make up for the deficiencies of the simulation. The director can make difficult and important decisions concerning the character's behaviour leaving less important and hopefully simpler choices to the simulation. A small amount of human control can add life to a computer controlled character while not taking a large amount of effort, so long as a large part of the behaviour is simulated.

In order to design this sort of simulation it is important to know what sort of controls it is desirable for the director to have. In order to do this it is necessary to look at the various possible applications of behavioural animation. The following sections discuss what I see to be the three main uses of virtual actors:

Film characters Animated films generally have an animator scripting and controlling the action and the motion of the characters. As such it is possible to produce an animated film without any sort of behavioural simulation. Animated characters in large budget films are generally either animated by motion capturing each individual sequence, as in Star Wars: The Phantom Menace or by hand animation as in some sequences from Toy Story. While these techniques produce extremely good results they are very time consuming and expensive which limits their use.

The main application of behavioural animation in film would be to make the animation cheaper and faster to produce, and therefore more accessible to lower budget productions, by automating some part of the animation process. Though the quality of the motion and behaviour produced may not not be as high as for a fully motion captured or hand animated piece, the greater accessibility of these techniques would be very valuable.

Given these requirements, the main aim of behavioural animation in film is to arrive more quickly at the result that the animator wants, or at least a good approximation. For this to be possible the animator must have considerable power to direct the actions of the character and must also be able to directly change the actions if the simulation has produced undesirable results. Also the simulation should be fairly limited in scope as it is far more desirable to have a method which only attempts to simulate simple behaviour (and as such requires large amounts of animator input to guide the simulation) but which produces good results than one which performs a complex simulation requring very little intervention by the animator but which produces poor or undesirable results.

Avatars Avatars in virtual environments are controlled by a user of the environment. They are the personification of the user in the environment and must perform the actions that the user wants to perform. They are therefore similar to characters in films in that the human user must have a fairly close control of the character's behaviour. However, the situation is different in a number of ways. Firstly, the user will generally be using the environment for entertainment or as a relatively minor part of their profession and as such cannot be expected to be an expert and have extensive training in using the tools. Secondly, films are produced off line but characters in virtual environments must react in real time. This not only means that the algorithms controlling the characters actions must run in real time but that the user must be able to enter all the input to control the character in real time, precluding complex, detailed control. Finally, virtual environments are often networked so that the bandwidth taken by the control of the character must be limited. All these problems indicate that controlling an avatar has to be very simple with small commands producing detailed results. Thus the role of behavioural animation must be to give the avatar the intelligence to interpret a simple command into a complex sequence of actions.

Another problem with networked environments is latency, the avatar must in some way respond effectively in the lag between the user sending a comand and it arriving at the other end. Behavioural animation can be helpful here to. By giving the avatar the ability to react to certain events independently, it can maintain the illusion of life while waiting for the user's commands to arrive.

These considerations mean that for avatars, behavioural animation must be able to enact simple, high-level commands in real time. The accuracy and exact correctness of motion is not as critical as for film. Some ability to interact with the environment is, however, also important as the character must deal with the environment independently for short periods due to the latency problem. It is also important because the user does not have time to input details of all the objects in the environment in his or her commands and so the avatar must be able to deal with those that it encounters.

The actual requirements for the behaviour of an avatar depend on the method of interaction. For full body interaction systems where the motion of the actor is directly mapped onto the avatar there is no need for autonomous behaviour, however, these systems are rare. More common are systems where some of the user's motion is recorded and the behaviour of the avatar has to be infered from this. In this sort of system it would be useful to have a behavioural model which is used to interpret the user's motion and use it to create avatar behaviour. The user's motion could be interpreted as a certain state of the behavioural model and the models could enhance the motion with other aspects of behaviour, for example, eye movement in non eye-tracked systems. Finally, the cheapest system of avatar control (and as such the one likely to become most widespread) is a normal keyboard, monitor and mouse setup. Since the user input information available in this setup is much lower than that required to fully specify the motion of a character a good behavioural model is needed. In general it ought to behave relatively autonomously but obey user commands.

Characters in virtual environments This form of actor is entirely autonomous, its behaviour is entirely controlled by the computer. This means that it has the greatest need of behavioural animation as all its behaviour must be simulated. However, it is also the hardest to apply behavioural animation to, as there is no user to guide the behavioural algorithms towards more convincing behaviour. The state of the art is still a long way from creating convincing algorithms for all of behaviour. There are, however, a few ways forward. Giving a character a well defined role can limit the need for a fuller simulation. The user will be encouraged to interact only in a limited number of ways. This means that, beyond a certain basis of low level actions, the characters can be controlled by a limited specialized piece of code. The designer could control the behaviour of the actors by means of a general script that determines the actors behaviour in various circumstances. This would not be a rigid script such as that for a film but one that is based on general reactions to events, to use Yrdén and Scerri's[YS99] example, it is closer to game plans designed by a soccer coach.

Another way of making characters more convincing is to endow them with a strong personality, emphasising differences between characters. If a character has a visible personality, users will be more willing to believe in them and overlook defects in their behaviour. It has been noted that in the ALIVE system, which is based on tracking user action with computer vision, users became very frustrated when the system failed to spot their attempts at interacting with inanimate objects but were much more forgiving when dealing with characters[MDBP95]. They were more likely to think that the character had failed to hear them or attribute the failure to some personality trait than that the system had failed. Thus when a character has sufficient personality to have an illusion of life, people will tend to interpret small idiocyncracies in their behaviour as displaying personality rather than as a loss of copresence. In particular Michotte has shown that people are willing to interpret a wide range of simple linear motions of simple geometric shapes such as circles as displaying intentionality and even something approaching emotion (e.g. "running away" which implies fear). Polichroniadis has also shown that it is possible to give a character the semblance of personality by using a few simple motion transforms[Pol00].

2.4.1 Behavioural competences

The conclusion that I have drawn from the above discussion is that it is not possible to create a useful complete simulation of human behaviour but on the other hand this is rarely neccesary. It is more important to create a limited simulation which an animator or designer can use to create the effect he or she wants. This means that the simulation has to (a) do something useful, (b) be easily invoked and used, and (c) be able to create the effect the animator wants rather than producing a single response to a situation. In particular, as the personality and individuality of characters is very important it should be possible to make different characters react differently to the same situation with the same algorithm controlling them. I therefore propose that, rather than attempting a complete simulation, behavioural animation should be aimed at producing a set of key competences for a character which perform a given action and can be invoked by the animator. Ideally this set would be easily extensible by an animator without programming skills. These actions can take arguments, for example, a behaviour designed to walk without collisions would take the end point of the walk as an argument and a behaviour which picks up a glass would take the actual glass as an argument. The decision of which action to perform would be made by the animator in an animation package or the user in a virtual environment. Specifying the action would be very simple as it would just consist of choosing the type of action and the argument, typically a point in space or an object. In fact, as each object might typically be the argument of only one or two types of action, specifying an object might automatically specify the action. This would allow the user of a workstation based virtual environment to specify complex behaviour in realtime with very little input. For body tracked users it might be possible to infer a competence from the user's motion and for the competence to add details to the final animation. This sort of inference is beyond the scope of this work, however. While avatars can be controlled solely by competences, characters in film could have their actions initially sketched out with competences and then the animator could adjust the details to produce the desired effect. This sort of approach is more suited to user controlled characters than computer controlled characters as it needs someone to choose the action to be performed. However, if the role of the computer controlled character is sufficiently well defined it might be possible to use competences as a building block for the behaviour of the character. Allowing the designer to script a behaviour in terms of a sequence of compentences.

As the actions should be performed differently by different characters, they should have a set of parameters to control how the action is performed and which can be set differently for different characters These parameters could change the way an algorithm works or even choose between a number of different alogrithms each performing the same task but producing different effects (you can get to the bottom of a flight of steps by walking down them or you can slide down the banister). Also the actual motion associated with the action should change. This is possible if the action is implemented by transforming existing motion rather than creating new motion. This means that the user can choose a particular piece of motion captured motion to use as the basis for a character thus increasing its individuality. This also has the advantage that using motion captured motion will produce more realistic motion.

Behavioural competences and how to create them will be discussed in more detail in chapter 5. Chapter 6 discusses a more complex competence for walking in a cluttered environment.

2.5 Summary

This chapter has given an overview of computer animated characters. It has discussed the various applications of animated characters and the areas of research involved, particularly the simulation of behaviour which is the main focus of the current work. This chapter has also described some of the features that a behavioural simulation would have to have if it is to be useful in real applications. The following chapters will describe behavioural simulations that have some of these features. The next two chapters describe models of perception that are useful for producing different type of behavioural competences, a psychologically based method of designing visual algorithms and a model of attention. The following two chapters describe how these models are applied to creating two types of competence, chapter 5 describes a general tool for building competences and chapter 6 describes a more complex competence for navigating a cluttered environment.

Chapter 3

Vision

The simulation of sensory perception is vital to the simulation of behaviour. Any complete behavioural simulation must include interaction with the actor's environment, which is fundamental to human day-to-day behaviour. This interaction implies that the actor must obtain information about the environment in order to perform an action. This means that any behaviour of this sort must by guided by perception. This chapter looks at visual perception, by far the most important sense in humans for gaining information at a distance and therefore the most important sense to simulate for animation. The aim of the work will not to be to simulate vision in general but to simulate visually-guided behaviour.

At first sight the task of simulating vision for animation seems similar to the task of designing sensory algorithms for the sort of autonomous robots described in section 2.3.1. However, there are two fundamental differences which must inform any attempt at simulating vision for animation. Firstly, the aim of the sensory systems in autonomous robots is to perform a task in the most efficient way possible. This is not necessarily related to the way humans perform a task and the most efficient algorithm may produce behaviour which would seem very strange if performed by a human. For robotics this is not a problem as no one expects robots to act like humans. In animation on the other hand the aim is to produce outward behaviour that appear to be human and so the algorithm chosen has to be less arbitrary than for robotics, it must both perform a task and perform it in a human-like manner.

The other main difference from robotics is that robots exist in real environments and virtual actors generally exist in virtual environments. Robots, therefore, must attempt to gain information about the environment through sensors which often give noisy, unreliable and impoverished data. Therefore complex computer vision algorithms are often needed to interpret this data and to translate the information into appropriate behaviour. In a virtual environment on the other hand all the existing information about the environment is already stored in the computer. This means that the intermediary stage of interpreting the input from sensors is unnecessary; information can be obtained much more quickly and reliably from the data structures storing the environment. This makes the task of vision much easier. However, it is still not trivial. Having complete information is not realistic; for example an actor reacting to events behind its back is not realistic. There need to be filtering mechanisms that ensure that the actor only receives information that would be available to a real person in an equivalent situation. More importantly, the use of vision needs to be part of visually-guided behaviour. This means that there still need to be mechanisms to translate the visual information into appropriate behaviour. There need to be algorithms which know what information to access in a particular situation and how to use that information to produce behaviour. These algorithms should ideally produce a low overhead to the animation system and therefore should minimise the information needed and the calculations needed on that information.

This chapter and the next look at how these constraints can be met in a behavioural animation system. This chapter looks at low-level visual algorithms and existing methods for simulating vision are explored. A new methodology based on the use of visual psychology is then described. Its implementation is described and it is then evaluated. The next chapter describes a related problem, the simulation of attention, where the actor is looking and what parts of the environment it is aware of.

3.1 Previous work

There are two main approaches to handling visual perception in the literature on behavioural animation. One uses *ad hoc* solutions for individual behaviours. The other uses computer vision techniques to simulate the low level vision processes. This section describes both approaches and various attempts to utilise them. Section 3.2 attempts to produce a methodology which has some of the advantages of both methods.

3.1.1 Ad hoc methods

The first attempts at perception driven behaviours used ad hoc methods which seemed convincing but were mostly geared solely to producing realistic perception in one particular behaviour pattern. In his work on flocking behaviour Reynolds [Rey87] attempted to provide the same information to his virtual birds (called boids) that a real bird would gain from sensory perception without directly simulating that sensory perception. He divides a boid's perception of the environment into two aspects which use different methods. Perception of other boids is used by the flocking behaviour and consists of merely returning the position and velocity of all the boids within a certain radius of the boid. Perception of environmental obstacles for collision avoidance is provided by an extra database of objects in the environment which uses simplified shapes. This is directly interrogated by the obstacle avoidance behaviour. This duplication seems rather wasteful and might tend to produce incorrect results compared with using the scene geometry directly.

McKenna et al [MPZ90] produced an animated cockroach which detected a "grabbing hand" and other objects in the environment by interrogating input devices and the graphical database. Tu and Terzopoulos [TT94] produced animated models of fish. They detect any object that is not fully occluded by another object and that is within a certain visibility radius and visibility angle.

Bordeux, Boulic and Thalmann [BBT99] model perception as a series of filters on the objects in the environment. Each filter selects a subset of the objects passed to it. This subset can then either be used by a behavioural algorithm or passed to another filter. Though the filters they implement are of the *ad hoc* type they could in theory use any vision algorithm. The idea of filtering objects in the scene is similar to methods used in chapter 4 for attention based filtering.

These methods can produce good results efficiently with simple techniques. They can also be well adapted to individual behaviours, producing a specialised, optimised algorithm in each case. However, their originators tend to express worries about their realism. Reynolds thinks his boids would have more realistic behaviour if they could actually see their environment rather than merely being given information about the position of objects. Tu and Terzopoulos suggest that more accurate behaviour could be produced by using computer vision techniques to simulate real vision and, in fact, their work was later extended in this way [TRG96]. The major problem is that though the database can be interrogated directly there still needs to be some sort of algorithm for using the data and for filtering it. Without a guiding principle these algorithms become rather arbitrary. There needs to be a better way of designing algorithms.

3.1.2 Computer vision based methods

These methods are characterised by rendering the scene from the point of view of the actor and then using computer vision techniques on the resulting images. These methods are normally motivated by considerations beyond the needs of merely animating actors in virtual environments. Blumberg and Gaylean [BG95] use this method in an augmented reality system (the ALIVE system) where an actor needs to be able to detect real objects as well as virtual ones. Terzopoulos, Rabie and Grzezczuk [TRG96] augmented the original artificial fishes with a sophisticated active, binocular vision system. This was aimed at testing computer vision techniques in a virtual environment rather than producing animations. Renault, Noser *et al* [RMTT90, NRTMT95] have been the only group in whose work the use of computer vision techniques has been purely for virtual environments. They use a computer vision type system augmented with extra information. Instead of simply rendering an image they render two buffers, one is a z-buffer, which makes available the distance of a pixel from the actor and the other contains an object identifier for each pixel, i.e. it gives the object which is visible at each pixel position. The position of an object can easily be extracted from the z-buffer. This technique can produce good results.

This approach has the advantage that it can use a wealth of algorithms that have already been developed in computer vision research. It also gives for free various aspects of vision that need to be simulated such as visibility and occlusion testing (see section 3.4.4 for more discussion). However, it does have a number of disadvantages. Though the overhead of having to render the scene from the character's viewpoint as well as for display can be reduced by taking advantage of modern rendering hardware, it still seems wasteful and it is unlikely to be scalable to large numbers of characters, especially as interactive 3D applications often use all the available hardware resources for display rendering.

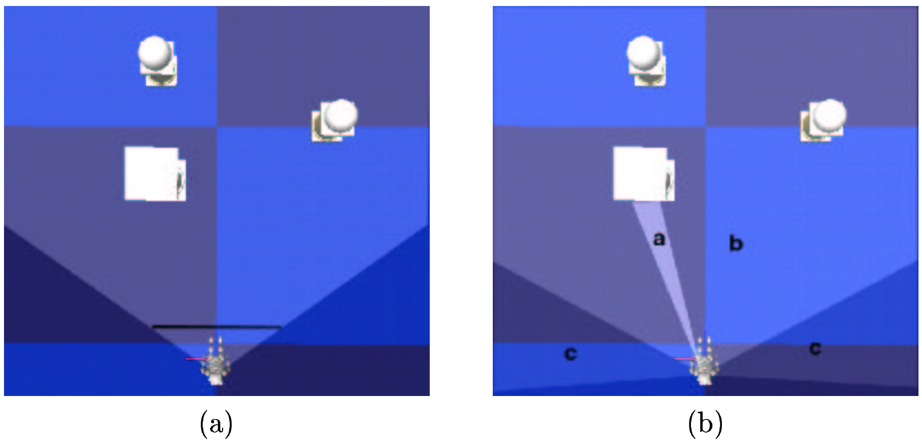


Figure 3.1: The contrast between image-based computer vision and real vision. (a) computer vision consists of a rectangular projection of a scene where all of the image is treated equally (b) real vision has a small high resolution fovea which corresponds to what a person is attending to (region a) there is also a periphery where the person may not be aware of everything but does notice some events (region b). If the person is moving their head the periphery can be considered larger as areas that are passing out of, or into, the field of vision can be considered (region c).

Also, though it is claimed that computer vision is a good model of vision it is often not. Current computer vision algorithms are still rather primitive compared to real human vision. This means that these techniques are unlikely to be able to produce higher level, more sophisticated behaviours. Renault, Noser *et al* do try to solve this problem by giving the visual system more information than is contained in an image.

These two problems stem from the fact that the computer has complete knowledge of the scene but that the computer vision type methods throw information away by projecting to 2D and then attempt to recreate it. Another problem is that computer vision deals with rectangular images as the basis of vision but real human vision is quite unlike this model, as shown in figure 3.1.

Real human vision consists of a relatively small centre of attention corresponding roughly to the contents of the fovea, the small high resolution area of the retina, region *a* in figure 3.1(b). The centre of attention is the area in which most intense visual behaviours take place. This is the area of vision where we read and operate tools. Outside of the centre of attention there is a large periphery of vision where the perceptual awareness becomes progressively reduced, region *b*. Though it might not be possible to see the details of objects in the periphery, some features, such as motion, can be picked up. Awareness in the periphery varies greatly, it can be very much reduced if one is concentrating on a task in the centre of vision but when the person is paying attention to their surroundings the periphery can be quite large, larger than a 180 degree arc if the person's head or eyes are moving, as the person will be able to pick up information about objects as they pass out of vision. Figure 3.1 illustrates the difference between real and image-based vision. People can even be aware of things that are not visible in the periphery of vision. Our memory gives us some predictive knowledge of objects that have passed out of our vision or have become occluded. We can also become aware of objects outside our range of vision using our other senses, particularly hearing, we will normally hear a car behind us before we see it. Thus there has to be some model of objects that are not in our visual field. Computer vision type methods can do this using some sort of internal model of the scene representing memory but this seems rather wasteful as there are now two models of the scene, one of the scene itself and one within the actor. It seems better to allow some perceptual awareness of objects outside the field of view. Chapter 4 discusses simulation of the centre and periphery of vision in more detail.

The next section will summarise the advantages and disadvantages of the two methods. For virtual actors it is easier to develop a vision algorithm than for a robot, as all the information about the world is contained within the computer and as such can be directly accessed. However, there still needs to be some sort of algorithm for interpreting and using this information. It is important that this algorithm produces realistic, human-like behaviour. The computer vision based method will tend to produce complex solutions which are limited by existing vision algorithms. The *ad hoc* approach will produce simpler solutions but there is less guarantee of realism. It would be desirable to find a method of designing simple visual algorithms of the kind produced by *ad hoc* methods but which have a sounder basis that is more likely to produce realism.

3.2 Psychological Approach

It is possible to design vision algorithms which have the simplicity of *ad hoc* methods combined with the closeness to human vision which computer vision methods are claimed to have (though often do not). This can be done by using results from psychology and psychophysics as a starting point or inspiration for designing algorithms. Visual psychologists have extensively studied human visual perception and its relationship to action. These theories can often be expressed very simply, abstracting away low-level details such as the retinal image or neural mechanisms. Also vision has been studied in conjunction with action to show how vision can allow some behaviour to be performed. The approach used in this work will be to use psychological theories as the basis of visual algorithms. These algorithms will not be a general-purpose visual system but will be closely tied to a particular action. This means that much of the general detail can be abstracted away and only the features that are necessary for a particular task need be implemented. It also means that the visual algorithms will always be aimed at simulating behaviour. When the details have been abstracted away the psychological theories are often simple and so can be implemented efficiently, assuming a virtual environment where extracting the necessary information merely consists of querying the graphical database. For example, section 3.5.1 describes a method which uses information (about the time to contact) which would be difficult and computationally expensive to extract in a robotic system but which is simple to calculate in a virtual environment.

Normally, vision will be simulated at a granularity of physical objects rather than lower level features such as colours or surfaces. This means that most unnecessary information can be abstracted away. The important features such as position and speed tend to be defined for whole objects. It has been shown[Pas98] that people's attention tend to operate on the granularity of objects so they are a suitable basis for the simulation of vision.

3.3 Some Psychology

This approach to the simulation of vision requires knowledge of current psychological theories[1]. It in fact places a number of constraints on the theories and approaches that are useful.[2] First the theories must be formulated at a high level. They must abstract away details of the underlying physical mechanisms of vision. They must deal with large scale vision competence. I mean by this that they must be posed in term of the perception of the sort of high level structures that are dealt with by the internal representations of the program e.g. objects, velocities and people, not retinal images or light intensities. The second constraint is that the theories must deal with vision as it is involved with active tasks. As the simulation of vision is being used to simulate behaviour and so produce action and motion, it is no use looking at a theory that deals only with the static perception of object and not with how perception is used to guide action. Nor is it useful to look at results which only deal with perception of a static scene where the observer does not move, as the actor's motion is integral to the sort of perceptual modelling that is used here. The final constraint is that it must be possible to produce a real-time computationally tractable algorithm from the theory. This leaves out theories that are too vague and general in their formulation and theories which rely too much on computationally expensive low-level details.

These constraints exclude a number of approaches. In particular it excludes physiological approaches to the study of perception. These largely study the structure of the eye and the visual cortex. These studies are generally too low level, producing theories which do not deal with the appropriate level of action. It also excludes other approaches which focus on lower level aspects of vision, such as much of Marr's work[Mar82]. Many experiments rely on techniques which will tend not to produce useful results for animation. The next few paragraphs consider experiments in visual psychology and discusses which ones can be useful for the purpose of this work.

The earliest empirical technique in psychology was *introspection*. This was used by the *structuralist* school. They believed that sensory perception could be decomposed into elementary "sensations". They trained special observers to attempt to decompose the perceptions they themselves were experiencing. This program ran into problems and was

[1]In writing this section I have been helped very much by two excellent text books by Bruce, Green and Georgeson[BGG96] and by Gordon[Gor89]

[2]This section makes various judgements about the usefulness for my research of various psychological approaches. I do not wish to imply from this a general judgement of any of the approaches. My knowledge of psychology is insufficient to judge which will be the most fruitful for the understanding of human vision. I reject certain results because they do not help me in my results not because they are in any way invalid.

heavily criticised by the *behaviourists*. The main problems were that observers tended to disagree about their sensations, and that their observations were influenced by their training and what they were expecting to find. Another problem was that many mental processes are unavailable for introspection. The behaviourists wanted to find a more objective basis for evidence in psychology. They focused on the external behaviour of people and animals believing that it was not valid to speculate about the internal processes of the mind. This resulted in a more laboratory-based experimental methodology. Their experiments, mostly on animals, would give a certain *stimulus* to a subject and then record the resulting behaviour.

Though the extreme idea that internal mental processes should not be considered in psychology has been largely abandoned, the central ideas of the behaviourists' experimental methods remain. Laboratory-based experiments in visual psychology developed into *psychophysics*. This sort of experiment was based on giving the subject a sequence of controlled stimuli and asking them to make a response from a limited set of options. Psychophysics is closely linked to the idea of a *threshold*, the minimal perceptible difference between two stimuli. Thus an experiment might consist of showing the subject two light sources and asking if one appeared brighter. There are three main classes of technique within psychophysics. *Minimal differences* consists of taking two stimuli, a control which is not varied, and one which starts off perceptibly different (e.g. brighter), this is then reduced (or increased) until the difference between the two stimuli is imperceptible, this point is noted. The stimulus is then reduced further until there is a perceptible difference in the other direction (darker) and this point is also noted. *Constant stimuli* consists of showing a sequence of pairs of stimuli, one is held constant and the other is varied. The subject must make a decision at each stage of which is greater. *Adjustment* is more task based. The subject is shown two stimuli and has some method for altering the value of one of them. He or she must adjust the stimulus until they both appear identical.

These experiments tend to simplify the problem of understanding human vision by choosing a single type of stimulus and applying it in a controlled laboratory environment. This makes it possible to design feasible experiments but makes their results rather unsuitable for my research. They tend to be very far from real behaviour in a real environment. In general subjects are passively presented with stimuli and so experiments tend not to take into account of motion and action which are central to the needs of this research. Also the stimuli are very simple, which does not reflect the real environment, or the complex virtual environments which actors inhabit. A school called the *ecological* psychologists, pioneered by J.J.Gibson[Gib79], criticise traditional experimental methods for precisely these reasons. Gibson notes that many results are due to the impoverished visual conditions of the experiments and have little to do with everyday vision. Often results only occur when subjects are unmoving and often only when looking with one eye but, to ecological psychology, movement is inseparable from perception. Also Gibson stresses that experiments should not be performed in synthetic environments against blank walls but in real environments using a normally textured ground as the background. Unfortunately, this sort of experiment is more difficult to design and get results from, but when they work they give an idea about perception in the sort of environments that it would be desirable to simulate for computer animation. Psychologists in this and other traditions have also made experiments with vision as we move particularly during walking or running. Patla, Prentice, Robinson and Neufeld[PPRN91] varied the timing of cues to either change direction or step over an obstacle, gaining information about when these actions where planned. Patla and Vickers[PV97] used eye tracking to find where people looked as they stepped over obstacles, though this is a rather difficult method to get right. Cutting, Vishton and Braren[CVB95] used computer displays of simple virtual environments to simulate the person moving in a controlled manner. They used this to investigate how people navigated environments. Experiments of this sort, involving people moving, are difficult to carry out but produce results that are very useful in my research. In fact the results of all three experiments are used in section 3.5.1 and chapter 6.

Lee[Lee80] stresses that perception as used for guiding action is not merely spatial but that the information is a combination of the spatial and temporal properties of the object measured with reference to properties of the viewer. For example, when judging when to jump while running the hurdles it is not necessary to judge the distance of the hurdle in any absolute measure. It is sufficient to know the time before the runner will reach the hurdle measured in terms of the time it takes the runner to take a step. This is a complex measure which involves the distance of the runner to the hurdle, their relative velocity, the length of the runner's stride and the temporal length of the runner's step. However, Lee argues that it is easier for the runner to judge this quantity directly than to judge each component and combine them. This will be discussed further in section 3.5.1.

3.4 Vision Implementation

This section describes the mechanisms by which actors gain information about objects in the environment. These are not the visual algorithms themselves but the ways in which the visual algorithms obtain the information they need. These mechanisms are not aimed at simulating the image on the retina of the actor. Nor are they aimed at allowing the actor to perceive features such as colour and shape. These are low level attributes of an object that are not directly helpful in producing behaviour. The aim of the visual simulation is to aid the simulation of behaviour, which it does by simulating the visual functions that directly inform behaviour. This means that vision is not a single general system but a group of algorithms that are tightly coupled to the algorithms producing behaviour. These algorithms work by obtaining various properties of an object from the object. These can then be used directly in the algorithm or can be used to calculate more complex properties. There are two types of feature that have to be distinguished. Many algorithms work on simple geometric properties or other properties that can be easily defined mathematically for all objects, for example, velocity and position. These will be referred to as *basic properties* of an object, and they will be described in section 3.4.2. However, sometimes higher level, more abstract properties are needed, for example, beauty or the property of being similar to a cup. This sort of property cannot be handled using a simple mathematical definition and cannot be manipulated by some standard algorithm. Also these properties are often needed in circumstances where it is undesirable to "hard-code" the property into an algorithm. Rather it is desirable for the animator to add properties as necessary. This sort of property is here handled using a mechanism called *object features*, described in section 3.4.3.

The rest of this section describes the details of the underlying vision implementation, the methods through which the actor obtains information about objects. This consists of the actor having access to various properties of the objects. The next section describes how the objects themselves are represented and how they are stored so as to reduce the number of objects that have to be considered by the various other parts of the vision system. Sections 3.4.2 and 3.4.3 describe how the two types of properties, basic properties and object features are represented and how the actor has access to them. Finally section 3.4.4 describes how objects that are occluded by other objects are handled, an important low-level part of the vision system.

3.4.1 Physical objects

All objects have a standard representation as described in appendix A.2. This enables actors to use objects to extract information without knowing what type of object it is. If it is known that an object is of a more complex type, for example, it could be another actor, more information can be extracted.

As there can be a large number of objects in a scene it is desirable to limit the number of objects each actor has to test for a given algorithm. This is done by maintaining a number of different lists of objects, as described in appendix A.2. Each corresponds to objects having a different property, for example there is a list of moving objects and a list of objects that are actors. The number of objects tested is further reduced by the actor's attention mechanism (see chapter 4), the actor will only test objects it is aware of. Some tests need only be performed on the object currently being attended to.

3.4.2 Basic properties

Each object is guaranteed to have a number of properties defined for it. These include, position, orientation, velocity and a bounding box. The actor can use these properties for the visual algorithms described in this dissertation. They can be used directly or to calculate more complex properties such as Time-To-Contact (see section 3.5.1).

These properties are defined as *Comms* which provide a simple and efficient way of obtain and using a property (Comms are described in appendix A.1). These allow the actor to obtain a property of an object without knowing the details of the object. Also as an *agent* within an actor can request to be notified when a Comm changes; the actor can react to changes in the properties of an object.

3.4.3 Object features

Not all visual information can be expressed as simple geometric properties. For example, the reasons some one might look at an object are often complex and rather abstract, for example it is "beautiful", "interesting" or it "looks like Aunt Jemima". These concepts have little to do with properties such as position, velocity or even colour. They require a rather more abstract representation.

This is done using *Object Features.* This is a very simple mechanism, an object feature merely consists of a name and a value between 0 and 1. Each object has a variable number of features. The values of these features can be edited by the user and new features can be added. The system itself does not attempt to give any meaning to each feature, this is entirely done by the user. In this way as little of the possible functionality is "hard coded" into the system and so it is hoped that there will be as much flexibility as possible for the user at the cost of slightly more work. However, it is better to leave the work to the user than to have an inflexible system. The user can also set corresponding feature values in an actor which specify how the actor reacts to a feature. For example, if an object with a feature "floral" with a value of 0.8 and the looking agent of an actor has a corresponding feature value of 0.4 then the actor will look at the object with a probability of 0.32. If more than one feature is relevant a mean is taken. Different actors can have different feature values and so react differently to different objects.

In order to use a feature the actor can request its value by name from an object. The object will either return the feature value or state that the feature is not present. The user can also edit the features of an object in such a way that it only affects one particular actor. This is done by each actor storing its own a list of substitute features which stand for the features of various objects. If the actor needs the features of a particular object and it has a substitute for it, the actor will use the substitute instead. This allows one actor to see a different set of features, and feature values, from another. This mean that the actors will have a different reaction to an object. This is important, as it is useful to have a general feature value representing the reaction of most actors and a default value. However, it is important if each actor is to have its own personality that some actors should behave in different ways with respect to certain objects. For example, various objects might be defined as glasses and actors will drink out of objects with a feature glass and not out of ones without. However, a character that the animator wants portray as a heavy drinker might have a gin bottle defined as a glass while most characters would not drink out of it. Conversely a child character might have any glass containing an alcoholic drink not defined as a glass so it will not drink out of it.

3.4.4 Occlusion Testing

One advantage that is often mentioned of computer vision type techniques is that they provide occlusion testing of objects for free. Objects that are occluded from the actor will not appear in the resulting image and so will not be considered in the algorithm. When fully examined, this argument is not as persuasive as it seems. Occlusion can be of many forms. An object that is at rest behind a wall is clearly occluded from the actor and the actor will not be aware of it or be able to make visual judgements about it. However, if the object has merely passed behind another object, such as a tree, as it or the actor moves, it is not occluded in the same way. The actor can still make visual judgements about it, based on its previous position and velocity: people do this all the time. To exclude this

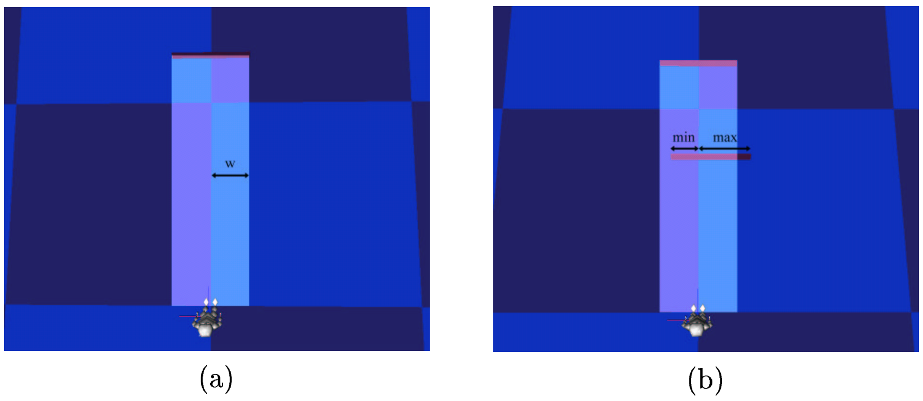
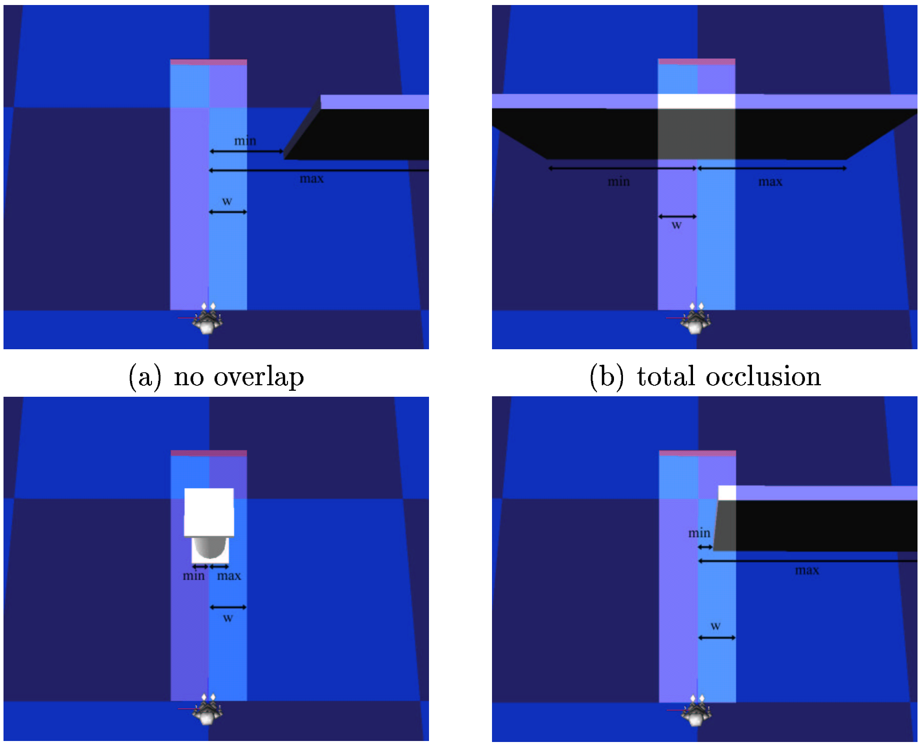


Figure 3.2: Occluding and occluded objects. (a)The occluding region of an object. (b) The minimum and maximum distances of points on the potentially occluding from the centre line between the actor and the potentially occluded object. These are used in the occlusion test.

sort of object from visual algorithms would produce very strange results. In order to deal with this sort of situation in a computer vision type framework, the actor would have to build up some sort of model of objects in the environment which are not visible but are remembered. This amounts to building up a representation of the environment for each actor as well as the representation of the environment itself. This is clearly wasteful. It is better to allow the actor to "perceive" the object while it temporarily goes out of sight. This means that it is desirable to test for occlusion only occasionally and so it does not matter so much if the test is slightly expensive. The occasions where it is necessary to test are: (a) when the actor first tries to look at the object, if the object cannot be seen the actor cannot look at it; (b) when the object undergoes a major change such as a large change of velocity, if the actor is tracking an object based on its velocity while it is behind an occluding obstacle and the object changes direction, the actor will track it incorrectly; (c) if the actor is tracking an object for a long period, it is worth checking occasionally that it has not passed behind an object in which case the actor should stop tracking it.

As well reducing the number of instances when occlusion has to be tested for, it is also possible to reduce the number of objects that occlusion is tested against. In general, most objects in an environment are too small to occlude other objects apart from over a very small viewing angle, thus moving only slightly will bring it into view. Also normally an actor has a vague knowledge of the positions of objects that it has to deal with so this sort of minor occlusion will not be a hindrance. For example, a chair will not hide an object effectively from a moving character. In particular, if the actor has to perform an action on the object behind the chair the actor will probably know roughly where it is and so move its head to see it. Also, it will have to move to the other side of the chair to perform an action anyway. If the actor has to perform some sort of navigation task such as avoiding a collision with the object behind the chair it will have to deal with



(a) no overlap (b) total occlusion

(c) occluder too small (d) partial overlap

Figure 3.3: The four possible configurations of an occluding and occluded object

the chair first. Thus most objects can be excluded when testing for occlusion. A special flag is set in certain objects, such as walls, showing that they are able to significantly occlude other objects. Thus when an object is tested to see if it is occluded it is only tested against these objects. These approximations are valid because we are simulating extremely complex human behaviour and human perception and memory allow a much greater awareness of the environment than a simple occlusion test allows. The methods are implemented as follows.

Visibility is tested when the actor first looks at an object (see chapter 4 for a discussion of the mechanisms of looking at objects). If the object is not visible the actor will not look at it (this in turn will mean that the actor might not know about the object). Visibility is also tested if there is a large change in the velocity of the object. If the object changes velocity abruptly while hidden, the actor will follow a false object with the original velocity of the object, this simulates the assumption some one would make while attempting to track and object that is temporarily occluded. Finally, the actor tests visibility at regular intervals while the object is being fixated and breaks off if the object is out of sight twice in a row (i.e. for a significant time). These tests are described in more detail in chapter 4.

The test itself iterates over a list of all the occluding objects. Each occluding object is tested to see if it intersects the region between the actor and the object being tested defined by the width of the object being tested, as shown in figure 3.2(a). A rectangular region is used rather than a triangular region as it is not so dependent on a static position for the actor. If a triangular region with the apex at the actor were used the result of the test might change with a slight movement of the actor. This is undesirable as what is being tested for is not instantaneous occlusion but a situation where the actor is unable to see the object for a long time and with a fairly wide range of head and body movement. A rectangular region gives a higher probability of being able to see large obvious objects where as small objects that it is likely that the actor would notice are occluded more easily. The height of the occluding object is also tested: if it is shorter than the object being tested it cannot occlude the object. Next the distances of the extreme points of the occluding object from the line between the actor and the occluded object are found, as shown in figure 3.2(b). These are tested against the width of the region between the actor and the occluded object. Figure 3.3 shows four possible configurations of the occluded and occluding objects. (a) can be trivially rejected as the occluding object does not overlap the region. (b) is within the region and min and max are larger than the width and on opposite sides of the centre line and so the object is occluded. (c) is within the region but as the occluding object is small min and max are less than the width of the region. (d) d is large enough but as it is off centre max is larger than w but min is smaller. The first two cases are tested for as follows:

$$(a)\ (min < -w \wedge max < -w) \vee (min > w \wedge max > w)$$
$$(b)\ min < -w \wedge max > w$$

If neither test is passed then one of the other two situations is true. In these cases the object is not entirely occluded but they cannot be rejected entirely as the occluder might occlude the object in combination with another occluding object as shown in figure 3.4. Thus the max and min distances are added to a list of partial occlusions. These represent the span for which the object is occluded. When a new item is added to the list it is checked against all the existing items. If the occluded spans of the new item and one of the old items overlap, i.e.

$$newMin < oldMax \vee newMax > oldMin$$

they are merged into a new item with the total span. If any of these merged spans is greater than then width of the occlusion region the object being tested is classed as occluded. Generally, as the number of occluding objects between the actor and the tested object will be small, the list of spans will not be very long and so testing each new span against it will not be very expensive.

3.5 Vision examples

The next two sections describe two examples of how psychology can be used to design algorithms for behavioural simulation. The first is a method for detecting collisions that

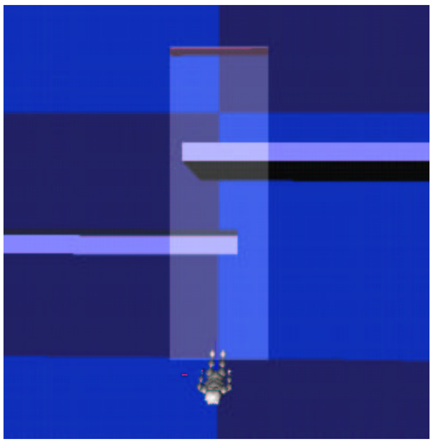


Figure 3.4: Two objects combining to occlude another object

is used in the later chapters. The current discussion will only describe the method, chapters 5 and 6 give a full description of its use.

The second describes a more specialised and self contained application, catching long balls in cricket or baseball. This is a more stand alone example and illustrates certain problems with using psychological theories.

3.5.1 Collision Detection

The first example is of how an actor can detect when it is about to collide with an obstacle. This is often called *collision detection*, the most natural name for the task, unfortunately it is slightly confusing as "collision detection" can also mean the low-level animation task of determining when two primitives actually do collide. Here, it will refer to the actor detecting when there is about to be a collision in order to take appropriate action. The task is relatively simple if all obstacles are stationary but becomes more difficult if moving objects are present. Chapter 6 gives an overview of how actors can avoid collisions. Detecting collisions can be useful for more than just navigation, however. Chapter 5 describes how the collision detection test can be used to perform actions on objects that approach the actor.

Differential parallactic displacements

Detecting whether an actor will collide with an object is itself a difficult problem. In the case that the actor is moving in a roughly straight path and the object is not moving the test is trivial, they will collide if the object is along the path of the actor. More exactly the test is if the object intersects the rectangle representing the height and width of the actor, swept along the line of forward movement of the actor. However, if the obstacles are moving the problem becomes more difficult. The obstacle no longer needs to be in front of the actor for there to be a collision, there is now the possibility of side on collisions and collisions from behind. This means that tests based purely on the direction of motion of the actor are no longer effective; the object's motion has to be taken into account. it would seem that in order to detect collisions it would be necessary to perform an intersection test on the movement paths of the actor and the object. This would be a complicated test as the paths are not necessarily straight lines, they could be general curves. In fact it would often be impossible as the paths are not always known in advance and could be discontinuous in velocity. Clearly it is not possible to find an algorithm that can make a correct collision judgement in every case. What approximate algorithm should we choose? The answer is *not* the one which makes the most accurate judgement in any given situation but the one that makes the same judgement that a real person would make in the same circumstance. The best starting point for finding such an algorithm is to attempt to use the same method that people use by basing the algorithm on current psychological theories.

Another important aspect of the collision detection algorithm is the simulation of attention discussed in the next chapter. In order to make correct collision judgements the actor must be aware of the object and in fact, as discussed below, be attending to it to make judgements. This has the advantage that the test only has to be applied to one object at a time allowing us to use a test that might be overly expensive if applied to all objects in the scene.

The test for potential collision is based on work in psychology by Cutting, Vishton and Braren[CVB95]. They were investigating precisely the problem of how people avoid collisions with obstacles in their environment. Their method was to show subjects displays which simulated the viewer moving around an environment. The displays manipulated the motion of objects in the foreground and background. Subjects were asked if they thought they would collide with a particular object. Their results support a potentially computationally expensive test which is used by humans but which can be reduced to a test which I have implemented very simply. By way of justification of the method and for the interest of the reader I shall briefly outline the complex method and then describe my own implementation. Cutting *et al*'s theory is based on the idea that, while people fixate an object, they monitor the motion of other objects in the periphery of vision. The test is based on the retinal motion of the objects, i.e. the motion of the objects on the person's retina. This is not merely the actual motion of the object, it is also affected by the person's motion and the rotation of the person's head. The motion of objects nearer than the fixated object is compared with those further away. The fixated object itself will not be moving as the person's eyes follow it. They call the relative motion of the object *differential parallactic displacements*. There are three possible cases:

1. the images on the retina of objects nearer to the observer than the fixated object move faster and in the opposite direction to those very far away. In this case the observer passes in front of the moving object.

2. the images of both near and far objects move in roughly the same way. In this case the observer passes behind the object.

3. the images of near objects move faster but in the same direction as far objects. In this case there will be a collision.

This test is effective and there is evidence that it is the test actually used by humans but it could be expensive to implement as it requires monitoring the motion of many objects in the foreground and background. Luckily it is identical to a far simpler test. Cutting *et al.* describe the conditions for a collision as seen by a walker:

> If four conditions are met a collision will occur: (a) if both observer and moving object maintain constant velocity, (b) if both are on linear paths, (c) if they maintain a constant gaze-movement angle between them, and (d) if the retinal size of the object increases for the observer.[CVB95, p636]

The first two conditions appear to be necessary for the detection of collisions. Lee[Lee80] has shown that people have trouble making judgements in the presence of non-constant velocities. (d) will be discussed in section 3.5.1. It is condition (c) that is checked by differential parallactic displacements. Figure 3.5 shows the three cases that can occur. The top row shows that if the angle between the actor's heading and the object remain constant there will be a collision. If the angle is constant the actor's head will not rotate while fixating the object and so movement of static objects in the environment will be purely due to translation. The speed of this motion is the reciprocal of the distance of the object and so far objects will move slower and in the same direction as near ones as required in case 3 above. If the angle decreases as in the middle row the object will pass in front of the actor. In this case the actor's head will rotate in the opposite direction to the actor's linear translation. Near objects will be more affected by the rotation and so will move faster but in the opposite direction to objects very far away which will be more affected by the translation. This is the condition described in case 1. Finally if the angle increases the object will pass behind the actor. In this case the actor's head rotation and translation are in roughly the same direction and so will produce movement of near objects and far objects in the same direction, thus making movement in the visual field roughly constant as required by case 2.

Now we have a very simple test for collision: when the actor first fixates an object the angle of that object to the actor's heading is taken and then compared to the angle at the time when the actor stops fixating it. If the angles are roughly the same a collision is detected. This is a very efficient test requiring only measurement of two angles every fixation period (which lasts on average about 15 frames).[3]

Unfortunately this test is only correct for single point collisions. This works well if objects are small as the angle of object will be roughly constant across its length. However, for large object such as a wall another test is needed. This is because the centre of the object can have a non-constant angle as the actor is not on a collision course with it but

[3]Actually this is a bit of a shortcut, the angle should be monitored throughout the fixation period. Monitoring only at the beginning and end introduces the possibility that the angle is not constant but happens to have the same value at the beginning and end. However, this is a very unlikely situation which would only happen in contrived circumstances.

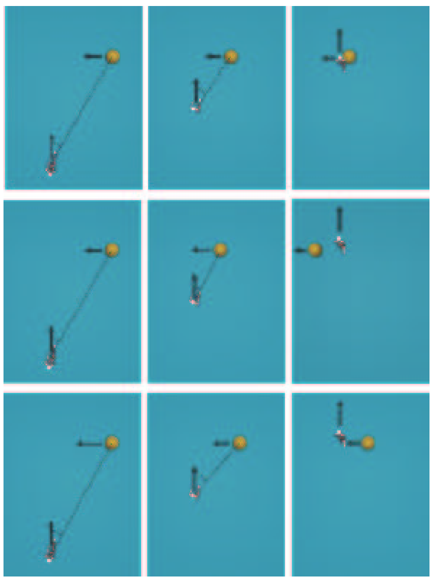


Figure 3.5: The three possible cases that can occur with the rate of change of angle of an object relative to the actor. In the top row the angle remains constant and the actor collides with the object. In the middle row the angle decreases and the object passes in front of the actor. In the bottom row the angle increases and the object passes behind the actor.

the actor still collides with a different part of the object. It is not desirable to subdivide the object and test multiple sections along its way as this would require too many tests. There are two solutions both of which seem to work equally well.

The first is to take the differences of the two angles at the two extreme points of the object. If they have different signs there must be a point along the length of the object where the angle change is zero and so there will be a collision. This is equivalent to using the same test along the length of the object.

The second test is based on a different but related theory of how collisions are detected proposed by Peper, Bootsma, Mestre and Bakker[PBMB94]. This proposes that a collision is detected if the rate of change of the angle of the object (as described above) divided by the rate of change of the angle subtended by the object is between $1/2$ and $-1/2$. The subtended angle can be approximated as the difference in angle between the extreme points of the object. It is not clear which of these two methods is closer to the method used by real people but there seems to be little difference in their effectiveness or computational efficiency.

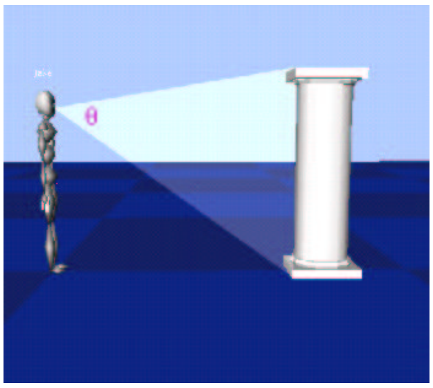


Figure 3.6: The visual angle of an object θ

Time-To-Contact

In the last section a test for potential collisions with an object was described. The test is valid, but only if the "retinal image" of the object is increasing in size. Also it does not provide information as to when the collision will be. It is necessary to know the time of a collision not only to know when to react to it but also how to react. A collision that will happen straight away needs a different reaction to one where the actor has a long time to react. Luckily both the timing of the collision and the expansion or contraction of the retinal image are given by the same test.

Lee [Lee80, WYL86] suggest that people can naturally perceive an approximation to the time before the person will collide with an object. This is given by the *Time-To-Contact (TTC)* measure, which is:

$$TTC = D/V$$

where D is the distance between the observer and the object and V is their relative speed. This is the exact time to the collision if both the person and the obstacle are moving with constant velocity. However, it is believed to be used by people to make judgement even when there is acceleration. People are known to have more difficulty making judgements based on acceleration than on velocity. There is a variety of experimental evidence that TTC is used to judge various actions in animals and humans[4]. Wagner[Wag82] analysed films of flies landing and showed that they were consistent with the flies using TTC to judge the landing. Lee and Reddish[LR81] show that the way gannets fold their wings while plunge diving could be predicted by them using TTC. This result is particularly interesting as TTC still seems to be used even if there is significant acceleration in which case TTC, a measure based only on velocity, is no longer accurate. Warren, Young and Lee[WYL86] showed that people adjust their steps based on timing and TTC could be a

[4]The evidence can be slightly inconclusive as the fact that an external action is consistent with the TTC hypothesis does not prove that TTC is used by the brain, but as animation deals with external behaviour they seem good enough.

measure used to judge the timing. Finally, Lee[Lee76] has shown that motorists use TTC when controlling how they brake.

Clearly TTC measure gives us the time before the collision but it is also directly connected with the expansion of the retinal image of the object. Lee proposes TTC as an alternative to making judgements based separately on distance and velocity, both of which are difficult to extract from the visual system, but for this to be true there must be a way of calculating it without calculating distance and velocity. Lee shows that TTC is equivalent to a quantity called visual τ which is based on the visual angle subtended by the object (see figure 3.6). Tresilian[Tre91] defines it as[5]:

$$\tau = -\frac{\theta}{\frac{d\theta}{dt}} \approx TTC$$

Now this gives information about the dilation of the retinal image $\frac{d\theta}{dt}$. If the τ is negative the retinal image is contracting. As τ is equal to TTC, negative Time-To-Contact indicates a receding retinal image. This is to be expected as a negative TTC indicates a collision that occurred in the past, i.e. the object is moving away from the actor. Clearly there is no need to calculate τ itself in a computer environment where distances and velocities are readily available, it is enough to calculate it directly (which just requires a floating point division).

TTC can have many other uses. Lee[Lee80] also discusses other related measures. As well as judging actions based on spatio-temporal properties, it is often important to judge distances with reference to the size of parts of one's body. For example, when stepping over an obstacle it is important to know how high it is compared with the height of one's leg. Lee calls this sort of measure *body-scaled measures.* He shows how, at least in theory, these measures are directly available to the visual system while their components are not. He shows how time to contact can be used to obtain the height and distance of an object scaled relative to eye height and distance relative to stride length. Height scaled by eye height can be used to judge stepping over an obstacle (assuming that leg length can be thought of as a fixed proportion of eye height). Distance scaled by stride length can be used to judge timing for such actions as hurdling or stepping over a hole since it is much more important to know which footstep to take action in than it is to know the absolute time.

Application of collision detection

The algorithm described here illustrates one example of how results from psychology research can be used to inform the implementation of behavioural simulations. This algorithm is used in chapters 5 and 6 for two different tasks. Chapter 6 describes how an actor navigating the environment can use the algorithm to avoid collisions with objects.

[5]Actually this is only one way of defining τ and it is different from Lee's original formulation[Lee80]. Different methods are thought to be used in different circumstances, see Tresilian[Tre91] and Warren[War95] for a full discussion.

Chapter 5 gives a more general application for tasks where the actor has to react to objects approaching it. Examples and evaluation of its use can be found in those chapters.

3.5.2 Ball Catching

The second example of the use of psychology in the simulation of visually guided behaviour is the simulation of how a fielder in cricket or baseball runs to catch a long ball. This is a difficult spatial activity that people can learn to do with ease but it is not clear what strategy people use to perform it. In particular, it involves both a prediction of the position at which the ball will land and also of the time at which the ball will get there. In fact, it has been shown that fielders trying to catch nearer balls do not run at the same speed as for further balls and then stop to wait at its landing point, but tend to run at the correct speed so as to arrive at the right place at the right time to catch the ball. This implies that they use a strategy which combines temporal and spatial elements.

Most psychologists who have studied ball catching do not believe that fielders predict the path of the ball and run directly to the right place, rather they run based on instantaneous visual cues and constantly update their velocity so as to arrive at the right place at the right time without actually knowing where it will land. This theory claims that there is a direct connection between the fielder's image of the ball and the fielder's motion which results in the correct path. The two schemes implemented fall into this category. Chapman [Cha68] produced an early theory based on keeping the rate of increase of the angle of elevation of the ball from the fielder constant (ψ in Figure 3.7). There have been various variations on this idea [Tre95, DM93, MD96]. Todd [Tod81] gives an overview of the visual information available to a fielder about the path of the ball. He also suggests and tests a number of strategies, one of which he found to be compatible with the empirical data. This strategy only works when the fielder is in the plane of motion of the ball, however. It is also unclear whether all the required information needed is available to the fielder. Tresilian [Tre90] has investigated methods by which fielders can time their catches correctly, particularly for grasping the ball. He says nothing about how to get into the right position to catch the ball, so his approach is unsuitable for my simulation. Finally, McBeath, Shaffer and Kaiser [MSK95] suggest a new strategy called the Linear Optical Trajectory (LOT).

Implementation

The work described was initially implemented purely as a simulation without animation. This implementation dealt only with the position, velocity and acceleration of the fielder and the ball. The position was updated based on the velocity without attempting to model the walking motion of the fielder. The work was later implemented as part of the *Gepetto* system and the motion of the fielder was represented as the actor walking. This was done using the curve following walk described in section 6.5.1. The algorithms described below were used to generate velocities which were then used to adjust the walking speed of the actor and generate control points of a curve for the actor to walk along. One drawback of this method is that, as described in section 6.2.1, the present walk generator only handles

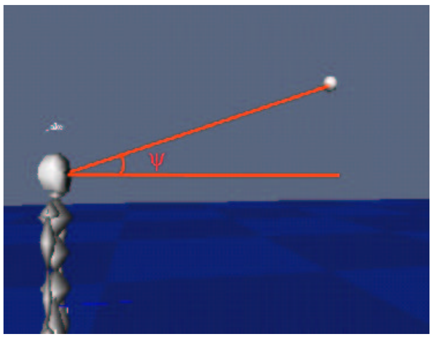


Figure 3.7: The angle of elevation of the ball, ψ

walking, not running gaits. This means that the actors motion tends to lack realism at high speeds where it would be more natural to run.

The ball is represented as a Geppetto physical object. It has a standard projectile motion given by the equations:

$$\begin{aligned} x_b &= Vt\cos\theta_1\cos\theta_2 \\ y_b &= Vt\sin\theta_1 - \tfrac{1}{2}gt^2 \\ z_b &= Vt\cos\theta_1\sin\theta_2 \end{aligned}$$

where V is the velocity at which the ball is launched, θ_1 is the vertical angle at which it is launched and θ_2 is the horizontal angle. Experiments were also done with drag present. The drag was assumed to be proportional to the square of the velocity (the drag on a cricket ball can depend on many factors as described by Mestre[Mes90], however, for the present purposes it is enough to show that the method works with a non-constant acceleration). Thus the ball is given a further acceleration given by:

$$\mathbf{a} = -\rho V^2 \hat{\mathbf{V}}$$

where ρ is a general drag constant that represents a large number of different factors (as described in Mestre[Mes90]).

Chapman's Strategy and Variants

In his 1968 paper Chapman [Cha68] suggests a strategy for running to catch a ball based on the angle of elevation of the ball from the fielder (ψ in Figure 3.7).

ψ is proportional to the optical projection of the height of the ball so this entire analysis can equivalently be applied to the height of the image of the ball at the fielder's eye. Chapman assumes that the fielder is standing in the plane of motion of the ball and shows algebraically that, if the fielder is standing at the point at which the ball will land, $\tan(\psi)$ will be:

$$\tan\psi = \frac{gt}{2V cos(\theta)}$$

where V is the speed of projection of the ball, θ is the angle of projection of the ball to the horizontal and g is the acceleration due to gravity. This means that if the fielder is not moving and the first derivative of $\tan\psi$ is:

$$\frac{d\tan\psi}{dt} = \frac{g}{2Vcos(\theta)}$$

then he or she is in the correct position to catch the ball. However, this is only the case in the special situation that the fielder starts off in exactly the right position to catch the ball. Chapman goes on to show that if the fielder runs in the right direction to catch the ball with a speed v such that he or she arrives at the interception point at exactly the right time to catch the ball then $\tan\psi$ will be equal to:

$$\tan\psi = \frac{gt}{2Vcos(\theta) - v}$$

Thus the fielder will catch the ball if he or she runs at the correct speed such that $\tan\psi$ increases at a constant rate given by:

$$\frac{d\tan\psi}{dt} = \frac{g}{2Vcos(\theta) - v}$$

This gives us a feasible strategy which *could* be used by cricketers (or baseballers) to catch balls. However, it does not actually prove that it is the strategy used by real fielders. Dienes and McLeod [DM93, MD96] point out that fielders do not run at a constant velocity as required by Chapman's analysis. They generalise Chapman's findings by showing that the fielder will catch the ball if he or she runs so that:

$$\frac{d\tan\psi}{dt} = c$$

for any constant c. This no longer requires that the fielder runs at constant velocity, nor does it require the fielder to judge exactly the correct rate of increase of $\tan\psi$.

So far, the analyses that have been described assume that the fielder is always in the plane of motion of the ball, and therefore only moving in one dimension. This makes the problem a one dimensional one and thus it is mathematically simpler. Chapman [Cha68] and McBeath, Shaffer and Kaiser [MSK95] point out that this is in fact not only an unusual special case but that it is in fact more difficult for the player to catch the ball, as there is less visual information available. Chapman suggests a generalisation to 2D by which the fielder runs so as to both keep a constant rate of increase of $\tan\psi$ and a constant horizontal bearing for the ball. Tresilian [Tre95], however, found that this strategy results in the fielder moving in the wrong direction. He proposes another augmentation to Chapman's method, in addition to the acceleration produced by the 1D case there is a second acceleration proportional to the rate of change of direction of the ball. Thus if α is the visual angle between the ball and an arbitrary reference direction then the magnitude of the acceleration is:

$$a_2 = c\frac{d\alpha}{dt}$$

This acceleration is at an angle β to the direction of the ball such that

$$\beta = \arctan\left(\frac{a_2}{a_1}\right)$$

where a_1 is the magnitude of the 1D acceleration from Chapman's method and a_2 is the new acceleration.

I have implemented a combination of Dienes and McLoed's [DM93, MD96] and Tresilian's [Tre95] variations on Chapman's strategy. I used the initial value of $d\tan\theta/dt$ as the target value and gave the fielder an acceleration opposing the rate of change of this value:

$$a_1 = -\frac{d\tan\psi}{dt}$$

This acceleration is in the direction of the ball. I also added Tresilian's second acceleration:

$$a_2 = \frac{1}{c}\frac{d\alpha}{dt}$$

Where α is the angle between the fielders local x-axis (an arbitrary direction) and the direction of the ball. c is a scaling constant (values around 100 seemed to work best). The acceleration was at an angle β to the direction of the ball where:

$$\beta = \arctan\left(\frac{a_2}{a_1}\right)$$

This method seems to work well in theory. When tested with pure simulation (without animating a character) the method resulted in catching the ball (i.e. the final position of the fielder coincided with that of the ball). Figure 3.8 shows some curves produced by this method. They show some problems with the method. Firstly, the curve always turns back. This means that the fielder goes forward and then must run backwards in order to catch the ball. This seems unnatural and contrary to how fielders catch. There is also a worse problem. In the figures the fielder's curve and the ball's curve approach each other but do not meet. This is because of a feature not shown in the figures. The strategy seems to work well throughout the flight of the ball with the fielder approaching the ball but at the end, as they get close, the fielder suddenly starts to oscillate wildly moving very rapidly and far from the ball, up to a kilometre. This is clearly undesirable. It is possible that fielders use a different strategy at the end, in particular McLeod and Dienes [MD96] find that their strategy breaks down near the end.

Figure 3.9 show an animation produced with this method. The problems shown in the curves are exentuated. The motion of the actor is planned one foot step at a time (this agrees with such results as Patla, Prentice, Robinson and Neufeld[PPRN91] that indicate

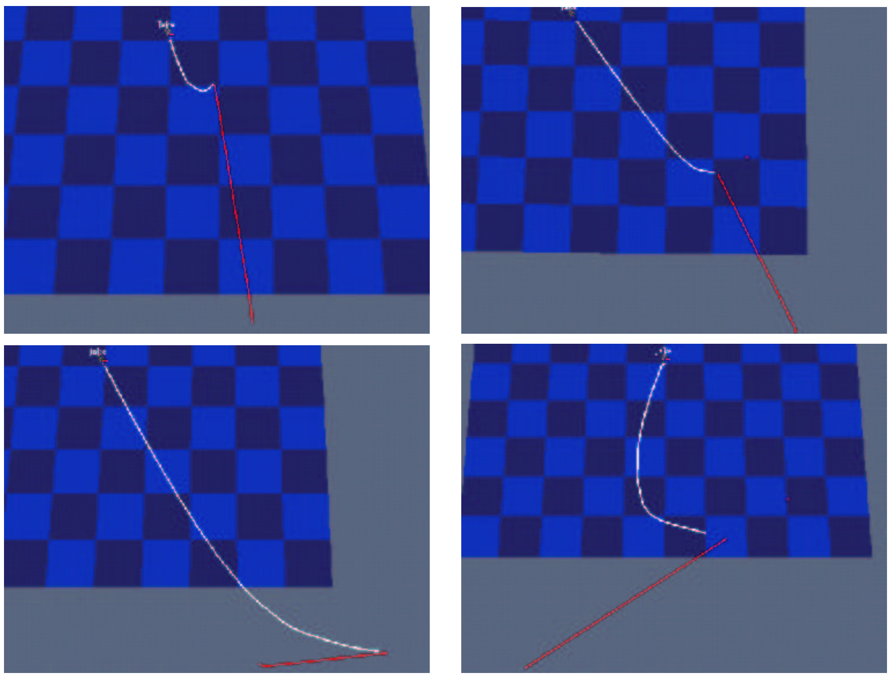


Figure 3.8: Running curves produced using Tresilian's variant of Chapman's method. The white curve is the actor's path and the red curve is the path of the ball.

that people have trouble changing direction in the middle of strides and plan steps two strides ahead). This means that the path is produced in a less continuous way. The turning back is now much more pronounced as it the actor has moved too far forward in the last footstep before turning back and so a sharp correction is needed as shown in the third frame. This means that the actor is not in position where it is able to catch the ball and the path loops back on itself as shown in the fourth frame. The animation involves high speeds and sharp turns that do not animate well. as shown in frames 3 and 4. After the frames shown the turns become too sharp for the walk generator to handle properly, the animation becomes very unrealistic and the actor fails to catch the ball. This is partly because the walk generator does not handle running or sharp turns properly, however, the curve produced by Chapman's method seems to produce too extreme turns even for a real fielder and seems unrealistic.

There is a final problem with this method. Tresilian does not give any justification for introducing the second acceleration a_2. It works but there seems no reason to choose this particular acceleration over another.

These problems mean that this method is unlikely to be useful for animation and so another method must be investigated.

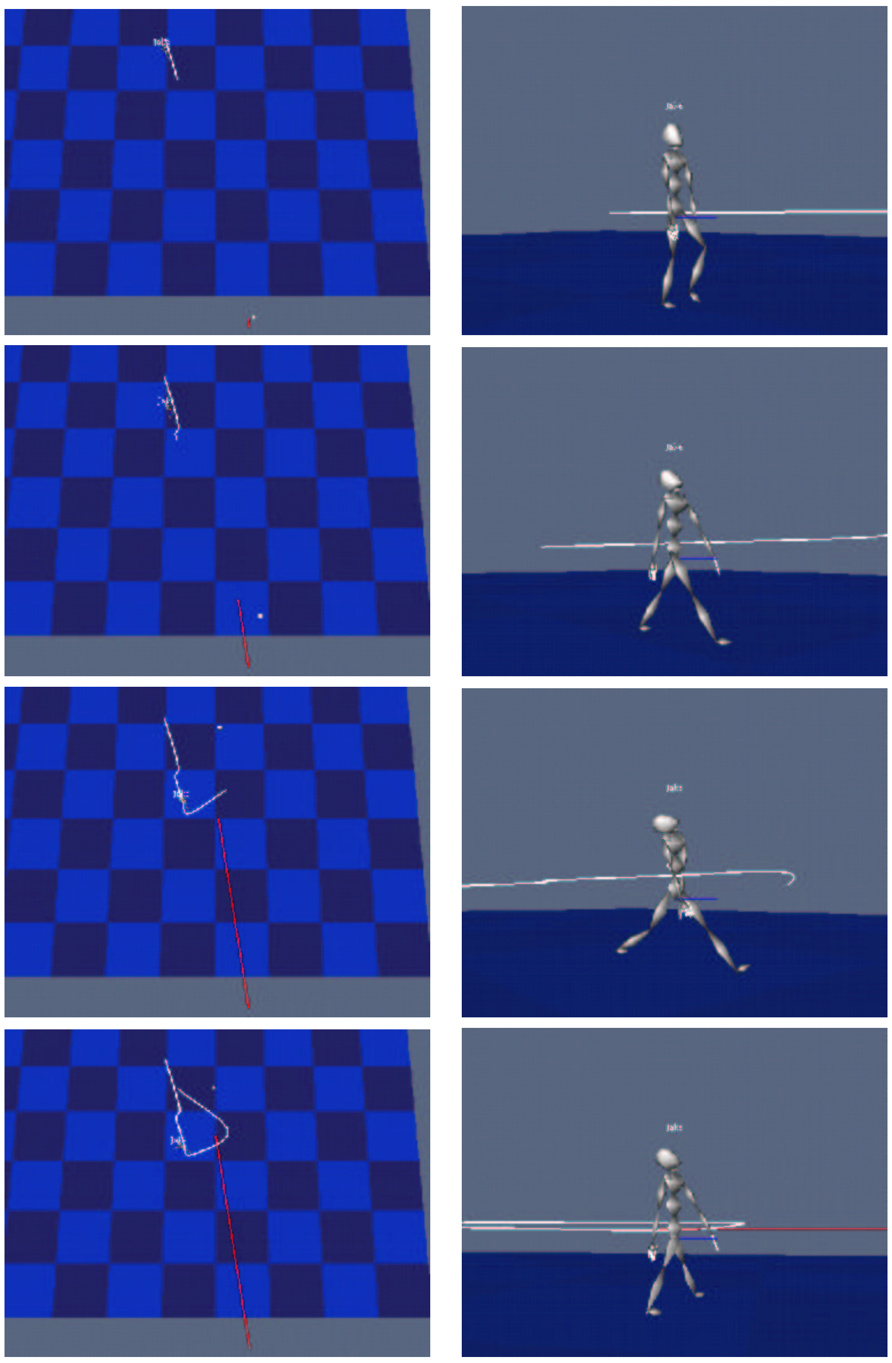


Figure 3.9: Animations produced using Tresilian's variant of Chapman's method, shown from both top and side views. The frames are equally spaced. The method works reasonably at first but then the actor turns back on itself more noticeably than in the pure simulation case (frame 3). Near the end as the ball is very close to the actor the method breaks down entirely(frame 4).(left to right, top to bottom)

The Linear Optical Trajectory Strategy

In their 1995 paper McBeath, Shaffer and Kaiser [MSK95] describe a different method for catching a ball in two dimensions. They point out that catching in two dimensions is easier than in one dimension so it would seem natural that the best way of describing how people catch is to use an innately two dimensional method rather than trying to generalise a one dimensional method.

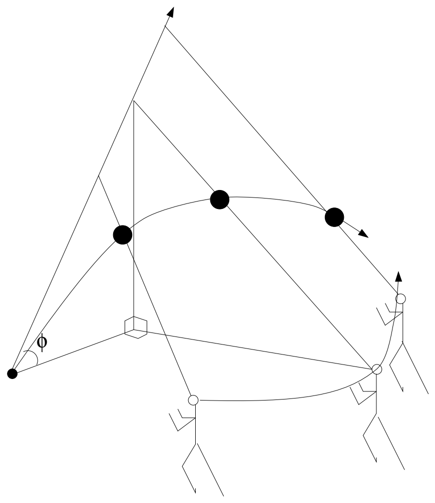


Figure 3.10: The Linear Optical Trajectory Strategy

They describe the Linear Optical Trajectory (LOT) strategy which states that fielders run so that their image of the ball travels upwards in a straight line. Figure 3.10 illustrates this. The theory is that, if the angle ϕ is kept constant, the ball would appear to be moving higher and higher. What is actually happening is that in the second part of the path the ball is moving downwards but, as the fielder is moving towards it, the ball is coming overhead and so is higher in the visual field. The ball can only keep rising like this if the fielder is moving so that he or she is underneath the ball as it completes its fall, i.e. in the correct place to catch a ball.

This theory is supported empirically by an experiment in which amateur baseball fielders wore head mounted cameras and attempted various catches. In most cases the optical path of the ball was roughly linear and so it seems feasible that people use this strategy.

McBeath, Shaffer and Kaiser's paper [MSK95] did not suggest a strategy for maintaining a linear optical trajectory and so it was more of a challenge to implement. Simply having a linear optical path for the ball is not enough. A linear optical path for the ball throughout its flight is not possible unless the fielder is in the plane of motion of the ball, otherwise the path will bend as the ball passes below eye level. If the fielder is in the plane of motion of the ball then any movement by the fielder in the plane of the ball

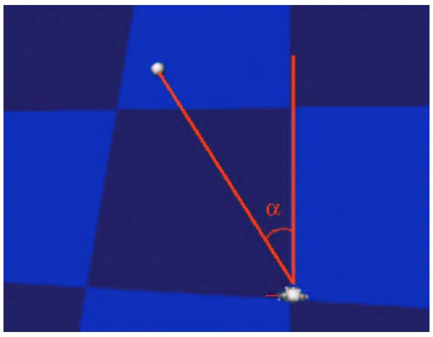


Figure 3.11: The horizontal angle of the ball, α

will result in a linear path. What is needed is an optical image of the ball which moves upwards in a straight line and a fielder motion which results in the fielder being in the right place at the right time to catch the ball. Though McBeath *et al.* suggest this method as an alternative to Chapman's strategy, Dannemiller, Babler and Babler[DBB96] points out that it is in fact equivalent. Keeping a linear optical trajectory means keeping the acceleration of the projection of the ball on the eye zero. Chapman's strategy of keeping the acceleration of $\tan\psi$ zero is equivalent to keeping the vertical component of optical acceleration zero. Thus the linear optical strategy becomes another way of generalising Chapman's strategy to two dimensions. This time, however, the generalisation is based on empirical results. The new method consists of keeping both the of $\tan\psi$ and the derivative of the tangent of the horizontal angle between the ball and some arbitrary reference direction (α in figure 3.11).

Keeping the rates of change of both angles constant is not as simple as it might seem. In particular the obvious strategy of giving the fielder two accelerations, each aimed at keeping one of the derivatives constant, does not work as the two accelerations are not orthogonal and so interfere with each-other. In particular the attempt to have one acceleration parallel to the line between the fielder and the ball and one perpendicular to it fails. Though this would appear to have one acceleration that affects $\tan\psi$ (the parallel) and one that affects $\tan\alpha$ (the perpendicular) and that they are orthogonal this is not in fact true. The reason is that though, at any instant, $\tan\psi$ is measured using the line between the fielder and the ball as both the fielder and ball move this direction changes. This means that, if the direction of the ball relative to the fielder at any given instant is used to adjust $d\tan\psi/dt$, after a while it will have a component perpendicular to the line between the fielder and the ball and so start to affect $d\tan\alpha/dt$.

Finally a simple method was devised. When the ball rises above eye height the values of $d\tan\psi/dt$ and $d\tan\alpha/dt$ are measured and these are taken to be the desired values that are to be maintained, let us call them V_ψ and V_α. The desired values of $\tan\psi$ and $\tan\alpha$ at a time t ahead of the current time are calculated:

$$\tan\psi_d = tan\psi + V_\psi$$
$$\tan\alpha_d = tan\alpha + V_\alpha$$

From these the desired position of the fielder at time t can be calculated. If the position of the ball at time t is (x_b, y_b, z_b) and the desired position of the fielder is $(x_{fd}, 0, z_{fd})$ with the fielder's eye-height being h; the tangents of the two angles will be:

$$\tan\psi_d = \frac{y_b - h}{\sqrt{(x_b - x_{fd})^2 + (z_b - z_{fd})^2}}$$

$$\tan\alpha_d = \frac{x_b - x_{fd}}{z_b - z_{fd}}$$

Substituting and rearranging we get:

$$z_{fd} = z_b - \frac{y_b - h}{\tan\psi_d\sqrt{\tan^2\alpha + 1}}$$

$$x_{fd} = x_b - \frac{\tan\alpha}{z_b - z_{fd}}$$

This gives the desired position of the fielder at a time t from the present time. This can then be used to calculate a velocity that will bring the fielder to that position in time t:

$$\dot{z}_{fd} = \frac{1}{t}\left(z_{bt} - \frac{y_{bt} - h}{\tan\psi d\sqrt{\tan^2\alpha + 1}} - z_f\right)$$
$$\dot{x}_{fd} = \frac{1}{t}\left(x_{bt} - \frac{\tan\alpha}{z_{bt} - z_{fd}} - x_f\right)$$

where $(z_f, 0, x_f)$ is the current position of the fielder and (x_{bt}, y_{bt}, z_{bt}) is the position of the ball at time t. This can be calculated from the current position using the current velocity and acceleration. Using the current velocity and acceleration means that the algorithm can work without knowing the details of the motion of the ball, i.e. how velocity and acceleration change over time. This means that it ought to work with a number of different drag functions. It will not work perfectly as the fact that the acceleration changes over time due to drag means that using the current acceleration will not produce an accurate estimate. However, tests using drag seemed to produce good results.

The LOT method was tested using simulation and animation. Figure 3.12 shows the curves produced. They do not show the problem of the fielder having to backtrack as seen in the previous method. Figures 3.13 and 3.14 show the animations produced by the LOT method. Sometimes the strides seem excessively long, as in figure 3.14 frame 3. This is a problem with the walk generator as noted before, at high velocities the actor should change to a run. The curves are now less smooth than the theoretical curves. This is due to the path being planned discretely at the start of each footstep as already discussed in connection with Chapman's method. This is likely to be slightly more realistic as Patla *et al.*'s results seem to show that people plan footsteps before their start rather than doing many mid-step corrections. Otherwise the results are reasonable, the actor coincides with the ball when it is roughly at eye height. The final frames give us an explanation of what McLeod and Dienes [MD96] note; that their strategy breaks down as the actor is about

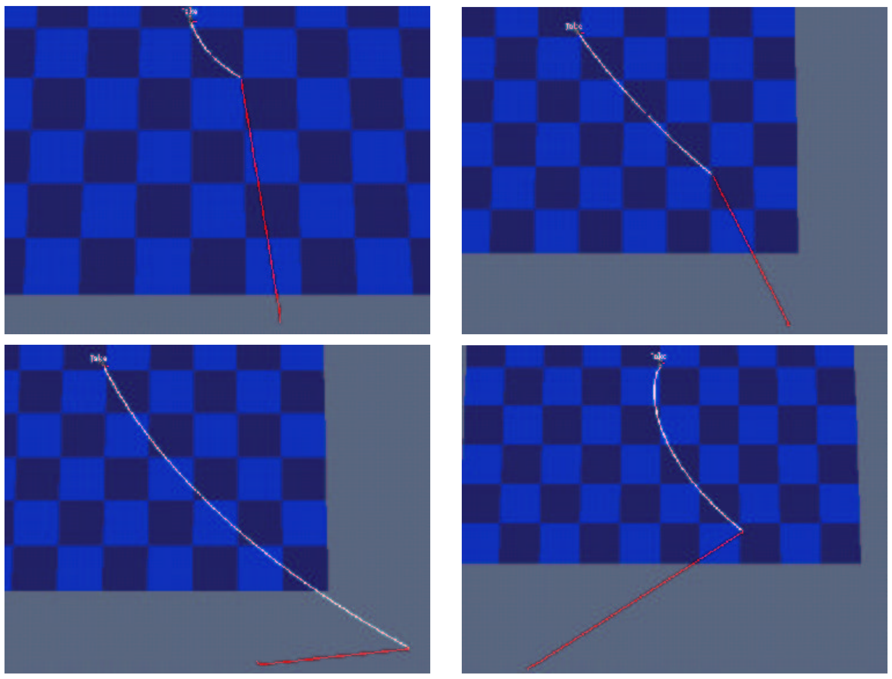


Figure 3.12: Running curves produced the linear optical trajectory method. The white curve is the actor's path and the red curve is the path of the ball.

to catch the ball. In the final frames of the two animations the ball coincides with the actor's head or chest. This is not desirable if the ball is to be caught in the hands but it is inevitable in any visually based strategy. If the fielder is judging visually how to run to catch the ball it is most natural to have a strategy that attempts to make the eyes and the ball coincide. Any other strategy would involve some sort of offset which would presumably be far more difficult to judge. However, as having the ball coincide with the eyes is undesirable (not to mention painful) it must be necessary to switch to a different strategy when the ball comes close and the fielder actually has to catch it.

Figures 3.15, 3.16 and 3.17 show the curves and animations produced with drag. The results are slightly less accurate, in figure 3.16 frame 4 the ball does not quite coincide with the actor's head, the deviation is not great however and the accuracy seems sufficient. Also the actor has to move faster resulting in longer strides as in 3.16 frame 3. As discussed above a running motion should be used here. However, even with a running motion there will be balls which require a speed that it too fast for the actor and so a maximum speed should be put in place above which the actor cannot run. This is done in figure 3.17. The actor now cannot run fast enough and does not reach the ball in time to catch it.

The LOT strategy seems to avoid some of the problems of the other strategy and does seem to agree with some experimental findings. However, it is still difficult to know if it is correct. Certainly, the psychology community have yet to agree. This means that though

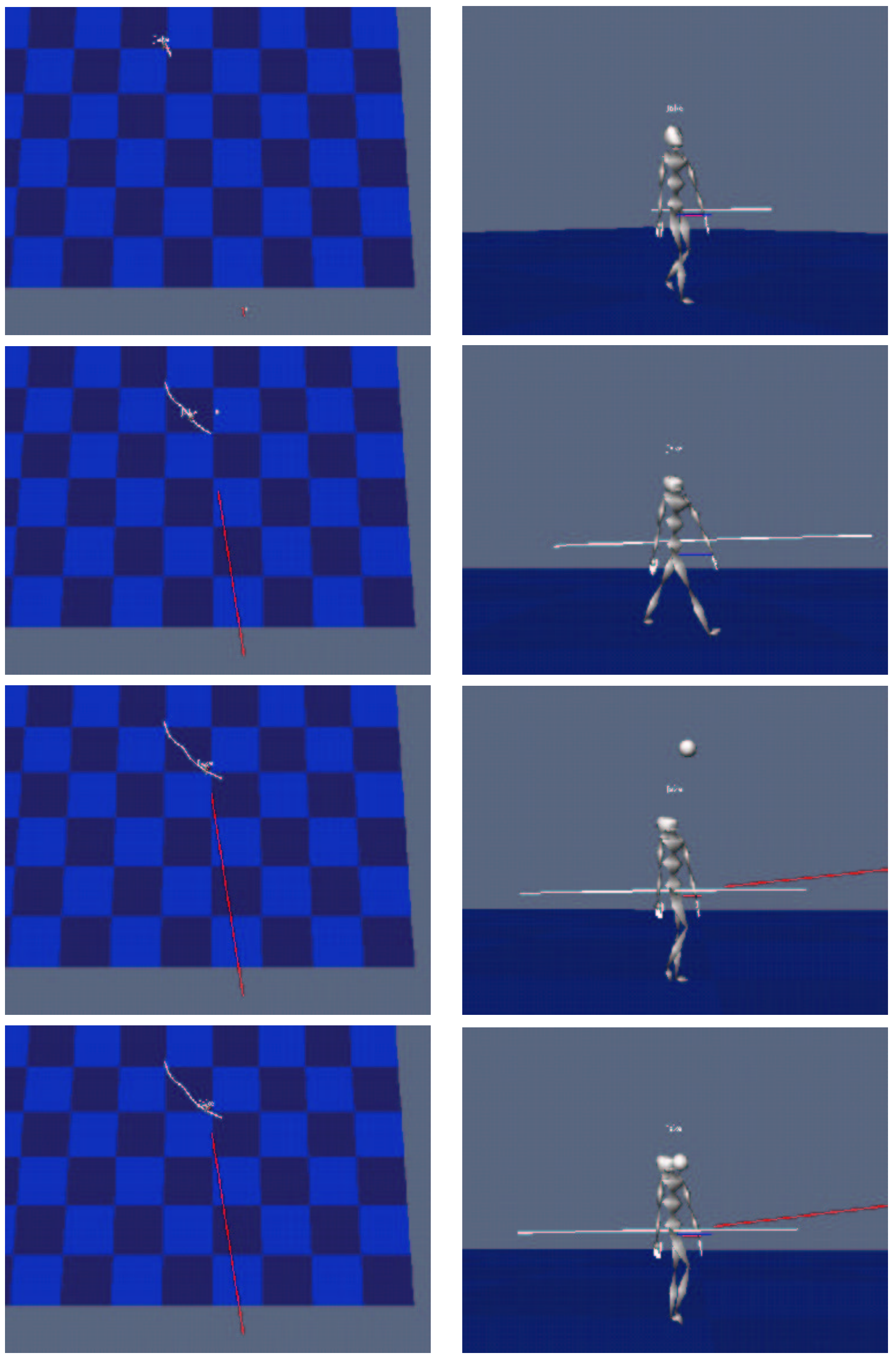


Figure 3.13: [chapter3/3-13lot.mov]Animation produced using the LOT method, shown from both top and side views. The frames are equally spaced.Note the actor watching the ball. An attention request (see chapter 4) was used to have the actor look at the ball. Note the breakdown of the algorithm as the ball gets very close, with the ball coinciding with the actor's head rather than an ideal catching position just in front of the actor's face. A different algorithm is needed for the actor to judge how to move its hands to the correct place to catch a ball. A method similar to that described in section 5.6.2 could be used.(left to right, top to bottom)

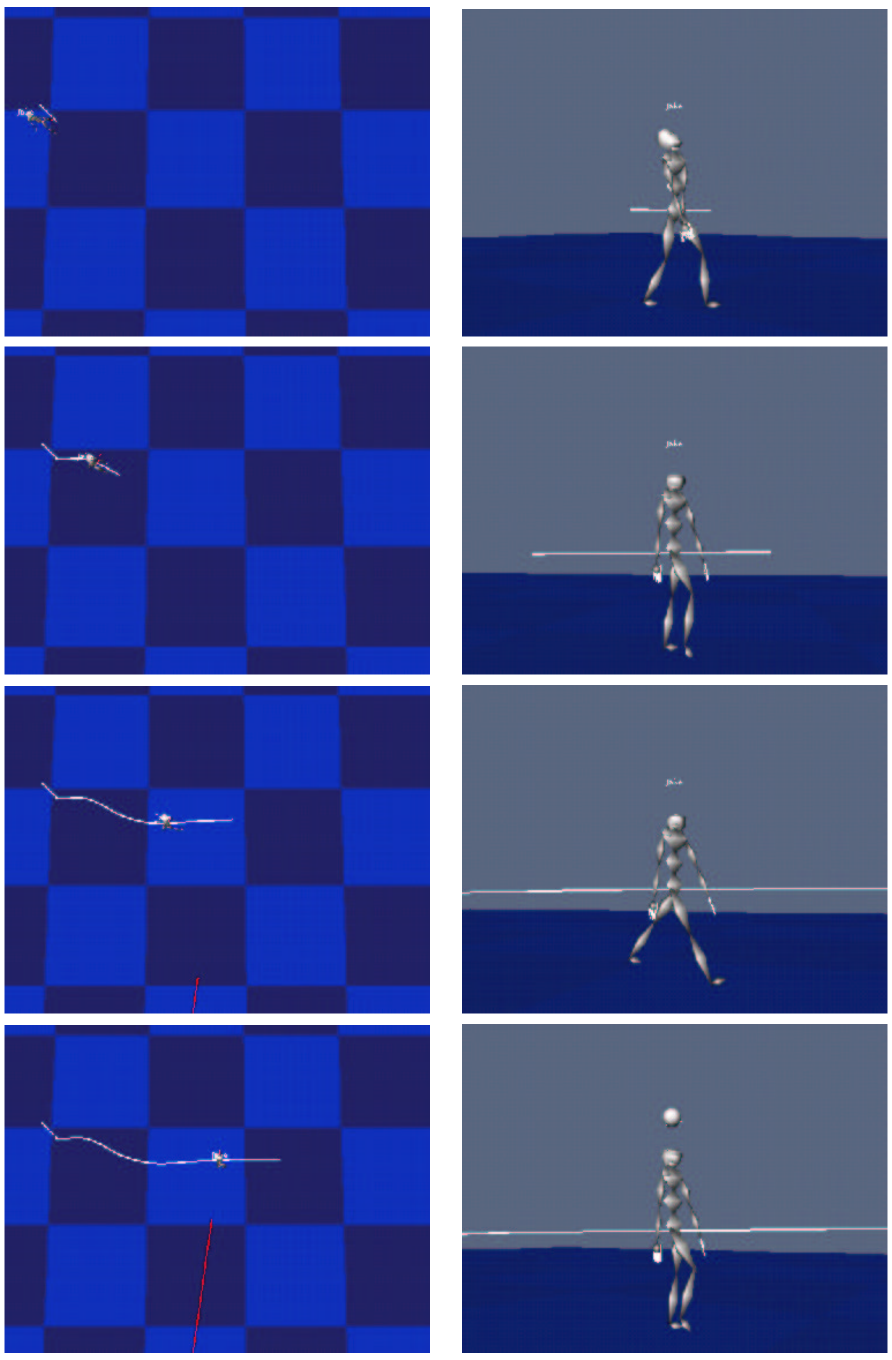


Figure 3.14: A second animation produced using the LOT method. In this case the ball goes very high and cannot be seen in all the frames shown from the top.(left to right, top to bottom)

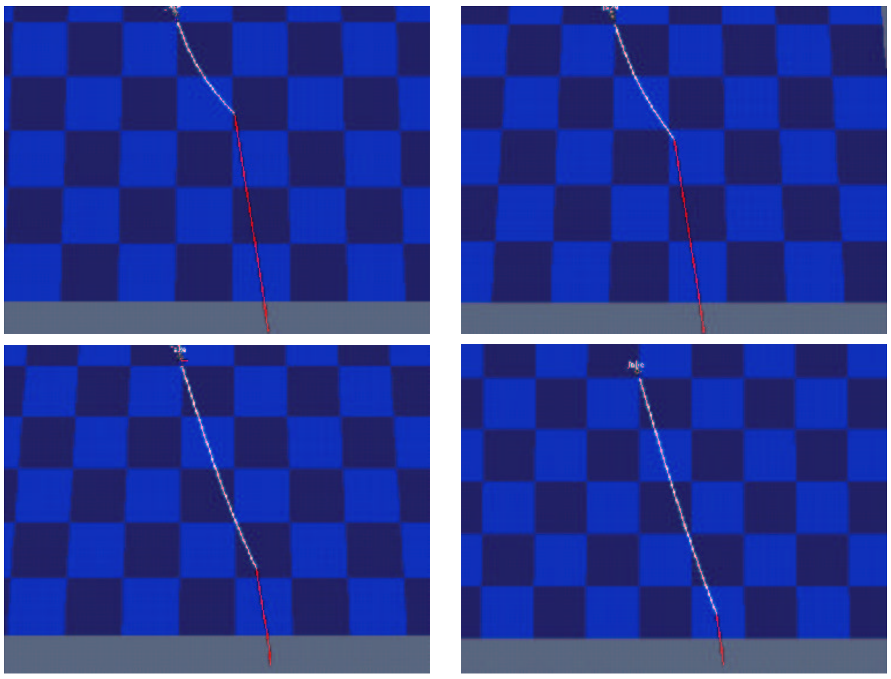


Figure 3.15: Running curves produced the linear optical trajectory method with drag present. The white curve is the actor's path and the red curve is the path of the ball. The three cases show the same initial conditions of the ball with different drag constants, 0.001, 0.005, 0.01 and 0.05

it is an algorithm that works well, its validity as a true simulation of human behaviour is still somewhat questionable.

3.6 Summary and Evaluation

This chapter has outlined a methodology for creating visually-guided behaviour based on psychologically inspired visual algorithms. It is possible to create very simple efficient algorithms such as the method for detecting potential collisions described in section 3.5.1. The psychology literature can often suggest a better, more efficient algorithm than a more normal computer science algorithm which would just be based on the geometry of a situation or on computer vision. For example, the method of detecting collisions is much quicker than trying to predict collision points geometrically. Section 3.5.1 gives a simple example of this.

However, the example of ball catching (section 3.5.2) shows some of the problems of the psychological approach. These are mostly connected to the imperfect understanding of the human visual processes. These processes are very complex and have only been studied for a relatively short time and so our current understanding is necessarily partial.

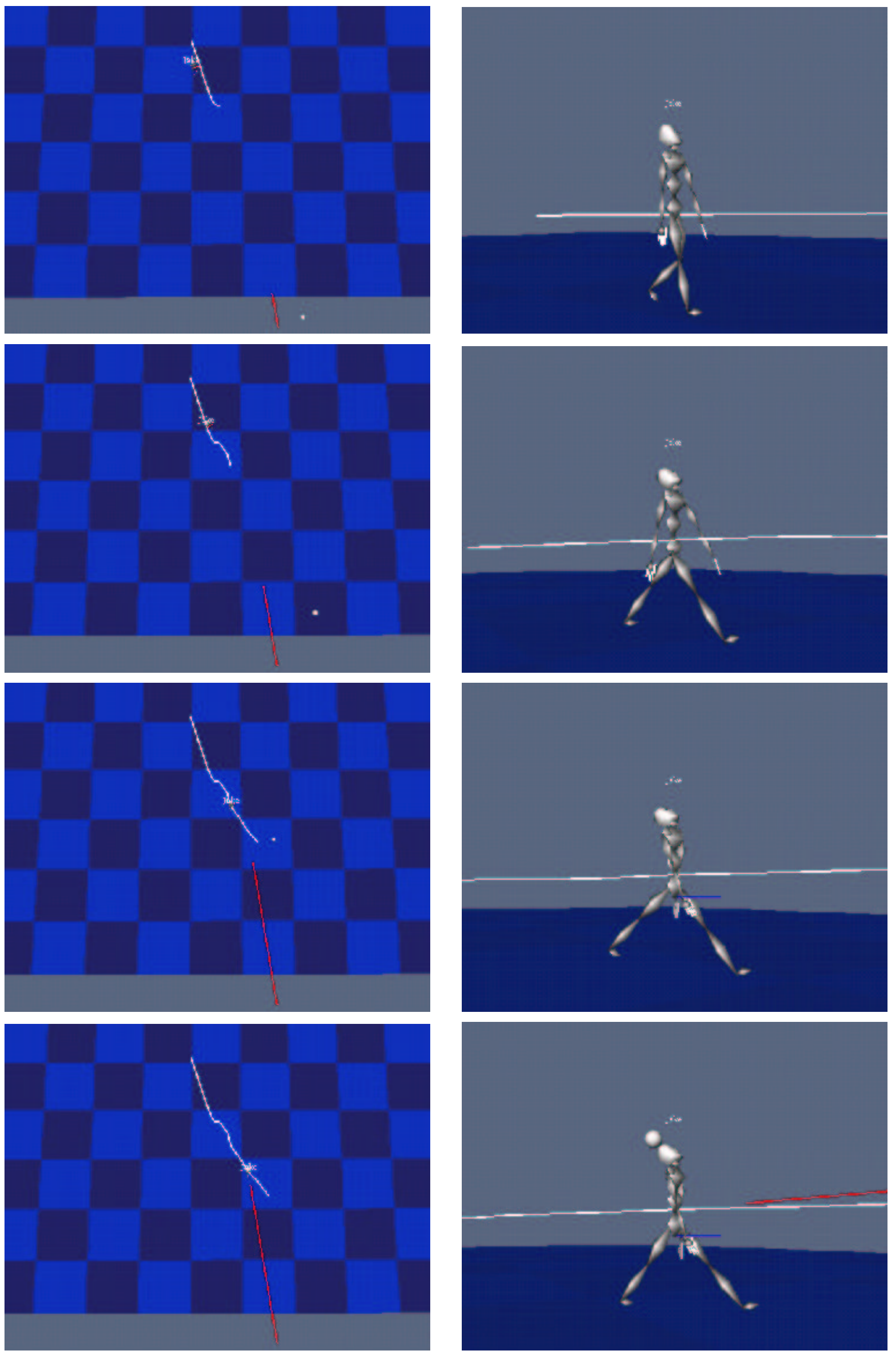


Figure 3.16: Animation produced using the lot method with drag present.(left to right, top to bottom)

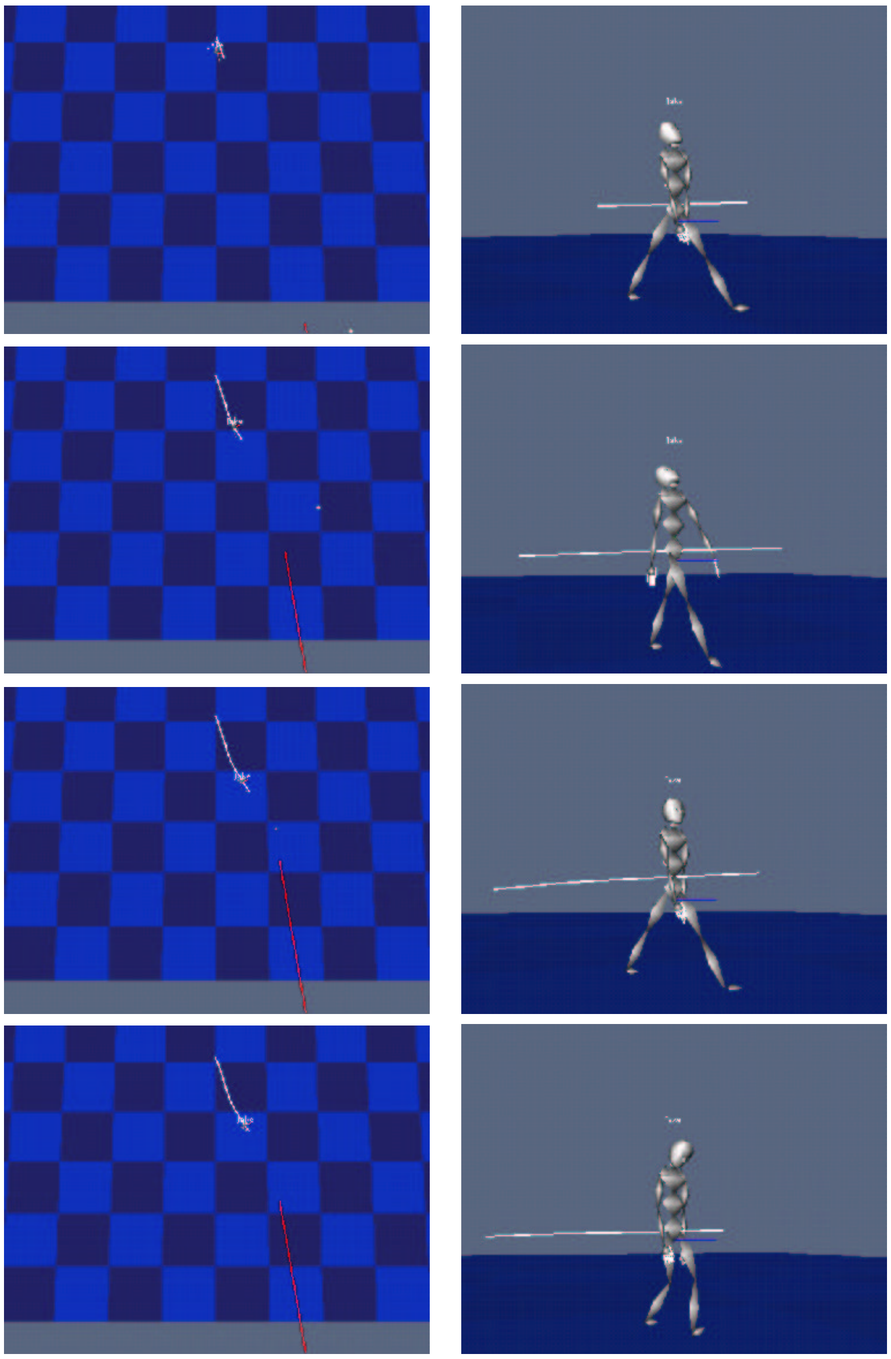


Figure 3.17: Here the actor is constrained by having a maximum walking speed, with the same ball trajectory as figure 3.16. The actor now cannot run fast enough and does not reach the ball in time.(left to right, top to bottom)

In particular, this example has shown that it is not necessarily agreed with much certainty how a particular visual process works. It is not known for certain which of the two, very different, strategies for catching a ball is used by real players, if either is. For computer animation the criterion for judging the correctness should be people's perception of the realism of the animation. For the moment the LOT strategy seems to produce a more convincing path. Another problem shown in the implementation of the LOT strategy is that the exact details of a strategy are not necessarily understood. This means that it is not trivial to convert a theory into an algorithm. This creates the possibility that the programmer's implementation will differ from the real human strategy in a significant way. The method I have described is only one way of maintaining a linear optical trajectory, and not necessarily the right one. A final related problem is that there are many many aspects to human visuo-motor behaviour and not all have been studied or understood. This means that there simply might not be a theory of how a particular activity is performed. If this is the case, the programmer must find a non-psychologically based algorithm for the behavioural animation of that activity.

The psychological approach is nevertheless a valuable methodology for designing behavioural algorithms. With some thought it can produce efficient and realistic algorithms which are widely applicable. It should therefore be considered as a starting point whenever a visual algorithm has to be designed. However, there are certain drawbacks. Firstly it is tempting to think that using a psychologically based algorithm will automatically produce a correct, realistic algorithm. This is not the case, however. The current state of psychological knowledge is still uncertain and a theory might not correspond to the truth, the theories of ball catching are uncertain in this way. Even if a theory is well established and very likely to be the truth there are still problems with the application and implementation of a theory. The behavioural algorithm might be an example of a case where the theory does not apply exactly. For example, it has been proposed that TTC is used in a wide range of behaviour but is uncertain which of these it is actually used in. Even if the theory is applied well there might still be uncertainty about how it should be implemented, for example, the implementation of the Linear Optical Trajectory method. Thus though an implementation might appear to implement a theory it might, in fact, differ significantly from how real people perform a particular action. This means that there is no guarantee that an algorithm produced in this way is a faithful representation of the psychological or psychophysical result, or a correct reproduction of human behaviour. Though it is tempting to thinks so, this method will not produce a provably correct simulation of human behaviour, and for the reasons described above it does not seem that it will be possible to produce such a provably correct simulation. Finally, there might not even be an applicable theory for a given task rendering the methodology useless.

The best advice to give is try the psychological approach first but to be aware that it might not yield a suitable result. This chapter has described cases where the methodology has been useful in creating good algorithms. In one case a new and powerful algorithm for the much studied problem of avoiding collisions. The next chapter, however, describes an the simulation of attention where the application of psychology is more problematic.

Chapter 4

Attention

Attention is a very important aspect of behaviour and one that is closely tied to vision. It consists of the focusing of a person's perceptual and cognitive capacities on a particular object or location. Though we are generally aware of the environment around us as a whole we generally attend to or "look at" one place at a time. Our perceptions of the focus of our attention are more detailed and we are more likely to be aware of and remember events that occur at the focus than away from it[Pas98]. For example, it has been shown in experiments that if a subject is reading text on a computer screen it is possible to completely change a word in the text without the subject being aware of it so long as the word is not the word being currently read. These factors make attention a vital part of human psychology. It is vital to understanding how we perform tasks and our awareness of the environment around us. However, it is also important in simulating human behaviour.

In simulating behaviour it is very important to take account of which objects in the environment the actor is aware of, otherwise the actor will react to objects and events that it does not know about. Vision is important in determining which objects the actor is aware of but it is wrong to think that what the actor is aware of is the same as what is in its visual field. This is for two reasons. Firstly, an object might have passed out of the actor's visual field but the actor will still be aware of its existence and probably rough position and so it is important to take it into account in certain tests. Also though an object is in the actor's field of vision the actor will not necessarily be aware of it. Though a person's field of vision as a whole is quite large the actor's attention is normally concentrated in a small region, close to the fovea of the eye. Objects outside this region might not be noticed. Large and prominent features might capture the attention, as would obvious motion, but much of the periphery would not come to a persons awareness. Attention handles both of these situations. Attention filters the set of objects in a scene into a set of those that the actor is actually aware of. The actor is aware of the objects that it is attending to. As well as the objects that the actor is directly attending to certain objects in the periphery will capture the actors attention, for example moving objects, and so the actor will become aware of them. Also certain objects in the periphery may be considered so obvious that the actor will be aware of them without attending to them. Attention can also handle the fact that the actor should also still be aware of objects that have passed out of its field of vision. Once the actor has attended to an object it can be stored as an object that the actor is aware of and so can be considered without it being in the field of vision.

It is thought that the function of attention in humans is to make efficient use of cognitive resources by applying them to one object at a time. Happily, a simulation of attention can perform a similar role in behavioural animation. Many computations are only performed on the focus of attention, for example, as described in section 3.4.4 only the focus of attention is tested for occlusion. It can also often be assumed that objects which have not been at or near the actor's focus of attention have not been noticed and so they can be excluded from other tests (of course, as has been said above, some objects can be so obvious that the actor will be aware of them even if they are only in peripheral vision). These factors, though included to increase realism, also make the behavioural simulation faster. Testing every object in a scene is slow and can also be too noisy with too many objects influencing the actor's behaviour resulting in uncertain or confused behaviour. Section 3.5.1 describes an algorithm which would not work if applied to every object in a scene but works well when applied only to the focus of attention.

Another important reason for simulating attention is that it is very closely linked to eye movements. In general people will look at an object or location that they are attending to. It is possible for someone to attend to an object that they are not looking directly at (as was first shown by Helmholtz in an experiment described in [Hel68]); however, this is an unusual situation that is difficult and requires concentration. This means that simulating where an actor is looking and simulating their attention patterns amounts to the same thing. Simulating the eye and head movements of a person as he or she looks around his or her environment is vital to creating a convincing character. An actor can move and act highly realistically but if their gaze is fixed and unmoving they will seem lifeless and inhuman. Gaze patterns are one of the most expressive aspects of human outward behaviour giving clues to personality, emotion and inner thoughts. Film makers make great use of their actor's patterns of gaze to suggest their thoughts and it can be vital in how we judge other people. Garau, Slater Bee and Sasse[GSBS01] have studied the effect an avatar's eye movements have on a person's interaction with the avatar. This was studied in the context of a one on one conversation mediated via avatars. They compared four conditions, only audio, a video of the other person and two avatar conditions. One condition had random head and eye movements and the other had head movements based on the users head and eye movements based on a simple model of eye contact during conversation. They found that, while the random avatar was no better than the audio condition, the appropriate eye movement avatar was almost as good as video according to some measures though not all. This shows that merely having eye movements is not enough, it is also important to have a good model. In fact, Garau *et al.*'s model was very simple (based only on average eye contact statistics) and many participants mentioned that the avatar seemed insensitive as its eye movements did not change suitably in sensitive sections of the conversation. For all these reasons if we are to have a good simulation of a character's behaviour it is vital for it to include what the character looks at and how it looks at it and for their to be a good behavioural model for these eye movements.

One important aspect of gaze patterns is that it is very expressive of the differences between different characters. This means that there cannot be a single answer to what patterns of gaze a character should display in a given situation but that it depends highly on the character itself. This means that a simulation of attention cannot be aimed at producing a single "right" answer but at providing a wide range of possible expressive behaviour.

Thus simulating attention is a combination of two tasks. The first is determining what the actor is aware of in order to simulate behaviour. The second is to generate appropriate eye movements. The requirements of these two tasks should determine the details of the simulation. In particular, for awareness, it is only really necessary to determine the actor's attention at the granularity of objects while for simulating the eye movements it is only necessary to generate shifts which will produce a noticeable change in the actor's angle of gaze.

This chapter will describe the simulation of attention. The next section discusses the application to attention of the psychological approach described in the last chapter. Section 4.2 gives some observations of human gaze behaviour given in the literature on psychology and anthropology as well as some further observations made by myself. Section 4.3 describes previous work on simulating attention. Finally section 4.4 describes my work.

4.1 Applying psychology to simulating attention

The obvious approach given the discussion of the last chapter would be to attempt to apply the psychological approach presented as a model of vision to the simulation of attention. However, there are a number of problems with this. Attention is a complex behaviour, sometimes consciously directed, sometimes not, sometimes driven directly by physical stimulus, sometimes directed by social convention, sometimes shifting in rather arbitrary, undirected ways. It is therefore very difficult to come up with a complete theory of how attention is directed, and no such theory exists. Shifts of attention are also innately connected to a complex environment. Normal human patterns of attention behaviour are only displayed when there are many stimuli competing for a person's attention. As such it is rather difficult to study the full range of attention behaviour in simplified laboratory conditions. Most experimental studies have focused on people's attention behaviour when presented with relatively static two-dimensional display[1]. These are very important in the study of reading, say, or for the design of computer displays but do not shed much light on the dynamic motion tasks discussed in this thesis.

Existing experimental data tends to deal with small eye movements and attention shifts, typically a few degrees of visual angle, over small time periods. For example, an important result is the discovery of inhibition of return, which means that, after leaving a focus of attention, the attention is inhibited from returning for the order of a few hundred milliseconds[PC82]. However, this time scale is small compared to that involved with simulating behaviour. Also this refers to small attention shifts whereas for animation

[1]see Pashler[Pas98] for an overview

the important factor is bringing to attention whole objects which would typically involve several micro attention shifts over the object each of which might be very short but taken together take a fairly long time. Also the small saccadic eye movements are not generally noticeable and so are not relevant to the animation of eye movements which should also deal only with large scale attention shifts. Some theories seem simply unsuitable for producing a faithful algorithm as described in the last chapter (Noton and Stark's scanpath theory[NS71] is an example in the current application).

A final problem is that scientific theories tend to produce general theories and average results, attempting to find the factors that are in common over all human behaviour. However, when simulating human attention and eye movement it is very important to represent the difference between different peoples' behaviour. This is what gives personality to actors. It is therefore more important to include the possibility of variation into a model than exact rules.

The approach for simulating vision should therefore not be to find theoretical models of human attention and attempt to implement them in an algorithm but to observe people's outwardly visible eye movements. These observations should give the range of different types of eye movements (and therefore attention shifts) that people produce. A general model can then be built that allows these different types of behaviour. The next section will give a variety of these observations.

4.2 Some observations

The observations of people's attention behaviour in the sort of situation I am interested is limited. There has been a lot in experimental laboratory conditions but this, however, is normally too different from the real world active tasks to be of relevance. There has also been a large amount of observations of people's gaze behaviour while in social situations. This mostly focuses on mutual gaze (where two people look at each other simultaneously) and other related actions. Argyle and Cook[AC76] give a comprehensive overview. However, this dissertation does not look at the relationship between two people's gaze patterns, section 7.4.4 will look at possible extensions of the current work to social interaction and will discuss issues of mutual gaze. Yarbus[Yar67] gives an excellent discussion of what it known about people's eye movements, which is very useful. However, I found that it was necessary to make my own observations in order to find a space of variation for human eye movements and the externally visible aspects of attention behaviour. These observations were made in an informal, non-experimental setting. This was done because it was important to observe peoples' actual attention behaviour in a natural setting. Also as the aim of the observations was to extract a space of variation it was important to get as wide a range as possible of behaviour and not limit it through a fixed experimental technique. Looking for a space of variation also meant that exact statistics and average behaviour patterns were not important.

The observations were made of people in a number of informal situations. Generally I observed people in normal everyday activities while attempting to not make it obvious that I was watching people. As well as observing patterns of behaviour that of large numbers of people I also looked for more unusual behaviour which might be more destictive or characteristic, and thus important to capture.In particular, an attempt was made to observe people's eye movement in films as this is arguably more relevant to producing an animation tool than people's actual real behaviour. The following points come from these observations as well as a number of texts.

- Though it is possible for people to attend to a location they are not looking at, eye movements will generally follow a person's attention [Hel68].
- People will generally maintain a constant angle of vision to the horizontal and normally only rotate their direction of gaze in the horizontal plane. This angle will generally point slightly downwards, or possible straight forwards. People rarely look up unless they are looking at something specific. A raised gaze tends to suggest confidence, a lowered one shyness or sadness (as noted by Polichroniadis [Pol00]). This observation was based on people walking around and so locations just ahead of the actor would be a natural place to look. It is possible that in other activities the natural direction would be different (for example, someone eating might look at their plate).
- There is a wide variation in the length of looks but it is notable that the distribution appears to have two peaks. People will tend to intersperse very short looks (which I will henceforth call *glances*) with longer looks (which will be called *gazes*).
- People, when walking down the street, will often have a behaviour pattern where they will have long looks forward interspersed with short glances elsewhere in the environment. This is likely to extend to other tasks where a person will look at the focus of the task interspersed with glances at other locations. The opposite behaviour is also common, people spend most of their time looking around themselves while occasionally glancing ahead of themselves, presumably to ensure they do not walk into something. People seem to exhibit either one behaviour or the other. I did not observe anyone change from one pattern of behaviour to another but since the length of observation of each person was short a solid conclusion cannot be drawn from this.
- In general people tend return their gaze repeatedly to the same place, looking a number of times at something that interests them (I will call this *monitoring*). Yarbus notes this behaviour in people whose eye have been tracked while looking at pictures [Yar67, page 194].
- The length of *saccades* (the movements of the eyes from looking in one direction to another) tends to very short. They take up less than 5% of all looking behaviour. The maximum length is about 0.08 seconds. This is the duration of two frames at 24 fps. Though this would be just noticeable the length is short enough that saccades can be shown as instantaneous without being too noticeable, especially when such long saccades are rare. When people appear to be moving their eyes slowly they are

in fact performing a number of short saccades. The same method could be used to produce the same effect in the simulation. This observation is from the results of experiments described in Yarbus[Yar67, chapter IV].

- Though the previous discussion has been in terms of eye movements the direction in which someone's eyes are looking can be hard to see. People's attention is normally seen through the orientation of their head. Apart from small changes of direction people will generally change the direction in which they are looking by moving their head. The actual eye movements only become important when the actor is in close up or where the actor is looking in the general direction of the viewer, when people are very good at determining if someone is looking at them or not.

- In films the characters will often look at an important item in a scene in order to emphasise it. This is similar to Argyle and Cook's observation that people will look at an object during conversation as a subtle way of pointing at it[AC76, page 121].

4.3 Previous work

Eye movement during conversation has been simulated in a number of ways as a form of non-verbal communication during conversation, for example Thórisson[Thó98]; Vilhjáalmsson and Cassell [VC98], or Colburn, Cohen and Drucker[CCD00]. However, it has not been studied often in other contexts or as a full attention system that it integrated with behaviour. Hill [Hil99] has a simulation of attention for a virtual helicopter pilot that include attention capture; selective attention based on features of objects and their importance to the actor's task, and perceptual grouping of objects. It is applied to a helicopter in a military situation not to the animation of a human figure, however, and does not produce animations of eye and head movements.

The only previous work that tackles the same problems as the present work is by Chopra-Khullar and Badler[CKB99]. They produced an architecture for producing gaze behaviour in a virtual actor. It is based around PatNets, behavioural agents using finite state machines. There is a PatNet called GazeNet which arbitrates between requests for gaze shifts coming from other PatNets and produces the gaze animation. The work presented in this chapter has used a similar overall structure with a central *attention manager* agent and external agents which make requests. However, the arbitration system is different. Chopra-Khullar and Badler divide attention tasks into three types, *intentional gaze behaviour*, *peripheral attention capture* and *spontaneous looking*. Intentional gaze behaviour results from some task and comes from other agents. This is the highest priority behaviour. Peripheral attention capture represents some object or event in the periphery of vision coming to the actor's attention and the actor looking at it. In Chopra-Khullar and Badler's system attention is only captured by moving objects. *Spontaneous looking* represents idling gaze behaviour, when there is no other important event capturing the actor's attention the actor will look at interesting aspects of the environment, section 4.4.3 discusses *spontaneous looking* in more detail. The arbitration between the various eye behaviours uses two queues, the IntentionList which contains Intentional requests coming from other PatNets or from user input and the PList which contains peripheral events which might capture the actor's attention. The actor first checks if there are any requests in the intentionList that need processing in which case the actor will normally process the first of these requests. However, there is a certain probability that the actor will be distracted by a peripheral event and process the first request from the PList instead. If the IntentionList is empty the actor will process requests from the PList. If both lists are empty the actor will perform *spontaneous looking*.

The author's initial experiments with simulating attention used a queue system similar to Chopra-Khullar and Badler's. However, there are disadvantages to this sort of system as it is unsuited to producing timely gaze shifts and not that suited to producing general monitoring behaviour. Certain eye behaviour is closely connected to a task and has to have occur at exactly the right time, synchronised with the task. The problem with a queued system is that it is difficult to make sure a gaze shift occurs at an exact time. If the queue is not empty when the request is issued it will be delayed behind other requests. Even if the queue is empty the eye request may be pre-empted by a peripheral event and so be delayed. These delays can result in the eye behaviour associated with a task happening after the task is finished which is clearly undesirable. It is therefore better to specially handle timely events. Even for looks that are not time constrained the queue system is not optimal. The problem is that most such gaze behaviour does not consist of a one-off look but rather an interest in an object or location to which the actor will often return. It is clumsy to produce this sort of behaviour with a queue-based system as agents have to add new request to the queue at rather arbitrary intervals, while keeping track of which requests have been sent in case they stop being relevant and so have to be removed from the queue (a common situation). It would be better to build a system around monitoring which involves just a single monitor request and a single purge request.

4.4 Attention Architecture

This section will describe the attention mechanisms themselves. They are based on a set of communicating agents as described in appendix A. The agents are shown in figure 4.1. The attention manager is the main agent of the attention architecture. It controls and arbitrates the attention behaviour and controls the eye movements of the actor. Other agents send requests for attention shifts to the attention manager. Examples of these are the *Peripheral Vision* agents described in this chapter or the *EyeAction* agent described in chapter 5. Other agents that interact with the attention manager are described in the next two chapters. The user can also request that actors make attention shifts. By clicking on an object the user can bring up a menu that includes various options for looking at the object (figure 4.2). For example, glancing at it or monitoring it.

As well as receiving requests from other agents the attention manager sends information about what the actor is looking at to other agents. These agents can react to this information while producing behaviour. The focus of attention is passed to other agents for use in their behavioural algorithms The focus of attention is only used directly if it

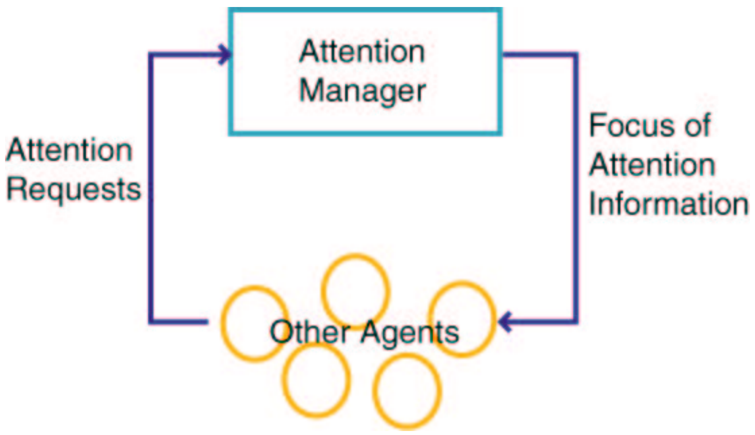


Figure 4.1: The Attention Manager and other agents. The attention manager receives requests for attention shifts from other agents. It then arbitrates between these and eventually executes them. These change the focus of attention which can then be used by the other agents.

is an object. If it is a location the actor is not looking at anything in particular and so the actor's awareness of the environment is only affected through peripheral vision (see section 4.4.6). If the focus is an object, it becomes the object on which the actor's main tasks of observation are performed.

4.4.1 The Attention Manager

The Attention Manager is the main agent involved with eye movement and attention and it performs various functions. Its major function is to supply a series of gaze directions to the *mannequin* agent and a series of foci of attention for those agents which rely on the attention of the actor. In order to do this it must manage the various requests for eye movements and attention shifts made by other agents and arbitrate between them, choosing a single one at a given time. It also has a secondary function of generating searching and undirected attention shifts which will be discussed in section 4.4.3.

Like Chopra-Khullar's system the attention manager receives attention requests which are then processed to produce attention shifts and eye movements. However, the requests are processed in a different way to Chopra-Khullar's. The next section describes the structure of the attention requests and section 4.4.4 describes how they are used.

Figure 4.3 gives an overview of how the attention manager works. Requests are sent to it at any time by other agents. When the attention manager is ready it will process one of these requests. It will do this by arbitrating various requests as described in section 4.4.3. It will then transform the request into actual attention behaviour, possibly moving the eyes as described in section 4.4.4. This attention behaviour will last for a length of time. When it is finished the attention manager will choose and process another request.

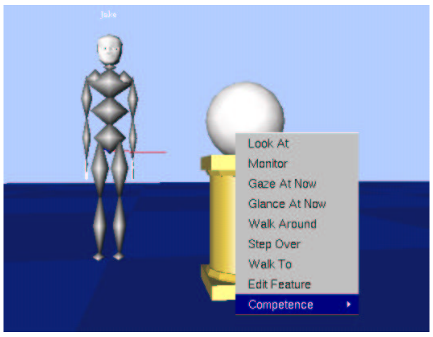


Figure 4.2: Clicking on an object brings up a menu with various options, including some that allow the user to directly command an actor to attend to an object. This can be done in various ways, monitoring as well as immediate requests for glancing and gazing

This timing can be overridden by another agent requesting that an attention shift occur immediately as described in section 4.4.3 in which case the current attention behaviour is interrupted.

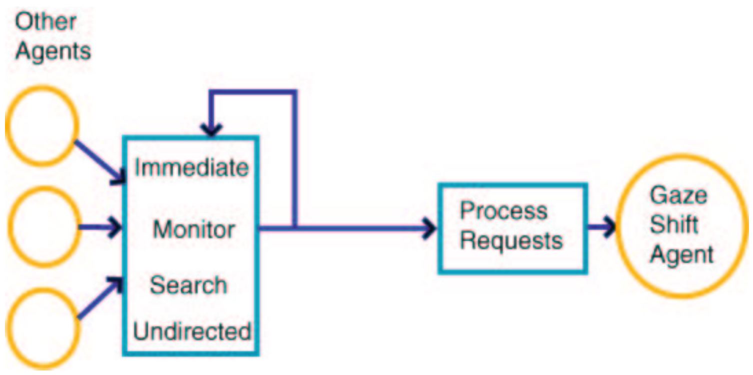


Figure 4.3: The Attention Manager receives requests of various different types: Immediate, Monitor or Search requests. It chooses between these, or if none are presents performs undirected attention to generate a request. When a request has been chosen it must be processed to turn it into an attention shift and eye movement. If necessary it is then sent to the gaze shift agent which moves the actor's eyes.

4.4.2 Attention Requests

The attention manager receives requests for shifts of attention via Comms which are shared between a number of different agents (Comms and agents are described in appendix A). These agents can put values onto the Comms, representing where the agent should look, and the attention manager can deal with them in its ParamChanged call. As

ParamChanged can be called more than once a frame many agents can put requests onto the same Comm in a single frame and they can all be received. Each request has a pointer to the agent that made the request so that the attention manager can notify the agent when the actor is attending to the request or if the attention manager fails to enact the request (section 4.4.4 describes which requests are rejected). The requests can be of three types depending on what the actor should attend to.

Location Requests represent a location in the world. They are specified by a position vector in world coordinates. As actor moves it moves its eyes to compensate for the motion and so keep fixating the same absolute position. This is used in situations where the actor has to look at a specific place in the world but one that does not easily correspond to an actual object, for example the actor's destination in navigation behaviour.

Local Requests are the simplest type of request, they consist of a vector representing a direction in local coordinates relative to the actor's head. Thus they are defined relative to the actor as opposed to globally in world coordinates. As the actor moves the eyes do not move but keep looking in the same direction relative to the actor's head. This type of request is useful for situations where the actor is looking at the environment in general rather than at specific places or things. An example of their use might be scanning the street ahead for obstacles or looking around as in *undirected looking* (see section 4.4.3).

Object Requests are requests to look at a specific object in the environment and are specified by the identifier of the object. The advantage of this form of request is that the eye manager has access to the object itself. This means that it can use the objects properties, for example, by tracking its moving position. It can also pass the object on to other agents which can use it to perform various visual algorithms as the focus of attention. This is the type of request most used by specific tasks as they generally require attention to a given object, for example if it is being stepped over. Also when attention is captured by a peripheral event it will normally be by an object. In the current work the actor looks at the centre of the object, it might be desireable to develope a more complex model (e.g. using Noton and Stark's scanpath model[NS71]).

Parameters of Attention Requests

As well as containing the location or object to be attended to the request also specifies various aspects of the actor's attention behaviour while attending to that object. These are specified in terms of flags and pieces of extra data to determine their effect:

- *glance* is a tri-value flag determining if the eye motion should be a short glance or a long gaze. 0 represents a gaze, 1 a glance and -1 (the default) is a don't care state where the attention manager determines which it should be.

- *move eyes* is another tri-value flag. If the value is 1 the attention manager will always perform both an attention shift and an eye movement. If it is 0 there will be only an attention shift. Again -1 is a default don't care state.

- *minimum distance* prevents the actor looking at objects that are behind it or too close for the current type of behaviour. If a request's location is less than its minimum distance in front of the actor it will be ignored, negative distances represent locations behind the actor. For example, while doing some detailed work on an object it would be normal to look at it if it were at a distance of under a metre, however, when walking around an environment the actor will look at objects further away and should have already taken close objects into account (see chapter 6 for more discussion). Also while walking around an environment the actor will tend to only look at objects in front of it, however, if the request is due to an object making a loud noise it might have a negative minimum distance.

- *interval* applies to requests that represent the actor occasionally monitoring a location. It is an approximate time between times that the actor monitors the location.

- *next look time* again applies to monitoring requests. It is the time at which the location should next be monitored. If it is initially set to the current time it will be looked at as soon as possible. The function of this parameter is fully described in the next section.

4.4.3 Types of Attention Behaviour

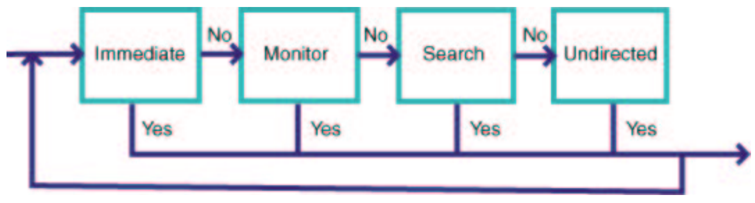


Figure 4.4: The sequence for choosing an attention behaviour. Immediate requests are chosen with the highest priority as they are time constrained. If none are present the monitor list is checked to see if it is time to monitor any of the requests. If neither is the case then there are no time pending actions. If the actor is searching for something it will continue to do so otherwise it will generate an undirected attention request.

There are three types of attention behaviour that can be requested by agents. *Immediate* attention shifts make the actor move its attention to the target as soon as the request is received. *Monitoring* behaviour makes the actor look at the target occasionally and *Searching* does not involve a direct target request but the attention manager generates attention shifts that look for a particular object. Finally if there are no requests from other agents the attention manager generates *Undirected Attention* behaviour, analogous to Chopra-Khullar's Spontaneous looking. This is a form of idling attention shifts. The attention manager arbitrates between the types of behaviour using strict priorities as shown in figure 4.4. Immediate requests have the highest priority. Monitoring has higher priority than search. This assures that monitoring happens with roughly the correct interval and that monitoring does not get held up by searching which could take up all of the actor's attention. Undirected looking has the lowest priority, happening only if there is nothing else for the actor to attend to.

Immediate Shifts

Immediate shifts are the simplest form of attention behaviour. Other agents can put an Attention request in a specific immediate request Comm which is sent to the attention manager. If the attention manager receives a request along this Comm it produces an attention shift in the same frame, delaying any other pending requests or attention behaviour. To prevent an immediate request placed just after another one from overwriting the first the immediate request Comm is locked when an agent places a request on it. This means that if an agent attempts to place an immediate request while another is active it is unable to do so and is notified. The agent can then wait for the Comm to become free and if necessary delay any behaviour that has to be synchronised with the attention shift. This is superior to using a queue to schedule multiple request as Chopra-Khullar and Badler do as it deals with simultaneous multiple requests while still allowing attention behaviour to be synchronised with other behaviour. If the actor's attention is caught by a sudden noise while about to perform a task that requires its attention it will wait until it can attend to the task before performing it. It will not perform the task and then attend to it at a later time.

Monitoring

It is often the case that an actor is interested in a particular object or location and needs to attend to it but not necessarily at a particular time. It is also the case that if an actor is interested in an object or location it will be interested in it for a period of time and will want to attend to it more than once. Monitoring is this sort of behaviour. Monitoring requests are received via a Comm and are placed in an array. When the actor has finished attending to a location or object this array is tested to see if the actor should attend to one of the monitoring request.

How often the actor should attend to a monitored object or location is determined by the *interval* and *next look time* parameters of the attention request. When the current time is greater than *next look time* the actor has to look at the target. Once the actor has attended to the target *next look time* is set to the current time plus *interval* so that the actor attends to the target roughly every *interval* frames. The attention manager checks the array of monitoring requests by subtracting the current time from each of the *next look times* of requests. It then takes the minimum of these. If this minimum is less than zero (i.e. the *next look time* is less than the current time) the attention manager processes that request. This means that if more than one request is current the one which is most past its *next look time* is processed.

A monitoring request can be removed for a number of reasons. There is a parameter in the request that is the maximum number of times the actor should look at the target. When this has been exceeded the request is deleted. Normally this is set to infinite times so that the actor monitors the target until told to stop. However, it can be set to a small number so that the monitoring mechanism can be used to implement the case where the actor needs to look at a target only once or twice but without exact time constraints (i.e. not an immediate request). Another parameter in the request is the *minimum distance*, this makes the actor monitor an object that is a certain distance in front of it. This is normally set to a small distance in front of the actor so that the actor no longer attends to targets that have passed behind it or are about to. It can also be set so as to include targets some way behind the actor or only those at are far from the actor (this last can be used for monitoring people on the street where it is socially acceptable to look at people from a distance but not close up). Finally agents can request that a particular item can be removed from the monitoring array. This can be when an agent is making the actor monitor an object for some reason but then finishes that task and so is no longer interested in it.

Searching

Searching is a more complex attention behaviour and is really a behavioural competence in itself. It enables the actor to find objects with a certain property in the environment. There has been a lot of experimental work on visual search but unfortunately it has mostly been performed in laboratory conditions involving searching for objects on simplified 2D displays. However, some of the results can be useful. The experiments generally take the form of subjects searching for a particular object in a 2D grid of *distractors*, other objects that are in some way different from the target object, figure 4.5 shows four such displays. The targets in figures 4.5(a) and 4.5(c) are very easy to identify, they seem to *pop-out* of the display. This is a function both of the target and distractors and what it is that differentiates them. It is believed that there are a number of basic visual features of objects that cause this pop-out, Wolfe[Wol98] lists a number including: colour, shape, orientation, curvature, size and motion. The way pop-out works is complex and depends on how different the target and distractors are. The cases of fig 4.5 (a) and (b) are relatively simple as the target and distractors differ only on one feature, colour for (a) and orientation for (b), (a) pops out obviously, the distractors are all the same colour and a very different colour from the target. The effect is much less obvious in (b), the distractors have different orientations but their shapes are the same as the target. Cases (c) and (d) are more complex as the target is determined by two features, this situation is called a conjunction. Conjunctions are more complex, there is some degree of pop-out effect in (c), finding the target is certainly quicker than linear search but not constant time. Sometimes conjunction search can be as efficient as single feature search, sometimes not.

How does all this apply to searches in the world as in figure 4.6? Clearly there are

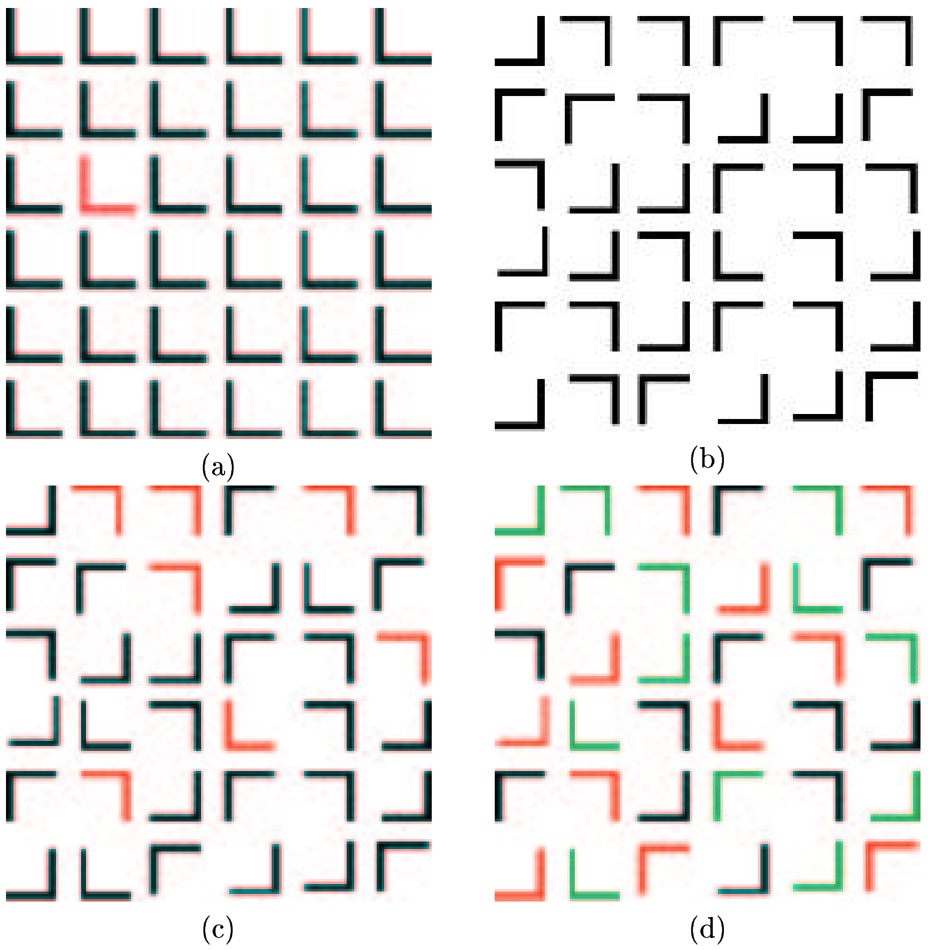


Figure 4.5: Four sample 2D visual search experimental displays. In (a) the red L seems to pop-out immediately. In (b) it is much harder to spot the back L from the other black shapes. Both searches are based on a single feature, colour or orientation but one is clearly easier than the other. (c) and (d) are based on two features. Finding the red L in (c) is fairly easy though not as easy as (a) whereas (d) is again difficult. Search on two features is more complex than on one feature but it can often be performed efficiently.

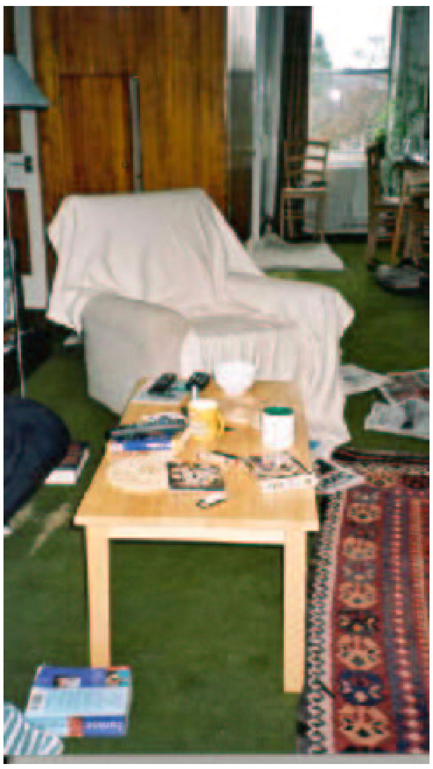


Figure 4.6: A real world search scene: find the white mug with a green interior. Clearly the situation is more complex than the laboratory style displays of the last figure. The white mug can pop-out to some degree, but so can other object such as the white bowl which is the same colour and a similar shape.

many multiple conjunctions of features some of which might or might not cause some degree of pop-out. If searching for the white mug it might pop out due to size or more probably colour. However, multiple objects might pop-out. For example, the orange mug might pop-out due to size and shape or the white bowl might pop out due to colour. In general, I will assume that a number of object will pop out of the periphery of vision with a probability depending on their similarity to the target. The number of possible features that could cause this pop out is too great and complex to encode directly into the program. Trying to do so would lead to incorrect results due to a simplistic model, as the workings of human search are unknown it would not be possible to make a complete model. Instead similarity to a target is encoded as a single *object feature* with the property for the target having value 1.0 (see section 3.4.3 for a discussion of object features).

Search behaviours are requested via a Comm like immediate and monitoring behaviours. However, they are not requested with an attention request as the target object might not be known, and the attention shift to the target will not be immediate. Instead the search request gives an *object property* to search for. Searching involves producing a number of attention shifts to possible positions of target objects, eventually resulting in an attention shift to a target if it is present.

First the periphery of vision is searched for objects that pop-out. Each object is tested in order. The order is just the order that the objects are present in the list of objects but the starting point is randomised to prevent the same objects always being found first. The object is first tested to see if it is in the periphery of vision (see section 4.4.6 for a discussion). Then the object is tested to see if it has already been searched, a flag is set for each object that has been looked at. Finally the object is tested to see if it pops out. The search property of the object is tested (if the property is not present it does not pop out). The object pops out with a probability equal to its property value. This means that target objects will always pop out. If an object pops out an object request is generated for that object and so the actor attends to that object.

If no objects pop out from the current periphery the actor will scan the rest of the scene. This is done by generating a location request either to the extreme left or right of the actor. These requests behave slightly differently to normal requests, the head and eyes move more slowly towards the target so as to scan the scene and the movement is interrupted every few frames to attempt another search of the periphery (if the next search fails this will result in the actor smoothly continuing the scan). When the scan in one direction has finished the actor will scan back the other way.

After all these scans have been attempted the actor starts walking around to look for the target. This is done by setting a destination somewhere at random in the semi-circle in front of the actor (see chapter 6 for more details of walking and destinations).

The actor does not stop searching when the first target is found. It is up to the agent that requested the search to stop the search and remove the request when the targets that it needed are found.

Figures 5.9 and 5.10 gives an example of the use of searching behaviour.

Undirected Attention

Undirected attention is the pattern of looking and attending that is produced when the actor has no definite attention required by its behaviour, i.e. no request sent by other agents. It is how the attention manager chooses a request when there is no immediate or search request and all the requests in the monitor list are dealt with. It is analogous with the term *spontaneous looking* used by Chopra-Khullar and Badler [CKB99] based on a term of Kahneman[Kah73].

My method for choosing where to attend to is different to Chopra-Khullar and Badler's. Chopra-Khullar and Badler's approach is image based, the scene is rendered to an image from the point of view of the actor and this image is used to determine places of interest. Areas of the image where the difference between the colours of neighbouring pixels is large are considered interesting and so the actor will look in that direction. Though it can produce suitable looking patterns this is not always a good heuristic. For example, an actor might be walking in a park over grass which is highly textured and so is likely to have a large pixel difference. During the walk the actor might pass a minimalist sculpture that is very smooth and so have a low pixel difference. However, the actor's attention is more likely to be drawn to the sculpture than the ground. In fact Yarbus states that, based on his experiments on tracking the eyes of people looking at pictures there is no connection between features of the image of an object and whether someone will look at it. Complexity is not a factor:

> any record of eye movements shows that, *per se*, the number of details contained in an element of the picture does not determine the degree of attention attracted to this element. This is easily understandable, for in any picture, the observer can obtain essential and useful information by glancing at some details, while others tell him nothing new or useful.[Yar67, page182]

He reaches the same conclusion about colour, brightness and contours. The only factor he accepts is the amount of information the observer can extract from a feature. In general it is not feasible to find a heuristic to determine what the actor finds interesting to look at as the reasons for finding something interesting are so varied. Instead I have tried to use a more general approach which allow the animator more control over undirected attention patterns.

There have in fact been two general types of behaviour observed in people moving or standing still in an environment without having a very definite object to attend to (see section 4.2). Some people tend to look forward and often slightly downward without moving their gaze around much. Others are much more likely to look around themselves at their environment. The looking forward behaviour is probably dependent on the fact that most of the people observed were on a street and mostly walking around. Different tasks might have a different default direction, for example, a person eating is likely to look at their plate. Thus the forward behaviour can be modelled as looking in a default direction. Thus the undirected looking request generator chooses between these two behaviour patterns with a probability that varies from actor to actor.

If the choice is to look at the default direction a local attention request is generated that results in the actor looking in that direction. There is no random variation in this gaze pattern as this form of behaviour seems to involve fairly constant gaze patterns. The default direction will normally be forwards and slightly downwards, unless some other agent overrides it.

If the actor looks around the environment its attention can be captured by an interesting object. This is different from the attention capture by important objects described in section 4.4.6 where the attention must be captured by an object which is relevant to the current task. That is directed attention. In undirected attention capture there is no particular reason to look at an object other than a vague interest. Each object has an interest value associated with it which determines how probable it is for an actor to look at it. For example, an animator might want to put a statue in a crowd scene and give it a high interest value so that many characters in the crowd will look at it. The interest value for an object applies to all actors but an individual actor can override the particular interest value for an object so that the actor can be more or less interested in a particular object than the average person (this is an example of the object features mechanism described in section 3.4.3). The undirected attention procedure chooses an object to look at by choosing an object at random from a set of objects it is aware of and then accepting it with a probability equal to its interest value. This results in every object being chosen with a frequency proportional to its interest value.

If there are few interesting objects in the environment choosing objects will result in constant repetition of the same gaze directions or even difficulty in finding an object to look at. For this reason the actor can also look at random locations around itself. This either happens by the system occasionally choosing at random not to look at an object or when it fails to find an interesting object after 15 attempts. If this happens it generates a random gaze angle about the vertical axis resulting in the actor looking somewhere in the $180°$ ahead of it. A rotation around the left to right axis is also generated. This can be a random angle but if the actor has a tendency to keep a preferred gaze angle to the horizontal it might be this angle. These two rotations are then combined to give a local request.

Both types of undirected looking are important for walking around an environment. Looking forward means that the actor knows if it is about to walk into something whereas looking around makes the actor more generally aware of the environment, if a cyclist is about to run into him or her for example.

4.4.4 Processing The Requests

The various attention behaviours will result in a request that must be turned into an actual attention shift and possibly an eye movement. This involves a number of steps as shown in figure 4.7.

Reject invalid requests

The first task is to test if the request is valid. There are three reasons that the request could be invalid. First if the location is nearer than the request's minimum distance see section 4.4.2. This is a simple test. The second reason is that the object is not visible because it is occluded. This test is described in section 3.4.4. Finally, a behaviour can specify that the actor has to keeps its head still, for example if the actor is eating and putting food into its mouth. If the actor has to keep its head still it cannot look at targets that require a head turn (i.e. if the direction of the request is further than a from the current gaze direction) and so these targets are rejected. If any of these tests fails the attention manager must find a new request in the same way. The agent that made the request is notified if it fails. Otherwise the request is successful and the current focus of attention is set to the location or object of the request.

Length of gaze

Next certain attributes of the request must be determined. The length of time a location is attended to depends on whether the request is a short glance or a long gaze. In each case a random length is chosen but the means and variances are different. If the glance

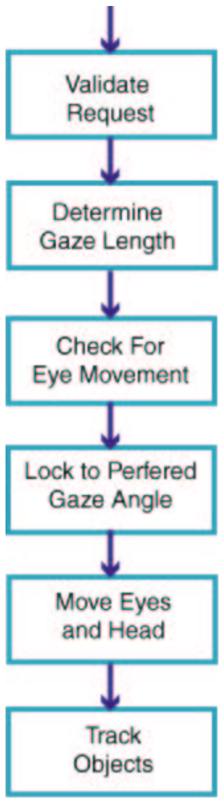


Figure 4.7: The sequence of actions that are performed on an attention request to execute it. These are the steps that must be take to create an actual attention shift and eye movement from a request.

flag is -1 (don't care) glance or gaze is chosen at random with a probability that can vary from actor to actor.

Eye movement or not

Similarly it is determined whether the request results in an eye movement based on the flag in the request or a random choice based on a probability particular to the actor. In general the probability of moving the eyes is set to 1 as attention changes without eye movements are of limited use in animation where things should generally be visible to the viewer. This agrees with normal human actions where eye movements and attention are generally coupled.

Preferred gaze angle

It has been noted (see section 4.2) that people tend to have a preferred vertical gaze angle (see figure 4.8). The angle will be the angle to the horizontal of the actor's default gaze

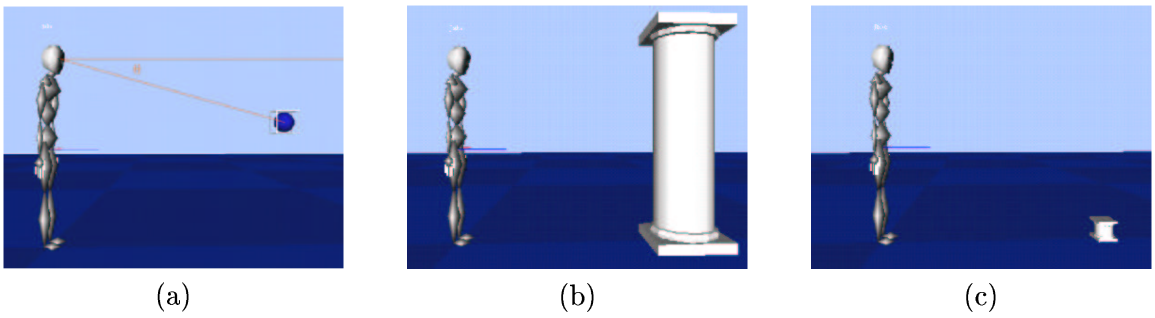


Figure 4.8: The vertical angle of gaze of the actor. The system tries to keep this constant as people tend not to change the height of their gaze often. Sometimes this is compatible with a given request as with the tall object in (b) but not always, the actor must look down to see the object in (c).

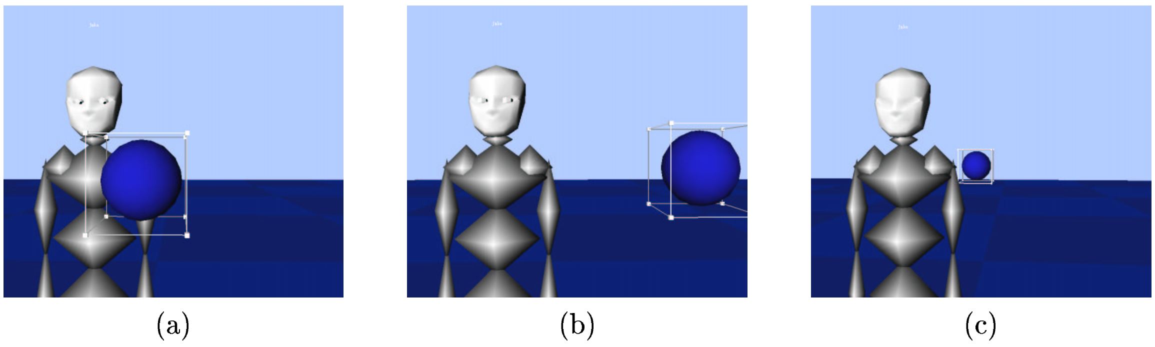


Figure 4.9: Though moving the eyes is sufficient is some cases (a) in looking in some directions can be awkward (b) or impossible (c) without turning the head.

direction (see section 4.4.3). Each actor has a probability with that they will maintain this angle which is randomly tested. If they do try to maintain this angle it must be checked that it is possible to maintain it by testing the height of the object being looked at. For the tall object in figure 4.8(b) the actor is able to look at the object without lowering their gaze but in panel figure 4.8(c) the actor must lower its gaze. If the gaze angle is maintained the XZ component is taken from the request and the Y component is chosen so as to maintain the angle. The gaze angle itself is specified relative to the unrotated head.

Moving the eyes and head

Once the details of gaze are determined the attention manager must actually make the

Thus it is important to turn the head. In fact as figure 4.10 shows turning just the head can also produce awkward positions and so the shoulders must also be turned. The

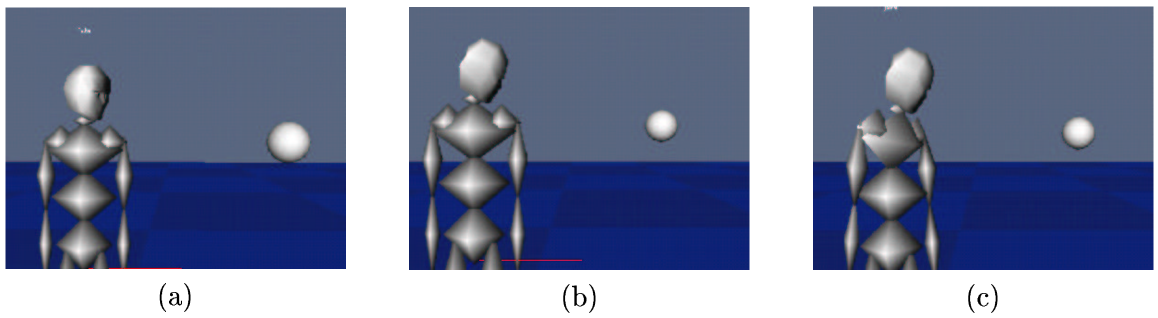


Figure 4.10: In (a) turning the head yields a natural looking position but in (b) the position looks awkward without turning the shoulders (c).

head and shoulders are not always turned, it is unnecessary for small eye movements as shown in figure 4.9(a). There are two threshold gaze angles for moving the head, the threshold for horizontal and vertical angles is different. If the gaze angle is within the threshold for the current head position the actor will not move its head. Also the actor will tend to return its head to the central, forward facing position, so if the gaze angle is within the threshold for the central position the actor's head will moving back to the central position. There are different, greater threshold angles for moving the shoulders. These thresholds can vary between actors. The use of threasholds is supported Sparks' obsevations[Spa89]. However, the threasholds themselves were left variable, rather than using experimental data to determine them, so as to leave more flexibility to the animator.

The head is moved by rotating it so as to point its local forward (Z) axis towards the target. The shoulders are turned by half that amount, so that they are angled half way between the forward direction and the target. This half shoulder turn produces a more natural result than either no turn or a full turn (see figure 4.10(c)).

Tracking Objects/Locations

Finally if the object being looked at or the actor is moving the actor's gaze must follow the object or location. This is done by updating the eye's fixation point every frame. If the actor's head is already moving or if the angle of gaze exceeds the threshold while it updates the head's rotation is also updated.

4.4.5 Parameters of the attention manager

There are a number of fundamental parameters that control the outward appearance of the behaviour produced by the attention manager. These are described here. They can be altered from actor to actor to alter the characteristics of the actor, the next chapter gives a description of their use.

LookForward is the probability of the actor looking forward rather than around itself. This is used in determining what sort of undirected attention behaviour is produced.

preferred gaze angle is the preferred angle of the actor's gaze to the horizontal (as shown in figure 4.8). Another parameter *maintain gaze angle* is the probability that on changing direction of gaze the actor will still keep the preferred angle of gaze to the horizontal. This is used both in altering existing requests or generating new requests during undirected looking.

glance at centre/surroundings: some people tend to look forward for long periods while occasionally glancing (i.e. looking for a short time) at their surroundings, others do the opposite. These two parameters give the probability that the actor will glance in either the forward direction or at the surroundings instead of giving a long look.

glance/gaze length and variance: these are the length and variance of the time that the actor maintains an object of attention. Clearly they are different for a short glance and a long gaze.

head rotation speed is the angular velocity at which the head rotates when moving to look at something.

head/shoulder rotation thresholds are the threshold angles about the Y and X axis of the head for the head and shoulders to rotate to look at an object rather than just the eyes.

keep head/shoulders still these flags tell the attention manager that it cannot move the actors head or shoulders. This could be because the actor is performing some action that requires the head or torso to move in a certain way and so a head or shoulder movement caused by the attention manager would interfere with it. If these flag is set the actor will only move its head or only move its eyes. It will not be possible to look at targets outside of a certain range and so these should be rejected.

4.4.6 Peripheral vision and attention capture

While the focus of attention is directed to a single location at a given time the actor is also aware of events in the periphery of vision. This is defined as being anything within 90° of the centre of vision. Though in general objects in the periphery are ignored, relevant objects can capture the attention of the actor. Unlike Chopra-Khullar and Badler's system where attention is only captured by moving objects there are a range of possible relevant or interesting objects that can capture attention. These can vary from task to task. For example, in the task of walking in a cluttered environment relevant objects are considered to be those which are moving and those which are in path of the actor. These are the objects with which there might be a collision.

Various agents perform the *attention capture* by *peripheral vision* depending on what is capturing the attention, for example, agents of the walking group perform the attention capture in the cases described above. They scan objects in peripheral vision and check if they are relevant to the actor's current action. For example, for navigating an environment relevant object are those that there might be a collision with, objects in front of the actor or moving objects. For a competence the relevant objects are those that the competence can be performed on, as defined by an object feature (see sections 3.4.3 and 5.6). If there is such an object a shift of attention might be requested from the attention manager. Some objects need to be reacted to quickly as they are imminently approaching the actor. These objects are automatically passed to the attention manager as immediate requests. Physical distance cannot be used to determine which objects need to be passed immediately, a close object might be moving away from the actor or both the actor and object might not be moving in which case there is no immediate hurry. Time-to-contact (see section 3.5.1) is used instead as it determines whether the object is moving towards the actor rapidly, which is more important than simply being close to the actor. If the TTC is not low the peripheral agent will either, with a set probability, send a monitor request or do nothing but add the object to the list of objects the actor is aware of. If there are two possible targets the one with the lower TTC takes precedence. These agents, together with the undirected attention, ensure that the actor is aware of its surroundings.

Peripheral vision also determines the object which the actor is aware of for the purpose of undirected attention. As explained above objects that pass one of the peripheral vision tests are either added to a list of objects that the actor is aware of or they are passed to the attention manager as attention requests (in which case the attention manager adds them to the list). Objects for undirected attention are then chosen from this list, thus the actor might look at them without them being directly relevant to the current task. The objects stay on this list until they pass behind the actor and so become more difficult to look at.

4.5 Evaluation

It is difficult to evaluate the system in a vacuum as the attention behaviour produced is designed to be dependent of what actions the actor is performing. Thus the attention behaviour will depend critically on what requests the other agents are sending. Also its believability will depend on the context in which it is happening.

The attention system described is not designed just to add attention behaviour to existing behavioural algorithms. It can be used to design algorithms that depend on the actor's attention and react to the actor's surroundings in an appropriate way.

Both these reasons mean that it is not possible to evaluate the attention mechanism without using it in behaviour patterns. The next two chapters give such examples. The next chapter describes a general method for creating behavioural competences which depend on attention. Chapter 6 shows how attention can be applied to a common task in behavioural animation, walking around an environment without colliding with obstacles.

However, we can give an initial example of undirected attention. Figure 4.11 shows the actor walking between two rows of columns (this is not really an example of navigation as the actor only has to keep a straight path). The gold pillar is classed as more interesting by its object features, and the actor looks at it in frame 7. The actor's parameters are set with a high gaze direction and a high probability of looking around itself. The actor tends to glance in front of itself and gaze at the surroundings. Figure 4.12 show close up views of the actor's head in this example.

Figure 4.13 is the same example but the opposite parameter settings. The actor mostly looks in front of itself with a low gaze angle. It only glances at the surroundings but gives long gazes at the floor. The gaze behaviour is less varied than in figure 4.11 (and as such close up views would not show much of interest and thus are not included).

4.6 Conclusion

This chapter has presented a general system for producing attention behaviour in a virtual actor. This system has a number of features:

- A variety of different types of attention behaviour are available for different contexts. Immediate shifts, monitoring and searching fulfil the requirements of different contexts. Undirected attention is for the context where no specific attention behaviour is required. However, it is still adaptable to different contexts and characters by changing the default direction, the probability of looking around, or the interest values of the various objects in the scene.

- The actor's attention behaviour is dependent on what other actions the actor is performing. The agents that are controlling the other actions can request different types of attention behaviour. They can use different ways of specifying a target, as an object, a location or a direction. There are also a number of parameters that the agents can use to control how the attention behaviour is performed.

- The user or other agents can control the way in which the actor performs its attention behaviour as a whole using various parameters. These can change the character of the behaviour.

- The attention mechanism can be used by other agents to determine what the actor is attending to or aware of. This can be used to create behaviours that depend on attention.

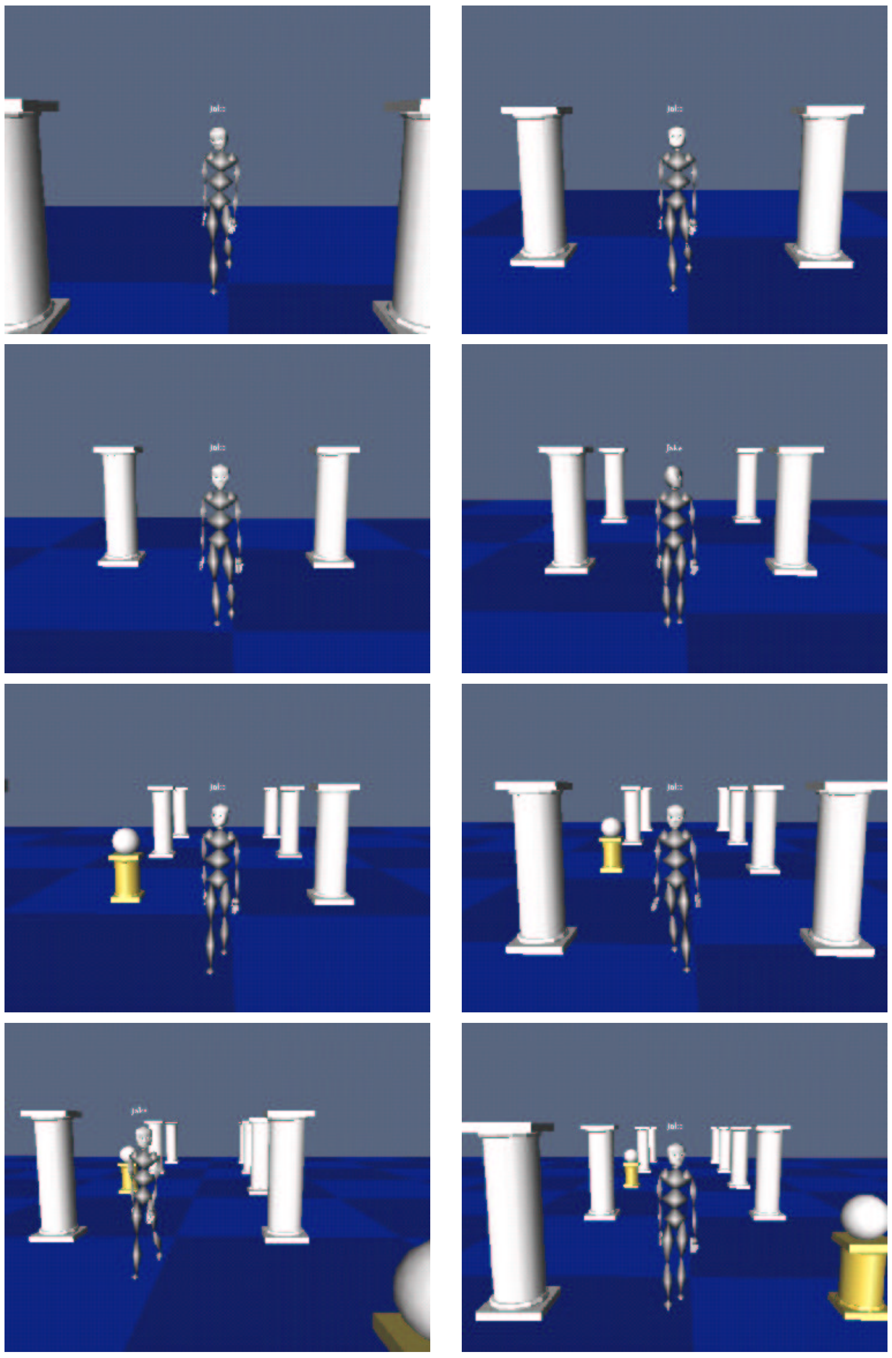


Figure 4.11: The actor walking between two rows of columns, demonstrating undirected attention. The gold column with a sphere on top is classed as more interesting by its object features. The parameters of the actor are such that its looks around itself more than looking forwards and has a high gaze angle.(left to right, top to bottom)

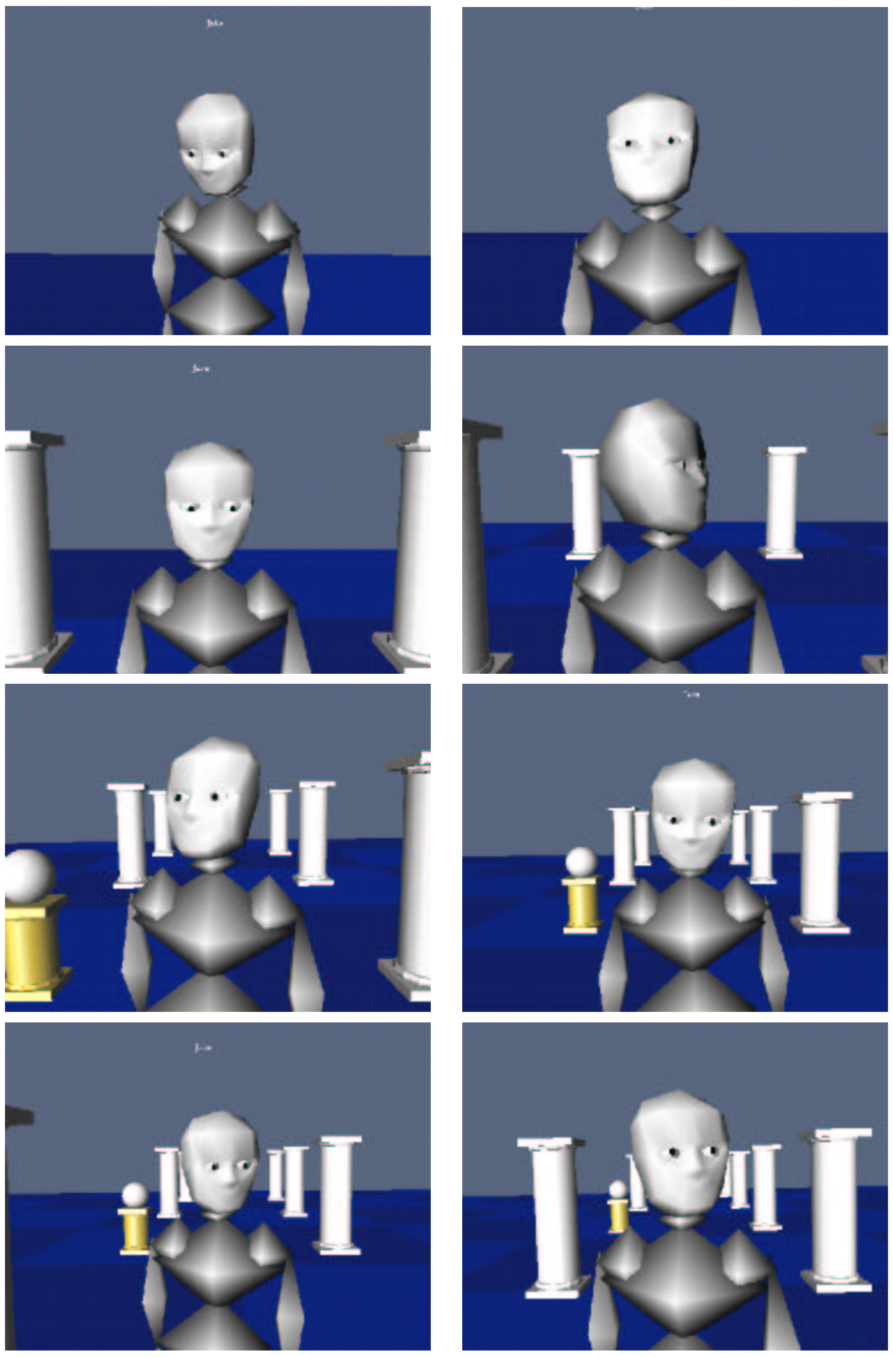


Figure 4.12: A close up showing the eye movements from figure 4.11(left to right, top to bottom)

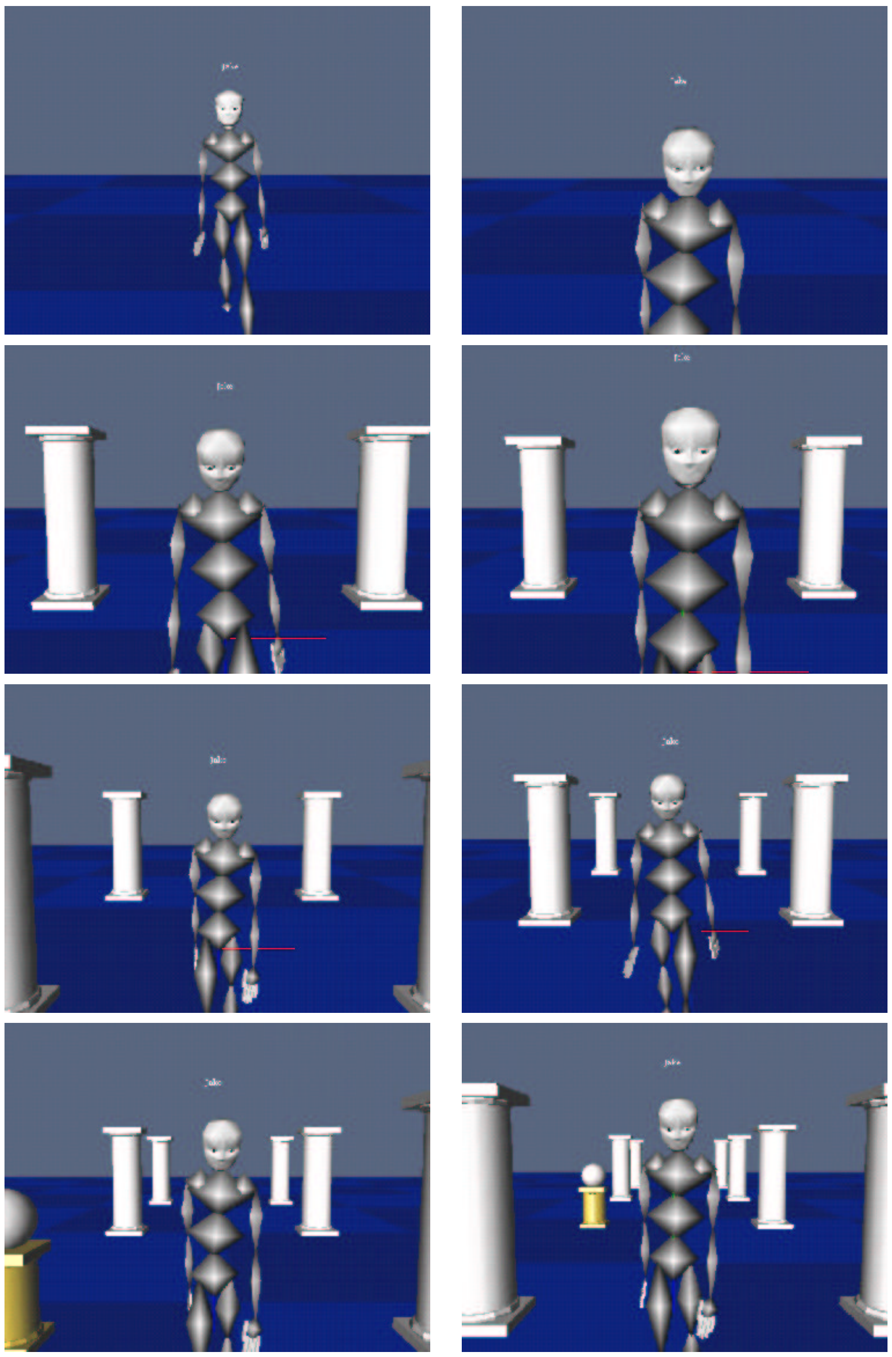


Figure 4.13: The actor walking in the same environment as figure 4.11 but with different parameter settings: a large tendency to look forwards with a low gaze angle. The character's gaze only occasionally moves from the ground in front of it. Its gaze raises slightly in frame 4 and looks slightly to one side in frame 7. Frame 2 is interesting. The actor looks up without raising its head.(left to right, top to bottom)

Chapter 5

Designing Behavioural Competences

This chapter introduces a method for creating simple behaviour patterns which involve attention behaviour, body motion and manipulation of objects. This method has been implemented as a tool which could be used by a person with some knowledge of computing and animation but without any programming experience. It aims to produce the sort of simple, flexible competences described in section 2.4.1. The *behavioural competences* are of a sort that can be invoked simply by the user through a mouse or menu based interaction. The motion of the character is based on a piece of captured motion of a person performing the action on a particular object or location. This is then adapted to a different object or location. The competence uses the attention mechanisms described in the last chapter to produce behaviour. Behaviour and attention are closely linked for two reasons. Firstly, a certain behaviour pattern is associated with the actor attending to certain places and so the competences should use the Attention Manager to generate appropriate attention behaviour. The type of eye movement behaviour can be designed so as to be appropriate to the action and it can also be adapted for different characters. This works as described in section 5.4.1. As it works with the attention manager described in the previous chapter, the competence can interact properly with other events that take up the actor's attention. The other reason why behaviour and attention are closely coupled is that what the actor is aware of affects what the actor does. Section 5.6 described how objects that the actor is aware of affect what actions the actor performs. The Competence can then be invoked by the user in a number of ways.

5.1 Previous Work

Though most work on behaviour has focused designing the actor's behaviour using normal programming techniques there has been some work on design of behavioural competences for non-programmers. The focus of this work has generally been different from the work described in this chapter. The essential problem that is being addressed is that designing a behaviour is a complex task and yet it needs to be presented to the user in a way that is comprehensible.

Some work has been done on using simplified languages for behaviour design. Funge, Tu and Terzopoulos[FTT99] propose using the *Cognitive Modelling Language (CML)* for specifying complex cognitive behaviour. This is a logical language for defining cognitive rules for actions. Though this language is well suited to the task of defining behaviour and so simplifies it, it does not change the fact that it is essentially a programming task. Lethbridge and Ware[LW89] have a language based on if-then rules acting on a number of primitive quantities, such as velocity, distance (to an object) or relative velocity of the actor and another object. This does simplify the task but is still a linguistic technique for specifying what is a spatio-temporal task. Also the range of behaviour is limited. Haumann and Parent[HP88] propose building behaviours out of simple "black-box" atomic behaviours. Again, this is still a programming task, in this case using LISP.

Wilhelms and Skinner[WS90] use a method similar to visual programming to produce behaviours. Behaviours are built out of three types of component: sensors, which sense the environment, effectors which act on the environment (in this case they only act as motors, altering the actor's velocity) and nodes which perform some transformation on their inputs. These are connected together using a graphical interface in order to produce more complex behaviours.

Scerri and Reed[SR99] use a graphical editor for designing finite state machines and a spreadsheet style editor for functions between sensor input and effector output. Ydrén and Scerri[YS99] have a system for designing strategies for players in a robot soccer competition. They use a map of the playing field onto which arrows, representing players' movement or passes, are placed. This is designed to be similar to the way real soccer coaches design strategies. These specify general actions that are then compiled into an agent architecture. These actions are higher level than the competences described here and a similar method could be used for designing the circumstances under which competences could be invoked.

In the above methods designing behaviour as a simplified programming task. Del Bimbo and Vicario[DBV95] have a different approach. They attempt to allow the user to specify a general behaviour by giving a particular example: specification-by-example. Behaviour is defined in terms of *extended spatio-temporal logic (XSTL)*, a language that specifies spatio-temporal relationships in vague, qualitative ways. This means that any definition is necessarily general. The specification occurs by the user controlling the actor and having them perform an example of the behaviour. This is then parsed into XSTL which immediately gives a general definition of the behaviour. The problem with this approach is that the mapping between the animator's actions in defining the behaviour and the resulting behaviour is abstracted away and so is not accessible. This makes it rather difficult for the animator to know what the effect of his or her actions will be. To get exact knowledge would mean manipulating the underlying representation which is expressed in a non-standard logic and so would not be readily accessible to an animator.

Yoon, Burke, Blumberg and Schneider[YBBS00] use a training system for actors that is analagous to training animals. They use reward and punishment to make the actor learn a task by forming links between rewarding stimuli and basic behaviour patterns.

The focus of all this work is rather different from mine. They all use techniques of end-user programming to make the programming task more accessible to the user, however, end-user programming has had limited success owing to the innate complexity of programming which is inherently inaccessible to untrained users. My work attempts to simplify the task of designing a behaviour so that it is no longer a programming task. This means that the behaviours produced will be less complex but that the tool should be more generally accessible. The limit on the type of behaviour is not a great disadvantage as it seems that all these techniques have limits. They mostly focus on simple stimulus response behaviour and it is not clear whether the methods scale to more complex behaviours. Also they all build behaviour out of basic primitives and so limit the range of possible behaviours. The current work builds behavioural competences from pieces of motion, which can include any human action. The current work also attempts to build the sort of behaviour that could be conveniently invoked by an animator. The simplicity has an advantage that the behaviours are more predictable and so less unlikely to have undesired results.

Another difference between the current work and previous work is that previous work abstracts away the low-level details of the actor's motion, typically dealing only in overall position and velocity. The current work focuses on the details of the motion, allowing the user to produce the exact motion desired.

5.2 Structure of a competence

A competence is a generic action that can be created by the user and then invoked in a variety of situations. It is represented as an agent in the actor architecture (see appendix A). The action itself will normally be fairly simple and normally involve manipulation of one or two objects. A Competence is based around a pre-existing piece of motion, which could be hand animated, motion captured or created using some sort of motion warping scheme[PW99, BH00, Pol00]. This piece of motion will typically be of an actor performing an instance of an action. The competence is a generalisation of this action. The competence then has the function of turning this piece of motion into something that can be applied in many situations, has appropriate attention behaviour and is able to interact with objects in the environment.

Firstly a number of *contacts* are defined by the user. These are where the actor touches some object or location in the world. These are needed as it is important to be able to change the positions of objects with which the actor interacts while using the same piece of motion. An example of a contact would be the door handle in a door opening action. The contacts are used to alter the motion of the competence to retarget it to a new object, for creating eye movements and for allowing the actor to move objects in the world. A contact has two parts, a *contact argument* which is the object or location with which the actor makes contact and the contact point which is the time in the motion where the actor makes contact with the argument, together with the end effector that makes contact. A single contact argument can be used by many contact points. *Contact arguments* can

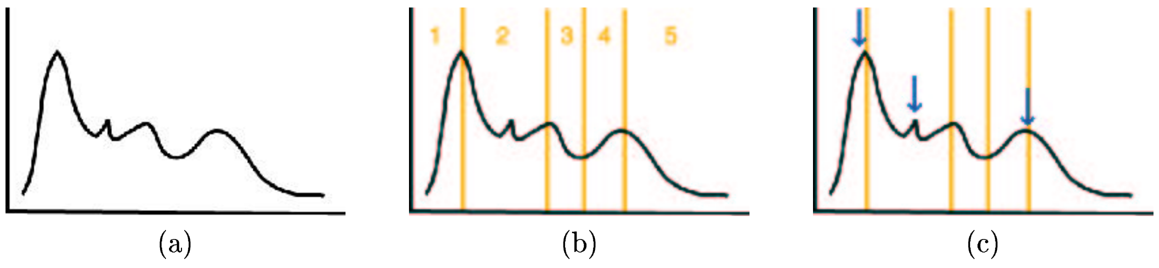


Figure 5.1: Competences tag the piece of motion on which they are based (here represented as a single curve(a)) with various important points. Firstly the motion is divided into various periods (b) which represent important sub-motions within the motion. Then various contact points are added (c). These are points in the motion where end effectors come into contact with objects or locations in the world.

also be added that do not represent a physical interaction with an object. These contact arguments do not have contact points and represent some sort of non physical interaction, such as a conversation. This sort of contact will be called a secondary contact. Contacts are discussed more fully in the next section.

As well as a piece of motion and a list of contacts, the competence contains a number of *secondary actions* that add extra features to the competence. The *secondary actions* currently specified are *EyeAction* which adds looking behaviour to the competence and *HandleAction* which enables the actor to move objects around. As the secondary behaviour may be different in different sections of the action, for example, the actor might look at a cup while picking it up but not while drinking from it. To achieve this, the designer divides the motion, shown as a single curve in figure 5.1, into *periods* which represent some self contained sub-motion. For example, for the motion of picking up a glass and drinking, the periods might be: grasping the glass, lifting the glass to the lips, drinking and putting the glass down. These periods are defined by their name and start time and they run until the start of the next period. Each Secondary action has a number of parameters that control how it works; for example, in *EyeAction* these include whether the actor looks at the position of a constraint point and for how long. Different periods can have different values for each parameter. This means that, for example, the actor can be made to look at the glass while it is grasping and lifting it, but not while it is drinking or putting the glass down. The full function of secondary action is described in section 5.4.

Figure 5.2 shows the structure of the competence. It contains a piece of motion, a list of constraint points, periods and secondary actions. Each period contains its own value for each parameter of each secondary action.

5.3 Contacts

A contact represents an object or location which had an important function in the original motion on which the action is based. When the competence is invoked it is replaced with

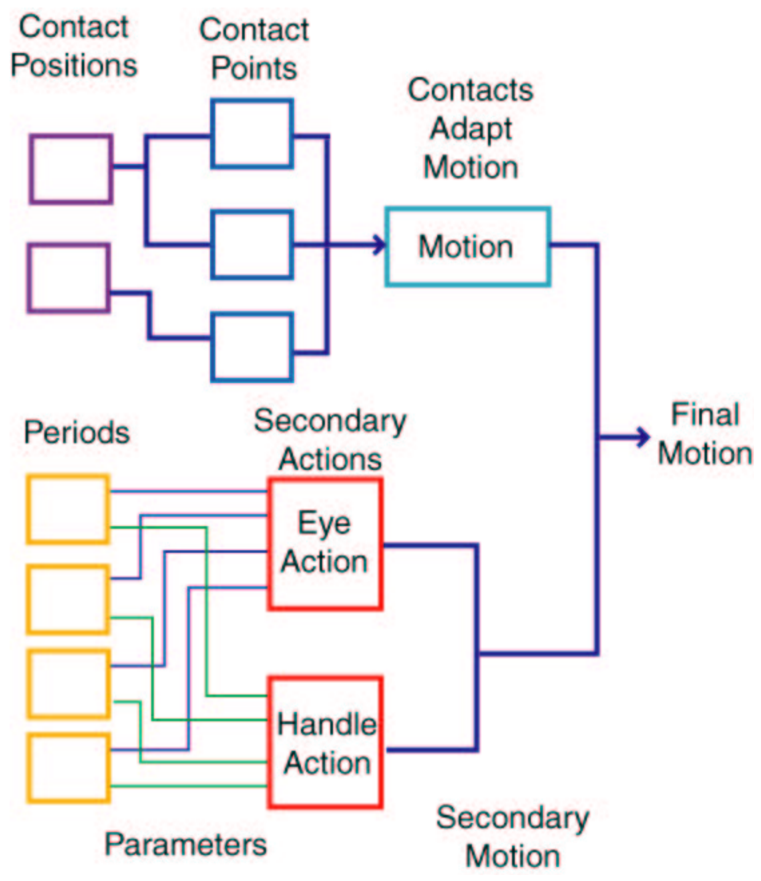


Figure 5.2: The structure of a Competence. Each competence is based on a piece of Motion. There are a number of Contact Positions with which are various positions which the actor interacts while performing the competence. These can be instantiated with object or location when the user invokes the competence. There are also Contact Points which are points in the motion where the actor touches the contact positions. These are used to adapt the motion to the new contact positions. The Motion is divided into a number of periods. There are also a number of secondary actions which add other secondary motions to the final motion. The secondary actions have parameters. There are different values for each parameter stored in each period. At the start of each period these values are loaded into the secondary actions to change what they do.

a new object or location. For example the contact argument could be an object that is manipulated by the competence or somewhere that the actor looks while performing the competence. There are two main uses of contacts. If they have a contact point specified they will affect the actual motion of the behaviour. These contact positions are the objects that the actors are manipulating or the locations they are touching. Also contacts can affect the secondary actions, for example by the actor looking at the argument. If a contact argument does not have any contact points it will not affect the motion but just the secondary actions associated with the competence. For example, while drinking a cup of coffee the actor will look at the cup, but if the actor is with a friend it will be likely to look at the friend more than at the cup. The friend could be specified as a secondary contact.

5.3.1 Adapting the motion

Contact points are specified by the end effector that is affected by the contact, the time that the contact takes place and an offset of the end effector from the argument. The contact argument gives a position for the contact in the original motion. When the competence is invoked a new position is given for the contact and the motion is adapted so that end effector is now the appropriate offset away from the new position at the right time. Adapting motion has been done in a number of ways [GL96, LS99]. Here I use Polichroniadis'[Pol00] method of transforming the positions and orientations of the end effectors of the motion.

This adaption is achieved by merging the position of the new contact argument into the old motion by adding a fraction of the displacement between the original position of the contact argument and the new position onto the current position of the end effector at each frame. Thus the end effector position is transformed by:

$$\mathbf{p}'_{\mathbf{ee}}(t) = \mathbf{p}_{\mathbf{ee}}(t) + \mathbf{F}(\mathbf{c}_{\mathbf{new}} - \mathbf{c}_{\mathbf{old}})$$

where $\mathbf{p}_{\mathbf{ee}}(t)$ is the position of the end effector at frame t, $\mathbf{c}_{\mathbf{old}}$ is the original contact position and $\mathbf{c}_{\mathbf{new}}$ is the new one. $\mathbf{F}$ is some suitable blending function. The problem is to find a suitable blending function to use. The one answer would be to use a function of the time between the current frame and the frame on which the contact occurs[1]. So the transformation now looks like:

$$\mathbf{p}'_{\mathbf{ee}}(t) = \mathbf{p}_{\mathbf{ee}}(t) + (\mathbf{c}_{\mathbf{new}} - \mathbf{c}_{\mathbf{old}})f(t - t_c)$$

where t_c is the time at which the contact occurs. However, this is not very suitable as some actions involve spending a long time near the contact position, for example, typing at a keyboard. However, a swift punching action might involve spending very little time near the target of the punch. It is not possible to choose a general drop-off with time that works in all cases. However, in the original typing motion the hands will remain near the keyboard for a long time while in the punch they will move quickly away and a

[1]This method was suggested to me by Tony Polichroniadis

transformed motion should stay near the contact when the original one is near the contact. Therefore it would be better to base the blend on the distance of the untransformed end effector position from the original contact position:

$$\mathbf{p}'_{\mathbf{ee}}(t) = \mathbf{p}_{\mathbf{ee}}(t) + (\mathbf{c}_{\mathbf{new}} - \mathbf{c}_{\mathbf{old}})f(|\mathbf{p}_{\mathbf{ee}}(t) + \mathbf{p}_{\mathbf{off}} - \mathbf{c}_{\mathbf{old}}|)$$

where $\mathbf{p}_{\mathbf{off}}$ is the offset of the contact. The only question left to answer is what sort of function f should be. In the current implementation the keyframe interpolation system guarantees smooth motion which means a very smooth function was not necessary so a linear fall off was used for speed. In a different system it might be necessary to use a smoother fall off, Gaussian or $\frac{1}{1-x^2}$ for example.

The use of spatial distance has a side effect that can be both desirable and undesirable in different situations. If the end effector comes near the original position of the contact argument at another time in the original motion other than the time of the contact it will be affected by the contact. This is normally a desirable property as contacts represent locations or objects that have some meaning and keep this meaning over time. So if the location is moved, it should be moved throughout the competence. For example, if the action consists of the actor picking up an object, manipulating it and returning it to its original position, it should move its hand to the position of the object both times. However, if this feature has undesirable consequences it can be eliminated by finding when the end effector enters the influence of the contact before the contact time and when it leaves afterwards. The contact can then be applied only between these times.

This is how the motion associated with the action is adapted by a contact. The other important function of a contact, and the only function of a secondary contact, is to influence the secondary actions that are associated with the contact. This does not depend on a particular end effector or at a particular time so there is no need for an argument to a secondary contact. How the contacts affect the secondary actions is described in the next section.

In order for the contact to affect the competence it needs to be supplied with a new position for each argument when the competence is invoked. This can simply be done by specifying a position vector. However, the contact becomes more powerful if instead of just a position, an object is specified. The contact can take either a position or object as an argument. If the argument is an object the competence stores the object itself rather than its position and so can act on its changing position rather than just its location. It can also affect properties of the object such as position and orientation. This becomes very important for secondary actions as described in the next section.

5.4 Secondary actions

While contacts adapt the behaviour displayed in the original motion of the competence to a new situation, secondary actions add in other aspects of the actor's behaviour that possibly were not present in the original motion but are important for a believable behaviour. The two types described are *EyeAction* which supplies eye movements and attention behaviour

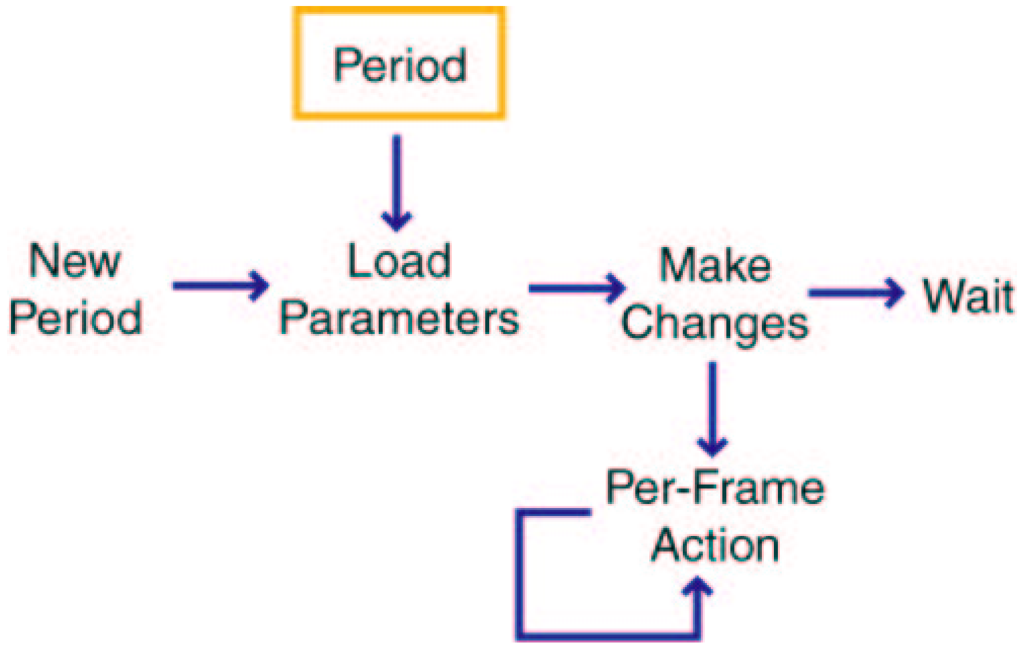


Figure 5.3: The secondary action loop.

and *HandleAction* which enables the competence to affect other objects in the world. Before the two types of action can be described, the general structure of secondary actions must be explained.

Each secondary action has a number of basic parameters that affect its behaviour. These can apply to all secondary actions of a particular type or there might be a separate parameter with a different value for each contact argument that affects the secondary action. For example, there might be a high probability of looking at one contact argument and a low probability of looking at another. It is in the use of these parameters that the periods of the competence become important as the value of each parameter varies from period to period. This is important as it enables the user or designer to make the actor behave differently in different sections of the action. For example, an actor might look at a keyboard frequently while typing at it but not while resting. How parameters may be set by the user or designer is described in section 5.7.

The values of the parameters are stored in the periods themselves. At the start of each period the appropriate parameter values from the period are loaded into each of the secondary actions. Then a method of the secondary action is called to perform the necessary actions. In general the secondary actions are only invoked at the start of periods but sometimes is is necessary to perform some action every frame. If this is necessary a flag can be set so that the competence calls a method of the secondary action every frame. This is used by *HandleAction* but not *EyeAction*. The workings of the secondary actions are shown in figure 5.3.

5.4.1 EyeActions

EyeAction uses the attention architecture described in the previous chapter to create the actor's eye movements while it performs the competence. The contacts of the competence provide a location or object to look at while the different periods provide times in which different behaviours can occur. For example, if the behaviour consists of drinking a cup of coffee the actor might look at the cup during the periods representing moving the hands to the cup and grasping it but not while actually drinking. Secondary contacts are important here as they provide places to look other than those the actor is physically acting on. In the example of drinking coffee if the actor is having coffee with a friend a secondary contact can make the actor attend to the friend as well as the coffee cup.

At the start of each period the *EyeAction* will create some attention requests which will be sent to the attention manager. The details of these requests depend on the settings for the parameters in the current period. The EyeAction has a choice of looking at any of the contact positions of the competence. Each contact position has a probability, which is a parameter, that the actor will look at them immediately on starting the period. This probability is tested for each contact in the order set at the time of creation of the competence. When one succeeds the action will issue an immediate request for the position of that contact and then stops testing. There is also a probability that each contact point will start or stop monitoring the contact argument in that period. In this case a monitor or purge request is issued.

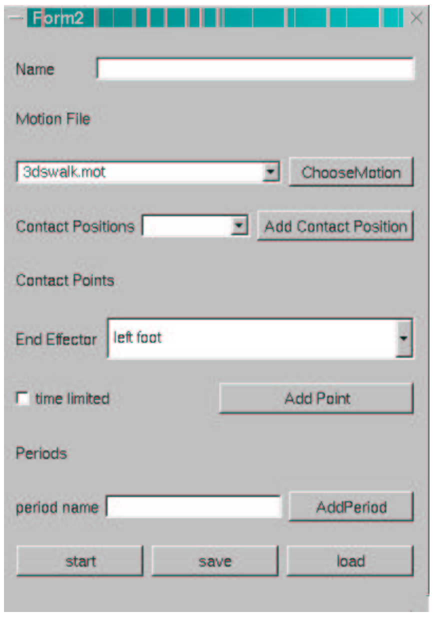
The remaining parameters of the *EyeAction* affect the details of the attention behaviour. For example, they can affect whether the actor will perform a long glance or a short gaze. These parameters correspond the the parameters of *EyeRequests* described in section 4.4.2.

Once the various parameters of the request have been set it can be issued. If the contact argument is simply a location, a location request will be issued. When the contact argument is an object, an object request will be issued so that the actor's gaze will follow the moving position of the object.

5.4.2 Handle Action

The *HandleAction* enables the competence to move objects in the environment. It acts on objects provided as the arguments of contact positions. If the argument is a location the HandleAction will have no effect. If the argument is an object it makes the object follow the associated end effector during some periods and not others. For example, in the coffee drinking example the cup will move with the hand from the time the actor grasps the cup to the time it puts the cup down but not outside this time.

At the start of each period the action tests to see if the object has to follow the end effector. This is set as a parameter of the secondary action. This means that the designer of the competence can explicitly state during which periods the object should follow the actor's end effector. If the object should follow the end effector a flag is set which makes the competence activate the action every frame in order to move the object. The positions and orientations of the end effector and object at the start of the period are stored. At



Motion File: 3dswalk.mot

End Effector: left foot

Figure 5.4: The dialog window for creating competences

each subsequent frame the differences of position and orientation of the end effector from its original state are calculated and then added to the original state of the object to get its new position:

$$\mathbf{obj_{new}} = \mathbf{obj_{old}} + (\mathbf{ee_{new}} - \mathbf{ee_{old}})$$

Where $\mathbf{obj_{new}}$ and $\mathbf{obj_{old}}$ are the original and new positions of the objects and $\mathbf{ee_{old}}$ and $\mathbf{ee_{new}}$ are the positions of the end effectors. The expression for the orientations are equivalent though slightly different as the orientations are defined as quaternions:

$$\mathbf{obj_{new}} = \mathbf{obj_{old}} \times \mathbf{ee_{new}} \times \mathbf{ee_{old}^{-1}}$$

5.5 Creating competences

The aim of competences is to have a mechanism by which a user without programming skill can create a simple behaviour pattern which can do some generic action and which can then be used by a largely unskilled user. This section gives an overview of how a user might create and use a competence. For clarity the more skilled user who creates the competence will be called the designer whereas the user who invokes the competence will be called the user.

Figure 5.4 show the competence creation dialog. This contains the basic functionality for creating a competence.

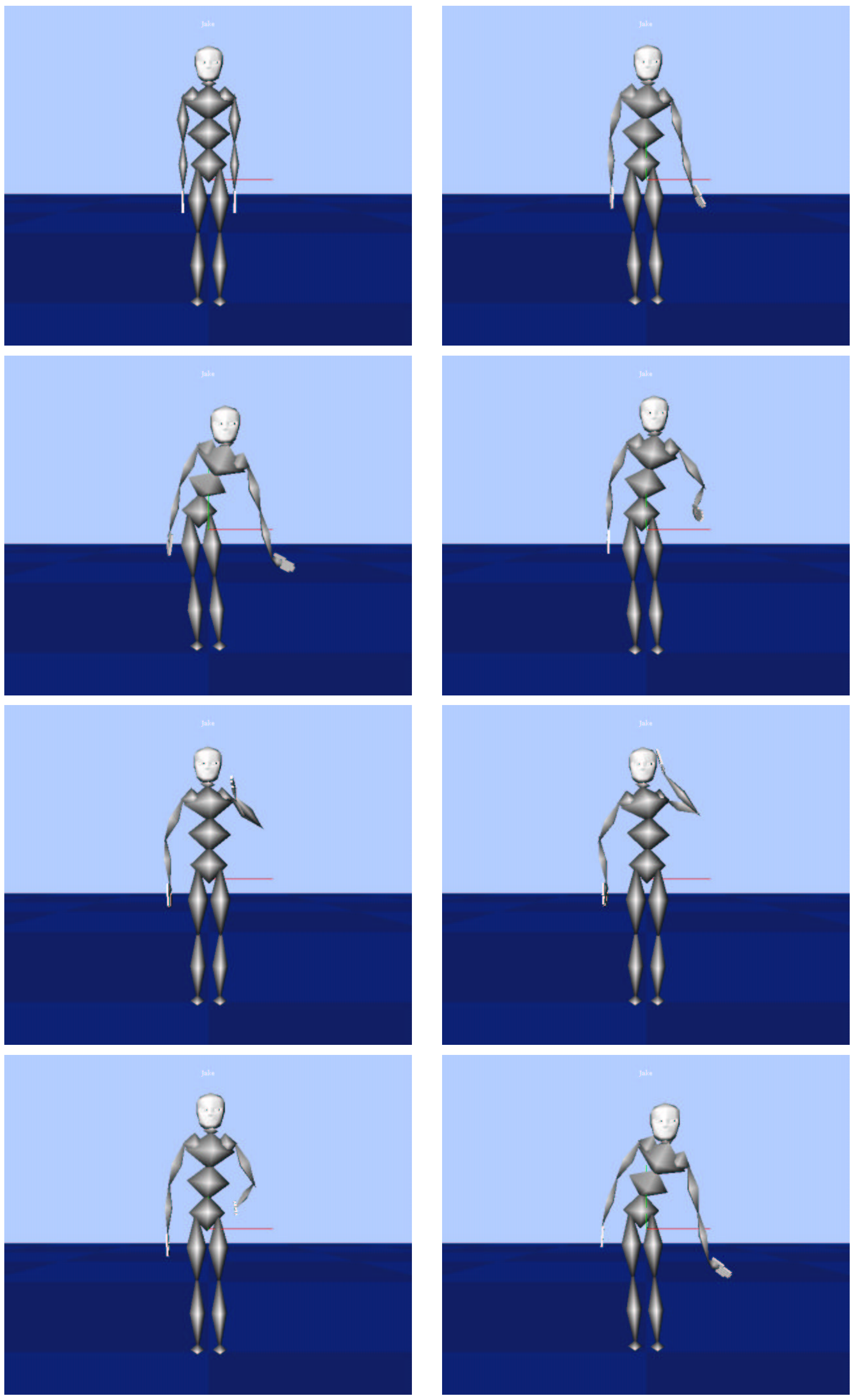


Figure 5.5: A piece of motion of the actor drinking. This is used in the following examples. Unfortunately, as only a primitive motion capture system was available, this motion has a few flaws. In particular the hand does not connect to the face well when drinking.(left to right, top to bottom)

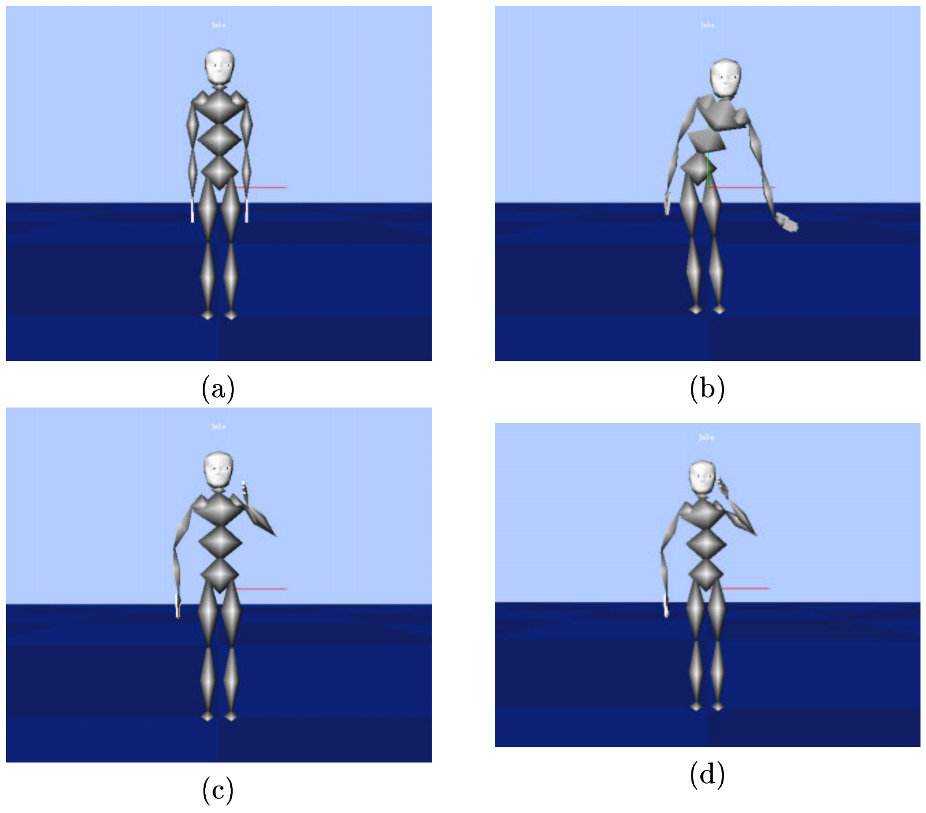


Figure 5.6: The start of various suitable periods in the drinking motion. (a) The start of the motion. (b) as the actor starts to pick up the glass. (c) as the actor starts to drink and (d) as the actor start to put the glass down.

Firstly the designer must select a piece of motion as the basis for the competence, this could be motion captured, hand animated or from some other source. These can be read from files in the Geppetto internal format[2]. The designer can then view the motion in the main window, decide whether it is the desired type of motion and look for times that are suitable as contacts and periods. Figure 5.5 shows the piece of motion used as an example in the following discussion.

The user must then add in the periods and contacts. The periods should be chosen to represent meaningful subsections of the motion such as the phase of lifting the cup to the mouth in a drinking motion. A new period should also start at important moments in the action such as grasping the cup in the drinking motion. The start of periods will normally coincide with contact points. Periods are set simply by moving through the motion to the start point of the period and then giving it a name in the creation dialog. Figure 5.6

[2]These can be created from industry standard files via translation utilities, currently only the Biovision (bvh) file format is supported.

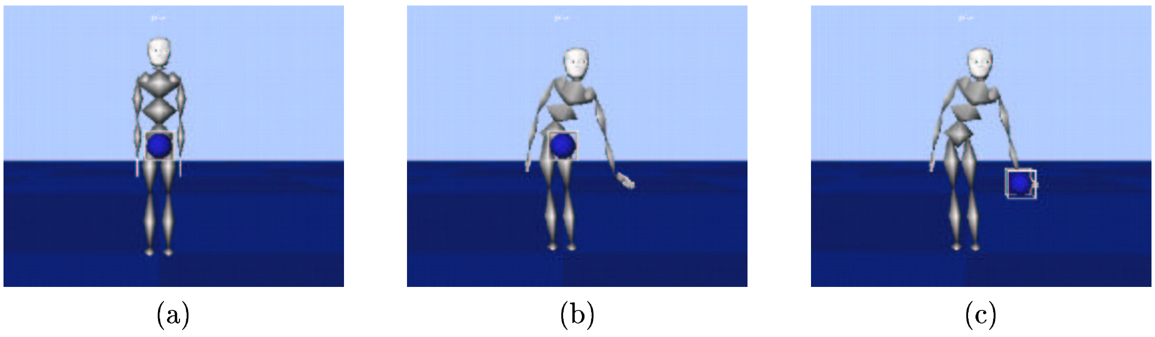


Figure 5.7: When creating a competence a place holder is used to indicate a contact position, the blue sphere. The contact position is set by moving the motion to an appropriate point (when the actor picks up the glass in (b)) and then moving the place holder to the position where the contact should be (c).

shows the start of suitable periods in the drinking motion.

Contact positions are set using a place holder object as shown in figure 5.7. This object is moved to the position that corresponds to the contact position in the original motion. For example, in figure 5.7 the place-holder is moved to the position where the cup is picked up from in the drinking motion.

Contact points are set in a similar way to periods, by moving to the appropriate point in the motion. For example, figure 5.6 (b) shows a suitable contact point for the cup in the drinking motion. For contact points, the end effector has also to be specified; this is done through the competence creation dialog (see figure 5.4). Contact points for different end effectors can be specified for the same frame.

The last task of the designer is to set up the parameters of the competence. This is described more fully in section 5.7. Once all the periods, contacts and parameters are set the competence is ready to be used, or saved to file.

5.6 Invoking competences

In order to invoke a competence the user must choose the competence and then provide arguments for each of the contact positions. The competence is chosen by loading it from a file. The arguments for the contact positions can either be locations represented as a vector or objects whose location is the position of the object. If no argument is provided for the position it will default to the position it takes in the original motion. The arguments are provided in one of two ways. Locations can be provided by bringing up a dialog box from a menu, as shown in figure 5.8(a). The user can choose which contact positions he or she wants to set and then set it by providing a positions vector.

However, there is a simpler way of invoking the competence. If the user click on an object with the mouse a pop-up menu appears (see figure 5.8(b)). This contains the various operations on the object. The user can set various parameters of the object such

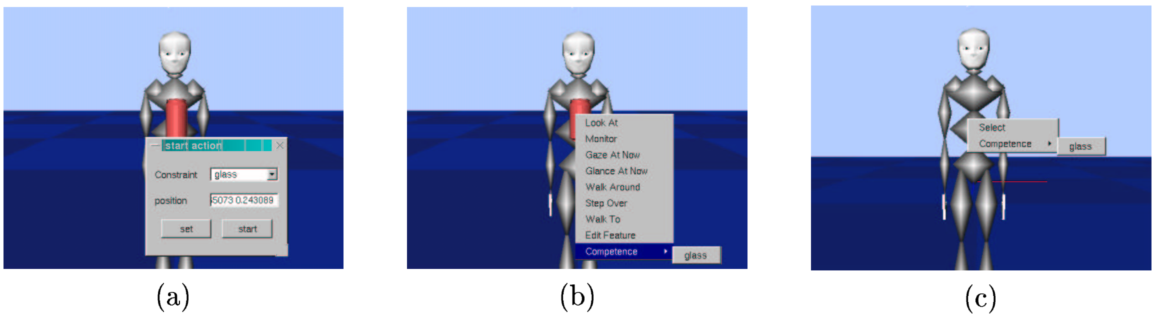


Figure 5.8: The various ways of invoking a competence: (a) through a dialog box by specifying locations for contacts as numerical coordinates; (b) clicking on an object brings up a pop-up menu that includes an option for specifying it as the argument for the contact; (c) a similar menu for a person will not specify the argument for the contact and the actor will either search for one or react to approaching objects.

as how likely it is to gain an actor's attention. There are also various actions that the currently selected actor can perform. Some are attention based such as monitoring the object or glancing at it. Others are specific actions such as walking to the object. Finally, there is an option for invoking the active competence of the currently selected actor. This brings up a submenu with all the contacts of the competence. The user can then select a contact that can use the object as an argument. Selecting an object automatically starts the competence though the user can then select other objects for contact positions before playing the animation. If the object or location is too far away to act on directly, the competence will pass it to the walking agents (as described in the next chapter) as a destination. The actor will then walk to the object and once the actor has arrived the competence will start.

5.6.1 Searching for targets

Thus far the user has specifically provided objects or locations to the actor and has not made use of what the actor is already aware of through the attention mechanism. The user can also have the actor itself find a suitable target for an action. This would be useful for more autonomous actors where the user wants the character to do something but does not want to specify the details. It would be particularly important for repeating actions on a number of different object, for example fruit picking. The user brings up a menu for the actor by clicking on it (figure 5.8(c)) and selects the name of the contact argument but does not specify a value. For each contact argument there is an equivalent *object feature* (see section 3.4.3) with the same name. An object feature value of 1 corresponds to the object being suitable, one of between 0 and 1 corresponds to the object looking like a suitable object but not actually being suitable. If the object feature value is 0 or is not present the object is irrelevant. The actor then checks all the objects it is aware of to see if they have an object feature value of 1. If so the actor will use the first object found

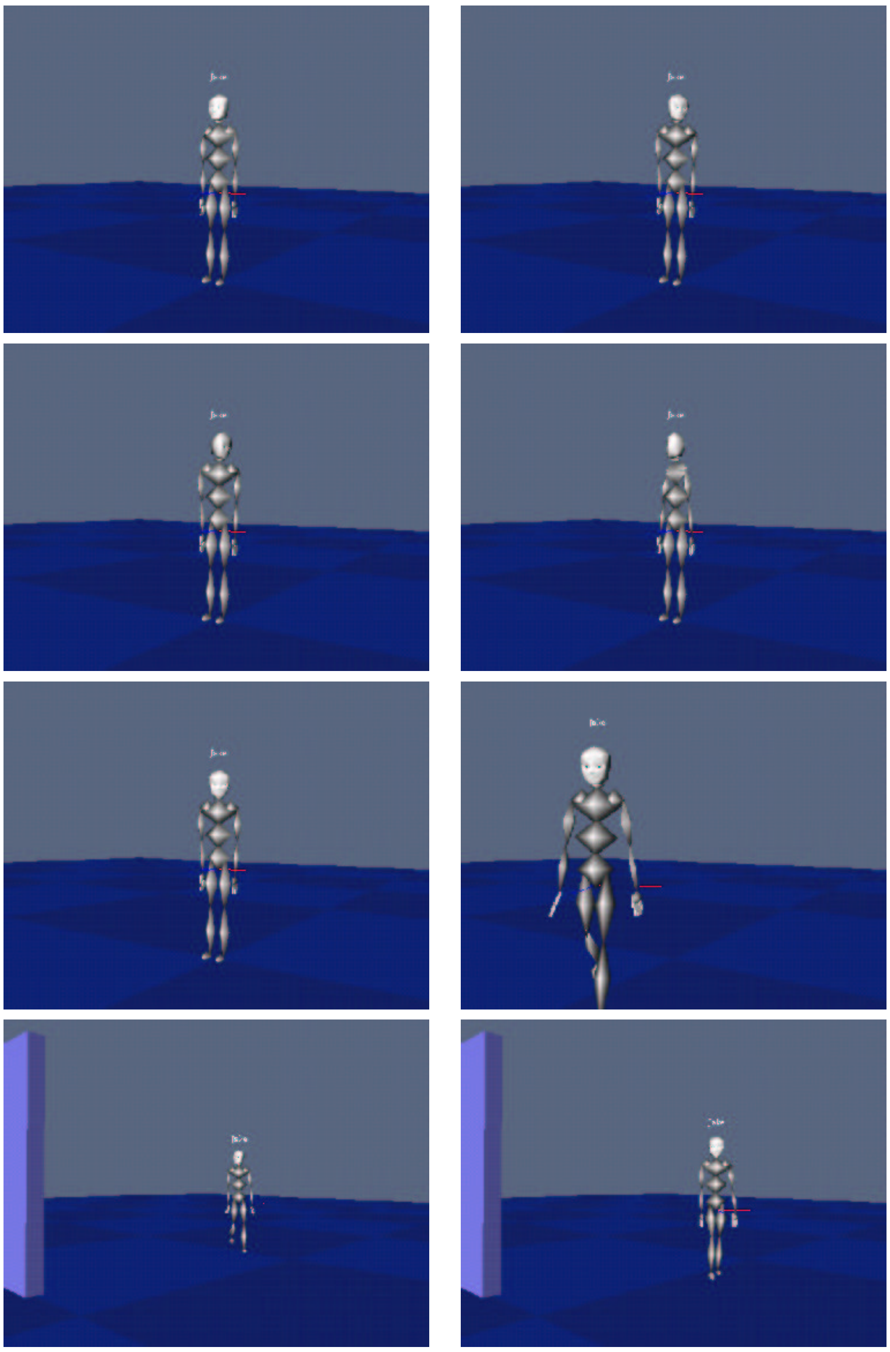


Figure 5.9: The actor searching for a target for the drinking competences. First the actor scans its environment. It does not find the target as it is occluded by the wall so it walks forward. (This scene is very simple, chapter 6 will show the actor walking in more complex scenes).(left to right, top to bottom)

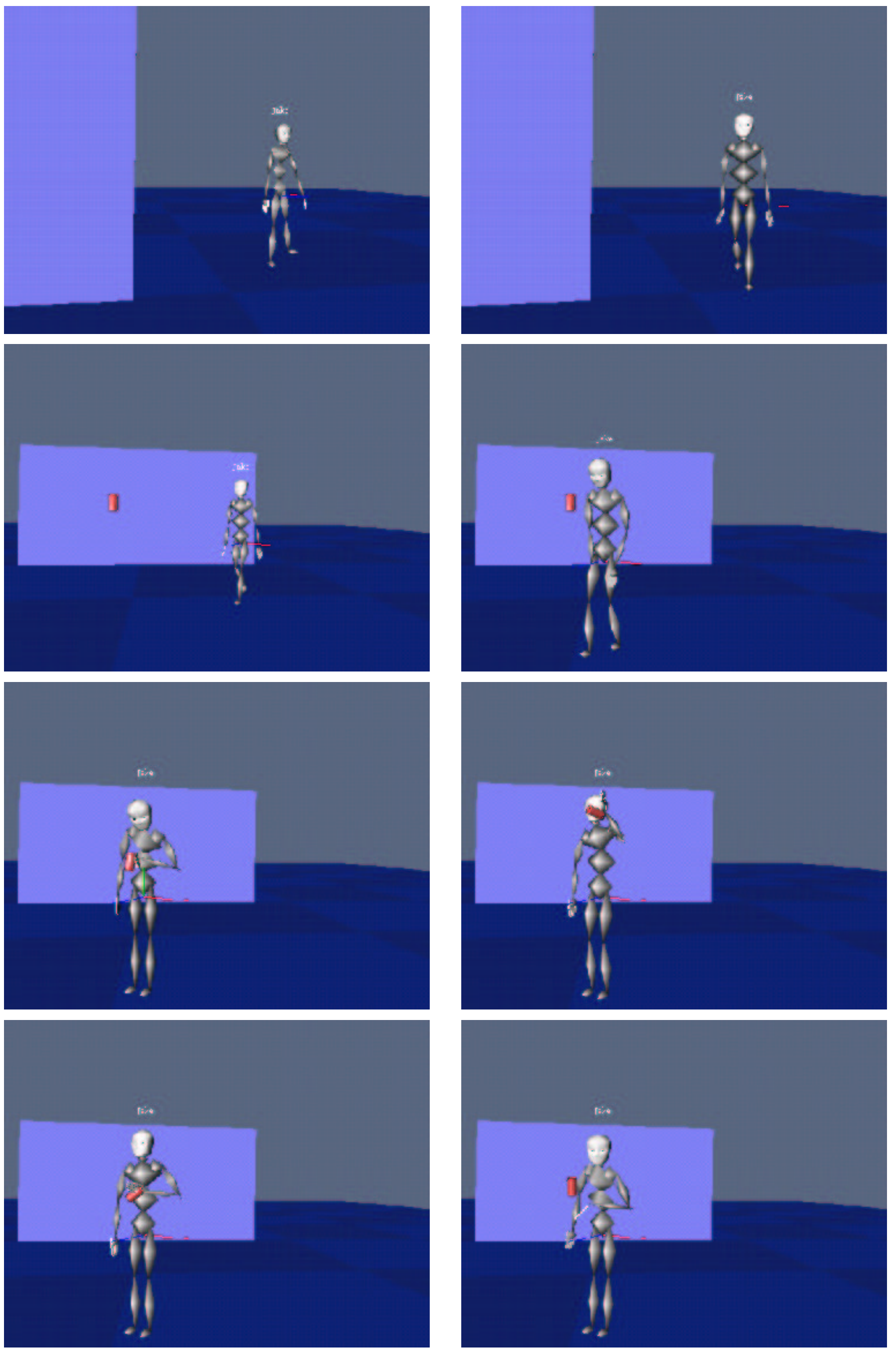


Figure 5.10: A continuation of figure 5.9. As the actor passes the wall it spots the can of drink and walks over to it. On arrival the competence is invoked.(left to right, top to bottom)

as the value of the argument. If none of the objects that the actor is aware of have an object feature value of 1 the actor will then search for objects using that object feature. The searching mechanism is described in section 4.4.3. When the first object is found, the search is stopped and the object is used as the contact argument. Figures 5.9 and 5.10 show an example of the actor searching for a target for a competence.

5.6.2 Reacting to events

The user can also specify that the actor should react to approaching objects by invoking the competence. This could be used for actions in sport such as catching a ball or hitting a volley in tennis. It could also be the basis of social actions such as greetings though, as it stands, it might be too simplistic.

The user requests that the actor should react to an object by invoking the competence using the object as one of the contact arguments. The actor will react to objects which have an object feature value with the same name as the contact argument. This is done by starting a peripheral vision agent that checks for objects with that object feature. If any are detected the attention manager will be sent a request and will look at the object. The collision detector agent (see section 3.5.1) will test if the object is on a collision course with the actor. If so the object is added to a list of potential collisions. The competence should then deal with the objects on this list.

The competence is started when the object is near enough. Bootsma and Oudejans[BO93] have shown that people can judge Time-To-Contact (see section 3.5.1) of object with locations displaced from their heads, for example, with their hands for catching. This means that TTC can be used to judge when an object is close enough. The object is judged to be close enough when the TTC between the object and interception point is equal to the time between the start of the motion and the point when the end effector meets the contact. The interception point is chosen as a point in the direction of approach of the object at a distance from the actor equal to the distance of the contact argument in the original motion.

Figure 5.12 shows the actor catching a ball reactively while Figure 5.11 shows the original motion for this action. Finally, as reactive motions rely on attention the actor has to be attending to the direction of approach of the object in order to react to it. Figure 5.13 shows what happens to the unfortunate actor that was looking in the wrong direction.

5.7 Using parameters

The secondary actions of the competences have various parameters that control the way in which they work. The attention behaviours described in the last chapter also have similar parameters as described in section 4.4.5. These parameters are present in order to make the behaviour created adaptable to reflect different characters. As so much of animation depends vitally on conveying the personality of characters in a piece, it would be undesirable to have every character behave in the same way. Thus, the parameters provide a way for the designer or user to be able to customise the behaviour to different characters.

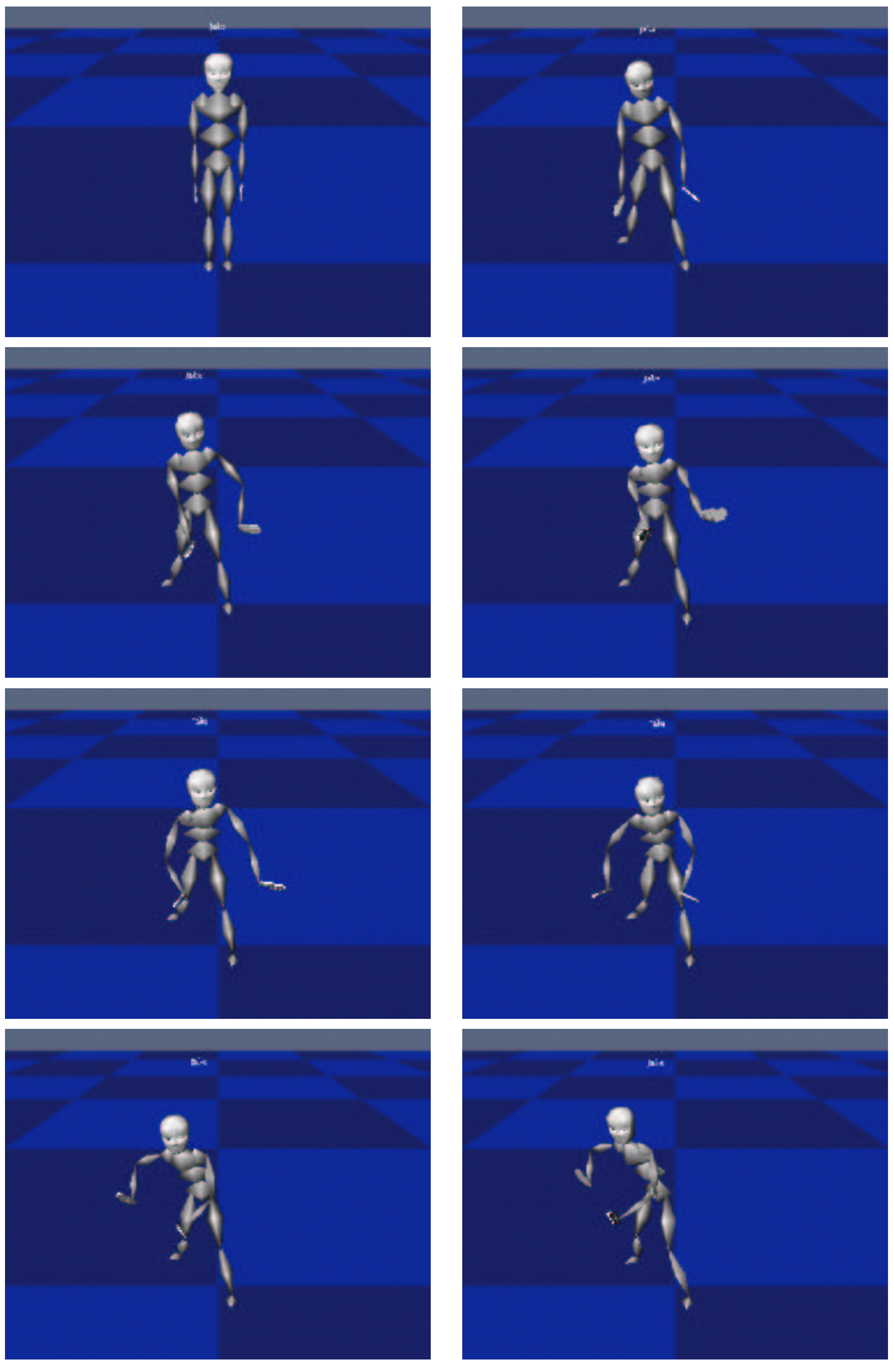


Figure 5.11: The motion of the actor catching a ball.(left to right, top to bottom)

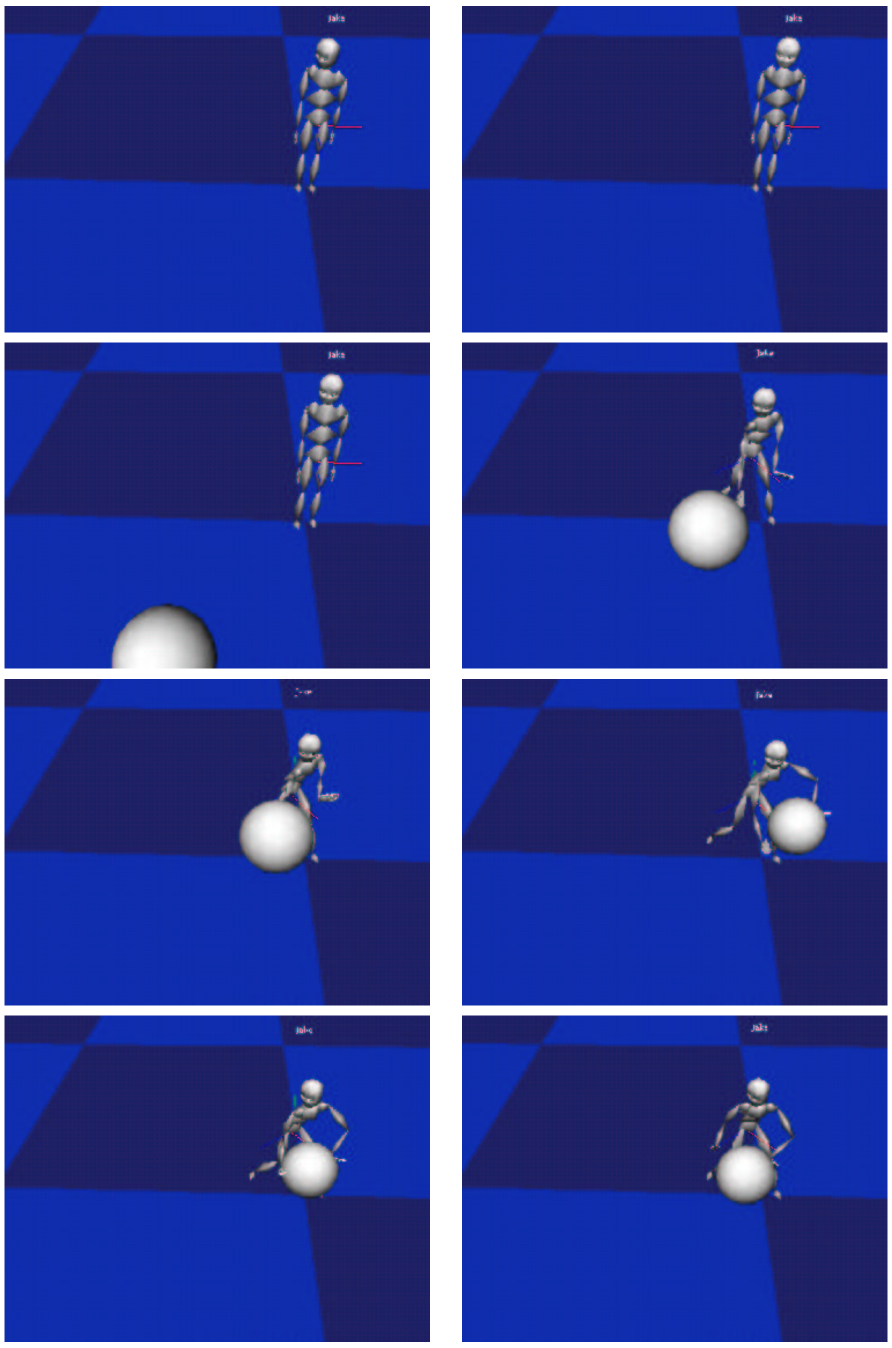


Figure 5.12: The actor reacting to an approaching ball by catching it. The actor looks around the environment (frame 1). It then spots the ball and watches it (frames 2-3). When the ball is close enough the competence is invoked. (A large ball was used as it was similar to the original motion. Using an object of a very different size causes difficulties as it involves an excessive warping of the motion. Similar techniques to those described in section 7.4.3 could be used to solve this problem).(left to right, top to bottom)

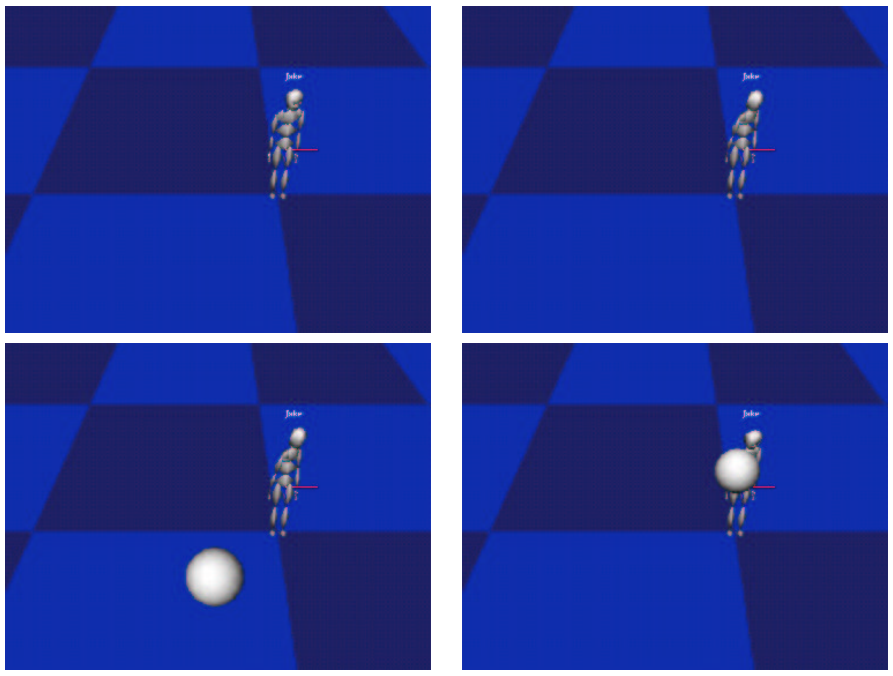
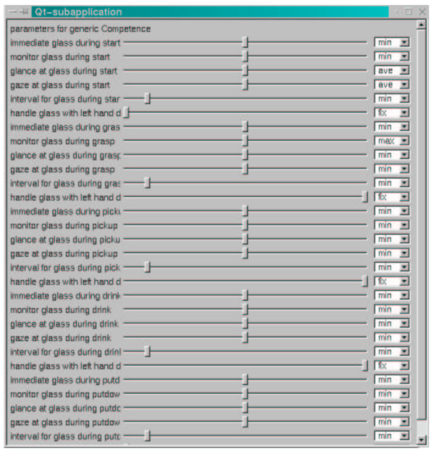


Figure 5.13: If the actor is looking in the wrong direction it will not spot the ball and fail to catch it.

The designer of a competence can also set constraints on the values of parameters to represent whether a secondary action is or is not suitable for a given action irrespective of which character is performing it.

The values of the parameters are controlled using a set of sliders shown in figure 5.14(a). Multiple sliders are shown for each parameter of a secondary action, one for each period of the competence. The user or designer can set the values of the parameter using the sliders and then view the resulting behaviour to see if it is suitable. Values of parameters can then be saved.

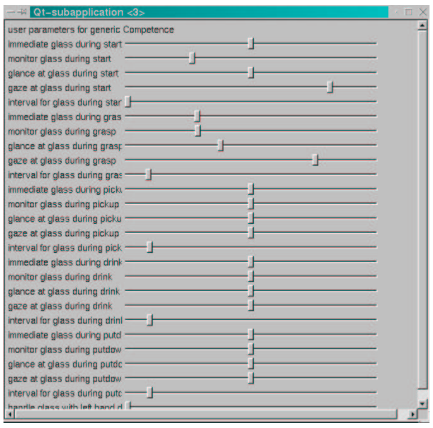
When the designer adjusts the parameters he or she is providing constraints on the possible values of the parameter. These can be of four types, Minimum and Maximum values; a Standard value which is the values which the parameter starts at by default, and a Fixed value which means that the parameter always has this value and the user cannot change it. Minimum, Maximum and Standard values may all be present together for a parameter but if there is a Fixed value none of the others are present. The user can also change the values of the various parameters that control the secondary actions, within the limits set by the designer. The user can manipulate them using a set of sliders similar to those used by the designer (shown in figure 5.14(b)). The end points of the sliders are now the Minimum and Maximum values provided by the designer. If any of the parameters have fixed values these parameters will not have sliders. The user also has similar sets of sliders to control the parameters of the attention and walking behaviour.



parameters for generic Competence

immediate glass during start	min
monitor glass during start	min
glance at glass during start	ave
gaze at glass during start	ave
interval for glass during star	min
handle glass with left hand d	fix
immediate glass during gras	min
monitor glass during grasp	max
glance at glass during grasp	min
gaze at glass during grasp	min
interval for glass during gras	min
handle glass with left hand d	fix
immediate glass during picku	min
monitor glass during pickup	min
glance at glass during picku	min
gaze at glass during pickup	min
interval for glass during pick	min
handle glass with left hand d	fix
immediate glass during drink	min
monitor glass during drink	min
glance at glass during drink	min
gaze at glass during drink	min
interval for glass during drinl	min
handle glass with left hand d	fix
immediate glass during putd	min
monitor glass during putdow	min
glance at glass during putdc	min
gaze at glass during putdow	min
interval for glass during putc	min

(a)



user parameters for generic Competence
immediate glass during start
monitor glass during start
glance at glass during start
gaze at glass during start
interval for glass during star
immediate glass during gras
monitor glass during grasp
glance at glass during grasp
gaze at glass during grasp
interval for glass during gras
immediate glass during picku
monitor glass during pickup
glance at glass during picku
gaze at glass during pickup
interval for glass during pick
immediate glass during drink
monitor glass during drink
glance at glass during drink
gaze at glass during drink
interval for glass during drinl
immediate glass during putd
monitor glass during putdow
glance at glass during putdc
gaze at glass during putdow
interval for glass during putc
handle glass with left hand d

(b)

Figure 5.14: The sliders controlling the parameters of the actions. (a) is the designers set of sliders. This sets the limits for the parameters. The designer can choose if the parameter value set is the minimum, maximum, average (starting value) or a fixed value that the user cannot change. The designer can set all of minimum, maximum and average independently for a given parameter but a fixed value will override all of these. (b) is the user's set of sliders. Note that parameters that have been tagged as fixed do not appear.

From a user interface point of view, it would be desirable to have a method of controlling the parameters that has a more intuitive connection between the method of control and its effects. Possible ways of doing this are discussed in section 7.4.2.

5.8 Conclusion and examples

This chapter has provided a relatively simple-to-use method for creating generic behavioural competences that the user can invoke. The attention mechanisms allows reactive behaviour to be produced relatively easily to the user.

At this stage the method is still too simple to produce the full range of behaviour that would be desirable and it is unclear whether it would be possible to produce a tool that could generate a full range of behaviour without adding so much complexity that it would be unusable. However, it seems likely that it would be possible to produce a tool complex enough to be useful and simplify the creation of animations, though some actions would still have to be animated by hand. Currently the motion produced is somewhat limited. As the competence is based on a single piece of motion, if the contact argument moves too far from its original position the motion will look unnatural (figure 5.17 shows an example). Section 7.4.3 discusses these limitations and potential solutions. However, the primary interest in competences from the point of view of this work is how they use vision and attention to produce behaviour.

The handle action mechanism is a simple way of having an object follow an end effector. The results are reasonable but the objects tend to move too much and there is some difficulty aligning them to other points such as the actors mouth in the drinking examples. Better motion data could solve this problem. It might also be possible to make the algorithm more sophisticated. For example, extracting a smooth path from the original motion and then applying a warped version of this to the object. The motion of the end effector could then be warped to follow this. This could also be used for the case where the actor is handling the object with two end effectors, such as the catching example. At the moment handle action only works with one end effector. This means that motion where the actor handles an object with two end effectors only looks realistic if the target is a similar size to the original object. There might be some problems with this sort of method as producing a smooth path that does not loose the important characteristics of the original motion might be difficult in the general case.[3]

The next section discuss how competences produce attention behaviour. Section 5.8.2 discusses how the attention mechanism can be used to invoke competences.

5.8.1 Competences and attention

The eye action aims to add attention shifts and their associated eye and head movements to a competence. This enhances the actor's perceived connection with the task by having the actor look at appropriate places. This is done by designating various positions as contact arguments and dividing the motion of the competence into various periods. The user can then define the probabilities of looking at a particular contact argument during a particular period. The user can also specify various properties of the attention request which determine how the actor looks at an object. Figures 5.15 to 5.19 show the actor performing competences with different values for the various probabilities and parameters. The *minimum distance* (see section 4.4.2) parameter is always set to a very low value as competences generally involve looking at objects that are close to the actor, this is in contrast to the attention requests produces by the navigation agents described in the next chapter. In all these examples take particular note of the eye and head movements.

Figure 5.15 show an action with the actor drinking from a glass, as do the next two examples. The contact arguments are the glass and another, secondary, argument representing a friend that the actor is talking to, shown as the second actor in the scene. The actor monitors both the friend and the can of drink. The looks directed at the friend are always long gazes whereas the can is glanced at. The actor looks at the friend for a much longer time and at the can only briefly. Note also that the actor's head is kept still while drinking and that as it does so it glances to the side without moving its head.

Figure 5.16 shows the same competence and scene but with different settings. Here the actor is given an immediate request for the can as it picks it up but no other requests

[3]The animations on the CDROM were produced by importing the motion of the actor and the objects into an external package (3D Studio Max). Aligning the two was slightly problematic so the relative position of the end effector and object are not identical to the original. It is hoped that they are as close as possible and can at least give an impression of what the original was like.

to look at the can. It monitors the friend. This time, however, it glances at the friend and spends the rest of the time in undirected attention behaviour looking around itself. As well as just keeping its head still while drinking a parameter is set so that it faces forward as well.

Figure 5.17 show the same competence again but without any specific requests so the actor will just perform undirected attention behaviour. However, this has been adapted to the situation by changing the default direction for the undirected looking to be the friend. This has a similar effect to monitoring the friend but is more likely to be interrupted by other types of attention behaviour. In the previous two examples the competence requested that the attention manager keep the actor's head still while the actor is actually drinking. This example does not do this and shows why it is important to do so, as the actor's head should be taken up by the act of drinking and looks wrong if it moves independently.

Figure 5.18 shows the competence from figure 5.16 without secondary actions, i.e. without the can following the actor's hand and without eye movements. The lack of eye movements means that the actor appears less lifelike and more rigid or robotic. It no longer seems as involved with its behaviour.

Figure 5.19 shows a different competence where the actor picks an object up from one place and puts down in another. The actor has an immediate request to look at the object when it picks it up and when it puts it down but not in between. Figure 5.20 shows the same motion without the motion warp or eye movements to show the difference.

These examples, show some of the range of attention behaviour it is possible to produce with the competence mechanism. It is possible to produce a reasonable range of useful behaviour patterns. The reader can judge for his or herself the quality of the motion by viewing animations on the accompanying CDROM.

5.8.2 Vision and attention for invoking competences

The vision and attention mechanisms are not just aimed at adding eye movements to a competence: they can be used to actively shape behaviour. There are two mechanisms by which the attention mechanisms can be used to find an appropriate target for a competence.

In the first method the actor actively searches for a target. This uses the attention manager's search mechanism which visually scans the environment looking for a target. Figures 5.9 and 5.10 shows the actor searching for a target, walking to it and performing the competence. The target is initially occluded so the actor has to walk for a while in order to find it. The competence produces reasonable results with the actor appearing to be purposefully looking for the target. If the actor is already aware of a suitable target it will use that.

The second method allows the character to react to an appropriate object by performing an action. This would be particularly useful for a non-player character in a computer game which would have to catch balls in a sporting game or to react to a player character in some way. This is done by using both the vision and attention mechanisms to detect when an appropriate object is approaching and then invoking the competences on it. This is a good example of using both the mechanisms together to produce behaviour. Figure 5.12 shows the actor reacting to an approaching ball by catching it. This works well and in tests detects any object that it ought to. However, if the actor is not paying attention it will fail to catch the ball as shown in figure 5.13. This is correct behaviour, the purpose of the vision and attention mechanisms is to determine what the actor is not aware of as well as what it is aware of.

These two methods are merely two of a number of modes of interacting with the environment that can be separated from the actual action being performed and used for designing and invoking competences. It would be possible to design others, for example an interaction system for dealing with competences that involve more than one person. It might be possible to to build up a full suite of these modes of interaction that could be used in designing competences. It is not yet clear, however, if it would be feasible to find a closed set which would represent a large enough proportion of the actions required from animation. Neither is it clear whether this set would become too complex to be easy to understand and use.

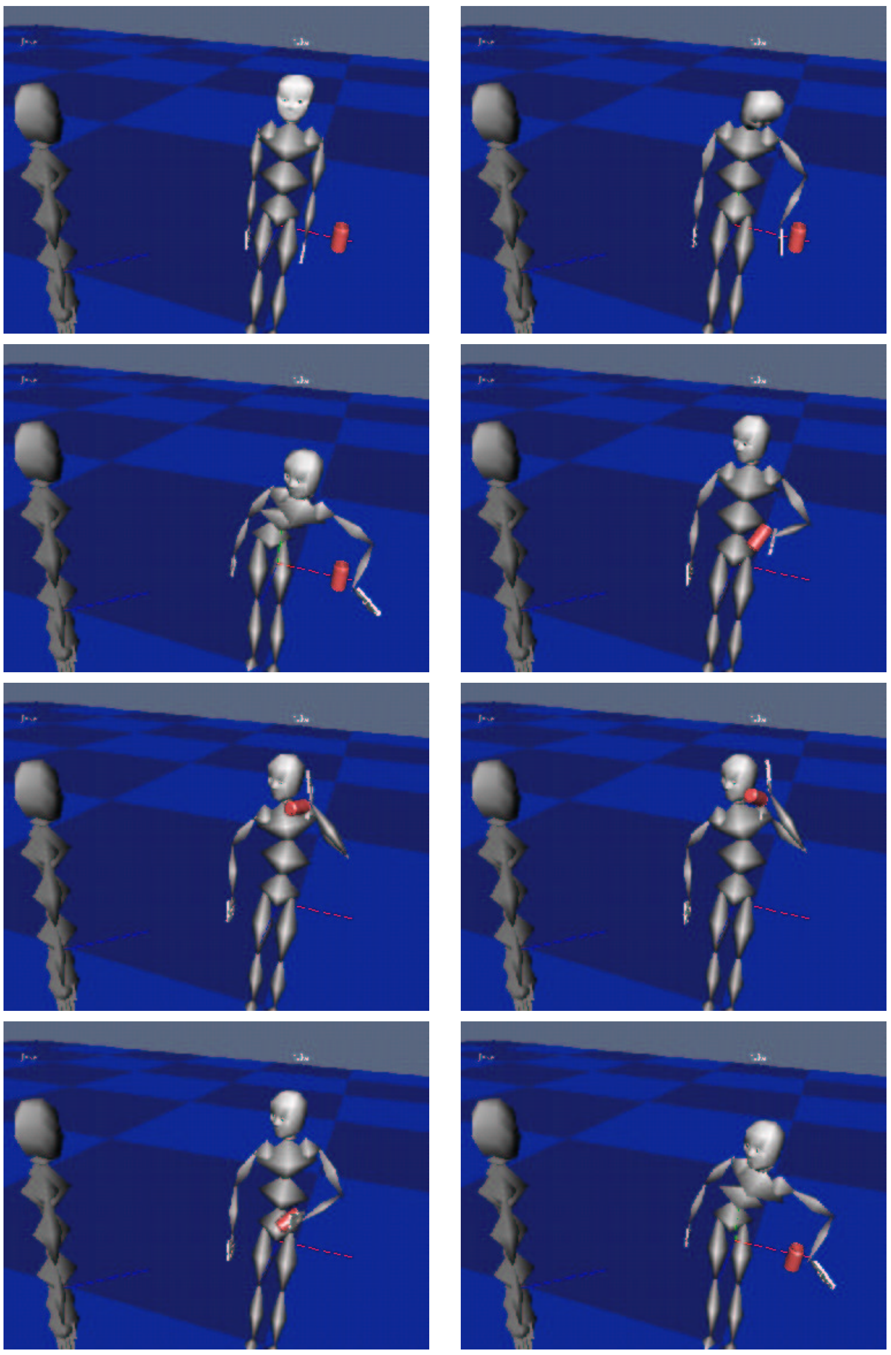


Figure 5.15: The actor performing a drinking competence. There are two contacts, the can of drink and a secondary contact representing a friend (shown here as the other actor). The actor monitors both the can and the friend but glances at the can and gazes at the friend. (left to right, top to bottom)

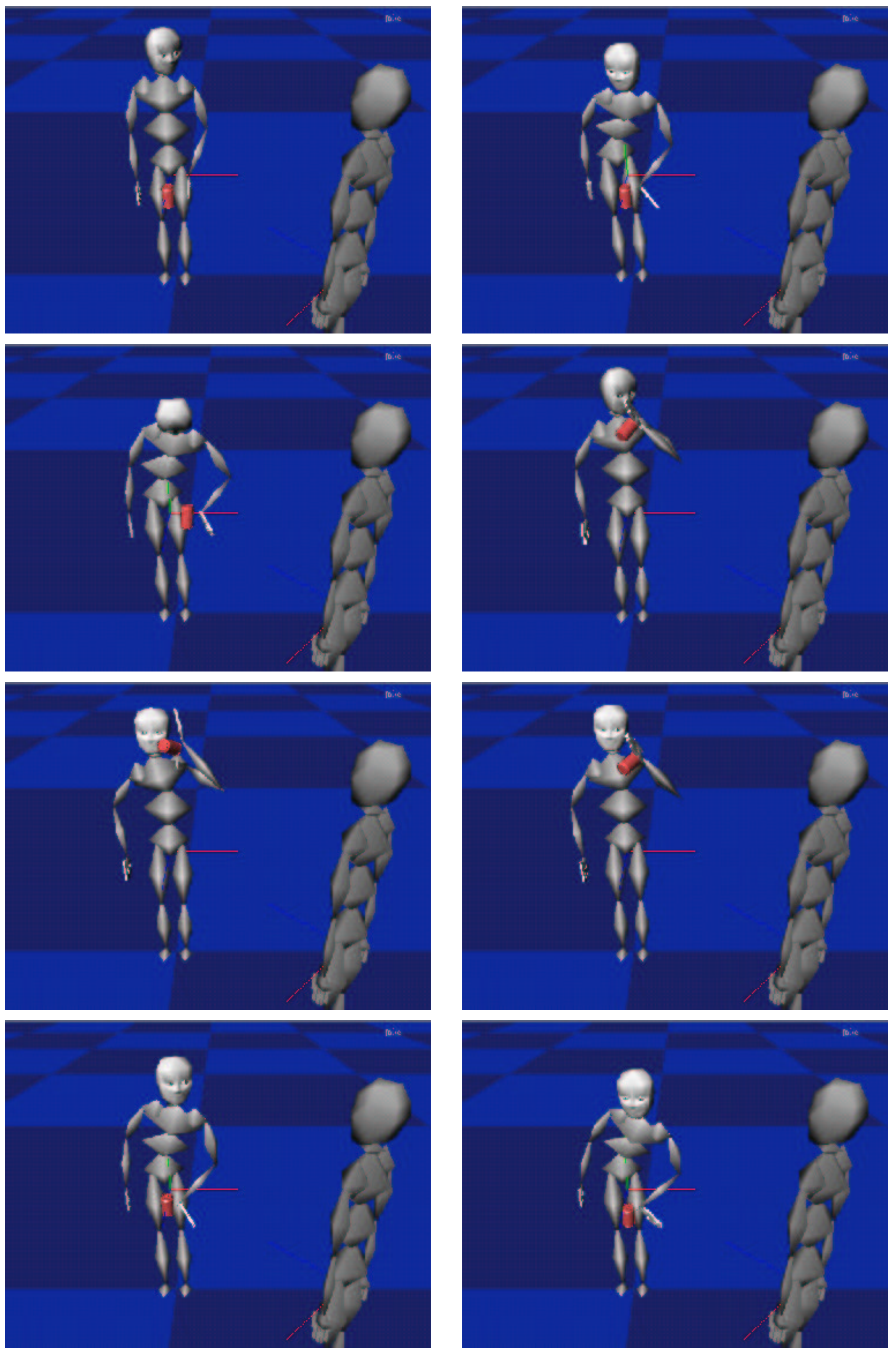


Figure 5.16: Another example of the drinking competence. Here the actor has an immediate request to look at the can as it picks it up and otherwise monitors the friend. It glances at the friend and so has time to perform undirected attention behaviour, looking around itself. The actor is also given a request to look forwards as it is drinking. (left to right, top to bottom)

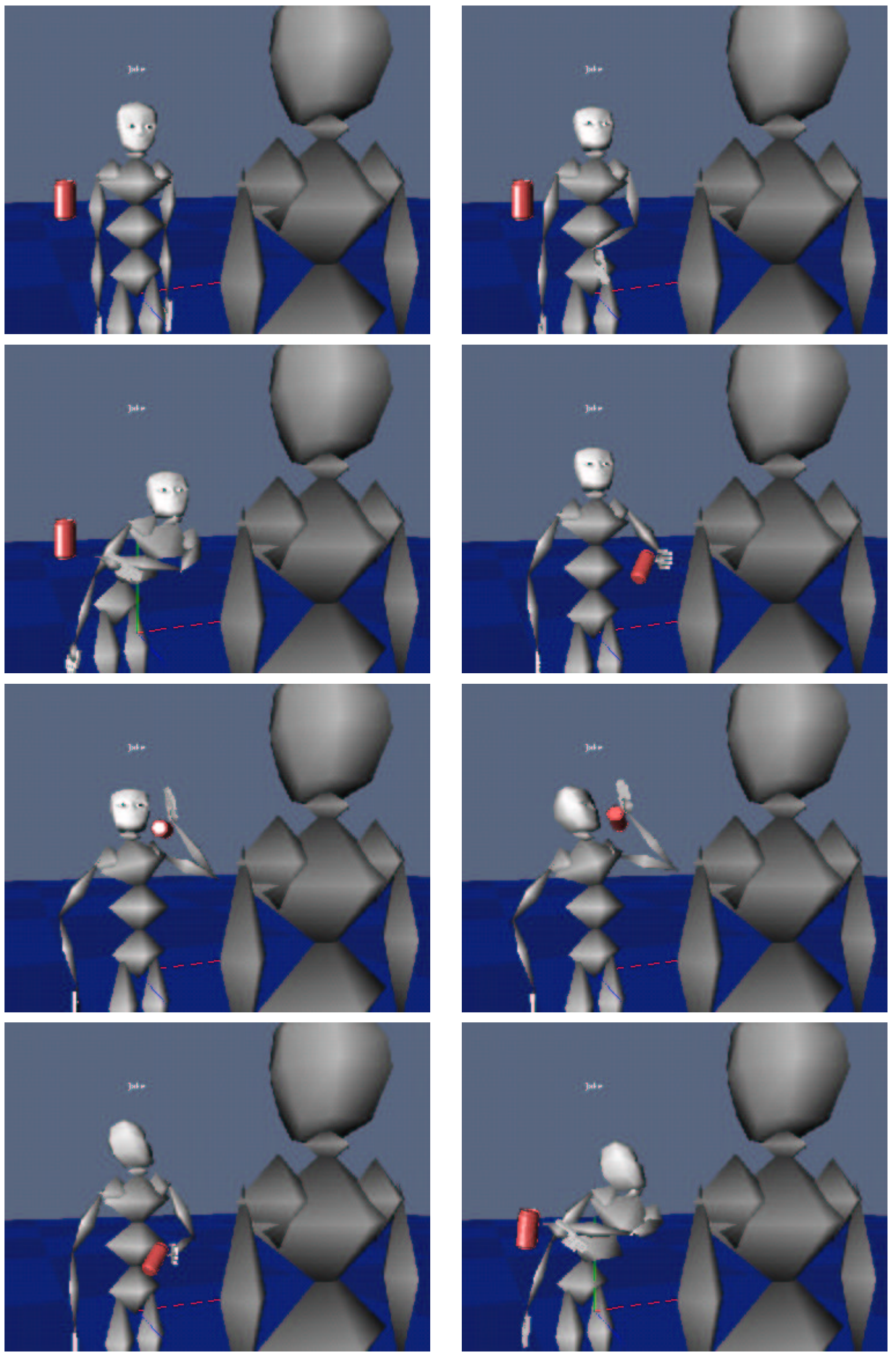


Figure 5.17: An example of the drinking competence with no particular attention behaviour specified. The default direction is towards the other actor. The head still flag is not set during drinking so the actors head moves while it drinks (frames 5 and 6). This is not desirable behaviour. (left to right, top to bottom)

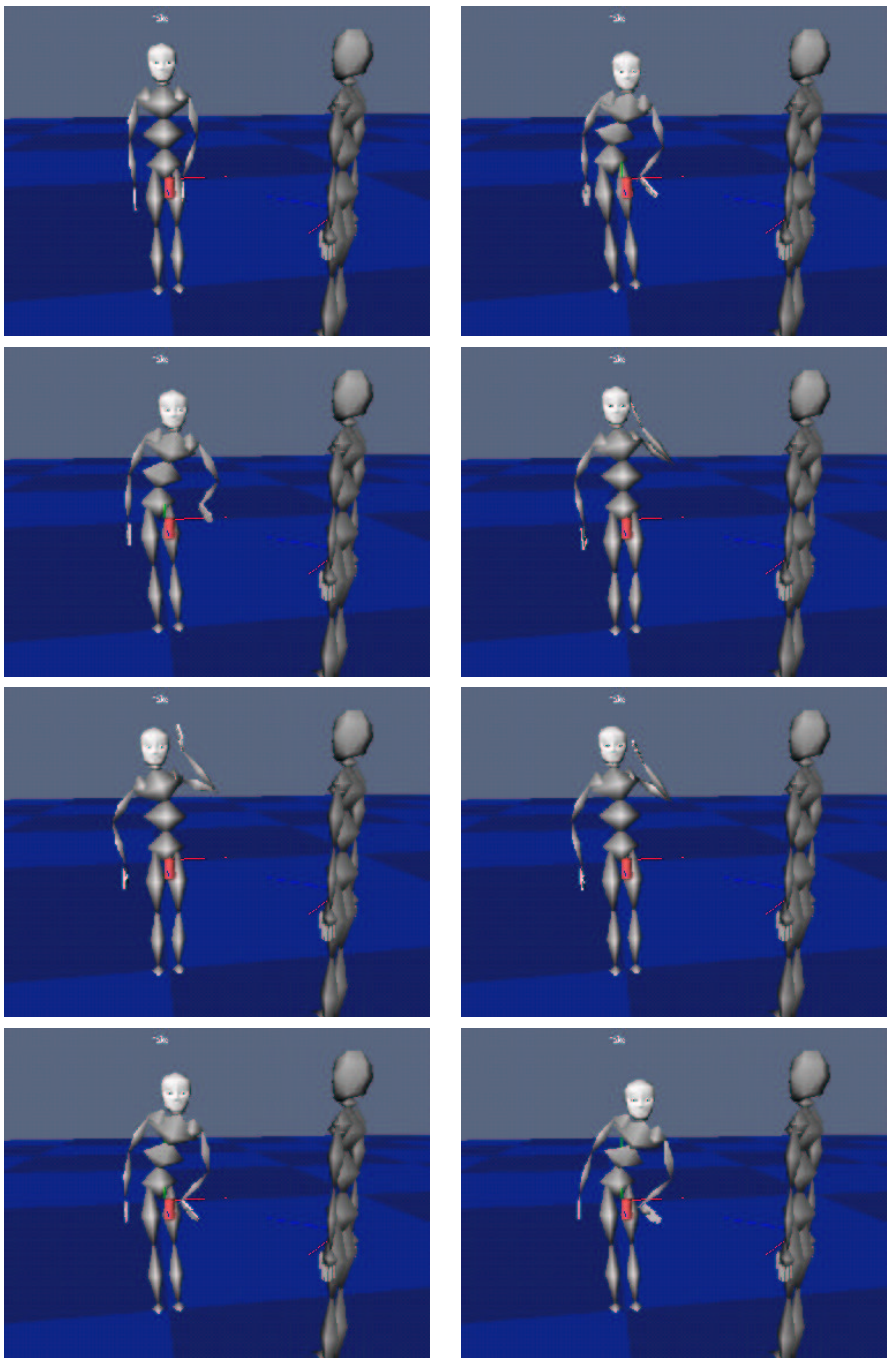


Figure 5.18: The drinking competence without secondary actions so the can does not follow the actors hand and there is no attention behaviour. The actor's gaze is unmoving. The actor appears more robotic and less realistic. (left to right, top to bottom)

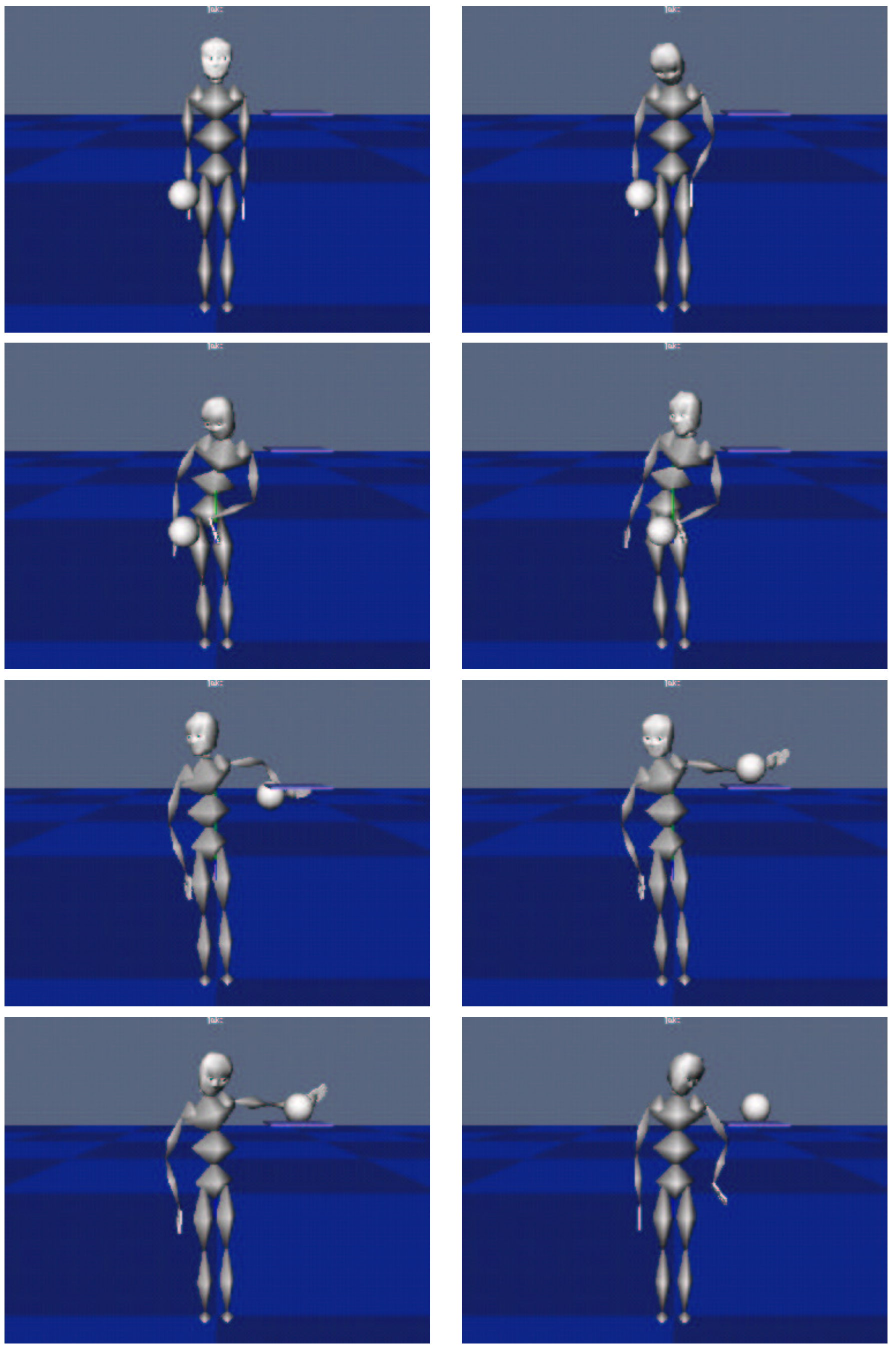


Figure 5.19: The actor putting a ball on a shelf. An example of a competence with two contacts. In frames 3 and 4 there is no specific attention behaviour specified by the competence and undirected attention results in the actor looking at a location to its right. (left to right, top to bottom)

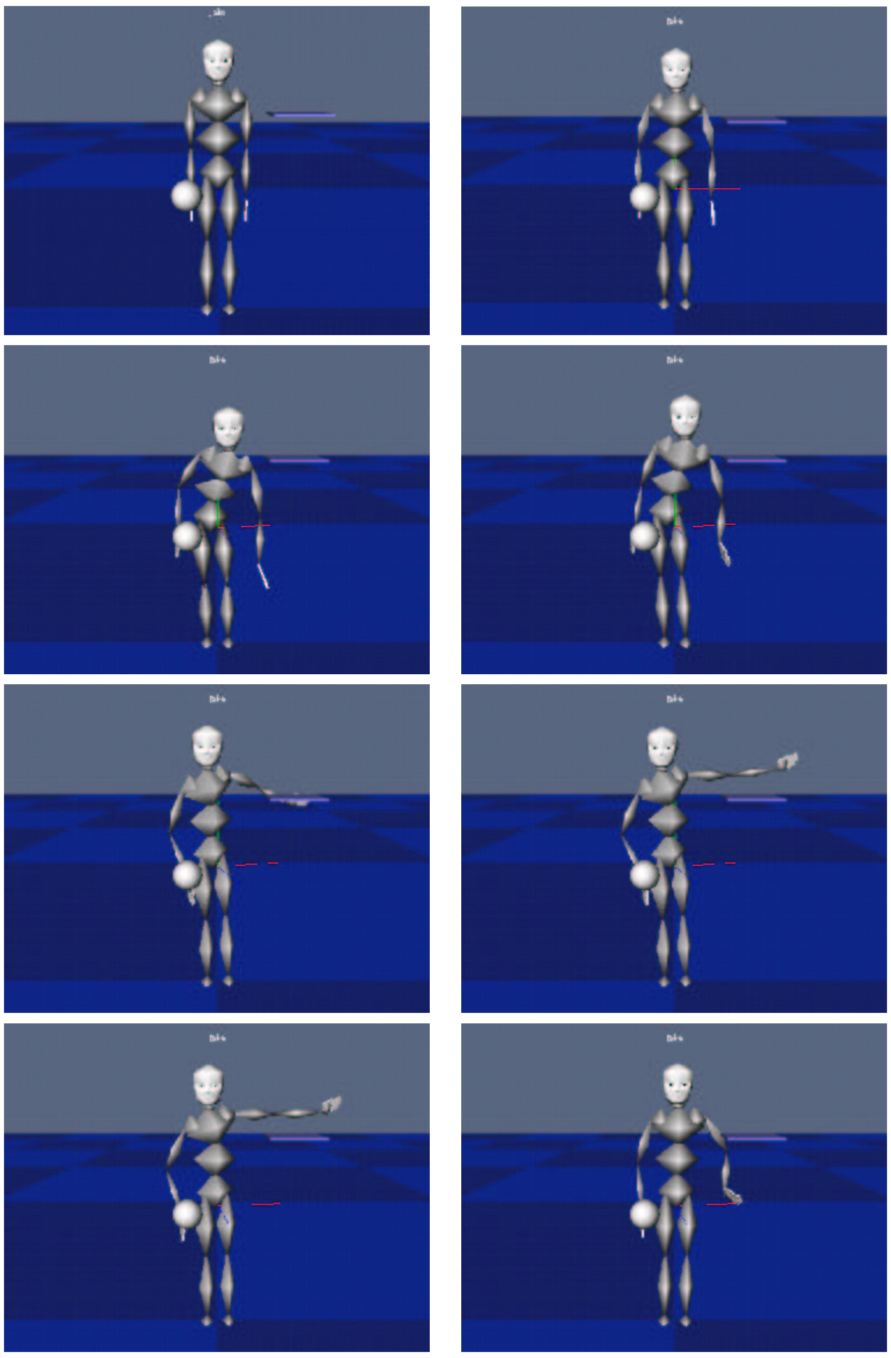


Figure 5.20: The original motion of figure 5.19. (left to right, top to bottom)

Chapter 6

Navigating a cluttered environment

This chapter presents an extended set of behavioural competences which involve navigating a virtual environment containing a variety of static and moving obstacles which the actor must avoid. This is a vital building block for actors which control their own behaviour as any interesting environment will contain a variety of objects and the actor will need to walk around this environment without bumping into the objects. Having a set of behaviours like this allows an animator to request that the actor moves from one point to another without having to define an exact path and without having to define the complex interactions between the actor and moving objects thus eliminating a time consuming aspect of the animation process.

6.1 The task

This section examines the task in detail. Section 6.1.1 discusses two techniques that have been extensively used in computer animation, *collision avoidance* and *path planning*, and how they fit in with navigation. Sections 6.1.2 and 6.1.3 discuss why a simulation of vision and attention is necessary for the navigation task. Finally section 6.1.4 gives an overview of my approach.

6.1.1 Collision Avoidance and Path Planning

Two of the most studied aspects of the navigation task are collision avoidance and path planning. They lie at opposite ends of a spectrum. Collision avoidance is concerned purely with reactively changing the path to avoid collision with obstacles; whereas path planning involves pre-computing a path around a set of obstacles in an environment. They both have their origins in robotics where they are critical in enabling a robot to move in its surroundings.

Collision avoidance involves detecting obstacles and then adjusting the character's heading in order to avoid them. As part of a SIGGRAPH course in 1988 Craig Reynolds gave a summary of the state of the art in collision avoidance methods and classified a number of types of method which I will describe here. The *steer away from surface* or *force field* method is one of the simplest and has been used extensively in robotics. It involves simply accelerating away from any surface with an acceleration inversely related to the actor's distance from the obstacle. This is a simple and effective method and has been very useful in robotics but has drawbacks in animation where realism is of utmost importance. The greatest of these is its lack of directional discrimination, the actor will steer away from any object irrespective of its direction, so it will accelerate away from obstacles it is already moving away from. This problem could be solved by limiting the object accelerated away from to a visual cone ahead of the actor but the actor would still be unable to walk beside a long wall without veering away from it (very strange behaviour if you consider a human acting in this way). Also it is difficult to make an actor approach an obstacle closely, which is also fairly normal behaviour. A variant of this method is the *steer away from centre* where the acceleration is given from the centre rather than the surface of the object, this has only slight advantages over steer away from surface. *Steer along surface* or *curb feeler* involves maintaining a virtual "probe" which monitors the area in front of the actor. If the probe hits an obstacle the actor is steered away in the opposite direction from the contact. This has less major defects than the first two methods. However, since the actor still has no knowledge of the overall shape and size of the obstacle and so can produce rather eccentric behaviour. For example, an actor might encounter a long object near one end of its length. It would be natural to avoid it by going around the nearer end. However, the *curve feeler* always steers away from the contact which might be in the opposite direction, making the character steer the long way around the obstacle. This sort of strange behaviour is unacceptable in an animation system which aims to simulate the behaviour of normal human beings. *Steer towards silhouette edge* is the method used by Reynolds in his simulation of flocking birds [Rey87]. It involves monitoring the silhouette of the obstacle as seen by the actor. If the silhouette is intersected by the actor's forward movement vector (the actor's local Z axis) then there will be be a collision. The actor then calculates the nearest point on the edge of the silhouette to the point of intersection and then steers towards a point just beyond this, off the edge of the silhouette. This method, if applied equally in all cases does not reproduce the vagaries of human attention and whim but it is an effective method and I use a variant of it as my lowest level avoidance strategy.

Other simple collision avoidance methods have been used. For example, Tecchia and Chrysanthou [TC00] use a grid representation of the environment with each cell containing heights. The actor only looks one cell ahead and changes direction if the height of the next cell is too great. This is a very simplistic method but is very efficient and they use it for a simulation of a very large area (a model of central London) containing thousands of people.

Path planning is a much higher level activity. It involves searching a cluttered environment for an optimal path from one point to another while avoiding all obstacles in between. Rather than merely avoiding one obstacle at a time the path planner takes into account all obstacles simultaneously to form the path. In general the techniques used are based on AI search and path planning in robotics and involve discretizing the environment into some intermediate representation and then performing some sort of heuristic search on this representation. Noser, Renault and Thalmann [NRTMT95] form a voxel representation of the environment where each voxel is either free if the actor can walk through it or occupied if it contains an obstacle. The voxels are grouped into an octree which is searched using a two-pass method which first finds a sub-optimal path and then refines this to a shortest path using a variant of a Moore-Dijkstra search algorithm. Kuffner [KJ98] uses a grid representation of the environment which is then searched using Dijkstra's algorithm. The resulting path is then used to guide an actor walking based on a motion-captured walk sequence. Bandi and Thalmann[BT98] use a grid representation which is searched using an A* search algorithm. The resulting path is smoothed by using a Digital Differential Analyser to form straight lines between points on the path. One problem with these path planning algorithms is that they are designed to find optimal shortest paths between two points. Though this can be desirable, often people will take non optimal paths for various reasons. Bandi and Cavazza[BC99] address this problem by using a modified A* which optimises simultaneously for short paths and for an arbitrary second heuristic. They give an example of an actor which walks a shortest path between two points while not turning its back on a door. The problem with this method is that it can only deal with one secondary heuristic and the heuristics are not complex enough to model, say, a character's wandering attention. Another problem with path planning is that it tends to be computationally expensive. This is fine in a relatively static environment as the path can be precomputed and then executed very quickly; however, in a dynamic environment with a number of moving objects the path will have to be recomputed frequently which would be costly. I propose using planning techniques to form a large scale path around the static elements of the environment such as buildings and roads and then using more reactive techniques which I describe in this chapter to alter the path to deal with smaller obstacles and moving objects.[1]

6.1.2 Vision and navigation

One aspect of navigation which is not addressed by most path planning algorithms is the simulation of vision. It is of course vital to take into account only objects that the actor can see and use only information which is available to the actor through vision (or hearing or smell etc). Various approaches have been used to simulate vision. Systems like Kuffner's [KJ98] or Bandi and Cavazza[BC99] assume that the actor has prior knowledge of the environment and so use a complete map of the environment for planning. Other actor systems use some *ad hoc* approach to vision, for example Reynold's[Rey87] boids use a special geometrical representation of the environment for collision avoidance. In general,

[1]Another possible approach which is not investigated here would be to take as a starting point the literature on peoples represention environments, inspired by Lynch's[Lyn60] work on people representations of cities. There is a large literature on this subject for example, Gärling, Bröök and Lindberg[GBL84] or Golledge[Gol95], which could be applied to designing represenations for actor navigation. Darken and Sibert[DS96] have applied this to studying how real people navigate virtual environments.

however, these methods cannot really be said to simulate vision. This sort of method, as well as the others described in this section are discussed in more detail in chapter 3.

More sophisticated methods perform vision-like sensing by rendering the scene from the point of view of the actor into some sort of grid representation. Reynolds[Rey88] mentions a few image-based techniques for collision avoidance. A 2D rendered image of a scene with objects segmented from non-objects can be used to implement the steer towards silhouette edge strategy (see section 6.1.1). This silhouetted image is extracted automatically by the segmentation. In fact, if the silhouette is *fuzzy*, i.e. the segmentation is a grey scale map with the uncertainty of the image boundary being represented by a gradient from white at the centre of the object to black in areas where there is certain to be no object. This gradient can be used to find which direction to move in to find the silhouette edge. Another sort of image that can be used for collision avoidance is a z-buffer image, with each pixel representing the distance from the actor to the nearest object in that direction. Reynolds suggests that this can be used to implement the steer along surface strategy (see section 6.1.1). Alternatively it could be used for a new strategy *steer towards longest clear path* where the actor moves in the direction where there is the longest unobstructed path. This is a robust method but will produce rather random movements and is not suitable for clearly guided movement towards a goal. Noser, Renault and Thalmann [NRTMT95] use both a z-buffer and a buffer containing the identifiers of the objects in each pixel. This removes some of the difficult vision problems of depth perception and object segmentation. From these two buffers they calculate the 3D position of the objects and use it to form their octree representation of the environment which is used for path planning (described in section 6.1.1).

One advantage of image-based methods is that they calculate the visibility of the object by the actor automatically, but this advantage is not as great as it may seem as discussed in section 3.4.4. These methods have a central problem which is that they always lose information in the projection from 3D to 2D. This means that the information has to be recalculated. This makes the process rather less efficient than using the geometric data directly (though using hardware rendering to produce the image can speed the process considerably). More importantly computer vision techniques are still primitive so even by having specialised images such as z-buffers and maps of object identifiers what can be done is limited to tasks for which algorithms exist. Also a rectangular 2D image is not a particularly good representation of the human visual field. Instead I have simulated how humans use visual data to perform navigation tasks rather than the low-level extraction of this data. These points are described more fully in chapter 3.

6.1.3 Attention and Navigation

Attention is a vital aspect of human navigation. We can only take avoidance action on objects which we know about, and we only know about the objects which we have attended to. Previous methods do not take this into account, they assume that so long as an object is geometrically visible to the actor, it knows about it and will take action about it. While this is not too unreasonable an assumption with large static obstacles, the direction to which we are attending can make a great difference to how soon we know about a moving object which might appear suddenly in our visual field. In fact this is also true of small static objects such as lamp posts which can suddenly enter our visual field as we move. Though only a very inattentive actor will be completely unaware of an object until they actually collide, the time an actor finds out about an object can greatly change both the time at which the actor reacts to it and how the actor reacts to it (this is discussed in section 6.3). Chopra-Khullar and Badler[CKB99] have a model of attention and eye movements while navigating an environment. However, their main focus is producing the eye movements so they do not fully explore the role of attention in producing the navigation behaviour. The following section discusses the simulation of navigation behaviour and in particular how the actor's attention interacts with this behaviour.

6.1.4 My Approach

Path planning is an important part of navigation. We all start with a pre-planned path to a destination. However, it is not reactive enough to deal with a dynamic environment and it does not reflect the vagaries of human attention and personality. A path planned by a real person is likely to take into account large scale, permanent features such as buildings but is less likely to take into account small scale feature such as rubbish bins. It certainly will not take into account temporary features such as abandoned umbrellas or moving features such as cars or animals. My system assumes that a large scale plan has already been made, expressed as a series of intermediary destinations. The actor will attempt to follow this path but will have to deal with obstacles, both moving and static. This will be done in a psychologically based way with a simulation of attention and vision. The algorithms themselves will be left parameterised rather than fixed so that different characters will perform actions in different ways.

6.2 The walk controller

Most of the time people have a regular, repetitive walking style with each step being approximately the same length and speed. However, it is often necessary to alter the gait for specific steps, to change direction, walk around obstructions or step over obstacles. For the task of navigating a cluttered environment this is particularly important. The function of the walking agents is to control the ordinary walking motion of the actor and then alter it to avoid or step over obstacles. It is an ordered hierarchy of agents with the walk generator agent at the bottom controlling the actual walking motion based on a piece of keyframed or motion-captured walking motion. Above the walking agent the rest of the hierarchy control how this walking motion should be altered in response to obstacles. This is done by altering various parameters in the walk generator agent which in turn result in various motion transforms being applied to the walking motion. As the actions of stepping over and walking around obstacles are performed in terms of transforms of an existing piece of motion rather than by generating new motions, the hierarchy can be used to control the walking of characters with a wide variety of walking gaits.

6.2.1 The walk generator

The bottom agent of the walking hierarchy controls the actual walking motion and is the only agent that directly affects the motion of the character. The motion is based on a pre-existing piece of motion. This could be motion captured walking or a hand-designed gait. There is, in fact very little restriction on the nature of the motion; it can include anything that has the properties of a walk, i.e. the legs have to take steps where one leg is moving forward and the other is pushing backwards so as to move the character forward and these steps have to alternate from one leg to the other. The motion itself is represented as a series of variably spaced keyframes, with no restriction on the number of keyframes. The agent automatically works out where various positions such as footfalls are within the motion so a piece of motion can simply be fitted into the walk as soon as it has been generated without the need for any preprocessing or annotating of the motion. This means an animator can use any piece of walking motion while using the controller, giving different characters different and expressive walking styles. The personality expressed by motion is very important in animation so a tool that did not allow the use of different motions would not be very useful to an animator.

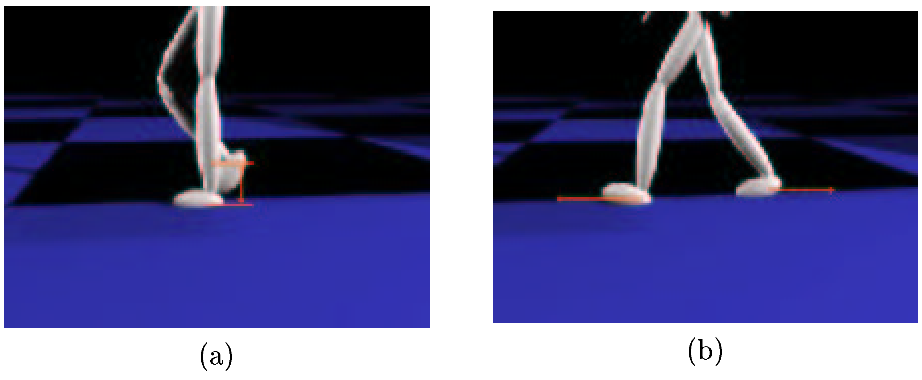

(a) (b)

Figure 6.1: The heuristics for determining which foot is planted. (a) If one foot is clearly lower than the other it is used as the planted foot. (b) If neither foot is clearly lower, the one moving backwards fastest or forwards slowest is used as the planted foot.

Ground

The motion used in the *walk generator* agent is defined in the local coordinate system of the actor, i.e. the motion of the limbs is defined relative to a root position of the actor rather than to some absolute origin. One consequence of this is that the motion contains no concept of the actor moving forward as it takes steps. An agent called *Ground*, developed jointly by Polichroniadis and myself, post processes the walk to convert it from a locally defined piece of motion to a globally defined walk that results in the actor moving forward in the world. This is done on the pre-existing motion without having to move the root of the articulation to the planted foot.

This is a rather complex task as it must work for most types of walking gait as well as more general motions that results in the actor moving forward. *Ground* assumes that, while in the locally defined motion both feet are moving relative to the actor's body, in fact one is planted on the floor and stationary in world coordinates while the other foot is off the floor and moving freely. It is assumed that the movement of the planted foot relative to the body is in fact that foot pushing the body forward. Two heuristics are used to determine which of the two feet are planted on the floor. The first is simple, if there is a significant difference in height between the two feet the lower foot is planted on the floor. This heuristic should be fairly intuitive to understand, as the feet cannot go below the floor, the lower one must be on the floor. However, if the difference in height is very small it is no longer clear which foot should be planted, as a slight error in the motion might mean that the foot that should be planted might be a few millimetres higher. Though this makes no visual difference it will stop *Ground* working. Thus, the first heuristic is augmented with a second one based on the velocities of the two feet. If neither foot is moving it can be assumed that the actor is still with both feet on the ground so it does not matter which foot is planted (it is left as the last planted foot). Otherwise the planted foot is chosen to be the one that is moving backwards quickest (or forwards slowest), this is effective as normally the planted foot is pushing backwards and the other foot is moving forwards. Normally the second heuristic will make sure that the actor is moving forwards. The two heuristics are shown in figure 6.1. Once the planted foot is chosen it is moved so that it rests on the floor and then its position is saved. At every frame until the planted foot changes, the difference in position and orientation from this initial state is calculated and is then subtracted from the position and orientation of the actor's root (i.e. the position and orientation of the actor as a whole). This results in a motion of the foot relative to the actor being transfered to a motion of the actor as a whole in the opposite direction. This means that the foot pushing backward during the walk cycle results in the actor moving forwards as desired.

Curve following

The main function of the walk generator is to transform a normal forward walking gait into a walk along a curved path. The path of the actor is defined by a series of destinations which can be specified by the user or could come from some other source, for example, a path planning algorithm. The actor has to walk in a smooth path between these destinations. This is done by using the destinations as the control points of a *Kochanek*[KB84] interpolating spline and then warping the walking motion so that it follows this spline. Further intermediate control points can be added to the spline in order to walk around obstacles. Figure 6.2 shows the actor following a curved path.

Curve following has been performed in a number of ways. Balder et al. [BPW93, pp154-161] have the centre of mass of the actor follow the curve and place the feet appropriately. They do not use pre-existing motion. Gleicher [Gle01] warps walking motion by moving a local coordinate frame along the path. Unlike these two methods the method presented here warps the positions of the feet rather than the root of the actor,

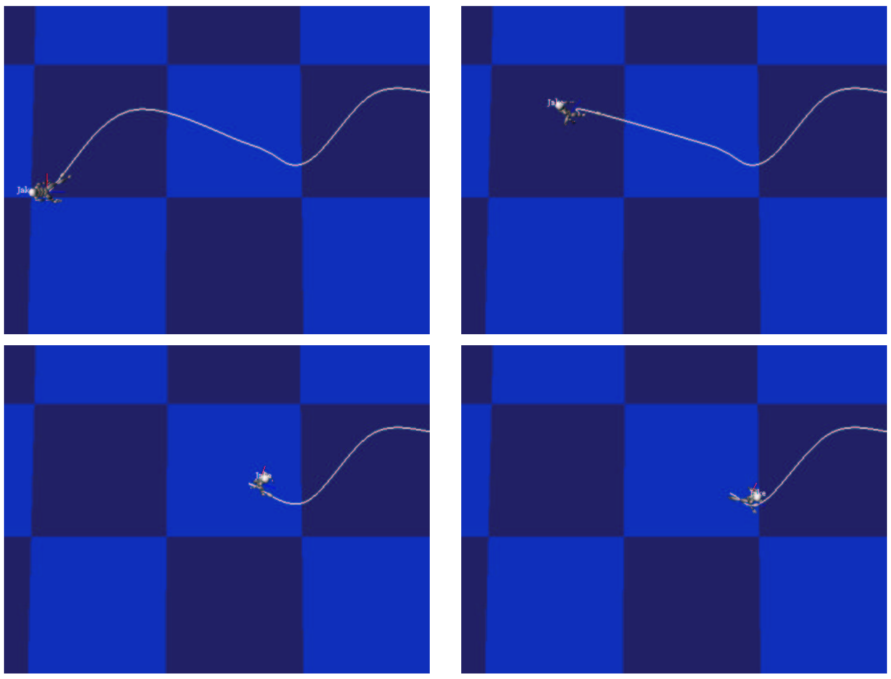


Figure 6.2: The actor walking along a curve

ensuring more accurate foot placement as is needed for stepping over obstacles.

The walking gait is warped on a step-by-step basis. At the start of each step a position for the foot fall is found along the curve. The motion is then transformed so that the foot fall lands at this position.

The position of the next foot fall is chosen as a point along the curve at a straight line distance from the current position of the foot equal to the length of a normal foot step (see figure 6.3). Finding such a point is slightly problematic as the arc-length along the curve is not an analytical function. This could be solved by using a *rational speed curve*. However, as the algorithm requires a straight line distance not a distance along the curve, the arclength would only be an approximation so using a slightly cruder approximation is good enough. This is done by taking the step length (sl) and dividing this by the distance between the current position of the actor's foot and the next control point (d). This is the proportion of the way to the next control point that the next foot step should be, if the next control point is closer than the step length the control point after that is used. This proportion is then used to calculate the parameter value for the next foot position. If t is the parameter value corresponding to the current foot position and t_i is the parameter value of the control point, then parameter value of the new foot position is t':

$$t' = t + \frac{sl}{d}(t_i - t)$$

This is used as a first approximation. The distance between the current foot position and

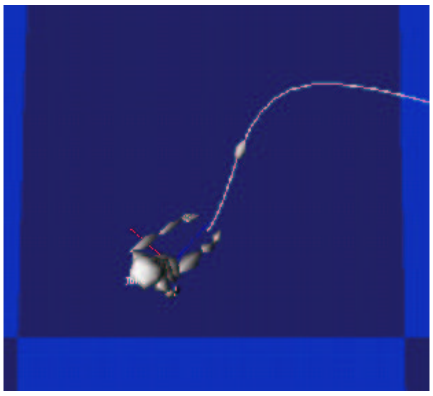


Figure 6.3: The next foot position is chosen at the start of each step. Here it is shown as the grey ellipsoid

the value of the curve at t' (d') is measured if this is not within a tolerance of the step length a new approximation is found:

$$t'' = t' - (d' - sl) * speed(t')$$

where $speed(t')$ is the *speed* of the curve at t'.This is the rate of change of arclength with the parameter value and is calculated from:

$$speed\,(t) = \sqrt{\left(\frac{d\mathbf{P}}{dt}\right)^2}$$

This process is repeated until a suitable parameter value is found. Normally only one or two iterations are needed. From this parameter value an appropriate foot position and orientation is found. This is parallel to the curve and placed so that the centre of the actor's body passes along the curve.

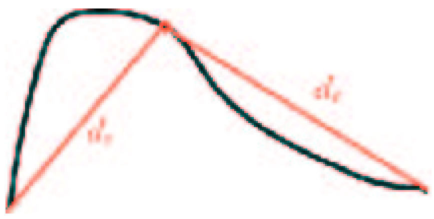


Figure 6.4: The distances of the foot position from the start and end of the motion

In order to make the actor's foot fall at this position the walking gait must be warped. This is done by taking the vector difference between the current position of the foot and

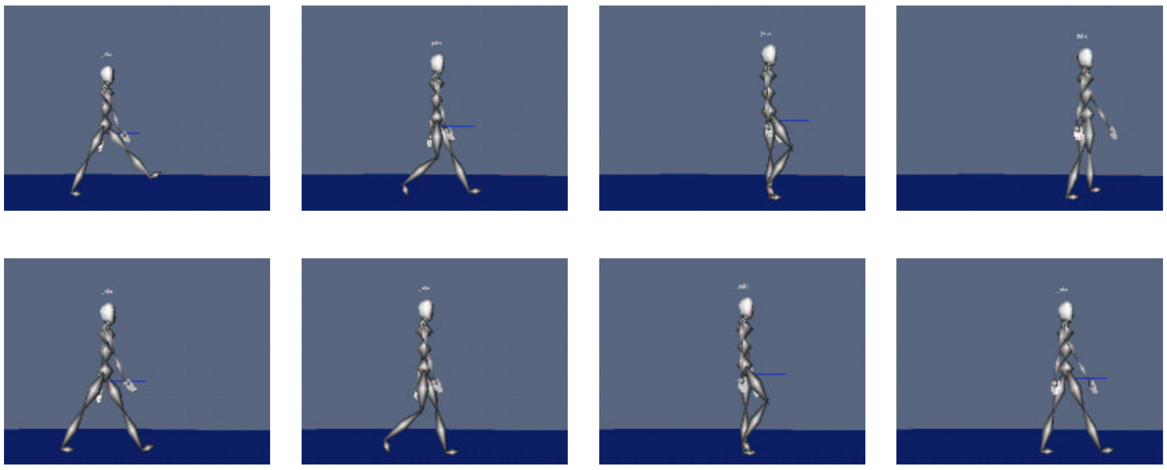


Figure 6.5: The difference in motion when the transformation is only applied to the lead foot as opposed to half being applied to each foot. The bottom row shows the correct motion where both feet have been transformed. In the top row only the lead foot has been transformed.

the desired footfall and subtracting from this the difference between the start and end of a normal footstep:

$$\mathbf{diff} = (\mathbf{footfall} - \mathbf{currentposition}) - (\mathbf{end} - \mathbf{start})$$

diff is used to transform the footstep. The current foot position will be due to a previous transform so that for the step to look correct the walk generator has to blend out of the old transform and into the new one. This is done based on the ratio of the spatial distances of the current position on the walk cycle from the start and end point d_s and d_e as shown in figure 6.4. While d_s is less than d_e the old transform is blended out, after that the new one is blended in. Blending one motion directly into the other would make the motion at the point where the legs cross each other incorrect. The blending is done by:

$$\mathbf{p}' = \mathbf{p} + \frac{d_e - d_s}{d_e + d_s}\mathbf{diff_{old}} \quad d_s < d_e$$

$$\mathbf{p}' = \mathbf{p} + \frac{d_s - d_e}{d_e + d_s}\mathbf{diff_{new}} \quad d_s > d_e$$

This transform is applied both to the position and orientation of the foot. In order to make the walk look balanced rather than just adding the difference to the moving foot half is added to the moving foot and half is subtracted from the planted foot. This has the same effect on the final position of the foot as moving the planted foot backwards in local coordinates results in moving the actor forwards in global coordinates. However, the motion looks better as each leg is transformed equally (see figure 6.5).

Walk transforms

The walk generator agent can perform a number of motion warps[WP95] on the walking motion. These were designed by Polichroniadis with a few extensions by myself. The first is performed automatically, the motion is straightened. This means that any curvature or zig-zag in the motion is removed so that it consists of forward movement along the positive Z-axis in local coordinates (obviously the local axis can be rotated so as to point along any direction in world coordinates). The remaining transforms are used to alter the motion to perform the various actions necessary to navigate the environment. The first transform is *Speed Modifier*: this changes the speed of the motion as a whole, making the steps take a longer or shorter time and so making the actor walk faster or slower. This transform is used to avoid side on collisions by speeding up to pass in front of an object. The second is *Transform Extent*: this changes the physical extent of the motion making the motion cover a larger or smaller area of space. In the walk generator agent this is used to make the steps longer or shorter and higher or lower in order to step over an obstacle. *Speed modifier* and *transform extent* are used together to produce a faster walking motion. *Speed modifier* reduces the time taken by each step and *transform extent* increases the length of the step. The last transform, *Shift Peak*, moves the peak of the motion of the foot (i.e. the highest point of the motion) to a different point in the motion. This allows the foot to be raised to any height at any point on the walk cycle, this is also used for stepping over obstacles. Figure 6.6 shows these transforms.

6.2.2 The Control hierarchy

The *walk generator* agent is controlled by a hierarchy of other agents as shown in figure 6.7. They control when and how the walk is altered to perform actions such as stepping over or walking around obstacles. Their architecture is inspired by Brooks' *Subsumption Architecture*[Bro85]. The subsumption architecture was originally conceived to control the actions of autonomous robots but can readily be adapted for multi-agent actor systems such as Blumberg's[BG95]. A subsumption architecture divides the control of a robot or actor into a series of layers in which lower layers can perform independently of any higher layer and perform some complete and useful (though possibly very limited) action. Brooks' early robots have obstacle avoidance as a low layer: a behaviour which maximises distance between the robot and any other objects. Higher layers add further functionality by using the lower layers to perform actions which are not normally part of their repertoire. For example in Brooks' robots a higher layer makes the robot move into spaces. It does this by using the obstacle avoidance layer's distance sensors to find a direction where the obstacles are furthest away and then *subsuming* the lower layer's movement control by using the same mechanism but overwriting the lower level's instruction with its own. However, if an obstacle is close enough that a collision might be about to happen, the lower layer takes control again to avoid the obstacle. Below I will discuss broadly what each layer does and how they interact.

The lowest layer of the hierarchy is the *walk generator* agent as described in section 6.2.1. This actually performs the transforms on the walking gait. The next three layers

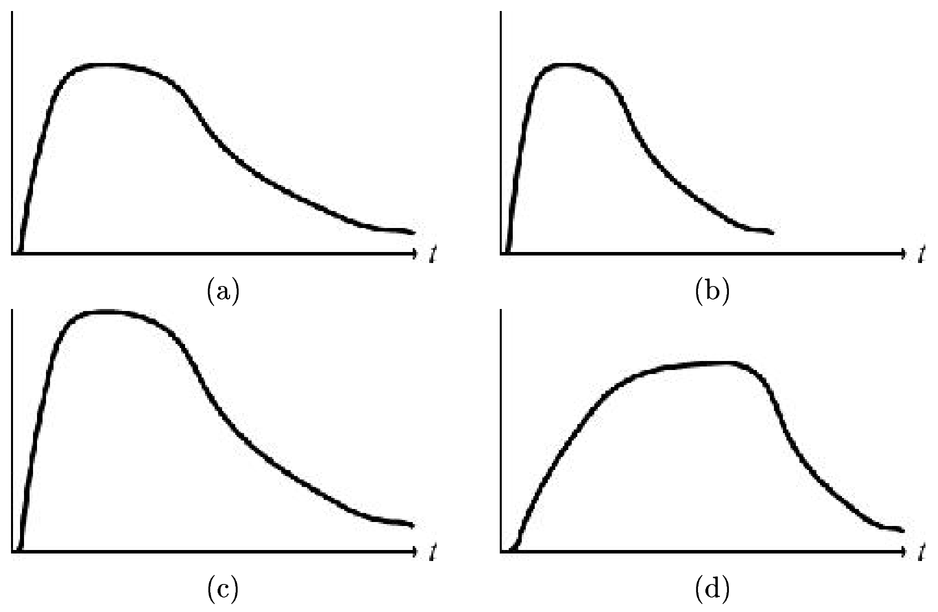


Figure 6.6: The transforms that can be applied to a piece of walking motion. (a) the motion represented as a 2D curve of the height of a heel over time. (b) the speed modifier changes the time taken by the motion, thus changing its speed. (c) transform extent changes the amplitude of a piece of motion (d) shift peak changes the motion so that its peak occurs later in the motion.

decide what actions should be performed. The *step over* agent notices when an obstacle is immediately in its way by checking the objects passed to it by *collision detector* and if so requests a step which will clear the obstacle by setting step length and height. It will try to step over any obstacle in its way so it must be subsumed by the next layer, *walk around* in the cases where the actor cannot or does not want to step over the obstacle. *Walk around*'s main function is to decide whether to step over an obstacle or walk around it. If it decides to walk around the obstacle it adds a new control point to the actor's path to make it pass around the obstacle. The top layer of the hierarchy is *avoid moving obstacles*; this checks for side-on collisions with moving objects and avoids them by either stopping to let the object go by or speeding up to pass in front of the object. The subsumption architecture works in such a way that each agent will make its own recommendation of an action and the higher agent will subsume the lower ones in such a way that an appropriate action is always chosen.

6.3 Attention leading behaviour: avoiding obstacles

The first aspect of navigating a cluttered environment without collisions is to know when there will be a collision with some object. Attention takes the lead in this task. Before

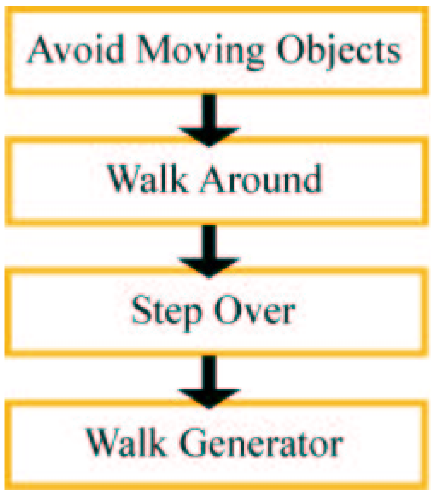


Figure 6.7: The agent hierarchy controlling the walking behaviour.

the actor is able to detect if there will be a collision with an object it must first be aware of the object and be attending to it, if the actor is looking in an opposite direction it will only know about the obstacle far later, possibly too late to avoid a collision. Thus the actor's pattern of attention goes a long way to determining its pattern of behaviour.

As described in chapter 4 the actor's attention is directed to a series of objects or locations sequentially. From a given centre of attention the actor's attention can be captured by nearby relevant objects, in this case objects with which there might be a collision (i.e. moving objects and objects in front of the actor). This is done by peripheral vision agents which monitor the periphery of the actor's vision for objects with certain features such as moving objects or objects directly in the actor's path. Once the actor is attending to an object it must determine whether there will actually be a collision with that object, the method used is described in section 3.5.1. Once it is known that there will be a collision the actor must take evasive action of some sort. There are various possible strategies, if the collision is going to be roughly side-on the actor can either stop to let the object go past or speed up to pass in front of it (this decision will be discussed in section 6.3.1). On the other hand if the actor is about to have a front on collision it can either change direction or, if the obstacle is relatively small, step over the obstacle (this is described in section 6.4).

6.3.1 Collision avoidance

Section 3.5.1 gives us a test for potential collisions with objects which is used here to safely navigate the environment. The actor's attention is captured by relevant objects. This is done by two peripheral vision agents (see chapter 4). One detects moving objects, the other detects obstacles that are in front of the actor. Objects that are detected are passed as requests to the attention manager and the actor will then fixate them. When the actor fixates an object it is passed to the *Collision Detector*. This sees if the angle of the object to its heading is constant over the period of fixation. As described in section 3.5.1 this is a test for potential collision. If this test is positive the Time-To-Contact is tested to see if it is positive, meaning that the object is moving towards rather than away from the actor. If all these conditions are met there will be a collision and action must be taken, so the object identifier is passed to the walking agents. [2]

Frontal collisions

A frontal collisions is defined as one where the angle of the obstacle's velocity to the actor's forward direction close to 180 degrees, or where the object's velocity is zero and the angle of the obstacles from the actor is less than 45 degrees. If the obstacle is in front of the actor, the actor will either have to step over the obstacle or walk around it. Section 6.4 describes how the actor makes this decision and steps over the obstacle. If the actor decides not to step over the obstacle but to walk around it the *Walk Around* agent merely has to add an extra control point to the actor's path, to one side of the object, so that the path passes around the obstacle. Which side the actor will pass the obstacle depends on a few heuristics. If the obstacle has a component of velocity perpendicular to the actor's heading the actor will go in the direction opposite to this component (i.e. it will pass behind the obstacle). Otherwise the direction will depend on the overlap of the actor and the obstacle. If only one side of the body overlaps with the object or in general if the centre of the object is to one side of the actor the actor will pass by the other way, in other words the actor will go the shorter distance (this, admittedly fairly obvious, heuristic is mentioned by Collett and Marsh[CM81] as having been observed in pedestrians). If the actor and the object are perfectly aligned (or close to perfectly aligned) so going either way is an equal distance the choice is arbitrary. Collett and Marsh mention that there appears to be a convention that pedestrians pass to the right. This could depend on nationality as they also noted that the convention in Australia is to pass to the left. This has been attributed to such unlikely causes as the reverse Coriolis effect, however, the more likely explanation is that it is due to the road traffic convention of driving on the left (British pedestrians, however, seem to pass to the right). In the current work the actor will make a random choice with a bias to the right.

Frontal collisions are shown in two figures: figure 6.8 shows the actor avoiding a collision with a static obstacle in front of it(figures 6.15 to 6.20 give more examples). In figure 6.9 two actors walk head on towards each other.

[2]In the case of objects that are directly in the actor's path this algorithm is not necessary for detecting a collision as there will always be a collision if the actor does not change direction. Using this procedure does ensure that the eye movments are suitable. However, if the peripheral vision agent that detects objects in front of the actor detects one that with which there is a low TTC (imminent collision) then it will pass its identifier directly to the walking agents. This ensures that the actor does not fail to react to objects it is aware of because it has not had time to fixate them.

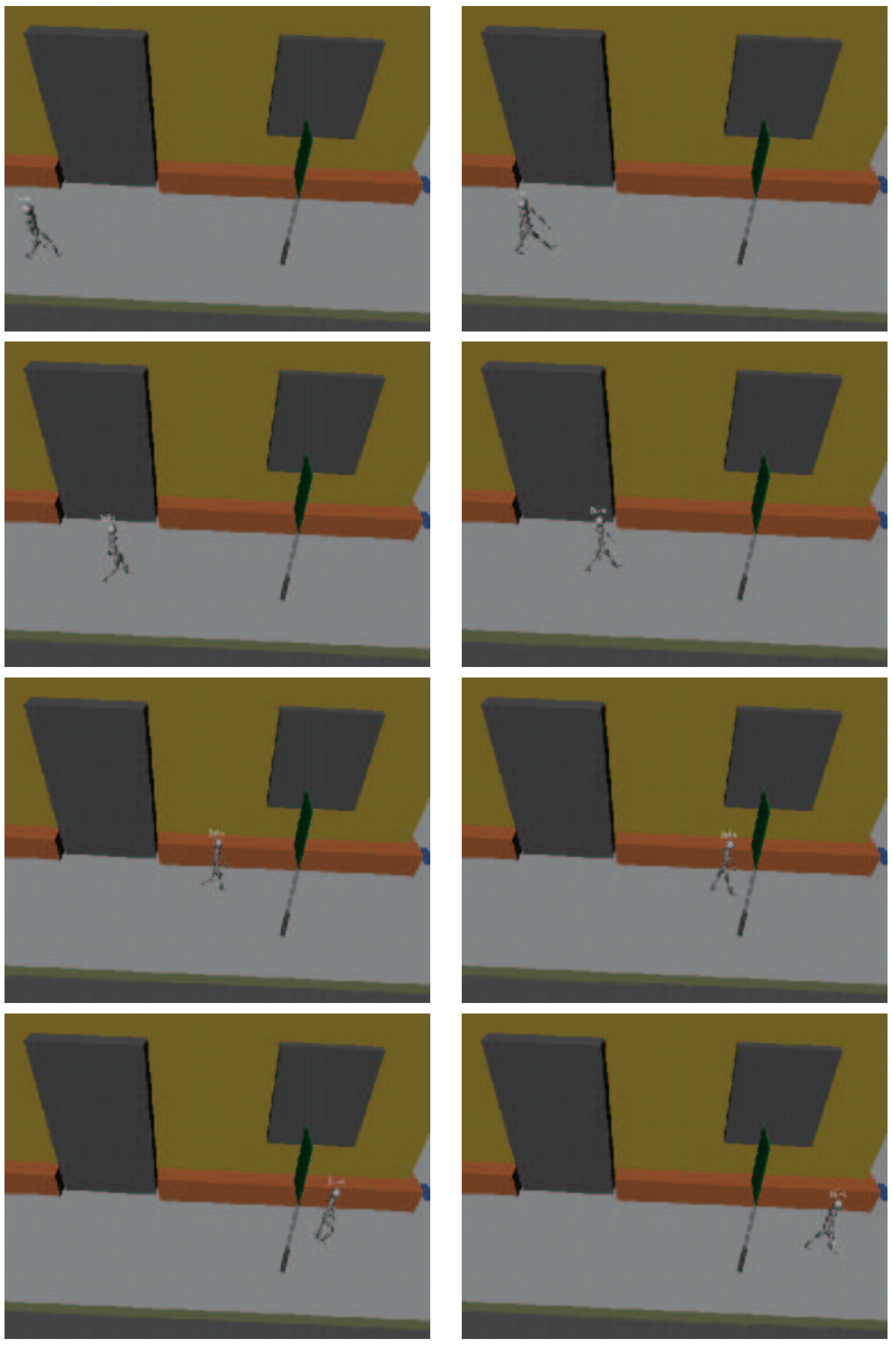


Figure 6.8: Here the actor alters its path to to avoid a static obstacle. This example is taken from the same animation as figures 6.15 and 6.16. (left to right, top to bottom)

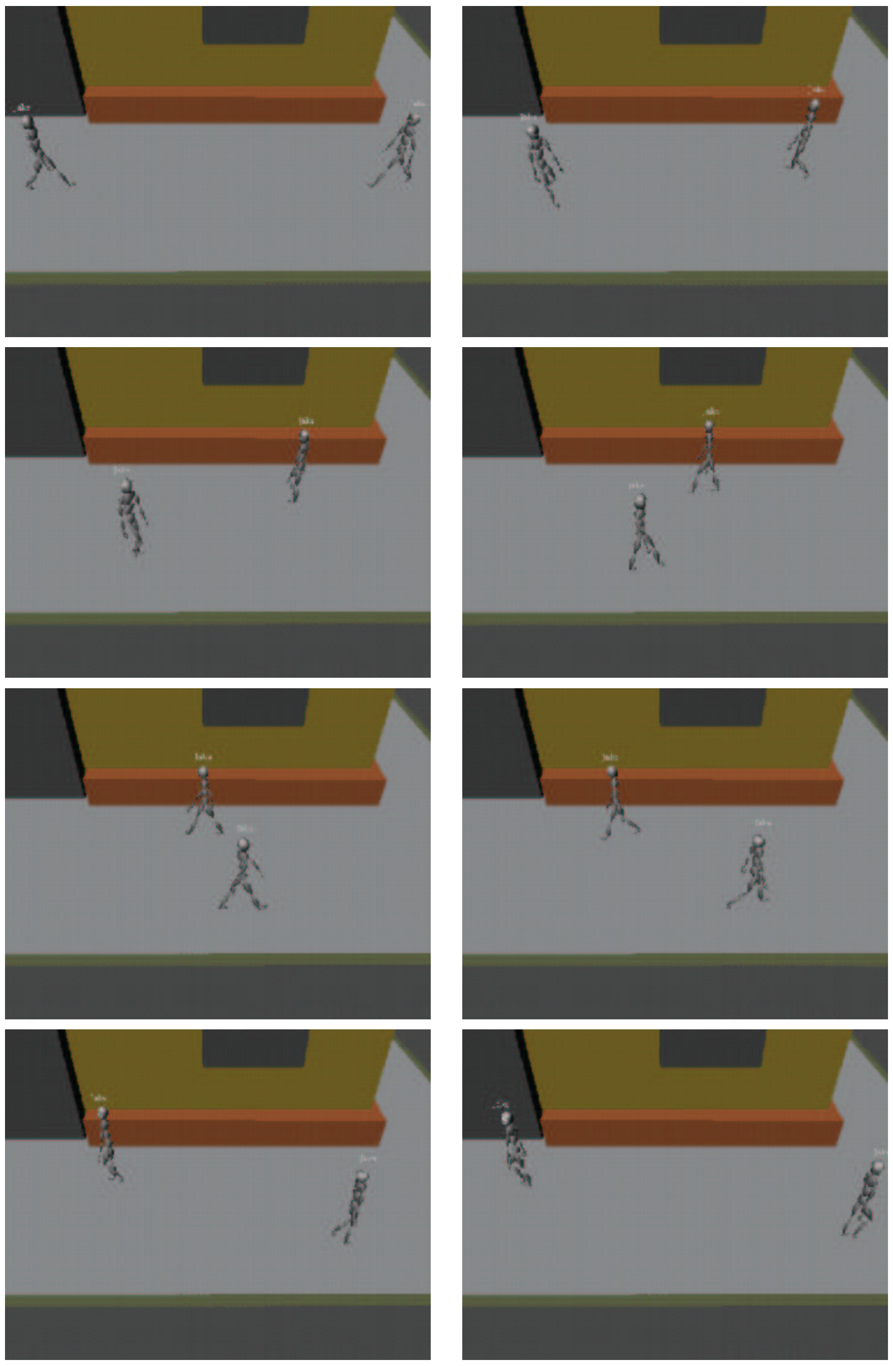


Figure 6.9: Two actors coming at each other head on. Both actor alter their paths to avoid a collision. (left to right, top to bottom)

Side on collisions

If the actor and object are on side-on collision paths the actor can either stop to let the object go by or speed up to pass in front of the object. Trying to pass in front gives a greater risk: running out in front of a car is dangerous. Whether the actor tries this depends on two factors: how far away the object is (i.e. how safe it is) and how willing the actor is to take risks. These are both expressed in the fact that each actor has its own threshold for what is a safe distance. So faced with a moderately close collision a cautious character might stop while a foolhardy one might run out but both would stop for a very close collision. Of course, distance on its own is not the correct measure as the speed of the object is also important. Clearly the judgement should be based on time to contact. If the TTC is above a certain value the actor will try to speed up to pass in front of the obstacle, as shown in figure 6.10. Otherwise, if it is too soon to stop, the actor might carry on walking normally for a while until the TTC goes below a second threshold when the actor stops. When the object has passed the actor or its TTC becomes negative the actor can start walking again. This behaviour is shown in figure 6.11.

6.4 Behaviour leading Attention: stepping over obstacles

We are trying to simulate how people notice and react to objects blocking their path. The actor must first decide whether it can step over the object or whether it must step around it. This is discussed in section 6.4.1. If it can step over it, it must decide when to take action. There are two parts to stepping over an obstacle, placing the feet in front of the obstacle so as to make the step as easy as possible, and then actually taking the step over the obstacle.

6.4.1 Deciding to step over

When faced with an obstacle in our path we have two options, we can either step over it or walk around it. The decision is based on many factors. The most obvious factor is whether we can step over it comfortably, without risk of tripping. This can be determined by the geometry of the object, its height and width. However, this is only one factor that will determines a person's behaviour. There might be a perceived risk in stepping over an object (is it a sleeping guard dog?), or the person might be old and infirm and find it difficult to take long strides or he or she might just make the decision on whim for no very good reason. Our actor makes the decision in two stages; a geometric calculation determines whether the actor can step over an obstacle while a random decision with probabilities dependent on the actor and object determines whether it will step over the object.

In order to judge whether to step over an object the actor uses body scaled measure. These are ratios of object sizes to sizes of features of the actor's body. They are described in section 3.5.1. The most useful ratios for deciding whether to step over an object are the height of the object scaled by leg length and the width of the object scaled by leg length.

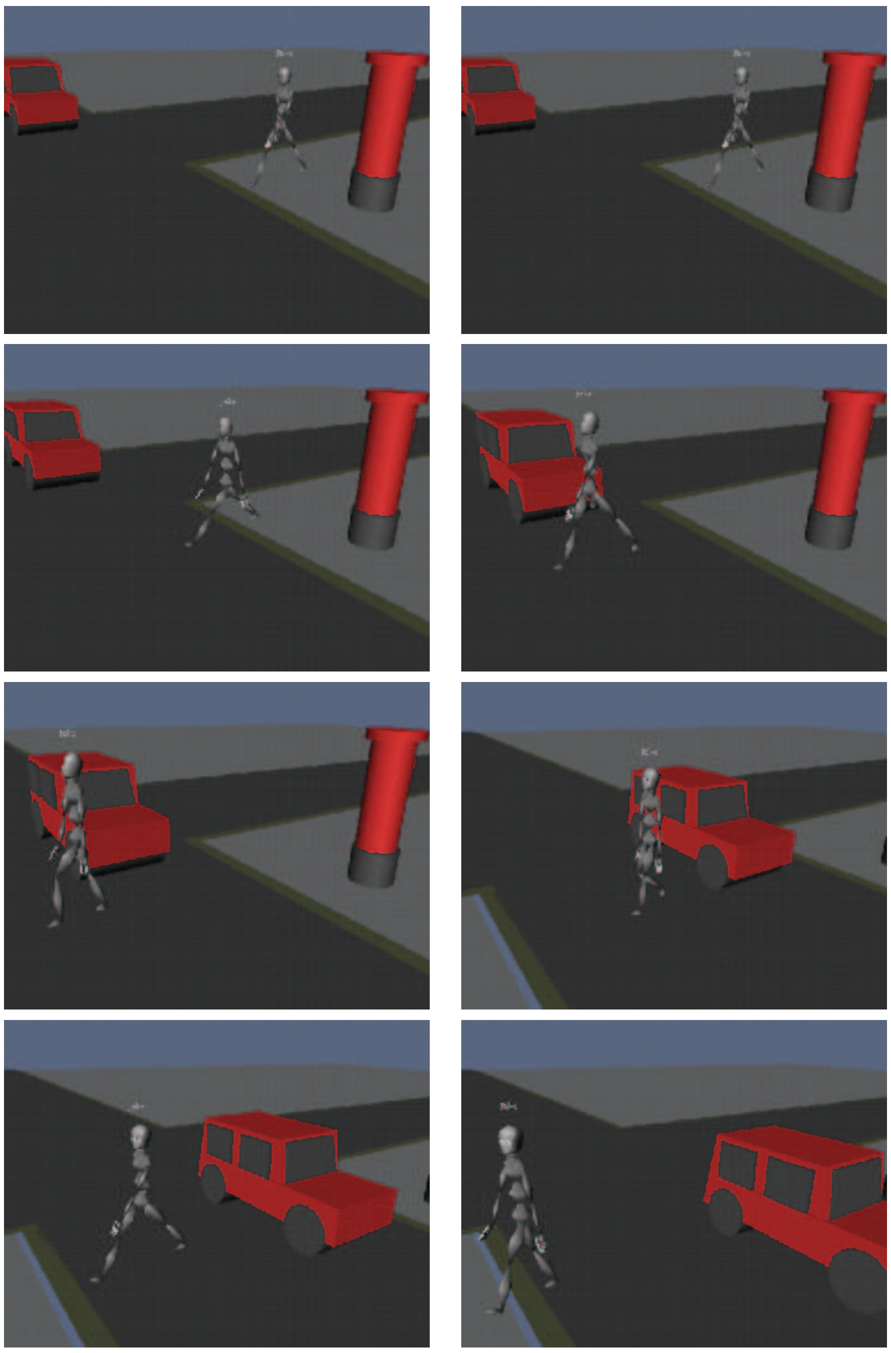


Figure 6.10: The actor avoiding a side on collision with a moving object. Note the increase in stride length as the actor walks faster. This example is taken from the same animation as figures 6.19 and 6.20 (left to right, top to bottom)

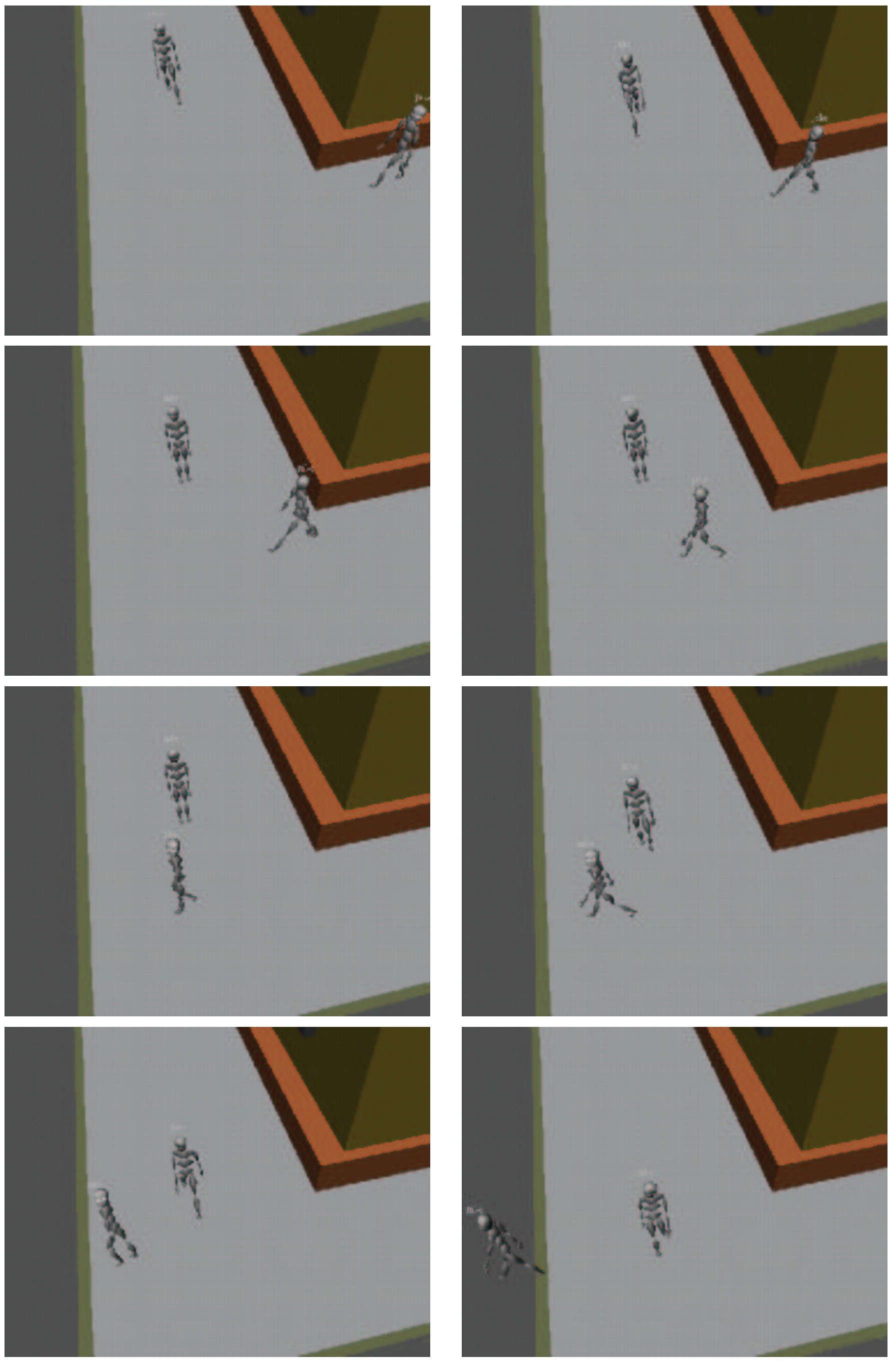


Figure 6.11: Here two actors are about to have a side on collision. One actor stops while the other walks by. (left to right, top to bottom)

Once these are know the actor can just compare them to a threshold value. These are two measure that humans use in making judgements as described by Lee[Lee80].

Before each step the actor checks the objects in its way as described and selects the one that it must react to. Before taking action it must decide whether to step over it or not, based on the height of the obstacle. Based on average body measure and the most extreme stepping position, Warren calculates the maximum height a person can step to be about 88% of leg length[War84]. However, this is an extreme measure and much higher than a person would step during normal walking. It is not a comfortable stepping height and it would leave the person unable to take a normal step afterwards. A lower value has to be chosen as the threshold for stepping over during continuous walking. Studies have shown that while crossing an obstacle the foot can pass over it by a height of as much as 45cm which is almost half the average leg length. Taking this into account and subtracting a comfort factor so that the leg never gets too high an object is defined as affording stepping over if its height scaled by leg length is less than about 40%. This proportion, called the maximum step height, can vary from character to character between 20% and 50%.

When the actor finds an object it must avoid as described in section 6.4 it checks the height of the object scaled by the lenght of the actor's leg. If this is less than the maximum step height it can be stepped over, otherwise the actor has to alter its heading direction to step around it.

The size of an obstacle is not the only factor on which people base their decisions of whether or not to step over something. Some one might be more likely to step over a cardboard box than a vicious dog even though they are the same size. It would be too complex to model all of the factors in these decisions; so, as well as testing the size of an object, a random factor is introduced into the decision. The random decision is made with a probability based on two factors, one coming from the actor, the other from the object. The first represents the character's tendency to step over objects in general; thus an older character whose movements are more uncertain would be less likely to take the risk of tripping entailed in stepping over an object than a young character. The other factor represents how likely characters in general are to step over a specific object, so people would be less likely to step over a sleeping guard dog than a cardboard box. These two factors are represented as object features (see section 3.4.3). These can be changed by the animator, changing the actor's behaviour relative to certain objects. These two factors can be set by the animator for each character and each object, controlling how the actors make their decisions.

This decision is made by the *Walk Around* agent (see section 6.2). If the agent decides to walk around the obstacle Walk Around itself will perform the action, otherwise the obstacle will be passed to the *step over* agent to determine how to step over it.

6.4.2 The step over agent

The *step over* agent takes an object that it knows it has to step over and then calculates the actual modifications the actor has to make to its regular walking gait in order to step over the obstacle. The implementation of the actual animation of the actor stepping

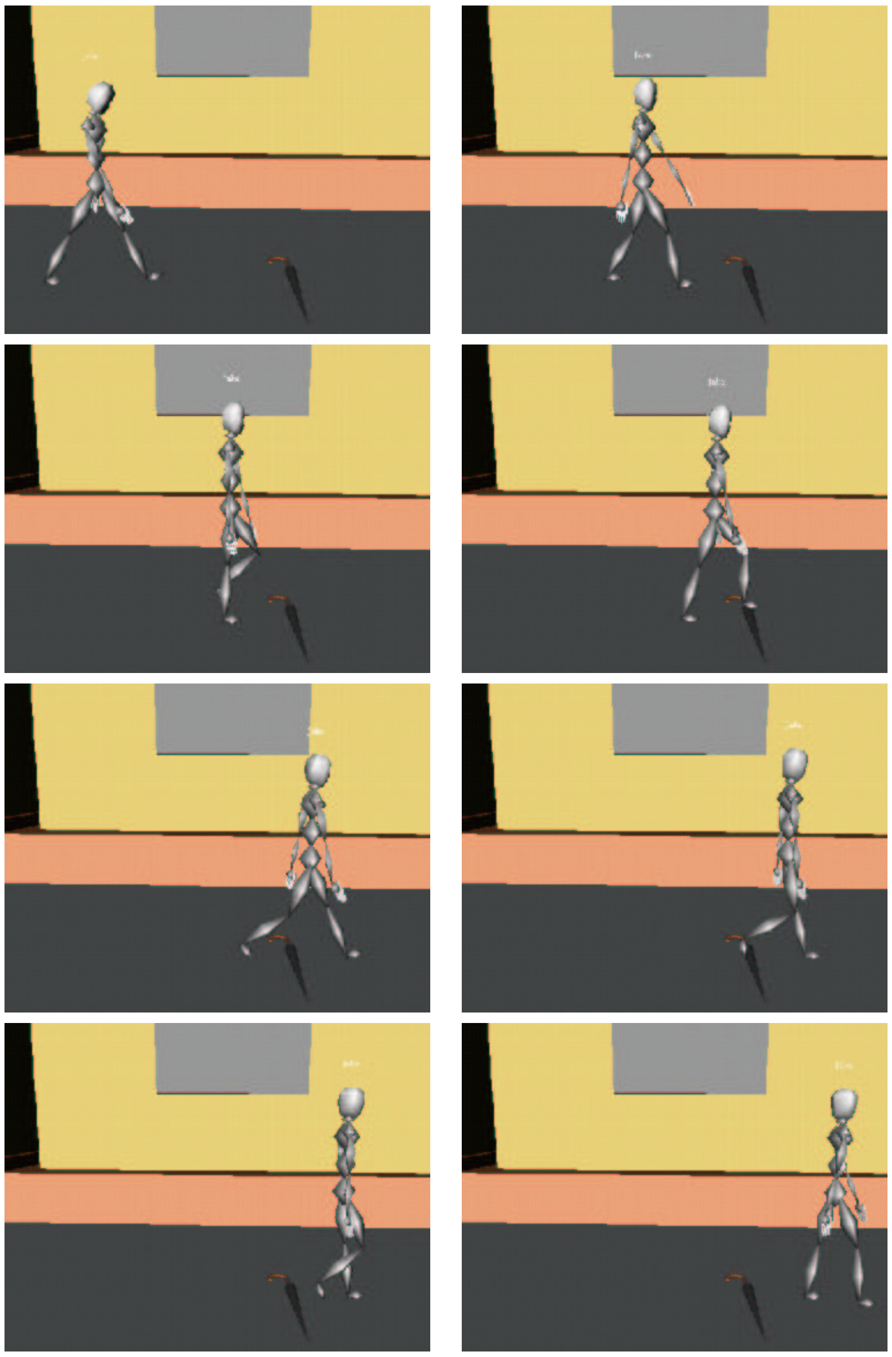


Figure 6.12: The actor stepping over an obstacle. This example is taken from the same animation as figures 6.15 and 6.16. (left to right, top to bottom)

over an obstacle is based on the results of a study by Sparrow, Shinkffield, Chow and Begg[SSCB96]. When stepping over obstacles people place their feet fairly accurately in the step before crossing the obstacle so that in the following step the first foot crosses the obstacle roughly 80% of the way through its stride and the second one 35% of the way[SSCB96]. This placement has to be done in the foot step before the step in which the actor will cross the obstacle. This means that the actor has to have a means for knowing how many steps ahead it will step over the obstacle. Measuring the distance to the obstacle in terms of stride length gives this simply. In fact this can be done purely in terms of the difference between Time-To-Contact of the obstacle and the time-to-contact of the foot. As the actor starts a stride it calculates the TTC of the position where the foot will come to rest. It then scans the objects in front of it and takes the difference between their TTC and the TTC of the end position of the foot. If, for any object, this is between one and two times the normal stride time t_s, the actor must take action, adjusting the length of the *next* foot step. The length is scaled by the ratio of the TTC of the correct foot position to the normal stride time. The actual scaling is done using the *transform extent* transform.

For the actual crossing step both the step length and step height have to be controlled so the obstacle can be cleared. The step length can be controlled as above. The height of the obstacle is known in terms of the eye height of the actor and so the height as a proportion of leg length is known (for a given individual leg length is a fixed proportion of eye height). Therefore, the height is controlled by simply raising the foot by an appropriate proportion of leg length. The trail foot (the foot which crosses the obstacle second) clears the obstacle by a greater margin than the lead foot (an average of 50cm against 15cm for the lead foot). This is an observed fact[SSCB96] which is thought to be due to the fact that the trail foot is behind the person and so invisible during the step and so it needs to be lifted higher to make sure it clears the obstacle.

The height of the step is increased using the *transform extent* transform. It is also important that the foot reaches its highest point at the correct time in the step. In the original motion this peak will be at some arbitrary point of the motion, normally fairly early. For the stepping over motion the peak has to coincide with the time that the foot passes over the obstacle. The peak is moved using the *shift peak* transform. These transforms are described in section 6.2.1.

Patla, Prentice, Robinson and Neufield[PPRN91] have shown that people plan footsteps two steps ahead, i.e. while they are taking the previous step with the same foot, obstacles which they become aware of after this time are harder to step over. Thus all of the calculations described in this section are done at this time; this is a matter of convenience of course, but it allows us to produce appropriate gaze patterns at the correct time. Once the appropriate step has been planned the parameters are saved and then passed to the *walk generator* agent when the step actually has to be made. This agent then performs the motion transforms which create the final shape of the walk cycle (this is discussed in more detail in section 6.2.1).

6.4.3 Eye movements while stepping over objects

The *step over* and *walk around* agents also generate appropriate eye movements while coming up to an obstacle. These agents first become aware that the actor has to do something about an obstacle when the obstacle's identifier is passed to them by the *Collision Detector* agent (see sections 4.4.1 and 6.3). When this happens *walk around* immediately checks if it has to do something immediately; if not it sends the *Attention Manager* a monitor request for the object. Patla and Vickers [PV97] have shown that people tend to look at an obstacle in the two steps before stepping over it but not while actually stepping over it. This agrees with the finding that steps are planned two steps ahead and the fact that the obstacle is either under or behind a person's head for most of the time the person is stepping over it. To simulate these eye movements the *step over* agent sends an immediate request to look at the obstacle while it plans the step (this makes the actor look at the obstacle immediately, see section 4.4.3 for more details). It also sends a monitor request, these might make the actor look at the object again as it approaches, depending if there is anything distracting the actor's attention. Finally when the actor starts to step over an obstacle all requests to look at it are purged from the monitor queue so that the actor does not look at it any more.

6.5 Discussion

This chapter shows how vision and attention can be applied to a commonly studied problem in behavioural animation: navigating a cluttered environment, and improve on current methods. There are three aspects of this work that need to be discussed: the aspects of the walking motion that are unconnected with vision and attention; the use of attention and vision to produce the navigation algorithms, and the generation of eye movements from the behavioural algorithms.

6.5.1 The walk generator

The algorithms described in this chapter allow a piece of walking motion to be adapted to a wide variety of circumstances. Figures 6.8 to 6.12 show the motion produced in various collision conditions: a potential collision with a tall static obstacle; a head on collision; a side on collision where the actor has to stop to let the other actor go by; a side on collision where the actor can speed up to pass in front of the other actor, and finally an obstacle that the actor can step over. Figures 6.15 to 6.24 show longer examples. Animated viersions of these examples are available on the accompanying CD-ROM. The motion seems reasonably convincing. The stepping over motion algorithm was designed based on psychological observations in a similar way to the vision algorithms, as stepping over obstacles has been extensively studied. The other situations are more complex and have been less studied so this was not possible. However, they still give good results over a range of walking behaviour at real time speeds.

There are, however, a number of problems with the current walk generation algorithms.

Firstly the ground agent that keeps the feet planted to the floor and ensures an appropriate forward motion of the actor's body during walking will always keep one foot on the floor. This rules out running motions where both feet leave the ground (this problem was noted in section 3.5.2). This could be solved by including some representation of feet leaving the ground in the motion representation itself.

Basing the walking motion on a single piece of forward walking motion is probably not the most realistic method. In particular large turns can be unrealistic, for example figure 6.19 frame 2. This could be tackled by using motion of an actor turning as well as the forward walking motion, this is discussed further in section 7.4.3. However, these detailed features of the motion are less important to the current work than the high level control of the motion.

Using a parametric curve to is a good way of controlling the walking motion of a character. It is a simple representation that automatically gives a smooth path. It is a more longer term method of control than planning a step at a time and so will produce more consistent walking paths. The path can easily be altered in real time by adding, removing or changing control points. Also the path representation is independent of the method used to generate the path. These last two features make it possible to integrate different path generation techniques, in particular, path planning and reactive collision avoidance. The reactive walk generation described here could be used to adapt the output of a path planner if that output is in the form of a parametric curve (getting a planned path in this form would simply be a matter of fitting a curve to the output of a planner). However, a smooth curve might not be the best method of interpolating a sequence of control points for walking. In particular, a sharp turn can produce a large bulge in the curve in order to maintain continuity. Finding a better method of interpolation would involve a detailed study of people's walking patterns.

6.5.2 Use of vision and attention

Using a psychologically based model of vision has helped produce an algorithm for collision avoidance which is effective and works equally for static and moving obstacles. Without the simulation of attention this method would not have been feasible. It would have involved monitoring every object in the scene and maintaining a value for the angle of each to the actor's heading. This would have been a large time and space overhead. Also it might have led to a noisy situation where many collisions would have been detected, many false. However, the attention simulation allows the tests to be performed on one object at a time. It also ensures that the actor is looking at the object at the time (which was part of Cutting, Vishton and Braren's[CVB95] theory of how collision detection was performed). This adds to the realism of the behaviour produced as the actor shows the outward signs of being aware of the objects that it avoids collisions with.

Figures 6.15 to 6.20 show the actor navigating a fairly typical street scene. It successfully avoids collisions with objects in its path such as the rubbish bin or the post box. It steps over an abandoned umbrella and avoids colliding with a car and another

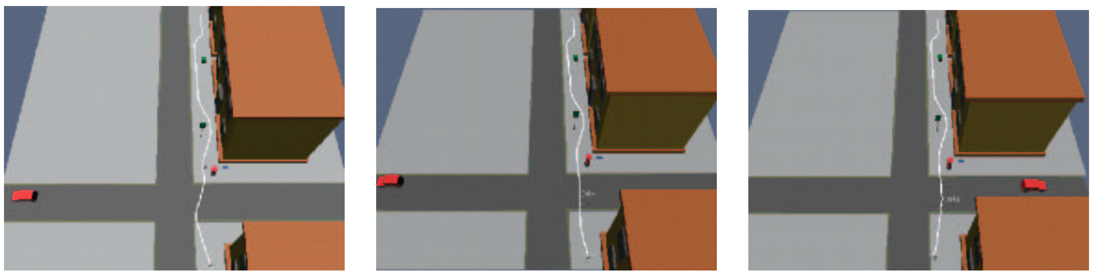


Figure 6.13: The walking paths taken by the actors in figures 6.15-6.16, 6.17-6.18 and 6.19-6.20 respectively

actor.[3] The navigation behaviour changes depending on what the actor is aware of. In figure 6.20 the actor looks around more and the car is not obscured by the building so the actor is aware of it earlier. This means that it can speed up to pass in front of it rather than stopping (it has a low threshold for passing in front of objects and so passes riskily close to the car). Figure 6.13 shows the paths taken in each case.

Figures 6.23 and 6.24 show an example of four actors walking around an environment containing a number of 'trees'. The actors mostly manage to avoid the trees. There is one case, however, of the actor looking in completely the wrong direction and walking into a tree, as shown in figure 6.25. This shows that attention model can simulate what happens if people do not look where they are going. [4]

Figure 6.26 show a number of problems with the use of attention to navigate. The environment is dense with objects and a complex path is needed to navigate it, taking into account a number of different objects at the same time. The attention mechanisms only deal with one object at a time and reactive collision avoidance is too limited to take a large number of obstacles into account. Planning is needed to determine a path. If the environment is too complex a planner could be invoked to generate a path and then control could be returned to the reactive walking agents. In figure 6.27 planning is simulated by adding a control point (the gold disc) that ensures the actor avoids the walls of the flat and successfully navigates the environment.

[3]In figures 6.16 and 6.18 the other actor was set up so that it would not attempt to avoid a collision with the first actor in order to make the to make the first actors collision avoidance more obvious. If this was not done the second actor, not being distracted by other objects in its path, would have noticed the first actor earlier and would have steered away early enough that the two would not have approached closely at all.

[4]In the current system when two objects collide they interpenetrate. Though detecting when they collide would not be hard having the actor react appropriately would itself be a difficult research problem. It would involve integrating the dynamics of the collision with the actor's attempts to stop itself falling and return to normal motion

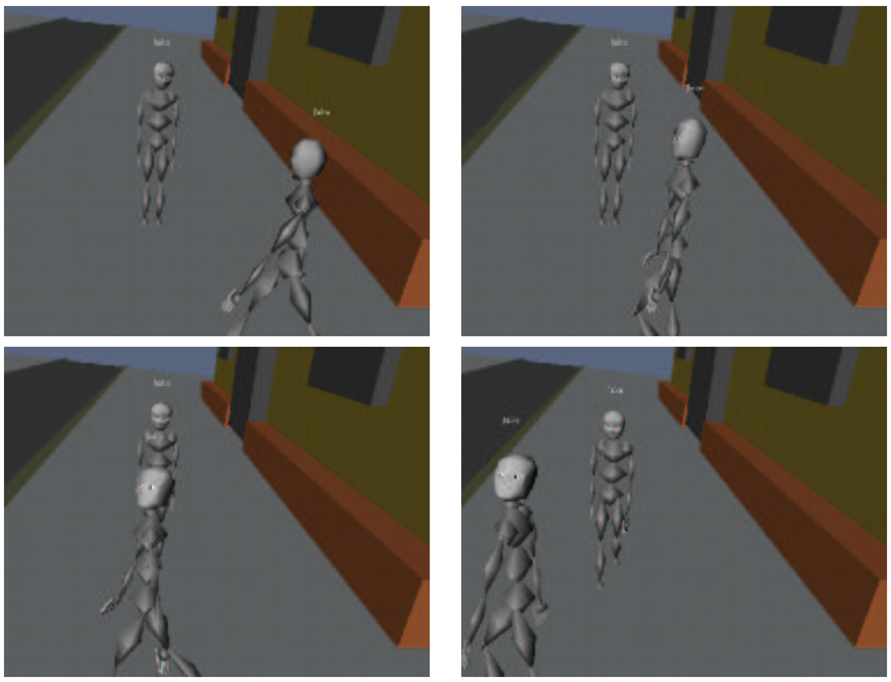


Figure 6.14: The central portion of the animation in figure 6.11, showing the stopped actor tracking the position of the other actor with its gaze.

6.5.3 Eye movements

Part of the aim of using attention in the navigation algorithms is to produce eye movements which give the outward appearance that the character is aware of the objects in its path and avoiding them rather than just walking around a set path. This means that the navigation algorithms themselves should request appropriate eye movements. Figures 6.9 to 6.20 show the eye movements of the actor while performing the various obstacle avoidance tasks. In general, the actor will monitor objects that it might collide with, showing that the character is aware of it. Sometimes, there will be an immediate request as the actor starts to deal with an obstacle, ensuring that the actor looks at the obstacle at the appropriate time, consistent with, for example, Patla and Vickers' [PV97] observations. This is shown in the stepping over example (figure 6.12) where the actor looks at the obstacle as it is about to change its step length (frame 1).

Another feature of the eye movements when avoiding collisions is that the minimum distance for a request is quite high, ensuring that the actor only looks at the obstacle as it approaches. The actor stops looking as it actually passes the obstacle and looks to next potential collision. Another feature to note is in figure 6.11 the stopped actor follows the walking actor with its gaze, this is shown in more detail in figure 6.14.

Figures 6.15 to 6.20 show longer examples. They show the same environment but with different settings for the eye manager. In figures 6.17 and 6.18 shows the scene with the same settings as figure 4.13. The actor has a high probability of looking in the default direction, in this case forward. The preferred gaze angle is low and the actor will tend to make long gazes to locations near the default direction and sort glances at the rest environment. Figures 6.19 to 6.20 are the opposite, the actor tends to look around itself more with longer looks than those to the forward location. The actor also tends to have a higher gaze. These settings are the same as figure 4.11. Figures 6.15 and 6.16 show more average settings but with a high probability of glancing. Figures 6.21 and 6.22 shows the same example as figures 6.15 and 6.16 but with out any eye movements. The actor appears more "robotic" and less involved with its environment. (As the behaviour partially depends on the direction of the actor's gaze the path taken is not identical).[5].

6.6 Summary and conclusion

This chapter has applied the simulation of vision and attention to navigation in a cluttered environment. It has described algorithms that have a number of advantages over existing methods

- The control of motion using a parametric curve and motion warping produces good motion in a range of circumstances. It is also a method of control that can be independent of the method of path generation and so could be used to integrate path planning and reactive walking.
- The simulation of vision and attention make it possible to produce an algorithm, based on real human behaviour, for collision avoidance which is efficient and handles both static and moving obstacles. It reacts appropriately based on what the actor is aware of and when it becomes aware of it.
- The integration of attention and navigation behaviour make it possible to accompany the navigation behaviour with suitable eye movements. The navigation behaviour both influences and is influenced by the eye movements. When an actor avoids an obstacle it looks at it appropriately, and the direction of the actor's gaze determines what it is aware of and therefore which objects it reacts to. The eye movement add personality to the animation and also a sense that the actor is involved with the environment as can be seen by comparing figures 6.15 and 6.16 and figures 6.21 and 6.22.

[5]Some of these examples would benefit from the use of heuristics specific to the situation. For example, when walking down a street it is customary to walk roughly parallel to the pavement and to avoid walking on the road if possible. Also it is only necessary to check for collisions with cars when about to cross a road. When two people cross on a street there is a lot of subtle body language and gaze behaviour that is not modelled here. These possible improvements are discussed in section 7.4.4

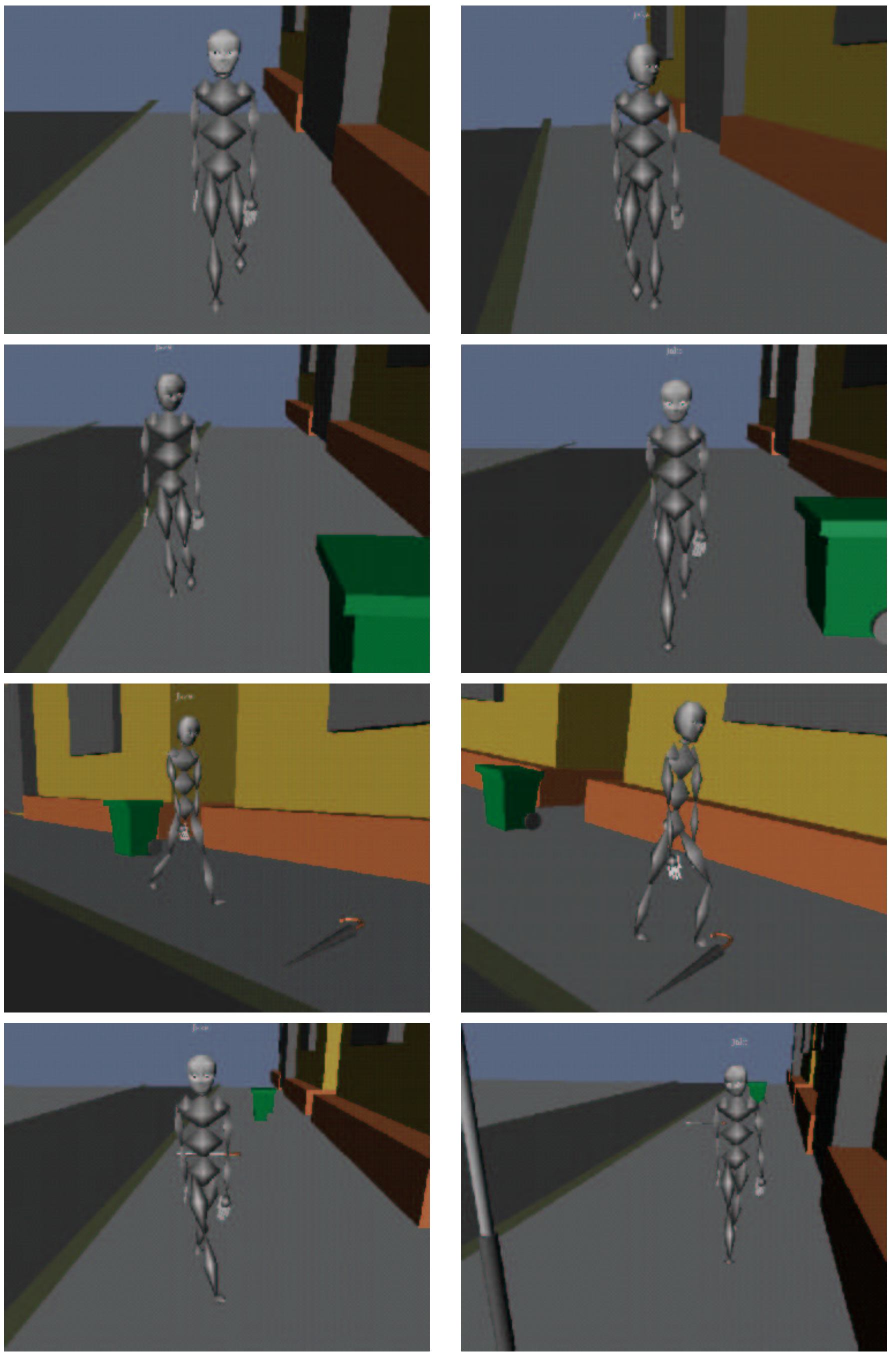


Figure 6.15: A longer example of the actor walking along a street. The actor has a high probability of glancing. (left to right, top to bottom)

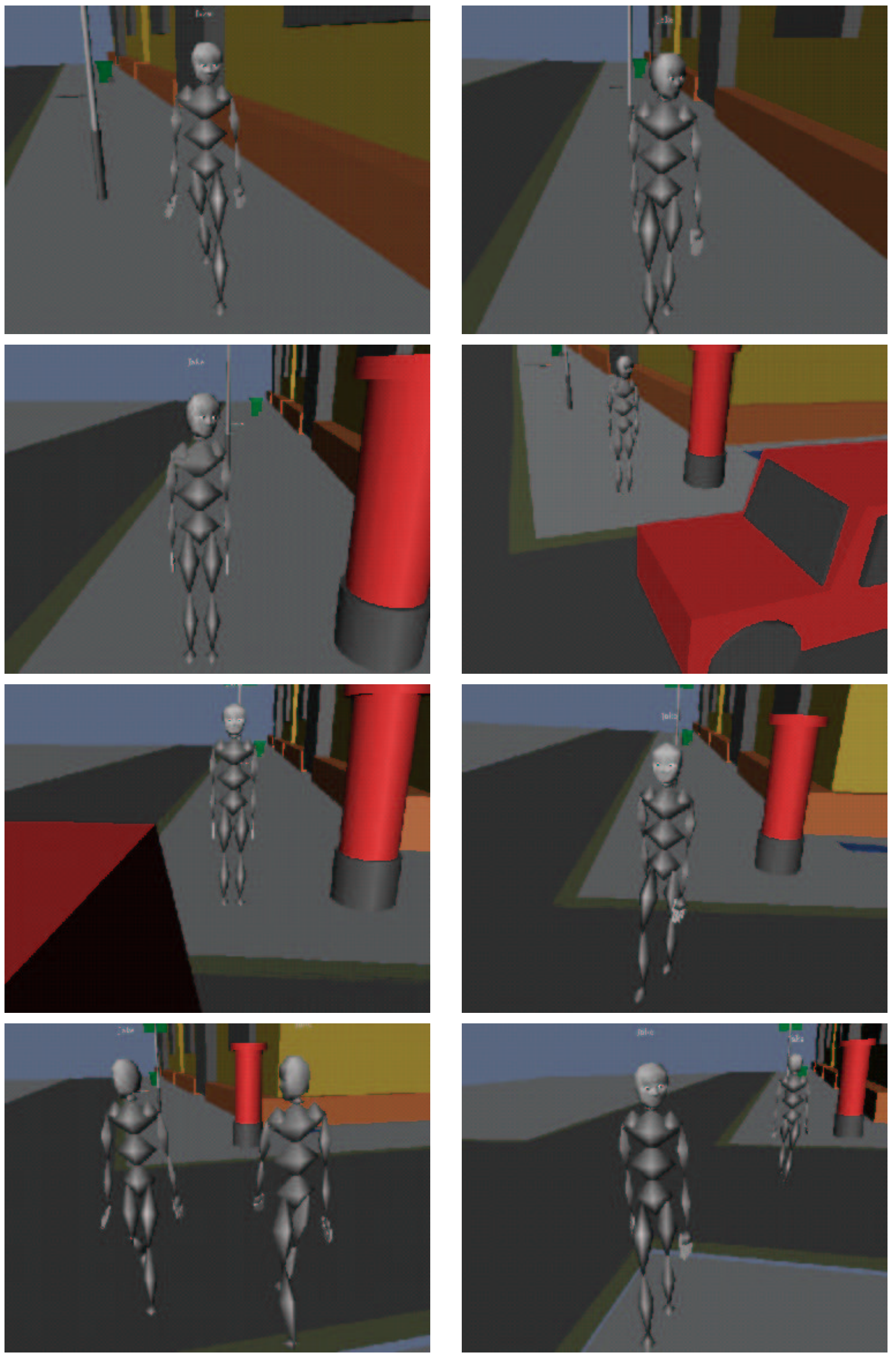


Figure 6.16: The continuation of figure 6.15. (left to right, top to bottom)

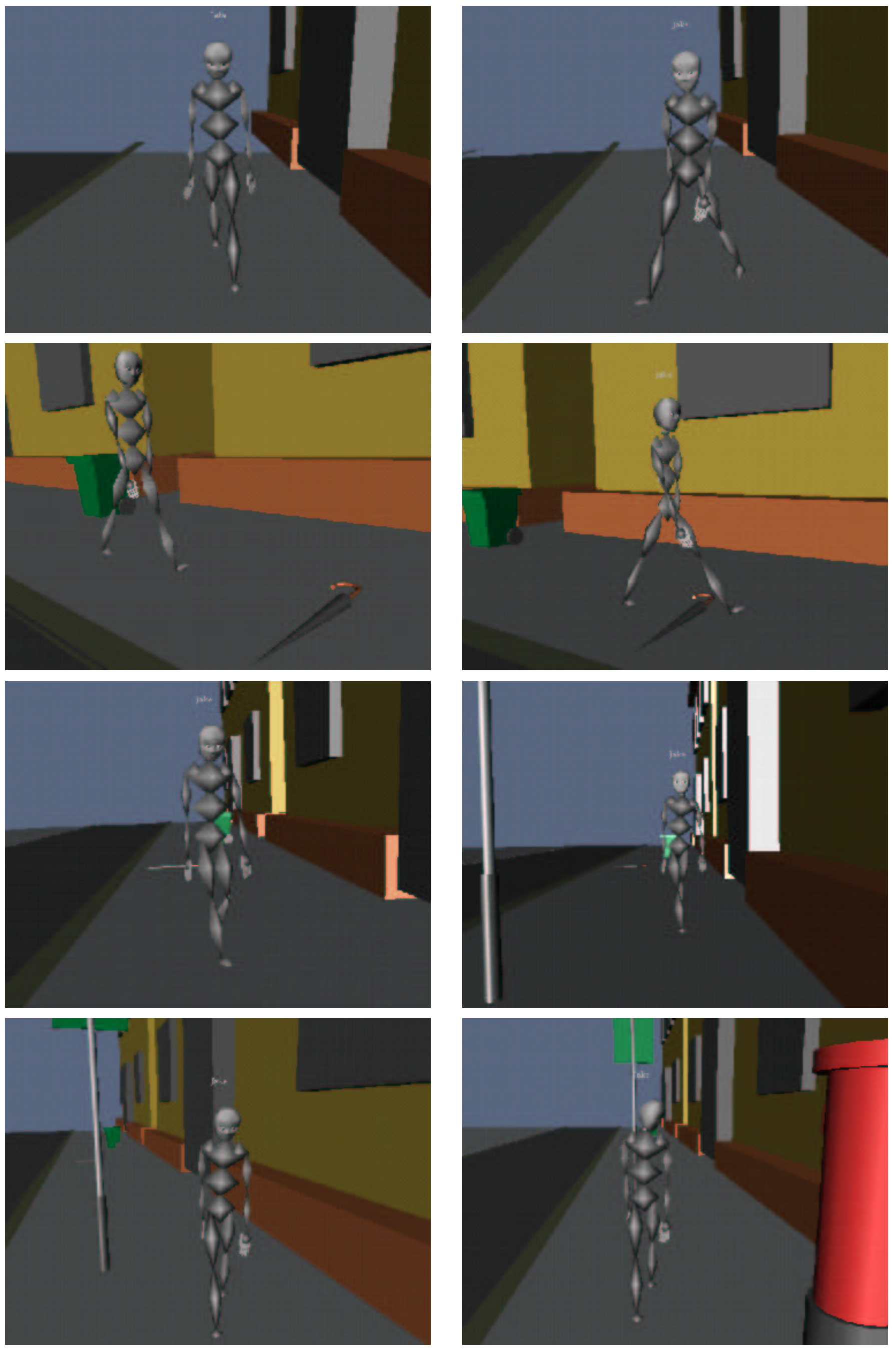


Figure 6.17: A similar scene to figure 6.15 but with different parameters settings. In this case the actors attention parameters have been set so that it does not look around itself often and it looks primarily at the ground in front of it. The parameters are the same as figure 4.13 (left to right, top to bottom)

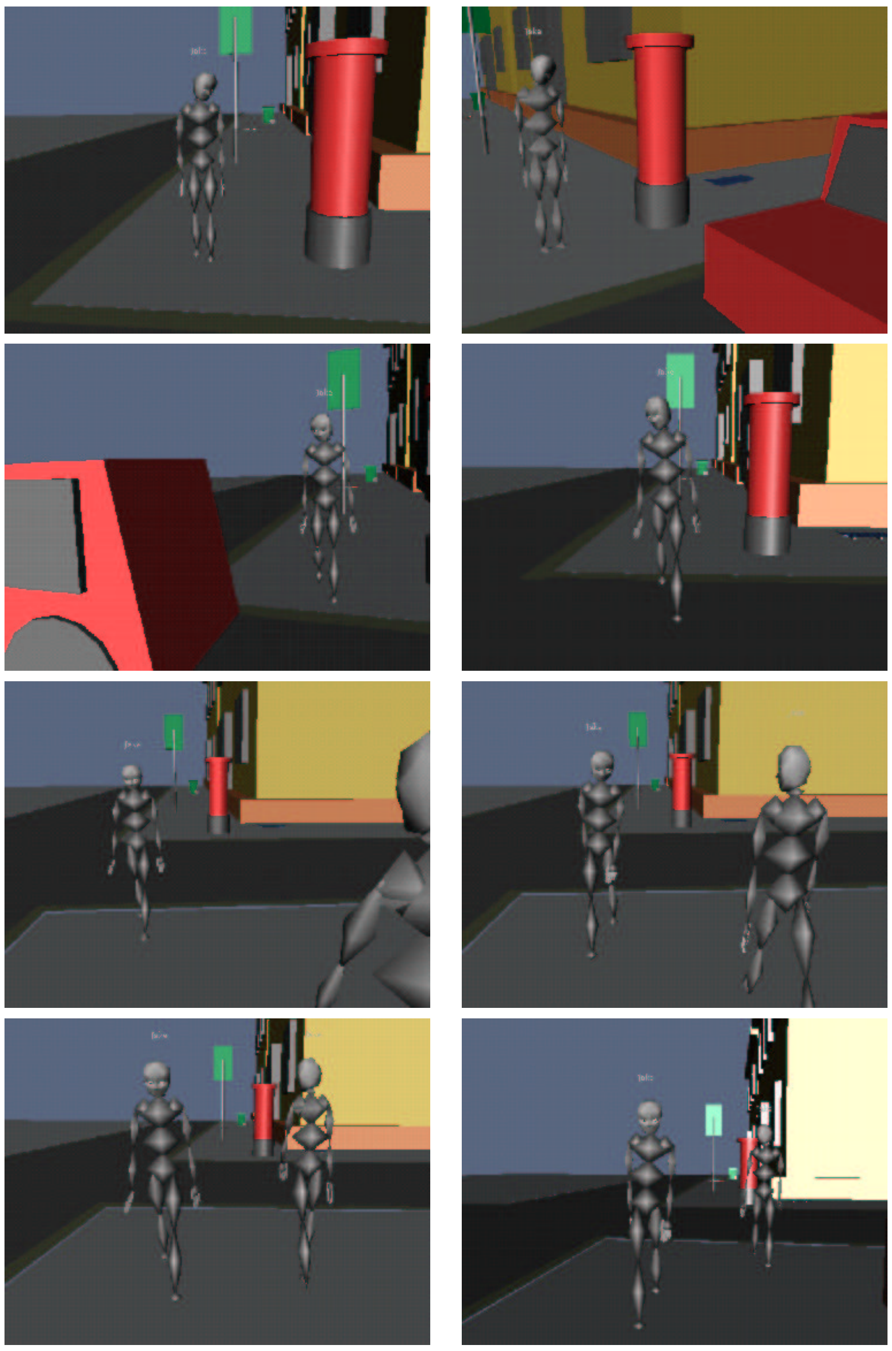


Figure 6.18: The continuation of figure 6.17. (left to right, top to bottom)

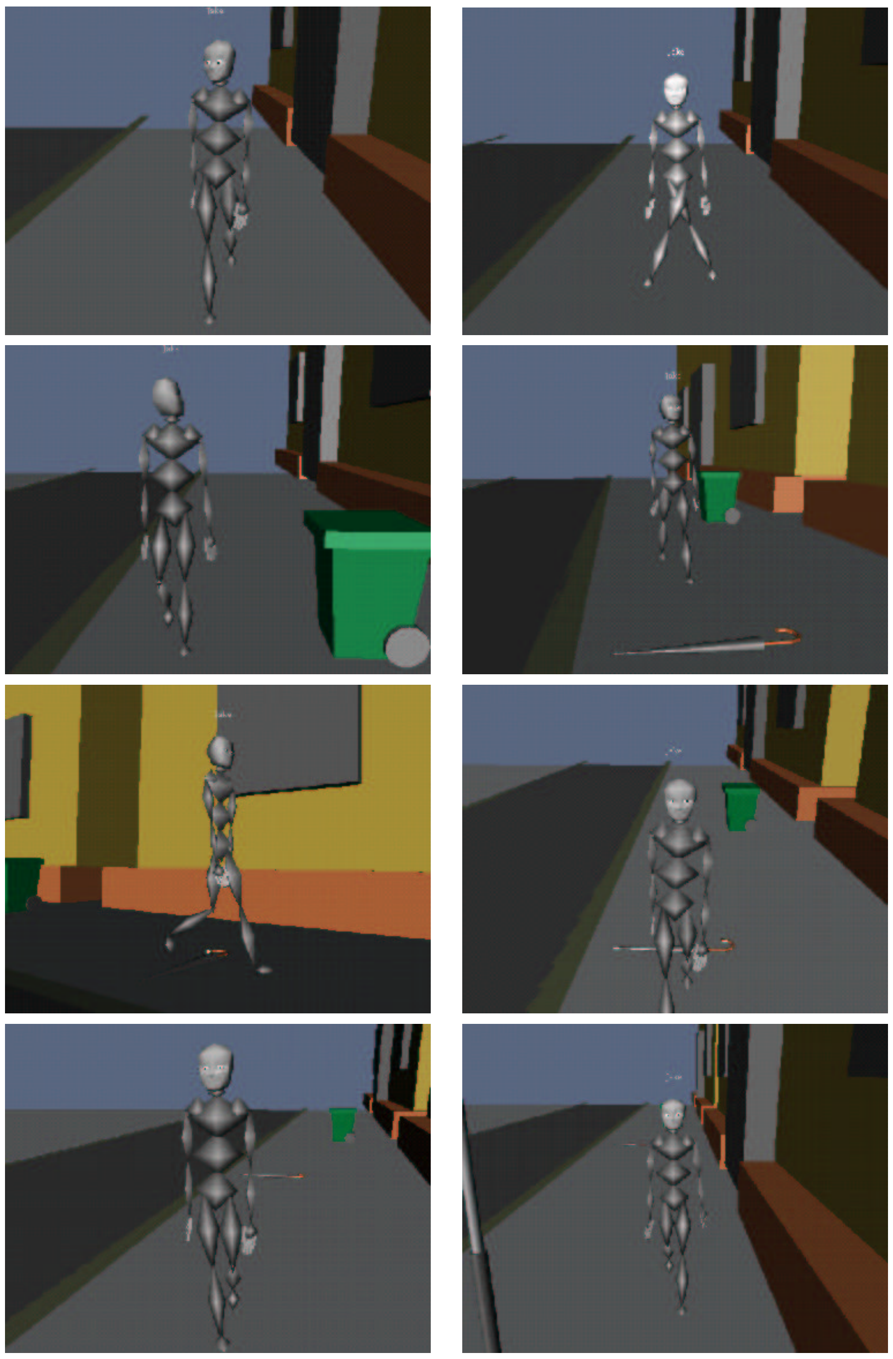


Figure 6.19: A similar scene to figure 6.15 but with different parameters settings. In this case the actors attention parameters have been set so that it looks up and around itself frequently and does not monitor the ground in front of itself often. The parameters are the same as figure 4.11 (left to right, top to bottom)

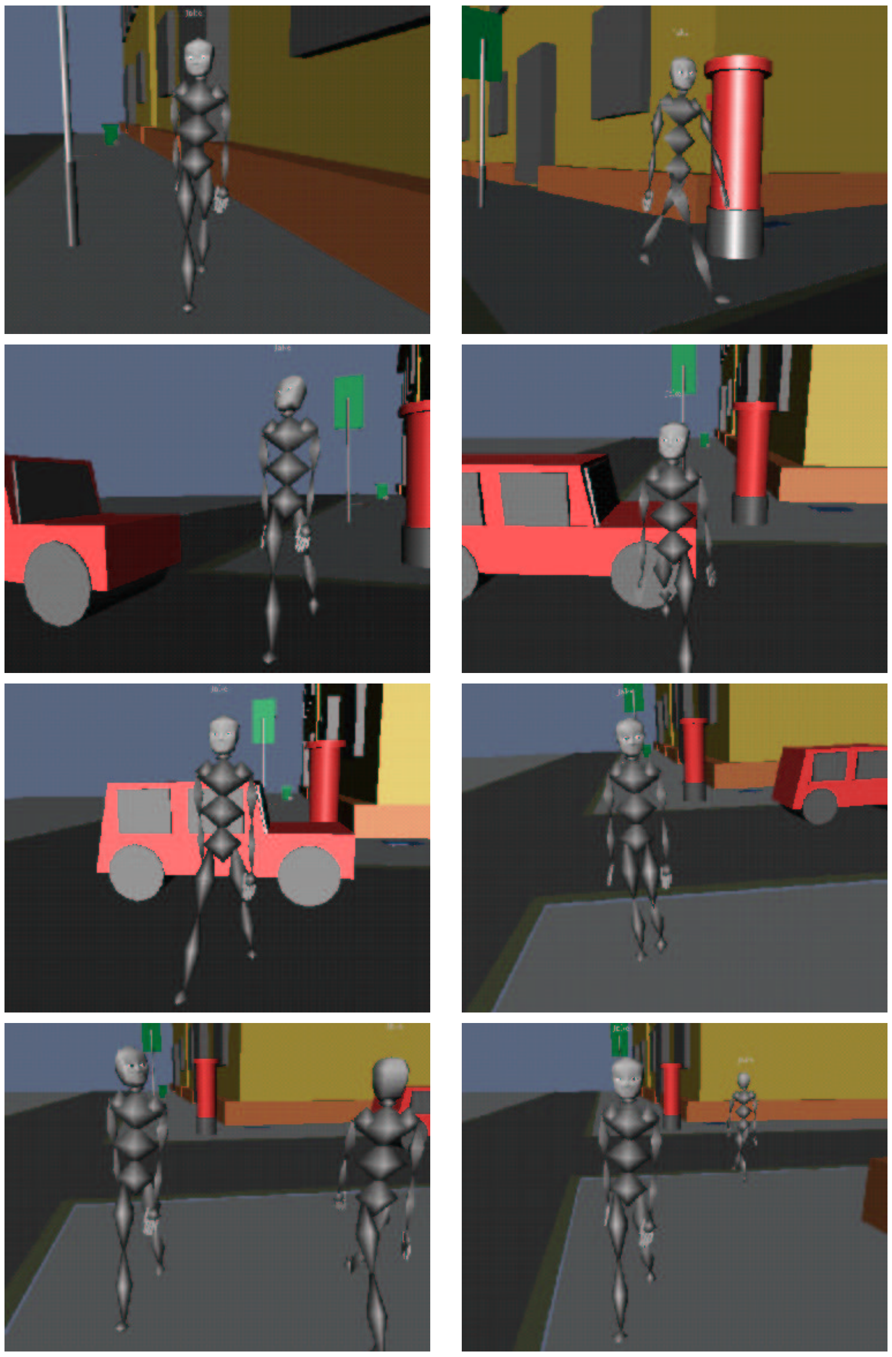


Figure 6.20: The continuation of figure 6.19. As the car is not obscured by the building the actor sees it earlier and can speed up to avoid it. (left to right, top to bottom)

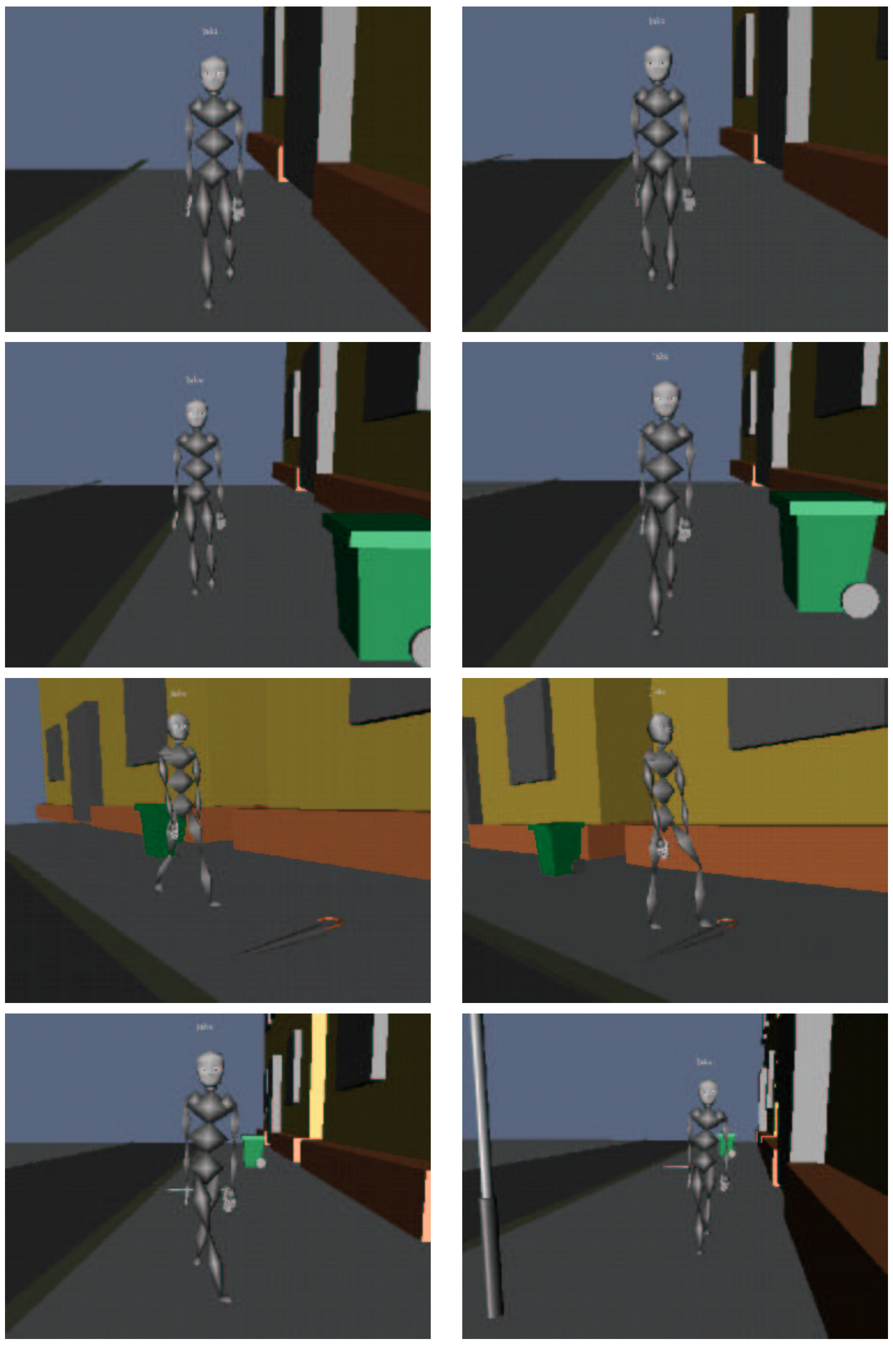


Figure 6.21: The same walking behaviour as figure 6.15 but without eye movements. (left to right, top to bottom)

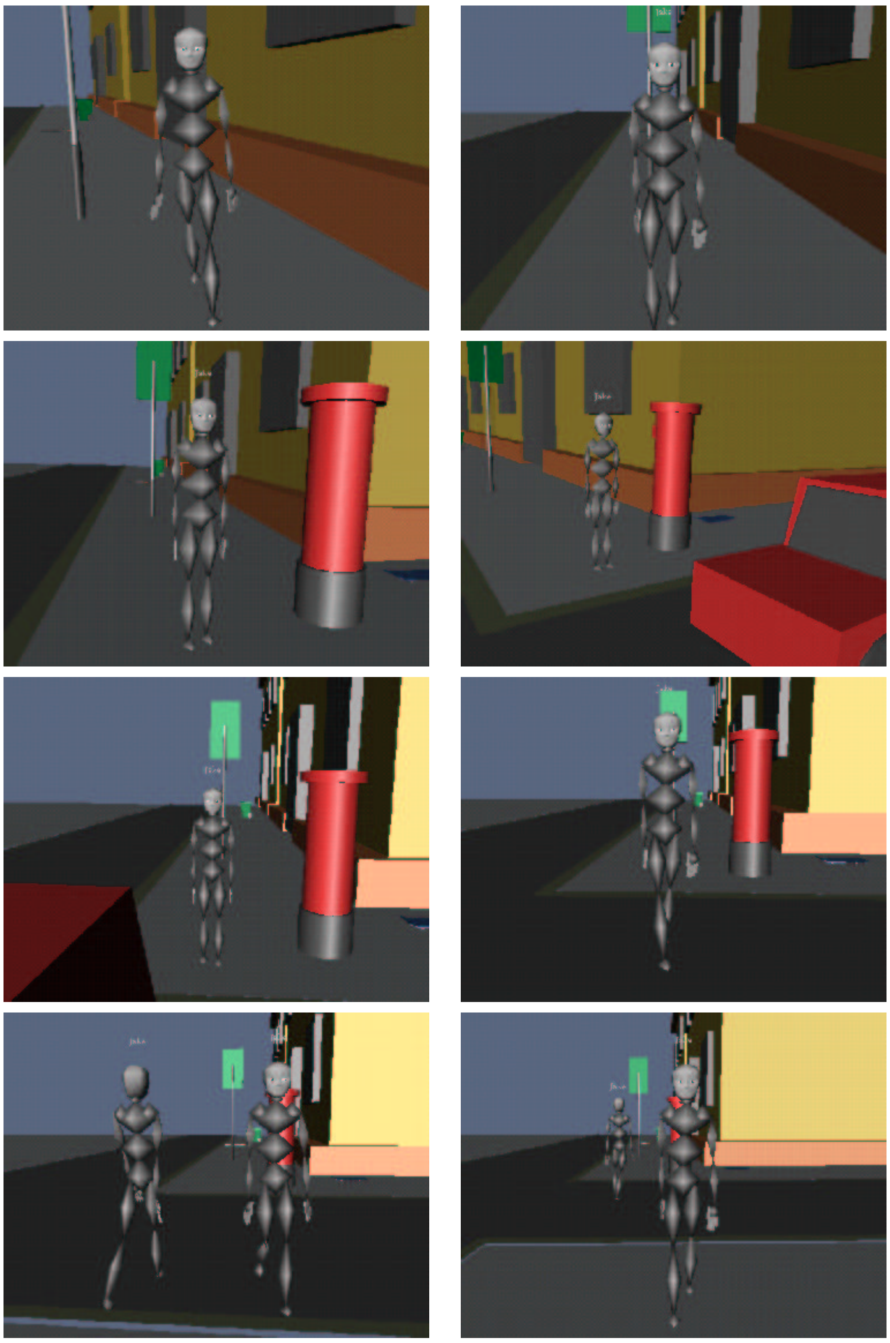


Figure 6.22: The continuation of figure 6.21. (left to right, top to bottom)

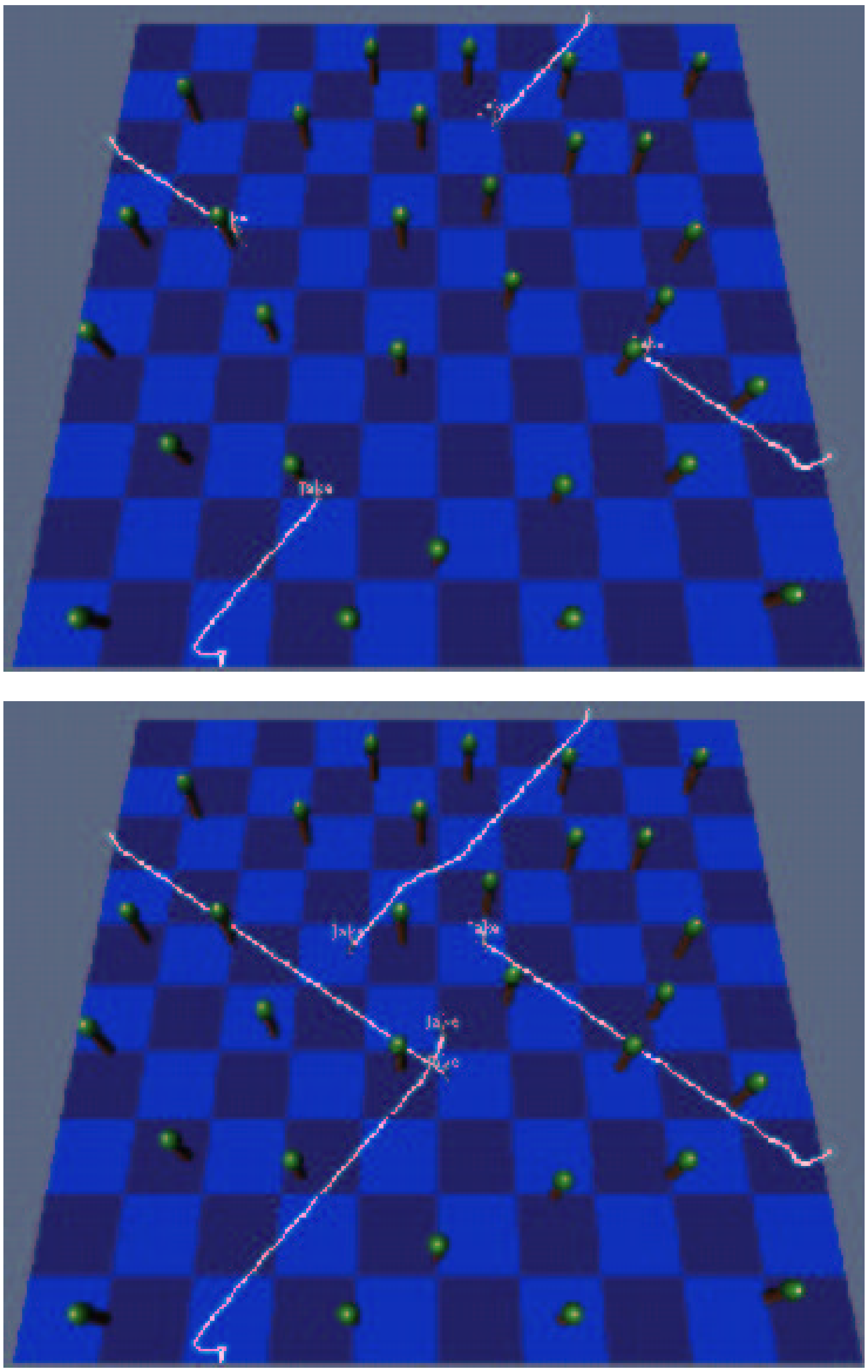


Figure 6.23: A longer example of four actors walking in a ‘forest’.

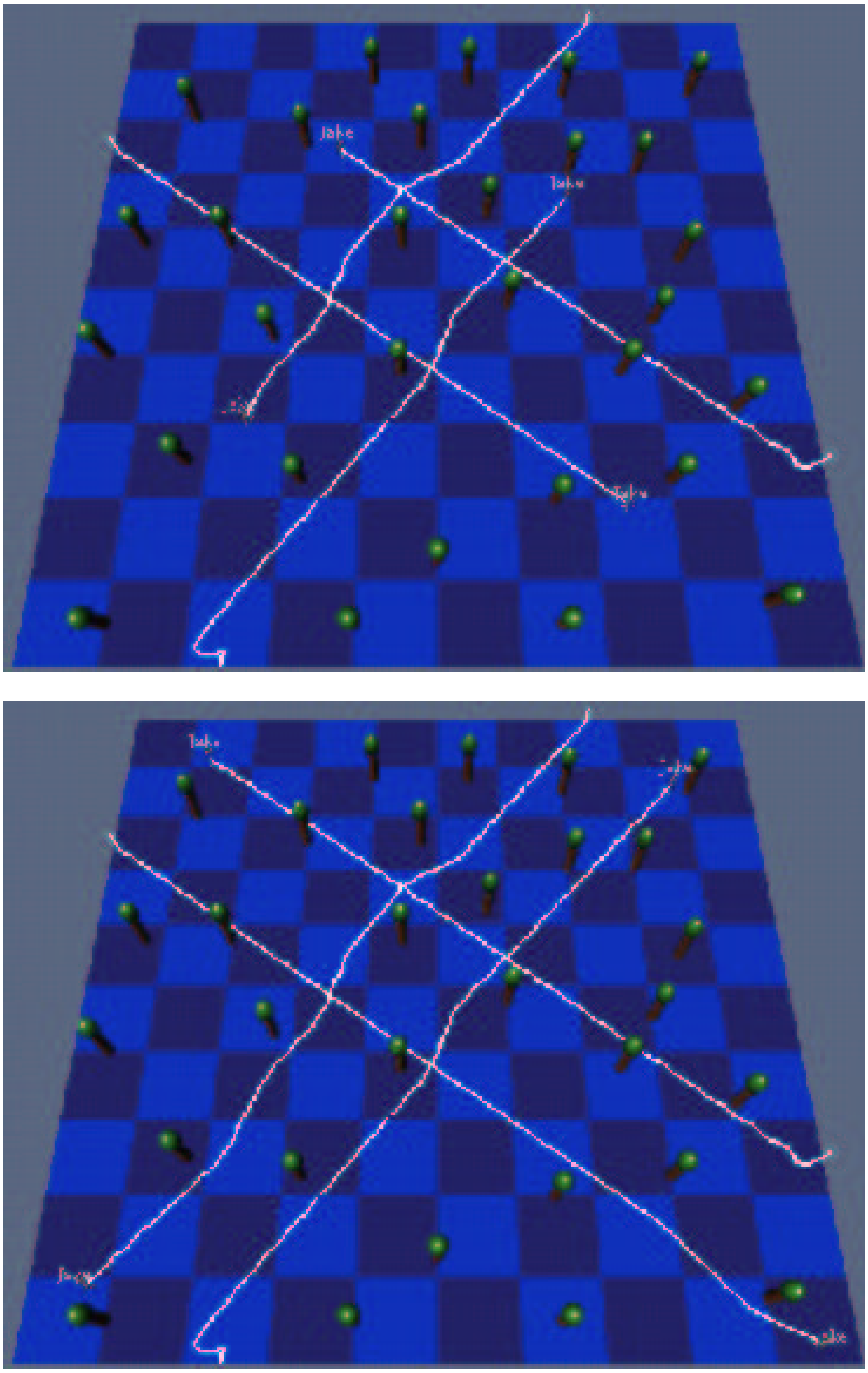


Figure 6.24: A continuation of figure 6.23.

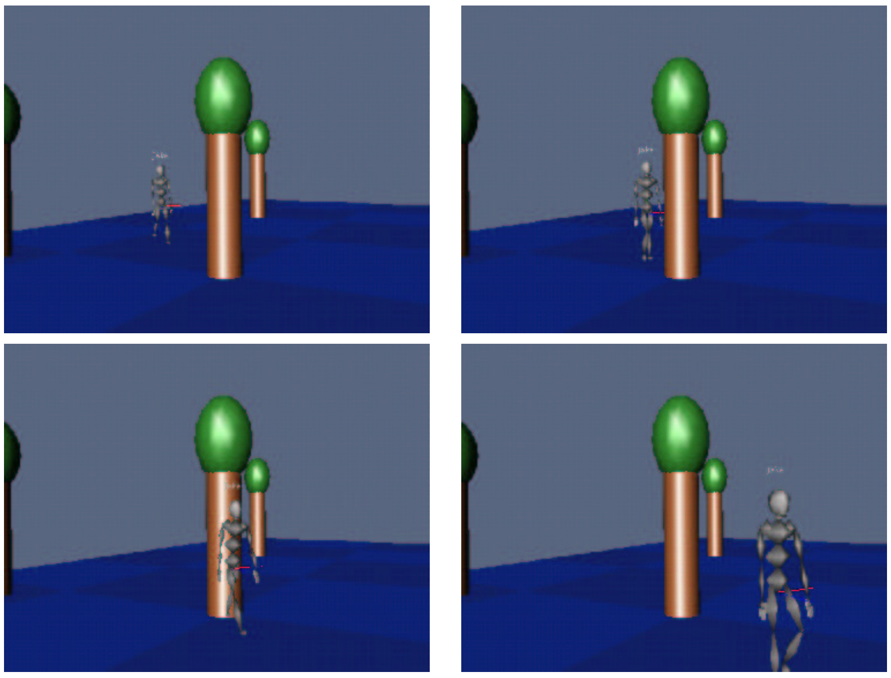


Figure 6.25: The actor is not looking where it is going and it walks into a tree. In the current implementation if an actor walks into an obstacle it will pass through it rather than stopping as there is no 'collision detection' in the sense of detecting when a collision occurs. (left to right, top to bottom)

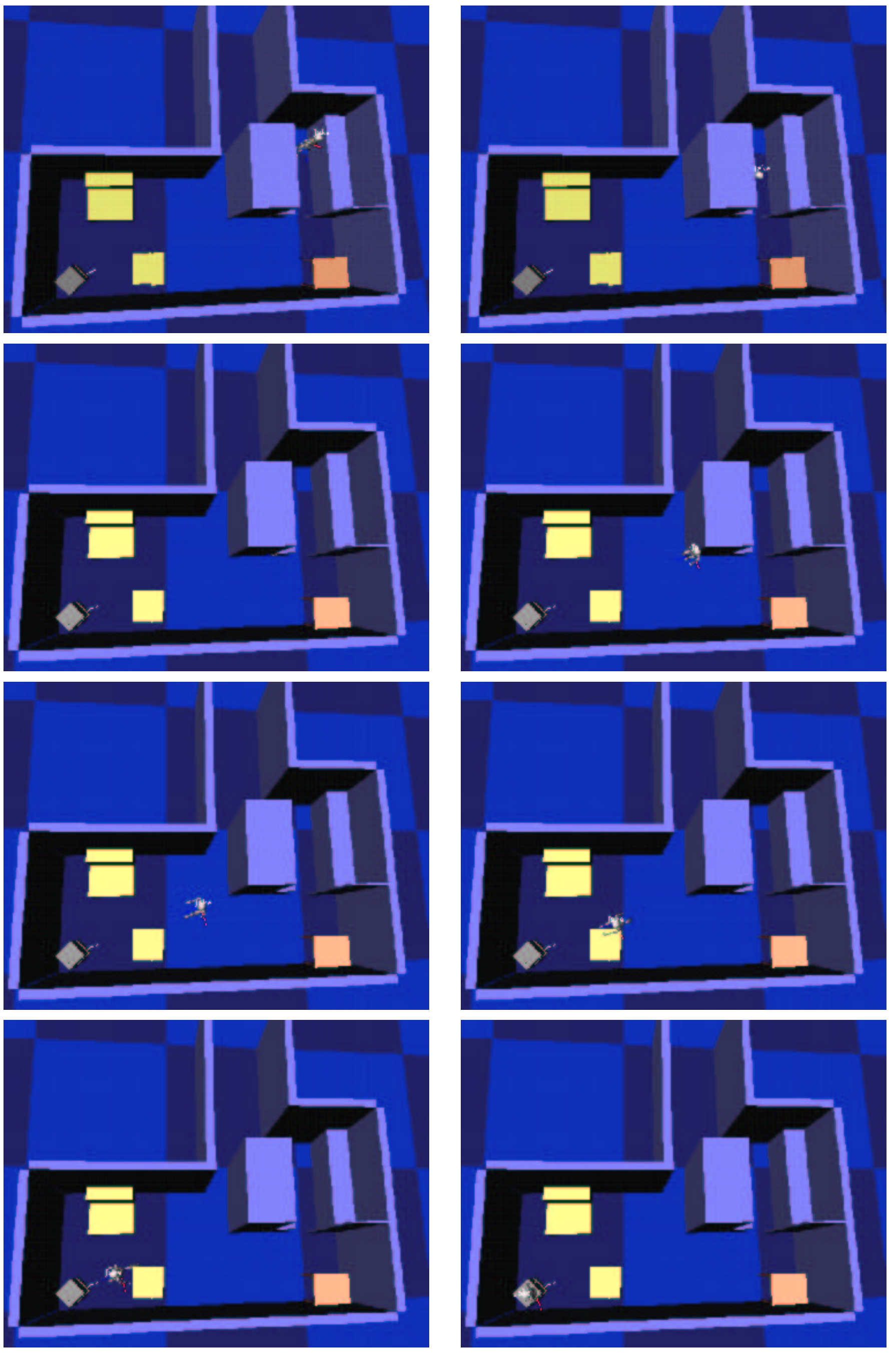


Figure 6.26: An environment modelled on my flat. The actor attempts to walk to the black television in the corner. This shows some difficulties with the algorithms described. They are unable to find a path through this very dense environment and so the actor ends up walking through objects. (left to right, top to bottom)

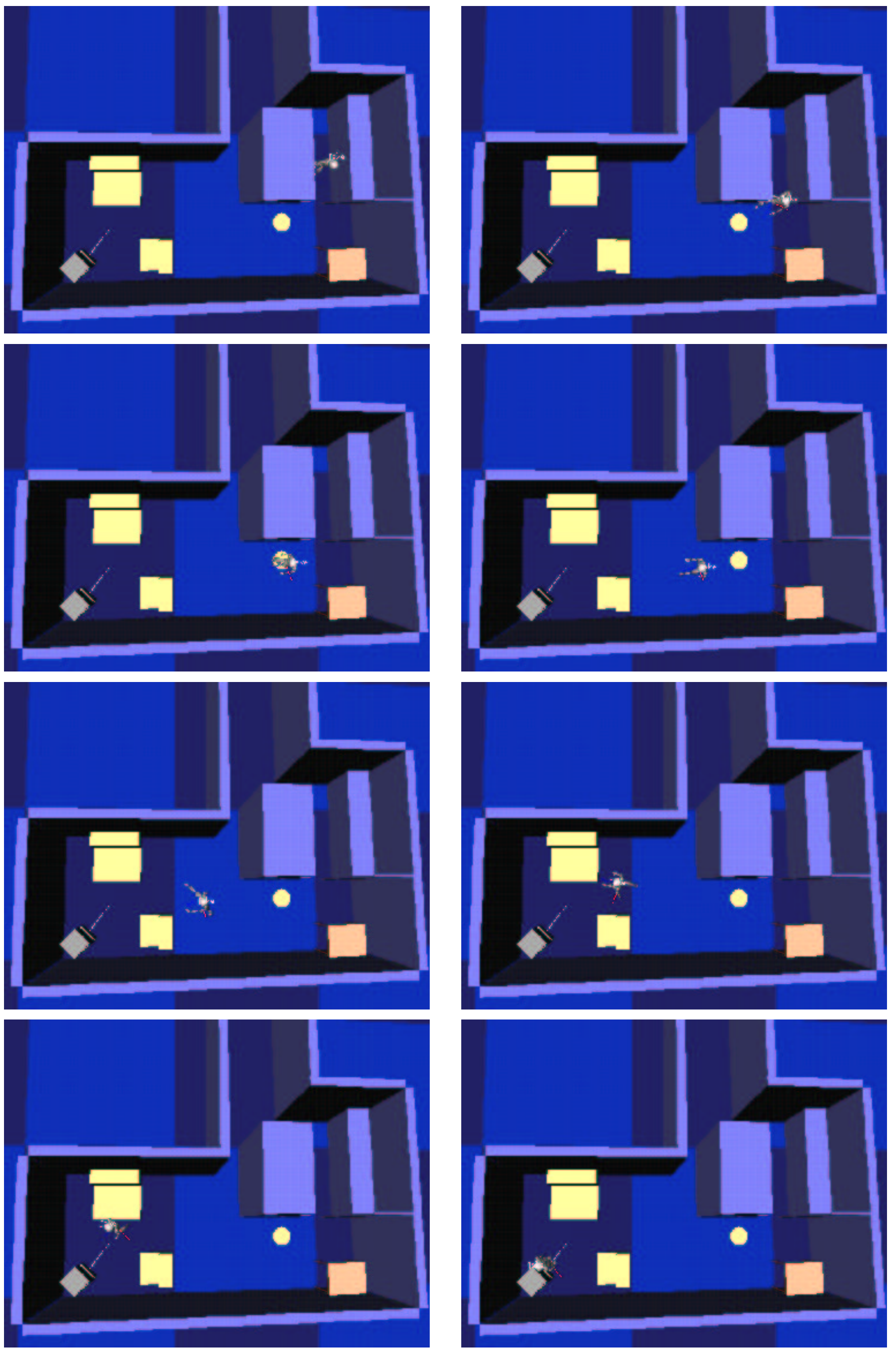


Figure 6.27: The previous example with an intermediary destination (the gold disc on the floor) representing a planned path. (left to right, top to bottom)

Chapter 7

Conclusion and further work

This chapter summarises and evaluates the work described in this disseration. It assesses how the simulation of vision and attention helps the simulation of behaviour for animation. It evaluates the quality of the animation produced by the algorithms described and the efficiency of those algorithms. The second half of the chapter describes possible extensions that could be made to the work.

The aim of this work is to investigate the simulation of vision and attention for behavioural animation. Chapters 3 and 4 describe an approach to the simulation of vision and attention and the implementation of a general simulation of attention. Three main conclusions arise from these chapters:

- The use of knowledge from psychology can produce realistic vision algorithms that are simple and efficient. They are a good starting point for algorithms dealing areas where human visual competences are well understood. However, where they are less well understood this method cannot be applied.
- The simulation of attention can add realism to behavioural simulations by taking account of the objects that the actor is aware of and attending to. It can also increase efficiency by limiting the objects that need to be taken into account, either to those that the actor is aware of or just to the one that the actor is currently attending to.
- The simulation of attention can make behaviour more lifelike by adding eye and head movements associated with attention shifts.

7.1 The examples

The simulation of vision and attention was further investigated through two examples, generic behavioural competences and navigation of an environment.

7.1.1 Behavioural competences

Chapter 5 describes a method by which a single piece of motion can be made into a generic action that can be applied to a number of situations. This is partially achieved through motion warps but the simulation of attention added two main features:

- The attention simulation can add appropriate gaze patterns which increase the illusion that the actor is truly involved with the task.
- Using vision and attention it is possible to design general ways of reacting to the environment that can be used to invoke competences, in this case searching and collision detection.

These are both beneficial to a method for designing competences and improve the realism and usefulness of the competences. However, the competence designer is far from being a complete tool. The methods of invoking competences need to be expanded and the motion produced needs to be improved (see section 7.4.3 for a discussion of improving the motion). However, the methods presented could serve as a starting point for a commercial tool.

7.1.2 Navigation

Chapter 6 applied vision and attention to a commonly studied problem in behavioural animation, navigating a cluttered environment. The result is an algorithm for avoiding collision which is both efficient and reasonably realistic.

It is efficient as it only involves testing one angle every few frames. The algorithm itself was inspired by results in visual psychology but its efficiency comes from the attention simulation. If the test had to be applied to every object in the scene it would be expensive in time and also space as the angles of all the objects would have to be stored for future reference. However, the attention simulation makes it possible to apply the test to only one object at a time. This is supported by the original psychological result as in that case the person had to be looking directly at the object to detect the collision. The test also works equally well for moving and static objects. This gives it a great advantage over planning algorithms such as A*. While A* can reduce the cost by only performing the planning once (an expensive operation) it can only do so if the environment is static. Moving obstacles would force the actor to revise the plan every few frames. Even then moving obstacles are not well handled as A* does not take their motion into account.

More *ad hoc* methods such as those described by Reynolds[Rey88] can deal better with moving obstacles than A* but the motion produced is often of dubious realism. Also they rely on testing all objects in the scene (though hierarchical space subdivision methods help), and so are still less efficient. The paths produced by the actor in this work are reasonably realistic, normally taking the shortest path to safely avoid an obstacle. The realism is greatly enhanced by adding eye movements that are inherently connected to the algorithms producing behaviour. This gives a sense of intentionality to the character as knowing where some one is looking gives a indication of what they are thinking about. The fact that eye movements and behaviour are coordinated enhances the perceived realism of both the eye movements and behaviour as they seem appropriate to each other.

7.2 Quality of motion produced

The motion of the actor produced by the algorithms described here is shown in the figures at the end of chapters 3 to 6. Animated versions are available on the accompanying CD-ROM. The reader can judge for his or herself the quality of the motion produced.

The quality of the motion is reasonably high by the standards of automatically generated motion. The motion is reasonably fluid and the head and eye movements make the behaviour seem more lifelike and convincing. However, the motion is still well below the quality of hand animated or motion captured motion. Also the warped motion it is only of good quality if the new action or footstep is not too different from the original motion as discussed in section 7.4.3. It is however, quicker and cheaper to produce than hand animated or captured motion. It can also be generated on the fly by the computer in real time applications. This makes it suitable for low budget animation, e.g. for television, and for interactive virtual environments where high quality hand produced motion is not feasible. It still cannot compete with the high quality motion in high budget production films.

Having said this the quality of the motion could be enhanced with further work, in particular that described in section 7.4.3.

7.3 Timings

The system described runs in real time on a SGI Octane 195MHz MIPS R12000 computer (the computer used is dual processor but the software is not multi-threaded).

Most of the algorithms involve testing each of the objects in the scene once and so run in roughly linear time. However, the attention simulation normally cuts down greatly the number of objects that need to be tested and so the result is often closer to constant time. The larger the scene the more objects would normally be rejected as being out of the focus of attention of the character. Using a hierarchical space subdivision method would also cut down the number of objects being tested.

The motion warping algorithms generally warp the motion key-frame by key-frame and so are linear in the number of keyframes.

Table 7.1 shows total elapsed times taken for runs of the software (not processor time), using full compiler optimisations, averaged over 5 runs. The timings were taken for two actions, the drinking competence shown in figure 5.15 and navigating the environment of figures 6.15 and 6.16 (but with only one actor). The timings were also taken for just playing the motion without any motion warps, attention behaviour or any enhancements described in this dissertation. There is also a timing for the scene to be navigated without the actor moving. The timings were taken by advancing through the stated number of frames and only rendering the final frame so rendering times are only a small factor. Since

action	number of frames	mean time (ms)	standard deviation	time per frame (ms)	% increase
catch motion	160	169.4	3.3	1.05	N/A
catch competence	160	553.2	11.2	3.45	226.8%
catch preprocessing	N/A	15.6	0.48	N/A	N/A
walk scene	500	868.6	8.88	1.74	N/A
walk motion	500	1160.2	27.36	2.32	33.5%
navigation	500	1457.4	27.53	2.91	25.6%

Table 7.1: Timings for a competence and for navigation

the motion for competences is warped as a preprocessing step before playing the motion it does not show up in the playback timings so a separate timing was taken. (The warping of the walking motion happens at each footstep and does show up).

The obvious question to ask is whether the algorithms are feasible for real time animation on standard PC hardware. The computer used to do the timings is a high end workstation but it is also about three years old and PC hardware is catching up, if it hasn't surpassed it already. A high end PC would be close to powerful enough to give similar timings. So we can assume that the timings are realistic for real-time animation. For high quality graphics a frame rate of 50fps is considered reasonable, this gives 20ms of processing time per frame. A large proportion of this would be taken up with rendering though hardware rendering is now becoming standard and so reduces this. It seems that the timings are good enough. A competence takes up 3.45ms total or 2.4ms overhead over the existing system and navigation takes 2.91ms, 1.17ms overhead including the motion of the character or only 0.59ms overhead over the walking motion. However, there are problems. The competence requires 15.6ms preprocessing when it starts. This is close to the maximum frame time and so would cause the animation to skip. The navigation is also somewhat bursty as the motion warp for the walk is calculated every footstep (about every 16 frames). It is less noticeable, however, as the piece of motion to be warped is shorter. The reason for this spike is that all of the motion is warped at the beginning. There is no reason to do this, each keyframe could be warped as it is needed. This would spread the computation out. Assuming there is a keyframe every 5 frames the spike would be of about 0.5ms every 5 frames which is acceptable (assuming the computation could be distributed perfectly between frames which is admittedly unlikely). This gives us an acceptable frame rate. However, this is only for one character, multiple characters would place a heavy burden on the computer. Three characters performing competences or four walking would use up half of the available time per frame which is a lot considering that a lot of time is required for rendering. However, the present software was written without particular attention to speed so it is possible that the present techniques could be made faster. Thus the timings indicate that the techniques are feasible for real time animation if the number of characters is small.

7.4 Further Work

This dissertation has described some work in what is a large field. There is much work that could be done to extend the work described. This section will briefly discuss five possible extensions. The first would provide a stronger basis for the algorithms used. The next two are simply improvements of the current work, a better user interface for altering the parameters of motion and improvements to how motion is produced for behavioural competences. The last two extend the scope and applications of the work. Firstly how to handle interaction between different characters, secondly how the internal state of the character could be used to control a camera.

7.4.1 Evaluation of attention heuristics

Chapter 4 describes a simulation of attention based on various informal observations. It would be desirable to place this system on stronger foundations by evaluating the observations more formally. This however, is problematic as it requires capturing the details of naturalistic behaviour that might not come out in experimental conditions. It would also require data about a wide variety of people performing a wide variety of actions.

One method would be to observe people's attention behaviour using eye tracking apparatus. Unfortunately this apparatus can be bulky and might restrict the range of possible actions it is possible to perform. Also it would require observing a large number of people as an important aspect of the heuristics is the variation between people.

It would also be possible to repeat my method of observation more formally. While I observed people in a natural setting completely informally it might be possible to attempt a more exact observation. Various classes of behaviour could be identified, for example glancing or looking forward. The number of time people performed each action could then be counted. Collett and Marsh use a similar method [CM81].

It would also be possible to evaluate the heuristics from the point of view of whether they produce convincing behaviour, rather than whether they are completely accurate simulations. This is a perfectly valid basis for evaluation, as the purpose is to animate outwardly convincing behaviour. In fact it could be said to be a more important evaluation than accuracy. This could be done by having subjects view and evaluate in some way the behaviour produced. As the algorithms are designed to be the basis of an animation tool it might be sufficient to do user tests on animators and if they are happy with the range of behaviour it is possible to produce the software can be deemed a success.

7.4.2 Using parameters

Currently the behaviour of the actor is controlled by algorithms that contain parameters. These parameters affect the behaviour produced with the aim of being able to produce different types of behaviour for different characters. The eventual aim of this would be able to endow characters with a "personality" that would influence their behaviour and distinguish them from other characters.

However, the problem with the parameters as they stand is that they are not easy for a user to interpret and change in such a way as to achieve a desired effect. Firstly there are a very large number of parameters. Figure 5.14 shows the sliders for altering the parameters of just the behavioural competenceseye movements, on top of that there are parameters for the eye movements and the walking behaviour. It can be daunting to have to edit all these sliders for each character. What is more it is often not intuitively clear what effect altering a parameter will have on the personality of the character. For example, what personality trait does increasing the gaze length produce, it certainly has an effect but does staring for a long time indicate an artistic interest in the world or mental instability? This is compounded by the fact that parameters interact with each other to produce different effects. For example, a short glance length might indicate nervousness or distractability if there is a high probability of glancing. If, however, glances are infrequent and are interspersed with long gazes at a central task, such as typing, the short glances might indicate wanting to "keep an eye" on the surroundings while concentrating on the task.

The difficulty of determining the effect of changing a parameters means that the user would have to try out the effect of each parameter. As there are a large number and as parameters have to be checked in different combinations to account for interactions between parameters this can become a daunting task. It would be better to have an interaction method which dealt directly in terms of personality traits rather than with the parameters themselves. This is rather difficult, however, as the mappings between parameters and personality are complex and open to interpretation. It would be insufficient to code in a few traits into the system, for example having a "happy" parameter setting as this would not give the user enough scope to achieve the desired effect and would not capture the range of nuances that could be contained in the word "happy".

Polichroniadis[Pol00] has suggested a method for mapping emotions and personality traits onto numerical parameters of a piece of motion. The traits are represented by *adjectives*, simply words that describe the trait. Adjectives can be any word that can be applied to a piece of motion and tend to be of many different types, examples include "happy", "old", "bow-legged" and "cowboy". The system initially maps adjectives to the parameters of the system based on user input. Once this has been done, and it only needs to be done once, the user can simply describe the character in terms of adjectives in order to get the desired effect.

This method appears suitable for the behavioural parameters. However, there are some problems with directly applying it. Polichroniadis applied the method to transformations of pieces of motion. These trasformations generally become apparent within a few seconds of watching the motion. However, many behavioural parameters consist of frequencies of certain events occurring, for example the frequency of glances. This means that long sections of animation must be watched before assessing a transform making the process much longer and making it harder to classify a large number of traits. However, it seems likely that some adaption of Polichroniadis' method could be used for behavioural parameters.

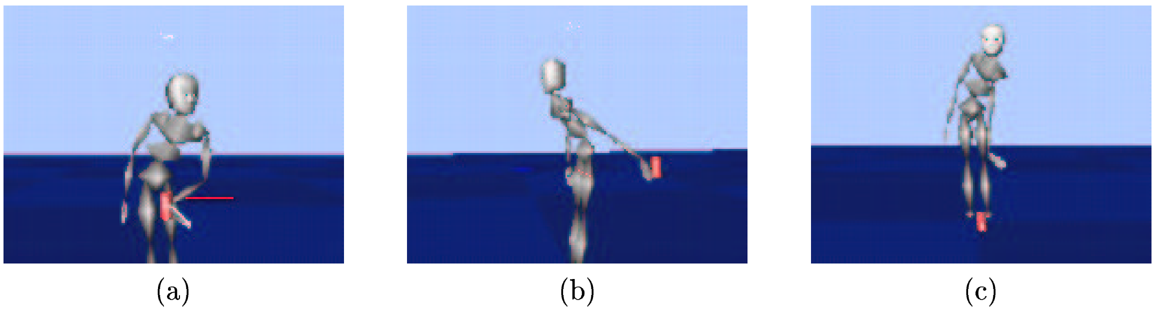


Figure 7.1: A competence designed with a drinking motion with a cup to the left and in front of the character works fine when the can is moved to the front (a). However, with the can behind the character (b) the lean forward of the shoulders looks strange, the character should really twist around to face the can. In (c) the can is at the feet of the character but the original motion contains no knee bend so the actor attempts attempts to pick it up from standing which leads it to attempt an end effector position which is too far to reach.

7.4.3 Extending competences

Chapter 5 describes a method for creating generic behavioural competences which include attention behaviour. This is an initial sketch of a possible tool for animators but there is much more work that could be carried out in this area. In particular there is one defect of the method that could be addressed. Currently the competence works by taking a piece of motion and adapting it to new instances, for example, changing the position of a cup in a drinking motion. This works well if the new position is near the original position in the piece of motion (figure 7.1(a)). However, as shown in figure 7.1(b) and (c) if the position is very different the effect stops being realistic.

A solution to this problem is to base the competence not on one but on several pieces of motion. An appropriate one could be chosen for each position. It would also be possible to interpolate motions in intermediate positions. The problem with interpolation is establishing correspondences between points in the motion. However, if the different motions were assigned the same periods, as they would have to be if the other aspects of competence are to work, they could easily be used to establish correspondences. The start of periods could be assumed to correspond. Then times within periods could correspond if they are the same proportion of the way along the length of the period. As periods could have different lengths in different motion the length would have to be adjusted, probably just by interpolating the lengths of the two motions. So if the motions to be interpolated were p_1 and p_2 on a period x with start times t_{1x} and t_{2x} and period lengths t_1 and t_2 and the interpolation variable is α, the interpolation could look something like:

$$p(t) = \alpha p_1 \left(\frac{(t - t_{1x})\tau}{t_1} + t_{1x} \right) + (1 - \alpha) p_2 \left(\frac{(t - t_{2x})\tau}{t_2} + t_{2x} \right)$$

where τ is the new length of the period

$$\tau = \alpha t_1 + (1 - \alpha)t_2$$

Sometimes the actor actually has to orient itself correctly before starting a motion. This could also be done with extra motions that start by turning the character around. These would have to work by adding extra periods that do not exist in other motions corresponding to the orientation time.

A similar method could also be used for improving the walk generator. Different motions could be used based on the direction of the path and the velocity. This would maybe produce better motion along very curved paths by basing the transform on a piece of motion where the actor is already turning rather than one in which it is walking straight. Also this walk generator could handle walking backwards by having a special piece of motion, and higher speeds by having running motions. It might even be possible to improve the head turning algorithms in the attention manager by using pieces of recorded motion to move from one position to another, rather than just interpolating angles. It is not clear whether this method would provide any real improvement, however.

Finally, there is another advantage to using more than one piece of motion. If the actor is to repeat an action several times it could seem repetitive if the same motion is used. This could clearly be solved by techniques such as Perlin noise [Per95]. However, if multiple motions are used variety could also be added by varying the proportion in which two motions are interpolated. This might be more realistic than using noise as all the variation is based on actual motion rather than being random.

The disadvantage of using multiple pieces of motion is that it puts greater demands on the motion used. If a single piece of motion is used it can be any sort of motion, it could be taken from a generic motion capture library. This would mean that the designer of the competences would not need to have access to expensive motion capture equipment and so the tool would be available to a wide range of users. However, motion for a competence based on multiple motions would have to be more specific. The motions would have to cover a reasonable range of relative positions of the actor and target. They would also have to be fairly similar to avoid jarring changes in the way the character moves. This would mean that it is unlikely that the designer would find generic motion of the correct sort and so would have to specifically capture the motion, increasing the cost. Of course, competences created by a designer with access to motion capture could still be used by lower budget users without motion capture facilities.

7.4.4 Social interaction

One of the largest and most important extensions to the current work is to include interactions between people, social interaction of one sort or other. Currently all the work has involved interaction with inanimate objects. However, the range of interactions with other people is very large. It is also in general more interesting, the great majority of the content of films deals with interaction between people. This makes it vital to include social interaction in any behavioural animation tool, otherwise it would be effectively useless, or at least very limited. Because social interaction is very important to us we are very sensitive to the nuances involved. They are much more subtle and expressive than those involved in non social action. This makes the task of producing behavioural algorithms much more difficult as slight inaccuracies will be less tolerated and it is important to be able to express a large and subtle range of emotions. The next three sections will describe the areas most relevant to social interaction.

Competences

It would, of course, be important to create behavioural competences that involve interaction between more than one actor. A simple example would be a hand-shake. The problems involved here centre on the fact that there is no longer only one actor involved. Both actors will be performing an action and they will have to coordinate between them. Each actor will have to react to what the other actor is doing. For a hand-shake this is fairly simple as it only involves finding a mutual position for the hands to meet. Once this is done the motion is constrained to fit together. More complex interactions demand more complex methods of resolving them. In particular, adverserial actions involve one actor trying to prevent the other performing an action. These sort of behaviours are common in sport, for example a football tackle. How is the competence to ensure that the actor reacts in an appropriate way to the other's action and how is it to decide which will succeed.

There are two possible ways of organising social competences. Each actor could independently perform the action, coordinating it by adapting the action based on communication with the other actor via *Comms*. This is close to how such actions are performed by real people. However, it might be better for the actors to spawn a shared agent which controls both of their actions in order to control the action as a whole. Having a single agent deciding the details of the action and arbitrating between the preferences of the two actors might make for a better looking interaction as it could take into account the action as a whole and would not have problems with the two actors failing to interact properly. The added complexity of social actions would, of course, raise new user interface problems for creating a suitable tool for creating such actions, how can a designer specify the interaction between two characters?

Eye movements

As noted in chapter 4 the eye movements of a character are expressive of the character's emotions. This aspect becomes far more important in social interactions. New aspects of the looking behaviour would have to be added. Firstly, deciding where to look when looking at another person is much more important than when looking at an inanimate object. Most of the work described here has just dealt with looking at an object in general rather than at particular parts. However, when looking at people this is no longer sufficient, looking at a person's eyes, chest or feet all mean something very different and are normally easily distinguished.

More importantly, looking at a person now includes the possibility that the person looks back, mutual gaze. This clearly involves some sort of coordination of the sort discussed above when dealing with competences. This interaction is, however, very complex and expressive. Argyle and Cook[AC76] describe the many ways in which people's gaze interacts. Much is involved with the conflict between meeting and avoiding gaze, what Argyle and Cook call the intimacy equilibrium model. There are many social rules on when it is necessary to meet gaze and when it is necessary to avoid gaze. Also many social situations involve complex interactions between meeting gaze and avoiding. During conversations, for example, the talker will avoid looking at the listener while the listener will tend to look at the talker. The listener will look away when about to start talking. This is thought to be due to the fact that looking at a person's face involves a high cognitive load and so will distract when talking. However, looking at some one while listening to them makes listening easier, by correctly orienting the ears and by partial lip reading, and also helps the listener pick up non-verbal cues to the talker's meaning. Another interesting interaction of gaze is brought on by a certain sort of ambivalent attitude to the person being looked at, involving a combination of interest and nervousness, found in situations such as sexual attraction or fear. This sort of situation results in very characteristic behaviour of a desire to look at the other person while also wanting to avoid gaze. This results in a pattern of avoiding gaze interspersed with brief glances.

Eye movements involve complex nuances. For example, a popular book on body language[Wai99] lists avoiding gaze as possibly meaning the person is uninterested in the other person's reactions; they do not like the other person; they are introverted; the other person is of higher status; they are sexually attracted to the other person, or they are suffering from certain forms of mental illness. Meeting the other person's gaze can mean, they are interested in the other person's reactions; they like or love the other person; they are being aggressive; they are extrovert or they are sexually attracted to the other person. These cases can often be distinguished but how can such nuances be captured in a behavioural algorithm?

The Attention manager could be extended to include some sort of coordination mechanism for organising mutual gaze patterns. These could be extended to include behaviour patterns such as those described above. It would be important to capture the range of eye movements that are considered expressive and to understand which nuances are important so that the system is able to distinguish them. This is a complex and very interesting area and would require a lot of further work. There has been some work in this area, focusing on eye movements and other non-verbal communication during conversations[Thó98, VC98, CCD00].

Navigation

Navigation as described in chapter 6 changes greatly when a person is navigating an environment populated by other people. First of all inanimate objects do not react to the presence of the actor so any avoidance behaviour will solely come from the actor. Other actors will, however, react and attempt to avoid collisions just as the actor is attempting to avoid collisions with them. This involves some form of coordination between the two actors. Generally, this is successful with people moving in opposite directions to avoid a collision but the behaviour is probably most noticeable when it fails and the two people perform an avoidance “dance”, each moving in the same direction and then attempting to counter by moving back in the opposite direction, with neither being able to pass. This sort of failure is rare, however, so how do people negotiate to decide in which direction to move in? Certainly there is no explicit communication, it is very rare that people talk to each other as they pass on the street. However, there is plenty of non-verbal communication and standard tactics for navigating among people.

Goffman[Gof72] notes that on crowded streets the most common tactic is *streaming* or *lane formation*. This is where groups of pedestrians moving in the same direction will form lanes, walking in lines, one behind the other, at roughly the same speed. People walking in the opposite direction will form a lane beside the first lane. This could form the basis of a navigation algorithm similar to Reynolds’ flocking algorithm[Rey87]. When the streets are less crowded non-verbal communication becomes more important. People have to make clear what they intend to do by their behaviour as they walk down the street, what Goffman calls “body-gloss”. He describes this as:

> ...the process whereby an individual pointedly uses over-all body gesture to make otherwise unavailable facts about his situation gleanable. Thus, in ... walking the individual conducts himself ... so that the direction, rate and resoluteness of his proposed course will be readable.[Gof72, pages 31–32]

Modelling this sort of gestural action would be an important part of simulating navigation amongst other people. In order to pick up this information people have to observe other people in their vicinity. This is in many ways similar to how the actors in this work observe inanimate objects in order to avoid collisions with them. However, the situation is more complex as there are many social rules which govern how we can and cannot look at other people, particularly strangers, as most people on the street would be. It is important to look at another person without seeming to be too interested or implying a desire to enter into an interaction with that person. This is achieved by looking at the person but not at their face; only glancing briefly; turning away immediately if eye contact is made, and other similar behaviour. These behaviour patterns would form the basis of simulating eye-movements and attention behaviour for characters navigating a populated environment.

7.4.5 Camera work

There has been an interest recently in algorithms for automating camera work within a scene in order to ensure it is always viewed from a suitable angle[LCD96, FTT99]. The general idea is to use the knowledge of cinematography built up over years of cinema history as a starting point for designing algorithms for generating camera positions and movements. If the behaviour of characters in the scene was being simulated and so internal aspects of the characters actions were known this knowledge could be used to direct the camera at appropriate places in the scene. For example, if the character were walking to a destination various interesting pieces of camera work could be performed involving the character and the destination. This sort of action is common in films and various types of shot are common. The camera could alternate between frontal shots of the character and destination as shown in figure 7.2. Another option is that the camera could follow the character as it walks and then perform some sort of pan or cut as it arrives at its destination. For example, in figure 7.3 the camera pans around to an frontal shot of the character and the destination, ready to capture any interaction between them.

Behavioural competences could have camera viewing angles built into them using secondary actions in a similar way to eye movements(see section 5.4.1). Various positions could be viewed by the camera, for example, the character's face; the end effector performing the competence or one of the contact positions. These could be viewed from different angles, the of view of the character; a frontal view of the character, or from a contact position (bringing the possibility of defining a contact position that acts as a camera). These various options could be presented as parameters to the secondary action in a similar way to the options for eye movements being presented in and *EyeAction*. The designer or user could then define probabilities that the camera would take various positions and angles. Clearly if more than one character is present in a scene there must be some mechanism for deciding which controls the camera. This could be done by competences sending requests to a central camera manager that arbitrates between them, possibly on the basis of priorities of requests (based on the type of action and which character is performing it). This camera manager might work in a similar way to the attention manager (see section 4.4.1), possibly including types of request similar to immediate and monitor request.

The character's attention would also be an interesting way of driving the camera. The most obvious way is having a camera that follows the character's attention, looks at what the character is looking at. This would be an extension of the sort of "first-person" views used in computer games. Though this would be a simple way of having the camera take into account the characters attention it is probably not that effective. It would tend to have a disorienting effect on the audience, people have a much lower tolerance of a camera's motion on-screen than for the motion of their eyes. On option is having a camera that is behind the characters head and following the character's attention from and "over-the-shoulder" shot. However, film makers have more sophisticated techniques. One example gets around the problem that characters' eyes are generally too small to see on anything but a close up, but that this does not permit the viewer to see what they are looking at. A character is shown in close-up moving their eyes or head to look at something off screen. The actual movement is shown finishing with the character's final gaze position, this provides an action for the shot and establishes that they are looking at something important. Then the camera cuts to the object being looked at, after the establishing shot it is clear that this is what the character is looking at. Figure 7.4 gives an example of this sort of shot. Alternatively, the camera could pan around from the establishing shot to an over the shoulder shot of the character looking at the object. This is one of many techniques that could be automated by using a virtual actor's attention mechanisms to produce camera angles.

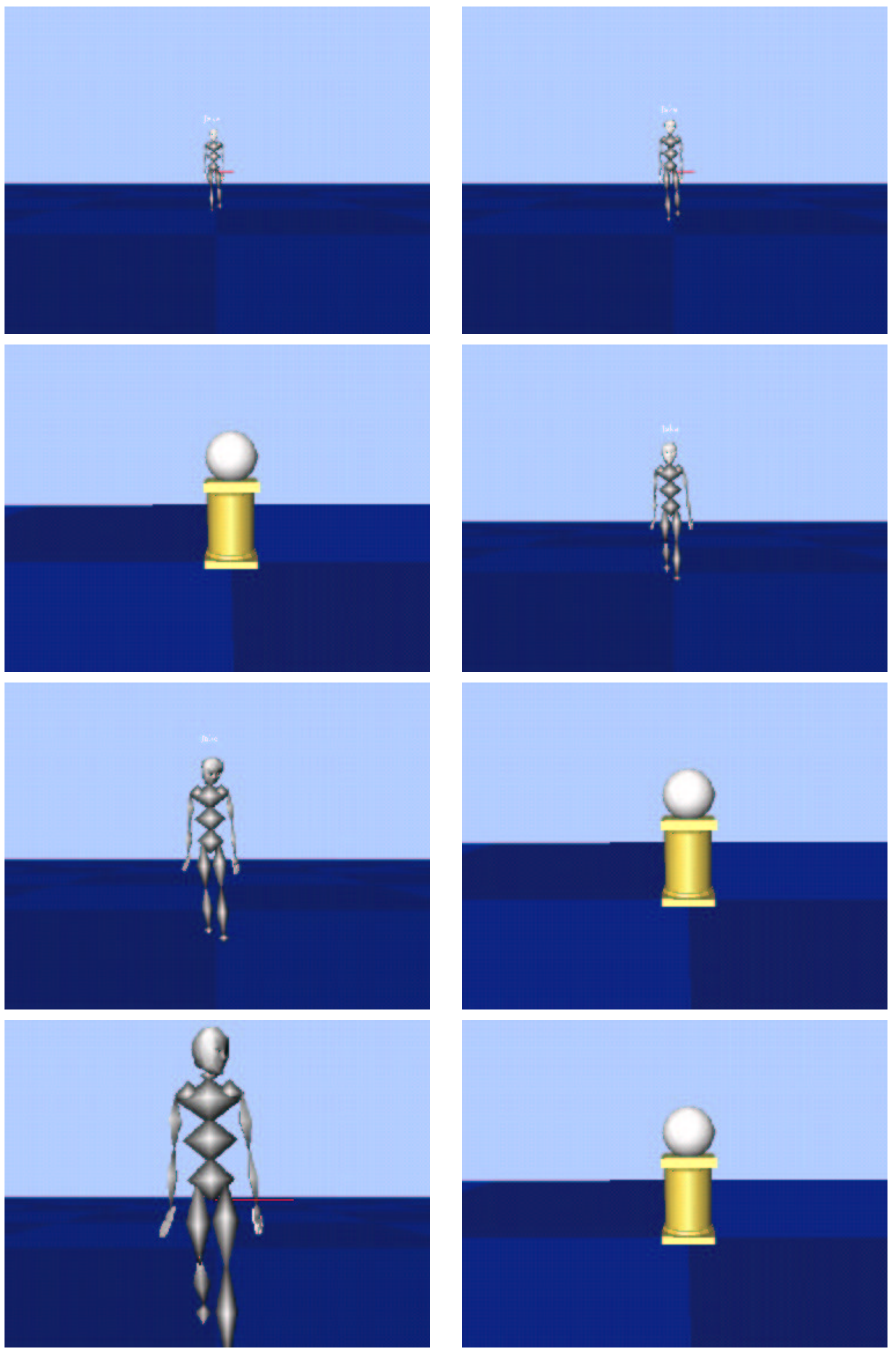


Figure 7.2: One possible way of filming a character walking to a destination (shown as the yellow column topped with a ball). In this case the camera alternates between the characters and his/her destination. (left to right, top to bottom)

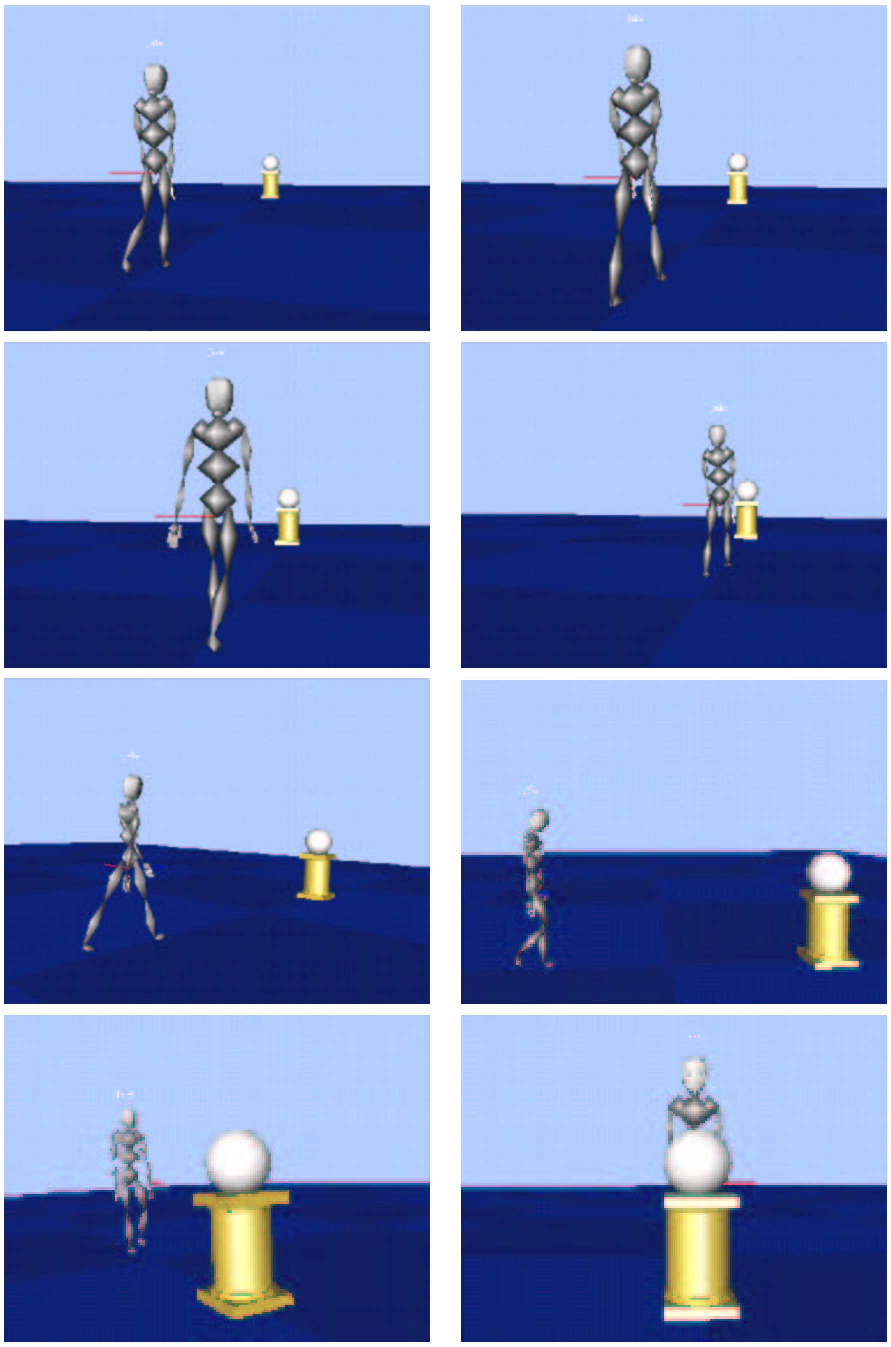


Figure 7.3: Another possible way of filming a character walking to a destination (shown as the yellow column topped with a ball). The camera follows the character with the destination in shot. At the end the camera pans around to a frontal shot of the character and destination.

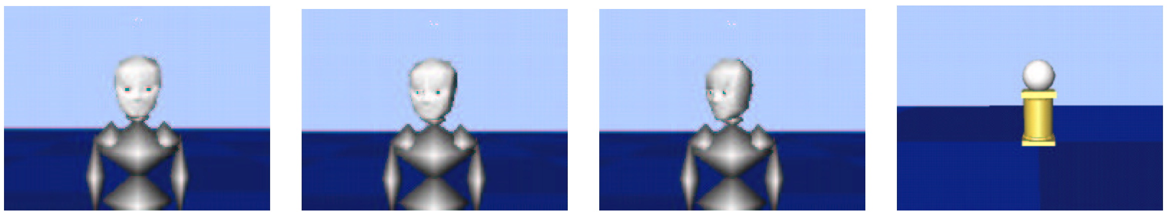


Figure 7.4: A typical way of filming a character looking at something. An establishing shot (frames 1-3) show the character turn his/her head to look at something. The camera then cuts to what the character is looking at. (left to right, top to bottom)

7.5 Final Word

I hope I have convinced the reader that vision and attention are important for the simulation of behaviour or computer animation. I also hope they are convinced that it is possible to produce a tool for behavioural simulation that gives the user enough control to design the behaviour he or she wants while still being faster than hand animation. The work presented here is far from being this tool. It does not solve all the problems involved and is not without its shortcomings and constraints. Maybe, however, it could serve as a starting point.

Bibliography

[AC76] Micheal Argyle and Mark Cook. *Gaze and Mutual Gaze.* Cambridge University Press, 1976.

[AL95] Micheal A. Arbib and Jim-Shih Liaw. Sensori-motor transformations in the world of frogs and robots. *Artificial Intelligence*, 72:53–79, January 1995.

[BBT99] Christophe Bordeux, Ronan Boulic, and Daniel Thalmann. An efficient and flexible perception pipeline for autonomous agents. In *Eurographics '99*, volume 18 of *Computer Graphics Forum*, 1999.

[BC99] Srikanth Bandi and Marc Cavazza. Integrated world sematics into path planning heuristics for virtual actors. In Daniel Ballin, editor, *Proceedings of the second workshop on intelligent virtual agents*, 1999.

[BG95] B. Blumberg and T. Galyean. Multi-level direction of autonomous creatures for real-time virtual environments. In *ACM SIGGRAPH*, pages 47–54, 1995.

[BGG96] Vicki Bruce, Patrick R. Green, and Mark A. Georgeson. *Visual Perception Physioloy, Psychology and Ecology.* Psychology Press, 1996.

[BH00] Matthew Brand and Aaron Hertzmann. Style machines. *Proceedings of SIGGRAPH 2000*, pages 183–192, July 2000.

[BMW87] Norman I Badler, Kamran H Manoochehri, and Graham Walters. Articulated figure positioning using multiple constraints. *IEEE Computer Graphics and Applications*, 7(6):28–38, June 1987.

[BO93] Reinoud J. Bootsma and R. R. D. Oudejans. Visual information about time to collision between two objects. *Journal of Experimental Psychology: Human Perception and Performance*, 19:1042–1052, 1993.

[BPW93] Norman Badler, C. Philips, and B. Webber, editors. *Simulating Humans: Computer Graphics, Animation and Control.* Oxford University Press, 1993.

[Bro85] Rodney A. Brooks. A robust layered control system for a mobile robot. AI memo 864, MIT Artificial Intelligence Laborotory, September 1985.

[Bro90] Rodney A. Brooks. Elephants don't play chess. In Pattie Maes, editor, *Designing Autonomous Agents*, pages 3–15. MIT Press, 1990.

[Bro91] Rodney A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

[BSZS95] W. Barfield, T. Sheridan, D. Zeltzer, and M. Slater. Presence and performance within virtual environments. In W. Barfield and T. Furness, editors, *Virtual Environments and Advanced Interface Design.* Oxford University Press, 1995.

[BT98] Srikanth Bandi and Daniel Thalmann. Space discretization for efficient human navigation. In *Eurographics '98*, volume 17 of *Computer Graphics Forum*, pages 195–206, 1998.

[BW95] Armin Bruderlin and Lance Williams. Motion signal processing. *Proceedings of SIGGRAPH 95*, pages 97–104, August 1995.

[CCD00] Alex Colburn, Micheal Cohen, and Steven Drucker. The role of eye gaze in avatar mediated conversational interfaces. Technical report, Microsoft Research, 2000.

[Cha68] Seville Chapman. Catching a baseball. *American Journal of Physics*, 36(10):868–870, October 1968.

[CHP89] John E. Chadwick, David R. Haumann, and Richard E. Parent. Layered construction for deformable animated characters. In *ACM SIGGRAPH*, pages 243–251, 1989.

[CKB99] Sonu Chopra-Khullar and Norman Badler. Where to look? automating visual attending behaviors of virtual human characters. In *Autonomous Agents Conference*, 1999.

[CM81] Peter Collett and Peter Marsh. Patterns of public behaviour: Collision avoidance on a pedestrian crossing. In *Non-verbal Communication; interaction and Gesture*, Approaches to Semiotics, pages 199–217. Mouton, 1981.

[CPB$^+$94] Justine Cassell, Catherine Pelachaud, Norman Badler, Mark Steedman, Brett Achorn, Tripp Becket, Brett Douville, Scott Prevost, and Matthew Stone. Animated conversation: Rule-based generation of facial expression, gesture and spoken intonation for multiple character conversational agents. In *ACM SIGGRAPH*, pages 413–420, 1994.

[CVB95] James E. Cutting, Peter M. Vishton, and Paul A. Barren. How do we avoid collision with stationary and moving obstacles. *Psychological Review*, 102:627–651, 1995.

[CVB01] Justine Cassell, Hannes Högni Vilhjálmsson, and Timothy Bickmore. Beat: the behavior expression animation toolkit. In *ACM SIGGRAPH*, pages 477–486, 2001.

[DBB96] James L. Dannemiller, Timothy G. Babler, and Brian L. Babler. On catching a flyball. *Science*, 273:256–257, July 1996.

[DBV95] Alberto Del Bimbo and Enrico Vicario. Specification by-example of virtual agents' behavior. *IEEE transactions on visualtization and Computer Graphics*, 1(4):350–360, December 1995.

[DM93] Zoltan Dienes and Peter Mcleod. How to catch a cricket ball. *Perception*, 22:1427–1439, 1993.

[DS96] Rudolph P. Darken and John L. Sibert. Wayfinding strategies and behaviors in large virtual worlds. In *ACM SIGCHI*, pages 142–129, 1996.

[FTT99] John Funge, Xiaoyuan Tu, and Demetri Terzopoulos. Cognitive modeling: Knowledge, reasoning and planning for intelligent characters. *Proceedings of SIGGRAPH 99*, pages 29–38, August 1999.

[GBL84] Tommy Gärling, Anders Bröök, and Erik Lindberg. Cognitive mapping of large scale environments; the interrelationship of action, plans, acquisition and orientation. *Environment and Behavior*, 16:3–34, 1984.

[Gib79] J. J. Gibson. *The ecological approach to visual perception.* Lawrence Erlbaum Associates, 1979.

[GL96] Michael Gleicher and Peter Litnowicz. Constraint-based motion adaption. Technical Report Tr 96-153, Apple Computers, 1996.

[Gle97] Michael Gleicher. Motion editing with space time constraints. In *symposium on interactive 3D graphics*, pages 139–148, 1997.

[Gle99] Michael Gleicher. Retargetting motion to new characters. In *ACM SIGGRAPH*, pages 33–42, 1999.

[Gle01] Michael Gleicher. Motion path editing. *2001 ACM Symposium on Interactive 3D Graphics*, pages 195–202, March 2001.

[GM85] Michael Girard and A. A. Maciejewski. Computational modelling for the computer animation of legged figures. In *ACM SIGGRAPH*, pages 263–270, 1985.

[Gof72] Erving Goffman. *Relations in Public.* Pelican, 1972.

[Gol95] Reginald G. Golledge. Path selection and route preference in human navigation: A progress report. In *Spatial Information Theory: A theoretical basis for GIS*, volume 988 of *Lecture Notes in Computer Science*, pages 207–223. Springer, 1995.

[Gor89] Ian E. Gordon. *Theories of Visual Perception.* John Wiley & Sons Ltd., 1989.

[GSBS01] Maia Garau, Mel Slater, Simon Bee, and Martina Angela Sasse. The impact of eye gaze on communication using humaniod avatars. In *ACM SIGCHI*, pages 309–316, 2001.

[GV91] Marie-Paule Gascuel and Anne Verroust. Animation and collision between complex deformable bodies. In *Graphics Interface 91*, pages 263–270, 1991.

[Hel68] H. Helmholtz. The origin of the correct interpretation of our sensory impressions. In Richard M. Warren and Roslyn P. Warren, editors, *Helmholtz on Perception: Its Physiology and Development*, pages 247–260. John Wiley and Sons, 1968.

[Hil99] Randall W. Hill. Modelling perceptual attention in virtual humans. In *8th Conference on Computer Generated Forces and Behavioural Representation*, 1999.

[HP88] David R. Haumann and Richard E. Parent. The behavioral test-bed: obtaining complex behavior from simple rules. *The visual computer*, 4:332–347, 1988.

[Kah73] Daniel Kahneman. *Attention and Effort.* Prentice-Hall Series in Experimental Psychology. Prentice-Hall, inc., 1973.

[KB82] J U Korein and N I Badler. Techniques for generating the goal-directed motion of articulated structures. *IEEE Computer Graphics and Applications*, pages 71–81, 1982.

[KB84] Doris H. U. Kochanek and Richard H. Bartels. Interpolating splines with local tension, continuity, and bias control. *Computer Graphics (Proceedings of SIGGRAPH 84)*, 18(3):33–41, July 1984. Held in Minneapolis, Minnesota.

[KJ98] James J. Kuffner Jr. Goal-directed navigation for animated characters using real-time path planning. In *Modelling and Motion Capture Techniques for Virtual Environments*, number 1537 in Lecture Notes in Artificial Intelligence, pages 171–186. Springer-Verlag, 1998.

[LCD96] L.He, M. F. Cohen, and D.Salesin. The virtual cinematographer: a paradigm for automated real-time camera control and directing. In *ACM SIGGRAPH*, pages 217–224, 1996.

[Lee76] D. N. Lee. A theory of visual control of breaking based on information about time to contact. *Perception*, 5:437–459, 1976.

[Lee80] D. N. Lee. Visuo-motor coordination in space time. In G. E. Stelmach and J. Requin, editors, *Tutorials in Motor Behavior*, Advances in Psychology, pages 281–295. North-Holland Publishing Company, 1980.

[LR81] D. N. Lee and P. E. Reddish. Plummeting gennets: A paradigm for ecological optics. *Nature*, 293:526–527, 1981.

[LS99] Jehee Lee and Sung Yong Shin. A hierarchical approach to interactive motion editing for human-like figures. In *ACM SIGGRAPH*, pages 39–48, 1999.

[LW89] Timothy C. Lethbridge and Colin Ware. A simple heuristically-based method for expressive stimulus response animation. *Computers and graphics*, 13:297–303, 1989.

[Lyn60] Kevin Lynch. *The image of the city.* MIT press, 1960.

[Mae94] Pattie Maes. Modeling adaptive autonomous agents. *Artificial Life Journal*, 1:135–162, 1994.

[Mae95] Pattie Maes. Artificial life meets entertainment: Lifelike autonomous agents. *Communications of the ACM*, 38(11):108–114, November 1995.

[Mar82] David Marr. *Vision.* W. H. Freeman and Company, New York, 1982.

[MCH94] Sang Mah, Thomas W. Calvert, and William Havens. A constraint-based reasoning framework for behavioral animation. *Computer Graphics Forum*, 13(5):315–324, December 1994.

[MD96] Peter McLeod and Zoltan Dienes. Do fielders know where to catch the ball or only how to get there? *Journal of Experimental Psychology: Human preception and performance*, 22(3):531–543, 1996.

[MDBP95] Pattie Maes, Trevor Darrell, Bruce Blumberg, and Alex Pentland. The alive system: Wireless, full-body interaction with autonomous agents. Technical Report 257, MIT Media Lab Perceptual Computing, 1995.

[Mes90] Neville De Mestre, editor. *The Mathematics of Projectiles in Sport*, volume 6 of *Australian Mathematical Society Lecture Series.* Cambridge University Press, 1990.

[Mey98] Jean-Arcady Meyer. The animat approach: Simulation of adaptive behavior in animals and robots. In F. Alexandre and J.D. Kant, editors, *Actes de la conférence Neurosciences Pour l'Ingénieur.*, 1998.

[Mil88] Gavin S. P. Miller. The motion dynamics of snakes and worms. In *ACM SIGGRAPH*, pages 169–173, 1988.

[Min95] Mark R. Mine. Virtual environment interaction techniques. Technical Report TR95-018, UNC Chapel Hill Computer Science, 1995.

[MJ00] Helen McBreen and Mervyn Jack. Empirical evaluation of animated agents in a multi-modal e-retail application. In *AAAI*, 2000.

[MPZ90] Micheal McKenna, Steve Pieper, and David Zeltzer. Control of a virtual actor: The roach. In *ACM Symposium on Interactive 3D Graphics*, pages 165–173, 1990.

[MSK95] Micheal K. McBeath, Dennis M. Shaffer, and Kaiser Mary K. How baseball outfielders determine where to run to catch fly balls. *Science*, 268:569–573, April 1995.

[MTT87] Nadia Magnenat-Thalmann and Daniel Thalmann. The direction of synthetic actors in the film rendez-vous à Montréal. *IEEE Computer Graphics and Applications*, 7(12):9–19, November 1987.

[NRTMT95] H. Noser, O. Renault, D. Thalmann, and N. Magnenat-Thalman. Navigation for digital actors based on synthetic vision, memory and learning. *Computers and Graphics*, 19(1):7–19, November 1995.

[NS71] David Noton and Lawrence Stark. Scanpaths in eye movements during pattern perception. *Science*, 171:308–311, 1971.

[Pai80] J. Paillard. The multi-channelling of visual cues and the organization of a visually guided repsonse. In Stelmach G. E. and Requin J., editors, *Tutorials in Motor Behavior*, Advances in Psychology. North-Holland Publishing Company, 1980.

[Pas98] Harold Pashler, editor. *Attention.* Psychology Press, 1998.

[PBMB94] Lieke Peper, Reinoud J. Bootsma, Daniel R. Mestre, and Frank C Bakker. Catching balls: How to get the hand to the right place at the right time. *Journal of Experimental Psychology: Human Perception and Performance*, 20(3):591–612, 1994.

[PC82] Micheal I. Posner and Yoav Cohen. Components of visual orienting. In *Attention and Performance X: Control of Language Processes, The proceedings of the 10th international symposium of attention and performance*, pages 531–556, July 1982.

[PD99] Tony Polichroniadis and Neil Dodgson. Motion blending using a classifier system. In *The 7th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media*, pages 225–232, 1999.

[Per95] Ken Perlin. Real time responsive animation with personality. *IEEE transactions on visualtization and Computer Graphics*, 1(1):5–15, March 1995.

[Pol99] Tony Polichroniadis. Integrating a multiagent communications architecture with the videotape metaphor for scripting animations. In *the proceedings of the Eurographics UK 17th annual conference*, 1999.

[Pol00] Tony Polichroniadis. *High Level Control of Virtual Actors.* PhD thesis, University of Cambridge Computer Laboratory, 2000.

[PPRN91] A. Patla, S. Prentice, C. Robinson, and J. Neufeld. Visual control of locomotion: Strategies for changing direction and for going over obstacles. *Journal of Exprimental Psychology: Human Perception and Performance*, 17(3):603–634, 1991.

[PV97] A. Patla and J Vickers. Where and when do we look as we approach and step over an obstacle in the travel path? *Neurophysiology*, 8(17):3661–3665, 1997.

[PW99] Zoran Popović and Andrew Witkin. Physically based motion transformation. In *ACM SIGGRAPH*, pages 11–20, 1999.

[Rey87] Craig W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In *ACM SIGGRAPH*, pages 25–33, 1987.

[Rey88] C. W. Reynolds. Not bumping into things. Notes from the ACM SIGGRAPH course on Physically Based Modeling, 1988.

[RGBC96] Charles Rose, Brian Guenter, Bobby Bodenheimer, and Micheal Cohen. Efficient generation of motion transitions using space-time constraints. In *ACM SIGGRAPH*, pages 147–154, 1996.

[RH91] Marc H. Raibert and Jessica K. Hodgins. Animation of dynamic legged locomotion. In *ACM SIGGRAPH*, pages 349–358, 1991.

[RMTT90] O. Renault, N. Magnenat-Thalmann, and D. Thalmann. A vision-based approach to behavioral animation. *The Journal of Visualization and Computer Animation*, 1(2):18–21, 1990.

[Spa89] D. L. Sparks. The neural control of orienting eye and head movements. In *Dahlen Conference on Motor Control*, 1989.

[SR99] Paul Scerri and Nancy E. Reed. The EASE actor development environment. Technical Report TACSIM-99-01, Real-time Systems Laborotory, Department of Computer Science Linköping University, October 1999.

[SSCB96] W. A. Sparrow, A. J. Shinkfield, S. Chow, and R. K. Begg. Characteristics of gait in stepping over obstacles. *Human Movement Science*, 15:605–622, 1996.

[SU93] Mel Slater and Martin Usoh. The influence of a virtual body on presence in immersive virtual environments. In *VR93:Virtual Reality International*, pages 34–42, 1993.

[SU94] Mel Slater and Martin Usoh. Body centred interaction in immersive virtual environments. In N. Magnenat Thalmann and D. Thalmann, editors, *Artificial Life and Virtual Reality*, pages 125–148. John Wiley and Sons, 1994.

[TC00] Franco Tecchia and Yiorgos Chrysanthou. Real-time rendering of densely populated urban environments. *Rendering Techniques 2000: 11th Eurographics Workshop on Rendering*, pages 83–88, June 2000.

[Thó98] Kristin Thórisson. Real-time decision making in multimodal face-to-face communication. In *second ACM international conference on autonomous agents*, pages 16–23, 1998.

[Tod81] James T. Todd. Visual information about moving objects. *Journal of Experimental Psychology: Human preception and performance*, 7:795–810, 1981.

[Tre90] J. R. Tresilian. Perceptual information for the timing of interceptive action. *Perception*, 19:223–239, 1990.

[Tre91] James R. Tresilian. Empirical and theoretical issues in the perception of time to contact. *Journal of Experimental Psychology: Human preception and performance*, 17(3):865–876, 1991.

[Tre95] James R. Tresilian. Study of servo-control strategy for projectile interception. *The quaterly journal of Experimental Psychology*, 48A:688–715, 1995.

[TRG96] D. Terzopoulos, T. Rabie, and R. Grzezczuk. Perception and learning in artificial animals. In *Artificial Life V: Proc. Fifth International Conference on the Sythesis and Simulation of living Systems*, May 1996.

[TT94] Xiaoyuan Tu and Demitri Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. In *ACM SIGGRAPH*, pages 43–49, 1994.

[VC98] Hannes Högni Vilhjálmsson and Justine Cassell. Bodychat: Autonomous communicative behaviors in avatars. In *second ACM international conference on autonomous agents*, 1998.

[vO90] C W A M van Overveld. A technique for motion specification in computer animation. *The Visual Computer*, 6:106–116, 1990.

[Wag82] H. Wagner. Flow field variables trigger landing in flies. *Nature*, 297:147–148, 1982.

[Wai99] Gordon R. Wainwright. *Teach Yourself Body Language.* Teach Yourself Books, 1999.

[War84] W. H. Warren. Percieving affordances: Visual guidance of stair climbing. *Journal of Experimental Psychology: Human preception and performance*, 10:683–703, 1984.

[War95] William H. Warren. Self-motion: Visual perception and visual control. In William Epstein and Sheena Rogers, editors, *Perception of Space and motion*, Handbook of perception and cognition, pages 263–325. academic press inc, 1995.

[Wil87] Jane Wilhelms. Using dynamic analysis for realistic animation of articulated bodies. *IEEE Computer Graphics and Applications*, 7(6):12–27, June 1987.

[WK88] Andrew Witkin and Micheal Kass. Space time constraints. In *ACM SIGGRAPH*, pages 159–168, 1988.

[Wol98] Jeremy M. Wolfe. Visual search. In Harold Pashler, editor, *Attention.* Psychology Press, 1998.

[WP95] Andrew Witkin and Zoran Popović. Motion warping. In *ACM SIGGRAPH*, pages 105–108, 1995.

[WS90] Jane Wilhelms and Robert Skinner. A “notion” for interactive behavioral animation control. *IEEE Computer Graphics and Applications*, 10(3):14–22, May 1990.

[WYL86] W. H. Warren, D. S. Young, and D. N. Lee. Visual control of step length during running over irregular terrain. *Journal of Experimental Psychology: Human preception and performance*, 12:259–266, 1986.

[Yar67] Alfred L. Yarbus. *Eye Movements and Vision.* Plenum Press, New York, 1967.

[YBBS00] Song-Yee Yoon, Robert C. Burke, Bruce M. Blumberg, and Geral E. Schneider. Interactive training for synthetic characters. In *AAAI National Conference on Artificial Intelligence*, 2000.

[YS99] Johan Ydrén and Paul Scerri. An editor for user friendly strategy creation. In Silvia Coradeschi, Tucker Balch, Gerhard Kraetzschmar, and Peter Stone, editors, *RoboCup-99 Team Descriptions. Simulation League.*, volume 4 of *Linköping Electronic Articles in Computer and Information Science*, pages 45–48. 1999.

[Zel82] David Zeltzer. Motor control techinques of figure animation. *IEEE Computer Graphics and Applications*, 2(9):53–59, November 1982.

[ZJ91] David Zeltzer and Micheal B Johnson. Motor planning: An archetecture for specifying and controlling the behavior of virtual actors. *The Journal of Visualization and Computer Animation*, 2:74–80, 1991.

Appendix A

Actor Architecture

This appendix describes the architecture of the *Geppetto* actor system within which the behavioural algorithms described in this dissertation have been developed. It is an animation system developed by Polichroniadis[Pol99] as a testbed for research into virtual actors. It is based on an network of communicating agents similar to Blumberg's[BG95] but it integrates this sort of control network with a keyframe animation system. It provides an architecture for representing agents and their communication as well as the representation of an actor and various methods for generating its movement. Most of the work described in sections A.1 and A.3, with the exception of the representation of the eye animation, was done by Polichroniadis not myself. The next section describes the fundamental communicating agent architecture, section A.2 describes how object in the environment are represented and section A.3 describes the representation of the actor.

A.1 The Agent and communication architecture

Gepetto is based on a set of *Agents*. An Agent is a self-contained program object which performs some function while communicating with other agents. An example agent is *StepOver* which determines when the actor has to step over an object and if so requests the appropriate change in the walking gait. Though they perform their actions independently the agents must communicate. This is done through a Communication Link or *Comm*. A Comm is a single, typed piece of information which is shared between agents. For example, the Step Over agent has a comm representing the height that the actor must step to clear an obstacle. Step Over writes to this and another agent reads it in order to implement the step. There is no limit to how many agents can read or write to a single Comm and new Agents can link to an existing Comm without the need for existing Agents knowing about it and so interacting with existing agent with out any change having to be made to these agents. Comms are grouped together into *Comm Bundles* which provide a logical grouping for the links.

The Geppetto Agent architecture supports animation, keyframing of actions and repeatability of actions. This allows users to set a keyframe for a particular value (represented as a Comm) which will affect the agents of the architecture. The user can also change these keyframes while keeping all the other keyframes the same. This can be done at any point in the animation and the user can scroll backwards and forwards throughout the animation. This is different from other agent systems where initial conditions are set up and then action proceeds in a linear fashion. This means that Comms have to store not only the current state but all the state for the entire animation. Also a change in the state at any part of the animation requires recalculation of all the frames after that point. To save space the Comms store their state in different ways. For example, *CommConst* has only one value for the entire animation and any change to it changes the entire animation and so it can be stored as a single variable. *CommContinuous* changes every frame and so needs an array with one value for every frame. A *CommKeyframed* is in between, only user keyframes are stored. The current value of a Comm is represented as a member variable of any type. Pointers to these variables can be cached in agents that use the Comm and so Comms can be used with only the overhead of a pointer dereference.

The Agents themselves perform various actions at various times. There are two basic triggers for an Agent to perform an action. *Advance Frame* is a trigger for an action to be performed at every frame of the animation, this is only performed when the frame is first calculated, after that values are cached and so the user can scroll backwards and forwards without triggering this action until some keyframe is changed requiring a recalculation. The other trigger is *ParamChanged* this represents the agent reacting to some event and is triggered when a certain Comm is changed. Each Comm has a list of dependent agents which are triggered when that Comm changes, any Agent can add itself to that list.

A.2 Environmental objects

The gepetto system contains an environment in which the actors move and perform their actions. This environment includes a variety of physical objects. These objects are any object that needs representation in the virtual environment, anything from household, inanimate objects such as cups or chair to the actors themselves.

The environment itself is represented by a *World* object which contains and manages the various objects. The objects themselves are represented graphically as *OpenInventor* meshes and internally as agents. The representation of objects is shown in figure A.1. Objects are guaranteed to have a number of properties available for other agents to interrogate. These include:

- A unique identifier.
- The Position of the Object calculated from the viewing transforms.
- The Orientation of the object also calculated from the viewing transforms.
- The two extreme points of the bounding box of the object, calculated from the mesh.
- The velocity of the object, calculated differently for different types of object.

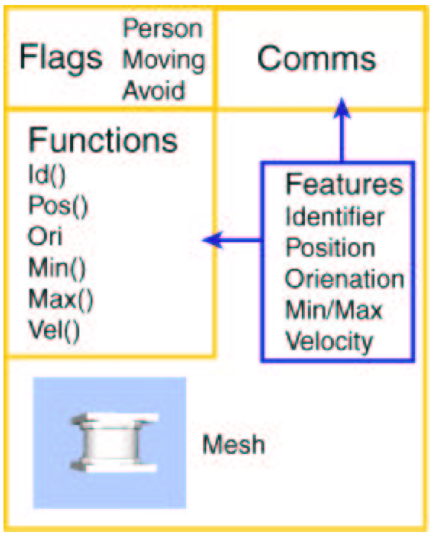


Figure A.1: The Object representation.

These properties are represented internally as Comms and are calculated directly from the display geometry of the mesh. Currently all values are calculated from the bounding boxes of the object, for efficiency, but new types of objects which use finer grained values could be added.

In order to speed certain tests objects also contain a number of flags that represent high level properties. For example, an actor does not have to bother trying to avoid collisions with the floor during normal walking (except, of course, if the actor falls over but this is rather different from tasks such as avoiding walking into a table). The main flags are:

- *person*, if the object is a person
- *moving* if the object is moving currently or might move.
- *avoid* if the actor must try to avoid walking into the object.
- *occluding* if the object significantly occludes other objects (see section 3.4.4)

These flags can speed tests by rejecting many objects based only on a boolean variable. However, for the most important flags the test is eliminated altogether as the *World* stores a number of lists of objects as well as the list of all objects. Each of these lists correspond to objects that have a certain flag set. For example, there is a list of people and a list of moving objects. Figure A.2 shows how objects are represented in the *World*.

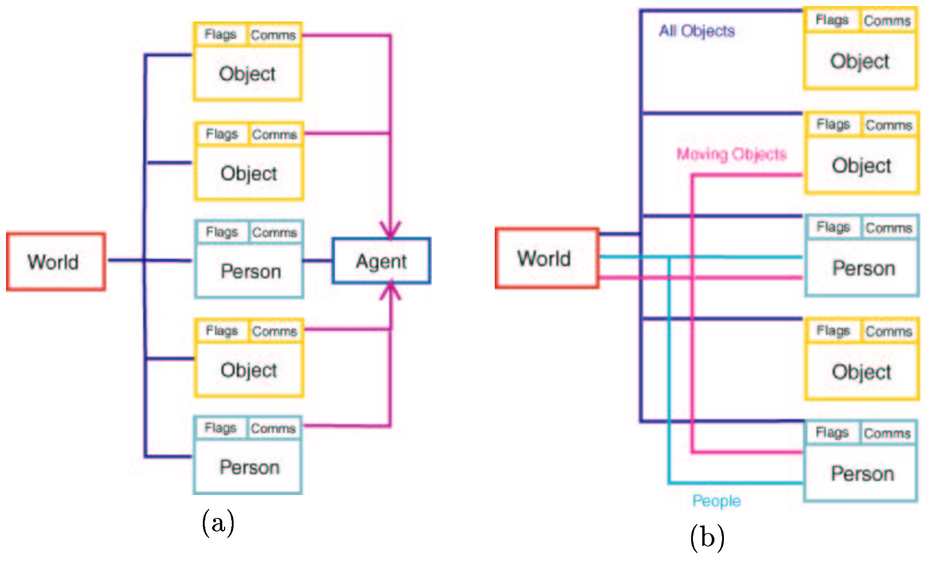


Figure A.2: The Object Architecture. (a) The Communications between Objects and Agents. The magenta lines are the connections between the Comms of the objects, representing features such as position and velocity, and the agents of the person which use these features. Only one agent is shown for clarity. There can be many such lists of object in the world corresponding to different types of object (b) The various lists of objects. The dark blue tree represents the list of all objects the others represent other lists based on properties of objects

A.3 The Actor representation

The actor, the humanoid character, is the central part of the Gepetto and there are many agents which deal with controlling the motion of the actor. At the lowest level the actor is displayed to the user as a body, a *mannequin* (see figure A.3). This consists of a series of segments connected by joints (for example the lower and upper arms, connected by the elbow joint). At the lowest level of control the Mannequin is moved by altering the angles of the various joints which moves the segments relative to each other (process known as forward kinematics), the positions of the joints are shown in figure A.3(a). The lowest levels of the actor hierarchy use this method of control internally but to the rest of the hierarchy the control of movement is done in terms of end effectors. These are seven positions on the actors body (head, neck, hips, left and right hand and left and right foot) whose position and orientation can be altered in order to control the movement of the Mannequin, the positions are shown in figure A.3(b). The representation of the posture of the Mannequin in terms of end effectors is converted into the joint angle representation by an *inverse kinematics* system.

At a higher level the motion of the actor is thought of in terms of pieces of motion. These are a series of keyframes for the position and orientation of the end effectors. They

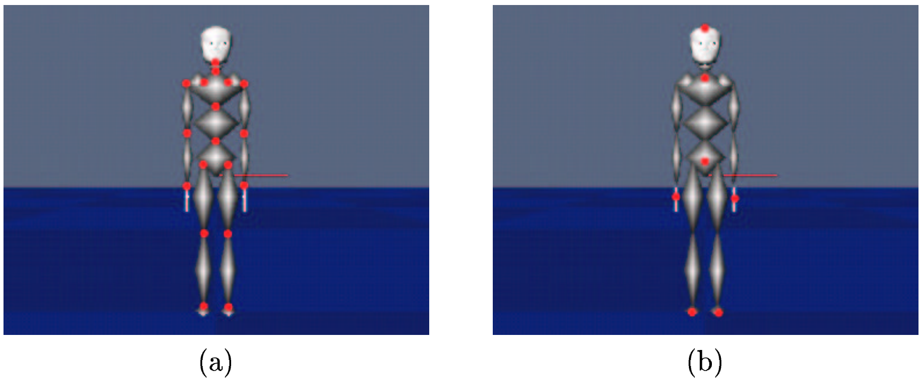


Figure A.3: The graphical representation of the actor, called the Mannequin. (a) shows the joints of the Mannequin. These are the points on the skeleton that can rotate and change the actors pose. Low level motion control is done in terms of the joints. (b) shows the end effectors, these are point on the body that can be moved to change the pose of the actor as a means of higher level motion control. Motion defined in terms of end effectors is converted to motion in terms of joints by an inverse kinematics system.

can be pre-existing pieces of motion such as motion captured motion, motion generated internally on the fly by the system or motion produced by transforming existing motion (such as the motion produced by the walk agent described in section 6.2.1). The actor can be controlled by starting or stopping these pieces of motion which can be played once or looped. The high level, keyframed, representation of the motion is converted into a representation which has an end effector position and orientation for each frame by the *Natural Movement* Agent which performs a cubic hermite spline interpolation on the keyframes. Different pieces of motion can also be blended into each other.

There are two Coordinate frames which are used to control the motion of the actor, a local coordinate frame to the actor and a global coordinate frame which is the same for all actors and which corresponds to the coordinate frame of the scene geometry. The origin of the local coordinate frame is called the *root* of the actor and it can be given any position and orientation relative to the global coordinate frame. It is not fixed to any particular part of the actors body, any end effector can be given and arbitrary position and orientation relative the root. This allows the origin of the coordinate frame to be moved a convenient position on the actor for controlling a particular action (it is normally positioned at the hips with its z-axis pointing forward, and in further discussion I will assume this unless I state otherwise). Most of the motion control is done in the local coordinates but it is necessary to convert to global coordinates for display and also so as to know the position of the actor relative of objects in the environment for some of the higher level Agents. This involves knowing what the position of the root is in global coordinates (see section 6.2.1 for some discussion of what this involves).

As well as body animation, the gepetto Mannequin has eyes which can be animated. These are controlled by giving a fixation point, the eyes are then rotated to face towards this point. This fixation point is given in a coordinate system which takes as its origin the actors head, though fixation points can be given in global coordinates and then converted into this head-based coordinate system.

Appendix B

Glossary of terms

The following is a glossary of terms used in this dissertation. Terms in **bold** are common terms in the literature. Terms in ***italics*** are introduced in this dissertation.

advance frame A function performed by each agent at the start of a new frame.

agent based artificial intelligence A system of artificial intelligence based on decomposing a system into small self contained competences.

agent A small self contained software competence which can communicate with other agents. They can work together to produce emergent behaviour.

articulation A connected series of rigid links used to control the motion of a character. Analogous to the character's skeleton.

attention The human ability to focus perception onto a single object or feature of the environment.

attention capture A shift of *attention* caused voluntarily or involuntarily by an external event.

attention manager The software system used to control the actor's attention. Described in chapter 4.

attention requests A message sent to the *attention manager* to request some sort of attention shift.

autonomous actor/agent A *virtual actor* whose behaviour is entirely autonomous, i.e. controlled by the computer.

avatar A *virtual actor* used to represent a user in a virtual environment.

avoid moving obstacles An agent of the navigation hierarchy that ensures that the actor avoids side on collisions with moving obstacles.

basic properties A number of properties of an object that any actor has access to, for example, position or velocity.

behavioural competences A simple behaviour pattern for a *virtual actor*. Generally simpler than a *behaviour*. (A method by which the user can design behavioural competences is described in chapter 5).

behaviours A single behaviour pattern for a *virtual actor*, for example, avoiding collisions with obstacles or searching for food.

behavioural animation A method of computer animation where the behaviour of a character is simulated in order to animate them.

body-scaled measures Some measure in the environment, for example distance from the observer, measured in terms of some physical feature of the observer, for example stride length, rather than some absolute measure.

Chapman's strategy A possible strategy by which a cricket fielder might run to catch a ball.

collaborative virtual environment A *virtual environment* used simultaneously be two or more people in which the users are embodied by *avatars* and in which they may communicate with each other.

collision detection An ambiguous phrase. It often means the process of detecting when two surfaces collide in order to stop them interpenetrating. However, it is also used, and is used in this dissertation, to mean the behaviour pattern whereby a *virtual actor* detects when it is about to collide with an object in order to take evasive action.

collision detector A visual *agent* that perform *collision detection*

comm A communication channel between agents.

comm bundles A collection of *Comms*.

contact A feature of a competence whereby the actor physically touches or in some way interacts with an object or location.

contact argument An object or location that the actor physically touches or in some way interacts during a competence. It may be changed to a new object or location for different invocations of the competence.

contact positions The point in a competence when the actor touches a contact argument.

differential parallactic displacements An optical feature that is believed to be used by humans to detect potential collisions.

ethology The study of animals in their environment. It has greatly influenced *agent based AI* and *behavioural animation*

eyeaction A *secondary action* that adds looking behaviour to a competence.

gazes A prolonged look. The word gaze is often used to refer to any look.

geppetto The *agent* based system of *virtual actor* animation in which the work described in this dissertation was implemented.

glances A short look.

ground An *agent* that keeps the actors feet from slipping on the ground and converts local motion of a planted foot into global motion of the entire actor.

handle action A *secondary action* that allows the actor to move objects.

immediate request an *attention request* that is executed on the frame immediately following it being sent.

inverse kinematics The process by which an actor is controlled by moving the end points of its *articulation*

kochenak curve A variant of a hermite interpolating curve.

linear optical trajectory (LOT) A possible strategy that might be used by cricket fielders to run to catch balls.

mannequin A representation of a *virtual actor* based on an *articulation*

monitoring The process of returning attention to a particular object or location a number of times.

motion capture A technique for animating characters that involves recording motion from a human actor.

motion tracker An *agent* that detects moving objects in peripheral vision and looks at them.

motion warp A transformation of a piece of motion.

object features A system for representing features of an object that are not well represented geometrically.

OpenInventor A 3D Graphics library produced by SGI and used in the *Gepetto* system.

optical frame In vision this is the the frame of reference of light falling on the retina that has been corrected for the rotation of the eye but not for other factor such as the orientation of the head. (compare with the retinal frame).

param changed A function of an *agent* that is called when a *comm* it is interested in changes.

path planning The process of pre-computing a path around an environment.

periods A section of a piece of motion used in a competence. A period ought to have some meaning distinguishing it from other periods.

peripheral vision The area of the visual field that is outside the focus of attention.

peripheral vision agent An agent that scans *peripheral vision* for objects with a certain property.

pop-out A psychological effect whereby it is possible to find certain objects in a display almost instantly.

rational speed curve A curve where the rate of change of arc-length with parameter value is a rational function.

retinal frame In vision this is the the frame of reference of light falling on the retina without being corrected for the orientation of the eye. (compare with the optical frame).

saccades A small, involuntary eye movement.

searching An attention behaviour that scans the environment in order to find objects of a certain type.

secondary actions A behaviour pattern that adds an extra aspect to the motion of a competence.

sensori-motor skills A behaviour pattern involving perception and motion.

shift peak A *motion warp* that moves the highest point of a footstep motion.

space-time constraints A system of *motion warping* that applies both spatial and temporal constraints in order to alter the motion.

speed modifier A *motion warp* that alters the speed of the motion.

spontaneous looking A term used by Chopra-Khullar and Badler, analogous to my *undirected attention.*

step over An *agent* that enables the actor to step over obstacles.

stimulus A minimal item of perceptual information.

subsumption architecture An architecture for *agent based AI*, developed by Rodney Brooks, in which agents are ordered in a hierarchy. Lower level agents perform actions where ever possible but when necessary higher level agents take over their function.

synthespian/synthetic actor Other words for *virtual actor.*

time-to-contact (TTC) A measure of the time until an object will collide with the observer, assuming constant velocities.

transform extent A *motion warp* that increases the amplitude of the motion.

TTC see *time-to-contact*

undirected attention An attention behaviour whereby the actor will choose objects or locations to attend to if there are no active requests from other agents.

virtual actor A representation of a character for computer animation. It can include physical appearance, motion control and behaviour among other things.

virtual environment A 3D computer generated graphical space which the user may view, move around and interact with.

visual τ A visual quantity that measures *time-to-contact*

walk around An *agent* that enables the actor to avoid frontal collusions by changing direction of motion.

walk generator An *agent* that generates walking motion.

world A representation of a scene which contains a number of *virtual actors* and physical objects.