Development Of A Java Based Simulation To Model The Interaction Of Flocks, And A Discussion Of Predators, Particularly Controlled Predators i.e. Sheepdogs.

Barrie Robertson

"BSc (Hons) in Mathematics and Computing"
University Of Bath
2005

Development Of A Java Based Simulation To Model The Interaction Of Flocks, And A Discussion Of Predators, Particularly Controlled Predators i.e. Sheepdogs.

Submitted by Barrie Robertson

COPYRIGHT

Declaration

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Batchelor of Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Signed ……

Signed …….

Abstract

Animals move in groups for a variety of reasons. Often a group is observed to be highly co-ordinated in their movement, as though the group actually manoeuvres as one being. Schooling, swarming and flocking are all descriptions of this type of behaviour, which one, is usually dependent on the species being described. For land animals, the term flock is most common. It is thought that flocking can be attributed to just a few underlying rules that each member of the flock obeys. This dissertation researches the possible candidates for such rules and attempts to apply them by creating a Java based simulation of a flock. The paper then progresses to discuss how the flock could then be attacked by a predatory species, or controlled by another dominant animal, as a dog herds sheep for instance.

Acknowledgements

I'd like to thank Dr Claire Willis for being understanding in times of difficulty and showing an active, enthusiastic interest in my work.

Thanks go to Rich Stone and Chris Feldwick for their sense of humour and support, always keeping me looking on the bright side.

Finally I'd like to thank Catherine Archer for her encouragement, reassurance and belief in me.

Contents

1. Introduction

Animals of the same species can often be found in large numbers, and there are some species that whilst even for the most part solitary, also gather to travel long distances together. Being in a group serves many purposes from both a social and survival standpoint. There are many reasons this may happen, but being in numbers clearly offers an increased safety for the individual. Being one of many does have some disadvantages, suddenly an individual is required to be more alert to the activity around them and competition for food, partners and territory can clearly become a problem.

However, when travelling, animals have been seen to display a level of teamwork often thought miraculous, managing to avoid each other, predators and other obstacles, whilst moving toward a common goal. This group behaviour has many names, but from now on will be referred to as flocking, and the group, as a flock.

Flocking has been a topic of computer research for almost 20 years. It is hoped that by studying how a flock operates we will better understand how to control large groups of our own species on the move. Crowd control at parks, festivals and stadium events needs to be properly planned and managed; the sheer volume of people in a confined space can be a dangerous situation. Similarly buildings need to be able to evacuate as efficiently as possible in an emergency, reducing the time needed to evacuate could clearly save lives. We appear to be on the move more and more, and traffic congestion both on the roads, and in the air is also reaching a critical safety level, but it would also be economically advantageous to reduce congestion.

Predators attempt to control and manipulate flocks in order to successfully isolate prey. This is achieved by creating disruption to the flock. However, in the real world we wish to control potentially chaotic situations, not disrupt them further. Hence it makes more sense to focus on circumstances where the flock is kept in check, not broken up. An obvious example of this is sheep farming, where dogs are used to control the movement of flocks of sheep, and reintroduce any wayward sheep back to the flock.

The behaviour of flocking although seemingly highly elaborate and complex is thought to be governed by a comparatively small set of laws. Each member of the flock is concerned with its immediate surroundings, and application of the rules ensures it moves safely within the group. From a distance this gives the impression that the flock is actually moving as one organism.

Section 2, the Literature Review, analyses articles from geographical and agricultural resources to identify a candidate set of governing rules. It looks at existing work in the field and the success of differing approaches. The 3 behaviours of evasion, cohesion and alignment are then highlighted as the pivotal influences that should be used to replicate flocking.

From there Section 3 discusses and outlines the requirements specification of the program and then is closely followed by Section 4, design, and Section 5, implementation. Section 4 focuses more on the mathematical nature of the problem and how best to represent the individual members of the flock, whilst considering what is realistically capable of being computed. Section 5 gives an overview of the code, paying attention to areas of change during implementation and indicating where the program could be extended in order to further the project.

Section 6 concludes that the weighting of the 3 core stimuli driving the flocking is critical to the behaviour of the group and the path which it takes. There then follows a discussion to how the simulated flock could be attacked by a predatory species, or controlled by another dominant animal, as a dog herds sheep for instance. This conversation deduces attacking and herding the flock are essentially opposite goals, and that predators and their behaviour will require deeper investigation, much of which needs to be sought outside of the computing world, starting with biological observations and reports, in order for the simulation to be realistic.

2. The Literature Review

2.1 Observed Findings

A flock is defined as 'animals of one kind as a group or unit', 'a large crowd of people', or 'people in the care of a priest or teacher' [1]. Hence it has also become the common name for a group of sheep, usually under the guidance of a shepherd. Similar terms used for other species are herd (usually land animals) and school (associated with aquatic life).

These terms are also often used to describe the motion of such groups of animals. A unit of flocking animals exhibits a grace in movement, appearing to move as a single organism. However each individual within the group is responsible for their own passage of travel. The intricacies of the interactions between the individuals is what allows flocks to travel as one.

Observations of schools of fish [3] led Partridge to make the following conclusions:

- Schooling serves to reduce the risk of being eaten.
- It takes 3 fish to form a school, but there is no limit to the size of the school.
- There appears to be two relatively simple rules governing schooling, namely each fish maintains an empty space around itself and its velocity is constantly adjusted to that of the average of the school.

With regards to the rules further remarks were made. The space maintained around a fish does not lead to the school taking the form of a regular geometric lattice. Rather there exists a 'preferred' distance and angle that a fish attempts to maintain to its nearest neighbour, usually around one body length and at an angle of 90 degrees. These are not rigidly maintained, but were noticed as a statistical average. Further there is only one nearest neighbour, and the ratio of distance between the nearest neighbour and 2nd nearest neighbour also seemed to take a statistical average, but varied between species.

Velocity was constantly adjusted by taking an average of the entire schools' velocity, not as had been previously thought by considering just the nearest neighbours. The contribution each member of the school makes to the acceleration of the individual is inversely proportional to the square/cube of the distance between them, depending on the senses the fish used to monitor this. In cases where sight was considerably dominant it was inversely proportional to the square.

There is no suggestion that the observations made about schools of fish do not apply more commonly to all groups of schooling/herding/flocking animals such as tadpoles, birds and livestock.

Grandin et al [4] claim that low stress cattle handling techniques to achieve a desired motion from an animal are based on instinctual behaviour patterns.

Cattle, as a prey species, have evolved to protect themselves from predators. Some instinctual behaviour patterns are very rigid and fixed and others can be modified by learning.

They have identified three basic instinctual behaviours which cattle adopt to avoid predators:

1. The maintaining of a flight zone and the tendency to face perceived threats
2. The point of balance at the shoulder and its effect on movement direction
3. The tendency to bunch together when they are threatened.

The flight zone of an animal is effectively its personal safe space. The size of a flight zone varies between individual animals, it has been suggested that the tamer an animal the smaller flight zone it has [4]. The size of the flight zone also depends on the state of the animal; an excitable or nervous creature has a larger flight zone. If a perceived threat is identified an animal will turn to face it, if it should enter the flight zone the animal will turn and begin to move away.

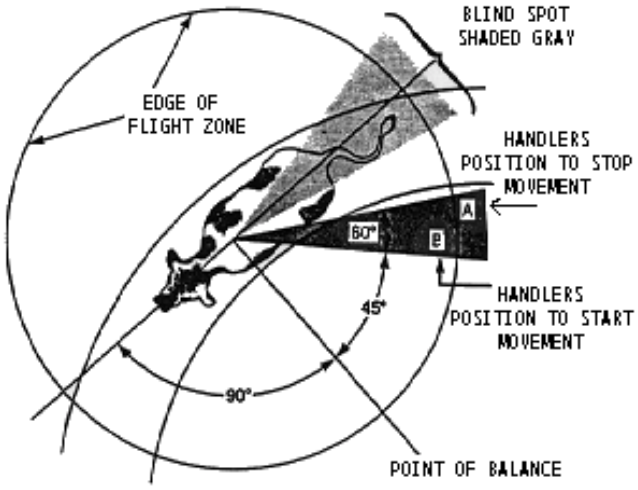

Figure 2.1: The Flight Zone & Point of Balance of an Animal

The point of balance is at an animal's shoulder. The purpose of the point of balance is to assist in the escape from a predator; an impala chased by a lion will run in the opposite direction when a lion passes it shoulder. All species of livestock will move forward if the handler stands behind the point of balance and they will back up if the handler stands in front it. Further, walking in the opposite direction tends to speed up movement and walking in the same direction tends to slow an animal down. These principles work with all herding animals.

A group of cattle moving as a herd maintains eye contact with each other, that way the entire herd can move as a coordinated whole. The next animal behind the leader is positioned just behind the leader's point of balance.

Observations of schooling fish [3] identified a number of traits and tactics to deal with predators. The most prominent trait was that the flock had increased polarization when a predator was nearby. This served mainly to increase the level of alertness to the position of the predator, by increasing the number of eyes on the look out. The group, once polarized was then better equipped to deal with attacks.

There were details of reactions to two types of attack. The first was a quick attack, a sharp penetration into the flock. In this situation all members of the flock made a burst of speed away from the group, visually creating a 'bomb blast'. The direction of the movement in this scenario is presumably to decrease the chance of collisions at such high speeds.

The second tactic was more complicated and is used in the case of a slow approach to a predator that has got too close. It is commonly called the 'fountain effect'. It starts with the flock moving as one, all in the same direction, with the predator approaching from behind. As the predator draws closer the flock split into two, down the centre in the direction of travel. The left hand side of the flock arc out left, turning a full 180 degrees, whilst moving wide enough to keep a safe distance from the predator. They then move past the predator, in the opposite direction, before arcing another 180 degrees to be swimming directly behind the predator, in the same direction. The right hand side of the flock does exactly the same, except that it arcs in the other direction.

Turning and facing a potential threat enables the animal to keep track of where the predator is. This behaviour can be seen below, as two handlers enter a pen full of sheep. Notice that the sheep are circling around the handlers whilst maintaining a safe distance and keeping the people in sight. Note that the sheep are moving in the opposite direction of the handlers.

Figure 2.2: The 'Fountain Effect'

2.2 Previous Works

Back in 1986 Reynolds created an impressive simulated flock of birds [2]. The flock is precisely described as ‘a group of objects that exhibit a general class of polarized, non-colliding, aggregate motion’. The simulation is an elaboration of a particle system.

There are two main differences between the boids (bird-oids) and a particle system, the objects now have geometrical shape and hence orientation and the boids also interact with each other, which particles do not. In other words a boid’s behaviour depends not only on its own internal state (as a particle does), but also the external state of the flock. Each boid navigates according to its local perception of the dynamic environment and the set of rules and behaviours programmed by the ‘animator’. The resulting flocking is simply the interaction between the individual boids.

Reynolds notes that natural flocks never become full, and the amount of ‘thinking’ that a bird does in order to flock is largely independent of the number of members in the flock. In contrast the complexity of the simulated flock definitely shows an upper bound.

After creating a boid that supported geometric flight Reynolds added behaviours to his boids in order to make them flock. They were:

1. Collision Avoidance; avoid collisions with nearby flock mates
2. Velocity Matching; attempt to match velocity with nearby flock mates
3. Flock Centring; attempt to stay close to nearby flock mates

Flock centring is not the desire for a boid to move to the centre of the flock as might be expected, rather as each boid has a localised perception the ‘centre of the flock’ is actually the centre of nearby flock mates. In nature there are many reasons why flocks may split up and a simulated flock must be able to replicate this. The flock centring behaviour above allows this to occur as a boid does not care if the rest of the flock turns away as long as it stays close to its neighbours.

Note that the behaviours that make up the flocking model are stated in terms of "nearby flock mates." In the implementation, the neighbourhood is defined as a spherical zone of sensitivity centred at the boid's local origin. The magnitude of the sensitivity is defined as an inverse exponential of distance, implementing observations made in [3]. Hence the neighbourhood is defined by two parameters: a radius and exponent.

Observations made by the paper [2] include:

- Static collision avoidance and dynamic velocity matching are complementary. The static collision avoidance (initial positions of the flock members) establishes a minimum separation distance; the velocity matching maintains the separation.
- The aggregate motion we intuitively recognize as "flocking" (or schooling or herding) depends upon a limited, localized view of the world.
- The field of sensitivity should realistically be exaggerated in the forward direction and probably by an amount proportional to the speed of the boid. A forward-weighted sensitivity zone would probably also improve the behaviour in the current implementation of boids at the leading edge of a flock, who tend to get distracted by the flock behind them.

2.3 Conclusion

Due to the nature of flocking animals, it would appear that a flock consists of numerous instances of the same creature interacting with its local environment. As such it would seem logical to take an object-oriented approach to creating a simulation. A flock would then consist of multiple initialisations of the same object. A predator could then be a similar object, regarding representation and animation, with a different set of behavioural rules.

As everything is considered on a reduced or 'local' scale it would be advantageous to utilise the object's own coordinate system, measuring things relative to the sheep's own position and state, especially when considering new acceleration requests.

As the boids are simulations of real creatures considerations such as maximum acceleration and velocity should be implemented.

The main influencing factors that would make acceleration requests appear to be:

- Evasion; to avoid all obstacles, and maintain a personal space
- Cohesion; to remain close to the group
- Alignment; to match the average direction of the group

In previous work [2] each instigated behaviour is reflected by making an acceleration request to the boid. The requests are weighted according to precedence and then averaged to give an acceleration vector. Problems were encountered where acceleration requests occurred in opposite directions and thus cancelled each other out, resulting in little change of direction. This mainly occurred when trying to avoid an obstacle, and if not rectified led to a collision.

Collision avoidance can be considered as two separate behaviours; static object collision avoidance and dynamic object collision avoidance. In many cases the static object collision avoidance would have no acceleration request, but when a fixed object is encountered the weighting of the request should be significantly high as to make the other requests simple fine tunings of the motion, opposed to having any real control, until the obstacle has been successfully passed.

There are a few different ways to model obstacle avoidance. A simple solution is to model everything as having a force field[11]. The force fields then repulse other objects, such as boids. The mathematics behind creating the correct acceleration request is fairly straight forward. However problems with the model have been identified [2]. If a boid approaches an obstacle at an angle exactly opposite to the force field, then the force field only slows the boid by accelerating it backwards and provides no side thrust at all. Not only that but boids should notice and turn away from walls as they approach them, but the wall should be ignored if the boid is flying alongside it.

Reynolds opted for a steer-to-avoid model, because it is a better simulation of a natural animal guided by vision. However, his model had no boundaries; the obstacles could be navigated around and passed. This gives rise to a further segmentation of static obstacle, those that can be passed and those that cannot, like walls. An impassable object dictates that animals radically change their direction, not simply loop around it. Therefore the force field approach will be retained for impassable objects. Avoiding the head-on slowing down should occur naturally by a small side thrust being initiated by the rest of the flock.

3. Requirements Specification

In Section 2 some key necessities emerged that must be included in the simulation. Here they are more formally stated and added to.

The intention is to replicate a flock of sheep, as a basis for discussion for further work. In the modern age most domestic grazing animals are kept in a confined space, a field. It therefore naturally follows that the simulation should be hosted in a field, with boundaries, beyond which the sheep cannot travel.

Natural movement of land-based animals (when ignoring jumping) is controllably in two dimensions. The landscape of the ground controls the third dimension. As such, for simplicity consider the field to be flat and the problem is reduced to dealing with two dimensions.

There needs to exist at least three animals within the field for possible flocking to occur.

Clearly vital information needs to be kept concerning each member of the flock, namely the position, velocity and acceleration.

Each member needs a graphical representation and to be animated in real time.

The creation of an object class, capable of representing all types of animals, could encapsulate the two above functionalities. In this way all animals can be illustrated and animated similarly. The behaviours that cause them to act differently, namely their acceleration requests can then be more precisely defined in extensions of the class. Within these extensions attributes such as maximum velocity and acceleration can be set, and if deemed necessary in the future, the graphical representation of a species could be over-written.

Each animal should consider the ‘world’ within its own co-ordinate space, or frame of reference. This will help to facilitate the desire to react to the animal’s local space.

Acceleration requests should be made based on three behaviours, evasion, cohesion and alignment.

Specifically there exist three types of evasion requests; dynamic object avoidance, impassable static object avoidance and passable static object avoidance.

Dynamic object avoidance can be modelled as the opposite of cohesion, which is to increase the space between two animals, not decrease it. In this way flight zones and the need to maintain a space to the nearest neighbour automatically occur.

The boundaries of the field, as impassable objects, should be programmed as having a force field, repelling the animals as they near them.

Any passable static objects should be modelled with steer-to-avoid acceleration requests.

Cohesion is to consider only ‘local’ members of the flock and attempt to move to the centre of these.

Alignment is to consider every member of the flock, and re-align to the average heading. The influence of a particular member should be taken as being inversely proportional to the square of the distance between the two flock members. This is a result of attributing the sensory intake to the vision of the animal, working as we are in a two dimensional space.

Until the design and implementation of predators occurs it appears unnecessary to consider point of balance and the influence of increased awareness to the front of an animal, where the senses are more acute. However, these shouldn’t be forgotten and some small consideration should be given has to how these could be introduced at a later stage.

4. Design

The most basic stipulation of the requirements is to be able to represent the animals graphically on the screen. This has to be done in such a way that the coordinates can be manipulated to accurately represent movement and facilitate acceleration.

As such the number of points of the graphical representation needs to be kept to a minimum.

It seems obvious that direction also needs to be indicated through the illustration, even if just looking at a snap shot.

The simplest geometric shape that appears to fulfil these criteria is an isosceles triangle. The triangle, as depicted overleaf in figure 5.1, has a natural centre of mass, making it an ideal point to model movement and acceleration through. It also provides a natural position for a shoulder, so that point of balance can be considered later.

The co-ordinates for each point of the triangle can be simply calculated in relation to the centre of mass, and thus facilitates displaying the shape on the screen.

The fact that one vertex of the triangle is further from the centre of mass gives rise to a natural interpretation of direction, and thus this vertex can be called the 'Nose' of the animal. The other two vertices can be denoted the 'Left Foot' and 'Right Foot' respectively.

Initially the nose seems an obvious candidate to compare with the centre of mass in the calculation of movement and acceleration. However the nose differs to the centre of mass in only one co-ordinate direction. A frame of reference requires two things, an origin and a basis vector for each dimension, in this case two. The nose cannot supply the second basis vector.

Conveniently, not only can one of the feet supply two basis vectors, but the right foot actually provides two, positive, orthonormal vectors as a basis, this is ideal.

Given the centre of mass and right foot of any animal, the heading, Θ, of the animal can be calculated. Thus all points within the field can be reconfigured to the frame of reference of that specific animal with simple conversions, $e_1 = (\text{Cos } \Theta, \text{Sin } \Theta)$ and $e_2 = (-\text{Sin } \Theta, \text{Cos } \Theta)$. This is done by creating a transformation matrix whose columns consist of e_1 and e_2 transposed and multiplying by the co-ordinate vectors of the flock members' centre of masses and right feet.

Figure 5.1: Initial Representation Of Animal

Figure 5.2: Revised Representation Of Animal

Yet consider that you need to firstly calculate Θ, then take a copy of every animal in the flock and transform it using the transformation matrix, calculate Θ of each animal in the flock in the subject animal's frame of reference, then compare each animal to the current subject animal to gain acceleration requests, compute the acceleration, and then reconvert the subject animal back to the natural frame of reference to update its calculated acceleration.

Now remember this all has to be done for each and every member of the flock. That appears to be a lot of operations to accelerate the system, which is not to move and update the members of the flock, just to accelerate them, and without the detail of how the acceleration is calculated.

This is clearly a very inefficient design.

In order to simplify the problem we must abandon the idea of considering each flock member in its own frame of reference. In order to make calculations as simple and thus as efficient as possible consider abandoning the centre of mass as the focal point of the animal, and instead use the centre of the base of the triangle, the mid-point between the two feet, here on referred to as the 'Tail'.

As illustrated in figure 5.2 the position of each vertex can now be calculated by just knowing the co-ordinates of the tail and the heading of the animal. Now the velocity of the animal can be calculated by just the heading and speed of the animal. The acceleration of the animal is simply a change in heading and a change in speed.

Suddenly there is the need to store just four values to be able to fully represent and manipulate an animal, namely the two values for the co-ordinates of the tail, one value for the heading, and one for the speed. This is a vast reduction on the myriad of values previously needed under the frame of reference system.

Given the four values x, y, Θ, v, which represent the x co-ordinate of the tail, y co-ordinate of the tail, heading of the animal and speed of the animal respectively, the velocity of the animal is given by (vSinΘ, -vCosΘ). That is the position of the animal one time phase later is (x + vSinΘ, y – vCosΘ).

Under the proposed system it is left to consider acceleration requests, which are specific to sheep.

By taking v to remain constant all acceleration requests become a change in Θ.

Alignment needs to consider every other member of the flock. The distance between the two sheep is calculated simply as the distance between the two tails, given simply by Pythagoras' theorem. The difference between the heading of each animal is calculated, a mere subtraction. The change of heading that the sheep being accelerated needs to undergo is then multiplied by a factor inversely proportional to the distance between them squared. This process is repeated for every member of the flock. All the scaled change of headings are then added together to produce an average change of heading, that re-aligns the sheep with the average heading of the flock.

Cohesion needs only to consider the members of the flock within a fixed locality of the sheep being accelerated. Thus each member of the flock is checked to see if it is close enough to have an influence. The distance between the two sheep is calculated as for alignment. If it is small enough the co-ordinates of the flock member are added to a temporary store. Once the process is repeated for the entire flock the temporary store is a total of the co-ordinates of each sheep close enough to have an effect. These summed co-ordinates are then divided by the number of sets of co-ordinates that went to making them up, producing an average co-ordinate. Using trigonometry the angle between the tail of the sheep being accelerated and the average co-ordinate is calculated. The change in heading needed to proceed in this direction is given by another simple subtraction, and is the acceleration request from the cohesion behaviour.

Dynamic object avoidance is similar to cohesion. This time the flock member is closer to the accelerated sheep and instead of moving towards the average co-ordinate the sheep accelerates in the opposite direction, another 180 degrees rotation to that calculated for cohesion.

Static non-passable object avoidance essentially means walls. For each sheep the distance to all four walls is calculated, this is simpler than calculating the distance between two sheep as it needs only consider one co-ordinate of the tail, the x or the y. If a wall is deemed close enough an acceleration request is made to turn the sheep in the direction travelling precisely away from the wall. This is scaled however by a factor inversely proportional to the distance from the wall, hence the closer the sheep to the wall the sharper it will turn.

Static passable objects are not to be instigated in the simulation, as they will bear little relevance to the discussion concerning predators, and so steer-to-avoid behaviour for them need not be calculated.

5. Implementation

The simulation has been implemented in Java, mainly due to the tasks object-oriented demeanour, but also due, in part to the familiarity with the language.

Due to the nature of flocking there is not a set of tests you can create to identify whether the group is acting as expected. Flocking is best analysed visually, we seem to intuitively recognise a flock when we see one. Hence if a behaviour looked wrong running, after initial coding, then I accepted it as incorrect. The code was then routinely analysed, re-planned, and re-written, until the sheep on the screen acted as I would expect them to. Consequently the process of implementation also systematically became a testing procedure, with a cyclic method of plan, code, run and evaluate until I was happy with what had been created before moving on.

5.1 Field

A window seemed a natural choice to represent the field, and with a desire to create a stand alone application, as opposed to an applet imbedded in a web page, a JFrame became an obvious choice.

The *paint()* method is used to display graphics on the screen, initially just drawing the members of the flock on. It proved useful to know exactly where the sheep were and the precise direction they were heading, as numerical values helped to complement the visual analysing of the method. For instance if *alignSheep()* is the only acceleration request sought then after a period of time all the sheep should end up on the same heading. To assist in identifying which data accompanied which sheep a small *if, else* structure is used to set the colour used to draw the sheep and data.

It was all well drawing one capture of the flock but to animate the field was the desired task. After reading around animation techniques [12] the logical approach to animating the JFrame was to give it an instance of the type Timer and implement the ActionListener class. The Timer produces ActionEvents at set intervals, these can then be handled by an *actionPerformed()* method. This holds instructions on what to do upon each occurrence of an ActionEvent. Thus this is where the body of the computation resides, moving and accelerating the flock, and then calling the *repaint()* method to update the JFrame with the correct information. By setting the Timer to different intervals the animation could be sped up or slowed down, depending on whether it was necessary to analyse behaviour over a long time scale or a step-by-step. The Timer has been left at a 2 second interval.

Originally all the acceleration commands were placed in the *actionPerformed()* method, but this caused over complication and was very messy. Ideally accelerating an animal should be a method of the animal. However the nature of the task would then require the passing of the entire flock to a single sheep, and this does not sit well with the object-oriented nature of Java. A flock has many sheep, and so a new object Flock was created, and methods of the new class were called to handle the movement and acceleration of the sheep.

5.2 Animal & Sheep

An Animal object simply stores the four necessary values for an animal to be animated, namely the x co-ordinate of the tail, the y co-ordinate of the tail, the heading of the animal, and the animal's speed.

It has the expected *set()* and *get()* methods. Notice that *setTheta()* makes certain the heading, theta is always in the range of 0 to 360 degrees. This helps ensure that acceleration requests are not unfairly influencing because they are greater than 360 degrees. It also enabled the acceleration requests to send negative requests, that is a rotation anti-clockwise instead of clockwise, and they would be treated as a large clockwise rotation, for instance -15 degrees becomes 345 degrees. It was found necessary to deal with negative angles during the body of the accelerations, in order to achieve accurate requests, however to keep things simplified it was necessary to rectify negative angles as quickly as possible to avoid inaccurate calculations elsewhere.

Perhaps unexpectantly there appears the methods *xValues()* and *yValues()*. In order to keep the *paint()* method in Field as simple as possible it was natural to get the co-ordinates of the vertices from the animal, opposed to calculate them explicitly on each call.

Class Sheep is simply an extension of Animal, inheriting all the methods of an animal, but with values representing the top speed and the maximum angular acceleration defined.

5.3 Flock

After realising that a sheep cannot accelerate without being aware of its surroundings acceleration ceased to be a method of a sheep and instead a method of the flock. The entire group must accelerate together, simultaneously. Indeed it is the interaction of the members of the flock which have bearing to how a sheep progresses.

The minimum number of animals needed to flock is 3, and so this is the number of sheep created in a non-argument call to the constructor.

The class is essentially made up of the constituent pieces that a flock's acceleration has been segmented to. *accelFlock()* calls *to accelSheep()* one animal at a time, the *accelSheep()* method then calls the methods which control the evasion, cohesion and alignment behaviour of that sheep, the workings of which were described in section 4. The manner in which each behaviour's generated acceleration request is dealt with is then handled in *accelSheep()* when all requests have been received.

6. Conclusion

The simulation has met with limited success.

When analysed individually each acceleration request performs as required, although problems occur from conflicting calls to change the angle from two adjacent sides if a member of the flock gets too close to a corner. A possible solution to this is to code every boundary to turn an oncoming animal clockwise, then as the animal heads into a corner both sides will be telling it to head in the same direction. However if the animal was approaching the wall with its right shoulder it is more natural to expect the sheep to turn left, the shortest way to avoid the collision. Perhaps to code a separate evasion of corners is the only way to accurately deal with the situation. The sheep also exhibit an undesired behaviour of turning away from a wall they are travelling parallel to. Although if the evasion of walls was written to turn an oncoming animal to a direction parallel to the wall instead of away from it, this may solve the problem.

Alignment works particularly well and this is likely to be due to the fact that this behaviour is the best researched and described in the research literature consulted.

However the current weighting of all three acceleration demands is not close to exhibiting true flocking behaviour. Qualitative analysis needs to be attributed quantative scores to enable statistical analysis of routine testing of different combinatorial equations.

Putting predators into simulations has been attempted, and met with some success under the various approaches.

Upon research the most common approach was to create a grid, with nodes where lines intersected. The predator and prey were objects that could occupy a node and move to adjacent nodes. This provided a simple basis upon which accurate mathematical techniques could be applied and algorithms created and refined to achieve a desired result.

Kachroo et al [6, 7] first took a 3x3 and then an NxN grid with the objective of herding the prey to the origin. In addition the objects were also allowed to move diagonally across the grid. They modelled every possibility of prey position relative to predator position with an equation to produce probabilities of the next movement of each. They then implemented 3 different search algorithms to assess the cost of each possible movement to further influence the predator behaviour in order to herd the prey. The result was that the pursuer moves to the state of the lowest cost of all adjacent states, then checks to see if the system is in an equilibrium state, before the process continues until the final state is achieved. The work is highly mathematical and whilst achieving the goal does not accurately simulate real behaviour between predator and prey in open space.

Sahin & Bay [8, 9, 10] have conducted extensive research into the use of Bayesian networks and influence diagrams to create an intelligence model. Software was then written to produce an intelligent agent, which was used to simulate a dog herding a sheep. Throughout their work they remain with a fixed grid of 3x3 size, comprising 16 nodes and the predator and prey are restricted to 5 movements, up, down, left, right and stay.

The intelligent agent has 5 levels; sensors, belief, preference, capabilities and actions. Each agent has an influence diagram, consisting of decision nodes, chance nodes, and value nodes. Decision nodes represent choices available to the agent, chance nodes are random variables or uncertain quantities, and the value node is an objective. The idea is to maximise the value node by appropriately selecting values of the decision nodes.

The belief level consists of a Bayesian network, used to model part of the world that the agent exists within, and the relations within the model reflect causal impacts between events. It creates probabilities for each of these events, which help in decision making. Effectively an automation of what occurred in [6, 7].

The influence diagrams and Bayesian network of the predator was enabled to be intelligent by allowing it to observe and record the behaviour of prey, effectively learning the prey's influence diagram. The learning is achieved by maintaining a database of start states and possible next states, and calculating which next state is most likely. The mathematics behind this is relatively simple, yet effective.

Once implemented the intelligent agent was not only successful in herding its prey, but the more times it ran, the more experience it had and thus it's knowledge bank built, the more efficient at herding the prey it was.

Subsequent work refined the process and carried out statistical analysis of the approach in order to improve it, but the basic idea remained.

Away from the gird approach Parenthoen et al [5] use fuzzy cognitive maps as tools to model emotional behaviours of virtual actors. They delocalized fuzzy cognitive maps on each agent level to model autonomous agents within a virtual world. The paper identifies autonomous entities as being one of the keys for believable simulations, and that this is reliant on sensorimotor autonomy. That is, each object is equipped with sensors and effectors enabling it to be acted upon and react to its environment, independently of the other objects. It further points out that the agents and fuzzy cognitive maps are autonomous entities, the first having sensors and effectors to decide its behaviour, the latter has perception, motor and emotional concepts. The work then uses fuzzy cognitive maps to aid in decision making, by fuzzyfication of sensors in the agent and defuzzyfication of the motor concept within fuzzy cognitive maps to give the agent effectors more realistic responses.

The researchers then build perception models, built upon sensation against perception. Sensation being the result from sensors alone, whilst perception is sensation influenced by internal state. A model for sheep and a model for a dog were created, implemented and run to create a reasonable simulation of a sheepdog moving a group of 3 sheep into a desired area. It must be noted that the sheep were not simulating flocking, but were separate entities, however the dog did achieve to move them all simultaneously in the same direction.

The Bud Williams Methods, found at Appendix A, gives an agricultural viewpoint of how to herd livestock and thus a benchmark for any type of artificial herding to try and recreate.

Clearly ideas in [5] and [8, 9, 10] work very well and future work should seek some form of amalgamation, the fuzzy cognitive maps allow the herding to be applied on a large simulated environment, not on a grid, whilst the influence diagrams and Bayesian networks ensure the predator is intelligent.

7. Bibliography

1. Thompson , D., The Pocket Oxford Dictionary, Oxford University Press, 8th Edition, 1996.

2. Reynolds, C., Flocks, herds, and schools: a distributed behavioural model, *ComputerGraphics,* July 1987, 21(4), 25-34.

3. Partridge, B.L., "The structure and function of fish schools," Scientific American, June 1982, pp. 90-98.

4. www.grandin.com

5. Parenthoen, M. et al, Put Fuzzy Cognitive Maps to Work in Virtual Worlds, 2001 IEEE Fuzzy Systems Conference.

6. Kachroo, P. et al, Dynamic programming solution for a class of pursuit evasion problems: The herding problem, IEEE Transactions on Systems, Man, And Cybernetics-Part C, Vol.31, No.1, Feb 2001.

7. Kachroo, P. et al, Pursuit Evasion: The Herding Noncooperative Dynamic Game-The Stochastic Model, IEEE Transactions on Systems, Man, And Cybernetics-Part C, Vol.32, No.1, Feb 2002.

8. Sahin, F., Bay, B.S., A Biological Decision-Theoretic Intelligent Agent Solution to a Herding Problem in the Context of Distributed Multi-Agent Systems, in SMC 2000, IEEE International Conference of Systems, Man and Cybernetics, Oct 2000.

9. Sahin, F., Bay, B.S., Learning from experience using a decision-theoretic intelligent agent in multi-agent systems, SMCCia/01, Mountain Workshop on Soft Computing in Industrial Applications, Virginia, June 2001.

10. Sahin, F., Bay, B.S., Structural Bayesian network learning in a biological decision-theoretic intelligent agent and its application to a herding problem in the context of distributed multi-agent systems,

11. Hodgins, J. K., Wooten, W. L., Brogan, D. C., O'Brien, J. F., Animating Human Athletics. Siggraph '95 (1995)

12. Deitel, H.M., Deitel, P.J., Java: How To Program, Prentice Hall, 4th Edition, 2002.

8. Appendices

Appendix A – The Bud Williams Methods, as described by Grandin et al [4].

The Bud Williams methods of calm, slow movement of cattle on pastures can be defined as a stimulus-response relationship. In cattle that have had no previous experience with herding, the "stimulus" is a person who simulates predator "stalking behaviour", which elicits predatory "avoidance behaviour" in the cattle.

First locate the herd. Then begin a slow survey of the herd by walking in a circular direction around the herd. The behaviour of the predator circling the herd causes anxiety in the animals. The cattle become uneasy over an impending attack by the predator and begin to loosely bunch together. This uneasiness and slight anxiety comes before the fear and flight elicited by an actual attack. It is important to remember before attempting to use these methods that it is anxiety that makes this technique work and not fear. When the method is first used it triggers instinctual bunching behaviour. The more a person works with the cattle, the calmer they become and instinctual bunching behaviour is gradually replaced with calm learned behaviour.

There are three steps in the process of moving cattle on large pastures:

1. Gathering and Loose Bunching:

This is the most critical step. The majority of the herd must be loosely bunched before any attempt is made to move the herd. Depending on herd size, wildness of the cattle, and the terrain, it will usually take 5 to 20 minutes to induce the herd to form a loose bunch. This is accomplished by applying very light pressure on the edge of the collective flight zone to induce the animals to move into a loose bunch.

The handler should locate the majority of the herd and start making a series of wide back and forth movements on the edge of the herd. You should move in the pattern of a giant windshield wiper. The arc of the zigzag movement must not exceed ninety degrees. Do not circle around the cattle. The movement should be straight or a very slight arc.

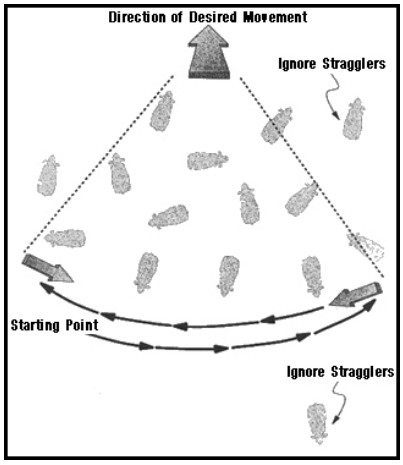

Figure A.1: Gathering Cattle

The handler can induce the rear animals to begin to move by giving them a "predatory" stare. This simulates the initial stalking behaviour of a predator sizing up the herd.

The handler should keep continuously moving back and forth. If you stop moving and linger too long in one animals' blind spot it may turn back and look at you. On open pastures, it is important to take your time. Six to twenty wide back and forth movements of 100 meters or more may be required to move the herd into a loose bunch. The handler should continuously walk back and forth and move enough to the side that the lead animals can see them.

Cattle that are off to one side of the pasture will be attracted as the herd moves into a loose bunch. Animals hidden in the brush or timber will be drawn out because they seek the safety of the herd. Do not chase stragglers.

2. Initiating Movement:

When the majority of the herd has come together into a loose bunch, increase pressure on the collective flight zone to initiate movement in the desired direction.

The handler continues the back and forth movements but presses closer to the herd to induce movement. This will cause the herd to move forward and begin to string out.

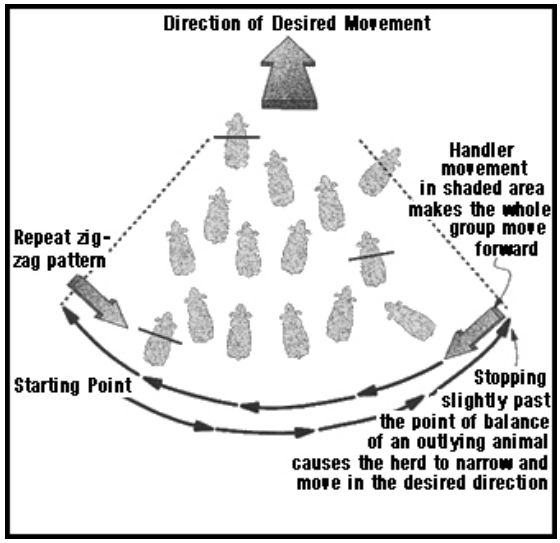

Figure A.2: Initiating Movement

Handlers need to differentiate between "good" and 'bad" movement of the cattle. When cattle have "good" movement, they can easily be driven in the desired direction. When animals have good movement that are all headed in the same direction and moving smoothly, they will look like a group of animals walking to water or making some other voluntary group movement on a large pasture. In a large group of animals, "good" movement starts with one animal and additional animals will gradually follow. "Good" movement entices the other animals to follow, and bad movements prevent other animals from following in an orderly manner. There are two types of "bad" movement;

1. Running, cutting back, and other panic induced movements
2. Animals stop moving as an orderly stream in the desired direction.

The first signs of bad movement are stopping, wavering towards motion or starting to turn away from the desired direction to look at the handler. The extreme form of type two movement is circular movement.

Good movement can be disrupted when the animals are attempting to locate the handler's position. This is a natural anti-predator behaviour of prey species. They want to know where the predator is and what its intentions are. Animals will turn and look at a person or a dog that is either in their blind spot behind their rear or is out side their flight zone. Handlers should not remain more than momentarily in any individual animal's blind spot. Walking through the blind spot will not cause a problem.

To make the group move pressure has to be applied to both the collective flight zone and individual animals within the moving herd. When an animal or a group responds to the handler's pressure on the flight zone, the handlers must immediately stop forward movement or change direction of movement to relieve pressure. This rewards the animal for moving in the desired direction and the animal is more likely to continue that movement. When the desired movement slows down, the handler must apply pressure again.

Every time you are working your animals you are training them. You can train them to be easy to handle and have good movement or you can train them to be difficult and have bad movement.

3. *Controlling Movement Direction:*

Animals must all be walking in the same direction before any attempt is made to change the direction of movement. When good movement is initiated, the handler can control the direction of movement by moving to the left to make the cattle turn right and visa versa.

To keep the animals moving in an orderly manner the handler alternates between penetrating the collective flight zone and withdrawing from the collective flight zone. Alternating pressure on the flight zone is more effective than continuous pressure. When the handler moves in the zigzag pattern they penetrate the flight zone when walking in the opposite direction of desired movement and retreat from the flight zone when walking in the same direction of desired movement.

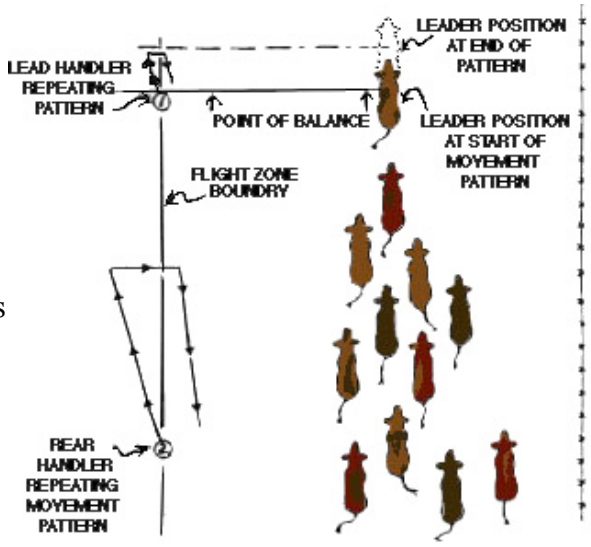

Figure A.3: Controlling Movement

Using the principles of flight zone behaviour, a handler is able to move cattle into a pen in a calm and orderly way. Using the positions shown on this diagram will enable the handler to control the flow of cattle through the gate. Cattle movement can be slowed or speeded up by moving forward or backward.

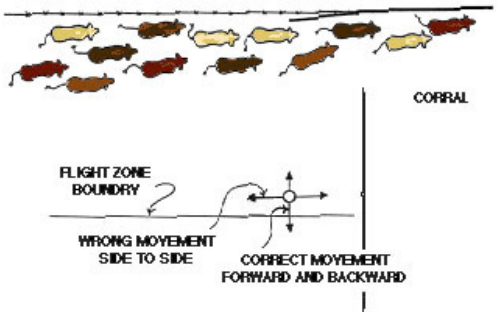

Figure A.4: Driving Through A Gate