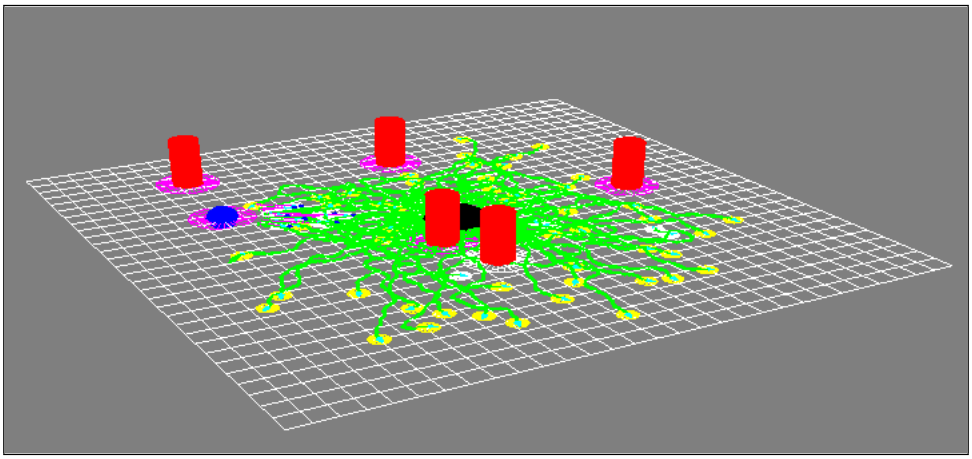
Swarm Intelligence
simulating ant foraging behaviour



Han Wang
MSc CAVE 09-10
Master thesis

August 20, 2010

Contents

Chapter 1

Introduction

In games and visual effects, it is needed to animate dynamically changing group behaviour. However, it is tedious and time-consuming to manage the motion path or behaviour of each individual manually due to the large quantity. So using swarm intelligence to simulate group behaviour controlled by simple rules would be an effective way to solve the problem.

This thesis illustrates the procedure of researching and implementing swarm intelligence based on pheromone which simulates ant foraging behaviour.

The first section is demonstrating previous works of emergent behaviour. Swarm intelligence, flocking system and other related research of the subject is introduced. Software and works that use this technique is also discussed in the section.

The second section is the technical background. Details of the algorithm is further explained, such as how flocking system works, and algorithm that use pheromone in the research paper.

The third section is design of the system, details of implementation, problems encountered and solutions to the problem.

The forth section shows the result of the project. Different result with different parameter specified of the system is discussed in this section.

The Fifth section states the conclusion, and discuss further work that can be put into consideration to improve the design, performance and usability.

Chapter 2

Previous Work

Emergent behaviour is a characteristic that relatively simple rules emergent into complex and uncertain result. There are different types of research on the subject. Some works focused on simulating the emergent behaviours of individual, such as Karl Sims' evolved virtual creature. Some works focused on simulating the collective behaviour of a group, such as Reynold's flocking system.

2.1 Swarm intelligence and flocking system

Swarm intelligence is inspired by emergent behaviour in nature, such as bird flock, ant colony, fish school. It is first introduced by Beni and Wang in "Swarm Intelligence in Cellular Robotic System". In the paper, swarm intelligence is described as "systems of non-intelligent robots exhibiting collectively intelligent behaviour evident in the ability to unpredictably produce 'specific' ([i.e.] not in a statistical sense) ordered patterns of matter in the external environment" Beni and Wang (1989).

Craig Reynolds' flocking system is one of the influential swarm intelligence example. Individuals in flocking system are called boids. They have three layers of motion, action selection, steering and locomotion. The three simple steering behaviours–aggregation,separation,alignment–can emergent into complicated result that looks like a flock of bird or a school of fish. In his paper, other steering behaviours such as obstacle avoidance, seeking, foraging are also added to the system to enhance the over all realism. Reynolds (1999)

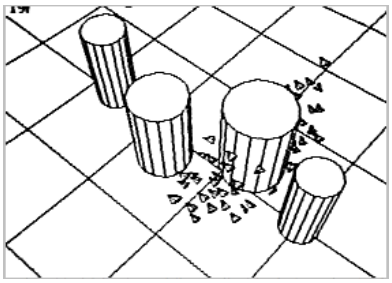


Figure 2.1: Craig Reynold's flocking system Reynolds (1999)

Another well known example of emergent behaviour would be Conway's Game of Life. It is a cellular automation with initial state specified and rules are set to decide whether the life live or die. With different initial state given, different pattern generate after certain time with the three rules. Figure 2.2 is a Gospers Glider Gun creating gliders. Conway's idea is a simplified implementation of John Von Neumann's attempt to build a machine that can build itself, which is also an analogy of the rise, fall and alterations of a society of living organisms Wikipedia (2010).

Figure 2.2: Gosper's Glider Gun creating "gliders" in Conway's Game of Life Wikipedia (2010)

Panait and Luke introduced a pheromone based utility to perform a relative effective method to implement foraging behaviour of ants. It is inspired by ant colony who is famous for finding the shortest route between nest and food with obstacles in between. An ant is an automata with internal state which perform actions and deposit pheromone based on rewards, value of pheromone in immediate neighbourhood Panait and Luke (2004). There is also an algorithm called ant colony optimisation which use the same theory and has been applied to path finding.

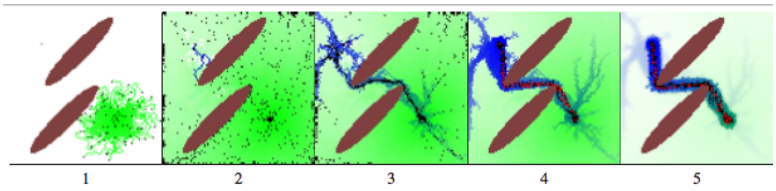


Figure 2.3: Ants, trail and optimisation of the path using pheromone Panait and Luke (2004)

2.2 Evolving creature and artificial ecosystem

In 1994, Karl Sims did a research project on simulating Darwinian evolution of virtual block creatures. Creatures are generated and evaluated by fitness function. The successful ones remain, and then they are combined, mutated to make offspring. This process continued and interesting and complicated behaviour emerge. Figure 2.4 shows evolved creatures competing with each other. Sims (1994)

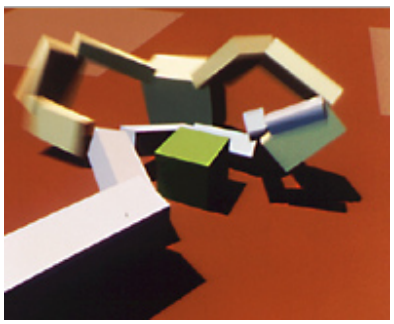


Figure 2.4: Creatures evolve to competition Sims (1994)

Polyworld is a simulator of ecosystem on a flat surface, which combines genetic algorithms and neural networks. Organisms and freely growing "food" are distributed in the world. Size, strength and other attributes of neuron are coded as organisms genome, while they have vision as an input and different behaviours as an output for the neuron. With limited resources in such a simulation, some species are able to sustain numbers through mating behaviour. Some species stay inside the

world by turning to avoid dropping outside of the world and die. Some just follow other organisms to get food. Behaviours like foraging, swarming also formed during the evolution Yaeger (1994).

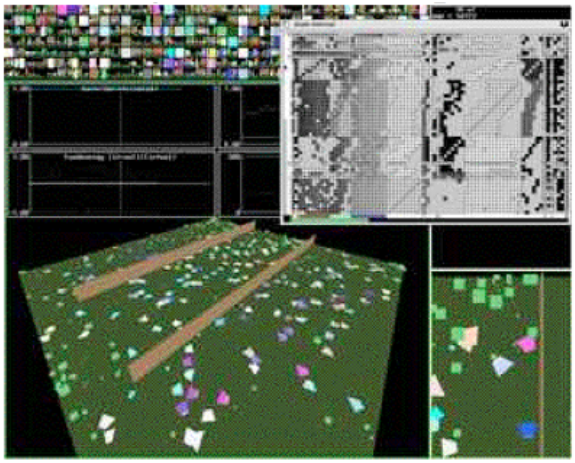


Figure 2.5: Polyworld. Small Squares at the top are vision of creatures in the world Yaeger (1994)

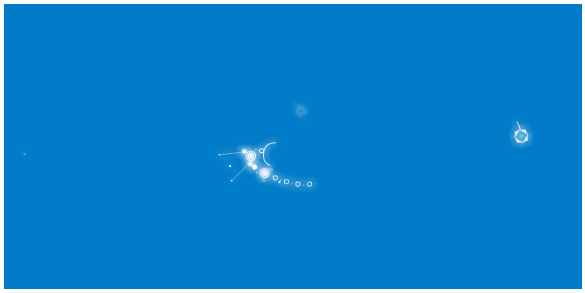


Figure 2.6: Life evolved with the interaction with the player

Different from evolution mentioned before, in flash game "Flow", you can have interaction with the evolving creature, and player's choices will affect how the creature will evolve and what they evolve into. Same mechanism is also used in game "Spore".

2.3 Massive

Massive is a commercial software package for visual effects which is initially made for "Lord of the Ring" to simulate battles that involve thousands of people. Now it is widely used in movie industry to effectively create crowd scenes. In pixar's "Ratatouille", Massive is integrated into the pipeline to produce the scene of large number of rats. In Massive, each agent is an AI with brain which can react individually to different situation. Fuzzy logic is also used to make the behaviour more human like. Massive (2010)

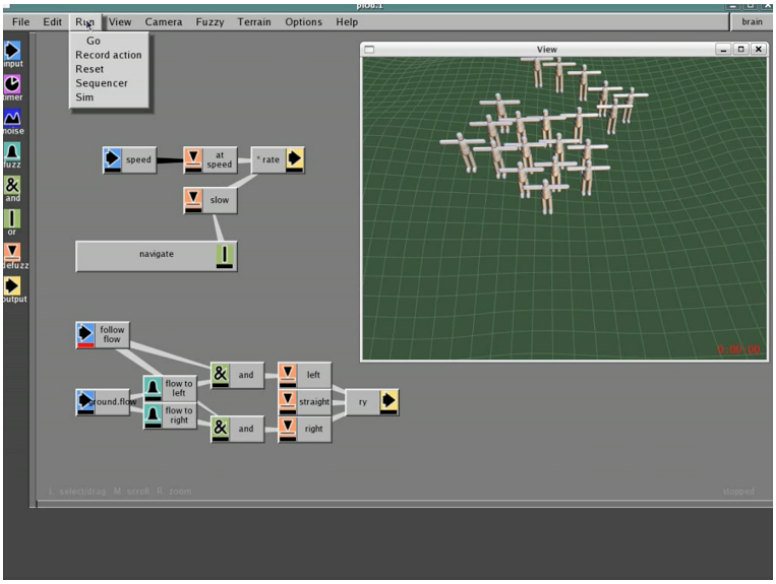


Figure 2.7: Fuzzy logic in Massive Massive (2010)

Chapter 3

Technical background

3.1 Swarm Intelligence

Swarm is a population with characters like interacting, decentralised, self organising. Usually members work collaboratively to achieve a certain goal by defining local behaviour of an individual. Population starts with randomness or chaos, but after the coordination between the members, intelligent like behaviour is formed as a result.

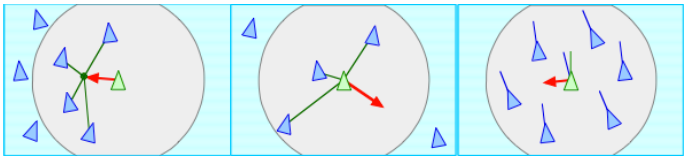
3.1.1 Flocking system

Flocking system is a type of swarm intelligence that members of the flock move according to the position and velocity of surrounding members. As mentioned in chapter 2, the autonomous character have three layers of motion: locomotion, steering, path planning. Locomotion is body representation of the model, it pass the information from steering behaviour to control the motion of the character, such as position, direction, rotation, Steering defines the behaviour that is going to affect locomotion of the model. Three central steering behaviour is cohesion, separation and alignment. Cohesion: make a boid move towards the average position of local flock mates. Separation: boid keeps a distance away from surrounding flock mates so that they won't crash onto each other. Alignment: keep the boid align with the direction of the nearby boids. Reynolds (1999)

Members of flock have access to all geometry information, but they only respond to flock mates that are near themselves. These flock mates are defined as neighbours within the area that is characterised with a radius and angle Reynolds (1999)

3.1.2 Ant colony optimization

Ant colony optimisation is a stochastic algorithm to find the shortest path between two places. Ants don't rely on their eye sight to find food , most of the communication is accomplished through laying pheromone. There are different types of pheromone, one of which is trail pheromone that marks up the path from one position to another, another is territorial pheromone to draw a line



(a) Cohesion (b) Separation (c) Alignment

Figure 3.1: Steering behaviour in flocking system Reynolds (1999)

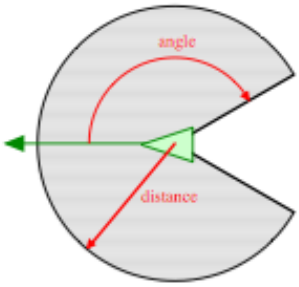


Figure 3.2: Definition of neighbourhood Reynolds (1999)

between different colony, and also alarm pheromone which is used to alert other members of the dangerous location. According to an ant's aim, they will walk towards to place with a higher density of target pheromone. Suppose when a group of ant is marching away from nest to search for food, there are obstacles that block their way. Randomly some of them choose the left path, some of them choose the right path as in Figure 3.3. Ideally all the ants move at the same speed. As in the figure, the left path is shorter than the right one, so in the same amount of time, there will be more ants pass through left path, thus accumulate more pheromone. Hence, when the rest of the ants came to the obstacle, they will choose the left path as in Figure 3.4. MONTES DE OCA (2010)

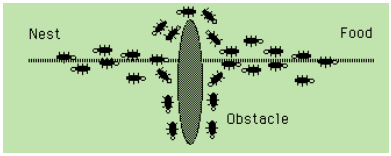


Figure 3.3: Initially, same amount of ants choose to walk on left and right MONTES DE OCA (2010)

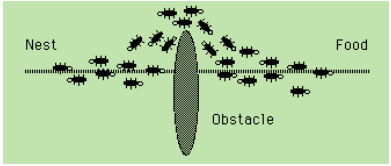


Figure 3.4: After sometime of accumulation of pheromone, ants move to the left side MONTES DE OCA (2010)

Chapter 4

Design and Implementation

The simulation is divided in to two part, C++ application and Rendering using Mel Script.

4.1 C++ Application

The application part is implemented using NCCA Graphics Library and C++. The aim is to prototype ant foraging behaviour on a flat surface. The world is consists of terrain, a colony of ants, trail created by ants, obstacles, food, nest where ants start from and carry their food back to. Position and movement of all objects is restricted on the terrain.

In order to intuitively demonstrate the simulation, different colour and models are used as shown in Figure 4.1. Terrain is a plane. The torus represents the nest. The blue sphere represents food source. The small blue spheres in front of ants represent the food taken from the source. Red cylinders represent obstacles ants can not pass through in the environment. Green trail is "to home" pheromone. Blue trail is "to food" pheromone. Circles around objects represent the bounding circle of objects, when they turn white, it means they are hit by other objects in the scene.

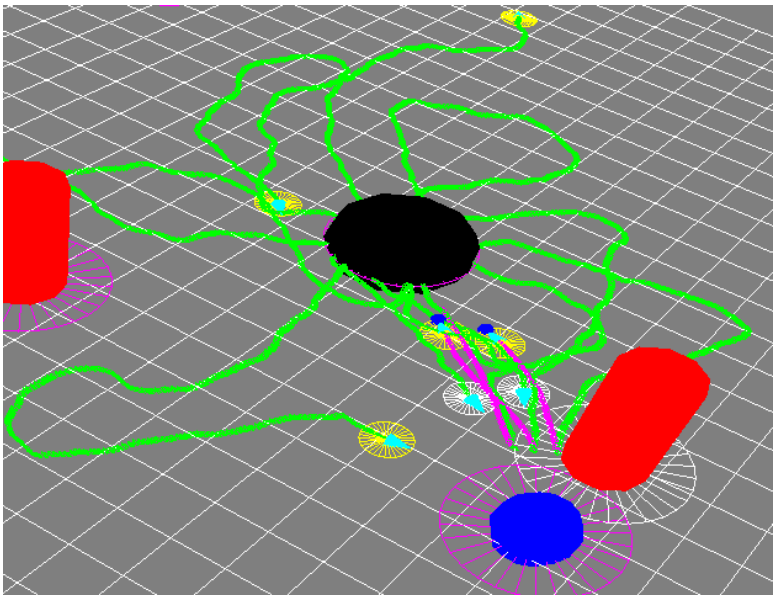


Figure 4.1: Ants search for food and carry it home

There are two kinds of pheromones: "to food " and "to home". Pheromone has weight to

indicate its lifespan, just like in real world that pheromone evaporate. The design follow the theory that ants always follow the direction which has a higher density of pheromone Panait and Luke (2004). Ants start randomly from the nest, looking for food. At the same time they deposit "to home" pheromone to mark up the way that they had used from home to food. They move randomly until they detect "to food" pheromone. Then they will move to the position with high density of the pheromone until they can detect the food source. After that they walk to the opposite direction, follow the density of the pheromone again until they detect nest. When they arrive at the nest, status of ants is reset and they start looking for food again. This procedure continued until all food resources no longer exist. Figure 4.2 show the state changes of ants.

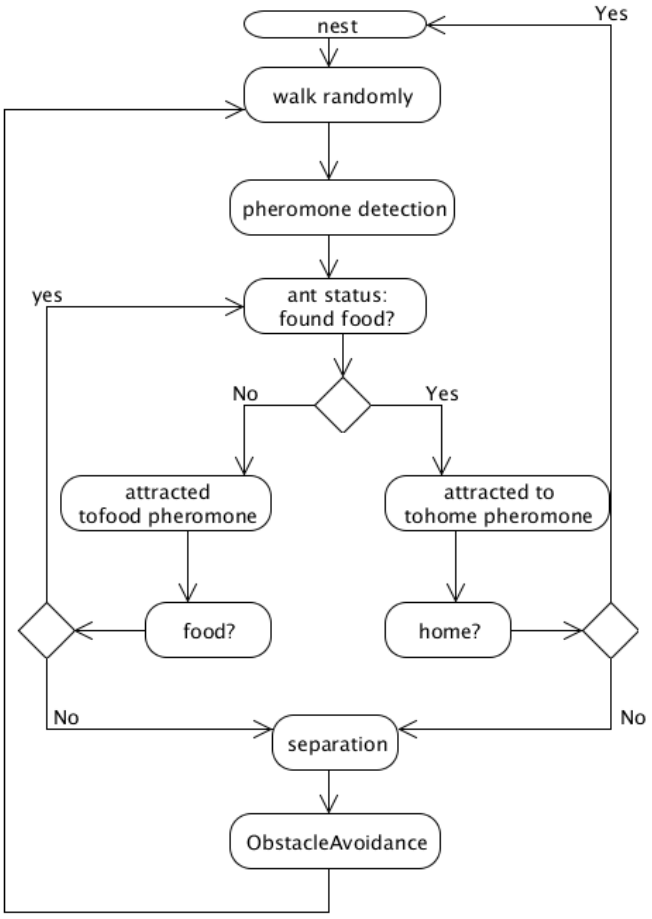


Figure 4.2: Flow chart

4.1.1 Class diagram

Class Ant specifies attributes of ants such as the status, locomotion such as moving direction, position and rotation, restrict ants within and on the surface, access height value from the texture, get vertex normal from ant position, define drawing method of ants.

Class Pheromone specifies attributes of pheromone, including type, position and also the drawing method of pheromone.

Class Trail maintains a list of pheromone, define the creation method, evaporation method and deleting method.

Class Colony manages ants in the colony and the trail created by ants, defines the steering behaviour of Ants, including obstacle avoidance, following pheromone, food detection, separation, nest detection, and also the rules that ants follow. Mel script is also written to a file in some methods in this class.

Class Object specifies attributes of other objects that are not ant, define drawing method and access height value from texture. Class Food, Nest, Obstacle are all derived from this class.

Class World manages all the objects, including colony, obstacles, nest, food resources. Mel script creating non ant objects is written to file in the method in this class.

Class Scene initialise Graphics Library, creates VBOprimitives using Graphics Library as model for objects to use in their drawing method, sets up openGL including material, light, camera set-up, processes events, display and update objects in the world. Events processing will process the mouse and keyboard input.

Class Global is a singleton class that stores global variable, sets up parameter for the simulation, create text file for Mel script .

Main function create instances of the classes, use a while loop and timer to update frames until users exit.

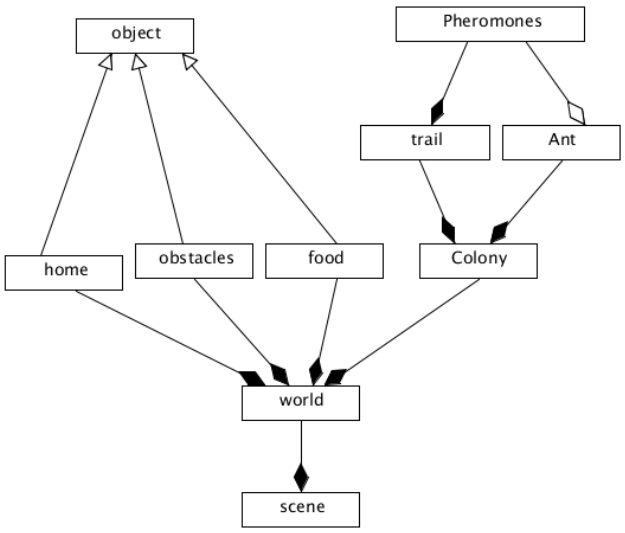


Figure 4.3: Class diagram

4.1.2 Three layers of behaviour

As in Craig Reynolds' "Steering behaviour of autonomous character", behaviours can be divided in to three layer. Reynolds (1999) The first one is locomotion. In the simulation, ants have moving direction, speed, position and Rotation. Position is modified by adding the product of speed and direction. Rotation is combination of spin and pitch. Second is steering behaviour. In the simulation, it includes obstacle avoidance, separation from local flock mates, following the pheromone, food detection, nest detection. Path planing and task: In the simulation, rules that decides the sequence of steering behaviours according to the status of ants is defined to make ants accomplish the foraging behaviour.

4.1.3 Stay in the world

Ants are requested to stay within the terrain surface. If the position assigned to ants after being modified by the direction is outside of the surface, the position is moved back to the edge of the surface, and a vector attraction which is perpendicular with the edge pointing to the centre of the surface is added to the direction to stop it from hitting the boundary. This also create an effect of wall following. This feature is implemented in Class Ant.

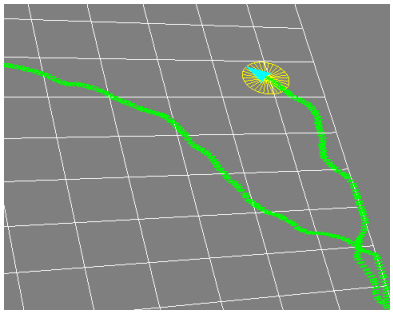


Figure 4.4: ant stay with in the surface

4.1.4 Random navigation and thinking

When an ant is not affected by surrounding objects or pheromone, it walks randomly. But generating a random direction creates an uninteresting look and produces non sustained movement Reynolds (1999) . So similar approach used for wandering in “Steering Behaviors For Autonomous Characters” Reynolds (2006)is adopted. A random offset is added to direction every frame to imitate ants behaviour of ”sniffing” around, meanwhile this maintains the sustained turning, According to observation, real life ants do not move at the same speed all the time. When they are navigating, they move some distance, and then stop to detect the direction it should move next. In order to achieve this effect, a walk counter is initialised at start. The counter is increased by one every time it moves, and function fmod() is used to make ant moves 47 steps and stop for 3 frame every 50 frames. Not all ants stop at the same time, so the counter is randomised at initialisation. This feature is implemented in Move method in Ant class

4.1.5 Obstacle avoidance

Each object in the scene has a bounding circle for detecting collision. If distance between ant and objects is less than the sum of the radius of two bounding circle, a vector that is opposite to the vector point from the obstacle to ants is used to modify the the direction. But this leads to the result that areas with more obstacles are not passable. When an ant detects an obstacle in the neighbourhood that may cause a potential collision such as right in front of it, the ant will try avoiding it. When obstacle is on the side, ant won’t collide with it so the ant wouldn’t be affected by this obstacle. As if a man is walking along a wall but won’t think he will collide with the wall and thus walk away. So, instead of just using a radius, an small angle is also used to predict the possible collision with the obstacle. Reynolds (1999) Same method is also used to prevent crowding with other ants in the simulation.

4.1.6 Neighbourhood

Perception of ants is limited, it should only respond to the objects which are within local area. Local area is defined by a radius and an angle with the moving direction. A search of all the objects in the scene is performed to decide which is neighbour. Only objects within the area is

considered to be neighbours. But this will result in the slow speed of the performances. Since this simulation is focused on the behaviours, speed is not put into consideration.

4.1.7 Pheromone and trail

Pheromone attraction used in this application is different from the rewarding mechanism in the Panait's pheromone model Panait and Luke (2004). Attraction to pheromone is designed similarly to the cohesion definition in flocking system. An average position of pheromone within the neighbourhood of ants is calculated and then ants direction is modified . Pheromone does not maintain the same density all the time, they evaporate. Weight is added as an attribute of pheromone. so weighted average position of pheromone within the sense area is calculated, an attraction from ants to the average position is used to modify the direction of ant. Thus, the ant shows a behaviour of moving towards position with the highest density of pheromone. However,the lifespan and evaporation rate of the pheromone will affect ants performance, different result is compared in the result section.

Managing the trail. A list called trail is maintained by the colony to store pheromone laid by ants. Pheromone's type is decided by ant's status. When an ant is looking for food, pheromone's type is "to home". When an ant has found food, pheromone's type is switched to "to food". Ants create pheromone every frame, the position of pheromone is the same with the ant. Information of what type of pheromone and position of pheromone is added to the trail. There is only one instance of trail which makes it easy to maintain, but difficult to tell which ant laid the pheromone. Every frame, pheromone evaporates, their lifespan decrease, and their age grow. When pheromone lifespan evaporate to 0, which means pheromone is "gone", it is deleted from the trail.

4.1.8 Food and nest

Ants find food and nest with the indication of pheromone. However, pheromone alone does not create desired behaviours. Trail of pheromone is an indication, but it is not the exactly the right way to go. When food source is in front of ants, ants can be confused by pheromone because pheromone density can be the same with in the sense radiation of ant near nest. So when an ant is close to food source, an attraction to the food is added to the ant as if the ant can see the food. When ants reach the food, food life span decreased, and ant is set to the opposite direction. Same with the nest, an attraction to nest is added to ants direction when ants are close to nest. When ants reach nest, they are reset.

4.1.9 Terrain

Terrain is generated by displacement in Maya using a grey-scale bitmap. The y value of position of the terrain is calculated in C++ application. Gray scale value of bitmap define the height of the terrain and is loaded in as a texture instance using Graphics Library. The position of the ant moving on xz plane is scaled from world size to 1*1 used as an index to lookup for the height information in the texture. and then value returned from the texture is scaled to the world size and assigned to y. This will cause errors on calculating the height because the function in graphic library require integer value to get the data which the application cannot provide. More accurate method is needed for getting the height information. Ratio of texture size should consist with macro that defined for height and width.

4.1.10 Rotation

Rotation is crucial part to make ants look realistic. Two rotations are applied to ants, one is rotation on xz plane around y axis called yaw, the other is rotation xy plane around z axis called pitch.

In order to get xz plane rotation, the angle between direction and ant model's direction is calculated by using the angle-between function from the Graphics Library. Whether the angle is

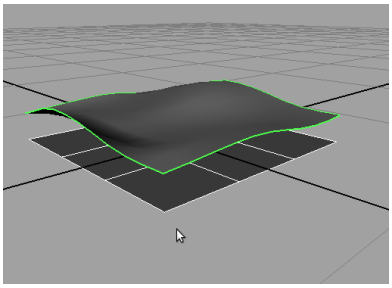


Figure 4.5: Flat surface and its displacement

Figure 4.6: Texture map

positive or negative is decided by value of z, if z is positive, rotate counter-clock, if z is negative, rotate clock wise.

In order to get xy plane rotation, terrain normal needs to be calculated. Because when ants move on the surface, moving direction is perpendicular to the terrain normal. A virtual vertex normal is calculated, Four points are defined around the current position of the ant, xz value of the points are defined by adding four different offset to xz value of ants position. Y value of position is got from looking up in the texture map. Four points and ant's position form four faces, thus the vertex normal is the average got from the normals of the four faces. When x value of moving direction is positive, angle between direction and normal is used to rotate the model, when x is negative, angle between opposite of direction and normal is used to rotate the model. Order of the rotation makes difference. Initial design of rotating xz plane first cause the failure of the xy plane rotate.

4.2 Visualisation in Maya

Colony and World class are in charge of writing Mel script to a text file, which is used to create the key-frame animation for ants, nest, food, obstacles. Terrain is generated using displacement, and then displacement is converted into polygon. The world is scaled to unit 1, so that the data is consistent between C++ application and Maya.

Initially, data of ants was written into a file and a script is written to load the data into Maya to create a motion curve for the animation. However, this method is inaccurate and ineffective, and extra script is needed to organise the object. So instead of writing data to file, a Mel script is written into a text file called "melscript.txt" when running the simulation.

The Mel script follow the order as bellow: Create objects–set current time–perform translation and rotation–key framing More details of the script can be examined in " melscript.txt".

One problem with the the first "current time" for Mel script is that the time is got at compile time. This is not what we want, so it is manually defined to be a small value which is close to 0. Attention should be paid that the models in c++ application should consist with the ones in Maya simulation, or they will cause unwanted effect like ants go through an obstacle.

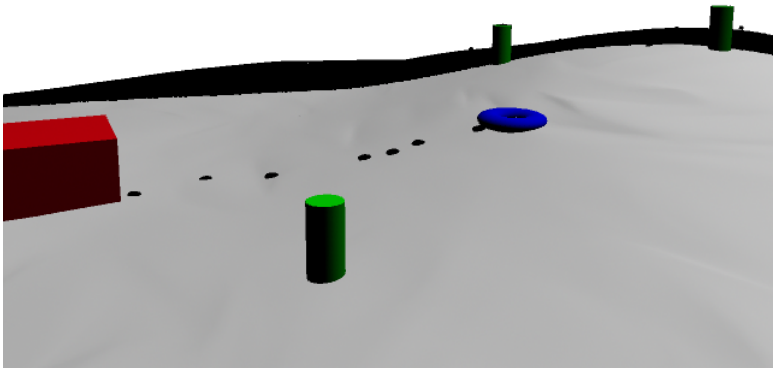


Figure 4.7: A rendered image in Maya

Chapter 5

Result and discussion

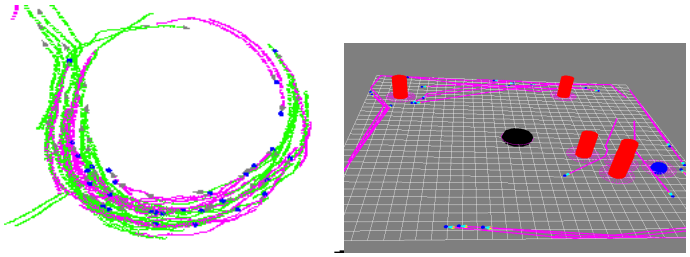
After some time, most ants are able to find the shortest path and keep transporting food between nest and food source, avoiding the obstacle. Simulation runs fluently until ants number reaches 50. Pheromone, obstacles and ants all need to be searched , so the longer pheromone last and the more objects there are, the slower the speed is.

5.1 Pheromone weight and perception

If a pheromone lasts too long, then the pheromones will become overlapped and the ant will get confused. If a pheromone evaporates too quickly, it will disappear before ants start to carry food home, so in order to get better result, evaporation life span should consist with the world size. When world is large enough and to-home pheromone evaporate too quickly, ants loose information, they will get confused, and chase the pheromone laid by other ants which are searching for food, This will form a circle, ant won't be able to get food, neither can they get home. This phenomenon is called ant mill and happens a lot in an ant species called army ant.

5.2 Errors

A well performed simulation needs tweaking the value of coefficient of all the steering attraction. For instance, when coefficient of obstacle avoidance is too small, errors like ant go though the obstacle will occur. If the attraction of pheromone is too large, ants do not go back to the nest and get lost in the world



(a) Ant mill (b) Ants get lost

Figure 5.1: Errors with different settings

5.3 Pheromone representation

In most cases, the definition of moving to the direction with a higher density of pheromone in the simulation works well on making ant choose the right direction, however, some errors occurred on their way searching. This might be due to the simplification of the mechanism of attraction of pheromone. So trail information still need to be improved and more research into actual ants behaviour and the reason behind it would help to improve the algorithm.

Chapter 6

Conclusion and further Work

6.1 Conclusion

With the ideas of two types of pheromone and references from flocking system, prototype of the swarm intelligence simulating ant foraging works on a displaced surface. The system can manage colony of ants using different pheromone to find effective way to transport food between food source and nest, output the Mel script into files which can be sourced in Maya to create animation. User can simulate in the C++ application until they are satisfied with the result and then use Maya to render the animation with more details and better quality. The behaviours of ants in the simulation is relatively realistic, however, this prototype still need further development to be used as tool.

6.2 Further work

As described in the implementation section, the points position got from GetHeight function probably won't be within the surface that created with Maya displacement. This will result in ants walking through the surface. There are two ways to solve the problem, generating terrain in the C++ application or improving quality of displacement.

When there is a larger displacement, ants velocity is increased, this will result in unnatural speed up in simulation. This could possibly resolved by generating terrain in C++application.

Pheromone is scent, it diffuse in the air. Diffusion should be put into consideration and colour representation of the evaporation would be help to research the effect of pheromone on ants.

Currently, there is no interface to interact with users. Most parameter need to be modified manually in the global class. A parser should be implemented to read in data from user interface. Obstacles and food is randomly generalised, however, allocation of obstacles by hand would be more user friendly and extend the usability of the simulation.

Optimisation using spatial hashing would be a good way to solve the slow speed problem when there are more ants and pheromone. Because users simulate in the application to see the effects first, a faster speed would help user made modification as soon as possible when they found problems with the simulation.

What ants can do now is simply follow the rule, 'sniff' for food and then carry them home until all food resource is carried to the nest. There are still many interesting behaviours can be added to the system. Ant does not only forage static food, living creatures are also attractive to their stomach. Different colonies have different smell, intruders will be recognised depositing different pheromone. So adding different colonies and implementing the way they conflict with each other will be an interesting extension to the behaviours.

More research on artificial intelligence technique such as state machine, fuzzy logic may contribute to the behaviour of ants. Besides, how genetic algorithm and neural network can affect the system would also be put into consideration.

The way code is organised will have much influence on the understanding and usability of the code. So a better design of the classes should be put into future work..

Bibliography

Beni, G. and Wang, J. (1989). Swarm intelligence in cellular robotic systems, Proceedings of NATO Advanced Workshop on Robots and Biological Systems Vol 102.

Massive, S. (2010). What is massive, `http://www.massivesoftware.com/`.

MONTES DE OCA, M. (2010). Behavior of real ants, `http://iridia.ulb.ac.be/~mdorigo/ACO/RealAnts.html`.

Panait, L. and Luke, J. (2004). A pheromone-based utility model for collaborative foraging, Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1.

Reynolds, C. (1999). Steering behaviors for autonomous characters, Proceedings of Game Developers Conference 1999 held in San Jose2, California.

Reynolds, C. (2006). Big fast crowds on ps3, Proceedings of the 2006 Sandbox Symposium.

Sims, K. (1994). Evolving virtual creatures, Computer Graphics, Annual Conference Series (SIGGRAPH 94 Proceedings).

Wikipedia (2010). Conway's game of life, `http://en.wikipedia.org/wiki/Conway's_Game_of_Life`.

Yaeger, L. S. (1994). Computational genetics, physiology, metabolism, neural systems, learning, vision, and behavior or polyworld: Life in a new context, Langton, C. ed. Proceedings of the Artificial Life III Conference. Addison-Wesley.