Chapter 8
Embodiment, Stigmergy, and Swarm Intelligence

8.0 CHAPTER OVERVIEW

The LEGO robots in previous chapters have exhibited various degrees of situatedness—from the thoughtless walkers, which (charitably) can “sense” elementary forces like gravity, to Braitenberg’s Vehicle 2, which has sensors for measuring light, and ending with the LEGO Tortoise, which can sense both light and obstacles. While these robots map out a continuum of situatedness, at first glance it would seem that they are all equally embodied, because they are all constructed out of the same types of building blocks.

However, some would argue that embodiment means more than just being physically constructed; it has been claimed the degree of an agent’s embodiment reflects the extent to which the agent can alter or manipulate its environment (Fong et al., 2003). From this perspective, a continuum of embodiment is also possible. However, the preceding robots do not map out this continuum particularly well, because they all react to—and fail to manipulate—the world in which they operate. The current chapter describes a new robot, the Lemming, which is designed not only to sense its environment, but also to change it. Like the Tortoise, the LEGO Lemming can sense and avoid obstacles. The Lemming also uses a light sensor. However, the function of this sensor assumes that the Lemming operates in a world in which coloured objects have been scattered on the floor. When an object is encountered by the robot, its colour is detected, and this controls the robot’s behaviour. In particular, the sensed colour determines whether the object will be moved and deposited near a wall at
the outskirts of the Lemming’s domain, or will be placed near other bricks at the interior of the Lemming’s world. This creates a higher degree of embodiment than was exhibited by any of the earlier LEGO robots. This is important, because the notion of cognitive scaffolding that was introduced in Chapter 3 requires that agents be able to manipulate their world. The “mind” of the Lemming has leaked into its world, because the coloured objects that it moves can be described as an external memory.

This chapter explores the Lemming’s “leaky mind” in two contexts. The first involves a single Lemming that manipulates its external memory of coloured objects. The second involves a small colony of Lemmings that manipulate the collective memory of the colony. This leads us to consider in more detail both the notion of embodiment and the notion of stigmergy (which was introduced in Chapter 1), and to explore some of the ideas that are fundamental to collective intelligence.

8.1 TRAVELLING SALESMEN

8.1.1 The Travelling Salesman Problem

The travelling salesman problem, or TSP, is one of the most famous and important problems in the combinatorial optimization literature (Gutin & Punnen, 2002; Lawler, 1985). The problem itself is easy to express: Imagine a salesman who must visit a sequence of cities, stopping at each only once. In what order should the salesman visit the cities, so as to travel the shortest (and presumably least expensive) route?

The TSP has been studied for a very long time. While it was first named by Menger in 1932, its form was first defined by Voight in 1831 (Laporte & Osman, 1995). The extent of modern research on the problem is indicated by the existence of a bibliography of 500 references relevant to it (Laporte & Osman, 1995).

One reason for the long history of research on the TSP is because of its importance; the TSP is applicable to a wide variety of real-world problems (Punnen, 2002). These include scheduling tasks on a machine to minimize the cost of setting the machine up for each new job, and assigning a different frequency to each of a network of transmitters so that interference between transmitters is minimized. Punnen also notes that other areas to which the travelling salesman formulation is relevant include data analysis in psychology, X-ray crystallography, overhauling gas turbine engines, warehouse order-picking problems, and wallpaper cutting.

A second reason for the long history of research on the TSP is its
difficulty. The TSP is a famous example of an NP-complete problem (Kirkpatrick, Gelatt, & Vecchi, 1983). This means that as the number of cities involved in the salesman’s tour increases linearly, the computational effort for finding the shortest route increases exponentially. For \( N \) cities, the number of possible routes to consider when doing an exhaustive search for the shortest route is \( \frac{1}{2} (N-1)! \). This means that for a 4-city tour, one needs only consider 3 different routes to find the shortest. However, for an 8-city tour, the shortest route is but one of 2,520 possibilities; there are approximately \( 4.421 \times 10^8 \) routes to compare to find the shortest tour of 30 different cities!

### 8.1.2 Solving the TSP

Given the importance and the difficulty of the TSP, a number of different approaches to its solution have been explored. Many of these approaches are algorithms that have a long history in the numerical optimization literature (Bellmore & Nemhauser, 1968).

Some more recent solutions to such problems have been inspired by physical metaphors. Annealing is a physical process, describable using statistical mechanics, by which an optimal structure is obtained by bringing a substance to a high temperature, and then slowly cooling it. Optimality is discovered because at high temperatures the state of the substance (e.g., arrangements of atoms) can be moved out of local minima with high probability; the slow cooling can result in the state achieving its most stable configuration (i.e., a global minimum). Simulated annealing, where the state being optimized is the cost of the tour, has been successfully used to provide excellent solutions to the TSP (Kirkpatrick et al., 1983).

Other approaches to the TSP are biologically inspired. Neural networks have been used to discover TSP solutions (Hopfield & Tank, 1985; Siqueira, Steiner, & Scheer, 2007). Evolutionary programming techniques, such as genetic algorithms (Holland, 1992; Mitchell, 1996), have also been successfully applied (Braun, 1991; Fogel, 1988). Even molecular computers, which encode problem states using DNA molecules, have been explored (Lee, Shin, Park, & Zhang, 2004).

Approaches to the TSP have also been inspired by observing how insects deal with real-world situations (Tarasewich & McMullen, 2002). These approaches are of interest to us because they raise the possibility of using teams of simple robots to solve problems that might be beyond the capability of any individual member of the team. Let us now consider these solutions and their implications.
8.2 SWARM INTELLIGENCE

8.2.1 Economical Ants

One reason for the travelling salesman problem’s importance is that being able to find the shortest route provides enormous advantages for a wide variety of human endeavors. However, the importance of this ability is not restricted to humankind. Any animal that must move regularly between two different locations, such as a nest and a food source, would benefit by identifying and using the most economical route (Goss, Aron, Deneubourg, & Pasteels, 1989). Is there any evidence that they do so?

For one example, consider the Argentine ant *Iridomyrmex humilis*. Goss et al. (1989) studied a laboratory colony of these ants by using a series of bridges that linked their nest to a food supply. In this bridge system there were two locations at which the ants had to choose between two different routes. At each decision point, one choice would lead to a route that was much longer than the one that would be followed if the other choice had been made. When the bridge system was first put in place, food was discovered in a matter of minutes. At this early stage, ants went in each direction at both decision points with equal probability. However, shortly afterward, a strong preference emerged: almost all of the ants chose the path that produced the shortest journey.

How do ants determine the shortest route between two locations? The answer to this question is rooted in local, computationally simple, ant behaviour. *Iridomyrmex humilis* leaves a trail of pheromones as it moves in either direction along a path between food and its nest. An ant that chooses the shortest path will return along it, and add to the pheromone trail at the decision points, sooner than an ant that has taken a longer route. This means that ants that arrive later at a decision point will find a stronger pheromone trail in the shorter direction, will be more likely to choose this direction, and will themselves add to the pheromone signal. “Each ant that passes the choice point modifies the following ant’s probability of choosing left or right by adding to the pheromone on the chosen path. This positive feedback system, after initial fluctuation, rapidly leads to one branch being ‘selected’” (Goss et al., 1989, p. 581).

The ability of ants to find shortest routes inspired a new approach to solving the travelling salesman problem (Dorigo & Gambardella, 1997). Dorigo and Gambardella programmed a colony of simulated ants to leave and follow pheromone trails, which also had a working memory that stored cities that had already been visited, so that the artificial ants would travel to a new city. They studied a number of different versions of the problem, and found that the simulated ant colony produced
solutions that were as least as good as, and often better than, solutions produced by a variety of other algorithms, including neural networks and genetic algorithms.

### 8.2.2 Emergent Intelligence

The ability of ants — simulated or otherwise — to choose shortest routes does not, importantly, require a great deal of computational power within each individual. Individual ants do not determine optimal routes; it is the ant colony as a whole that solves the problem. “The selection of the shortest branch is not the result of individual ants comparing the different lengths of each branch, but is instead a collective and self-organizing process, resulting from the interactions between the ants marking in both directions” (Goss et al., 1989, p. 581).

### 8.3 Collective Contributions

#### 8.3.1 Swarm Advantages

In Section 1.7.2, we saw that an organism could be defined as a coordinated system of activities that could obtain environmental resources, produce new activities, and adapt to environmental disturbances (Wheeler, 1911). This permitted entomologists like Wheeler to define the colonies of social insects as superorganisms, from which emerged more complex results (such as elaborate nests) than one would predict from examining the capabilities of individual colony members. Swarm intelligence is an interesting evolution of the idea of the superorganism. It offers advantages that may not be provided by other computational methods. “Nature-inspired intelligent swarm technology deals with complex problems that might be impossible to solve using traditional technologies and approaches” (Hinchey, Sterritt & Rouf, 2007, p. 113). What is provided by swarm intelligence that might be missing from traditional approaches?

Importantly, a swarm’s components are only involved in local interactions with each other. This characteristic is the source of many of the advantages of swarm intelligence (Balch & Parker, 2002; Sharkey, 2006). For instance, a computing swarm is scalable — it can be comprised of varying numbers of agents, because the same control structure (i.e., local interactions) is used regardless of how many agents are in the swarm. For the same reason, a computing swarm is flexible — agents can be added or removed from the swarm without reorganizing the entire system. The scalability and flexibility of a swarm make it robust — it can continue to compute when some of its component agents no longer function properly. A second source of robustness comes from the nature of the swarm’s agents themselves. For instance, if each agent
is autonomous, and is capable of reacting or adapting to environmental changes, then these individual advantages will be inherited by the swarm as a whole.

8.3.2 Robot Collectives

When a swarm is composed of autonomous, embodied, and situated robots, it may be particularly well suited to solving some important real-world problems (Beni, 2005; Brooks & Flynn, 1989). One reason for this is that a robot collective would have all of the advantages of swarm intelligence that were mentioned in Section 8.3.1. A second reason for this is that robot collectives are capable of manipulating real-world objects and environments, and therefore can serve as real-world tools.

For example, NASA is interested in preparing landing sites on distant planets. A swarm of robots provides one possible solution to this problem (Parker, Zhang, & Kube, 2003). Parker et al. were inspired by a behavior in some ants, called “blind bulldozing” (Franks, Wilby, Silverman, & Tofts, 1992), in which nests are constructed stigmergically by pushing material away from a nest site. Parker et al. designed a robot collective for blind bulldozing. An individual robot in the collective is usually in a plowing state, in which it moves straight in some heading, pushing debris as it moves. When the friction caused by an accumulation of debris exceeds a threshold, the robot switches into a finishing state, which causes it to turn a random amount before re-entering the plowing state. The robot could also switch into a colliding state when it randomly turns because of a collision with another robot in the collective. Parker et al. created variously sized robot collectives that created “nests” by pushing away gravel while following this algorithm. They found that a nest could be constructed by a single robot, but that the use of multiple robots decreased the time that was required to accomplish the task.

Robot collectives are not appropriate for all tasks, but are ideally suited for many (Balch, 2002). As we shall see, typical tasks for robot collectives include foraging, material transport, and sorting. It has also been argued that a collection of simple, mass-produced robots that do not require central control provide an ideal and inexpensive medium for conducting exploration of remote planets (Brooks & Flynn, 1989).

8.4 CRITICAL NUMBERS OF AGENTS

8.4.1 When Is a Swarm Intelligent?

In swarm intelligence, a problem’s solution emerges from the activity of a collection of agents, suggesting that having a collection of agents is better than having a single agent working on the problem. However,
swarm intelligence depends on more than mere numbers of agents. For a swarm to be considered intelligent, the whole must be greater than the sum of its parts. This idea has been used to identify the presence of swarm intelligence by relating the amount of work done by a collective to the number of agents in the collection (Beni & Wang, 1991).

Consider, for example, a collection of completely independent agents foraging for food. As the number of agents increased, one would expect that the collective would forage faster. However, if the agents worked completely independently of one another—if the whole were equal to the sum of its parts—then there would be a linear relationship between the amount of work accomplished and the number of agents, as shown in the dashed line in Figure 8-1. Beni and Wang (1991) would take this linear relationship to indicate the absence of swarm intelligence.

In contrast, consider agents that can interact with each other. A small number of agents may not have much opportunity for interaction, and therefore may not perform better than the same number of independent agents. However, after some critical number of agents is reached, agent interaction becomes more likely, and makes the swarm more efficient...
than a non-interacting collective. This is shown in the solid non-linear function in Figure 8-1. With 2 or 3 agents, there is little difference between the solid and dashed functions. However, when there are more than 3 agents, the interactive collective is far more efficient than the non-interactive one. The non-linear relationship between the number of agents and the amount of work accomplished is taken by Beni and Wang (1991) to indicate the presence of swarm intelligence.

8.4.2 A Foraging Example

One study of robot foraging tested Beni and Wang's (1991) theory (Sugawara & Sano, 1997). In this study, obstacle-avoiding robots moved through an arena, collecting pucks, which they then brought back to a home location. In some conditions in this experiment the robots interacted: when a puck was encountered, the robot stopped for a set period of time and emitted a light that attracted other robots to that location.

Sugawara and Sano (1997) found a non-linear relationship between the number of robots and the number of pucks foraged over time, but only when robots interacted, supporting the theory of Beni and Wang (1991). When robots did not interact, only a linear relationship between these two variables was observed. Interestingly, though, robot interactions were not always helpful—they only improved efficiency if the pucks were unevenly distributed throughout the environment. Under such conditions, a robot emitting light would attract other robots to a high concentration of pucks. However, if the pucks were evenly distributed throughout the arena, then interactions actually caused a decrease in efficiency relative to a swarm of non-interacting robots. Thus, direct communication in a swarm does not always lead to improved performance. In the next section, we will see that an important issue in swarm intelligence is the degree to which communication is used to coordinate the activities of swarm members.

8.5 Coordination, Communication, and Cost

8.5.1 Costly Coordination

Early research on robot teams studied small groups of homogenous robots (Gerkey & Mataric, 2004). Modern research examines much more sophisticated robot collectives that can consist of different types of robots that carry out diverse tasks at varying locations or times (Balch & Parker, 2002; Schultz & Parker, 2002). “It is no longer sufficient to show, for example, a pair of robots observing targets or a large group of robots flocking as examples of coordinated robot behavior. Today we reasonably expect to see increasingly larger robot teams engaged in
concurrent and diverse tasks over extended periods of time” (Gerkey & Mataric, 2004, p. 939).

It is desirable to coordinate the actions of a team of diverse robots (Gerkey & Mataric, 2002, 2004; Mataric, 1998). One must determine the tasks carried out by individual robots in a team at any given time, in order to optimally achieve some global goal. This called the multi-robot task allocation problem.

With the hardware capabilities of modern robots, one general approach to solving the multi-robot task allocation problem is to employ intentional co-operation (Balch & Parker, 2002; Parker, 1998, 2001). Intentional co-operation is achieved by adopting some form of communication between robots so that task allocation can be negotiated, or so that one robot will be aware of what others are doing so that it does not unnecessarily duplicate efforts or work in opposition to the current efforts of other team members.

Intentional co-operation provides some particular advantages. If it is possible to have intentional co-operation amongst robots, then it should also be possible have a robot team co-operate with the needs of a human user (Gerkey & Mataric, 2004). As well, a robot team governed by intentional co-operation should be more efficient than one that is not (Sugawara & Sano, 1997). “By sharing information and leveraging each others’ skills, a group of robots can truly be more than the sum of its parts” Gerkey & Mataric, 2002, p. 758).

A robot team governed by intentional co-operation may be extremely efficient at performing a task, and may also be able to use communication to structure robot activities so that a single collective can efficiently solve a diversity of problems. However, these advantages are not achieved without cost. First, the extent to which intentional co-operation imposes central control on the members of the robot team is the extent to which many of the advantages of robot collectives described in Section 8.3 are diminished. For instance, communication between robots is costly, and as more robots are added to a communicating team there is likely to be a “communications bottleneck” that makes the team less scalable (Kube & Zhang, 1994). Second, as communication makes the functions carried out by individual team members more specialized, the robustness of the robot collective might be jeopardized (Kube & Bonabeau, 2000).

In short, while the study of intentional co-operation for dealing with multi-robot task allocation is ongoing and important, other approaches are still worthy of consideration. In particular, is it possible for a robot collective to coordinate its component activities, and solve interesting problems, in the absence of direction communication?
8.5.2 A Stigmergic Solution

One answer to this question can be provided by studying the extent to which stigmergy can be used to control a team of robots (Kube & Bonabeau, 2000). As we saw in Chapters 1 and 3, stigmergy occurs when there is indirect communication between agents via the environment. An agent makes some change to the environment, which in turn signals another agent to perform a different behaviour than it would have in the absence of this signal. We have seen that stigmergy provides important control over the behaviours of social insects; these insect behaviours have inspired many studies of robot teams, which have shown that such collectives are capable of solving interesting problems without the need of intentional co-operation. Let us now consider some prototypical examples of such research.

8.6 CO-OPERATIVE TRANSPORT

8.6.1 Robots that Push Boxes

Consider the New World army ant *Eciton burchelli* (Couzin & Franks, 2003). A foraging party of 200,000 ants from a colony will kill as many as 30,000 prey in a dawn-to-dusk swarming raid, and move this food back to the nest, often using co-operative transport (Franks, Sendova-Franks, & Anderson, 2001). Co-operative transport involves a group of agents working together to move a large object.

Roboticists have studied co-operative transport using the box-pushing task, in which a box, intentionally too heavy for a single robot to move, must be pushed to a goal location. Imagine a group of five small robots sitting in the corner of a laboratory room. In the middle of the room is a large box (Kube & Bonabeau, 2000). A spotlight hanging from the ceiling of the room is turned on, and the robots begin to move throughout the room. Then, a light in the middle of the box is turned on. Suddenly, the behaviours of the robots change, as shown in this video available from http://www.cs.ualberta.ca/~kube/ra97/multi-robot.mpeg.

When the box is lit, the robots move toward it and come into contact with it. Some of the robots remain in contact with the box, and push it toward the part of the room lit by the spotlight. Other robots are in the wrong place to accomplish this, and move around the box to take up a more effective position. Still other “robots leave the task, seemingly at random, and wander off only to return and join the group effort in transporting the box towards its goal” (Kube & Bonabeau, 2000, p. 99).

The robots move the box toward the goal, but not smoothly: the
box may veer off in an incorrect direction, or sometimes stop moving. However, at each juncture the robots realign themselves, and eventually push the box to the intended goal. At this point, if the spotlight is turned off, and if a different spotlight is turned on, then the robots will reorganize themselves, and again—in a moderately erratic fashion—the box will be pushed toward the new goal location.

### 8.6.2 Stigmergic Co-operation

The robots’ box-pushing behaviour mimics some of the co-operative transport behaviours observed in ants, and therefore might serve as a model of this insect ability. However, this co-operative behaviour emerges without direct communication between robots. “Rather a form of indirect communication (stigmergy) takes place through the environment by way of the object being manipulated” (Kube & Bonabeau, 2000, p. 100). This is important because other approaches to solving the box-pushing problem usually involve direct communications between robots, so (for example) one robot “knows” not to duplicate the efforts of another (Mataric, 1998; Parker, 1998, 2001).

Kube and Bonabeau (2000) achieved stigmergic control of box pushing by providing robots with behaviours that were elicited by simple stimuli. Robots used both touch and infrared sensors to detect (and avoid) other robots. Light sensors were used to locate the box as well as the goal spotlight. If a touch sensor was depressed, and the box-detecting light sensor was above threshold, then the robot had detected that it was in contact with the box. If it was in such contact, and could see the goal, then box-pushing behaviour was initiated. If it was in contact with the box, but could not see the goal, then other movements were triggered resulting in the robot finding contact with the box at a different position.

These behaviours caused the robots to seek the box, push it toward the goal, and do so co-operatively by avoiding other robots. Furthermore, as the robots acted on the environment, and changed the position of the box, this could change the situation sensed by other robots, and produce corresponding changes in behaviour. For instance, a robot pushing the box might lose sight of the goal because of box movement, and would therefore leave the box and use its other exploratory behaviours to come back to the box and push it from a different location. “Cooperation in some tasks is possible without direct communication” (Kube & Bonabeau, 2000, p. 100).
8.7 COLLECTIVE SORTING

8.7.1 Spatial Sorting by Ants

We have seen that ant colonies are able to find the shortest routes, to clear nests using blind bulldozing, and to move large objects by employing cooperative transport. Ants are also very adept at sorting objects—producing order from disorder—into useful and interesting spatial patterns. These patterns are frequently described as the products of collective intelligence (Franks & Sendova-Franks, 1992; Holland & Melhuish, 1999; Sendova-Franks, Scholes, Franks, & Melhuish, 2004).

One example of such behaviour is called patch sorting. In patch sorting, two or more classes of objects are placed in separate clusters, so that each cluster contains only one class of objects, and each cluster is spatially separated from the others (Holland & Melhuish, 1999). Ants exhibit patch sorting when they place eggs, larvae, and cocoons into separate piles (Deneubourg et al., 1991) or when they place corpses into different clusters when constructing a cemetery (Theraulaz et al., 2002).

Another example is the annular sorting of Leptothorax ant colonies (Sendova-Franks et al., 2004). A colony’s brood is placed in a single cluster within the nest, but this cluster is highly structured: it is a set of concentric annuli, with each ring comprised of a different type of brood items. “Eggs and small larvae are in the middle whereas medium and large larvae are in concentric annuli increasingly further out towards the periphery” (Sendova-Franks et al., 2004, p. 1095). As a result, brood items that require more care are more easily accessed because they are in the outer rings of the sorted structure.

8.7.2 Stigmergic Sorting by Robots

Researchers have investigated whether interesting spatial patterns be produced by sorting procedures that are completely under stigmergic control (Holland & Melhuish, 1999). Robots used a “gripper” in front to capture a Frisbee lying on the floor. They could also sense Frisbee colour, and had proximity detectors that could register the presence of obstacles such as walls or other robots. The gripper also had a micro-switch that would not be triggered if the robot gripped a single Frisbee, but would be triggered if two or more Frisbees were pushed by the robot.

Holland and Melhuish (1999) explored a number of different algorithms that were used to control the behaviour of each of a collection of 10 robots. One, the pullback algorithm, consisted of three simple rules. First, if the gripper held a Frisbee, and an obstacle was encountered, then the robot made a random turn away from the obstacle. Second, if the gripper held a Frisbee, and another Frisbee was encountered, the robot would...
pull the Frisbee it held back a distance that depended on the Frisbee’s colour, drop it, and turn away from it. Third, if no object was held, then the robot simply moved forward. This algorithm sorted Frisbees into a single cluster that was roughly annular in organization—one colour of Frisbee was in the centre of the cluster, which was surrounded by Frisbees of a different colour. Simpler sets of rules—for instance rules that were not sensitive to Frisbee colour—produced patch sorting.

Importantly, this sorting behaviour is produced by stigmergy. None of the robots directly communicate with one another. Instead, they indirectly communicate by moving Frisbees to different locations; these newly positioned Frisbees will in turn alter the behaviour of the robots, and spatial sorting emerges from this stigmergic system.

Variations of the pullback algorithm are capable of producing more striking annular sorting, and the sorting capabilities of robots and ants have been directly compared (Melhuish, Sendova-Franks, Scholes, Horsfield, & Welsby, 2006; Scholes, Wilson, Sendova-Franks, & Melhuish, 2004; Wilson, Melhuish, Sendova-Franks, & Scholes, 2004). This research demonstrates an interesting interplay between disciplines in which ant behaviour inspires robotics research, which in turn is being used to develop theories about ant behaviour. It also demonstrates that stigmergy is capable of producing spatially organized patterns.

8.8 Stigmergy and Degrees of Embodiment

8.8.1 Extending the Mind into the World

In Chapter 3, we were introduced to an important idea in embodied cognitive science, the leaky or the extended mind (Clark, 1997, 1999, 2003, 2008; Wilson, 2004, 2005). According to the extended mind hypothesis, the mind and its information processing is not separated from the world by the skull. Instead, the mind interacts with the world in such a way that information processing is both part of the brain and part of the world—the boundary between the mind and the world is blurred, or disappeared. We saw in Chapter 3 that this can occur because of cognitive scaffolding. A simple example of cognitive scaffolding is extending memory by using external aids. However, full-blown information processing can be placed outside the traditional mind, into the world, by using appropriate artifacts. We saw an example of this in our discussion of the nomogram (Hutchins, 1995) in Section 3.10.1.

In order for cognitive scaffolding to occur—in order for the mind to extend itself into the world—cognitive agents must be able to interact with and alter the physical world. This was the reason that Chapter 3
argued for a central role of action in the series of cognition.

The extended mind hypothesis can be applied to single cognitive agents. However, when information processing leaks into the world then the extended mind hypothesis can also be applied to a group of agents that operate in a shared environment (Hutchins, 1995). The examples of stigmergy that have been described in the current chapter, as well as those that were introduced in Chapter 1, are also examples of scaffolded or extended group cognition. In particular, stigmergy places the control structure of the information-processing collective into the environment, removing the need for members of the collective to directly communicate with one another. Section 3.12 argued that such stigmergic control could also be considered to be a central characteristic of a modern production system.

Importantly, stigmergic control also requires that agents be able to manipulate the environment in which they are situated. Ants can only solve the travelling salesman problem by laying down a pheromone trail. Robots can only solve the box-pushing problem by contacting and moving the box in order to communicate to other robots that they need to move and push the box from a different location.

8.8.2 Degrees of Embodiment

The Lego robots from previous chapters of this book make it clear that there are different degrees of situation. The walking robots described in Chapter 5 are minimally situated, because they have no sensors and only passively react to physical forces. Vehicle 2, investigated in Chapter 4, is more situated because it uses light sensors to modify motor speeds. The Lego Tortoise, detailed in Chapters 6 and 7, is even more situated because it uses both light and touch sensors.

While these robots illustrate degrees of situation, at first glance it would seem that they are all equally embodied, in the sense that they are all physical artifacts. However, there are other definitions of embodiment that suggest that agents can be embodied to different degrees (Fong et al., 2003).

Fong et al. (2003, p. 149) argue that “embodiment is grounded in the relationship between a system and its environment. The more a robot can perturb an environment, and be perturbed by it, the more it is embodied.” As a result, not all robots are equally embodied. A robot that is more strongly embodied than another is a robot that is more capable of affecting, and being affected by, the environment. Clearly an extended mind, or a stigmergic league-controlled collective, requires strong embodiment. The robots that we have built are all equally embodied, but
that is because none of them is designed to affect the environment.
We now turn to describing a more strongly embodied robot, which can therefore be used to explore ideas such as the extended mind and swarm intelligence.

8.9 THE LEMMING

8.9.1 Lemming Situation

Lemmings was a video game that was introduced in 1991 by Psygnosis for the Commodore Amiga. A player of this game had to assign abilities to a small number of lemmings so that they could alter the environment, and the behaviour of other lemmings, in such a way as to prevent mass migrations that would lead to disaster. In honour of this game, and the behaviours of the agents within it, we have named our more strongly embodied LEGO robot the Lemming. This is because we hoped that it could affect the behaviour of other LEGO Lemmings by manipulating its environment—in particular, by moving light and dark LEGO bricks to different locations in an enclosed arena.

In order to do this, the Lemming (shown below in Figure 8-2) is situated in its environment using three different sensors. An ultrasonic sensor mounted on the top of the Lemming is used to detect and avoid walls and other obstacles, such as other Lemmings. A second ultrasonic sensor mounted near the base of the Lemming is used to detect the presence of to-be-moved bricks. Finally, a light sensor mounted inside the “brick catcher” at the front of the robot is used to analyze a captured brick. In particular, the light sensor detects the colour of the brick, which is then used to elicit an appropriate colour-dependent behaviour from the Lemming.
8.9.2 Lemming Embodiment

The embodiment of the Lemming is also critical. In particular, the “brick catcher” at the front of the robot has a very definite shape. First, its shape is such that when an object is contacted by the brick catcher, it is moved against the light sensor so that its colour can be detected. Second, the shape of the brick catcher is asymmetrical. As a result, if the Lemming turns to its left, the object will remain trapped inside and can be pushed to a new location. However, if the Lemming turns to its right, the object is released from the catcher, and can therefore be left behind in some new position where it can affect the behaviour of a Lemming that encounters it later.

8.10 FORAGING FOR ROBOT PARTS AND WORLD PARTS

8.10.1 Robot Parts

The Lemming is a fairly simple robot to build, and requires the parts that are illustrated below in Figure 8-3. The pages that follow describe how to construct this robot. If the reader would prefer to use wordless, LEGO-style instructions, they are available as a pdf file from the website that supports this book (http://www.bcp.psych.ualberta.ca/~mike/BricksToBrains/).

8.10.2 Bricks to Move

In addition to the robot itself, it is necessary to build objects that are placed in the world for the Lemming to manipulate. These objects are two LEGO bricks high, 4 studs long, and 4 studs wide, as illustrated in Figure 8-4. We built each of these objects using four $2 \times 4$ LEGO bricks.
Importantly, half of these objects are all black, and the other half are all white, because the colour of the object affects robot behaviour. We constructed 28 different black objects and 28 different white objects, requiring a total of 112 black and another 112 white 2 × 4 LEGO bricks.

### 8.11 CHASSIS AND REAR WHEELS

#### 8.11.1 NXT Brick as Chassis

Other NXT robots described in this book (Vehicle 2 in Chapter 4, the Tortoise in Chapter 6, and antiSLAM in Chapter 9) are constructed by building a central spine of beams and liftarms that serves as a chassis to which other components, such as the NXT brick, are attached. A different approach is illustrated in the Lemming, which uses the NXT brick itself as the primary chassis to which other components are attached. As well, in this robot the NXT brick is positioned vertically instead of horizontally, as shown in Figure 8-5. The first step in building the robot is illustrated in this figure. Liftarms are attached to the NXT brick in order to support two small rear wheels that will help keep the robot stable.
8.12 MOUNTING MOTORS

8.12.1 Motors and Cables

The liftarms that were used to create mounts for the rear wheels are also used to attach two motors to the front of the robot, as shown in Figure 8-6. Note that additional parts are attached to each motor; these parts are used later to mount other Lemming components. As well, some of the cables that are later plugged into sensors are best run between the motors and the NXT brick, and they should be attached in this second step for this reason. The ports into which each of the four cables used in this step are inserted are indicated in Figure 8-6.

8.13 UPPER ULTRASONIC SENSOR AND FRONT WHEELS

8.13.1 The Upper Ultrasonic

The next step in constructing the Lemming is to mount an ultrasonic sensor to the top of the robot, following the instructions that are provided at the top of Figure 8-7. This sensor will be pointing slightly upward, and is mounted high on the robot, so that it won't detect objects on the floor in front of the robot. (If the sensor refuses to stay pointed upward when
mounted, a paper clip can be used to keep it pointing in the desired direction.) The purpose of this ultrasonic sensor is to detect larger obstacles, such as the walls that define the arena in which the robot operates.

### 8.13.2 Front Wheel Drive

The Lemming is a front wheel drive machine. The wheels are LEGO 56 × 26 tires mounted onto hubs that are the standard wheels for NXT robots. They are attached to axles that are inserted directly into each motor, and held in place with a half bush, as illustrated at the bottom of Figure 8-7.

8.14 MOUNTING THE LOWER ULTRASONIC SENSOR

#### 8.14.1 Angled Ultrasonics

One of the goals of the Lemming's design was to have it situated in its world in such a way that it could detect, and move toward, objects on the ground in front of it. This was accomplished by mounting a second ultrasonic sensor to the robot; this one is mounted on the front of the robot near its bottom, as illustrated in Figure 8-8. Note that this sensor...
is attached in such a way that it points downward toward the floor. The angle at which this sensor is tilted is important, because it affects how far away the objects are that can be detected. When this sensor is angled as shown in Figure 8-8, then it should be able to detect objects as far away as 50 cm. The tension of the cable used to connect this sensor to the NXT brick should be sufficient to hold it at the desired angle.

8.15 designing the brick catcher

8.15.1 Important Embodiment

The main function of the Lemming is to trap, and to move around, coloured square objects that are in its environment. This is accomplished by making a “brick catcher” that is attached to the front of the robot, and is pushed along the floor by the Lemming. The embodiment of this Lemming component is important in several ways. First, it must be large enough to be able to trap a brick (Figure 8-4) that is encountered, regardless of the orientation of the brick with respect to the brick catcher. Second, it must not be too large—only one brick will be trapped at a time. Third, when a brick is in place, the shape of the brick completes
the shape of the brick catcher, converting it from a catcher into a plow. Fourth, once trapped, the brick must slide into a position that permits its colour to be examined by the robot. Fifth, the shape of the brick catcher is critical for later robot behaviour. The brick catcher is shaped in such a way that if the robot turns to the right, the brick remains trapped, and can be moved by the robot. However, it is also shaped in such a way that if the robot turns to the left, the brick will escape from the brick catcher, and can be left behind by the robot.

The brick catcher begins by attaching various beams and liftarms together in the fashion that is illustrated in Figure 8-9.

8.16 BRICK CATCHER, BRICK PROCESSOR

8.16.1 Embodiment and Situation

Figure 8-10 illustrates how to complete the remainder of the brick catcher. The upper part of this completed structure permits it to be attached to the Lemming’s chassis. The lower part of this completed
structure defines the shape that permits bricks to be captured, analyzed, moved, and released as was noted in Section 8.15. Note that one critical aspect of this shape is a light sensor that forms one of the “walls” used to push a captured brick. The idea behind this design is that the movement of the robot will force the brick directly against this light sensor. Once there, the light sensor can measure whether the captured object is white or black; the sensed colour will determine the subsequent behaviours of the Lemming.

8.17 Completing the Lemming

8.17.1 Final Construction

The Lemming is completed by attaching the brick catcher to the front of the robot as illustrated in Figure 8-11, and by making sure that all sensors and motors have the appropriate cables connected to the ports that were earlier indicated in Figures 8-6 and 8-8. With this construction
completed, we can now turn to developing a subsumption architecture for the Lemming that will permit it to drive forward, avoid walls, and detect, capture, analyze, move, and release coloured objects.

8.18 LEVEL 0: DRIVE AND CALIBRATE

8.18.1 Driving

The most basic level in the subsumption architecture for the Lemming enables it to move forward. This is accomplished by turning both motors on in the forward direction, as indicated in the short NXC program for this level that is listed below. Note that this program is separate from the calibration program that is listed below it after the two double lines of ==.

The Level 0 program uses the `OnFwdSync` command, which includes a `Sync` variable to control both motors. The `OnFwdSync` command is a useful NXC command that permits two motors to be coordinated in a single line of code. However, the default value for `Sync` in Level 0 does not cause the robot to move straight ahead. Instead, it causes the motors to run at slightly different speeds so that the robot gently turns to the right. The value for `Sync` is set in Level -1 (described in Section 8.22) which is used to set motor speeds by pooling conditions detected by higher levels in the architecture. The value that is used requires that each Lemming be calibrated.
8.18.2 Calibration

In some instances, the Lemming will detect objects and move straight toward them. However, no two NXT motors are exactly the same, so an individual Lemming must be calibrated to find, in essence, the precise value for Sync that will produce straight movement. The program that is listed below the Level 0 code is a separate program for calibrating a Lemming. The program is run, and the behaviour of the Lemming is watched. If it does not go straight ahead, the value for straight_calibration is changed as described. When the Lemming moves in a straight line, the value for straight_calibration is recorded and used in the main task that is described later.

```c
/* ===== Level 0: Drive the Lemming Forwards ========= */
task Drive(){
    while(true){
        OnFwdSync(BothMotors, DriveSpeed, Sync);
    }
} /* task Drive ends here! */
/* ------------------------------- */
/* ------------------------------- */
/* CALIBRATION PROGRAM BELOW */
calibrate so the Lemming can drive straight
--------------------------------------------- */
#define LeftMotor OUT_B
#define RightMotor OUT_C
#define BothMotors OUT_BC
/* = Drive = */
int speed = 65; //set all the lemmings at this speed
int straight_calibration = 0;
/* use this value in your code. Adjust it so that the lemming will
drive in a straight line when you run the code. Try
zero first. If not zero it will probably be between -5 and 5 */
task main(){
    while(true){
        OnFwdSync(BothMotors, speed, straight_calibration);
    }
}
```

246 FROM BRICKS TO BRAINS
8.19 LEVEL 1: DODGE OBSTACLES

8.19.1 The Lemming’s Umwelt

A Lemming moves around white and black bricks that it detects on the floor of a room. The room’s walls are obstacles that can damage the robot if it collides with them. Other Lemmings might also be part of the environment; they too need to be avoided.

8.19.2 Avoiding Obstacles

Level 1 in the Lemming’s subsumption architecture uses the top ultrasonic sensor to detect to-be-avoided obstacles. If an obstacle is detected, the robot spins around away from it. The direction in which it spins depends upon whether it is carrying a brick, and upon the colour of that brick. There are two subtleties in the code below.

First, because the Lemming can exist in an environment that includes other Lemmings that are also emitting ultrasonic signals, it checks its ultrasonic signal twice, with a short delay between checks, to ensure that it is not responding to the signal sent by another Lemming. The constant movement of the robots makes it unlikely that such a “rogue ultrasonic” signal will be detected twice!

Second, in rare instances a Lemming might be pointed at such an angle so that it will encounter a wall, but cannot receive an echo from it. The CheckStuck routine is used to solve this problem. It too works by checking the ultrasonic signal twice. Because Lemmings constantly move, it is unlikely that they will receive identical ultrasonic signals a second or so apart. If this does occur, the CheckStuck routine treats this as being stuck, and initiates avoidance behaviour by setting the collision variable.

```c
/*===== Level 1: Dodge ==============================*/
If the wall is within the wall sensor's range, spin for a little while. */
int wallDistance, lastReading, SpinDirection;
const int KEEP_DIRECTION = 1, DUMP_DIRECTION = -1, HALF_SPIN = -50, COLLISION = 27;
int spinTime = 750, stuckTime = 1500, NearWall = COLLISION;
bool collision;
task Dodge(){
collision = false; SpinDirection = DUMP_DIRECTION;
while(true){
    until (collision || (SensorUS(WallSensor) < NearWall));
Wait(50);
    if (collision || (SensorUS(WallSensor) < NearWall)){ //double check to make sure it is not a US error!
        collision = true; Wander = 0;
    }
}
```

Chapter 8 Embodiment, Stigmergy, and Swarm Intelligence 247
DriveSpeed *= SpinDirection; Spin = SpinDirection * HALF_SPIN;
Wait(spintime);
Wander = 1; DriveSpeed = BaseSpeed;
Spin = STRAIGHT; SpinDirection = DUMP_DIRECTION;
NearWall = COLLISION;
collision = false;
}
}
task CheckStuck()
{
    while(true)
    {
        lastReading = SensorUS(WallSensor);
        Wait(stuckTime);
        if(lastReading == SensorUS(WallSensor)) collision = true;
    }
}

8.20 LEVEL 2: SEEK BRICKS

8.20.1 Brick Attraction

The main purpose of the Lemming is to move large bricks around in its environment. In order to do this efficiently, it would be convenient if the Lemming could sense the presence of bricks in front of it, and move itself toward these detected objects. In short, it would be nice if bricks could attract the Lemming.

8.20.2 Using the Lower Ultrasonic

Level 2 uses the lower ultrasonic sensor to accomplish “brick attraction.” The lower ultrasonic sensor is mounted in such a way that it can detect the presence of one or more of the large bricks (Figure 8-4) when they are 50 cm or more away from the Lemming. Brick attraction is accomplished by having the Lemming move straight (i.e., stop its gentle turning) when the lower ultrasonic detects an object. It moves straight for a set period of time that is long enough for the Lemming to physically encounter the object that was sensed by the ultrasonic sensor.

The NXC code for Level 2 is provided below. Note that it essentially works by setting the variable Wander to 0 for a set period of time if a brick has been sensed; Wander is set to 1 when no brick is sensed and when the straightTime has elapsed. Wander is a value that is passed to the to-be-described Level -1, which uses this information, as well as information from other levels, to determine motor speeds at any given time.
/* ===== Level 2: Seek Bricks =========================
If the brick sensor detects a brick, straighten out. The sensor is forward-
mounted, so this will result in a brick attraction. */

int brickthreshold = 120, brickDistance, straightTime = 1000;
task Seek(){
    while(true){
        if(SensorUS(StickSensor) < brickthreshold){
            Wander = 0; Wait(straightTime);
        }
        else Wander = 1;
    }
}

8.21 LEVEL 3: PROCESS BRICK COLOURS

8.21.1 Bricks and Behaviour

While the general purpose of the Lemming is to move bricks around,
its behaviour is more specific because where it pushes a brick to, and
where it leaves the brick, depends on the colour of the brick. That is,
one a brick is trapped by the brick catcher, the brick is pushed against
the light sensor mounted on the bottom of the Lemming. The light sen-
sor is used to classify the brick as being either light (e.g., white) or dark
(e.g., black). If the brick is light, the Lemming acts as a blind bulldozer.
It pushes the light brick forward until it encounters a wall (via Level 1).
It then spins from the wall in such a way that the light brick is left at
the wall. However, if the brick is dark, the behaviour of the Lemming
is quite different. It might push the brick to a wall, but when the wall
is detected it will turn in the opposite direction, so that the dark brick
is not left by the wall. Instead, the Lemming will only spin in a direc-
tion that deposits the dark brick when it detects another brick using
the lower ultrasonic sensor (Level 2). In short, the Lemming attempts to
cluster dark bricks together in the interior of its environment.

These behaviours are accomplished by setting the direction of the
spin on the basis of detected brick colour, as shown in the NXC code
below. The variables set in the code below affect the motor behaviours
that are controlled by Level -1, which is described in the next Section.

/* ===== Level 3: seeBricks ================================
If carrying a dark brick, drop it at other bricks but keep it at walls.
If carrying a light brick, drop it at walls but keep it at other bricks. */
// to cope with inaccuracy of sensor at close range.. collision detected from further away..
// drive straight, then, dump
const int FAR_COLLISION = 65;
int noBrick = dark_threshold, lightBrick = light_threshold, brickCollision = 30;
int currentBrick, wallFar = FAR_COLLISION, untilDump = 500;
bool foundDark, foundLight;
task seeBrick(){
    while(true){
        until(ColourSensor > noBrick);
        if (ColourSensor > lightBrick){
            SpinDirection = DUMP_DIRECTION; foundLight = true;
            NearWall = COLLISION; // With light brick, come close to wall
        }
        else if(!(collision) && (SensorUS(BrickSensor) <= brickCollision) &&
            (SensorUS(WallSensor) > wallFar)){
            PlayTone(587,200);
            Wander = 0; Wait(untilDump);
            if (ColourSensor < lightBrick){ //double check to make sure it is not a light brick
                SpinDirection = DUMP_DIRECTION; collision = true;
                foundDark = true;
            }
        }
        until(!(collision) || (ColourSensor > lightBrick));
    }
}

8.22 LEVEL -1: INTEGRATE LEVELS TO CONTROL MOTORS

8.22.1 Multiple Motor Influences

Levels 0, 1, 2, and 3 of the Lemming’s subsumption architecture are all very simple, but they lead to different effects on the motors. Level 0 moves the robot forward, and requires it to turn slightly as it moves. If Level 1 detects an obstacle, then the motors must spin the robot to avoid it. If Level 2 detects a brick, then the Lemming is steered straight toward it. If Level 3 has detected a coloured brick, then this will affect the direction that it spins when it next encounters a wall or another brick.

The purpose of Level -1 is to integrate these different motor control signals so that the behaviour of the Lemming reflects the demands of the other levels of the architecture. The NXC code for this level is provided below. Interestingly, this is done by computing a value for Sync that reflects the signals that are coming from the other levels. Sync is
the constant that controls the relative speeds of the two motors, and is used to control motor speeds in a single line in Level 0 (Section 8.18). The *Speed task sets the Sync variable by considering several different variables that will affect the relationship between motors: Should the robot be moving straight, or should it be wandering with a slight curve? Should it be avoiding an obstacle by spinning? This level computes Sync with a simple equation that combines variables that are affected by higher levels, and which produces the desired motor behaviour of the robot.

```c
/*===== iLevel -1: Integration ===========================================
Sets the two motor speeds by combining the upper layers’ controls. */
const int SET_MINOR_TURN = 10;
/* the Lemming turns slightly to the right when not detecting a brick, SET_MINOR_TURN
determines how much it turns. default = 10 */
const int STRAIGHT = straight_calibration;
const int FULL_SPIN = 100, MINOR_TURN = STRAIGHT + (-1 * SET_MINOR_TURN);
int DriveSpeed, BaseSpeed, Wander, Spin, Sync, BaseSync = MINOR_TURN;
task Speed(){
    DriveSpeed = BaseSpeed;
    BaseSync = MINOR_TURN;
    while(true){
        Sync = (BaseSync * Wander) + Spin; //Compute Sync by combining variables
    }
}
```

### 8.23 Putting all the levels together

#### 8.23.1 The Main Task

The complete program for the Lemming is available from the website that supports this book (http://www.bcp.psych.ualberta.ca/~mike/BricksToBrains/). The main task, whose NXC code is listed below, is used to define the constants that are used in the various levels, to assign values for parameters that calibrate individual Lemmings, and to start all the previously described levels running. Note that one of these tasks is `keepAwake`, whose code is also provided below. The behaviour of a Lemming is usually observed by running it in its test environment for a considerable length of time (30–40 minutes). The NXC brick will automatically turn itself off after a few minutes in order to conserve battery power. The `keepAwake` task is run to reset the brick’s timer to prevent this from happening.

Note too that the `straight_calibration` value, obtained from playing with the calibration routine that was described in Section 8.18, is assigned in this final section of code. For the Lemming that was to be tested using
this subsumption architecture, the value of this variable was found to be equal to 3. The light_threshold and dark_threshold values are used to identify light bricks and dark bricks, and could be modified to permit the Lemming to classify bricks whose colours were other than black or white.

```c
#define WallSensor S1
#define BrickSensor S2
#define ColourSensor SENSOR_3
#define LeftMotor OUT_B
#define RightMotor OUT_C
#define BothMotors OUT_BC

/*===== Calibration: Set the lemming specific constants here. */
const int straight_calibration = 3, light_threshold = 540, dark_threshold = 340;

/*===== KeepAwake ====================================================
Reset the sleep timer so the lemming does not shut off automatically! */
const int TEN_MINUTES =36000000;
task keepAwake()
{
    while(true)
    {
        Wait(TEN_MINUTES);
        ResetSleepTimer();
    }
}

/*===== Main Task ====================================================*/
task main(){
    SetSensorType(WallSensor, SENSOR_TYPE_LOWSPEED);
    SetSensorType(BrickSensor, SENSOR_TYPE_LOWSPEED);
    SetSensorType(S3, SENSOR_TYPE_LIGHT_ACTIVE);
    SetSensorMode(S3, SENSOR_MODE_RAW);
    Wander = 1;
    Spin = 0;
    BaseSpeed = 65;
    start Speed;
    start Drive;
    start Dodge;
    start CheckStuck;
    start Seek;
    start seeBrick;
    start keepAwake;
}
```

252 FROM BRICKS TO BRAINS
8.24 THE LONELY LEMMING

8.24.1 Lemming Behaviour

Now that the Lemming has been constructed and programmed, we are able to observe its behaviour. Furthermore, we can construct copies of the Lemming, give each of them the same program, and explore whether a collection of Lemmings behaves any differently than does a single machine.

To start our behavioural explorations, let us create a checkerboard pattern of 28 black and 28 white squares (each square constructed as shown in Figure 8-4) and place this pattern in the middle of a small testing room that was 2.4 metres long and 1.65 metres wide. Figure 8-12 is a photograph of the floor of the room, taken from above, before the robot is released.

Now, a single Lemming is released into the room. We expect that it will rearrange bricks on the floor. Given the program that we have created, we might also expect that it will create a roughly annular rearrangement of these objects, by pushing the white squares out toward the walls, and by keeping the black squares in the middle of the room. The behaviour of a single Lemming in this environment is provided as part of Video 8-1, which is available from the website.
An examination of the behaviour of the Lemming in the early part of this video supports our expectations. The Lemming moves about the environment, pushing the objects aside as it wanders. On occasion, a square will be captured in the brick catcher. The behaviour that immediately follows this event depends on the colour of the captured object. If it is white, then the robot moves across to the wall that it is facing, and deposits the brick near the wall. If it is black, then the behaviour is less regular. The robot seems to spin a bit, seeking a direction, moving in this direction, and then suddenly choosing a different path. Usually after behaving like this for a moment, the black square is deposited in the middle of the room, near other bricks.

However, if the single Lemming is left to run long enough (70 minutes in the example below), the resulting sorting of the bricks is not quite as expected, as is illustrated in Figure 8-13. Most of the black bricks remain in a loose cluster in the middle of the room, which is expected. However, the majority of the white bricks are not just near a wall: instead, these “junk” bricks have been bulldozed into one of the corners of the room! The Lemming was certainly not intentionally designed to do this: none of its sensors or programmed tasks were designed to detect corners. How is it able to push most of the white objects, and few of the black ones, into the corners of its environment?
8.25 COLLECTIVE COLLECTING

8.25.1 Two Lemmings

The bricks on the floor of the testing room provide the potential for stigmergic control of Lemming behaviour. That is, the behaviour of one Lemming is determined in part by the colour of an object that it has captured; when it moves this object to a different location in the testing room, this has the potential to affect the future of this behaviour, or the behaviour of some other Lemming that might be sharing the environment.

To begin to explore the collective behaviour of Lemmings, the objects in the testing room were set up in the pattern that is illustrated in Figure 8-12, and two Lemmings were released into the room. The behaviour of a pair of Lemmings in this situation is also demonstrated in Video 8-1.

After 30 minutes, the objects are arranged in the room as shown in Figure 8-14. This final arrangement is very similar to the one produced by a single Lemming (Figure 8-13): almost all of the white bricks are in one of the four corners of the room, and almost all of the black bricks are in a loose cluster in the middle. The only noticeable difference between this end state and the one provided in Figure 8-14 is that in the latter all of the white objects have been removed from the middle of the room. That is, the white bricks that are not in corners are very close to walls, and appear to be quite far from black bricks.
8.25.2 Three Lemmings

Further explorations were conducted by letting three Lemmings work in a room that began as shown in Figure 8-12. When done, they too had arranged the bricks by placing most of the white ones in the corners, and a loose cloud of the black ones in the middle of the room. Their work is shown in Figure 8-15. There is very little to distinguish this result from the one illustrated in Figure 8-14, except the time of the run: almost all of the white bricks had been moved into corners after Lemming activity that only lasted about 11 minutes. In the next section, we use comparisons of Lemming sorting speeds to consider whether colonies of this robot seem to exhibit collective intelligence.

8.26 EXPLAINING SORTING INTO CORNERS

8.26.1 Corner Analysis

One of the surprising behaviours of the Lemming is its strong tendency to push white bricks into the corners of the testing room. Consider this behaviour from an analytic perspective: Imagine giving a designer some photographs that depicted the starting condition of the testing room (Figure 8-12), as well as the goal condition of the room (any of Figures 8-13, 8-14, or 8-15). One could also provide the designer with a pre-constructed Lemming. The designer’s task would be to program
the robot to convert the starting state of the room into the goal state.

In this scenario, the designer is likely to adopt an analytic approach. He or she would examine the photographs in an attempt to determine the difference between the start and goal states, and use these to sketch a general algorithm. A plausible algorithm might be: Find a brick. If it is black, leave it in the middle of the room. If it is white, leave it in a corner. Repeat.

This algorithm raises some challenges. How does one program a Lemming to find a corner of the room? From the analytic perspective, developing the desired program would be difficult.

8.26.2 Corners for Free

From our synthetic knowledge of the Lemming, it should be clear that it does not use the analytic algorithm described above. We know that its program does not identify corners and move white objects to them. If this behaviour is not programmed directly into the robot, then how does it occur?

Consider a completely different system that is also constrained by “walls”: an artificial neural network that is called the brainstate-in-a-box (Anderson, Silverstein, Ritz, & Jones, 1977). This network consists of a set of processors that send signals to one another. Each processor produces internal activity values, but they cannot be lower than -1, or higher than +1.

In the brainstate-in-a-box, the set of neurons can be represented as a vector in a space, where the vector coordinates are the activity values of each processor in the network. When the processors signal one another, this vector grows in length. Soon, one or more of the neurons reaches a value of -1 or +1. This is equivalent to the vector (the brainstate) hitting a “wall” of a box that surrounds the vector. This wall pushes the vector along it. Eventually, the brainstate hits other walls, and finally is pushed into a corner of the surrounding box. In this corner, the brainstate is forced to stop growing, and its activities represent a memory that has been recalled from the network.

The testing room in which the Lemming roams restricts its movement in much the same way. When the robot wanders, it has a tendency to gently turn to its right. However, when it encounters a wall of the room, its movement is restricted. The result is that the Lemming moves along the wall that it encounters. Many examples of this behaviour—which emerges from an interaction between the Lemming and its environment—can be seen in Video 8-1. (The difference between the Lemming and the brainstate is that the Lemming has the ability to turn away from corners!)

How does this cause white bricks to be pushed into corners? The
answer to this question comes from realizing that these bricks are not usually pushed directly into corners. Instead, they are first pushed near one of the walls of the testing room. Later, when a Lemming is moving along this wall (because of the interaction between its wandering behaviour and the structure of the testing room) it will catch this brick. It will then push this brick along the wall until it encounters another wall. Where will this wall be? It will be at a corner of the testing room. Now the white brick will be deposited at its final location, the corner, from which it will not be moved because the Lemmings tend not to venture into corners.

In short, sorting white bricks into corners emerges from the interaction of two Lemming behaviours: pushing white bricks to walls, and moving along walls when wandering.
8.27 DO LEMMINGS HAVE COLLECTIVE INTELLIGENCE?

8.27.1 “Speed” of Work

As was noted earlier, Beni and Wang (1991) argued that one of the characteristics of collective intelligence was that as more agents were added to the collective, the amount of work done rose exponentially (instead of linearly, see Figure 8-1). Is this the case for a collective of Lemmings?

Typically, a single Lemming is finished sorting the blocks (i.e., has moved about 85% of the white bricks into corners) after approximately 70 minutes. In contrast, two Lemmings achieve the same result after about 30 minutes, and three Lemmings take only 11 minutes.

We converted these times to “speed” of work by taking the single Lemming’s time as the standard, and dividing each of the times noted above into this standard value 70. If there is no collective Lemming intelligence, then this value would be equal to the number of Lemmings working together. However, the results that are plotted in Figure 8-16 show a much different pattern. Note that the graph in Figure 8-16 is very similar to the graph in Figure 8-1 that represents Beni and Wang’s (1991) definition for collective intelligence. Thus it would appear that the phrase “collective intelligence” can be applied to a colony of Lemmings.

8.28 EXPLAINING COLLECTIVE INTELLIGENCE

8.28.1 Brick Dynamics

The results presented in Section 8.27 indicate that a small colony of Lemmings demonstrates collective intelligence. That is, three Lemmings sort the room considerably more than three times faster than does a single Lemming. Such collective intelligence is not directly programmed into a colony of Lemmings. The program that was described earlier controls the behaviour of a single robot. It does not detail any changes in this behaviour if other robots or detected, nor does it provide any explicit communication between robots. How, then, does collective intelligence emerge in a colony of such simple machines?

Clues to answering this question are provided by examining the parts of Video 8-1 that compare the behaviour of 1, 2, or 3 Lemmings working together. The first clue comes from ignoring the movement of the robots, and just paying attention to very general changes in brick positions over time.

How do the positions of the bricks in the room change from start to finish? The dynamics of the bricks are similar regardless of the number of robots involved (with the exception of the time course). First, the bricks start out in a regular grid. Second, the grid is destroyed, both
by catching bricks and plowing bricks, so that white and black bricks are intermingled in a more compressed and random arrangement. Third, bricks migrate away from the middle of the room, as both black and white bricks are pushed to varying distances from the four walls. Fourth, white bricks migrate from walls to corners (see Section 8.26), while black bricks migrate back toward the middle of the room.

The second clue to accounting for collective intelligence comes from considering Lemming movement. The brick dynamics discussed above indicate that a general tendency in brick sorting is for bricks to be pushed away from the middle of the room, and then for some bricks (i.e., the black bricks) to be pushed back into the middle. Both of these results require that Lemmings move through the middle of the room. However, the likelihood of this happening is one of the striking differences between the behaviour of a single Lemming and a small colony of Lemmings.

When a single Lemming begins in the testing room, it has a marked tendency to remain at the room's outskirts, and rarely moves through the middle of the room. In contrast, when a group of Lemmings starts in the room, one or more of the robots moves through the room's middle almost immediately, and robots are frequently seen in this area. As a result, the progression of brick dynamics that is associated with sorting occurs much more rapidly. It seems that an increased tendency to explore the middle of the room is the primary cause of accelerated sorting. But why is it that a group of Lemmings shows this tendency, while a single Lemming does not?

8.28.2 Interaction and the Middle

There are two types of interactions that cause groups of Lemmings to venture into the testing room's middle more frequently than does a single machine.

First, all Lemmings turn away from obstacles, but the only obstacle that a single Lemming will encounter is a wall of the room. In contrast, in groups of robots, each robot serves as an additional, moving obstacle, increasing the frequency avoidance behaviour, changing the location of this behaviour, and causing Lemmings to venture into the middle of the room very early.

Second, once Lemmings have moved through the middle of the room, the bricks in the starting grid are pushed into tighter groups, and these groups are more likely to be detected by the lower ultrasonic sensor of the robot. That is, clustered groups of bricks are more likely to attract a robot to them than are bricks in the original grid. The early destruction
of the starting grid of bricks by a colony produces clusters of bricks that attract robots and accelerate the dynamics of brick sorting. As a single robot destroys the starting grid much later—because of its avoidance of the room’s middle—brick dynamics proceed at a much slower pace.

8.29 Implications and Future Directions

8.29.1 Implications

One idea emerging from embodied cognitive science is the extended mind: the idea that when external resources are used to scaffold cognition, the mind has extended beyond the confines of the skull, and has leaked into the world (Clark, 1997, 2008; Clark & Chalmers, 1998; Wilson, 2004). This idea was discussed in more detail earlier in Chapter 3.

A cognitive agent with an extended mind must be able to manipulate its environment. It must exhibit a higher degree of embodiment, in the sense of Fong et al. (2003), than do the LEGO robots described in preceding chapters. The Lemming is an example of a simple LEGO robot that is more embodied than these previous machines, because it is capable of moving objects from one location to another. This embodiment was demonstrated when the Lemming “blind bulldozed” white bricks to the outer edges of its environment, while it moved black bricks to a more central location. This task was inspired by studies of sorting in ants (Deneubourg et al., 1991; Sendova-Franks et al., 2004; Theraulaz et al., 2002) and robots (Holland & Melhuish, 1999; Melhuish et al., 2006; Scholes et al., 2004; Wilson et al., 2004).

The sorting behaviours of collectives are cited as examples of stigmergy (Holland & Melhuish, 1999), because sorting can be accomplished by groups of agents that do not communicate with one another directly, but do so indirectly by manipulating the environment (i.e., by changing the locations of to-be-sorted objects in the shared world). The Lemming’s sorting behaviour illustrates stigmergy in this sense. Clearly, changing the location of a brick in the testing room had the potential of altering the future behaviour of a Lemming that might encounter this brick in its new location. An unexpected result of the stigmergic interactions of the robots was the dramatic acceleration of sorting that resulted when more than one Lemming was turned loose in the testing room. The fairly simple notion of stigmergy demonstrated by the Lemming can produce an interesting example of collective intelligence, according to one definition of this phenomenon (Beni & Wang, 1991).
8.29.2 Future Directions

Nevertheless, the kind of stigmergy illustrated by the Lemming is very simple. Fortunately, the Lemming is a platform that could be easily extended to explore the notions of stigmergy and embodiment in more complicated ways.

Some of these extensions would entail altering the subsumption architecture of the robot, and possibly “tweaking” its sensory abilities. For instance, what would be the effect of altering the behaviour of a machine as it wanders, permitting it to turn to the right or to the left? Would it be possible to change the brick release behaviour of a Lemming so that it more closely resembled some of the algorithms for annular sorting, such as the pullback algorithm (Holland & Melhuish, 1999), and as a result produce more compact clusters of bricks at the end of sorting? The current version of the Lemming works by distinguishing light from dark bricks. Might it be possible to program the Lemming to distinguish more than two brick types — and generate more than two types of behaviour — by elaborating its light sensing routines?

Other interesting extensions would entail recognizing that the extended mind can involve both a scaffolding world and internal representations. Imagine a Lemming that keeps a memory of the last one or two bricks that it has captured, and whose behaviour depends not only on the colour of the current brick, but also these memories. Would such a device be capable of carrying out simple computations, using the floor of the testing room as an external memory or scratchpad?

That the Lemming could be used to explore such questions suggests that it is much more than a LEGO toy. The conception of the LEGO robots as toys, totems, or tools is the topic of the next and final chapter of this book.