3 Distributed intelligence

All evidence indicates that flock motion must be merely the aggregate result of the actions of individual animals each acting solely on the basis of its local perception of the world.

Craig Reynolds (1987)

In the last chapter we concluded that pattern formation in natural systems occurs as a consequence of simple local rules. We had a look at plants, at artificial creatures in the game of life, and at seashells. We now look at the emergence of behavioral patterns, which can be interpreted by an external observer as some kind of "distributed intelligence". In this chapter we will proceed by inspecting some examples of robots and natural agents. We start with an experiment in collective robotics. We then discuss self-organizing phenomena in insect societies. Next, we briefly present Craig Reynolds's famous *boids*. Finally, we discuss some "guiding heuristics for decentralized thinking", as outlined by Mitchel Resnick.

3.1 An experiment: the Swiss robots

The Didabots are cleaning up

In what follows we summarize experiments conducted by Maris and te Boekhorst (1996) who studied a collective heap building process by a group of simple robots, called Didabots (see Figure 3.2a). Instead of predefining "high-level" capacities, Maris and te Boekhorst exploit the physical structure of the robots and the self-organizing properties of group processes. The main idea behind the experiments is that seemingly complex patterns of behavior (such as heap building) can result from a limited set of simple rules that steer the interactions between entities (e.g., robots) and their environment. This idea has, for example, been successfully applied to explain the behavior of social insects (see below).

Now have a look at Figure 3.1.

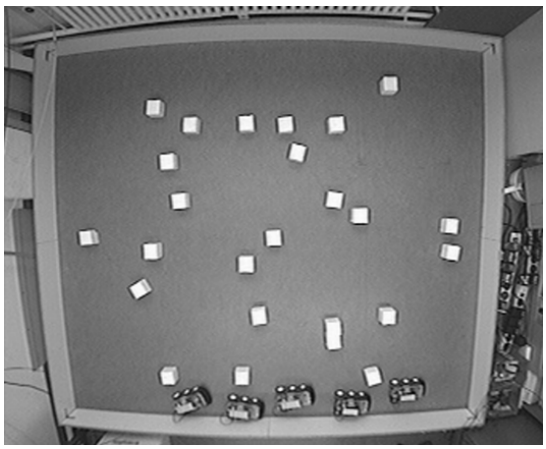

Figure 3.1: *Didabots in their arena. There is an arena with a number of Didabots, typically 3 to 5. All they can do is avoid obstacles.*

The Didabots present in the arena are equipped with infrared sensors that can be used to measure proximity. They show high activation if they are close to an object and low or zero activation if they are far away. The range of the infrared sensors is on the order of 5 cm, i.e., relatively short range. The sensors are located on the left and on the right side of the robots (see picture 3.2 (b) below). All the Didabots in this experiment can do is avoiding obstacles. They are programmed with the following simple control rule: If there is sensory stimulation on the left, turn (a bit) to the right, if there is sensory stimulation on the right, turn (a bit) to the left.

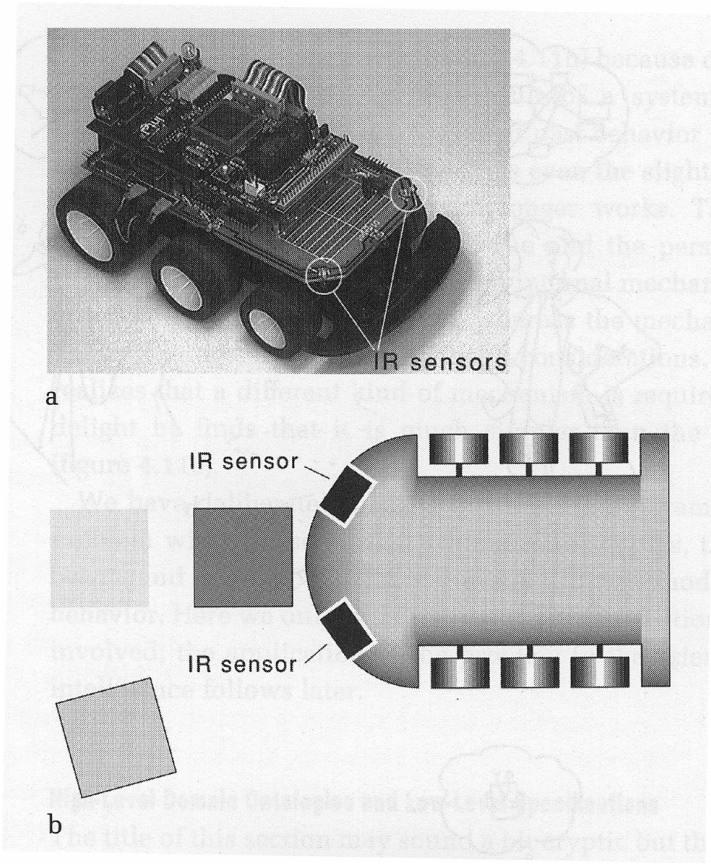

Figure 3.2: *(a) Picture of a Didabot. (b) Infrared- Sensor configuration of Didabot.*

Now look at the sequence of pictures shown in Figure 3.3. Initially the cubes are randomly distributed. Over time, a number of clusters start to form. In the end, there are only two clusters and a number of cubes along the walls of the arena. These experiments were performed many times. The result is very consistent — there are always a few clusters and a few cubes left along the walls. What would you say the robots are doing?

"They are cleaning up"; "They are trying to build clusters of cubes"; "They are making free space". These are typical answers, and they are fine if we are aware of the fact that they represent the observer's perspective. They describe the behavior. The second answer even attributes an intention by using the word "trying". Since we are the designers, we can say very clearly what the robots were programmed to do: to avoid obstacles!

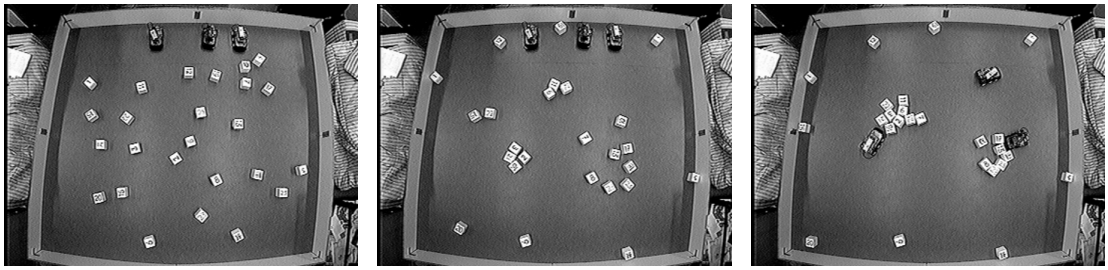

Figure 3.3: *Example of heap building by Didabots. Initially the cubes are randomly distributed. Over time, a number of clusters start to form. In the end, there are only two clusters and a number of cubes along the walls of the arena.*

The complexity of the behavior is a result of a process of self-organization of many simple elements: The robots with their simple control rule. The Didabots use the sensors on the front left and front right parts of the robot. Normally, they move forward. If they are too close to an obstacle, i.e., the obstacle is within reach of one of the sensors, they simply make a turn to the other side. If they encounter a cube head on, neither the left nor the right sensor detects an obstacle and the Didabot simply continues to move forward. At the same time, it pushes the cube. However, it pushes the cube because it does not "see" it, not because it was programmed to push it. For how long does it push the cube? Until the cube either moves to the side and the Didabot loses it, or until it encounters another cube to the left or the right. It then turns away, thus leaving both cubes together. Now there are already two cubes together, and the chance that yet another cube will be deposited near them has increased. Thus, the robots have changed their environment, which in turn influences their behavior. While it is not possible to predict exactly where the clusters will be formed, we can predict with high certainty that only a small number of clusters will be formed in environments with the geometrical proportions used in the experiment.

The kind of self-organization displayed by the Didabots in this experiment is also called self-organization without structural changes: If at the end of the experiments, the cubes are again randomly distributed and the Didabots are put to work on the same task, their behavior will be the same — nothing has changed internally. This is also the kind of self-organization displayed by physical systems, see for instance the famous Bénard cells (when a heat gradient is applied to a liquid, the individual molecules organize into "rolls"). As soon as the energy input is switched off, the system gets back to its original state. We talk about self-organization with structural changes, whenever something changes within the agent in order that the future behavior of the agent will be different. Such processes of self-organization with structural changes are found in the artificial chimp societies of Hemelrijk (see Section 5.2). They are also found in the ontogenetic development of the brain. It is crucial that the organism changes over time, since otherwise it could not improve its behavior.

Similar principles as the ones observed in the Didabot experiments can also be found in natural agents such as ants and primates. Whereas in ants seemingly sophisticated group decisions may raise suspicion and induce a search for simpler mechanisms, this is not the case for primates (i.e., monkeys and apes). Let us now look at a number of examples.

3.2 Collective intelligence: ants and termites

Self-organization in a "super-organism"

In his article in the NZZ (see references) Rudiger Wehner describes societies of social insects composed of thousands of individuals, which have "cognitive abilities" that by far transcend the abilities of each of the individual members. This happens, as if the society is ruled by the invisible hand of a central organizer.

The distribution of brood and nourishment in the comb of honey bees is not random, but forms a regular pattern, which is organized in such a way, that the central brooding region is close to a region containing pollen and one containing nectar (providing protein and carbohydrates for the brood). Despite the fact that, due to the intake and outtake of pollen and nectar, this pattern is changing all the time on a local scale, observed from a more global scale, the pattern stays stable. By performing experiments, it has been discovered that this is not the result of an individual bee being aware of the global pattern of brood- and food-distribution in the comb, but of three simple local rules, which each individual bee follows. Please note, that the individual bee does not know whether and how the cells of the comb are filled with nectar and pollen, but it only follows the three simple rules stated below. In other words, these three rules are sufficient to create the global pattern.

1. Deposit brood in cells next to cells already containing brood.
2. Deposit nectar and pollen in discretionary cells but empty the cells closest to the brood first.
3. Extract more pollen than nectar.

By following these three local rules bees create a global distribution-pattern. Thus the distribution-pattern is an emergent phenomenon resulting from the application of local rules and an example of a process of self-organization in biology.

Another example of the combined application of local rules leading to a global result is the process of food-allocation and food-collection in honeybee colonies. The decision how many bees are collecting food, and where they are collecting it, depends on the time they have to wait when delivering the food at the entrance of the comb to the bees in charge of depositing such food in the respective cells. Based on the number of cells already filled, these bees need more time to find an empty cell and consequently the "search-bees" have to wait longer. This in turn leads them to search qualitatively more valuable food, which is usually more difficult to find and consequently, to spend more time searching for food.

Thus no central coordinator is needed to organize the search for food, and its storage. The parallel application of simple local rules solves the complex problem in a much more flexible and efficient way. Social insects are individuals, which by "working together in parallel" create a super-organism capable of solving even the most complex problems without any central organizer.

This idea has been taken up by social scientists and economists, who use computer simulations to study complex social and economic behavior. They design autonomous agents, which follow simple local rules in a specific environment and by copying ideas from biology and evolution they succeed in growing artificial societies bottom up.

The article on self-organization in a "super-organisms" by Rüdiger Wehner was distributed in class. Exact reference: Wehner, R. (1998). Selbstorganisation im Superorganismus. Kollektive Intelligenz sozialer Insekten. NZZ Forschung und Technik, 14, Januar 1998, S.61. This collective intelligence unites biologists, computer scientists, and economists in an interdisciplinary endeavor.

Referring back to our robot experiments in the previous section, we have another instance of sorting behavior. Sorting behavior is also observed in ants.

Deneubourg's model of sorting behavior in real ants

The examination of an ant's nest yields that brood and food are not randomly distributed, but that there are piles of eggs, larvae, cocoons, etc. How can ants do this? If the contents of the nest are distributed onto a surface, very rapidly the workers will gather the brood into a place of shelter and then sort it into different piles. Deneubourg and his colleagues show that this sorting behavior can be achieved without explicit communication between the ants.

The model works as follows. Ants can only recognize objects if they are immediately in front of them. If an object is far from other objects, the probability of the ant picking it up is high. If there are other objects present the probability is low. If the ant is carrying an object, the probability of putting it down increases if there are similar objects in its environment. Here are the formulas:

$$p(pick\ up) = \left(\frac{k^+}{(k^+ + f)} \right)^2$$

where f is an estimation of the fraction of nearby points occupied by objects of the same type and k^+ is a constant. If f=0, i.e., there are no similar objects nearby, the object will be picked up with certainty. If f=k^+, then *p(pick up)*=1/4 and if f approaches 1 *p(pick up)* decreases. The probability of putting down an object is

$$p(put\ down) = \left(\frac{f}{(k^- + f)} \right)^2$$

where f is the same as before and k^- is again a constant. p(put down) = 0 if f = 0, i.e., if there are no similar objects nearby the probability of putting the object down approaches 0. The more objects of the same type that are nearby, the larger is *p(put down)*. The development of the clusters for real ants and for a simulation is shown in Figure 3.4. Sorting is achieved by these simple probabilistic rules. There is no direct communication between the ants. The sorting behavior is an emergent property.

While many people would agree that artificial life-like models have explanatory power for ant societies, they would be skeptical about higher animals or humans. For instance, Charlotte Hemelrijk and Rene te Boekhorst, two primatologists, are convinced that this kind of modeling technique can also be applied to societies of very high-level mammals like chimpanzees or orangutans. Hemelrijk uses computer simulations to study emergent phenomena in societies of artificial creatures, which, for her, are abstract simulations of orangutans. In an instructive paper entitled "Cooperation without genes, games, or cognition", Hemelrijk (1997) demonstrates that cooperation in the sense of helping behavior is entirely

emergent from interactive factors. Often, what seems to be a *tit-for-tat* strategy, as suggested by game theorists, turns out to be a side effect of interactions between agents. A tit-for-tat strategy is one where the individuals keep track of what has happened and only give back as much as they have received (see chapter 6.2). The more parsimonious explanations based on local rules of interaction also obviate explanations resorting to high-level cognition. For example, participants in a conflict are thought to keep track of the number of situations in which they have received help from, and they have given help to another individual. For more detail, the reader is referred to Hemelrijk (1997). Along similar lines, te Boekhorst did a simulation of artificial "orangutans" demonstrating that travel band formation in orangutans can be explained in very simple ways (te Boekhorst and Hogeweg, 1994). This kind of research is predominantly done at the simulation level since often high-level operators like "recognize dominance rank" are used which cannot be translated to real robots in a straightforward manner.

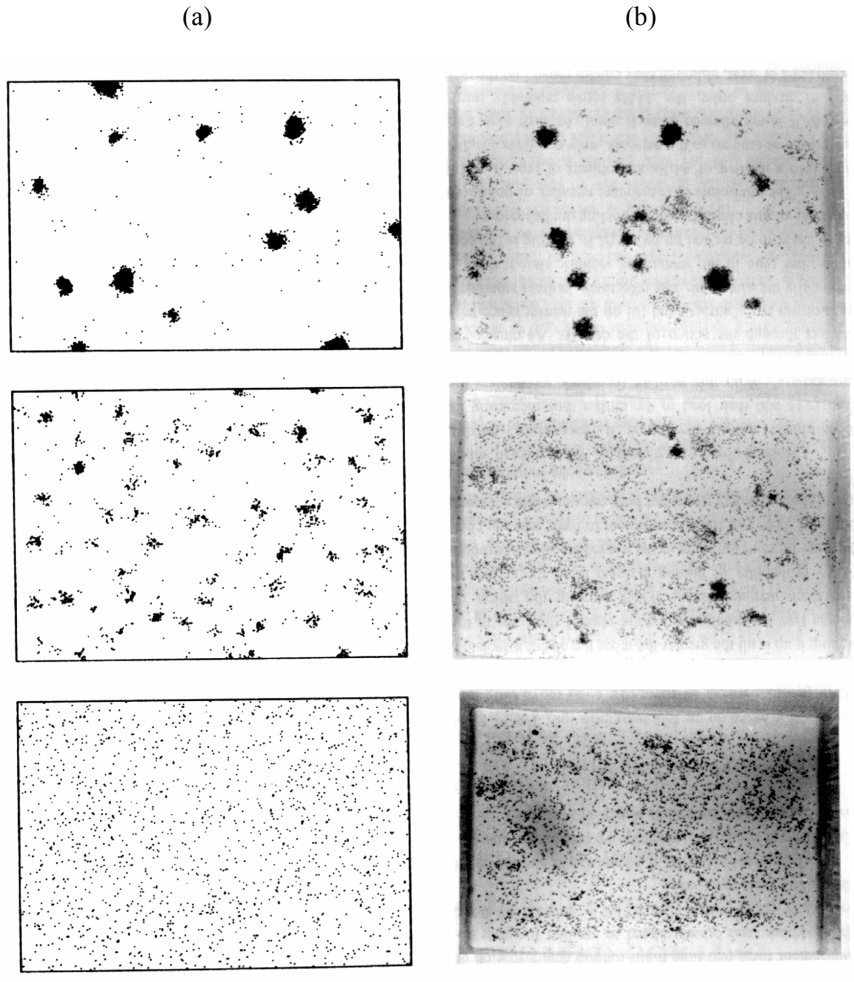

Figure 3.4: *Development of clusters of objects in a society of ants. (a) Simulation. (b) Real ants. The simulation is based on local rules only. The simulated ants can only recognize objects if they are immediately in front of them. If an object is far from other objects, the probability of the ant picking it up is high. If there are other objects present the probability is low. If the ant is carrying an object, the probability of putting it down increases as there are similar objects in its environment. This leads to the clustering behavior shown.*

Ants find their way to a food source

In their experiments on ants, Deneubourg and Goss (1989) tried to find an answer to the question of whether the complexity of social interactions might be attributed to the individuals or to their interactions. For instance, colonies of certain species of ants appeared to be able to select the nearest food source among several that were present at varying distances from the nest. Attributing the complexity of this phenomenon to the individual ants would imply that individual ants compare the distances to several food sources and on the basis of this knowledge choose the nearest food source. This would entail ample cognitive calculations. Instead, Deneubourg and Goss clarified this choice as a consequence of the pheromonal marking and following system of the ants. Ants mark their path with pheromone when they leave the nest to search for food as well as on their journey back to the nest. At crossings where several paths intersect, they choose the most heavily marked direction. Ants return sooner from nearer food sources and as a consequence, shorter paths are marked more intensively than those leading to sources further away. Self-reinforced differentiation of the degree of marking of a path is also called an *autocatalytic* process, a particular instantiation of self-organization (remember the autocatalytic processes in pattern formation in seashells). Such autocatalytic processes have been invoked to elucidate several other aspects of insect behavior as well, e.g., the observed strict spatial distribution of honey, pollen and youngsters in the comb of bees (see Wehner, 1998, which is referenced above) and the way a comb is built. In all cases autocatalytic effects form an alternative view to the idea that patterns are controlled centrally by a blueprint or higher cognitive capabilities of the individuals.

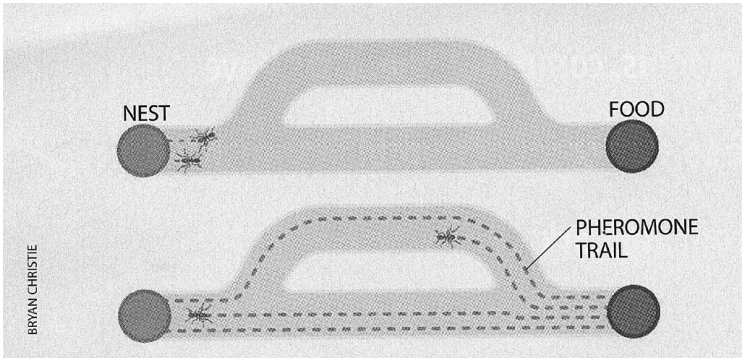

Figure 3.5*: Pheromone trails enable ants to search for food efficiently: Two ants leave the nest at the same time (top), each taking a different path and marking it with pheromone. The ant that took the shorter path returns first (bottom). Because this trail is now marked with twice as much pheromone, it will attract other ants more than the longer route will.*

3.3 The simulation of distributed systems: NetLogo

NetLogo grew out of MIT's StarLogo simulator. Both are based on the programming language Logo, which was originally developed for preschool children. Logo was simple enough in order that children could easily program simple robots, known as "turtles". Initially, the turtles were real robots capable of moving around on a flat surface. They were equipped with a pen, which could be in one of two positions, either up, or down. When it was down the turtle drew a line on the ground while moving. The Logo turtles were designed so that children could playfully and intuitively learn concepts from geometry, mathematics, and

engineering that are often considered hard to understand (see Papert, 1980). For example they could explore ideas of feedback, geometric figures like polygons and circles, and infinitesimality. Later, the turtles were "virtualized" and became a means for drawing on a computer screen rather than on the physical ground (see also the "turtle graphics" discussed in Section 2.3 in the context of Lindenmayer systems). The commands that could be given to the turtles, like move forward, turn, pen-down remained the same.

NetLogo is an extension of the traditional Logo language. First, NetLogo allows for an unlimited number of turtles, where Logo only has one. In other words, it is a massively parallel simulation environment. Second, in NetLogo turtles are equipped with sensors. While traditional Logo turtles were mainly used for drawing purposes, NetLogo turtles have to behave in ways that strongly depend on their environment, in particular their local environment. The behavioral rules, which can be defined for the turtles, make it possible to model true turtle-environment interactions. Third, NetLogo offers the possibility to define so-called *patches*, which can be used to model the properties of the environment. For example, pheromones deposited by the ants will automatically evaporate, or food that is eaten by the turtles grows back at a particular rate. The patches are similar to the CAs, which were introduced in Chapter 2. NetLogo provides a simple yet powerful scripting language to control the turtles and the patches.. We will see precisely the same type of distinction later on, when we will discuss the Sugerscape model (Section 5.1).

NetLogo can be downloaded from here: http://ccl.northwestern.edu/netlogo. The manual contains a very nice and easily digestible tutorial, which is all you need to get started on complex simulations.

3.4 Flocking — the BOIDS

The *boids* are among the most famous creatures in field of artificial life. They were invented in the mid-80s by the computer animator Craig Reynolds. In Culver City in California where he lived, he would observe flocks of blackbirds. He wondered how he could get virtual creatures to flock in similar ways. His hypothesis was that simple rules were responsible for this behavior. It was clear to him that the boids would have to be agents: they would have to be situated, viewing the world from their own perspective, rather than from a global one. Their behavior is controlled by a certain number of local rules. He came up with the following set (Reynolds, 1987):

(1) *Collision avoidance:* avoid collision with nearby flockmates

(2) *Velocity matching:* attempt to match velocity with nearby flockmates

(3) *Flock centering:* attempt to stay close to nearby flockmates.

The first rule, collision avoidance defines the tendency to steer away from an imminent impact. Static collision avoidance is based on the relative position of the flockmates and ignores their velocity. Conversely, velocity matching, set out in the second rules, is based only on speed. The third rule is about flock centering. It makes a boid want to be near the center of the flock. Because of the situated perspective of the boid, "center of the flock" means the perceived center of gravity of the nearby flockmates. If the boid is already well within the flock, the perceived center of gravity is already at it's position, so there is no pull further towards the center. However, if the boid is at the periphery, flock centering will cause it to deflect

somewhat from its path towards the center. Together, these three rules lead to surprisingly realistic flocking behavior (Figure 3.6).

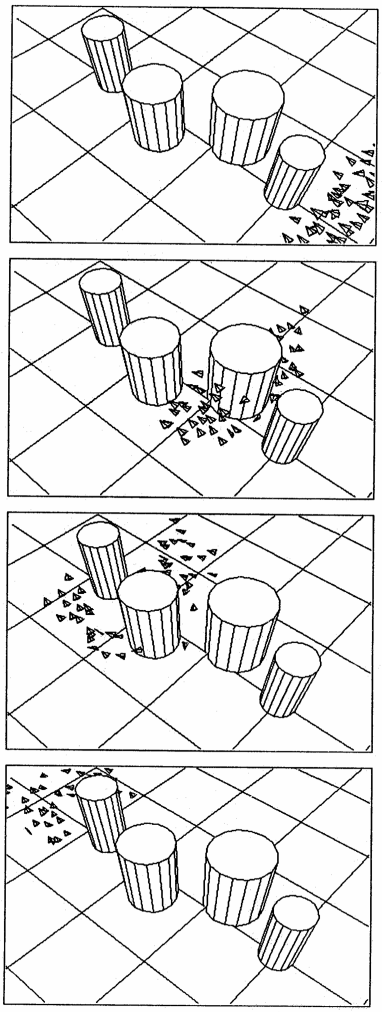

Figure 3.6: *Craig Reynolds's "boids" engaged in flocking behavior. They encounter a cluster of pillars. Amazingly enough, the flock simply splits and rejoins after it has passed the pillars. Note that "splitting" is not contained in the set of rules. It is truly emergent: the result of several parallel processes while the boids are interacting with their environment. Even Reynolds was surprised by this remarkable and beautiful (although fully explainable) behavior.*

Reynolds was interested in what would happen when the flock encountered obstacles. Would the boids continue to flock? Would they all move past the obstacle on one side? Or would they split? The latter happened (as can be observed in nature). Note that "splitting" was not programmed into the boids. Both, the flocking behavior and the splitting behavior are truly emergent phenomena. What is there, is just a number of internal parallel processes (obstacle avoidance, velocity matching, and flock centering), which are based on the boids' situated view of the environment. The flocking behavior is very robust though. This is due to the local distributed nature of the mechanism. Another wonderful example of how sophisticated behavior emerges from simple rules.

Researchers in artificial life claim that their creatures are behaving creatures in their own right. Boids are digital creatures as such — they are not only models of real birds. "Flocking in boids is true flocking, and

may be counted as another empirical data point in the study of flocking behavior in general, right up there with flocks of geese and flocks of starlings." (Langton, 1989, p. 33).

Rodney Brooks, Pattie Maes, Maja Mataric, and Grinell Moore used rules almost exactly like Reynolds to achieve flocking behavior in real robots. At an IROS conference in 1990 (International Conference on Intelligent Robots and Systems), they suggested to use a swarm of robots to prepare the lunar surface for a manned mission. Maja Mataric implemented flocking on her robots using a variation of Reynolds' rules. Her robots, like the boids, exhibit robust flocking behavior. Again, flocking is emergent from local rules. Let us finish with a quote by Dan Dennett, a champion of the philosophy of mind: "Maja's robots are flocking, but that's not what they think they are doing." (Dennett, 1997, p.251) What they think they are doing is applying Reynolds' rules. This is another example of the notorious frame-of-reference issue. For those interested in collective robotics, Maja Mataric has investigated the field for many years. A thorough review would be beyond the scope of this chapter. The interested reader is referred to some of the review papers (Mataric, 1995, 1997).

As seen above there are two possibilities to reach a given behavior, either one simulates such behavior on a computer as Reynolds did with the Boids or, one builds robots which have to be able to behave in the given way in the real world, i.e., exhibit robust flocking behavior as Mataric did.

In general it is easier to simulate a behavior on a computer than to build robots and make them behave accordingly. In a simulation not only the agents but also the environment are predefined by the designer consequently the agents are familiar with the environment. Building robots, which are able to show a certain behavior in the real world, requires the designer to build the robots so that they can adapt to different environments.

Let us now look at rule number two "velocity matching" as an example. It is relatively easy to simulate flocking on a computer by designing agents, that are able to distinguish other agents from different objects in their environment, which always know where other agents are, and in which direction and at which velocity the other agents are moving. Building robots makes one realize that due to the limited technical means available (sensor, motor and computing technology), the aforementioned tasks are very difficult to achieve. One solution to this problem is to redesign the rules used in the simulation. By altering rule 2, in order that the robots only need to know the direction in which other robots are moving, one circumvents the difficult task to regularly check place and distance of all other robots. Such a simplified rule resulting from the difficult implementation of a behavior in the real world can then be transferred back into the computer simulation.

Online resources are found at:

- www.cs.toronto.edu/~dt/siggraph97-course/cwr87/ (by Craig Reynolds)
- www.cse.unsw.edu.au/~conradp/java/Boids/example.html (a very nice Java applet)

3.5 Guiding heuristics for decentralized thinking

To conclude our discussion of distributed intelligence, and in order to make decentralized phenomena easier to grasp, this section provides a brief summary of the major points discussed in this chapter (adapted from Resnick, 1997).

(i) Positive feedback is not always negative

Positive feedback has a "negative image" problem. Whereas people see negative feedback as something, which helps to keep things under control, positive feedback is often seen as destructive. Below some examples of "negative" effects of positive feedback:

— Screeching sounds that result when a microphone is placed near to a speaker

— Population growth

— A vicious circle (German: Teufelskreis) is based on positive feedback. For example, reduction of service in public transportation, leads to such a vicious circle: Reduction in service leads to frustration of users, leading to a decreases of users, which in turn necessitates further reduction of services, which ...

However, positive feedback is not always negative, but can also have positive effects. It can help create and expand patterns and structures underlying a large number of phenomena in nature and society.

Some positive examples are:

— The emergence of Silicon Valley as an example of the geographic distribution of cities and industries. After a few high-technology companies had started, the required infrastructure in that area developed, and this made even more high-tech companies move to the region leading to even better infrastructure. This started a snowball effect.

— Winning standards/operating systems → "Macintosh problem". Due to good marketing strategies, availability and user friendly software, Windows became popular. Its popularity lead even more user to make use of Windows, with even more software, and turned it into the most successful software of all times. Of course, this development is seen as negative by many people.

— The formation of ant trails is influenced by the concentration of pheromone on the trail, the more pheromone the more ants take the respective trail (section 3.2).

— In the robot-clustering task (section 3.1) the positive feedback is given by the fact that the more cubes there are in a cluster, the higher is the probability, that yet another cube will be deposited nearby.

Often there is an interaction of positive and negative feedback. An instance of this interaction is the evaporation of pheromones in the formation of ant trails. As long as there is enough food in a given location most ants take the shortest trail to reach this source and the concentration of pheromones on this trail is constantly high, when the food source is exploited less ants choose the respective trail and the pheromone starts to evaporate which makes the ants switch to a different trail.

(ii) Randomness can help create order

Randomness is not just "disorder". Random perturbations – often created by a system itself - can have an important role in self-organizing systems being the "seed" needed to start the formation of patterns or structures:

— Traffic jams: As long as cars are distributed evenly on a road and all cars are driving at the same speed no traffic jams are formed. But even small fluctuations in traffic density and slightly different velocities of the cars can serve as "seed" for traffic jams; positive feedback then accentuates these density fluctuations, making the seed grow into full-fledged traffic jams;

— Randomness is required to achieve adaptivity in pheromone trails. If the ant societies are to remain adaptive, there must always be a random component in the ant's choice behavior (see the shortcut problem, Chapter 4). This is important if this idea is to be applied to ant-based control, i.e., to message routing in telecommunication networks (Chapter 4).

— Randomness together with positive feedback leads to phenomena like rhythmic clapping of an audience. Initially most clapping is unsynchronized but if accidentally some people are clapping simultaneously this often leads - accentuated by positive feedback - to rhythmic clapping.

(iii) A flock is not a big bird

One important and critical point in describing decentralized systems and self-organizing phenomena is the distinction of different levels. Interacting objects at one level lead to new objects in another level, objects in the first level often behave differently from the resulting object at the next level. The different phenomena and the different levels in which such phenomena occur should not be confused.

— People tend to confuse the behaviors of individuals within a group with the behaviors of groups as a whole, e.g., interactions among individual birds give rise to flocks, but the flock does not follow the same rules as its respective members.

— In many cases, the individuals at one level behave differently from objects at another level: watching traffic jams they tend to move backward, even though all of the cars within these jams are moving forward;

— In the Sugerscape model (see Chapter 5), there are diagonal migration patterns even though the agents can only move up/down and left/right;

— The "leader" in a flock is always changing, but the flock as a group remains and does not change its behavior.

(iv) A traffic jam is not just a collection of cars

Beside the confusion about different levels, confusion might also arise based on the definition of objects. Although an object often consists of different individual parts, looking at it only as collection of parts might lead to misunderstandings. It is important to realize that some objects ("emergent objects" resulting from the interaction of lower-level objects) have an ever-changing composition, but as a whole form the same object, with its own set of rules.

— Over an individual's lifetime, old cells die and new cells are born, but the individual remains the same.

— Ants that make up an "ant bridge" change continuously.

— Water making up the shape of a fountain is always different, but the shape is preserved.

(v) The hills are alive (the environment has an independent dynamics)

People often focus on the behaviors of individual objects, overlooking the influence of the environment that surrounds these objects and interacts with them. Especially in decentralized and self-organizing systems it is important to take the role of the environment into consideration since the same behavior of the individual objects leads to different patterns in different environments.

— The heap building process of the Didabots (Section 3.1) can be influenced by the way the cubes are distributed at the beginning of the experiment (randomly but evenly distributed or in clusters);

— The path taken by ants from their nest to a food source is influenced not only by the capacity of the ants, but also by the complexity of the surroundings, i.e., the environment.

3.6 Conclusion

In this chapter we looked at some examples of distributed intelligence. We saw that behavioral patterns emerge from the interaction of individuals, which are equipped with simple local rules, and which are organized without a central organizer or coordinator. This process of self-organization can be observed in robots (heap building process of the Didabots), in computer simulations (flocking of the boids), and in natural agents (organization of nests in insect societies). The concept of self-organization has also been taken up by modern economists, which grow artificial societies in order to explain market phenomena. NetLogo represents a massively parallel computer language, which is particularly well suited for simulating decentralized self-organizing systems and system-environment interactions. Finally we outlined guiding heuristics for decentralized thinking and at the same time tried to prevent some misunderstanding that often appear when looking at decentralized phenomena and self-organizing systems.

Bibliography

Ball, P. (2004). *Critical mass*. Random House Group Limited: London, Sydney.

te Boekhorst, I.J.A., and Hogeweg, P. (1994). Effects of tree size on travelband formation in orangutans: data analysis suggested by model study. In R. Brooks, and P. Maes (eds.): *Artificial Life IV Proceedings*. Cambridge, Mass.: MIT Press, 119-129.

Deneubourg, J. L. and Goss, S. (1989). Collective patterns and decision-making. *Ethology, ecology and Evolution,* 1:295-311.

Dennett, D.C. (1997). Cog as a thought experiment. *Robotics and Autonomous Systems*, 20:251-256.

Hemelrijk, C. K., (1997). Co-operation without games, genes or cognition. In P. Husbands and I. Harvey (eds.) Fourth *European Conference on Artificial Life.* Cambridge, MA: MIT Press, 511-520.

Langton, C. (1989). Artificial Life. In Ch. Langton (ed.). *The proceedings of an interdisciplinary workshop on the synthesis and simulation of living systems.* Redwood City: Addison-Wesley.

Maris, M., and te Boekhorst, R. (1996). Exploiting physical constraints: heap formation through behavioral error in a group of robots. In *Proc. IROS'96, IEEE/RSJ International Conference on Intelligent Robots and Systems.*

Mataric, M. (1995). Designing and understanding adaptive group behavior. *Adaptive Behavior*, 4:51-80.

Mataric, M. (1997). Learning social behaviors. In R. Pfeifer, and R. Brooks (eds.). *Robotics and Autonomous Systems*, Special Issue on "Practice and Future of Autonomous Agents", 191-204.

Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. New York: Basic Books.

Resnick, M. (1997). Turtles, termites, and traffic jams. Explorations in massively parallel microworlds. Cambridge, Mass.: MIT Press.

Reynolds, C.W. (1987). Flocks, herds, and schools: a distributed behavioral model. *Computer Graphics*, 21:25-34.

Wehner, R. (1998). Selbstorganisation im Superorganismus. Kollektive Intelligenz sozialer Insekten. NZZ Forschung und Technik, 14. Januar 1998, 61.