Nikolay Popov

Paper: Exploring Architectural Possibilities with Flocking Algorithms.

Topics: Architecture

Author:
Nikolay Popov
BLA, MLA [Distinction]
Lecturer
Unitec Institute of Technology
Department of Landscape Architecture
Auckland, New Zealand
www.unitec.ac.nz

Abstract:
Complexity theory offers a new way of understanding spatial patterns as self-organising morphologies. This provides a promising paradigm for exploring spatial organizations as the emergent outcome of dynamic relations between simple elements bounded together by multiple feedback loops. Self-organising spatial morphologies can be defined as a part of a process, usually a simple one, and modelled employing iterative algorithms.

This paper reports on how various versions of the canonical flocking [1] algorithm can be utilized to interactively evolve emergent spatial patterns. The reason for selecting flocks as a study area is the fascinating asymmetry between the simplicity of the rules and the spatial complexity of the outcomes, when observed from a synoptic viewpoint. The flocks are modelled as Agent Based Systems using Netlogo [2] language. Together with traditional behaviours (separate, align, and cohere) the models employ up to five additional rules and a variety of parameters. The focus of the models range from obstacle avoidance, to learning and evolutionary flocking. The aim of the research is to investigate how complex architectural possibilities can be generated bottom-up, using distributed representation.

Theoretically the research is related to the work of Paul Coates [3], Sebastian von Mammen**,** Aaron Westre, James Macgill, and many others.

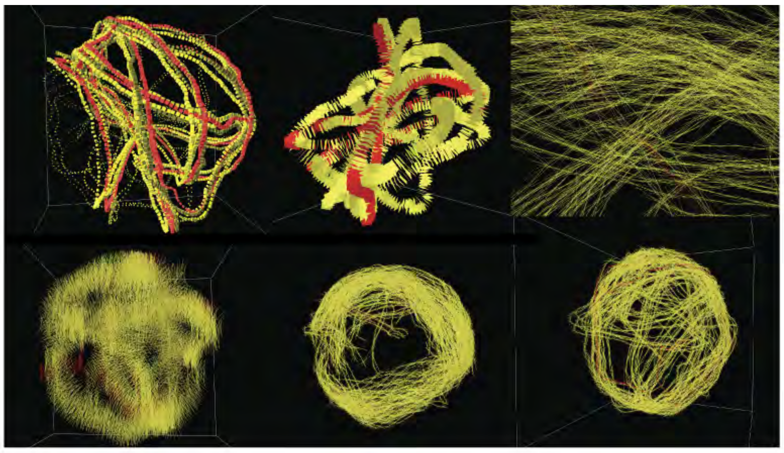


Image. Some initial outputs of flocking models.

References:
[1] Craig Reynolds, "*Flocks, Herds, and Schools"*, Paper presented at the ACM SIGGRAPH, New York, 1987
[2] Uri Wilensky *"Netlogo",* Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, USA, 1999.
[3] Paul Coates, "*Programming.Architecture"*, Routledge, London, 2010.

Contact:
npopov@unitec.ac.nz

Keywords:
Architectural Possibilities, Agent Based Modelling, Flocks, Distributed Representation, Complexity, Emergence.

Exploring Architectural Possibilities with Flocking Algorithms

Nikolay Popov BLA, MLA [Distinction].
Department of Landscape Architecture, Unitec Institute of Technology, Auckland, New Zealand
www.unitec.ac.nz
e-mail: npopov@unitec.ac.nz

Premise

Abstract:
Complexity theory offers a new way of understanding spatial patterns as self-organising morphologies. This provides a promising paradigm for exploring spatial organizations as the emergent outcome of dynamic relations between simple elements bounded together by multiple feedback loops. Self-organising spatial morphologies can be defined as a part of a process, usually a simple one, and modelled employing iterative algorithms.

This paper reports on how various versions of the canonical flocking algorithm can be utilized to interactively evolve emergent spatial patterns. The reason for selecting flocks as a study area is the fascinating asymmetry between the simplicity of the rules and the spatial complexity of the outcomes, when observed from a synoptic viewpoint. The flocks are modelled as Agent Based Systems using Netlogo language. Together with traditional behaviours (separate, align, and cohere) the models employ up to five additional rules and a variety of parameters. The focus of the models ranges from obstacle avoidance, to learning and evolutionary flocking. The aim of the research is to investigate how complex architectural possibilities can be generated bottom-up, using distributed representation.

1. Introduction

Complexity Theory offers a new epistemology, i.e. a new way of understanding and knowing patterns and structures that we observe in the world, as the emergent properties of iterative parallel actions of very simple autonomous processes. The complex outcome emerges entirely bottom-up from the multiple interactions between these events.

Artificial Life (AL) [1], on the other hand, is a computational approach based on distributed representation that attempts to model the above principles and to employ them in a wide variety of applications. It gives us computational frameworks like Genetic Algorithms, Cellular Automata, and Agent Based Systems.

The theory of Autopoesis pioneered by Maturana [2] occupies a central place in AL. He introduced the concept of structural coupling and based his philosophy on the notion of cognition, which he saw as a natural effect of being embodied in the world. Structural coupling describes dynamic mutual co-adaptation without allusion to a transfer of some ephemeral force or information across the boundaries of the engaged systems [1]. There are two types of structural coupling - system coupling with its own environment and system coupling with another system. These systems allow scientists and designers to explore space, cognition and intelligence by building simple feedback systems between agents and their environments.

This paper reports on initial results of research which set out to explore the potentials of flocking algorithms for generating architectural possibilities. The flocks are modelled as Agent Based Systems using Netlogo [3] computer language, outlined in section 2, and some of the results are exported and rendered in Rhino. The focus is on variations and alternatives of the canonical flocking algorithm, as proposed by Reynolds [4], which are grouped in the following categories: interactions between agents and interactions between agents and their environment, explained in sections 3 and 4. The paper concludes with a discussion of the advantages, drawbacks and limitations of the models with respect to their applications in architecture.

2. Agent Based Modelling and Netlogo

Agent Based Modelling (ABM) has its origins in Object-Oriented Programming, Artificial Life, and is paradigmatically supported by contemporary Complexity Theory. Agent Based Models consist of agents, autonomous little computers that operate within environments to which they are uniquely adapted. Agents interact with each other and with their environments according to their strategies in a parallel and repetitive manner. The agents' strategies are driven by transition, typically entirely local rules. Any number of rules can be devised to govern the activities of agents, such as the goals that agents seek to satisfy, or the 'preferences' that agents might have. Agents also have a set of attributes, or states, that describe their characteristics. True mobility and dynamism are inherent qualities in these types of models. It is important to note that in order to have a bottom-up, self-organising outcome the transitional rules need to be local and not overly specified.

ABM offers the possibility to study configurations of space as self-organising phenomena. As Coates [1]states:

This approach seems to provide a nice paradigm for architecture as the emergent outcome of a whole lot of interconnected feedback loops, which replace the top-down geometry and the reductionist tradition with dynamic relations and emergent outcomes not defined in the underlying model.

Netlogo is an agent based programming language and modelling environment developed as a parallel computation machine which provides a powerful experimental tool for exploring and demonstrating the effects of massively parallel populations of interacting agents in biology, physics, geometry, social systems and ecologies. It is particularly well suited to modelling complex systems evolving over time. Modellers can give instructions to hundreds or thousands of independent ‘agents’ all operating concurrently. Agents are autonomous beings that can follow instructions. All of the agents can carry out their own activity simultaneously. In Netlogo, there are four types of agents: turtles, patches, links, and the observer. Turtles are agents that move around in the world. The world can be two dimensional or three dimensional and consists of grids of patches. Each patch is a square or cubic piece of space over which turtles can navigate. Patches cannot move, but otherwise they're just as "alive" as turtles and the observer are. Links are agents that connect two turtles. The observer does not have a location, it is looking over the world of turtles and patches.

3. Flocking 1 (interactions between agents)

This section presents variations and alternatives based on the Reynolds’ flocking algorithm. The rules governing the flock model, introduced by him are explained; then the modified behavioural rules of the swarm agents are illustrated. They add richness and variability to the performance of the flocks and yield visually interesting and dramatic results.

3.1 The Reynolds’ flocking algorithm

The flocking model was originally proposed by Reynolds [4] in 1987 as an algorithm for computer simulation of the flocking behaviour of birds, both for animation purposes and as a way of studying emergent behaviour and, since then, has been employed in a wide variety of applications. Researchers from diverse fields have employed and interpreted flocking behaviours in their attempts to study an impressive variety of phenomena. Flocks have been made to represent virtually every system of flow - from pedestrian, crowd dynamics and traffic to movement of animals and distribution of plant species in ecosystems [5]. Flocking algorithms have been used as search and optimization mechanisms in metric [6] and n-dimensional phase-spaces [7]. Architectural interpretations of flocking algorithms have been studied by Paul Coates [7], Aaron Lane Westre [8], and Olga Linardou [9] to name just a few.

Flocking is an example of emergent collective intelligence based on local and simple stimulus-reaction rules. There is no leader, i.e. no global control. The overall pattern emerges from the local interactions. Each agent has direct access to the geometric description of the whole world, but reacts only to flock mates within a certain field of view. The basic flocking model consists of three kinds of simple steering behaviours:

Separation gives an agent the ability to maintain a certain distance from others nearby. This prevents agents from crowding too closely together, allowing them to scan a wider area.

Cohesion supplies an agent with the ability to cohere (approach and form a group) with other nearby agents. Steering for cohesion can be computed by finding all agents in the local neighborhood and computing the average position of the nearby agents. The steering force is then applied in the direction of that average position.

Alignment gives an agent the ability to align with other nearby characters. Steering for alignment can be computed by finding all agents in the local neighborhood and averaging together the 'heading' vectors of the nearby agents.

All the agents or turtles implement the navigation rules simultaneously and iteratively in the following order [10]:

```
to flock
  find-flockmates ; each agent finds other agents that are within its cone of vision
  if any? flockmates
    [ find-nearest-neighbor; each agent finds the closest agent among its flock mates
       ifelse distance nearest-neighbor < minimum-separation ; when two birds are too close, the "separation" rule overrides the other two, which are deactivated until the minimum separation is achieved.
        [ separate ]
        [ align
          cohere ] ]
end (Figure 1)
```

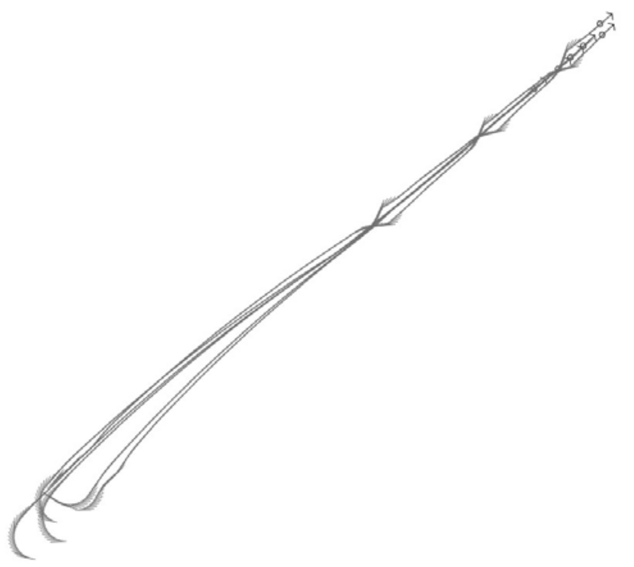


Figure 1 Steering trajectories with constant speed

In this version of the model the rules affect only the birds' headings and all the agents move at a constant speed. The steering behaviours are also entirely deterministic, i.e. apart from the initial positions of the agents there are no random numbers in the algorithm. Provided the deterministic nature of the rules the flock formation is still dynamic, once a flock is together there is no guarantee that it will keep all of its members. There are seven parameters that can be adjusted by the user in this model: *population, vision-distance, vision-angle, minimum-separation, max-separate-turn, max-align-turn, and max-cohere-turn*. By adjusting the sliders numerous behaviours can be achieved - tighter flocks, looser flocks, fewer flocks, more flocks, more or less splitting and joining of flocks, more or less rearranging of birds within flocks, etc.

In order to achieve even more diverse collective behaviours the following rules were added:

Acceleration: Agents that are not part of a flock can fly faster to catch up, while birds that have flock mates adjust their velocity to the average velocity of their flock mates and, as a result, the whole flock slows down.

Chase, Avoid and Eat rules: These rules were developed in order to further diversify the simulation. There are two types of birds: predator and prey. The predators can diverge from their flock and chase the closest prey. Prey agents on the other hand try to avoid any predators in their field of view. (Figure 2)

Obstacle avoidance rule: This is very similar to the separation rule. The agents detect the closest obstacle in their 'cone of vision' and determine the distance between themselves and the obstacle. They then turn left or right in such a way that, as they get nearer to the obstacle, they veer away from it, retaining their smooth movements. (Figure 3)

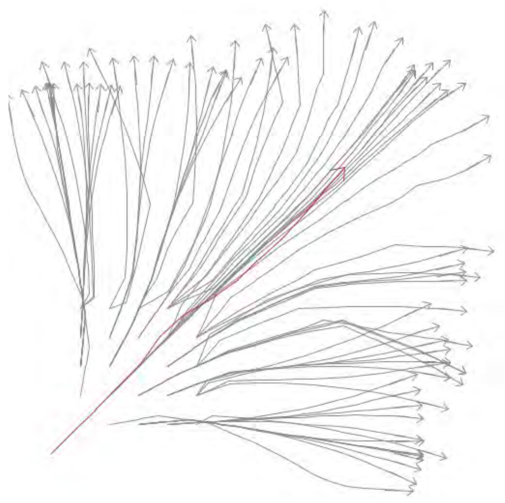


Figure 2 Steering trajectories generated from separate align, cohere rules combined with acceleration, chase and avoid rules

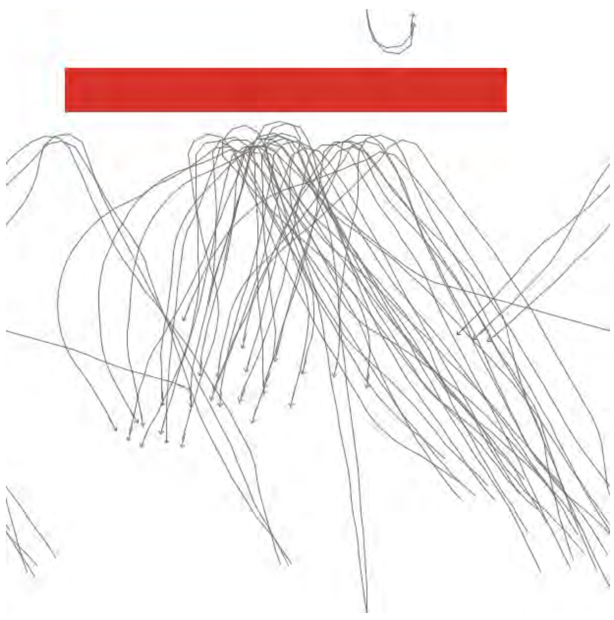


Figure 3 Steering trajectories generated from separate align, cohere rules combined with obstacle avoidance rule

3.2 V-shaped flocks

An intriguing alternative to Reynolds' flocking algorithm was introduced by Nathan and Barbosa [11] in their study of the emergence of V-like formations during flight of migratory birds. Apart from having an unobstructed view in the direction of their flight, for relatively large birds the aerodynamics of flight seems to be the main reason for the formation of V-shaped flocks. The essence of the aerodynamics is that each flying individual creates an upwash region behind it, just off the tips of its wings, such that another individual benefits greatly (in terms of requiring less exertion during flight) if it places one of its wings in that region. The rules again are entirely local and are executed in parallel manner. They are the following [12]:

If a bird cannot see any other birds in its cone of vision, it will continue to fly straight at its normal base speed. Otherwise, each bird follows four rules, given in this order of priority.

1 If a bird is further than the distance for getting an updraft benefit from the nearest visible bird, it will turn toward that bird and speed up to get near it.
2 Once a bird is near enough to another bird it will move randomly to one side or another until its view is no longer obstructed.
3 If a bird gets too close to another bird it will slow down.
4 Once the three conditions above are met the bird will set both its speed and its heading to that of its closest visible neighbour. (Figure 4)

The behaviour rules from both models can be interwoven in a limitless variety of ways. Agents can also die, reproduce and change their headings randomly. All the rules were made probabilistic, i.e. the user can adjust the probability, expressed as a percentage, that the agent will follow certain rules. Birds also keep a record of their trajectories and the previous values of their variables

By altering rules and adjusting probabilities, a wide range of behaviours can be studied. Because of the stochastic nature of the rules, and the non linear relationships between the rules and the resultant behaviours, new and entirely artificial configurations are continuously invented through the set up of simulations.

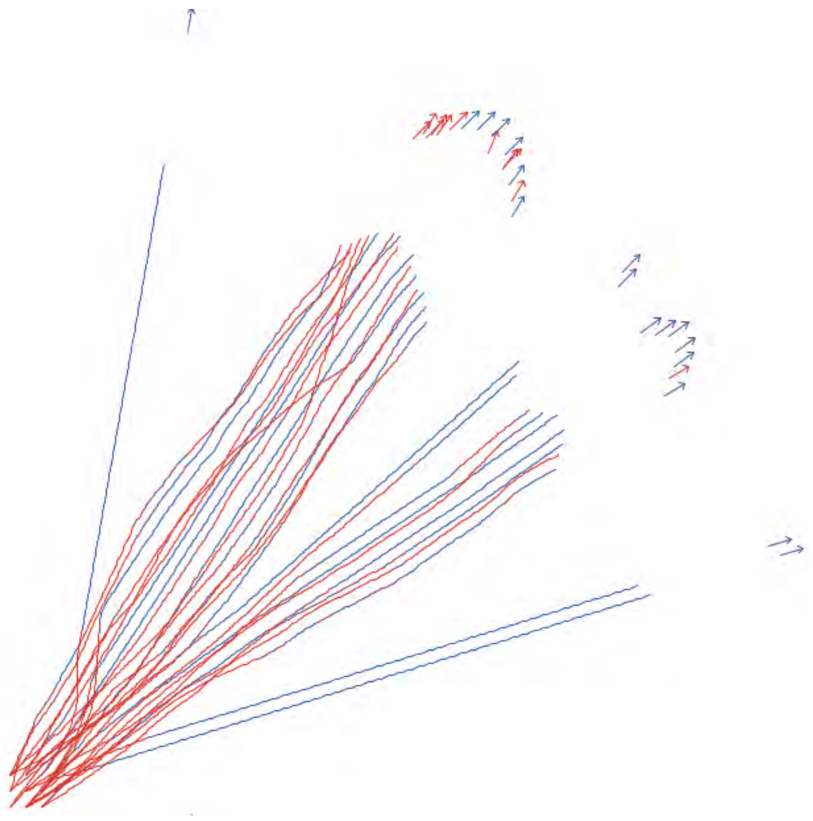
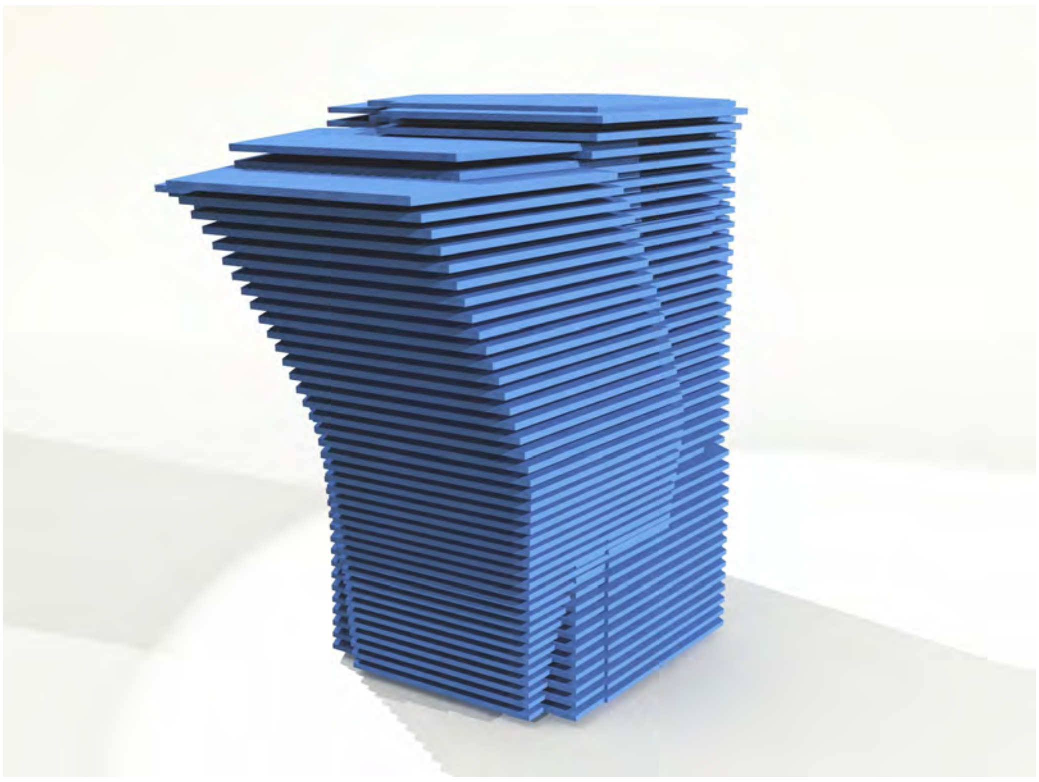
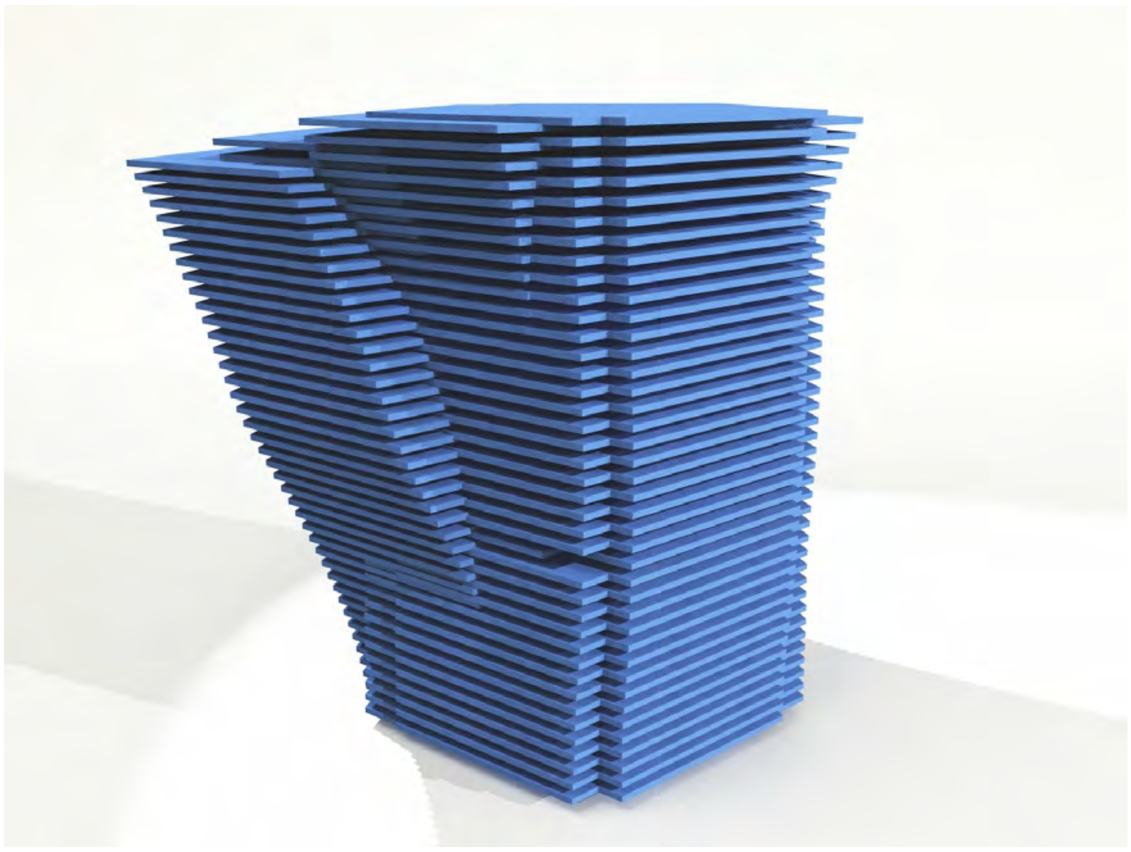


Figure 4 V-shaped flocks. The red colour indicates unhappy boids, i.e. they either do not benefit from the upwash or have an obstructed view

Agents, and their trajectories through space, are simply collections of points and the values of agent's variables in the models. These collections form three dimensional point clouds with lists of data, representing agent variables attached to them. These initial datasets can be seen as frameworks over which a wide variety of geometries can be mapped. For each flock, shapes can be specified for both the location of the agents and for the paths those agents trace through space. These shapes can be transformed utilizing the agent's variables like heading vector, velocity, and acceleration, number of flock mates, etc. Loft surfaces can be wrapped using birds' trajectories. (Figure 5)

Regardless of the geometric interpretations of the flocks the main question is what model setting can produce well-formed spatial configurations that can be further developed into architectural designs? It is important to note that even the most basic simulation – deterministically applying just the three fundamental flocking rules (separate, align, cohere), with the same amount of birds and same initial positions, the total size of the parameter space (all possible combinations of parameter settings) is incredibly large - 41x360x21x81x81x81 = 164 725 452 360. This issue requires further research and is partially addressed in the 'Evolutionary Flocks' chapter.





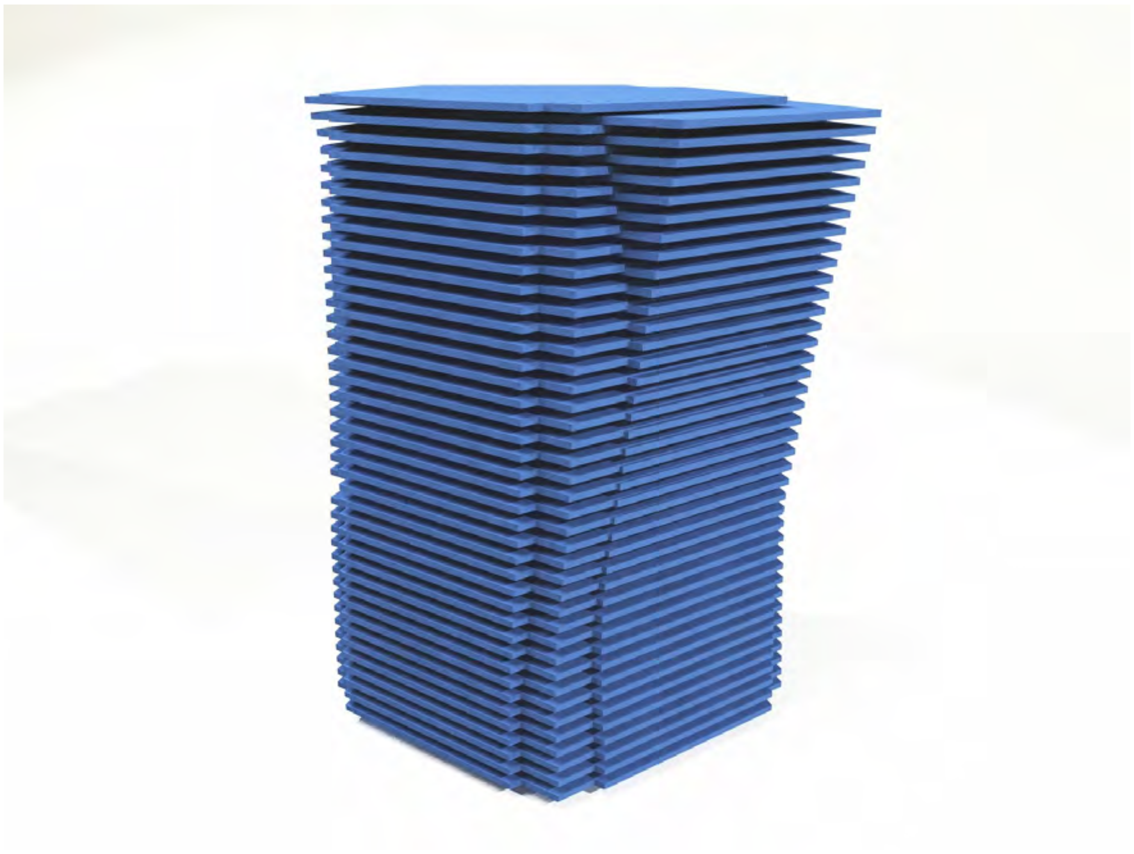


Figure 5 Views of three geometric interpretations of the same model's output

4. Learning (Exploratory) Flock

In 'The use of Flocks to drive a Geographic Analysis Machine', J. Macgill and S. Openshaw [6] study how the emergent flocking behaviour might be used as an effective search strategy for performing exploratory geographical analysis. Their method relies on the parallel search mechanism of a flock, by which if a member of a flock discovers an interesting area, the mechanics of the flock will attract other members to explore that area in detail.

This technique utilises variable velocities of the agents, with common minimum and maximum for all agents and variable colours for the agents. Both velocity and colour have 'meaning' in regards to the success of an agent in finding an interesting area. The learning flock is governed by the following rule sets:

Every agent assesses its current location and the number of other agents in its cone of vision. After that it changes its colour as described below:

If there are no other agents the agent in question dies.
If there are other agents but nothing interesting in the environment then the agent turns blue.
If there are other agents and some interest in the environment then the agent turns green.
If there are other agents and more interest in the environment then the agent turns red.
If there are other agents and a significant interest in the environment then the agent turns yellow and stops.

Then each agent adjusts its heading according to the following list of instructions:

If the closest neighbour is too close then separate from it regardless of its colour.
If the closest neighbour is green, disregard it.
If the neighbour is red or yellow, feel attracted.
If neighbour is blue, then avoid it.

Every agent takes the weighted average of all target points generated above and moves towards that point with the following velocity rules:

If I'm blue move faster (This area is not interesting).
If I'm red move slower (There is some interest and I don't want to miss anything).
If I'm yellow, don't move.

This means that when agents find something interesting in the environment they will slow down and cluster in order to explore the area in more detail. This happens because their speed is low and they have the inertia to remain there. The agents in the neighbourhood that have not detected anything of interest will speed up and be attracted to heavier and slower agents. The idea is that the information is stored in the velocities of the agents. Speeding up corresponds to 'forgetting' in the system. With this algorithm, the flocks will move around discovering areas of interest. If the area does not have enough weight compared with another, it will not be able to attract enough agents. After iterating the algorithm numerous times, the flock will 'forget' the areas of low interest. The main issue is, once again, the nonlinear relation between the parameter settings of the model and the discoveries made by the birds.

5. Evolutionary Flocks and Behaviour Search

The models discussed in previous chapters collectively proved that the rules and the outcomes are independent to a certain degree, and we can claim that these kinds of models are epistemologically autonomous, i.e. the resultant patterns are not in the algorithm. To compare the rules and the parameters to ‘a genotype’ and the various outcomes of the model to ‘a phenotype’ is a natural extension to this type of logic. Von Mammen [13] coined the term *swarm grammars* in order to describe particular settings of a flocking model, i.e. number of individuals, behavior rules, parameters, etc. He also translated the swarm grammars and the outcomes into biological terms. Swarm grammar configurations represent genotypes and their simulations compute the corresponding architectures, the phenotypes. As indicated earlier the number of possible flock grammars is incredibly vast – the most basic model illustrated in section 4 has nearly eleven million possible grammars. The stochastic nature of the models ensures that running the simulation with the same grammar, or phenotype, will produce a virtually infinite number of phenotypes or architectural possibilities. The configuration of the grammar does not reveal the emergent spatial configuration; the algorithmic game has to be played first.

The combination of these factors makes a full, brute-force exploration of the parameter space infeasible [14]. Designers can respond to this difficulty in variety of ways. Exploring sub-spaces, or the whole space, at low resolution is one of them [14].Varying a single parameter at a time, while keeping the rest constant, is another approach. The models described represent complex systems with non-linear interactions; these methods can neglect grammars, or genotypes, that will output interesting, or unexpected, behaviours from the model.

The question that arises is how to configure grammars that are likely to generate innovative and well formed spatial outcomes from architectural viewpoint? Von Mammen [15] provides the link: using the concept of evolution and genetic algorithms.

BehaviorSearch [16] is an open-source software tool that interfaces with Netlogo and provides several search algorithms that can be used to explore Agent Based Models written in Netlogo. The heuristic techniques to search the parameter-space include uniform random search (RS), a random-mutation hill climber (HC), and a genetic algorithm (GA).

Exploration of a model comprises four steps: definition of the parameter space of the model (varying parameters and allowed parameter ranges), design of a quantitative measure for the behaviour of interest (fitness function), selection of search techniques that optimize the fitness function, running the search and examination of the results.

Experimenting with BehaviorSearch will be the next step of the research. The biggest challenge will be the design of the fitness function. In order for flocking algorithms to generate well-formed configurations of space from an architectural viewpoint, the fitness function needs to encompass a wide range of criteria, some of them of a qualitative nature, while others involve complex quantitative relationships.

5. Conclusion

This paper reported on variations and alternatives to Reynolds' flocking algorithm and their abilities to generate architectural possibilities. The flocks were modelled as Agent Based Systems and theoretically positioned within an Artificial Life paradigm. The models were examined as examples of emergent collective intelligence. The outputs of the models reveal intriguing and unexpected spatial patterns. These patterns are emergent phenomena overall based on local and simple stimulus-reaction rules. The models studied included the canonical flocking algorithm, obstacle avoidance, predator-prey, v-shaped flocks, learning and evolutionary flocks.

The major challenge to architects and designers that want to experiment with flocks in particular or with ABM in general, is the non-linear genotype-phenotype like connection between a model's rules and parameters and the emergent spatial outputs. Searches through the vast parameter-spaces in these models can be a very difficult and challenging task. Research in these areas is still in its infancy. The proposed tool – BehaviorSearch – was released in 2010 and is still a beta-version. As has been explained, to design a quantitative measure for the behaviour of interest (fitness function) for an architectural project is another big challenge.

Despite the challenges and difficulties, ABM definitely has a future in architecture. ABM is one of the few computational frameworks that can be used to study complexities of the structures that we observe in the world.

6. References

1. Coates, P., *Programming.Architecture*. 2010, London: Routledge.
2. Maturana, H., *The organization of the living: A theory of the living organization.* International Journal of Man-Machine Studies, 1975. **7**.
3. Wilensky, U., *Netlogo*. 1999 http://ccl.northwestern.edu/netlogo/., Center for Connected Learning and Computer-Based Modeling, Northwestern University: Evanston, IL, USA.
4. Reynolds, C., *Flocks, Herds, and Schools*, in *ACM SIGGRAPH*. 1987, ACM Press: New York.
5. Arand, B. and c. Lasch, *Pamphlet Architecture 27: Tooling*. 2006, New York: Princeton Architectural Press.
6. MacGill, J. and S. Openshaw 1998, *The Use of Flocks to drive a Geographic Analysis Machine*. (Online paper) http://www.geocomputation.org/1998/24/gc24_01.htm
7. Miranda, P. and P. Coates. *Swarm modelling. The use of Swarm Intelligence to generate architectural form.* In *Generative Art Conference*. 2000. Milan Italy.
8. Westre, A.L., *Complexity Machine 1: Drawing 3D Form with Behavioral Simulation*, in *Graduate School*. 2008, University of Minnesota.
9. Linardou, O., *Towards Homeostatic Architecture: simulation of the generative process of a termite mound construction*, in *Bartlett School of Graduate Studies*. 2008, University College London: London.
10. Wilensky, U., *NetLogo Flocking model* . http://ccl.northwestern.edu/netlogo/models/Flocking. 1998, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.
11. Nathan, A. and B.C. Barbosa, *V-like formations in flocks of artificial birds.*

Artificial Life, 2008. **14**(2): p. 179 - 188.
12. Wilkerson-Jerde, M., F. Stonedahl, and U. Wilensky, *NetLogo Flocking Vee Formations model*. ttp://ccl.northwestern.edu/netlogo/models/FlockingVeeFormations 2009, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.
13. Jacob, C. and S.v. Mammen, *Swarm grammars: growing dynamic structures in 3d agent space.* Digital Creativity: Special issue on Computational Models of Creativity in the Arts, 2007. **18**.
14. Stonedahl, F. and U. Wilensky. *Finding Forms of Flocking: Evolutionary Search in ABM Parameter-Spaces.* in *Proceedings of the MABS workshop at the Ninth International Conference on Autonomous Agents and Multi-Agent Systems*. 2010. Toronto.
15. Mammen, S.v. and C. Jacob. *Swarm-Driven Idea Models — From Insect Nests to Modern Architecture.* in *Eco-Architecture 2008*. 2008. Wessex Institute, Winchester, UK.
16. Stonedahl, F., *BehaviorSearch*. 2010, Center for Connected Learning and Computer Based Modeling, Northwestern University, Evanston, IL.