UNIVERSITY OF CALGARY

Swarm Grammars:

Modeling Computational Development through Highly Dynamic Complex Processes

by

Sebastian von Mammen

A DISSERTATION
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF DOCTOR OF PHILOSOPHYs

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

May, 2009

UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a dissertation entitled "Swarm Grammars: Modeling Computational Development through Highly Dynamic Complex Processes" submitted by Sebastian von Mammen in partial fulfillment of the requirements for the degree of DOCTOR OF PHILOSOPHYs.

Supervisor,
Dr. Christian Jacob
Department of Computer Science

Internal External,
Dr. Ken Fields
Department of Art

Supervisory Committee,
Dr. Jörg Denzinger
Department of Computer Science

External,
Dr. Una-May O'Reilly
Massachusetts Institute of Technology, Boston, USA

Supervisory Committee,
Mr. Gerald Hushlak
Department of Art

Date

Abstract

We have developed *swarm grammars* (SGs) as an integrated representation of artificial swarms and developmental models. SGs have evolved in three steps: *basic swarm grammars* [1–3], *extended swarm grammars* [4, 5] and *swarm graph grammars* (SGGs). In this thesis, we present each of these types of SGs together with their respective motivations. We dedicate chapters to the evolutionary exploration of SGs and their application in interdisciplinary works in the fields of art [6] and architecture [4, 6]. The breeding methods we applied for evolving architectural design required the introduction of complexity measures in SGs. These in turn triggered a study on swarm networks [7] that promoted the development of SGGs. Through SGGs, swarm grammars have matured into a universal, bio-inspired modeling framework for complex developmental systems.

Acknowledgements

Rarely, I have found a person that is similarly motivated for dedication through personal freedom as I am. Luckily, my supervisor Christian Jacob shares this preference and has always granted me more freedom than anyone could have expected—in respect to my research, to needed logistic arrangements and to scientifically motivated travels. At the same time, Christian's praise of my achievements, his kind responsiveness and insightful advice frequently recharged my enthusiasm and optimism. His generosity and providence have been an infinite source of motivation.

Many thanks also go to the other two members of my supervisory committee Jörg Denzinger and Gerald Hushlak. Taking an excellent course on multi-agent systems, I had the opportunity to learn first-hand about Jörg's research which has accompanied my studies ever since. He also greatly supported me with concise and helpful advice for my thesis work and through countless reference letters for various scholarship applications. I know from the ratings alone, that if only his reference letters had been counted, I would have been blessed with a comforting financial cushion throughout my Ph.D. work.

In Jerry I found a patron for my inclination to exercise the visual arts. He opened many doors for me to broaden my horizon and to experience the world from new perspectives. During sizzling discussions about digital creativity, Jerry worked hard to convey limitless creativity and productivity. This spirit nourished me whenever I felt like starving in webs of formalisms that I had trapped myself in.

Una-May O'Reilly and Ken Fields joined the committee for my last defense. Thank you greatly for your efforts, your welcome feedback and your insightful advice.

I also want to thank many other academics who provided invaluable feedback during my dissertation: Julian Togelius, Richard Levy, Peter Deacon, Priscilla Greenwood, Gabriella Kókai, Mary-Ann Hushlak-Fraser, Daniel Röhr, Martin Despang, and Illaria Mazzoleni.

The feeback from members of my research group, the department of Artificial Intelligence and the Evolutionary & Swarm Design lab, deserve special credit. I gained a lot of insight from discussions with Marcin Pilat, Lance Hanlen, Navneet Ballah, Jan-Philipp Steghöfer, Scott Novakowski, Scott Steil, Ryan Moniz, Ian Burleigh and Namrata Khemka.

During the explorative phases of my research, interdisciplinary collaborations gained great importance. I would like to thank Joyce Wong for her continuous dedication to swarm art, Thomas Wissmeier for his inspirational contributions to the *Swarm Momentum* exhibition. Naz Erfan, already an invaluable supporter of the art exhibition, joined the supporters of my architectural endeavors. In this context, John Holash and especially Wayne Long provided great help. In the field of biology, Florian Menzel, Kerstin Fröhle and Guy Theraulaz deserve my gratitude for providing important illustrations and references.

I would also like to thank Jon Klein, creator of the simulation environment *BREVE* [8], for his immediate responses on my inquiries, and to Jürgen Schneider who introduced me to graph transformation systems and offered his graph visualization package for the LaTeX text processing system [9].

Of course, without the corresponding infrastructure, my work would not have been possible. Therefore, I'd like to especially thank Camille Sinanan, Lorraine Storrey and Jennifer Erno of the Department of Computer Science, as well as Sheila Harland, Rick Calkins and Nathan Tremblay of the Department of Art.

The greatest support I have been receiving came from the families Novakowski, Jacob, Hushlak and my own. I consider myself very fortunate to be given your love and support. Without you, my studies would not have prospered so well, or might not have taken place at all. Thank you.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In swarms large numbers of simple units interact in ways that make complex patterns emerge. A swarm is a highly dynamic complex system. It continuously evolves through re-configuration of the relationships among its constituents. This concept is generally referred to as the *swarm metaphor* and it is perceived in numerous modeling examples and on different scales. Instances are simulations of molecular interactions that result in gene regulatory networks [10], cellular interactions in the human immune system [11], traffic systems [12], flocks of birds [13] (Figure 1.1), and nest constructions of social insects [14].

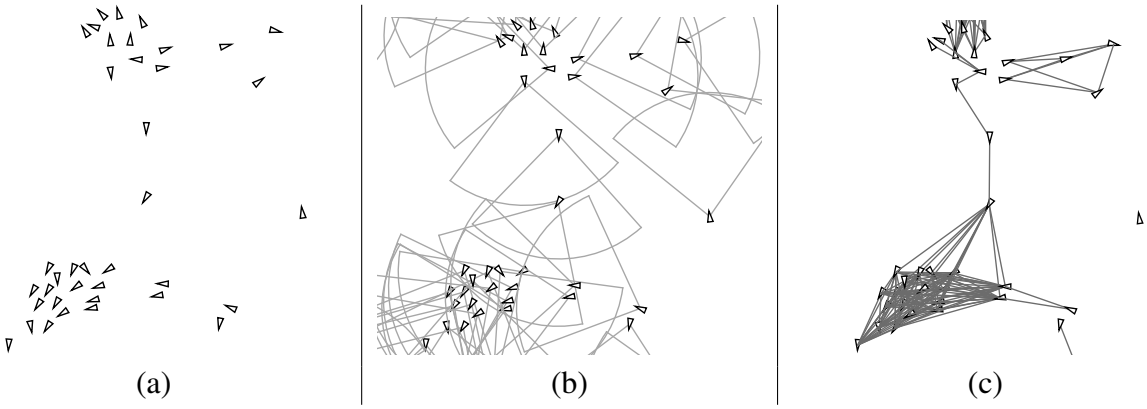


Figure 1.1: (a) A flock of virtual birds, or *boids* [13], in a loose flight formation. (b) The model assumes the illustrated geometric fields of perception of the boids. (c) Edges depict the continuously changing qualitative interaction network among the boids.

The latter example of swarm-built architecture is distinct as the complex interaction patterns of a swarm materialize in a static outcome[1]. It is also of great relevance as the goal-driven manipulation of the environment is an endeavor usually associated with human civilization [15]. However, the insects' approach to construction is very different from ours [16] and opens

[1]We have demonstrated the link between swarm choreography and emerging structures in [1, 5].

up new opportunities for architectural methodologies and designs.

The formulation of a generalized swarm-inspired computational model for construction processes provokes a great challenge: The model might become so expressive, so powerful that it is difficult to control its emergent processes and produced artifacts. However, through analysis of the model's inherent complexity, we are able to harness the creativity of computational evolution and breed novel swarm-driven sculptures and designs. In the long run, numerous functional aspects should be incorporated into the artificial design breeding processes as to break away from virtuality and implement the achieved designs in reality.

From another perspective, such artificial constructive swarm models may be helpful for the systematic study of complex phenomena in general. Swarms bear a high degree of complexity due to their intrinsic neighborhood dynamics. In addition, in constructive swarms a static artifact emerges from complex interaction processes, and thus provides a measure for complexity.

We have developed *swarm grammars* (SGs) as an integrated representation of artificial swarms and developmental models. SGs have evolved in three steps: *basic swarm grammars* [1–3], *extended swarm grammars* [4, 5] and *swarm graph grammars* (SGGs). In this thesis, we present each of these types of SGs together with their respective motivations. We dedicate chapters to the evolutionary exploration of SGs and their application in interdisciplinary works in the fields of art [6] and architecture [4, 6]. The breeding methods we applied for evolving architectural design required the introduction of complexity measures in SGs. These in turn triggered a study on swarm networks [7] that promoted the development of SGGs. Through SGGs, swarm grammars have matured into a universal, bio-inspired modeling framework for complex developmental systems.

The remainder of this thesis is structured as follows. The next chapter introduces related work which is divided into six parts: Nest-building social insects serve as biological inspiration of SGs. Consequently, general aspects of their construction methods and architecture are provided first (Section 2.1). From a computer science point of view, SGs can be classified as

computational developmental model (CDM). Other seminal CDMs are presented in the second part of the chapter on related work (Section 2.2). Section 2.3 is dedicated to the creative potential of CDMs. In this section, we also point out that CDMs and numerical simulations are becoming an integral part of any design work based on the fact that automatically solved design tasks can handle well-defined, computable challenges. The designer faces a shift from hands-on design of a desired artifact toward designing a suitable representation that provides the means to evolve complex solutions. Again, a computer science perspective sheds light on this 'evolutionary' design approach, and several examples are provided (Section 2.4). Although numeric simulations are themselves designed systems, they can exhibit unforeseen, 'emergent' behaviors. Section 2.5 addresses this issue by outlining numerous aspects of complexity research, including the categories, the causes and the measures of complex systems. Finally, Section 2.6 amasses models of artificial swarms which have indeed occurred in all the mentioned contexts. They were inspired by biological swarms, utilized for construction of artificial structures—like CDMs, applied in interactive art installations, subjected to computational evolution and formalized in order to gain insights into complex systems.

Chapter 3 introduces basic and extended SGs and illustrates these concepts with systematic construction experiments and interdisciplinary art works. Chapter 4 presents different ways to breed SGs through evolutionary computation: Interactive evolution allows an external breeder to drive the process solely through manually rating sets of phenotypes. Immersive evolution enables the breeder to actively engage in evolutionary processes and to tinker with SGs like a gardener with his plants. If an SG design task is mathematically specified, an automatic evolutionary process can yield a number of desirable designs and their underlying SG configurations. Chapter 5 details how the latter approach was used to breed architectural idea models with SGs. While working on architectural design generation, we have sharpened the notion of interaction complexity in artificial swarms which resulted in an according study presented in Chapter 6. Eventually, in Chapter 7, we condense previously introduced SG models into

a graph-based representation. The resulting SGG model brings about a powerful, clear and analytically graspable framework for modeling swarm systems. A summary and a chapter on future work conclude this thesis.

Chapter 2

Related work

The work presented in this thesis draws from an interdisciplinary assortment of scientific fields. Natural processes and products have been inspiring the search for fundamental principles in complex developmental systems. Various computational models had already been in place to support the materialization and subsequent evaluation of novel concepts and representations. Abstractions of physical and chemical theories smoothed the way to phenomenal studies in the regime of the complex that proved to become an increasingly important aspect of our investigations. Again, some theories had already been in the works to measure complexity in computational models.

During repeated cycles of exploration and development, we touched upon artificial life [17], bio-complexity [18], ethology [16, 19], developmental biology [20], art [21], architecture and design [22]—academic disciplines whose combinations allowed for productive experimentation with algorithmically phrased models and their iterative improvements.

It is hardly possible to encompass all the relevant works and reference all the seminal literature in the respective areas for huge bookshelves could be dedicated to each of them, still missing out on some important piece or another. Therefore, this chapter guides through (some of) the related disciplines, hints at certain links and provides many details that proved invaluable on our journey of developing swarm grammars as a complex, dynamic, developmental model.

2.1 Social insects, nests and the stigmergic script

Observations of the environment have always had great impact on our (human) ways of life. Yet, there are still innumerable secrets to unearth and outstanding discoveries to make. Nature's creative powers are exemplarily shown[1] in Figure 2.1 which depicts a massive termite mound found in the northern part of the Kakadu national park of Australia's Northern Territory.

Figure 2.1: The picture shows a termite mound in the northern part of the Kakadu national park of Australia's Northern Territory.

Termites and humans alike belong to those 3 to 5% of all animal species that have evolved as social organizations [23]. Due to the synergy of collectives, this small percentage has been

[1]The picture was generously provided by Florian Menzel, Biocenter, University of Würzburg, Germany.

having a major impact on life on earth. This is, for instance, reflected in the homes of social earth-dwellers: They enjoy architectures of ingenuity, complexity and (relative) size unsurpassed by any solitary species.

2.1.1 Collective construction processes

It seems, however, that our methodological approach to architecture deviates drastically. We diligently engineer plans to reach our goals, whereas social insects seem to follow scripts that determine simple stimulus-reaction behaviors [24, 25]. On the one hand, the insects are unaware of their works' contribution to the greater good. On the other hand, they naturally construct intricate nests. They construct efficiently, are well coordinated and utilize the massive power of large numbers of individuals that build in parallel [14, 26]. Honey bees, for instance, often initiate the construction of three cells all at once. After about half a minute of construction, they are relieved by co-workers that immediately recognize the state of the construction and take the required next steps [19]. The community effort of nest construction happens in several phases. In each phase a certain subshape is built which in turn determines a module for the next phase (Figure 2.2).

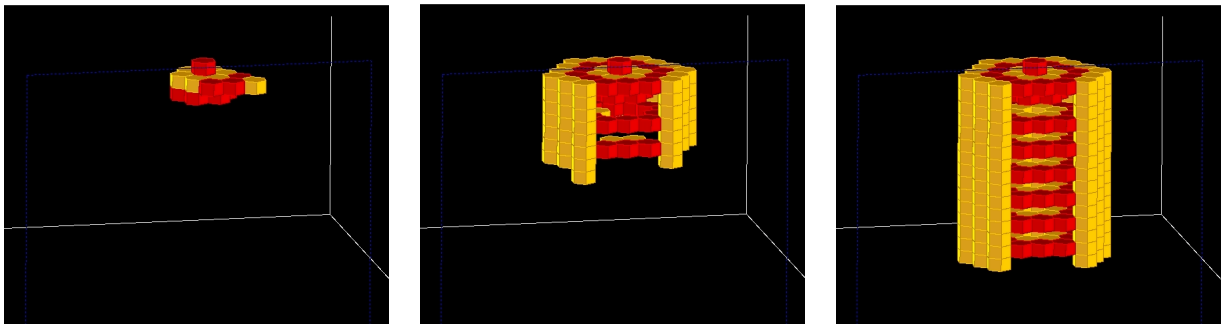


Figure 2.2: Screenshots of a stigmergic construction process. The structure is similar to the nest structure of the *Chartergus* wasp. The video footage was generously provided by Guy Theraulaz. © Guy Theraulaz, CRCA, CNRS, Universit Paul Sabatier, Toulouse, France.

Nest construction could actually be realized solely through manipulation and interpretation of the built constructions, which is coined *sematectonic communication*. It is important that the construction phases are clear and no sematectonic cues overlap [14, 25–27]. This would trigger uncoordinated behaviors in the individuals, leading to incoherent buildings. Additionally, communication among insects happens through the placement and smell of pheromones. *Stigmergy* refers to the communication through the environment in general and thereby subsumes sematectonic and pheromone-driven communication. While investigating nest constructions of termites, Grassé formulated the term stigmergy as follows [28]:

> Pour qu'il y ait continuité du travail, il faut un changement de stimulation par la création de nouveaux stimuli, en l'espèce de nouvelles constructions modifiés par le travail qu'accomplissent les ouvriers. Cela fait bien comprendre le rôle et la signification de ce que nous nommons plus loin la stigmergie.

Stigmergy can further be differentiated as (1) *qualitative*, where a specific pheromone or construction triggers a behavior, and (2) *quantitative*, where the intensity of a stimulus correlates with the triggered activities [25].

Stigmergic cues might not be readily available at the beginning of a construction process. But since the individuals align their constructions with the omni-present magnetic field of the earth, their work is still well-coordinated. In this fashion, the initial construction efforts can be easily merged at a later stage [19].

The processes involved in the construction of insect nests are as numerous as they are diverse. New construction materials need to be produced, old materials are collected, transported and recycled, different materials such as wax and propolis (in bees) are offered to the construction workers that are blended in accordance with the construction tasks at hand (or mandible, respectively), and former constructions are torn down to make space for new nest's extensions, etc. [19]. This small selection of procedures underlines that the construction activities that are investigated by ethologists describe only a small part of an immensely complex, yet

well-coordinated interaction process [25, 27]. The investigation of insect constructions offers the great advantage that the complex interaction processes are accompanied by an emerging artifact, which serves as another manifestation for scientific investigation [29].

2.1.2 Nest architecture

Insect nests primarily serve reproduction, i.e. protection and breeding, but they also realize social functions: swarm cohesion, identity and communication [27]. For predator defense they are equipped with chemical repellents and have evolved construction features such as envelopes and strategic entrance points [30, 31]. Nests can be built despite the impact of harsh natural forces, such as wind [32]. They offer shelter against temperature fluctuations through overlapping and insulating envelopes [27], or through ventilation systems that interconnect chimneys and chambers of varying exothermic properties [16].

Frisch [19] and Hansell [29] provide comprehensive views on the construction behavior of various animals including that of social insects. Studying the authors' broad overviews, it becomes clear that the nest architectures of social insect colonies are determined by their construction abilities and building behaviors, the available materials, the situational context of the nest, and the climate. If not stated otherwise, the information presented in the remainder of this subsection is distilled from the aforementioned overviews on animal architecture.

The interior structure of ants' nests is usually made up of a tunnel system connecting chambers that serve various purposes, e.g. a brood chamber, living chambers, refuse depositories. On the outside, various kinds of construction materials are aggregated. Only few objects, e.g. twigs and leaves, are pulled and pushed as far as the ants' hill's peak which produces the typical round shape of a nest. Small particles, such as grains of sand and mud, are used to patch the surface and increase its protective density. The ant hill's size and shape greatly contribute to the heating of the nest which has fundamental impact on the insects' breeding processes [33]. In an ant hill, workers continuously swap the construction material from the inside out to prevent

the formation of mold. A large variety of construction techniques is practiced by ants, especially since most of the social insect species seem generally open to the utilization of different available and suitable construction materials. An example of this habit is illustrated in Figure 2.3 where ants of the species Aphaenogaster swammerdami, subfamily Myrmicinae, protect their nest's entrance with shed snake skin[2].

Figure 2.3: The picture shows the entrance of an ant colony's nest. It is supported by shed snake skin.

Weaver ants create their nests by sewing leaves. Having reached a mature state, they are unable of producing silk. Thus, if the need arises, they rely on their larvae that produce silk for pupation. A weaver ant uses a larva like a combined tool of a needle and a spinning wheel: It carries the larva with its mandibles, directs the larva's mouth and closes the larva's head to stitch. Another living construction tool is deployed by leaf-cutter ants. These ants cut leaves to

[2]The picture and the classification of the species were generously provided by Florian Menzel, Biocenter, University of Würzburg, Germany.

feed fungus which they grow in underground chambers. Well-tended [34], the fungus in return provides for the (vegetarian) ant societies. In addition, the fungus works as a load-bearing construction for the subterranean fungus chambers.

Bumble bees, honey bees and social wasps construct one comb for each larva. The sizes of the combs vary, of course, as do the means to access them for feeding, e.g. through drilling holes into the cells, opening and closing the waxen cell covers, or simply leaving them open from one side. Accordingly, the interior designs of the nests vary greatly. There are, for instance, nests with one centralized tunnel or those with roaming space beneath the nests' protective layers. Various links between protective requirements and resulting nest architectures of wasps are illustrated by Jeanne [30].

The construction performance of bumble bees, with comparably few members of a hive, is usually ranked below the standards met by honey bees and wasps. The latter two follow a strict hexagonal comb design. Exact measures are maintained with only minimal deviations due to a precise construction performance and the use of a standardized construction tool kit which is anchored deeply into the insects' anatomy. The insects' anatomies support their construction through unified dimensions of their limbs, the ability to measure their spatial orientation through the gravitational forces exercised on their heads, and a disposition to execute pre-scripted movements within accurately predictable time intervals.

Bumble bees are sufficiently learned to cover the surface of earth-bound nests with a layer of wax to exclude dampness. But they are also smart enough to utilize feathers or hair of roe deer to build layers of insulation—again, the availability of the construction material plays a big role in the emerging nest architectures. Bumble bees, just like honey bees, make combs from wax. Frequently, recycling of this precious resource has been observed: Old nests were dismantled and flocks of bees transported the old wax to the newly founded colony. More abundant are the construction materials of potter wasps that blend clay and sand to mortar, or of wasps that produce a light paper secretion from shaved-off wood particles. Although the

cardboard material used by paper wasps is very light, the resulting constructions obtain good robustness due to longitudinally aligned wood fibers.

Termites are often considered the master-architects among the insect orders. Indeed, they seem to incorporate most of the key technologies mentioned above. Although they do not build single-celled combs, some create perfectly regular architectures similar to those of bees and wasps. Similar to wasps, termites also produce their own construction materials. Some termites build paper nests as well. But overall, termites are better known as specialists in producing and processing secreted mortar. After dropping a pellet they turn around to model and smooth the material with their mandibles, thus being able to build intricate architectural structures such as arcs. Similar to the nesting habits of other insects, termites build in trees as well as beneath and above the ground. But termites are also infamous for the peculiarity to digest wood, thereby causing great damages to man-built constructions. Their buried nests appear in great variety. They build compact nests several meters beneath the ground that are connected to their environments through webs of galleries. Often, however, their underground dwellings resemble those of ants—with interconnected chambers for tending fungus, food storage, and reproduction. A queen and king take on the responsibility for the latter activity, usually inhabiting a specially dedicated chamber at the center of the nest. Above ground, termitaria rise to impressive heights, as seen in Figure 2.1. Usually, the termites' nests are only extended above the ground when their subterranean levels have already gained considerable dimensions. The termites' ability to model mortar renders it possible to maintain well-protected mounds for many decades (the king and queen pass their scepters on to new royalty when they pass away).

Considering the insect nests' interior climates, several technologies were mentioned: Excluding dampness through wax layers, the construction of protective layers for insulation, ant hill formation for sunlight exposure, and drying cycles of damp construction materials. Additional individual behaviors contribute to thermal regulations. Wasps, for example, create heat through quick contraction and stretches of their abdomens and they cool off the nest through

transporting water into the nest. Ants on the other hand, increase the temperature through extended sunbathing and subsequent descent into their nests. Yet, there is another phenomenon of climate regulation which is observed in various ant [35, 36] and termite nests [37]. The architectural designs are adapted to the corresponding local climates, a trade-off is found between air ventilation and heat dissipation. The key technologies offered by the environment and followed by the insect builders are the respective dominant factors of wind and sun. The exact mechanisms still need closer investigations but it is assumed that both thermal radiation as well as wind-induced flow drive the ventilation of insect nests.

2.2 From developmental models to complex systems

The developmental processes that result in organic and architectural structures are extraordinarily complex and of great scientific interest. It is therefore not surprising that computer scientists have been eagerly creating algorithmic models to retrace these phenomena. Developmental processes might find their manifestations in temporary state configurations of complex, volatile systems, or might, even more abstractly, traverse states in a theoretical model that seeks empirical confirmation. Computational developmental models (CDMs), however, usually focus on the manifestation of structural artifacts in virtual spaces [38]. In particular, CDMs can be seen as algorithmic systems in which a considerably small *database* is *amplified* in an iterative process of growth or unfolding[3].

2.2.1 Cellular automata

In *cellular automata* (CA) the processing units (*cells*) are organized in a lattice structure and are set to an initial state that is changed in accordance with a set of rules that consider the states of all neighboring cells. Von Neumann developed CAs to model phenomena of self-reproduction

[3]The expression "database amplification" is used in literature in the context of computer graphics applications of generative and developmental systems. It underlines the relationship between a small amount of provided information and the possibly large amounts of automatically derived data [31, 39].

based on the interplay among a large number of finite state machines [40, 41]. With a successful implementation that immensely simplified von Neumann's first modeling attempt, Codd provided a first completely self-reproducing artificial machine [42]. Later, Langton discovered that there is no requirement for a self-reproducing automaton working as a universal computer. He provided an according CA-based system which became known as *Langton's Loops* [43]. Eventually, Conway drastically reduced the complexity of self-reproducing CAs once again. Considering only three rules, self-reproducing, propagating and intriguingly interacting patterns emerge on a two-dimensional CA lattice and rules that take eight neighboring cells into account (*Moore neighborhood*):

1. Each cell with one or no neighbors dies,
2. each cell with four or more neighbors dies,
3. each cell with three neighbors becomes populated.

In fact, in 1998, Reggia et al. showed that self-replicating loops automatically evolve on different scales in a CA that copes with eight different cell states and which is seeded with an initial random allocation of one fourth of all cells [44].

2.2.2 Artificial chemistries

In Miller's famous "primordial soup" experiment in 1953, he artificially synthesized simple proteins, providing fundamental insights into some of the mechanisms contributing to the origin of life [45]. The analogous idea—to have a selection of molecules react *in silico*—is realized by so-called artificial chemistries (ACs). An AC usually comprises a set of molecules S, a set of reaction rules R and an interpretation system, or algorithm A to drive the simulation of chemical reactions [46]. Based on a reaction rule, two (or more) molecules, represented as strings of symbols, would interact and possibly create a new molecule, be reconfigured or be

merged. Figure 2.4 shows an artificial chemistry system in action [47]. Six different particles, $a0$ to $f0$ are contained in a two-dimensional space, wandering randomly.

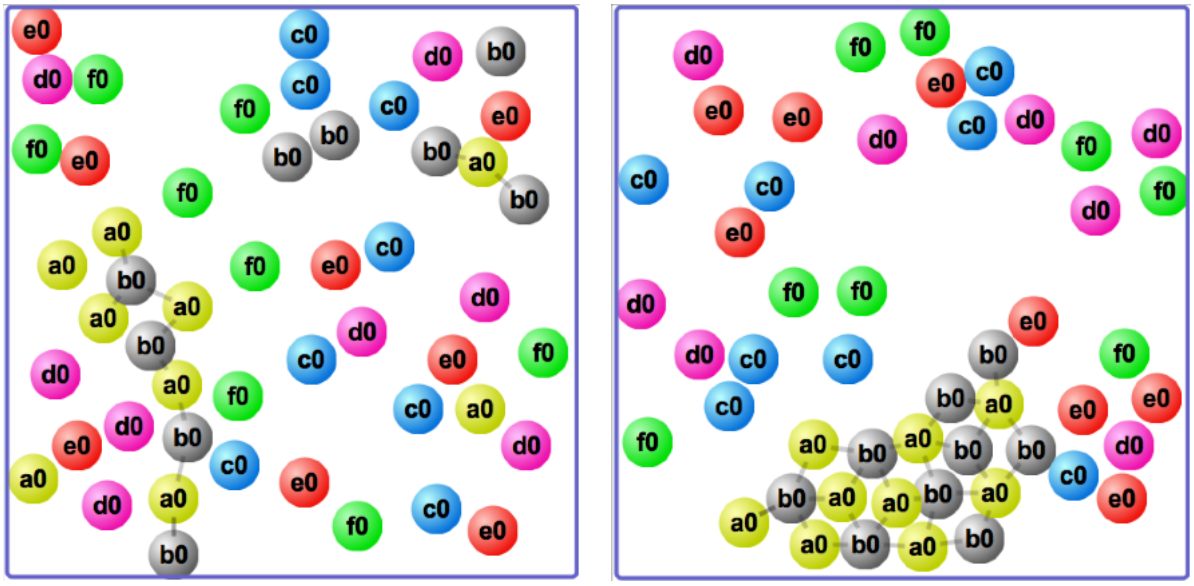


Figure 2.4: Six kinds of atoms interact. The reaction rule $a0 + b0 \longrightarrow a0b0$ makes the corresponding atoms cluster into one molecule seen at time step t_1.

Dozens of implementations of ACs have been proposed and utilized for different kinds of experiments. A detailed overview is presented in [48]. Mostly, the interactions of the molecules would be executed in a random order. However, there are ACs that consider topologies on the molecules which in turn determine the succession of reactions (e.g. the *chemical casting model*). There are stochastic ACs providing several reaction rules for the same pairs of interacting molecules. In *artificial molecular machines* and *abstract automata* the (active) reaction rules directly share the reaction space with the (passive) molecules. The concept of ACs was further adjusted to their computer hosts with molecules that bundle assembly code and reactions that realize the molecules' interpretation (*assembler automata*).

ACs found applications mainly in three areas:

1. In modeling of biological, chemical, evolutionary, self-assembly, social and ecological

systems.

2. In information processing to compute the outcome of (real) chemical experiments.

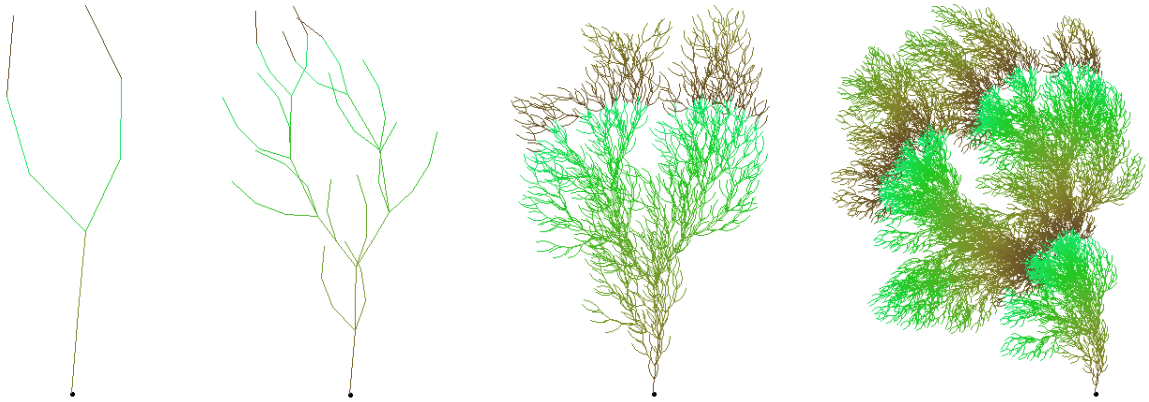
3. In optimization for combinatorial tasks—especially the computing processes of non-topological ACs can easily be distributed to multiple computers and be executed in parallel.

2.2.3 L-systems

Like in psychological developmental processes, there are many biologically researched phenomena that also require a more abstract, high-level description. Aristid Lindenmayer and Przemyslaw Prusinkiewicz, for instance, developed *L-systems* as a CDM for the growth of biological organisms, especially of plants [49,50]. Hereby, formal systems are utilized to generate the developmental stages of simple organisms in so-called cellular arrays that are represented as strings. Repeated application of grammatical production rules generates a cellular array which can then be interpreted graphically. One important difference to traditional formal languages is the parallel application of production rules.

While certain symbols of the resulting string represent cells, other auxiliary symbols, like brackets, determine the recursive branching properties of the structure. The *turtle interpretation* serves as a metaphor for the graphical transposition of a generated string: A "turtle" is drawing a line while moving according to the string symbols' meanings like F for "forward", $-$ for "turn left", $+$ for "turn right", [for "remember position" and] for "resume last position". Figure 2.5 depicts the development of an L-system drawn by means of the turtle interpretation [51].

Lindenmayer and Prusinkiewicz investigated the relation of the expressiveness of L-systems and their classification in accordance with the Chomsky hierarchy of formal languages. Context-free 0L-systems suffice for modeling the development of systems without intercellular commu-



gen. 0 gen. 1 gen. 2 gen. 3

Figure 2.5: The development of an L-system structure: Starting with an axiom x, the rules $x \rightarrow F[+x]F[-x]+x$ and $F \rightarrow FF$ are repeatedly applied in parallel yielding four generations of growth. Small degrees of randomness in respect to rule application, varying angles and lengths generate an organic look.

nication. Otherwise, one- or two-sided input systems are necessary (*1L- and 2L-systems*). There are many extensions to the original L-system idea. For example, *parameterized L-systems* allow for the assignment of a numeric vector to each symbol. In the course of an organismal development these numeric values can capture and effect continuous changes. Parameterized L-systems also allow for more complex production rules that consider mathematical constraints on the carried numerical values. Other variations of L-system include *propagating L-systems* that do not contain erasing rules, or *deterministic L-systems* in which only one rule is applicable in any given case[4]. Neighborhood dependent cell division can be simulated with *map L-systems* [53]. In map L-systems markers are first written into the cellular array string. Subsequently, these markers indicate a compartmentalization of the developing structure through the insertion of edges.

[4]The solution to the *inference problem*, that is to create an adequate L-system for a given structure, is discussed and depicted in Jacob's book "Illustrating Evolutionary Computation with Mathematica" [52].

2.2.4 Universal CDMs

Giavitto et al. summarize several approaches to CDMs [54]. Firstly, there are *dynamical systems* with sets of state variables that determine their global states. Secondly, there are *structured dynamical systems* which are dynamic systems that can be divided into subsystems. Finally, there are *dynamical systems with dynamical structures*, abbreviated to $(DS)^2$-systems, for instance a "developing multi-cellular organism" [55]. In addition, Giavitto et al. describe developmental models as tuples comprising both a topology and a formalism. For instance multi-sets in tandem with multi-set rewriting, sequences (like symbolic strings) and the concept of L-systems, uniform topologies that are processed in group-based data fields (GBF)[5] [56], or combinatorial topologies and map L-systems that are mentioned above.

There are, however, also formalisms that explicitly integrate the topology of the systems, such as graph-grammars, multiscale tree graphs (MTGs)[6] and the modèle géneral de simulation (MGS) which represents changes of *topological collections* of units by *transformation paths* on a symbolic notation [57].

Kniemeyer et al. have developed *relational growth grammars* (RGGs) which promise, like MGS, to be a universally applicable representation of CDMs [58]. They use RGGs as extensions of parametric L-systems with object-oriented, rule-based, procedural features. In fact, modeling CDMs by graph grammars, like in RGGs, allows for the expression of all developmental data structures we know about: multisets, strings, axial trees, and relational structures (edge-labeled directed graphs). RGGs can therefore be seen as a universal modeling language, being able to simulate standard L-systems, artificial chemistries, CAs and ecological systems alike. Kniemeyer et al. successfully applied the RGG model to grow multi-scale models of plants integrating their structure and function [59], and just recently to grow architectural models [60]. They also suggested that RGGs could support *agent-based* modeling (see Section

[5] GBF define a homogeneous, fixed topology on sets of units.

[6] Parallel, modular growth on different scales can cause severe complications in MTGs.

2.5.2)—by interpreting nodes as agents, edges as inter-agent relations, and by driving their interactions through sub-graph substitutions [61].

Almost 20 years before Kniemeyer presented RGGs, Culik et al. had extended L-systems with the means to describe plants through graph structures and their growth through graph grammatical substitutions, which were further on referred to as *graph L-systems* [62]. Shortly afterward, Nagl investigated the relationship between graph grammars and graph L-systems, concluding that graph grammars can be reduced to graph L-systems and vice versa [63]: identical graphs can be achieved by either sequential graph grammar productions or by parallel sub-graph substitutions as realized in graph L-systems. About another decade later, Lindenmayer argued that relying on maps instead of graphs bears many advantages, e.g. a clear method for mapping between the representation and the growing structures and better performance due to the avoidance of transformations of the representations [64].

2.3 From developmental models to creative design

Shortly after Chomsky presented formal grammars as a system for the generation of syntax [65], CDMs started thriving in the arts and design. Soon CDMs were utilized to model the generative composition of texts and patterns and their decompositional analysis, respectively [66]. CDMs were, for instance, applied to generate patterns of cattle brands [67] and to retrace two-dimensional blueprints of Palladian Villas [68]. Just recently, the computational generation of Palladian Villa architecture transcended virtuality by the means of three-dimensional rapid-prototype printers [69]. Until today, novel ways to produce architectural blueprints through generative grammars are being explored. A very recent example is provided by the interactive generative architectural modeling tool called ArchiDNA [70]. Here, the two-dimensional symbolic shapes of nucleic acids are used as geometric building blocks to compose (two-dimensional) blueprints as instigated by the architect Peter Eisemann. Similarly

to these architectural blueprints, CDMs were also used to retrace the composition of art pieces, for instance in regard to Richard Diebenkorn's Ocean Park paintings [71].

2.3.1 Evolving art & design

Often, the immediate goals of applying CDMs in design and art have exhibited a reproductive character and fostered only small deviations from an original design work or work of art. Another approach to harness the potential of computer-generated creativity is based on the creative power of computational evolution [72–76]. Based on L-Systems, and other mathematical, fractal methods intriguing sculptures of artificial aesthetics can be evolved as well [77, 78]. Whole worlds of seemingly living organisms grow in virtual spaces [75, 79, 80].

In 1987, Dawkins showed how vast varieties of *biomorphs*, recurrent, symmetric, branching line structures that resemble organic organisms (Figure 2.6) could be bred interactively by means of random mutations and selection through an external breeder [81]. Similarly, Sims applied evolutionary computation to breed three-dimensional plant structures and images [82]. Todd and Latham provided another intriguing perspective on evolutionary art, focusing on elaborate sculptural works [73]. Jacob developed *Evolvica*, a framework for evolutionary experiments written in Mathematica [52]. Later, Kwong and Jacob complemented Evolvica with *Inspirica*, a user-interface that allows to interactively breed structures and interaction processes. By means of Inspirica, they interactively bred pattern formations in simulated flocks [13] and three-dimensional structures [83, 84].

The interplay of computationally and user-created graphical content can be extended to interactive, real-time art installations [86]. Until today, innumerable artistic designs were bred [87, 88] and evolutionary art installations set up, e.g. [80]. Interesting chapters on evolutionary computing and art and design are provided in [72] and [74]. In the latter reference, for instance, one chapter is dedicated to Hemberg et al.'s *Genr8* system, a sophisticated design tool for architects [89, 90]. Genr8 bundles the technologies of surface generation by L-systems and

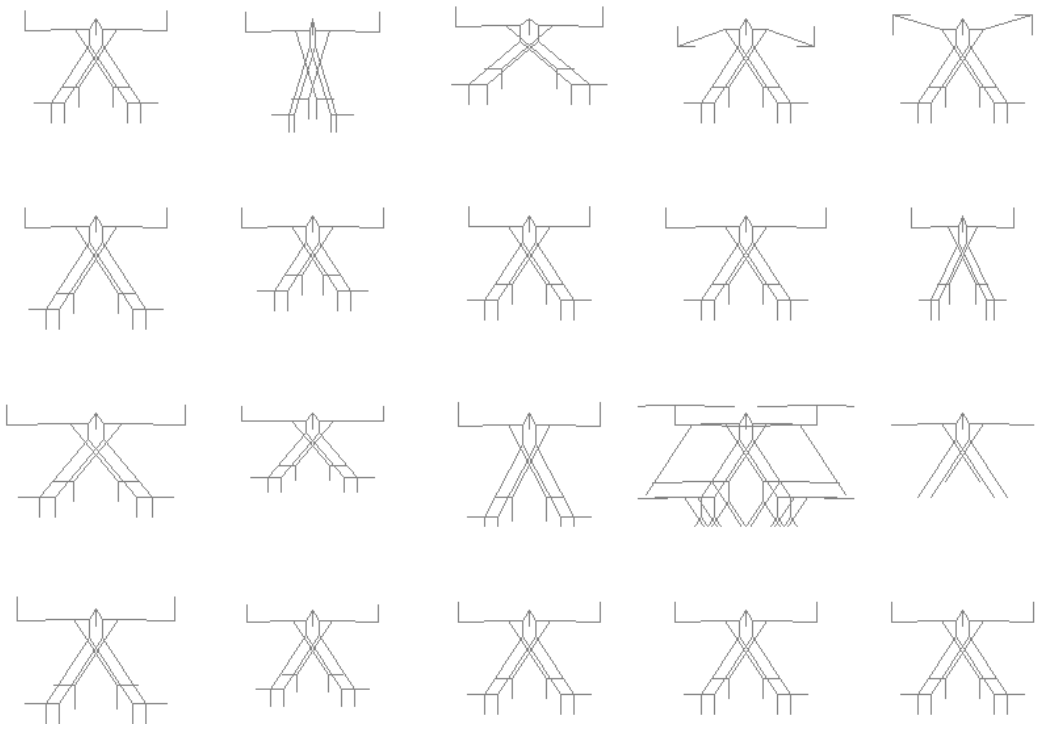


Figure 2.6: The figure shows a screenshot of a biomorph implementation by Nardella [85] which closely resembles the original one [81].

evolutionary algorithms and makes them accessible in a user-friendly *Maya*[7] plugin.

The combination of CDMs, evolutionary computation and user feedback provides a comprehensive toolset for artists and designers. In this context, King differentiates between *arbitrary* and *algorithmic synthesis* of artistic content. These terms are coined through the corresponding processes during which either an artist determines the object composition arbitrarily or an algorithm automatically performs the task [77]. But even when automatically evaluating CDM-driven computer art, human-made works frequently provide the statistical basis for judgements [93, 94].

[7]Maya is a popular 3D modeling environment for artists, designers and architects [91, 92].

2.3.2 Integrating organic and functional design

Seemingly opposed to the problem of artistic creativity, the original motivations for some evolutionary computation methodologies were industrial design problems. Rechenberg and Schwefel, for instance, were experimenting with chance-driven optimization of a nozzle's shape to increase its thrust before formalizing the respective elaborate algorithmic optimization framework of *evolution strategies* [95]. The generative character of evolutionary algorithms is especially evident when looking at Koza's method of *genetic programming* in which programs are evolved by growing, crossbreeding and refining trees of code [96]. His systems exhibited unquestionable creativity when, for instance, generating novel designs of high-performing electric circuits [97].

It cannot be argued that some artistic works seek truth beyond the quantifiable and, therefore, one may have the opinion that aesthetics cannot, or should not, be subjected to any kind of optimization method—as which evolutionary algorithms are frequently referred to. At the same time, however, it can neither be argued that with the advancement of tools and technology our design methodologies change. Just as drawing has become part of architectural practice during the Renaissance [98], computer-aided design has within the last decades as well. In fact, when provided with basic procedural and structural building blocks (a CDM), an algorithmic system (for instance evolutionary algorithms) by itself can generate solutions to predefined problems, or at least optimize design in respect to imposed constraints, e.g. certain required functionalities.

Instead of the obsolescence of theory in design and architecture due to the autonomy of generative machines, Frichot [99] suggests to embrace the opportunities of design intelligence [100] that are offered by the new rising paradigm which is also refered to as projective architecture [101]. After all, the 3D constructions produced, for instance by Genr8 [90], serve as digital *idea models for architectural designs* similar to those that are traditionally manually shaped from wood [102].

By mimicking natural construction processes in a CDM, complex, nature-inspired design elements can be adopted [103, 104]. Indeed, nature-inspired design has become a well-established part of architecture since the 1990s [22]. According to Pearson, predominance of rectangular, cubic elements, the emphasis of the "straight line" is a relic of the industrial revolution and not desirable [105]. Instead, he claims that an organic style with "free-flowing curves" should be favored which is supported by novel construction processes and materials. Although the automatic generation of architectural design bears the potential to consider attributes beyond organic looks and sound statics, form prevailed over function and true complex pretensions in most of the examples documented in [22].

A rather big challenge for automatic design generation might be the requirement of architecture to be integrated into the environment [103]. There are, however, factors such as waste water disposal, energy efficiency and other aspects of ecological and economic performance [106] that are quantifiable [107] and, thus, could be immediately addressed in automatic design generation processes.

2.4 Evo-devo: development on multiple scales

In analogy to the fact that genes cannot be read as linear code instructions [108], Jacob pointed out that "the art of genetic programming" encompasses both a complex developmental process arising through a CDM and an evolutionary algorithm that produces and hones the CDM configuration [109]. Above, we refer to systems that fulfill these criteria, thus being able to create novel designs, e.g. Genr8 [89, 90]. When models consider both the long-term impact of evolution as well as the short-term process of development, they are referred to as *evo-devo* models [20, 38, 110]. Some CDMs have specifically been developed to work well with evolutionary algorithms resulting in genuine computational evo-devo models, whereas other CDMs were designed independently just to be subsequently optimized for their use with com-

putational breeding techniques. In this section, a coarse perspective on evo-devo models is provided.

2.4.1 The evolution of evolution

Miller's famous experiment on the synthesis on protein molecules is mentioned above to introduce the idea of artificial chemistries. However, it also lends itself to the fact that evolutionary processes have been happening in which atomic and molecular interactions self-organize to form increasingly complex systems. Kauffman calls this phenomenon "self-organization for free" [111], whereas others refer to it as *chemical evolution*. Developmental processes and evolutionary processes are tightly interwoven: Molecules and sets of molecules interact, grow, change, adjust, reproduce, exchange parts—they evolve. Evolutionary pressure is exercised as some of those molecules persist over time and others do not. Sooner or later, those that have successfully occupied an ecological niche, or those that have reproduced very effectively, appear in growing numbers and with various alterations. Again, some persist and others do not—the evolutionary engine is running. Margulis and Sagan claim that real leaps in the evolution of life happen when independently evolved organisms form new synergetic bonds [112]. A famous example is the eukaryotic cell that contains one or several mitochondira whose genetic information is not only separately stored but also strongly resembles those of bacteria.

Taking these insights one step further, Griffiths and Gray argue that "developmental processes, rather than genes or traditional phenotypes, to be the units of evolution", whereas "[...] the prime unit of evolution (unit of self-replication) is the developmental process, or life cycle." [110]. So they claim that the observations in chemical evolution result from the same principles as those seen in *biological evolution*. Of course, many other evolutionary perspectives support this idea, for instance the evolution of ideas themselves [113], the evolution of the corresponding cultures [114], and the evolution of language [115].

In systems of evolutionary computation the relation to developmental models is especially

evident through the separation of *genotype* information and the computation of the correlated *phenotype* [52, 97]. Therein, firstly, *genetic operators*, e.g. selection, mutation and recombination, are applied on the level of (abstract) data structures, and secondly, this genotypical information drives developmental processes to realize the phenotype—anything but a $1{:}1$-mapping between genotype and phenotype can be interpreted as a CDM. In the following paragraphs several examples of such computational evo-devo models are presented.

2.4.2 Gene regulation and embryonic systems

Kumar and Bentley named their integrated evo-devo approach *evolutionary developmental system* (EDS) [38]. In particular, it is an object-oriented system that incorporates proteins, genes and cells and promotes the development of genetic regulatory networks for an embryonic development. Cells are stored hierarchically, in an n-ary data structure. The root represents the zygote, whereas progeny cells are attached as subtrees. Proteins reside in their according host cells. Through hierarchical traversal of the described data structure, signal pathways between cells direct the cellular developments. The graphical and spatial representation is realized through the creation and interpretation of *Virtual Reality Modeling Language* (VRML) expressions. The superimposed evolutionary algorithm is responsible for the configuration of the biological model and promotes (1) inner-cellular gene regulatory networks and (2) the development of pre-defined embryonic shapes.

Despite the preliminary success of the outlined model, alternative embryonic evo-devo models put an emphasis on other, inter-cellular mechanisms of interaction. In [116] asymmetry is established to greatly influence the initial and subsequent developmental patterns of embryonic processes. It comes about in two fashions. (1) Developmental signals are localized in the egg, determining the fate of cells in different partitions. This kind of asymmetric cell differentiation is, for instance, accounted for in the developmental processes in the fruit fly, Drosophila melanogaster, and the roundworm Caenorhabditis elegans. (2) In the soil-living

amoeba Dictyostelium discoideum, on the other hand, solely the local environment seems to determine the differentiation of a cell. This determination of a cell's fate also drives its migration to the correct position in the embryo.

2.4.3 Artificial neural nets and morphologies

Apart from *artificial neural networks* (ANNs) as computational means to analytically or stochastically abstract sets of training data [117, 118], there are computational models that actually try to capture the growth of neural nets and to retrace the behavior of the physiological nervous system. In [119] Astor and Adami present an autonomous, agent-based approach to neural growth and interaction. Local substrates that are modeled by artificial chemistries and internal genetic information determine the cells' activities. Manually coded genetic information drive the emerging gene regulation networks that allow for the emergence of deterministic structural development, self-limiting cell growth, regenerating structures, directed growth, logical computation, and system adjustments similar to traditional ANN learning. The authors believe that these functionalities could support more complex behaviors of the system, but they point out that the computational effort to evolve such behaviors is immense and would require the computing power of a massively distributed system.

In analogy to the understanding of the control of our physical activities induced by our nervous system, artificial neural nets have been evolved in tandem with the body plans of artificial creatures [120]. A comprehensive overview, a standardized description and implementation is provided by Pilat [121]. The goal of successful body control is frequently measured as the distance that is overcome by an evolved creature in a physically simulated environment. Dellaert and Beer, for instance, presented an according model of developmental evolution [122]. They see certain benefits in the developmental aspect of their model, especially (1) robustness to genetic changes (filtering effects of small mutations), (2) broad possible impact of a single mutation—depending on its influence during the developmental process, and (3) compact en-

coding for complex phenotypes: incremental enhancement, supporting symmetry and modular design. Despite of their rather abstract modeling approach, they conclude that "Development is an important, powerful and integral element of biological evolution.". Hornby put special emphasis on advantageous properties of the genotype representation in such morphological experiments [123]. He concluded that a generative representation allows for "the encapsulation, coordination, and reuse of assemblies of parts". He also pointed out that basic features of a universal computing system are captured by the generative representation: control flow, combination and abstraction.

2.4.4 The spatiality of evo-devo models

All the mentioned examples of evo-devo models embrace physical aspects of simulation. This is not surprising as the developmentally grown structures' success, or fitness, is measured against some physical constraints. The fundamental challenge in this context is the prediction of function associated with form and vice versa—on a molecular [124], a cellular [125] and an organismal level [121].

And indeed, basic principles are being unearthed that consistently appear on all these levels of organization. *Transegrity*, for instance, refers to the fact that numerous natural structures, e.g. neural cells or the human body, are not simply obeying all penetrating forces like gravity, but they are able to withstand, to keep and even to self-organize their form (and thus their functionality). Transegrity, Ingber explains, emerges through a structural web of flexible and solid construction elements [126]. Paul et al. have successfully applied evo-devo models to finding transegrity structures [127].

Mechanistic interactions on a molecular level are nonessential for the simulation of transegrity of the human musculature. Instead, abstract descriptions of the workings of muscles and bones, their mechanistic and material characteristics suffice to achieve adequate numerical results [128]. Nevertheless, it is also deemed necessary to follow eager concepts of precise and

detailed simulations as some important phenomena might only emerge on the level of elementary interactions.

Smith devised a *vertex-vertex* language to directly manipulate vertices in triangular mesh data [129]. Triangle meshes have advanced to an established, inherently spatial data representation in current computers. By creating a language to transform these data structures in accordance with physical and bio-chemical theories, a direct mapping between vertices and the units of a model (e.g. molecules) can be easily implemented. This can greatly facilitate the implementation of complex biological simulations that are based on extensive computations in three-dimensional space [130].

Even a theory that could potentially explain the emergence of all physical phenomena, a so-called *Theory of Everything* (TOE) [131], could not lead to an all-encompassing computer simulation—after all, the universe consumes all its resources to constantly compute itself [132][8]. Which leaves but one possibility: to find and improve suitable approximations of certain aspects of reality (abstractions) and to diligently consider the resulting insights for governing our lives. In the next section, some insights are described that seem of great importance to successfully cope with the complexity of life.

2.5 Complexity

Although *complexity* is attributed to a broad variety of systems and phenomena, it lacks a clear definition. In order to narrow it down, this section provides answers to the following three questions. (1) Under which circumstances do we attribute complexity? (2) What is its cause? (3) How can complexity be quantified?

[8] This is but one argument that renders innumerable other arguments for the impossibility of a Laplace's demon unnecessary.

2.5.1 Categories of complexity: identifying complex phenomena

Not only was von Neumann's goal of creating self-reproducing systems finally achieved [133], but cellular automata also became a general model for complex systems based on neighborhood-dependent state changes [134]. Systematic investigations of the global state changes of one-dimensional CAs led Wolfram to a general classification scheme of the complexity of patterns emerging in CAs [135].

Wolfram's efforts were supported by an important advantage of one-dimensional over other n-dimensional CAs: The development of a CA over several iterations can be illustrated at a stretch (Figure 2.7). Wolfram described the discovered qualitative categories of CA complexity as follows:

1. Spatially homogeneous state
2. Sequence of simple stable or periodic structures
3. Chaotic aperiodic behavior
4. Complicated localized structures, some of them propagating

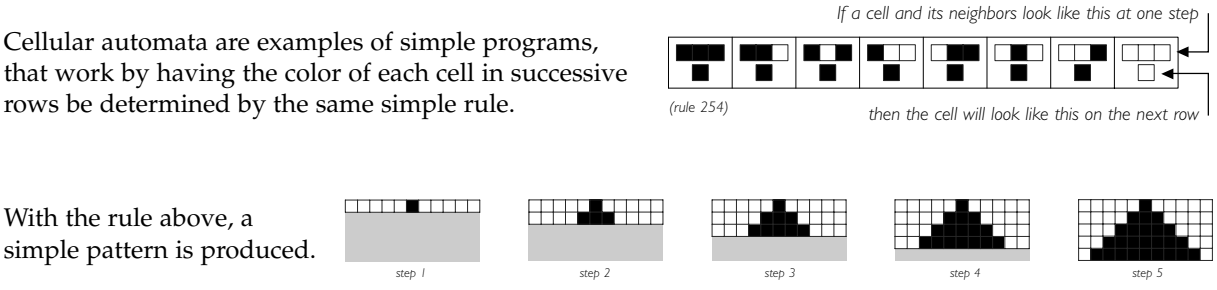


Figure 2.7: An illustrative example of the workings of 2D CAs. Illustrations and text are directly copied from supplementary material [136] for the book "A New Kind of Science" [134].

As opposed to the fixed spatial cell arrangements in CAs, *Random Boolean Networks* (RBNs) abstract from the notion of space [111]. In RBNs each cell can be connected to any

other one, forming an information propagating network. As in CAs, the configuration of all cells defines the global system state. With a novel representation, a new terminology was introduced: If a specific global state of an RBN is reached starting from a set of independent initial states, it is called an *attractor state*. Graphically, the transition paths of state transformations that converge to an attractor state can be depicted as streams of a *basin of attraction*. It might also happen that an RBN does not reach a specific system state: Instead, the system might reach a *fixed* or *steady state*, or it might infinitely loop through a subset of states, be trapped in a so-called *limit cycle*.

A slightly different perspective on global system states was brought about by investigations of pattern formation in artificial chemistries. *Autocatalytic metabolisms* were first discovered by Bagley and Farmer [137]: An autocatalytic set of molecules is formed through catalytic reactions and sustains itself. As another example, *hypercycles* were established in an abstract automata configuration where *machines* and *tapes* share the (virtual) reaction space. Schuster and Eigen identified hypercycles as "networks of 'functionally completed, self-replicating entities'" [138].

From a more general point of view, Banzhaf suggested a classification scheme for CDMs that are usually associated with complex pattern formations or complex behaviors [46]. In particular, he differentiates between three kinds of systems: (1) *self-assembly*, (2) *self-formation* and (3) *self-organization* systems. In self-assembly systems patterns emerge through repeated application of an invariant mechanism. A self-formation system traverses through a sequence of system states, at each step enabling novel means to react to the environment. Self-organization systems reach a stable, self-maintaining state without the necessity of a specific initial configuration nor a fixed succession of events. Banzhaf also pointed out that a developmental model should be coined *constructive*, if the emerging structures grow according to a provided complexity measure, as introduced in the remainder of this section.

2.5.2 The cause of complex behaviors

Generalizing models of complexity usually involve interactions of multiple units or *agents*[9] [139]. This observation automatically raises the questions of when, i.e. under which circumstances, and how, i.e. yielding which consequences, these interactions take place. The first question is usually addressed by considering a *topology* or *structure* on the interacting agents. The second issue is often subsumed by defining the agents' *behaviors*.

Regarding the topology of complex interaction networks, it has been speculated that the degrees of protein-protein interactions which regulate the gene expression in fruit flies follow a power-law distribution [140]. Even though different complex networks have different mathematical properties [141], it is insightful to study the networks of interacting units in developmental systems. From a power-law degree distribution of interactions in a system one could, for example, derive that it is robust to random noise [142]. Power-law degree distribution is also an indicator for the *small-world* property of the investigated network [143–146]. This property attributes a small mean path length between any pair of nodes and a relatively high cluster coefficient, resulting in highly connected subgraphs.

Albert et al. conducted experiments to create and analyze random RBNs with the same distributions of degrees of connectivity as those shown in gene regulatory networks [147] and in other scale-free networks [142]. It has been demonstrated that such RBN models show state transition patterns very similar to those of natural networks [148].

The behaviors of the involved agents is closely interwoven with the emerging topology of interactions: The agents need the possibility to interact which is provided by their *senses*. In CAs, cells know the states of their immediate neighboring cells, in RBNs a unit 'sees' an arbitrarily defined set of other units, in ACs *collisions* trigger the application of a reaction rule. Obviously, the interaction network in a complex system strongly depends on the sensory

[9] Depending on the field of science one is most involved with, agents might also be referred to as *cells*, *units*, *objects*, *agencies*, *molecules*, *particles*, etc.

capabilities of the involved agents.

Having decided to react upon some sensory information, an agent takes an action, it triggers an actuator. This action in turn can influence its own state or the states of its neighbors. Denzinger provides the following broad defnition of an agent [149]. An agent is defined as the quadruple $Ag = (Sit, Act, Dat, f_{Ag})$. Ag can find itself in any of the situations expressed in Sit. It can perform the actions described by the set Act. Its internal data areas, i.e. local variables or memory cells, are determined by the set of possible values Dat. Based on the perceived situation and its internal data values, the agent determines the next action through a decision function $f_{Ag} : Sit \times Dat \rightarrow Act$.

Numerous investigations of natural complex systems support the assumption that these systems are caused by the interplay of large numbers of agents. However, it is a great challenge to artificially design complex systems. It is hard to engineer features that render systems robust, self-organized, sustainable or highly adaptive, because these and other global traits of a complex system are often considered *emergent* [150, 151], i.e. they cannot be foretold by the designs of the individual agent behaviors. Instead, the interaction dynamics will bring them about. Therefore, it is of great interest to identify phenomena in complex systems that link local agent behaviors with global emergent occurrences. The search for such links has unearthed various measures to describe complexity.

2.5.3 Measures of complexity: quantifying different complex phenomena

Schuster summarized different approaches to measuring complexity. Complexity can be expressed in the *ecological diversity* of a system, in the *complexity of construction* in respect to its functionality, as the *internal complexity* of a system referring to its *logical depth*, or in a system's *hierarchical organization* [138]. Schuster himself reduced complex design to the processes that give rise to the artifacts and stated that: "nature uses optimization to deal with scarcity, she takes advantage of abundance to create innovation, and her recipe to master

unpredictability is tinkering and modular design.".

Hornby supports the idea of modularity in order to create complex designs. He showed, for instance, that through the reuse of assembly modules, a single parametric, generative representation can host a wealth of similar designs which he calls *families of designs* [152]. He furthermore defined a complexity measure considering the *reuse, modularity and hierarchy* (RMH-measure) as provided by the genotype, or the code implementing a CDM [153]. Hornby pointed out, however, that this measure has to be normalized by the *design size* of the phenotype or the *algorithmic information content* of the genotype in order to be scalable for varying design tasks.

2.6 Completing the cycle: complex and constructive swarms

The development of swarm-inspired construction algorithms was motivated by the simulation of collaborative stigmergic building processes, i.e. the intricate nest constructions of social insects. At the same time, these algorithmic models qualify as CDMs that integrate large numbers of locally interacting builder agents. From these, non-linear complex interaction dynamics emerge that manifest in architectural artifacts. Thus, complexity measures can be applied to investigate the link between local behavioral programs, the emerging interaction processes and the corresponding artifacts.

As the last section of this chapter, it ties these facts loosely together and prepares the stage for the contribution of this thesis, namely the design of a complex, dynamic, developmental swarm model. It introduces *boids*, a seminal model of artificial swarms, investigations about its complexity, experiments with boids in the context of computational evolution, and models of *constructive* swarms.

2.6.1 Natural swarms are complex systems

In Section 2.1, the nest-building approaches of social insects were outlined. It is easy to interpret insect colonies as *multi-agent systems* (MAS) encompassing large numbers of individuals [154]. As function follows form, insect individuals differentiate into workers, warriors or potential successors to the crown [19]. Stigmergic signals trigger highly evolved behaviors that yield well-coordinated, thus effective, and efficient collaborations. Those behaviors are scripted and, even though they address vast assortments of situations, they can be decoded when investigating certain subtasks, such as foraging or nest-construction [25, 27]. In the latter example of collaborative productivity, built artifacts foster the investigation of the outcome and the preceding complex networks of interaction [141]. Thereby, local individual behaviors can be connected to global emergent phenomena. In brief, social insect colonies provide an inspirational model that encompasses a multitude of aspects of complex systems. However, for a step-wise investigation into swarm complexity, models of flocking swarms, e.g. flocks of birds or schools of fish, have gained great attention.

2.6.2 Flocking models

The swarm metaphor stands for large numbers of individuals that follow simple scripted behaviors but achieve globally emerging phenomena, as in schools of fish or flocks of birds. In 1987 Reynolds presented a simulation concept that mirrors the flocking dynamics of birds [13]. In his model, a simulated virtual bird, also called *boid*, is represented as a pyramidal graphical object oriented towards its velocity. In order to be able to react to its neighbors, a boid possesses a conic field of perception that is determined by a viewing distance d and an angle α. Rather loosely, Reynolds explained which urges for acceleration are triggered by the flocking mates perceived in a neighborhood. Figure 2.8 illustrates the according urges schematically.

Alignment adjusts the direction to the neighbors (Figure 2.8(a)), *cohesion* draws the individual towards its neighbors (Figure 2.8(b)), and an urge for *separation* prevents the boids from

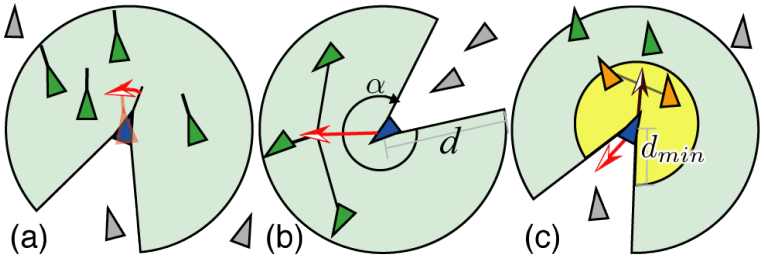


Figure 2.8: The three basic flocking urges alignment (a), cohesion (b) and separation (c) are depicted as they would influence (red arrows with two-colored head) the central agents (blue). Grey agents are out of scope, green ones are within the neighborhood vicinity and yellow ones are close enough to trigger separation. The diagrams are adapted from Craig Reynolds' website [155].

bumping into each other (Figure 2.8(c)). Some model implementations explicitly state that the separation urge acts only upon those neighbors that come closer than an individual's minimal distance d_{min}. In combination with some randomness, these rudimentary urges let large crowds of agents flock smoothly in virtual 3D spaces. Numerous artificial intelligence development frameworks provide a basic boid implementation [8, 156–158]. The respective implementations may slightly vary in respect to the acceleration urges and the system wide integration step sizes for computing the physical state changes. Our interpretations are discussed in detail in the subsequent chapters.

2.6.3 Physical investigations into swarm systems

The emergent flocking formations in Reynolds' boids simulations are of great interest to physicists that study the possibilities of symmetry breaking in stochastic systems. Considering an according simplified model (spherical perception and reduction of the number of considered urges), Vicsek et al. have shown that solely the alignment urge, explained in the last few paragraphs, suffices to break the rotational symmetry of random movements and to cause a collectively coordinated transport [159–161]. Derényi et al. discovered that such phenomena of collective transport are strongly related to the density, the size and the initial distances of the

involved agents [162]. Huepe et al. investigated the impact of scaling noise on emerging flocking formations [163]. Thereby, they could identify characteristic statistical values that allow to sort out realistic ethological models, e.g. avoiding that too many individuals simultaneously interact.

It does not take much effort to link the investigations into the interactions of large numbers of agents or particles with other classes of physical systems, i.e. those coping with statistical mechanics. *Coupled map lattices* (CML) represent an according model to study spatiotemporal couplings [164]. In analogy to cellular automata, the involved particles are arrayed on lattices that define their neighborhood couplings. CML do not restrict the involved units' states to discrete values—they may be described through high-order functions. One- and two-dimensional CML, which bring about easily readable visualizations (as in standard 1D and 2D CAs), are used to study emergent patterns in [165]. Mathematical analysis was applied to solve CMLs for the stability conditions in chaotic fluid solutions [166]. Lemaître et al., for instance, have connected CMLs and their inherent coupling strengths with collective behaviors [167]. In this research, the focus is set on the units' coupling strengths and corresponding emergent pattern formations, which directly relates to the systems' interaction topologies (Section 2.5.2).

Tanner et al. explicitly describe a variation of boids through interaction networks [168]. They consider graphs for local inter-agent perception (the *sensing network*) and for broadcasting certain information, such as the agents' velocities, via a *communication network*. In their model, the agents are drawn together and kept away from each other through energy potentials. With a fixed network topology—sensing and communication network being identical—a flocking configuration is achieved that minimizes the induced inter-agent potentials: the distances between pairs of agents are kept minimal and the individuals share the same velocity. In case of a non-fixed communication network it is instrumental that although the topology is changing, the individuals remain connected. Otherwise, stable flocking might not emerge.

In his comprehensive work on flocking systems, Olfati-Saber shows that the basic flocking

urges introduced by Reynolds can be inferred when considering inter-agent energy potentials as *stress elements of a graph* [169]. In addition to coherent velocities and inter-agent distances, he underlines the flocks' abilities of splitting, rejoining and "squeezing maneuvers" that occur when biological flocks are confronted with physical obstacles.

2.6.4 Swarm art

Emergent choreographic flocking of bio-inspired swarms have influenced a great number of art works. While spontaneous creativity of swarms is reflected in numerous paintings [170], their potential to coordinate and to show surprising vividness is, for instance, applied in automatic and assisted music generation [171]. The same features render them ideal as interacting units of interactive swarm art installations [86] that exhilarate large audiences. Khemka et al. relied on an artist to guide interactive evolution to generate *SwarmScapes*, continuously evolving virtual paintings that emerge from flocking interactions [172].

2.6.5 Evolving swarms

Kwong and Jacob implemented boid systems with homogeneous sets of individuals. Including the standard boid urges (Section 2.6.2), each individual was equipped with the following parameters [83, 84].

angle α Angle of a conic field of perception.

distance d Viewing distance of the perceptional field.

d_{min} Constant value that defines when neighbors are considered too close (originally referred to as "crowding").

max_{accel} Maximal acceleration value.

max_{vel} Maximal velocity value.

$\vec{w}$ A world center or the goal of the flock's trajectory.

w_{align} Weight to orientate towards neighbors' average velocity (alignment).

w_{coh} Weighing the drag toward the neighbors' geometric center (cohesion).

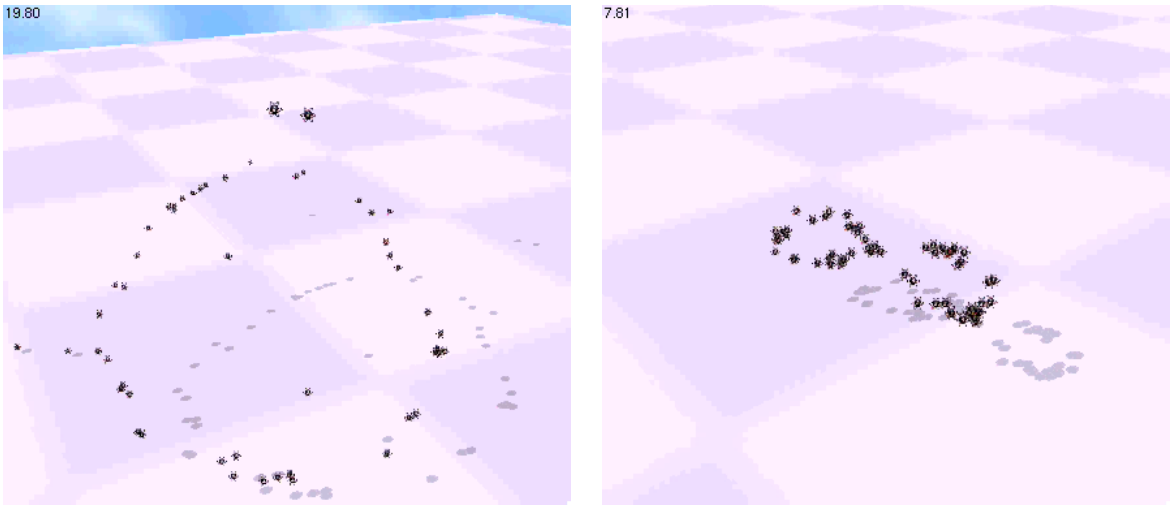
w_{sep} Weighing the urge to separate from neighbors that are too close (separation).

w_{world} Weighing the urge towards a globally defined world center (world center).

w_{rand} Weight of a randomly chosen unit-vector (randomness/noise).

The constants d, d_{min} and α determine whether an individual reacts to its peers, whereas a weighted sum of the listed coefficients w_x in combination with the respective urge vectors yields the acceleration of a boid. The boid's freedom of movement is limited by max_{accel} and max_{vel} to avoid an unrestrained growth of its acceleration and velocity, respectively. Additionally, a world center $\vec{w}$ is provided that determines the swarm agents' flight destination. Based on the listed, characteristic parameters Kwong and Jacob discovered boid configurations that exhibit emergent flocking formations (Figure 2.9). Their search was guided by means of interactive evolution.

In *SwarmEvolve 1.0* Spector et al. extended the array of weights w_x by one that activates the urge towards the closest "energy source" [173]. In their simulation, the boids fed on the offered energy which they needed in order to flock and reproduce, to outweigh collisions with peers, and to fend off confrontations with boid individuals that belong to other species. An individual's energy multiplied by its achieved life-span was used to measure its evolutionary fitness. Those individuals that ran out of energy were substituted by a genetically mutated variation of the currently best phenotype of the corresponding species. As a result of this asexual reproduction, sustaining clusters of boids evolved in which bodies of individuals protected the clusters they belonged to against other species. Only one individual in each cluster was responsible for the energy supply. The authors concluded the following.



(a) (b)

Figure 2.9: (a) and (b) show the same choreographic swarm. Due to its tendency to switch from rings to sinuous lines, Jacob and Kwong named it "Big Ring Snake". (b) depicts a typical figure-eight formation [83, 84]. The screenshots were generously provided by Christian Jacob.

> The entire feeding cloud can therefore be thought of as a genetically coupled collective, or even as a multicellular organism in which the peripheral agents act as defensive organs and the central agents act as digestive and reproductive organs.

In the same work, Spector et al. also introduced *SwarmEvolve 2.0* in which flocking agents evolve "autoconstructively". Instead of a fixed formula and a vector of parameters, the agents' reproduction and flight evolve as genetic programs [96]. Evolved programs even determine the genotypes of the agents' progeny. As a novelty, the agents obtained the capability to feed others as well. In the course of the simulated evolution, sharing food became more frequently an established practice in case of wandering food sources which resulted in an "unstable environment".

Wright et al. pursued the evolution of emergent phenomena in artificial swarms through an entropic measure Ω that captures the system's degrees of freedom in respect to possible movements [174]. They demonstrated the link between Ω and cluster formation in a very sim-

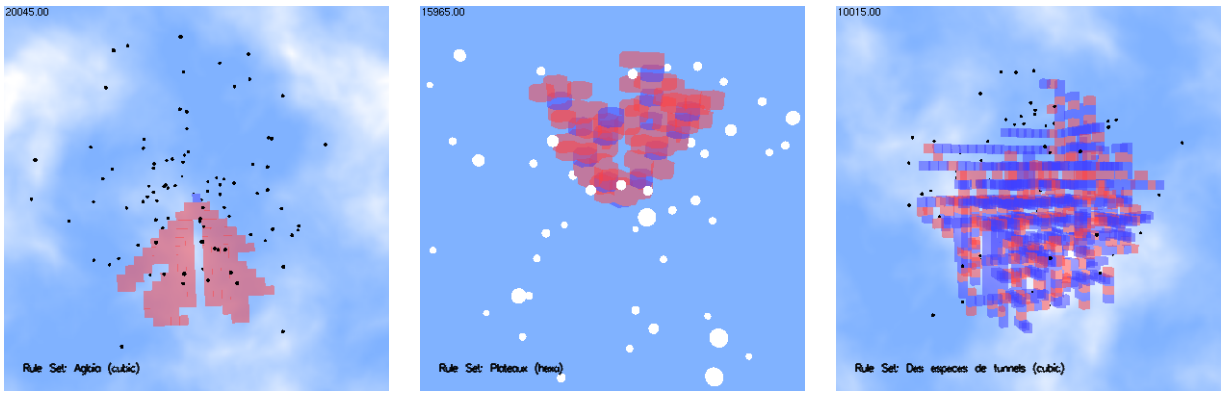
ple flocking simulation comprising 17 individuals subjected to forces based on fluid dynamics equations: a transition in Ω coincided with a phase transition in the complex flocking behavior. The authors claim that their entropic measure can be universally applied and they used it to evolve altruistic feeding behaviors, similar to the ones discovered by Spector et al. Unfortunately, the results by Wright et al. were not conclusive because they could only assume that their identified altruistic behaviors were better than some randomly evolved behaviors. Therefore, their main contribution lies in the demonstration of an integrative approach of swarm dynamics, evolution and complexity measures.

2.6.6 Constructive swarm models

In their simulations, Theraulaz, Bonabeau et al. focused on the construction algorithm as such, abstracting from any builder agents, solely activating 10 simultaneously executed construction steps [14, 25, 26]. Nest-like constructions emerged when building with two types of construction elements in a three-dimensional lattice space. In tandem, the different brick types triggered the corresponding probabilistic and qualitatively stigmergic construction rules. On the one hand, these algorithms showed that construction behaviors can be identified to retrace the architecture of numerous wasp species. On the other hand, these numerical experiments also showed that novel architectural structures can be built based on stigmergic construction processes.

Figure 2.10 (a) shows a nest-like structure that emerged in a simulation by Pilat [175]. Although Pilat mainly reproduced the results by Theraulaz et al., his implementation deviated slightly: the construction process was actually performed by randomly moving agents. This algorithmic design introduced physicality into the model that had previously been abstracted through probabilistic rules. Pilat also resumed the idea of Bonabeau et al. [25] and evolved various architectural structures, two examples of which are seen in Figures 2.10 (b) and (c).

Tower structures emerged when Jacob changed the boids simulation in the agent software

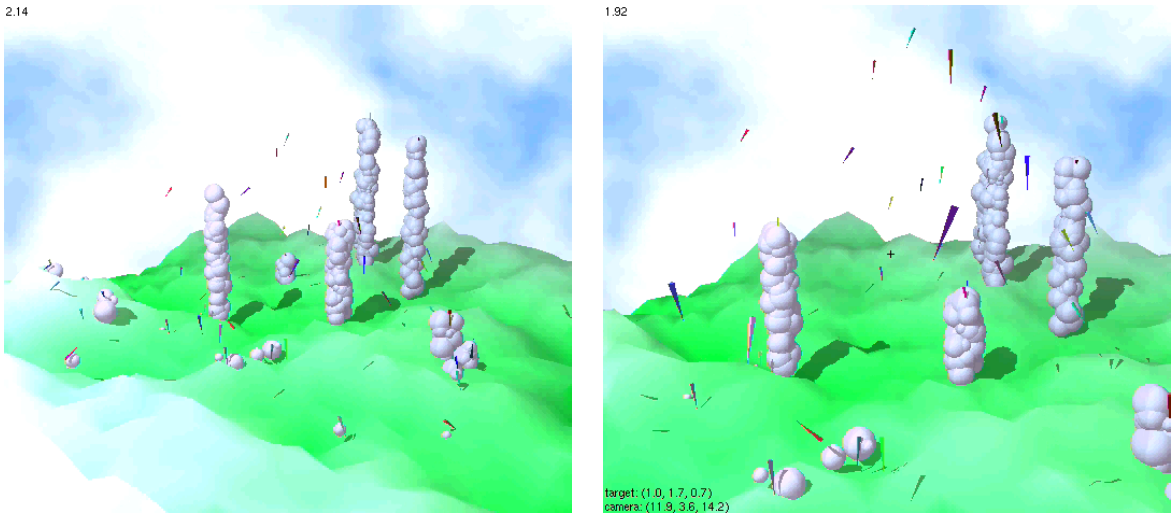


(a) (b) (c)

Figure 2.10: Pilat's rule-based lattice-swarm has successfully built a construction resembling the nest of the wasp family Agelaia [175]. The screenshots were generously provided by Marcin Pilat.

environment *BREVE* [8]. He reprogrammed the agents to follow a strong upwards urge and leave spherical particles behind (Figure 2.11). Thus, Jacob had abstracted from the biological inspiration of social insect nest constructions and presented artificial swarms as a simple computational developmental model.

In 2004, we set up evolutionary experiments to explore the architectural power of artificial constructive swarm models [176, 177]. Figure 2.12 displays two screenshots of our first constructive swarm models: Figure 2.12 (a) shows the outcome of a constructive swarm that used spherical construction elements of varying radii and colors. The attributes and placements of the particles as well as the flocking parameters of the involved agents were determined through an artificial neural network [117]. Its input nodes were fed with state and perception of the individuals and its weights were trained through interactive evolution. As the ANN considered a large number of parameters, its training drew too much upon the external breeder: the cycle of phenotype computation and evaluation took a relatively long time and had little impact on the swarms' behaviors. Therefore, another approach was designed and implemented in which rule-based constructing boid agents were evolved automatically. Their evolution was driven

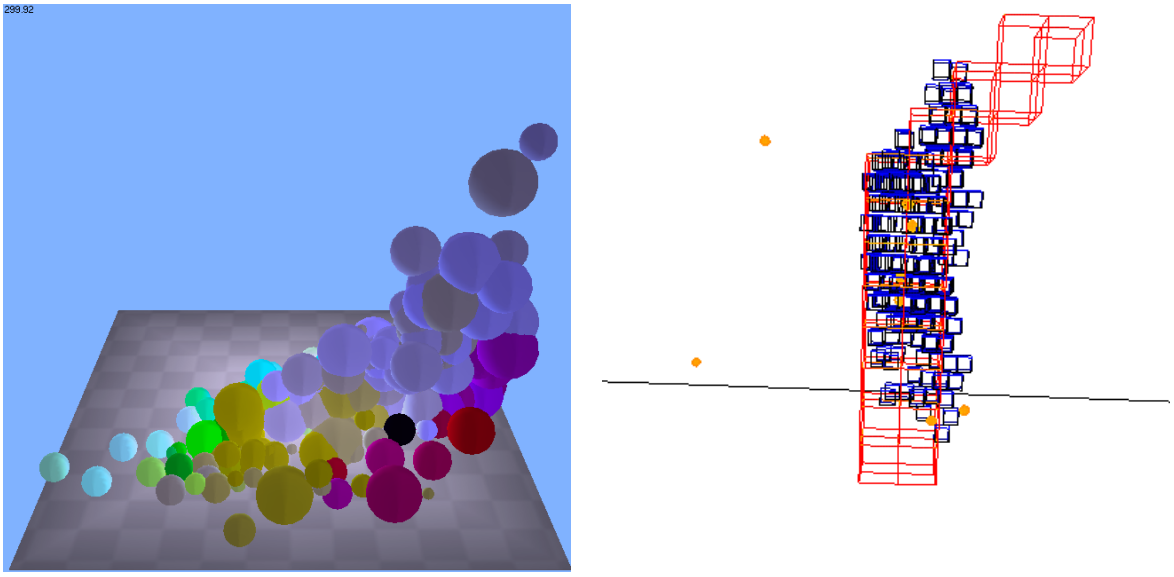


(a) (b)

Figure 2.11: Boid-like agents with a simple upwards flight behavior that pile up spherical particles.The screenshots were generously provided by Christian Jacob.

through pre-defined three-dimensional structures—the construction activities of the swarms received high fitnesses when building inside those pre-defined structures. A sample screenshot of the corresponsing experiments is displayed in Figure 2.12 (b).

In 2007, Zeng et al. presented a system to reproduce "human-like" architecture by means of evolving artificial constructive swarms [178, 179]. Based on a three-dimensional lattice space, quantitative stigmergy reigns their system. Pheromone quantities diffuse through the lattice and decay over time. The agents perceive the encountered pheromone levels as "density maps" which determine their actions of construction and movement—they crawl on the floor but can climb obstacles. Agent behaviors were bred by means of a genetic algorithm. As a fitness function, Zeng et al. also considered the difference between a pre-defined shape and the actually constructed buildings. They succeeded in re-building the contours of a rectangular house, even considering space for an entrance door.



(a) (b)

Figure 2.12: (a) The construction of an ANN-based swarm that was bred through interactive evolution [177]. (b) A pre-defined shape (the larger cubes) guided the evolution of the displayed construction (smaller cubes) of a rule-based swarm [176].

Chapter 3

Swarm grammars

Early 2006, we developed a computational model to integrate the dynamics of swarm systems [83, 84, 180], their collaborative features [14, 25, 26, 175] and the productivity of generative representations, similar to L-systems [50]. This effort resulted in the first model of swarm grammars (SGs) [1–3].

In L-systems, a formal grammar specifies rules that capture the step-by-step growth process by rewriting a string of symbols, which are subsequently translated into graphical objects through a turtle interpretation. A turtle is a virtual drawing device that is navigated in 3D space following the symbolic commands of the string. In SGs we substitute the turtle interpretation by a swarm interpretation. Instead of a single turtle following the path described by an L-system, a swarm of turtle agents interpret the grammar rules. This simple expansion from one interpreting turtle to a swarm reveals new dimensions in performance, dynamics and complexity of the resulting structures. The swarm agents are not only controlled by the grammar rules, but have the potential to interact among each other and with their environment. In fact, collision resolution among branching structures can be accounted for quite easily through parallel swarm-based turtle interpretation. This does not only lead to more interesting designs emerging from the swarms dynamics, but also engages the designer in an interactive dialog with the creative process, by introducing alternate swarms or other static and dynamic environmental components that can influence a swarms developmental processes. In the following section our early SG system is presented and tested for the effects of different configurations in respect to flocking, reproduction and interaction behavior.

Subsequently, this chapter introduces an extended SG system developed in 2008. Therein, the formulation of conditional reproduction and construction fosters the occurrences of differ-

ent types of individuals and coordinated building efforts as inspired by social insect populations [4–6]. After presenting the extended SG model, some of its creative powers are illustrated through selected, swarm grammar-based art pieces.

3.1 Basic swarm grammar system

In this section we describe the two key parts of a swarm grammar system: (1) a set of rewrite rules, which determine the composition of agent types over time, and (2) a set of agent specifications, which define agent type specific parameters that govern the agents' interactions.

3.1.1 The swarm grammar

A swarm grammar system $SG = (SL, \Delta)$ consists of a rewrite system $SL = (\alpha, P)$ and a set of agents $\Delta = \{a_0, a_1, ..., a_n\}$. The rewrite system SL is an L-system with axiom α and production rules P [52]. In the simplest form of context-free 0L-systems, each rule has the form $p \rightarrow s$, where p is a single symbol over an alphabet Ω, and s is either the empty symbol (λ) or a word over Ω. Each agent a_i is characterized by a set of attributes, which can include its geometrical shape, color, mass, vision range, radius of perception, and other parameters such as separation or cohesion urges that determine its behavior while encountering its environment. If the reproduction of swarm grammar agents is not globally timed, an according threshold might also be stored with the agent definition. The frequency of construction events to leave a trace behind the agent could also be determined globally or individually. Figure 3.1 gives an example of such a swarm grammar with two types of agents.

The rewriting process begins with start symbol A. In the first iteration of applying any matching rules, only the first rule is applicable, hence A is rewritten into AB. At the next iteration, both rules apply: A is rewritten into AB, and B is rewritten into A. The resulting string is ABA. Further rewriting will result in the following word sequence:

```
productionRule {"A"} = "AB".
productionRule {"B"} = "A".

startSymbol = "A".
```

```
separationUrge {"A"} = 0.
separationUrge {"B"} = 1.7.
wanderUrge {"A"} = 0.01.
wanderUrge {"B"} = 0.01.
color {"A"} = (1,0.5,0.5).    # pink
color {"B"} = (0.5,0.5,1.0).  # blue
```

Figure 3.1: The boxes show *BREVE* code fragments that determine an SG system with a set of reproduction rules and an axiom (left box) and sets of agent attributes (right box).

t_0 : A

t_1 : AB

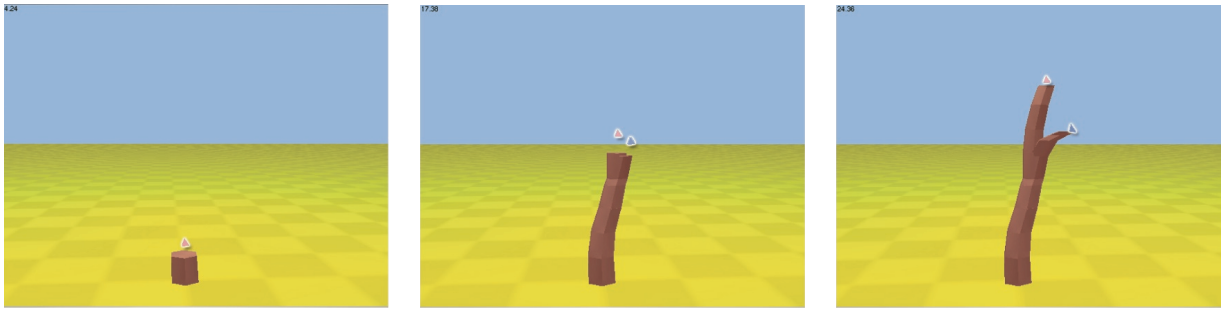
t_2 : ABA

t_3 : ABAAB

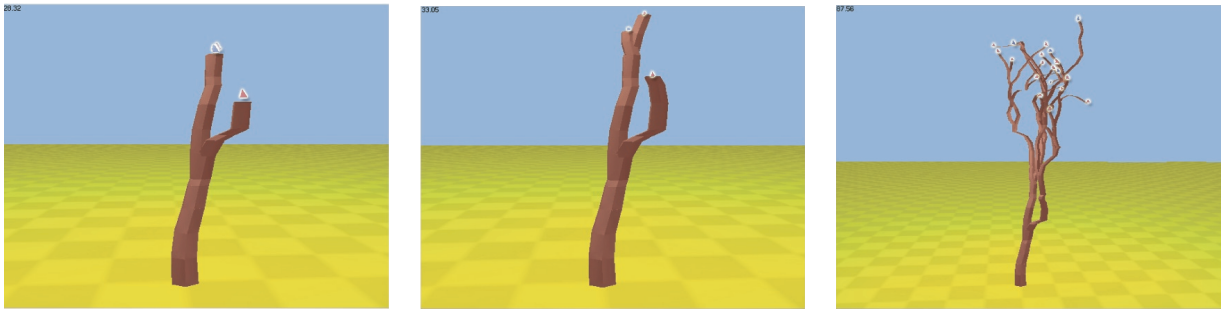
t_4 : ABAABABA

...

Here each t_i represents a decision point where an agent triggers the application of the next SL-system iteration with the string describing the current composition of the swarm. In the example above we have five type-A and three type-B swarm agents after decision point t_4. Figure 3.2 shows the first steps of the swarm interpretation in 3D space. The single type-A agent starts its vertical ascent, building a cylindrical shape on its way. At decision point t_1 agent A is replaced by a new agent of type A and a type-B agent. A-agents are the ones that move upwards, whereas B-agents build a bent branch tip (Fig. 3.2(c)). At time point t_2 agent A is replaced by agents of type A and B, and the former B-type agent is replace by an A-agent. Figure 3.2(f) illustrates the branching structure resulting after a few more iterations.



(a) $t_0 : A$ (b) $t_1 : AB$ (c) $t_1 < t < t_2$



(d) $t_2 : ABA$ (e) $t_3 : ABAAB$ (f) $t > t_3$

Figure 3.2: Step-by-step illustration of swarm interpretation in 3D space (see text for details).

3.1.2 SG agent behavior

In analogy to the boids flocking model introduced in Section 2.6.2, a swarm agent in our demonstrations is represented as a pyramid pointing in the direction of its velocity vector. Its flocking behavior is also geared to the boids model and its extension (Section 2.6.5). The boid parameters, including the description of the field of perception and the weights of various acceleration urges, are part of an agent's attribute set.

The respective acceleration urge vectors for alignment ($\vec{v}_a$), cohesion ($\vec{v}_c$), separation ($\vec{v}_s$) as well as the urge towards the world center ($\vec{v}_w$) are computed at each simulated time step according to the following assignments. At any given point in time, N_i denotes the set of neighbors perceived by boid i, $S_i \subseteq N_i$ comprises all those neighbors whose distance to i is

smaller than d_{min} and $\vec{p}_i$ denotes i's position.

$$\vec{v}_a := \frac{1}{|N_i|} \sum_{j \in N_i} \vec{v}_j \tag{3.1}$$

$$\vec{v}_c := \frac{1}{|N_i|} \sum_{j \in N_i} \vec{p}_j \tag{3.2}$$

$$\vec{v}_s := \frac{1}{|S_i|} \sum_{j \in S_i} \vec{p}_j \tag{3.3}$$

$$\vec{v}_w := \vec{w} - \vec{p}_i \tag{3.4}$$

Fluctuations are introduced into the flight patterns by adding a weighted random unit-vector $\vec{v}_r$. The acceleration vector $\vec{a}_i$ of a boid results from the following weighted sum of urges[1]. Integration of the boids' accelerations and velocities is executed automatically by the *BREVE* simulation engine using its default integration step size [8].

$$\vec{a}_i := w_{align}\vec{v}_a + w_{coh}\vec{v}_c + w_{sep}\vec{v}_s + w_{world}\vec{v}_w + w_{rand}\vec{v}_r \tag{3.5}$$

During the agents' flight, grammatical production determines the agents' transformation. That is to say, the agents can produce new agents of identical or different types, turn into different types or remove themselves. The following sections provide examples of these grammatical rules. Additionally, the basic SG model incorporates the individuals' ability to build structures by leaving construction elements in virtual space. A schematic illustration of this process is depicted in Figure 3.3.

3.1.3 Pseudocode

The connection between boids and L-Systems on the one hand and basic swarm grammars on the other hand becomes especially clear when looking at the according pseudocode. In a generic boids simulation (Section 2.6.2), two loops are executed in tandem (Algorithm 1).

[1]The basic boid flocking parameters can be read in many ways. If not applied to living organisms such as bacteria, birds, or even human beings, they could be interpreted as repelling and attracting forces of a physical or chemical nature as well.

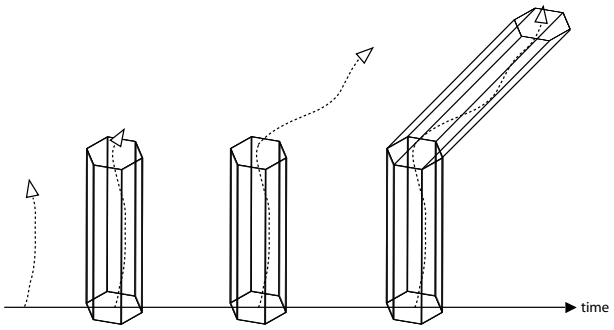


Figure 3.3: This qualitative diagram shows an SG agent building cylinders along its route.

One computes the neighborhood dependencies of a boid from which its acceleration vector is inferred. In the second loop, integration of this vector and the resulting velocity results in the boids' new locations. An implementation of a basic L-system (Section 2.2.3) generates a string similarly to a formal grammar, but through concurrent instead of sequential substitution of symbols (Algorithm 2). Afterward, the generated string is transformed into graphical structures by means of the turtle interpretation. The basic swarm grammar algorithm (Algorithm 3) merges both systems by (a) integrating the construction of graphical structures into the flight mechanism of boids and (b) by promoting agent reproduction.

Algorithm 1 Boids

Require: number of boids M, flocking parameters F, #iterations, dimensions D
  initialize M boid agents with small random velocities within D
  for #iterations **do**
    for all boids **do**
      compute neighbors N based on F
      compute acceleration based on N and F
    end for
    for all boids **do**
      update velocity according to its acceleration
      update location according to its velocity
    end for
  end for

Algorithm 2 Basic L-system

Require: axiom α, production rules P, #iterations
Ensure: development string S
  initialize S with α
  for #iterations **do**
    concurrently substitute the symbols in S according to P
  end for
  for all symbols in S **do**
    feed the given symbol graphically into a 'turtle interpreter'
  end for

Algorithm 3 Basic SG

Require: axiom α, production rules P, set of agents Δ, #iterations
  initialize agent $\alpha \in \Delta$
  for #iterations **do**
    for active agents $a_i \in \Delta$ **do**
      compute neighbors N
      compute acceleration based on N and a_i
    end for
    for active agents $a_i \in \Delta$ **do**
      update velocity of a_i according to its acceleration
      update location of a_i according to its velocity
      if construction event **then**
        add current location to geometrical trace
      end if
      if rule application event **then**
        substitute a_i according to P
      end if
    end for
  end for

3.2 Exploring the basic swarm grammar model

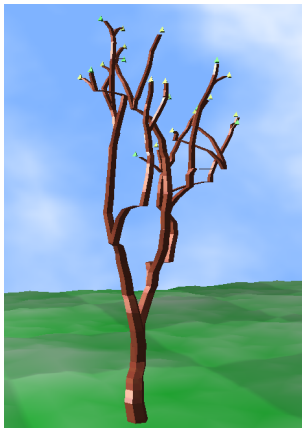
Now let us have a look at the effects that emerge when we modify the set of production rules and the agent parameters that determine their flocking behaviors. The following examples will demonstrate the high degree of interaction dynamics and the resulting variety of outcomes to be expected from swarm grammar systems that build 3D structures.

3.2.1 Changing the SL-system rules

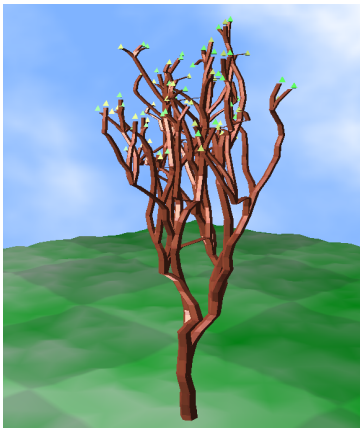
We first discuss a small sample of tree-like structures that result from various sets of production rules. In order to illustrate some of the basic effects, we use only a fairly limited number of swarm agents. We focus on three swarm agent types, A, B and C. Initially, all agents are oriented upwards, and will move towards the top (increasing their y coordinate). Some of the agent types are urged to separate and to move randomly. The corresponding flocking weights are: $w^A_{sep} = 0$, $w^A_{rand} = 0.01$, $w^B_{sep} = 1.7$, $w^B_{rand} = 0.01$, $w^C_{sep} = 13.7$, $w^C_{rand} = 0$. The remaining flocking urges (w_{align}, w_{coh}, w_{world}) are set to zero. All the generated examples ran for 80 simulated seconds.

The interpretation of swarm grammar $SL_a = (\alpha = A, P = \{A \rightarrow AB, B \rightarrow A\})$ results in a tree-like structure with sparse branches, which makes it easy to analyze (Fig. 3.4(a)). The natural look of the overall tree can be attributed to the small degree of random movements of both types of agents. A-type agents move upwards with no urge to separate, whereas any B-agent moves away from agents of type A, due to its urge for separation ($w^B_{sep} = 1.7$). Hence the arrangement of the branches is mainly a consequence of the agents interactions. Even with the grammar SL_b, the style of the tree looks similar to the structure from SL_a (Fig. 3.4(b)). A different branching pattern is shown in Figure 3.4(c), where a slightly larger number of A-agents is generated at each decision point by adding an extra A-type agent compared to SL_a. This leads to bursting agent reproductions, a more expansive growth of the branches, and the

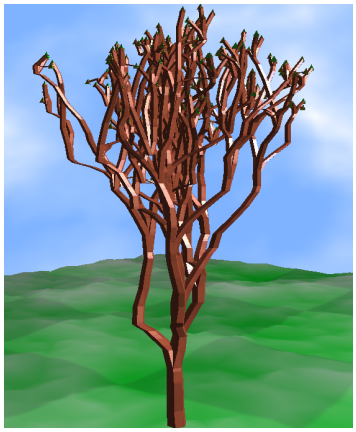
formation of a denser canopy. The small green objects at the branch tips represent the swarm agents that are still to finish their next building step. However, an increased number of generated agents does not always mean that the complexity of the emerging structures increases as well. The SL-system in Figure 3.4(d) produces a large number of agents, but the outcome is quite simple, as type-B agents only get the chance to establish a short side branch and are removed before the next building step. In Figure 3.4(e), a third agent type, C, is added, which has a very high separation urge with no random component added. As C-agents are also oriented vertically at their time of creation, they are responsible for the vertical branch endings.



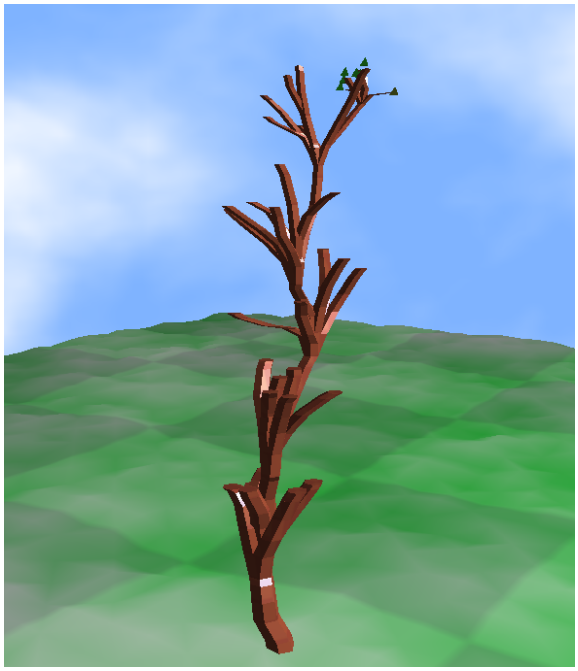
$P_a = \{A \rightarrow AB, B \rightarrow A\}$
(a)



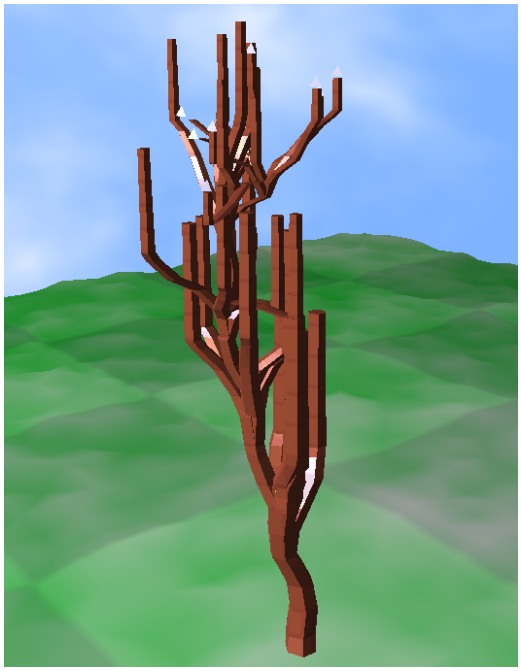
$P_b = \{A \rightarrow BAB, B \rightarrow A\}$
(b)



$P_c =, \{A \rightarrow ABA, B \rightarrow A\}$
(c)



$P_c = \{A \rightarrow BBBABBB, B \rightarrow \lambda\}$
(d)



$P_d = \{A \rightarrow BBBABBB, B \rightarrow C, C \rightarrow \lambda\}$
(e)

Figure 3.4: Examples of branching SG structures.

3.2.2 Changing the agent parameters

Instead of changing the SL-system rules, we are now going to modify the agents flocking parameters and look at the consequences with regard to the generated 3D structures. We start from a swarm grammar with a single rule that enables forked branching: $SG_{simple} = (\alpha = A, P = \{A \rightarrow AA\}, \Delta)$. At each iteration step, one type-A agent reproduces into two A-agents. As there is only one type of agents (a *homogeneous* swarm), they all share the same flocking parameters listed in Table 3.1. These settings were reported by Kwong [83] who investigated swarm interaction patterns and their evolution in more detail, see also Section 2.6.2. Figures 3.5(a), (b), and (c) show snapshots of a line formation, a ring formation, and a loose cluster emerging from the parameter sets (1), (2), and (3) in Table 3.1, respectively.

	w_{align}	w_{coh}	w_{sep}	w_{world}	w_{rand}	d_{min}	a_{max}	v_{max}
(1)	5	10	1	14	1	0.14	39	9
(2)	7	8	5	8	5	0.14	38	13
(3)	2	0	5	7	6	0.23	40	6
(4)	7	3	2	6	3	0.01	40	6

Table 3.1: Flocking parameter sets that lead to: (1) the so-called large ring formation, (2) a line formation, (3) a loose stationary cluster swarm, and (4) a messy figure eight formation [83, 84].

The bottom images in Figure 3.5 show the structures that result from using the same types of agents to interpret swarm grammar SG_{simple} as described above. The building blocks of the depicted structures bear different colors (or grey levels) so that their composition over time is visualized. Lighter-colored building blocks are built earlier. The structure in Figure 3.5(d), for example, was built from left to right, with intermittent changes of the swarms direction. This construction does not seem to involve any branching due to agent separation urges. The smooth bands originate from the agents almost perfect flight coordination while constructing very similar, almost parallel fibers. Looking a little closer, however, reveals a small gap at a U-

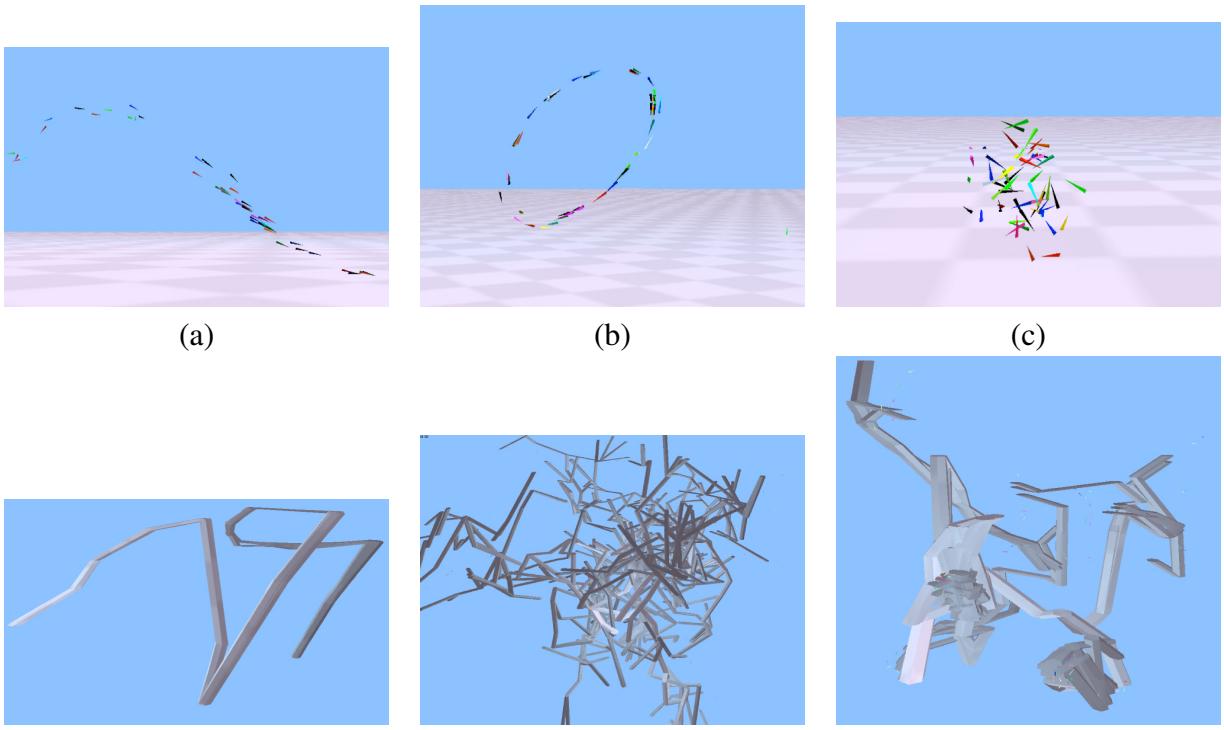


Figure 3.5: From flocking choreography (a,b,c) to the corresponding SG sculptures (d,e,f).

turn slightly off the center at the top right of the image. The structure in Figure 3.5(e) evolves spherically from a center point. The large ring flocking behavior of the swarm contributes to a spiky and impulsive character of this growing sculpture. Our third example of combining choreographic swarms with swarm grammars involves flocking behavior where the agents form loose, temporary clusters, then disperse and regroup to form new clusters at a different location. This behavior is induced by the parameters in Table 3.1(3). The formation of one of these clusters is depicted in Figure 3.5(c). Looking at the corresponding structure built by the swarm grammar agents, the sites of cluster formation are clearly identifiable as knots. Since the flocking parameters allow for a rather dynamic flight, single agents can leave one cluster and join another one at a different location.

3.2.3 Interaction with the environment

In this section we present three different kinds of interaction with both static and dynamic elements within the environment. Table 3.2 lists the parameters for the six types of agents we are going to employ.

	w_{align}	w_{coh}	w_{sep}	w_{world}	w_{rand}	d_{min}	a_{max}	v_{max}
D	0	0	0	10	10	0	30	2
E	0	10	0	1	2	10	30	5
F,G,H	10	80	0	1	4	10	10	27
I	11	33	10	5	0	1	27	2
J	10	0	80	1	4	10	10	4

Table 3.2: Flocking parameters of a set of agents.

Swarm–object interaction

Figure 3.6 shows an example of agents interacting with non-moving objects in their environment. Agents of types F, G and H tend to move towards the world center, which, in this case, is located beyond the wall and far up in the sky (like a sun). Whenever a swarm agent tries to penetrate the wall, it bounces back as its velocity vectors x- and z-coordinates are reversed. This implements a simple collision detection with static objects. As soon as the swarm structure has outgrown the wall, the agents are no more prevented from moving towards their destination. As soon as the world center becomes dynamic, its movement pattern is reflected in the construction of those swarm agents that tend towards it. In Figure 3.7 the world center orbits far up in the sky and around the y-axis of the simulation. Both agent types, D and E, are attracted towards the moving world centre. Consequently, the structure they build reflects an upward, twisted growth pattern. In order to better recognize the constructors, D-type agents are assigned a very light and agents of type E a darker color. As D-agents do not feel the urge to separate from their neighbors, they almost perfectly drive up-wards around the y-axis.

The constructions from agents of type E outgrow the ones from the D-type since E-agents are allowed a greater maximum velocity (compare Table 3.2).

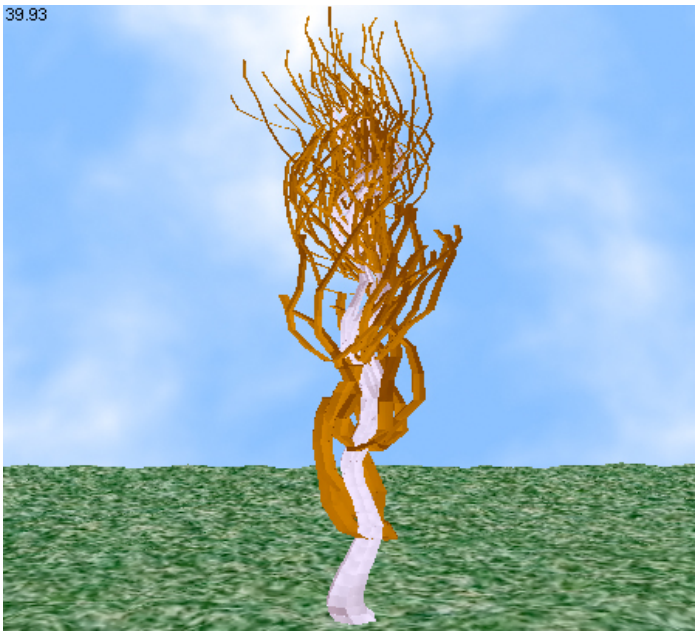
(a) $SL_a = \{F \rightarrow GHFGH, H \rightarrow G, G \rightarrow \emptyset\}$ (b) $SL_b = \{J, \{J \rightarrow JJ\}\}$

Figure 3.6: Examples of SG interaction with a static environment. (a) The SG agents are blocked from their destination by the wall. (b) Additional bricks are standing out to further impede the SG agents' progress.

Swarm–swarm interaction

In the previous examples, the swarm grammar agents were interacting with either static or dynamic objects. Now, consider a second swarm that is not part of a swarm grammar, but exhibits flocking behavior within the environment. Both swarms influence each other as soon as some of their individuals enter the field of vision of the other swarm agents. These swarm–swarm interactions are hard to capture in a screenshot. However, the swarm grammar agents witness the exertion of influence from the other swarm by leaving a trace in the 3D construction space. We look at another simplistic swarm grammar:

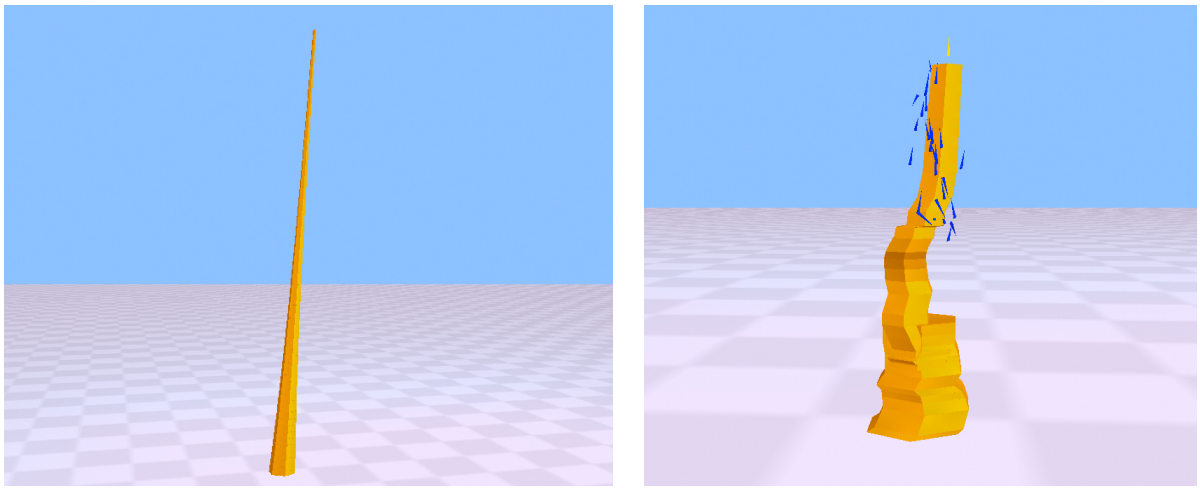
$$SG_{straight-up} = (\alpha = I, P = \{I \rightarrow I\}, \Delta)$$



$SL = \{D \rightarrow EDEDE, E \rightarrow \emptyset\}.$

Figure 3.7: Example of SG interaction with an oscillating, dynamic object.

Figure 3.8(a) shows the structure that is built by this swarm grammar, with no elements interacting with the swarm agents. The movements of the type-I agents are not driven by any randomness, so that any deviation from the presented structure has to be seen as the result of other external factors. The agent parameter settings are listed in Table 3.2. Figure 3.8(b) displays a scene where the interaction between both flocking and swarm grammar agents is still in progress. The blue pyramidal shapes represent (non-building) agents that organize their flight in a figure eight formation (parameters according to Table 3.2 and taken from [83]). As a result of the interactions between the building swarm and the flocking swarm, a completely different structure emerges. When one observes this construction during run time, the influence of the swarm grammar agent on the other swarm is fascinating to watch: as long as the swarm grammar agent is present, there is a very high probability of the other flock-mates to interact with it, as the figure eight formation usually occurs around the world center.



(a) (b)

Figure 3.8: An example of interactive SG interactions. (a) An agent of type I, which considers a simple upwards draft. (b) Agent I influenced by 30 agents that fly in a messy figure eight formation (Table 3.1).

3.3 The extended swarm grammar model

We learn from social insect swarms that when stigmergic interplay directs the collective construction efforts, sophisticated and robust buildings can emerge (Section 2.1). Thus, we have extended the basic SG model by event-based construction and reproduction rules. The activation of a behavioral rule (Figure 3.9) can be triggered by *timers*, the *perception of a specific construction element* or *swarm mate* (Figure 3.9(a)), a *pheromone*, or plain *chance*. For instance, in Figure 3.9(a) green agents (triangles) and construction elements (boxes) are within the neighborhood perception of the blue swarm agent. This agent is urged to *align* with the perceived agents' orientations (upper arrow) and to *separate* from its flock mates (lower arrow) at the same time. From time to time the agent also places construction elements along its way (blue boxes). Empty rule heads result in the unconditional application of the rule body, whereas several conditions are interpreted conjunctively. Correspondingly, all directives listed in a rule's body are executed successively. The swarm agent can change its world center to the fixed location of a construction element or template and even project it to a dynamically re-positioned fellow agent. As in the basic SG model, a rule can cause a (grammatical) swarm agent substitution, thereby reproducing, differentiating or removing the agent from the simulation. Third, instead of continuous construction, an agent places a construction element or a template as result of an according behavioral rule. Templates, like pheromones, disappear after a certain time interval and do not contribute to the outcome of the construction but help to coordinate the construction process. Templates can be evaluated *qualitatively* or *quantitatively* (see Section 2.1.1). Different basic construction elements are provided to support the building efforts. As common in architecture [102], we suggest the differentiation of *rods*, *bodies* and *layers*. Figure 3.9(b) depicts the encoding of a rule taken from an evolved swarm agent: With a probability $p = 0.5$ the agent places a template and a cubic body construction element in

space at each time step[2]. The behavioral rule displayed in Figure 3.9(c) orders the agent to produce the agents A and B and to place a construction element (rod) when it finds a construction element within its neighborhood.

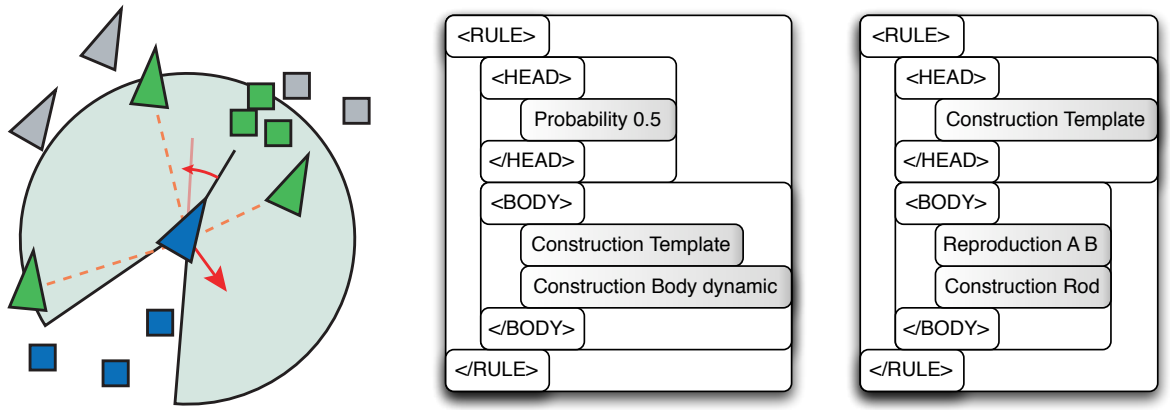


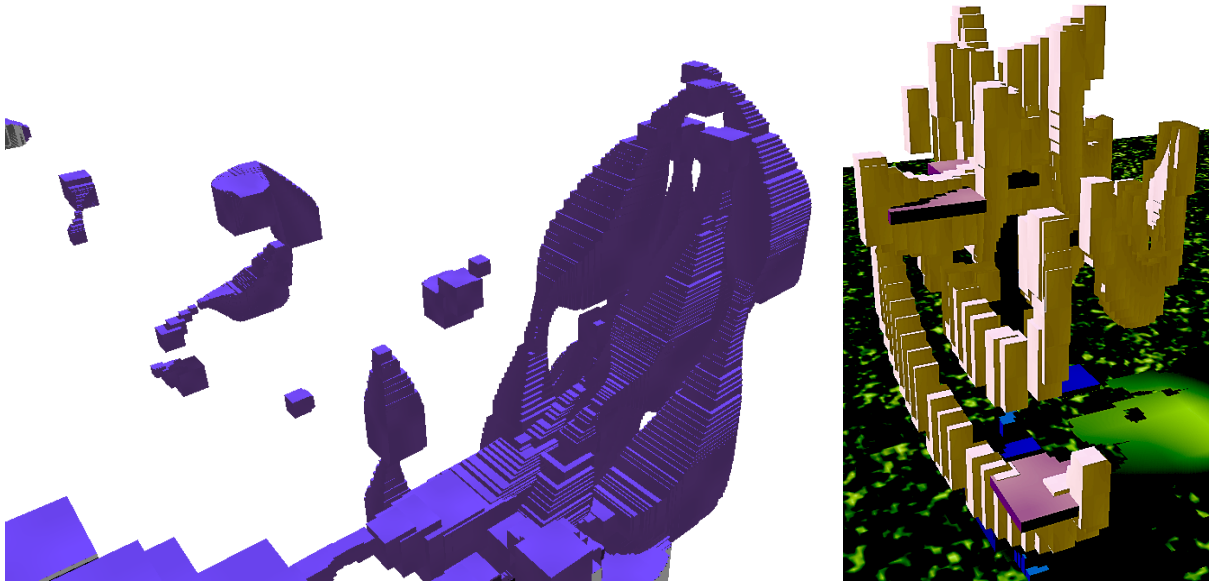
Figure 3.9: A new design of SG agent behavior. (a) All stimuli, swarm mates and construction elements, govern the agent behavior.(b) Construction of two elements is triggered with probability 0.5. (c) On sight of a template, the SG agent executes reproduction and construction.

3.3.1 Swarm grammar art

Three-dimensional sculptures made by SG systems (Figure 3.10) served as inspirational basis or integral visual and structural parts for several collaborative projects between art and computer science. The deployed SG sculptures were computationally evolved (see the next chapter) or manually designed. Figure 3.11 displays two art pieces that integrate three-dimensional sculptures grown by basic SG systems.

Stimulated by the architectural capabilities of extended SGs (Figure 3.10(b)), we combined swarm structures to create surreal, artificial worlds (Figure 3.12). In about 40 interactive evolutionary experiments, the artist bred the utilized SG structures, relying on the *Mathematica*

[2]The keyword *dynamic* that occurs in the rule in Figure 3.9 means that the body construction is rotated according to the agent's orientation.



(a) (b)

Figure 3.10: (a) An SG system that implements a pulsating size of the built construction elements. (b) An architectural design by an extended SG system furnished with several different construction element.

library *Evolvica* for the evolutionary algorithm [52] and the user interface *Inspirica* [83]. A chameleon and a bighorn sheep are immersed in two complementing artificial environments. The original SG structures are displayed in Figure 3.13.

During the evolutionary runs, we mainly pursued two objectives. Firstly, firm beams should emerge that convey robustness and can form a structural mesh, openening vast spaces by their mere existence. Secondly, fuzziness, continuity and organic looks should warrant the authenticity of the generated virtual worlds. The color gradients in the backgrounds reflect the extreme climates of the habitats of the projected animals. They also highlight the sound, wholesome, fluent structural architecture in Figure 3.12(a) and the liveliness and dynamics caught in the erratic structures of Figure 3.12(b) with 'warm' and 'cold' palettes, respectively. The decision to leave white spaces for the only elements in the paintings that do not originate from binary computations, namely the animals, can be interpreted in many ways. An exciting explanation

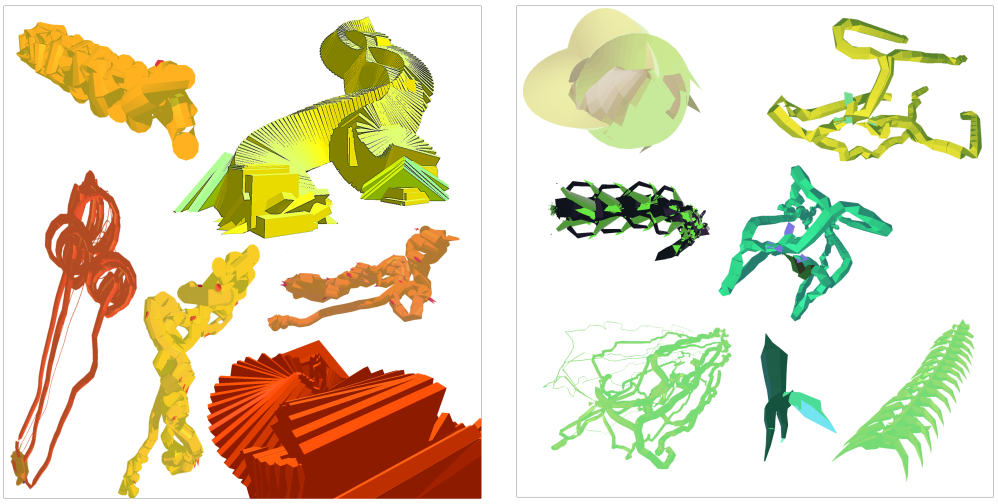
could be our early stage in computational development and that the unification of virtual worlds and reality has not yet streched out far enough to seemlessly blend.

Figures 3.14 and 3.15 show two examples in which an SG structure (left-hand side) inspired traditionally crafted and painted art (right-hand side) by the Canadian artist Joyce Wong [5]. The base of the pyramidal swarm construction (Figure 3.14) is *rigid* and shines in *cold, steel-blue* colors. In contrast, tentacle-forming swarms *wind* from its peak. The *explosive polarity* of the pyramid inspired the piece [Manifest] that places the swarm pyramid into a new context, in which "unrequited thoughts seek ways to escape" (12" x 24" black gesso, acrylic on Masonite). Painted layers were scratched away to reveal the raw Masonite surface. Soft swipes led to a semi-transparent reflection of the rigid pyramid foundation. Energetic cuts at the pyramid's top lend the painting real structure. In the first panel of the diptych [Outlining Blues] (Figure 3.15) "a distortion of the swarmettes formulates a Whale-like specimen as it swims in a peaceful surrounding. The second panel leaps backward in time and depicts first organisms coming into existence" (both 12" x 24" oil paint and rusting agents on metal).

Figure 3.11: Two art compositions made from basic SG structures.

(a) (b)

Figure 3.12: Diptych of the two pieces (a) “caméléon” and (b) “bighorn sheep”. Acrylic medium on canvas, 23” x 38”.

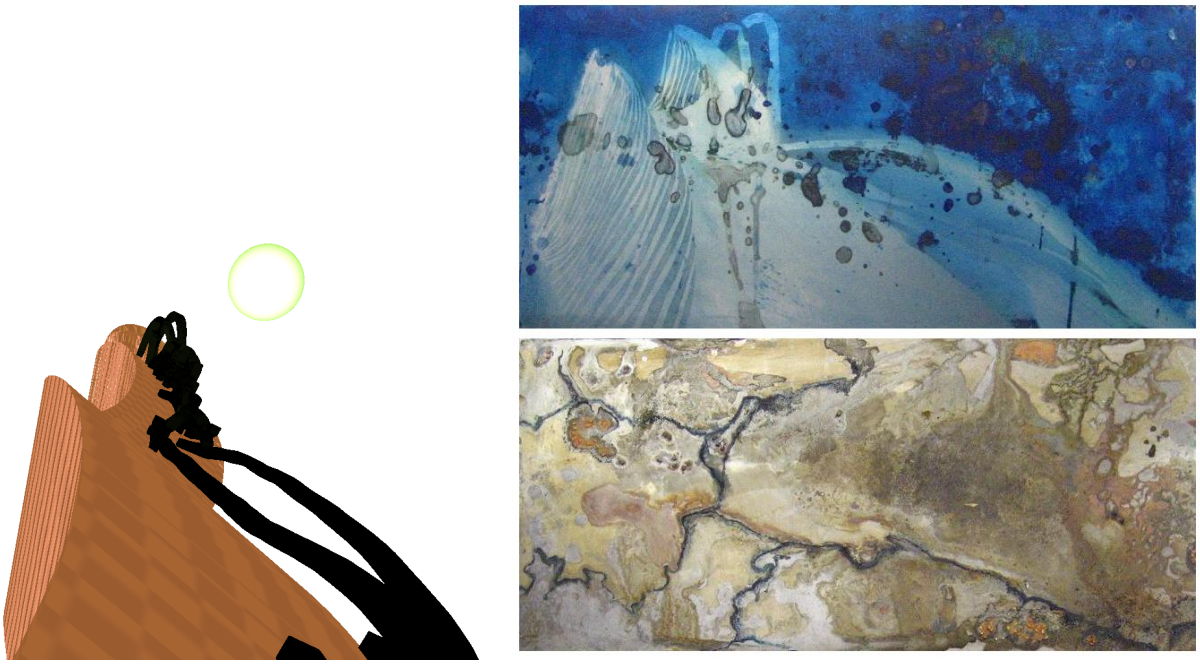


(a) (b)

Figure 3.13: Selections of SG structures bred for the diptych displayed in Figure 3.12. They are arranged similar to their appearance in the paintings.

(a) (b)

Figure 3.14: (a) Swarm grammars growing a pyramidal structure inspired (b) the artwork 'Aftermath'.



(a) (b)

Figure 3.15: (a) Swarm grammars built on the rapid interplay of black outlining agents and orange stem growth inspired (b) the artwork 'Outlining Blues'.

Chapter 4

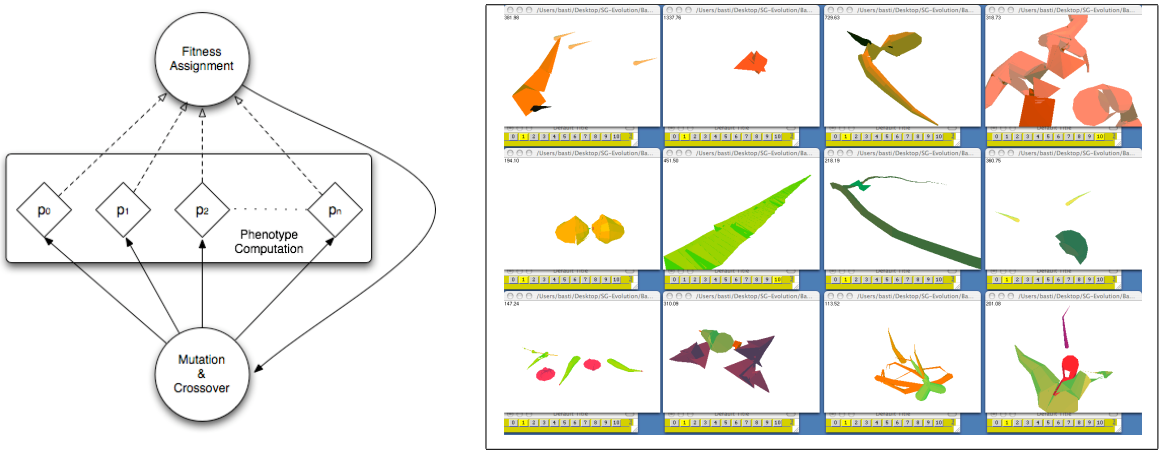
Breeding swarm grammars

A swarm grammar system usually consists of several agent types, each one comprising ten flocking parameters and possibly large numbers of behavioral rules. Visual features of the emerging structures might require additional bits of information. The entirety of the resulting configuration data in combination with the corresponding emergent interaction processes allows for a vast variety of swarm grammar structures. In order to explore a broad variety of structures, we rely on *evolutionary computation* (see Sections 2.3 and 2.4).

For furthering the swarm grammar model, an approach of interactive evolution has proven useful [2]. This approach and the results are presented in the first section of this chapter. Afterwards, explorations of an immersive breeding approach are described [3]. This chapter concludes with recently implemented ideas to automatically evolve swarm grammar structures [4].

4.1 Interactive evolution

We use an extension of Inspirica [83, 84], one of our evolutionary design tools, to explore the potential of generating swarm grammar systems that exhibit intriguing constructions. As illustrated in Figure 4.1, a collection of swarm builder simulations is simultaneously presented to the user. Figure 4.1(a) illustrates the concept of interactive evolution schematically. The phenotypes p_0 to p_n are computed separately and manually inspected and rated. The assigned fitness values determine the selection probabilities of the genetic operators: mutation and crossover. Arrows illustrate the flow of information, whereas dashed lines represent visual inspection. Figure 4.1(b) depicts the Inspirica GUI for interactive evolution. Each window shows the in-



(a) (b)

Figure 4.1: (a) The concept of standard interactive evolution. (b) The Inspirica [83] user interface helps to evolve swarm grammars.

terpretation (phenotype) of a different swarm grammar configuration (genotype). The windows display the construction processes as they occur. All designs are true objects in 3D space, hence can be rotated, zoomed and inspected in various ways. After assessment of the twelve presented structures, the external breeder assigns fitness values between 0 and 10 to each solution, and proceeds to the next generation. The number of presented specimen was set in accordance with the monitor size. In the given case, 12 windows fit comfortably on a 20" cinema screen, still revealing enough details to the breeder to quickly evaluate the structures in each window. We also successfully ran versions with 9 and 15 windows for 12" and 23" monitors, respectively. The assessment values 0 to 10 are provided by Inspirica as the default evaluation scale. We did not change this setting as it complies with the breeder's desire for a wide range of choices but does not present an overwhelmingly large variety of values.

Inspirica is the GUI for interactive user evaluation of the presented specimen. *Evolvica*, on the other hand, is the evolutionary computation engine, also written in Mathematica, that operates on the genetic material and its associated fitness values. To integrate swarm grammars into Evolvica, the rewrite rules and agent parameters are represented as symbolic expressions,

so that genetic programming (GP) can be used to evolve both the set of rules as well as any agent attributes [52]. The following paragraph depicts the template structure used by Evolvica to generate expression patterns for swarm grammars.

```
SwarmGrammar[
_AGENT,_AGENT,_AGENT,
_RULESET,
_SEEDS
],
RULESET[_RULE,_RULE,_RULE,_RULE,_RULE],
AGENT[
_WANDERCONST,
_WORLDCENTERCONST,
_MAXVEL,
_MAXACCEL,
_RGB,
_WORLDCENTERURGE,
_NEIGHBORHOODSIZE,
_NEIGHBORHOODRADIUS,
_NEIGHBORHOODURGE,_NEIGHBORHOODURGE,_NEIGHBORHOODURGE,
_ENERGYLOSS,
_ITERATIONSTILLDRAWING,
_ITERATIONSTILLBRANCHING,
_SCALE,
_NUMBEROFEDGES
],
RGB[_RGBR,_RGBG,_RGBB],
```

```
WORLDCENTERURGE[_WCX,_WCY,_WCZ],
NEIGHBORHOODURGE[
_AGENTSYMBOL,
_VELOCITYCONST,
_SPACINGCONST,
_CENTERCONST
],
RULE[_HEAD,_BODY],
HEAD[_AGENTSYMBOL],
BODY[_AGENTSYMBOL,_AGENTSYMBOL,_AGENTSYMBOL],
SEEDS[_SEED,_SEED,_SEED,_SEED,_SEED],
SEED[
_AGENTSYMBOL,
_SEEDX,_SEEDY, _SEEDZ
]
```

Developmental rewrite systems are usually rather sensitive to changes in the genotypes which can result in vastly different growth structures and developmental processes. Therefore, for the examples we present here, only context-free rules with a maximum string length of three ($|s| = 3$) are applied. We allow at most five rules per SG-genotype. GP mutation and crossover are the only genetic operators. In addition to the values that determine the agents' neighborhood perception (_NEIGHBORHOODSIZE and _NEIGHBORHOODRADIUS) and the weights to accommodate the basic flocking urges (_WANDERCONST, _WORLDCENTER-CONST, _VELOCITYCONST, _SPACINGCONST, _CENTERCONST), there are several values that specify the construction geometry and visualization of an SG agent. Before explaining those values in detail, we want to emphasize that in the given model an agent makes a distinction regarding the types of its neighbors. Since we limit the amount of agent types to three,

an agent is equipped with an according set of _NEIGHBORHOODURGES for each possible agent type. _ENERGYLOSS defines the amount of energy an agent looses at each computational step. The initial amount is fixed but the metabolism can differ individually, allowing an agent to live for an arbitrary time-span. _ITERATIONSTILLDRAWING and _ITERATIONSTILLBRANCHING determine the numbers of iterations that have to pass until a construction event or a rule application event are triggered (Algorithm 3). _SCALE is a factor to define the relative size of the constructed cylinders, whereas _NUMBEROFEDGES specifies their shape. Five axiomatic SEEDs are initialized at the beginning of an SG simulation. Since all the seeded agents are placed on the ground, only two dimensions (x and z in the given case) have to be randomly chosen ($y = 0$). The following paragraph shows the detailed value ranges of the according parameters:

```
wanderConstRange={0,1};
worldCenterConstRange = {0,1};
worldCenterRange = {-1000,1000};
maxVelRange = {0,25};
maxAccelRange = {0,40};
iterationsTillBranchingRange = {20,150};
iterationsTillDrawingRange = {15,30};
rgbRange = {0,1};
neighborhoodSizeRange = {50,150};
neighborhoodRadiusRange = {2, 6.28};
neighborhoodUrgeRange = {-2, 2};
energyLossRange = {0,0.25};
scaleRange ={0,2};
numberOfEdgesRange = {3,13};
xzSeedRange = {-50,50};
```

```
agentSymbolsRange = {0,3};
```

Figure 4.2 shows selected examples of such evolved structures which reveal the potential of form generation through SG systems. Example (a) shows pointy yet smooth conic nodes that connect with long thin branches. The structure in (b) resembles a flower-like structure. Whenever ramifications in the structures tend to thin out, it is due to an internal loss of energy that the building agents experience and which determines the diameter of the building blocks. The individual energy consumption is part of the agents' attribute sets and may consider flight, reproduction and construction. (c) reminds of an organismic structure with growing tips. Example (d) shows the interwoven 3D pattern created by several groups of spinning and whirling swarm agents. The strong contrast of thin, darker pins and bulkier construction elements in the somewhat symmetric structure (e) triggers the idea of a functional design, e.g. for grasping. (f) and (i) are two more instances that directly evoke associations with common plants—pepper and roses in the presented cases. In (g) an architecturally interesting design is shown that was also utilized in an arts collage [5]. Most characteristic for the structures (h) and (k) are the symmetric separation urges that exercised on the building agents and the tandem of lean and voluminous cubic construction elements. Due to the continuous and interconnecting construction elements in (j) and (l) both structures obtain an 'organic' look.

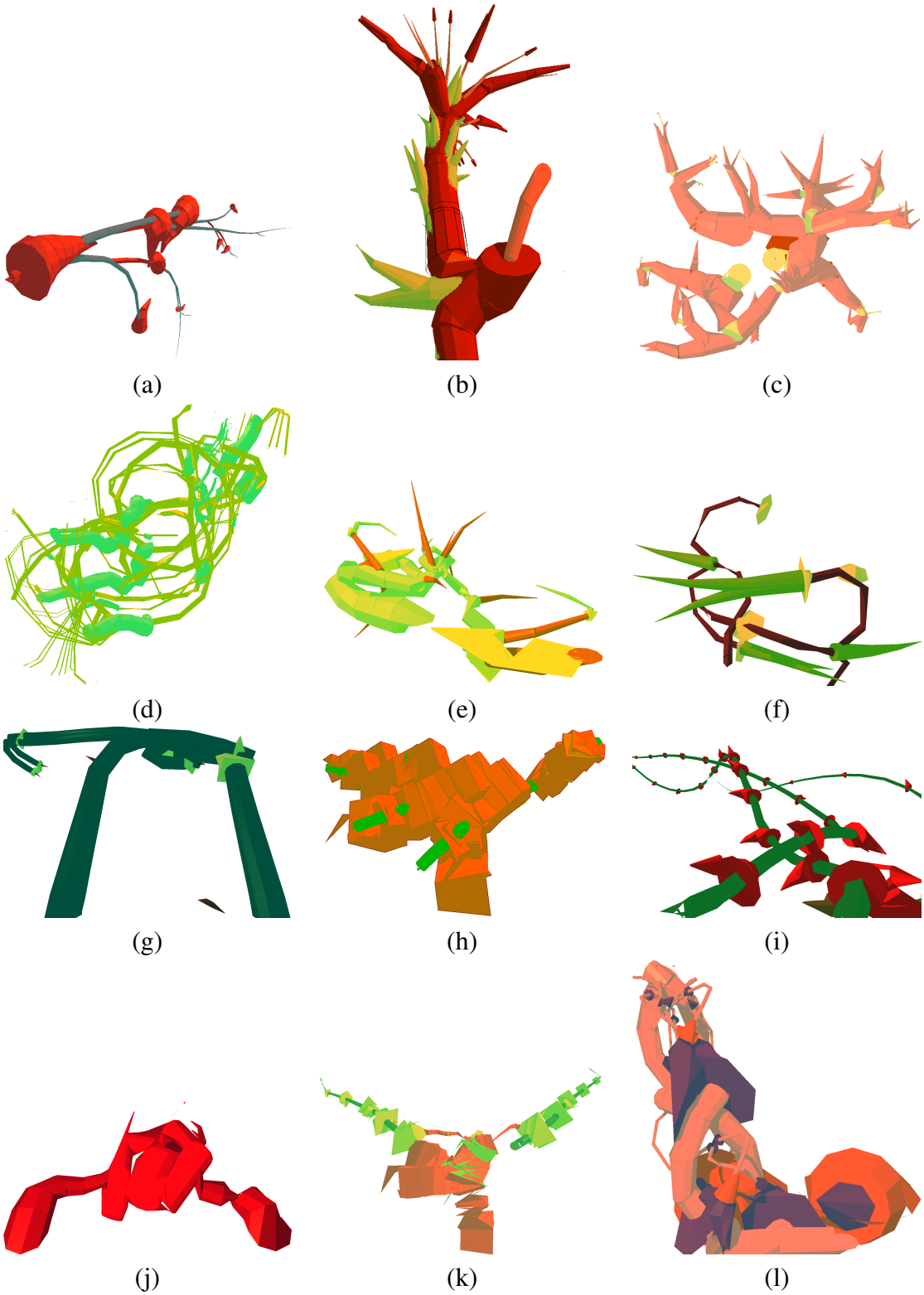


Figure 4.2: Examples of computationally evolved swarm grammar structures.

4.2 Immersive evolution

The isolated swarm grammar phenotypes (Figure 4.1(b)) develop independently of each other. Their individual breeding spaces as well as the interface of the supervisor can be integrated into one comprehensive virtual environment.

A single breeding ground has several advantages such as the ability to concurrently oversee large numbers of phenotypes (Figure 4.3) or to flexibly experiment with the provided specimen. In a co-existing and co-evolutionary setup, the encountered phenotypes can be the result of massive interactions of swarm agents. Thus, one can identify robust swarm grammars that generate stable phenotypes that thrive in isolation and in densely populated environments.

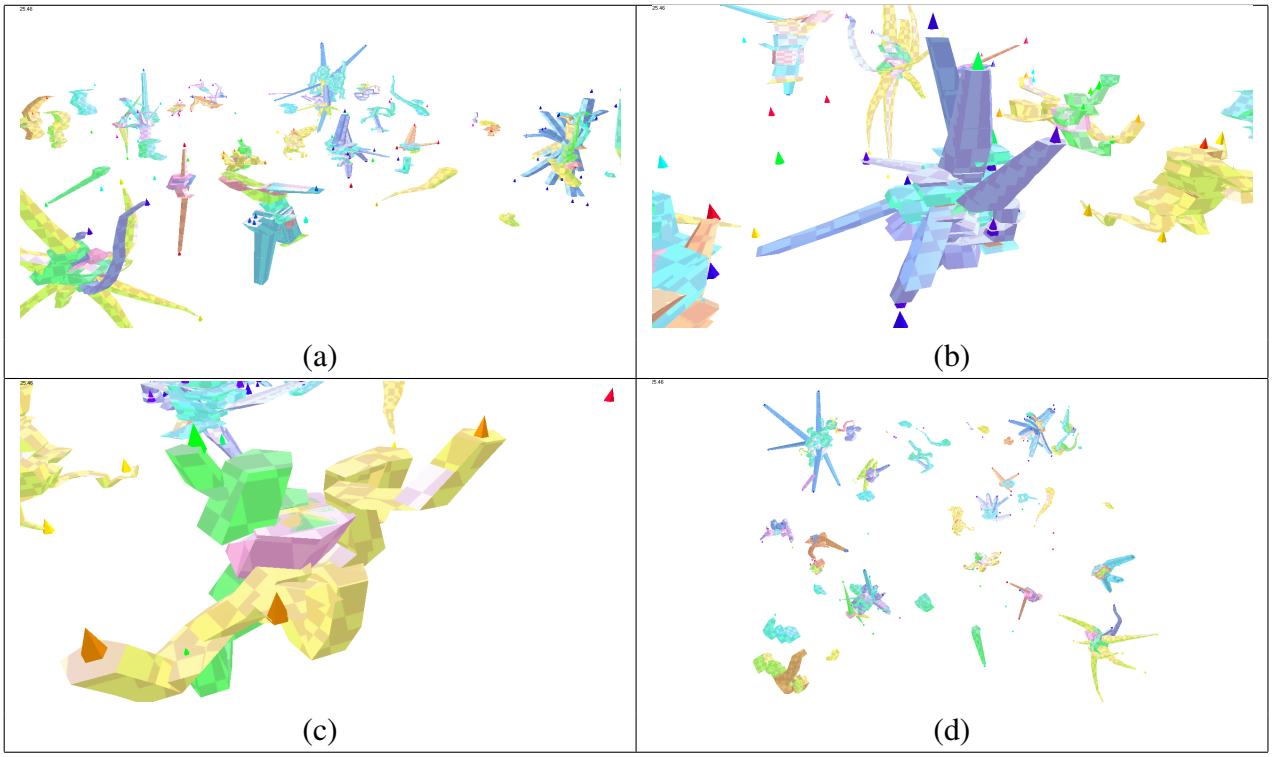


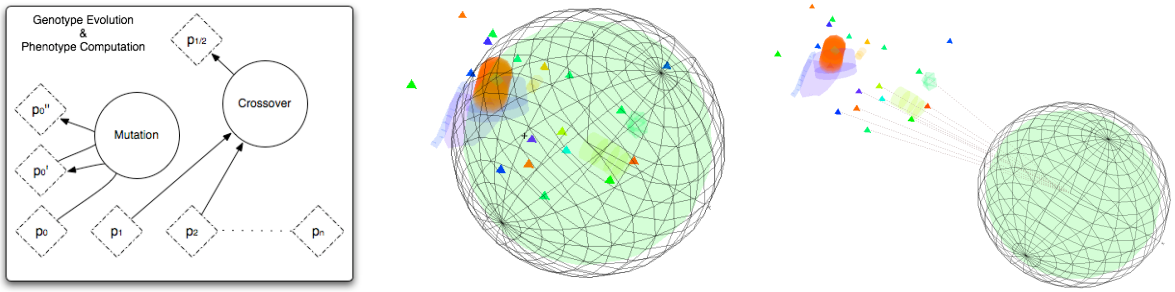
Figure 4.3: Screenshots of an exploratory trip into immersive breeding grounds. In (a) the external breeder is browsing through a population of SG specimen. (b) and (c) show close-up impressions of SG structures that are under development. (d) provides an overview of the breeding ground.

4.2.1 Spatial breeding operators

Figure 4.4(a) schematically illustrates an immersive evolution scenario. It integrates the computation of the phenotypes, p_0 to p_n, as well as the evolutionary manipulation of the underlying genotypes. In the diagram arrows depict the flow of genetic material, induced by spatial breeding operators.

Our user interface for an according immersive evolutionary scenario integrates two aspects: (1) visual representation and (2) intuitive manipulation by an external breeder or designer. The visualization interface enables moving, rotating, and zooming of the camera, or saving and restoring specific views and scenario settings. Most of these procedures are already incorporated in the agent software environment *BREVE* which we use as the primary visualization and simulation engine in our breeding experiments [8].

In addition to aspects of visualization, the supervising breeder is equipped with tools to select, group, copy, and move swarm grammar agents, thus being able to influence the course of evolution within the emerging scenario. The set of possible manipulations also includes mutation and crossover operators to manually trigger changes of the genotypes that encode the swarm grammar rules and the agent parameters. Any of these operations can be performed through a *breeder volume* that can be created and moved in 3D space to enclose several swarm grammar agents, see Figures 4.4(b) and (c). Once selected, the agents' states or properties can be manipulated, or genetic operators can be introduced. Swarm agents that pass through a volume (a sphere in this case) can be influenced in various ways. We use breeder volumes for the crossover and mutation operators, for moving and copying swarm agents, and for boosting their energy levels. Analogous to the watering of plants fitness evaluations are only given implicitly by providing more energy to selected groups of agents. Having selected a subset of agents, the breeder volume can still be moved in space. The previously established selection of agents is visualized by a set of corresponding, connecting lines (Figure 4.4). Through this mechanism, spatial operations, such as relocation of the agents, can also be performed.



(a) (b) (c)

Figure 4.4: (a) Schematic diagram of an immersive evolution approach. (b) A breeder volume to select and manipulate a subset of SG agents. (c) Previously enclosed agents remain associated with the breeder volume.

4.2.2 The swarm grammar gardener

Figure 4.5 illustrates how a breeder can influence the emerging building processes within a simple ecology of swarms. In Fig. 4.5(a) two swarm agents have built a cylindrical structure with a side branch. Both agents, which have run out of energy, are still visible at the top left and to the right of this construction. In the next step (Fig. 4.5(b)) a breeder sphere is introduced so that it encloses the agent on the right. Through a contextual menu, this agent is 'revived' by replenishing its energy reservoir. Subsequently, the agent resumes its building process, generates an additional side branch and extends the overall structure further to the right (Fig. 4.5(c)). A similar procedure is applied to the agent on the left. It is captured by the breeder sphere and triggered to first replicate, i.e., make copies of itself, and then resume construction (Fig. 4.5(d,e)). This generates further expansions of the structures and—after further energy boosts (Fig. 4.5(f))—results in the structure depicted in Figure 4.5(g). The pattern continues to grow until the agents run again out of energy.

This is only a simple example of how external manipulation by a breeder, the 'gardener', can influence the agent behaviors, the building or developmental processes. Their evolution as agents can change their respective control parameters during replication. Agents of a specific

type share a swarm grammar, but agent groups can be copied as well, so that they inherit a new copy of their own swarm grammar, which may also evolve over time, either automatically or through direct influence from the gardener. Due to the large number of growth processes that are taking place concurrently, detailed observations of the developments are not easily possible. Consequently, the breeding decisions in immersive space are strongly influenced by subjective perceptions. Once the seemingly interesting specimen are isolated, systematic experiments can be conducted. To further the immersive breeding approach, we therefore suggest a focus on usability and practicability for follow-up implementations. For instance, we recommend an automatic storage database for genotype samples with an interface for instantaneous and easy access from within the virtual space. There should also be a vast number of breeding grounds available so that selected specimen can be copied and isolated independently from the ecology of their first appearance. Furthermore, graphical visualization (conceptual or textual) of the operations performed on the genotypes could be helpful.

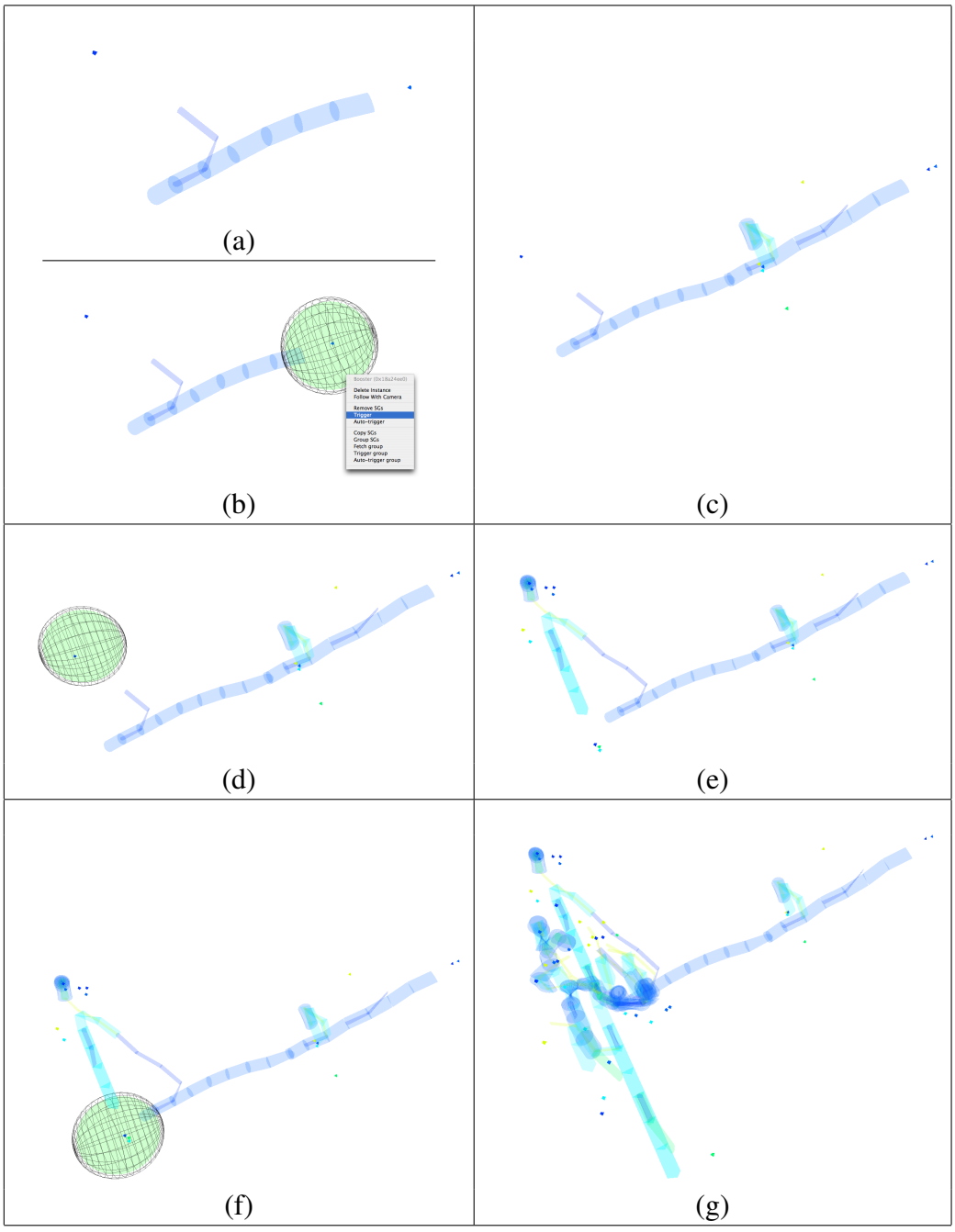


Figure 4.5: Illustration of interactive manipulation of SG agents by an external breeder.

4.3 Automatic evolution

Once a concrete design task is chosen for a swarm grammar system, certain constraints on the growing structure can be formulated. Afterwards, the power of innovation of evolutionary computation can be harnessed to automatically create assortments of SG designs. In addition to the analysis of the genotype of a swarm grammar (also referred to as the code or the configuration of the SG), fitness assignment can be mainly performed on two other levels: (1) the construction processes and (2) the emerging structures. Structural analysis is either very course grained, considering for example the overall volume and the proportions, or computationally very costly, for instance when attempting to identify hierarchies and re-occurring modules. Therefore, we put an emphasis on the observation and classification of the construction processes.

In particular, we promote productivity, diversity and collaboration and prevent computational outgrowth. In order to measure productivity the SG constructions are compared with pre-defined structures. Diversity is determined as the ratio of possible agent types and actually expressed agent units, as well as the according ratio of deployed construction materials or construction mechanisms. In order to foster collaboration between the SG agents, we protocol the average ratio of perceived neighbors - too low values imply that no interactions are taking place, whereas too great values mean that the agents are trapped within small spaces. As the basic swarm grammar model can be extended in accordance with the respective application task, SG agent collaboration may also be reflected in the amount of effective communication between the agents.

Randomly initialized swarm grammar systems can quickly exhaust the provided computing powers: Fast, possibly unconditional sequences of SG rule applications may result in exponential agent reproduction rates. Temporarily such explosions of activity could be beneficial, for example in designs integrating large numbers of ramifications. In the long run, however, such

an outbreak of computing requirements has to be stopped. As a simple means to prevent destructive outgrowth, yet allowing for temporary leaps of activity, we suggest setting a time limit for the construction process and filtering out inefficient SGs during the evolutionary experiments. A concrete implementation of the suggested automatic evolution process of SGs is presented in the next chapter.

Chapter 5

Swarm grammar architecture

An implementation of the constraints mentioned above led to a variety of aesthetically innovative *architectural idea models* [4]. Traditionally, these models are the first three-dimensional realization of an architectural idea, still omitting details of its actual construction. In regards of rewarding productivity of a swarm grammar, we measured the emergent architectural structure against a solid cube. Construction elements built inside the pre-defined cubic shape contributed positively to a SG's fitness, whereas constructions outside the cube diminished it [176]. We count the events of successful stigmergic communication in SGs to foster coordination of the agents' activities. In particular, we consider it a successful communication event whenever a construction or reproduction rule of an agent is triggered by a stigmergic stimulus.

5.1 Evolutionary setup

First, we describe how swarm grammars are encoded and modified during the evolutionary process of this approach. Second, we explain details of the process of fitness evaluation that directs the evolutionary search for architectural idea models.

5.1.1 Genotype and GA

In our breeding experiments we evolved populations of 20 swarm grammars over at least 30 generations. Each swarm grammar comprises 5 swarm agent types which are described by their flocking parameters [13, 83] and by sets of at most 10 behavioral rules as described in Section 3.3. As genetic operators we apply *fitness proportionate selection*, *elitism*, *mutation* and *crossover*. How the two latter operate on the swarm grammar genotypes is explained in

the following paragraphs. The genotypes are encoded in tagged lists of key-value pairs, as illustrated in Figure 3.9.

Only one fourth of the next generation is subjected to mutation which is applied with a 50% chance to each gene. Numerical values are changed in accordance with a normally distributed maximal step size of 0.2. Hereby, the following intervals are considered: The flocking weights for *cohesion, alignment, separation*, for the *world center* urge and for the *random* urge are normalized to values between 0.0 and 1.0. An agent's mobility is limited to the absolute values 15 for velocity and 30 for acceleration. Its perception extends at most to a 5.0 radians radius and reaches at most ten units. The separation urge impacts an agent's acceleration only if its neighbors are not further than 5.0 units [83]. On mutation, rule conditions and actions are equally likely generated anew, deleted, or inserted. In the generation of conditions and actions each available directive, e.g. 'Change focus' or 'Reproduce', are chosen with the same probability. If a directive requires a parameter, it is chosen randomly as well. For instance, if 'Construction' has been determined as new directive, 'Rod', 'Body', 'Layer' and 'Template' are equally likely chosen as its parameter.

$5/8th$ of the next generation of swarm grammars result from recombination of the parents. Each behavioral rule and the lists of flocking parameters that appear in the genotype of a swarm grammar are used for recombination. Here, too, we apply the operation on each considered gene with a 50% chance. An alternative crossover implementation considers only the agents of a swarm grammar for recombination. Elitism transfers the fittest eighth of the parents to the next generation.

Individuals that are not assigned a fitness value greater than zero are considered extinct. If the parent population is diminished, the genetic algorithm generates an equally reduced population of successors. However, the population is automatically filled up with newly generated swarm grammars. This mechanism counterbalances the negative influence the genetic operators can exercise due to incomputable rule sets.

5.1.2 Fitness evaluation

At the beginning of a simulation all M swarm agents[1] of a swarm grammar are expressed and initialized around the center of the virtual space (up to 10 units in x and y direction). Close by, at the bottom center of the pre-defined shape, at coordinates $(5, 5, 0)^T$, a template appears hinting at an ideal spot for construction (Figure 5.1(a)). For a specified period of simulated time $\Delta t_{sim} = 8sec$ the swarm agents coordinate, build and reproduce. Then the construction process is stopped, all data is written into a file and the next swarm grammar is computed. The fitness is evaluated based on the goals to *limit computational and constructional outgrowth* and to *promote production, diversity* and *collaboration*. We will explore these constraints in more detail in the following sections and then propose an according fitness function.

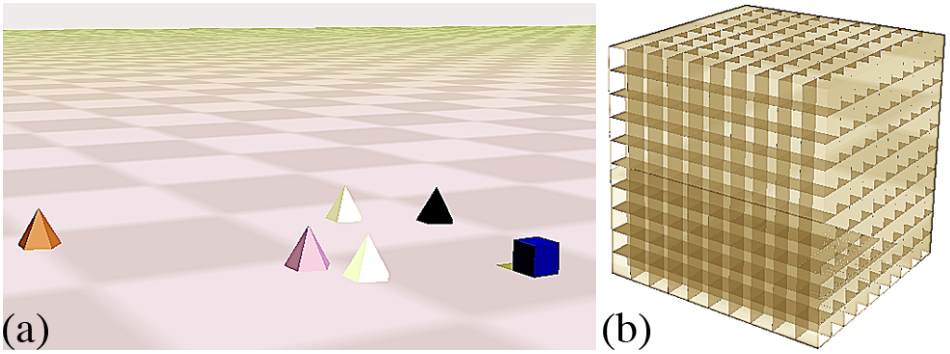


Figure 5.1: (a) Initial simulation state: 5 agents (polygons) are randomly placed in the vicinity of a template (cube). (b) Emerging structures are compared against this pre-defined shape consisting of 10^3 small cubes.

Preventing escalation

Uncontrolled agent reproduction can quickly lead to an exponentially growing demand for computing resources. In order to avoid such an excess of resource usage, a simulation process taking longer than 100 real seconds is terminated and is not considered for further evolution. Additionally, the actual computing time t_{real} for a swarm grammar is stored as a variable

[1] For our experiments $M = 5$.

in order to determine a specimen's fitness. On the one hand a certain degree of complexity in the emerging structures is desirable. On the other hand an outgrowth of (computational) complexity has to be avoided.

The same idea is realized when the swarm construction is compared against a pre-defined shape at the end of the simulation: constructions within a certain range of the target template are rewarded, whereas outgrowing the pre-defined limits is unproductive. Hereby, as in [176], the pre-defined shape consists of smaller cubes with edge size 1.0 (Figure 5.1(b)). So at the end of the simulation, at $t_{sim} = 8$, we determine the ratio r_p between the number of these cubes that are penetrated by construction elements versus the total number of cubes comprised by the pre-defined shape.

Adding the ratio r_p to a swarm grammar's fitness value rewards the swarm's productivity but only within certain boundaries. From a different perspective, it promotes constructions that retrace the provided pre-defined shape. Independently of the pre-defined shape, the total number n_c of construction elements that were deployed during the whole simulation process can be utilized to further assess productivity and to limit the extent of construction as well.

Promoting diversity

Since the construction patterns of individual swarm agents may vary, a broad diversity in constructions can be expected, that are built by a large number of different swarm agents. Even a homogeneous set of swarm agents can achieve greater diversity than a single swarm individual, as (1) the agents can influence each other's behavior, and (2) the same construction processes can be conducted in parallel. We express these observations numerically by r_a, the ratio of different agent types that are activated throughout the course of a simulation to the total number of available agent genotypes, and by m_a, the number of agent individuals that were registered during the simulation at one point or another.

Whenever different types of construction elements (rods, layers, bodies) are employed, an

increase in structural diversity can be expected. As a consequence, the ratio r_c of employed construction element types that are deployed during the simulation to the number of available types is also considered for fitness computation.

Fostering collaboration

As an alternative to the deployment of construction elements, swarm agents may drop templates that do not contribute to the construction and last for a short period of time only (for 20 iterations in the presented simulations). Consequently, time-critical signals can be propagated through templates, thus promoting collaboration among the swarm agents. We therefore also measure the ratio r_t of created templates to those that actually trigger a behavioral rule. Again, the value stored in the constant r_t results from counting and integrating occurrences of the observed phenomena throughout the course of the simulation, i.e. between the beginning of the simulation at $t_{sim}^{start} = 0$ and its end at $t_{sim}^{end} = 8$ simulated seconds.

Swarm interaction is based on each agent's awareness of other agents. Therefore, we compute the ratio of agents that see each other to the total number of agents at each computational step. This value is averaged over the course of the simulation and assigned to the variable r_n. For larger r_n the swarm agents stick together, whereas smaller r_n values reveal a very loose flight pattern—both of these extreme situations render collaboration difficult. For instance, a swarm grammar with $r_n = 0.87$ might form a clump as seen in Figure 5.2(a), whereas $r_n = 0.08$ can be an indication for uncoordinated growth as seen in Figure 5.2(b). Further studies on the course of neighborhood relations during swarm simulations are described in Chapter 6.

Proposed fitness evaluation

The factors explained above are taken into consideration by the following fitness assignment for a swarm grammar, f_{SG}. The terms g_n, g_c and g_a transform the corresponding variables to normalized values between 0.0 and 1.0 according to their semantics: A neighborhood ratio not

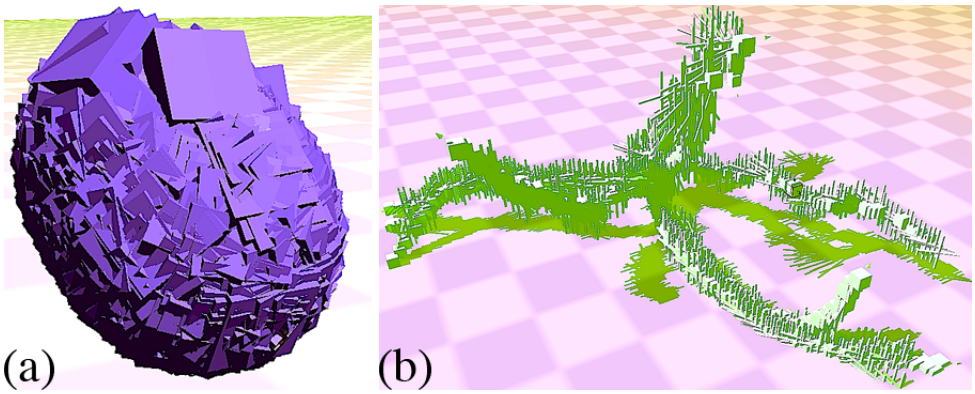


Figure 5.2: Neighborhood perception r_n during the construction process can sometimes be linked to the emerging structures. (a) A very compact structure emerges with $r_n = 0.87$. (b) Swarm agents drift away from each other, which yields a low perception rate $r_n = 0.08$.

too close to 0.0 or 1.0 is presumably beneficial. Reasonable amounts of expressed agents and placed construction elements are contributing to the fitness as well, especially, if these efforts do not overly extend the computation time t_{real}. We therefore arrive at the following fitness assignment of a swarm grammar which is used in our experiments.

$$
\begin{aligned}
f_{\text{SG}} &:= r_p + r_a + r_c + r_t + g_n + \frac{g_c + g_a}{\sqrt{\max(t_{real}, 1)}} \\
g_n &:= \sin(\pi \cdot r_n) \\
g_c &:= \sin(\pi \cdot 0.005 \cdot \min(n_c, 200)) \\
g_a &:= \sin(\pi \cdot 0.005 \cdot \min(m_a, 200))
\end{aligned}
$$

5.2 First results of bred SG architecture

A successful search for architectural idea models heavily depends on the effectiveness of the genetic algorithm, especially on the crossover operator and on the fitness evaluation. Therefore, we first discuss our findings about the influence of the operator and fitness function on the

resulting architectural constructions, before a variety of phenotypes is presented and analyzed.

5.2.1 Fitness evolution and crossover points

Figure 5.3 depicts representative graphs of the fitness evolution in two independent experiments. In the first experiment we apply a crossover operator $c1$ on rules and sets of flocking parameters only. Based on the components of a swarm grammar system as depicted in Figure 5.4(a), $c1(i, j)$ exchanges the genes Cx and R_y^z among two selected swarm grammars SGi and SGj. In the second experiment each of the swarm grammars' M agents is considered a gene for recombination (crossover operator $c2$). Referring to Figure 5.4(a), $c2(i, j)$ exchanges the modules Ax only. The number of agents in two SGs as well as the number of rules of two agents might vary. In this case, instead of an even exchange of genetic material, a one-sided transfer from the specimen with the greater number of genes takes place. The average and the maximum fitness values of each generation are shown in the graphs avg_c1 and max_c1 in regards to $c1$, and in avg_c2 and max_c2 in regards to $c2$, respectively.

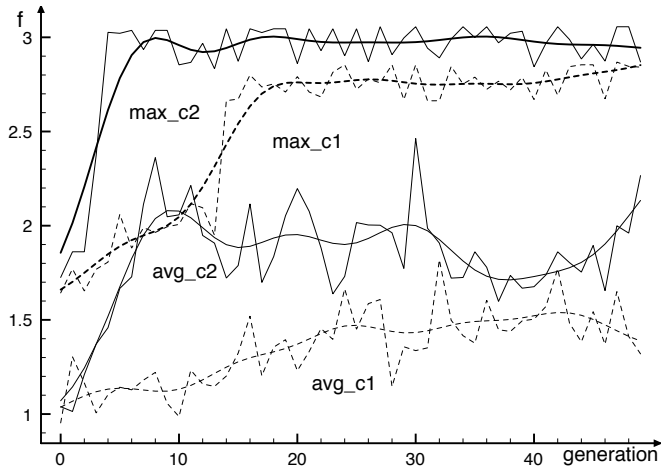


Figure 5.3: Fitness evolution in two experiments implementing different crossover operators. The upper graphs represent the maximal fitnesses achieved in each generation, whereas the lower graphs depict the average fitnesses of each generation.

Elitism ensures that the best individuals are transferred unchanged into the next generation.

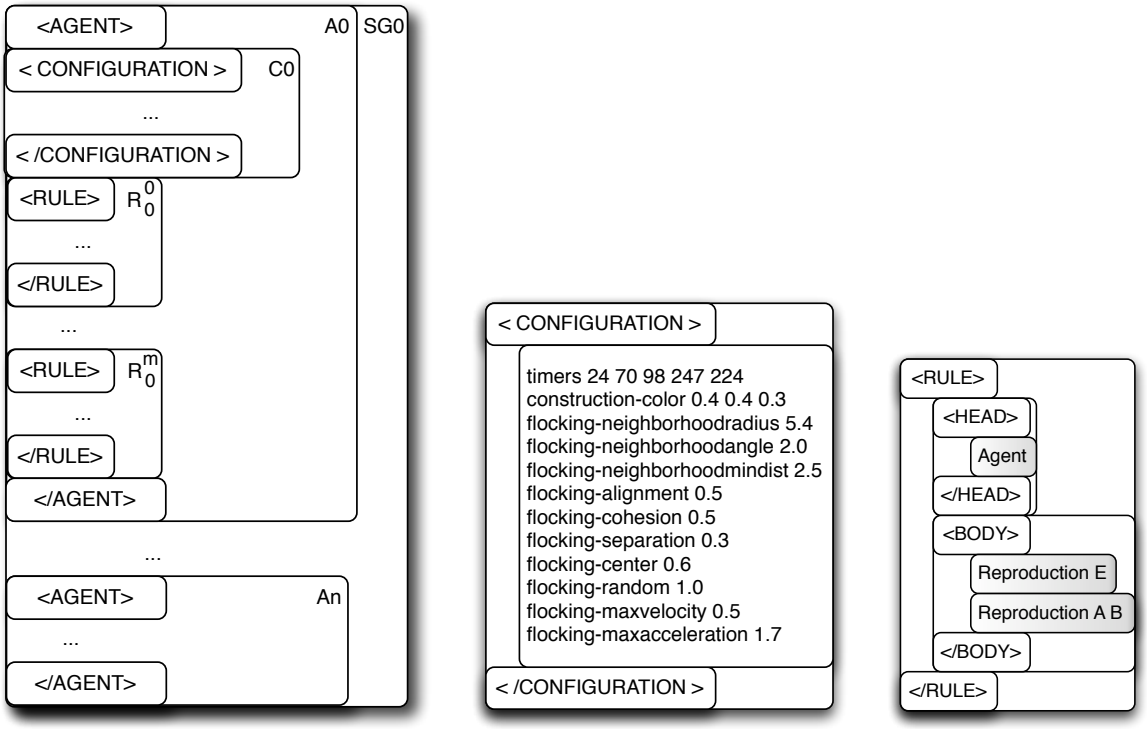


Figure 5.4: (a) The modules of a generic SG system, $SG0$, comprising $M0 = n + 1$ agents, $A0...An$, each defined by a configuration module, $C0...Cn$, and of a set of behavioral rules, e.g. $A0$ has m rules $R_0^0...R_0^m$. (b) The configuration of the operative agent that wraps rods around the skeletal structure in Figure 5.10. (c) The spawning rule to delegate construction, employed in Figure 5.10.

Noise in the sequence of maximum fitness values (max_c1 and max_c2) is due to randomness in the simulations. As shown, max_c1 usually rises slower but does not differ much from max_c2. The development of average fitness values is of particular interest. The tendency of avg_c1 to stay considerably below avg_c2 is not a coincidence. If only agents of relatively successful swarm grammars are exchanged, the offspring's success mainly depends on the agent interaction encoded in their behavioral rules. Underachievement and thereby extinction can happen, but is less frequent than with recombination working on the building blocks of the agents' genotypes. Especially the exchange of behavioral rules can lead to a swarm grammar's quick extinction. As soon as the agents reproduce themselves too frequently, the computing

time rises and can easily exceed the maximum allowed timeframe. While the average fitness of the crossover on agents achieves a better development, the other crossover operator leads to a population of much greater diversity. On the one hand, the recombination possibilities are much greater when genetic information on the agent behavior level is considered. On the other hand, the high extinction rate allows new genotypes to enrich the gene pool.

In our experiments certain fitness properties were obtained faster than others. r_p, g_n, g_c and g_a had fast and great impact on f_{SG} and, consequently, on the evolutionary development. We were not able to promote rising values for r_a, and r_c which mostly exhibited erratic changes, or for r_t which did not contribute at all. Consequently, the following, simplified fitness assignment might have sufficed to breed the presented examples.

$$f_{\mathrm{SG}}^{\mathrm{simple}} \quad := \quad r_p + g_n + \frac{g_c + g_a}{\sqrt{\max(t, 1)}}$$

Also, since the ‘tasks‘ that correspond to the ineffective variables seem too difficult to be learned instantly, either partial task fulfillment (e.g. first, the placement of a template and second, the response) should receive a reward. Alternatively, the generation of behavioral rules could be constrained, thereby reducing the search space for ‘useful‘ rules.

5.2.2 Architectural designs

The outlined experimental setup results in a wide variety of architectural designs, a selection of which is presented in the following paragraphs. We differentiate between three structure categories depending on the actual construction elements: rod, body or layer. This classification schema concurs with actual architectural categories [102]. Additionally, we introduce a category for swirly architectural idea models. Figure 5.5 displays an architectural model that we did not want to assign to any specific class of models, as it shows an example that integrates all basic construction elements equally well.

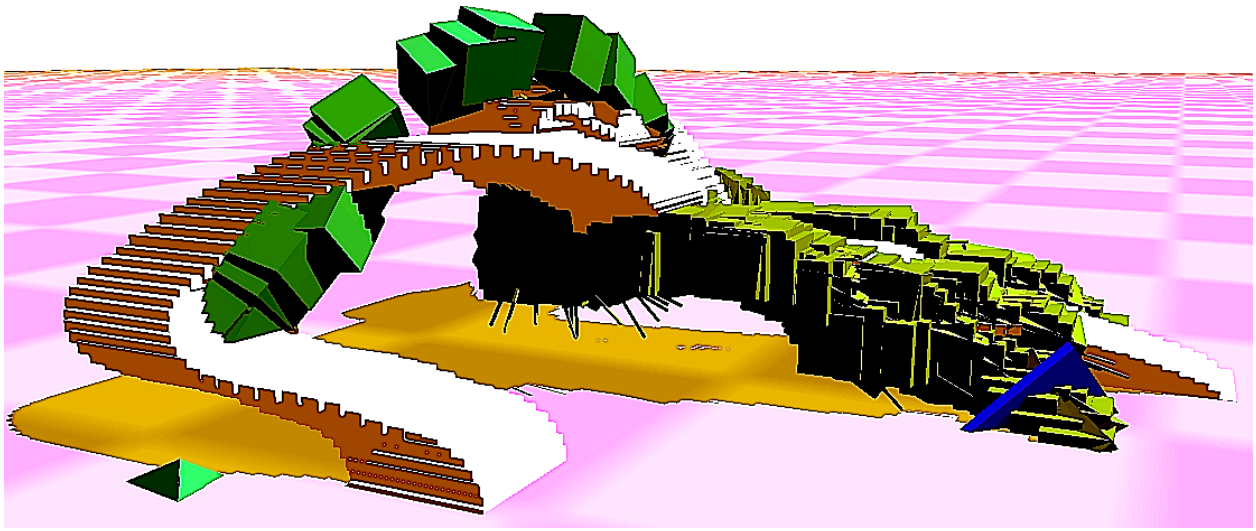


Figure 5.5: An architectural idea model built with equal presence of all three basic construction elements.

The discussion of the examples underlines that the mapping from a swarm grammar genotype to the corresponding structure is not trivial. The provided characteristic measures drive the evolutionary process, yet one can hardly infer specific architectural categories from these measures, as we will demonstrate. The presented architectural models are created by swarm grammar systems that roughly share the following percentages of different kinds of rule conditions to trigger construction, reproduction and communication:

- 44% unconditional,
- 18% probabilistic,
- 15% on template sight,
- 14% on agent sight,
- 10% timers.

The presented swarm grammars' tracing success value r_p and neighborhood perception r_n are

listed in Table 5.1.

Model	r_p	r_n	Model	r_p	r_n
Fig. 5.6(a)	0.89	0.16	Fig. 5.8(a)	0.77	0.51
Fig. 5.6(b)	0.59	0.50	Fig. 5.8(b)	0.71	0.30
Fig. 5.6(c)	0.98	0.41	Fig. 5.8(c)	0.85	0.54
Fig. 5.6(d)	0.19	0.43	Fig. 5.8(d)	0.53	0.19
Fig. 5.7(a)	0.30	0.28	Fig. 5.9	0.81	0.35
Fig. 5.7(b)	0.32	0.005	Fig. 5.10	0.55	0.52
Fig. 5.7(c)	0.80	0.10	Fig. 5.11	0.89	0.64
Fig. 5.7(d)	0.75	0.16			

Table 5.1: Characteristic values of the presented swarm grammar architectures.

Rod architectures

Figure 5.6 shows four examples of constructions in which rods dominate their visual character. In fact, the structure depicted in Figure 5.6(c) only comprises about 60 rods, a mere 3% of the employed construction elements. The remaining three architectures, Figure 5.6 (a), (b) and (d), however, are based on 50% to 60% rods. Investigation of the genotypes reveals that the rod-architecture swarms' behaviors are not synchronized through timer conditions. Otherwise, however, they comply with the general behavioral conditions outlined above.

The four phenotypes displayed in Figure 5.6 show diversity. Figure 5.6(a) exhibits three completely different segments, that were grown from right to left. The first segment does not only mix cubic construction elements and elongated rods, but also mixes two colors. The second one resembles a spiky armor, and the third segment looks like a pile of sheets. The continuous development of many of the presented swarm grammar architectures aligns the lineage of morphogenesis and reproduction of swarm grammar individuals with sequentially arranged construction elements. The analogous, linear emergence of the architectural model in Figure 5.6(b) is again expressed in three segments. From the bottom-left of the image a lattice tail loosely connects to the main part of the model. From there on, rods are laid out horizontally

resembling stairs that lead to the top of an impenetrable spherical heap of rods. Figure 5.6(c) shows a multifarious construction. Cubic elements are arranged at the bottom and the top. They are interconnected with a densely packed, dynamically shaped hose. Rods are floating in a wave-like fashion around the model's peak. The model in Figure 5.6(d) embodies the swarm dynamics of the construction process. The movements of the flocks of swarm agents create the impression of dynamic parts. This vivid impression is supported by the rough looking combination of layers and rod elements.

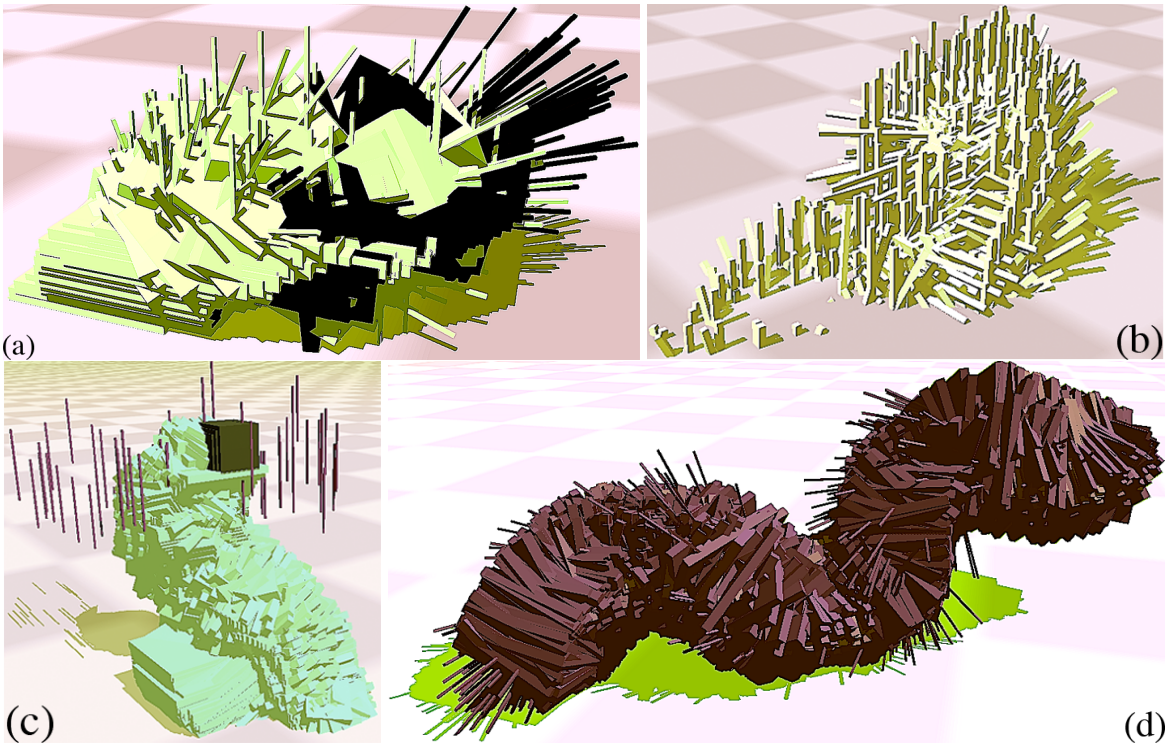


Figure 5.6: Rod-based architectural idea models.

Body architectures

Figure 5.7 presents architectural idea models that are mainly assembled of (cubic) body construction elements. In fact, their share of all utilized construction elements varies between 30% and 50%. Simple as it might appear, the construction in Figure 5.7(a) achieves a very good ap-

proximation of the pre-defined shape (Table 5.1) and grows an extended set of bodies and rods on top of the bottom-up sequence of layered elements. The second example, Figure 5.7(b), is distinct by its sparse use of different construction elements. An interplay of flocking swarm agents is obviously not required for the displayed model, as an extremely short perception radius of 0.6 units (maximally 10.0) keeps the swarm agents' neighborhood perception very low (Table 5.1). Figure 5.7(c) presents a futuristic design that emerges through three interwoven construction mechanisms. (1) Cubic body elements form the main part of the model. (2) Layers flank the main part along the entire edge length. (3) Both layers and cubic body parts are rising in tandem to complete the construction with an elevated, inclined platform. The construction rule shown in Figure 3.9 belongs to an agent involved in the construction of Figure 5.7(c). In fact, another rule makes the same agent differentiate upon sight of a body construction. The last instance of body-based architectures is shown in Figure 5.7(d). Here, a dynamic character is introduced into the otherwise rather strict body architectures as seen in Figure 5.7(a), (b) and (c).

Layer architectures

Figure 5.8 displays four tower constructions that are coined by the employment of layer construction elements. In fact, Figure 5.8(c) only utilizes 5 layers that can be spotted at the bent, the remaining 99.95% of the model consist of rods and body construction elements. Figure 5.8(a) and (b) look very similar. Yet, they originated from completely independent experiments. Their characteristic values, too, resemble each other, except for the neighborhood ratio r_n (Table 5.1). Figures 5.8(a) and (b) consist of 25% and 17% layers, respectively. During both construction processes, agents transform/reproduce 15 times. Their visual resemblance is striking: From the bottom a rather rigid and straight stem is drawn upwards for about $3/4th$ of the total height. Then, body construction elements rise to a podium that is ornamented by several rods. During the construction of Figure 5.8(d), agents reproduce 76 times. The increas-

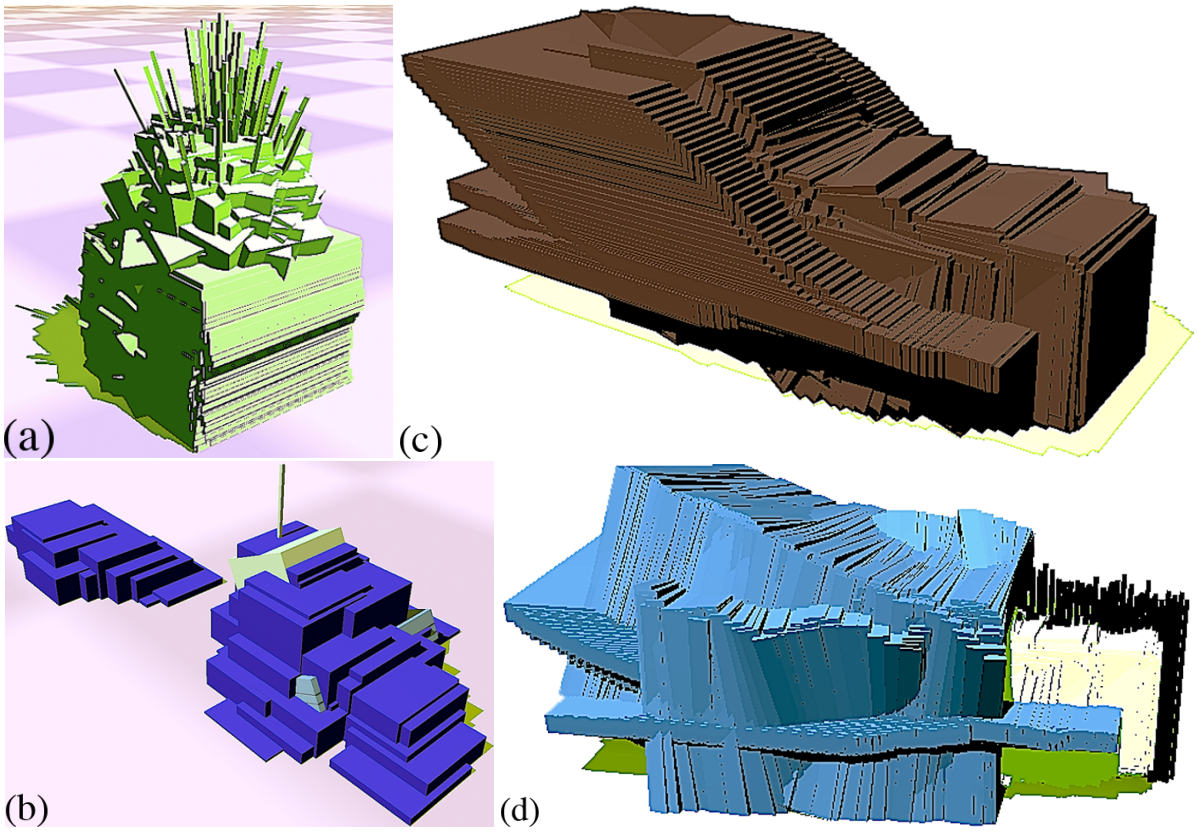


Figure 5.7: (Cubic) bodies coin the character of these architectural idea models.

ing number of identical agents steadily widens the diameter of the construction (67% layers). The interplay of the swarms results in a rhythmic construction pattern that gains momentum towards the model's peak.

Swirly architectures

Figure 5.9 depicts an architectural idea model that is composed of several interwoven ripples of layer constructions. A homogeneous swarm of five individuals swirls around a declining path while dropping layers and rods. At the bottom (Figure 5.9(a)), two flocks are gaining distance, thereby splitting the construction. The resulting construction receives good credit for the approximation of the pre-defined shape and proves that a relatively low neighborhood ratio $r_n = 0.35$ may very well lead to an intriguing, vivid swarm architecture (Table 5.1). The

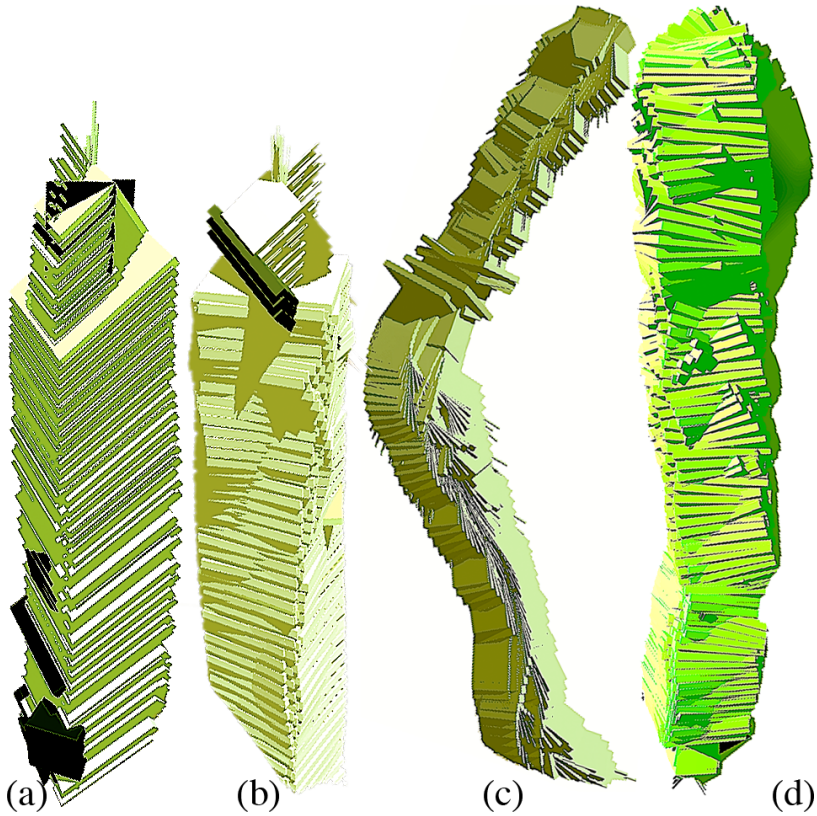


Figure 5.8: These tower architectures are mainly assembled of layer construction elements.

skeletal structure of Figure 5.10 is assembled of rods and cubic construction elements. Several swarm individuals wrap around and cement an inner construction with waves of rods. Crucial for this interplay is a probability-driven reproduction of the ‘foremen‘ and their differentiation into mere operative swarm individuals that do nothing but place construction elements. The emergence of a tight flocking pattern also strongly influenced this outcome. Figure 5.4(b) depicts the whole set of flocking parameters that determine the operative agent’s flight. Cohesion and alignment are forces to keep the agents orderly together. When combined with a tendency for separation and randomness, the bulge formations can emerge. Figure 5.11 displays a very complex swarm grammar phenotype: During the construction process agents spawn 725 times which might have led to the long computation time of $t_{real} = 63.3sec.$ During the interplay of the expressed swarm individuals, one of them is responsible for the reproduction and

differentiation—the corresponding behavioral rule is displayed in Figure 5.4(c). One individual only places rods, another one only layers. A fourth involved individual places a rod, a cube and a layer construction element all at once but with a very low probability $p = 0.2$.

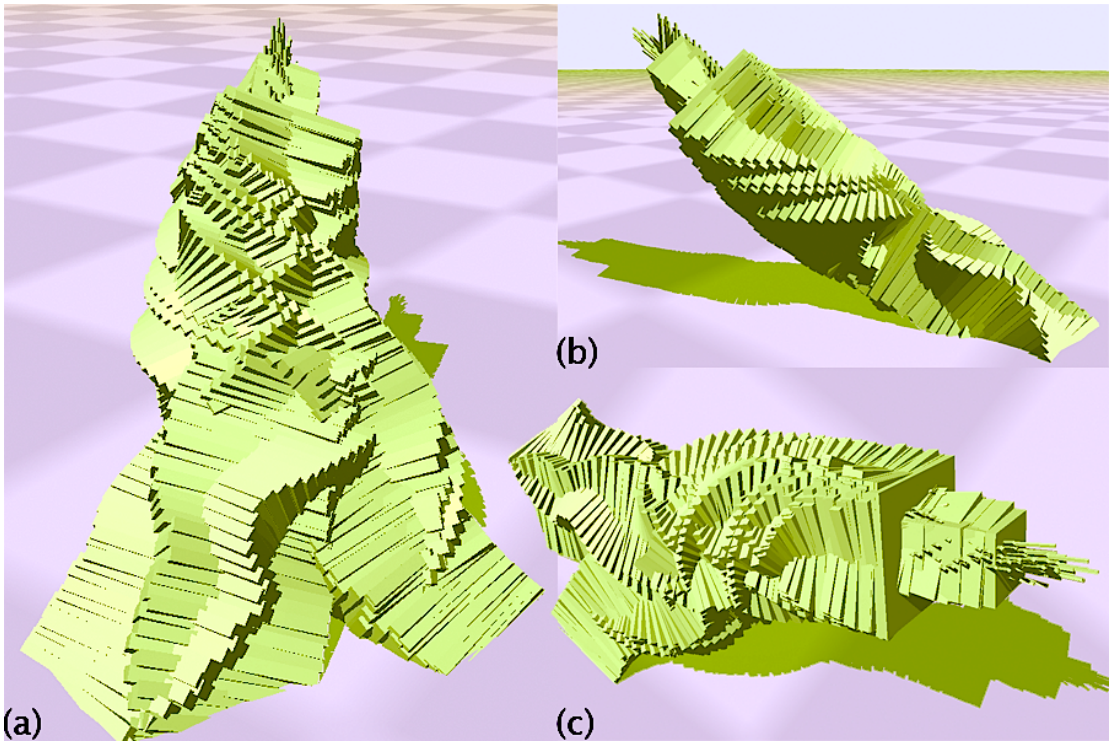


Figure 5.9: A swirly swarm architecture from different perspectives: (a) front view, (b) side view, (c) top view.

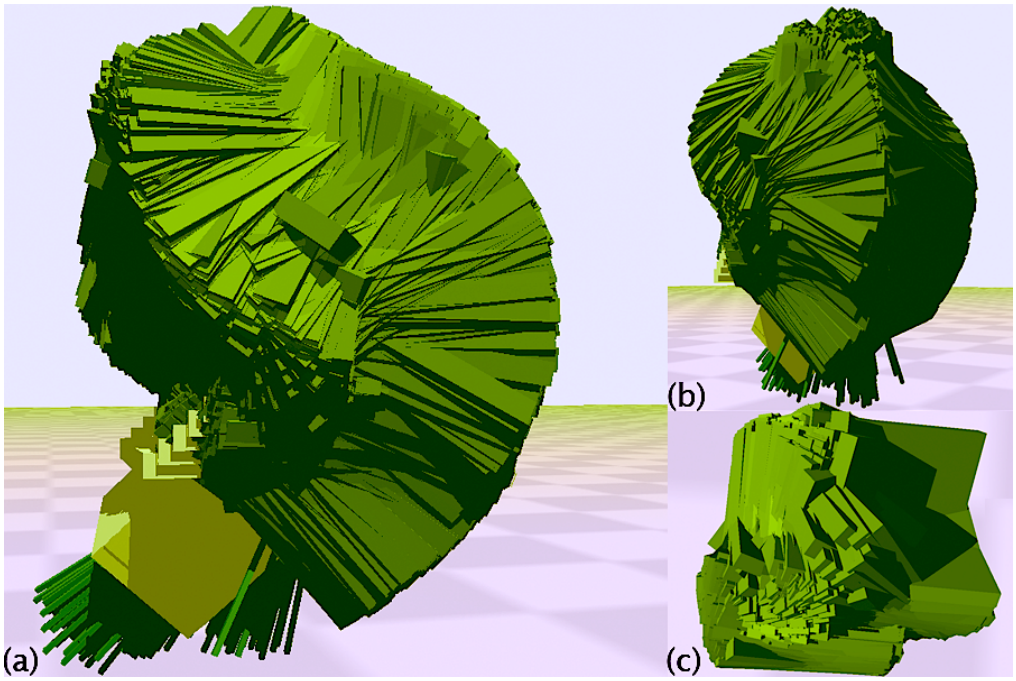


Figure 5.10: A swirly swarm architecture from different perspectives: (a) side view (b) front view, (c) top view.

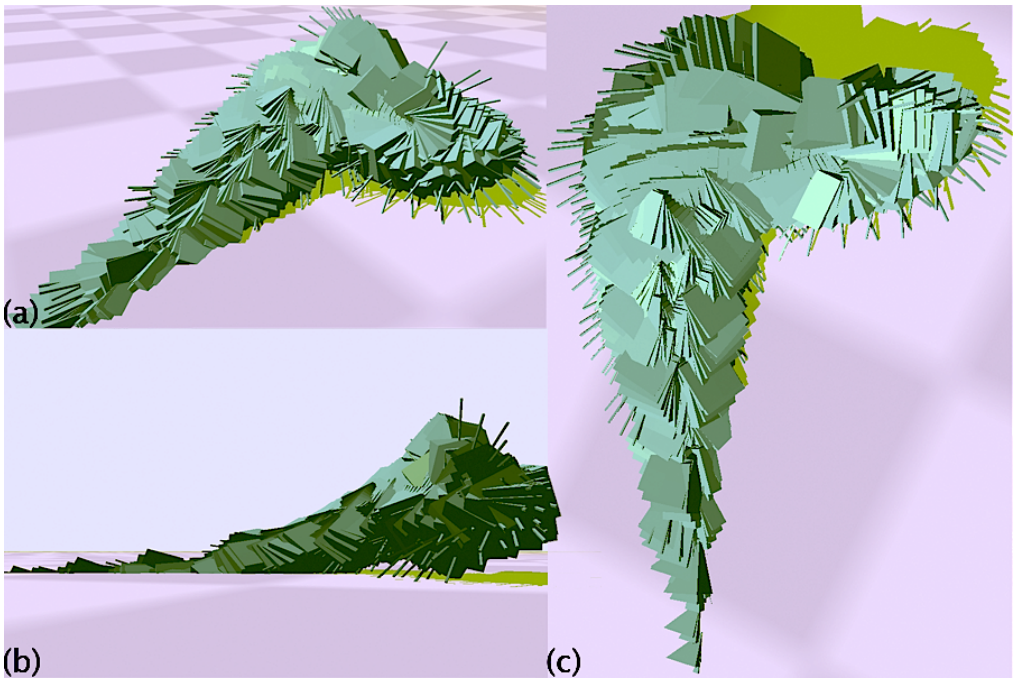


Figure 5.11: A swirly swarm architecture from different perspectives: (a) from a 45° angle, (b) side view, (c) top view.

5.3 Ecological features of swarm constructions

The presented architectural idea models demonstrate how biological construction processes can be adopted to create nature inspired architectural models [103, 104]. As we will explain in the following paragraphs, the outlined bio-inspired approach inherently promotes organic aesthetics, produces individual solutions for specific environments and offers dynamic and diverse designs.

The architect David Pearson claims that the predominance of rectangular, cubic elements, the emphasis of ‘the straight line‘ is a relic of the industrial revolution and not desirable. Instead, an organic style with ‘free-flowing curves‘ should be favored which is supported by modern construction processes and materials [105]. Numerous local interactions of swarm grammar individuals realize this aesthetic demand. Especially Figures 5.9 to 5.11 bristle with round shapes, ripples of construction elements and harmonically interwoven structures.

Modern architecture needs to be integrated into the environment, the ‘site-specific context‘ has to be taken into account [103]. The swarm-driven construction approach responds to this requirement on two levels. On the one hand, swarm individuals are aware of their environment and act accordingly. Through this stigmergic mechanism the presented constructions emerged. On the other hand, artificial evolution of constructive swarms can be utilized to optimize the constructions in regards to waste water disposal, energy efficiency and other aspects of ecological and economic performance [106]. Therefore, it is necessary to integrate supplementary software modules that evaluate these features of performance [107]. In addition to swarm heterogeneity and collaboration, ecological performance would direct the evolution of swarm-built architectures.

Chapter 6

Swarm complexity

The measures to promote diversity, productivity and collaboration outlined in the last chapter are favorable starting points to evolve complex swarm activities and emerging SG structures. Complex systems in nature usually comprise large numbers of interacting units, as for instance immune system cells that swarm in our bodies to fight off pathogens and remove damaged cells [181]. However, it already takes great effort to create and analyze stochastic models of only a few interacting units [182].

Numerical experiments have been playing an increasingly important role in the investigation of complex systems [183]. In order to build numerical models of complex systems, it is necessary to identify those features of natural systems that are crucial for the emergence of the phenomena of interest [11]. In particular, complex patterns that appear in natural systems, form in space and unfold over time, have been reproduced in models built from large sets of computational units that change their states in accordance with their local neighborhoods. Cellular automata [135] and random boolean networks [111] are examples of such models, both of which are outlined in Chapter 2 on related work.

Like in natural swarms—such as bird flocks or fish schools—, the neighborhoods of artificial swarm individuals change constantly and depend on preceding interactions[1]. Therefore, the dynamics of a swarm can be measured as the fluctuations in perceived neighbors. Based on this approach we are able to characterize the dynamics of boid swarms that exhibit various flocking formations. Hereby, we also identify phases and phase transitions of a boid system, including limit cycles and steady states.

[1] In [184], we demonstrate how the notion of artificial swarms as highly dynamic complex network systems can be exploited to develop novel, engaging gaming concepts.

In order to capture the formation of neighborhood relations in swarms, we measure the numbers of neighbors for every individual at each simulated time step, within a particular neighborhood radius. We show examples of neighborhood evolutions and discuss these through swarms that exhibit specific flocking formations. We demonstrate that switching and oscillating neighborhood formations can be achieved in homogeneous swarm systems whose flight is solely regulated by a linearly scaled acceleration of the individuals.

Agent states and neighborhood relations are inseparable in swarm systems. Vice-versa, we evolve boid configurations to approximate characteristic neighborhood functions. Here we show that non-linear and oscillating neighborhood developments can emerge in spatially organized homogeneous swarms that solely rely on linearly scaled flight acceleration based on repulsion and attraction.

As in studies of complex pattern formations in two dimensional CA [135], our experimental data suggests that:

1. Varying initial conditions and noise influence the evolution of boid flocking patterns locally. The general characteristics of the emerging patterns, however, are mainly based on the flocking parameters of the swarm.

2. Different swarm configurations can lead to very different pattern formations.

3. Chaotic behavior—unexpected, chance-based phase transitions—can occur in systems that initially show orderly, periodic patterns.

6.1 Swarms as a model of complexity

Once determined, the neighborhood relations between the units of a CA or RBN remain fixed. One may assume, however, that in many natural systems different forces draw and push the involved units so that they change their positions, as is observed in bird flocks, fish schools, ant

colonies or cell development. Thereby, of course, the neighborhoods of the units do not remain static.

Exactly this idea is captured in the 'swarm metaphor'. Large sets of swarm agents, attract and repel each other. The neighborly influence felt by one individual determines its action for the next time step. A swarm agent changes its velocity and position, thereby gaining a new neighborhood perspective and, at the same time, altering its neighbors' perspectives. Consequently, a feedback loop of actions and reactions emerges. Unlike in CA and RBN, state changes directly impact neighborhood bonds. We argue that this feedback loop between agents and their changing neighbor arrangements is the key feature to model spatially organized systems, since locality plays a crucial role for any effective interaction.

The neighborhood relations that drive the simple boids model outlined in Section 2.6.2 depend on the sight, the orientation and the position of the seeing individual, on the position of the potentially perceived individual, as well as on time.[2] The emerging causal chain can be expressed as follows (Fig. 6.1): The actions of a swarm agent i change its state which influences all those agents that are seeing i. At the same time i's new state results in the perception of a certain set of neighbors. These neighbors influence i's actions and the feedback loop starts all over again.

It is worthwhile noting that the system state and the neighborhood configuration are inseparable in the outlined swarm model. As a consequence, the observation of alterations of neighborhoods can be utilized to describe the system dynamics. Therefore, we measure the numbers of perceived neighbors $n(t) = |N_i(t)|$ of each swarm agent i at any given point in (simulated) time t. We characterize a single state of the whole swarm by the average neighbor value of all M swarm individuals. That is, we define the time-dependent neighborhood function for a swarm with M agents as

[2] All vectors $\vec{a}_i$, $\vec{v}_a$, $\vec{v}_c$, $\vec{v}_s$, $\vec{v}_w$ and sets S_i and N_i are time-dependent, but we will not denote the time variable explicitly.

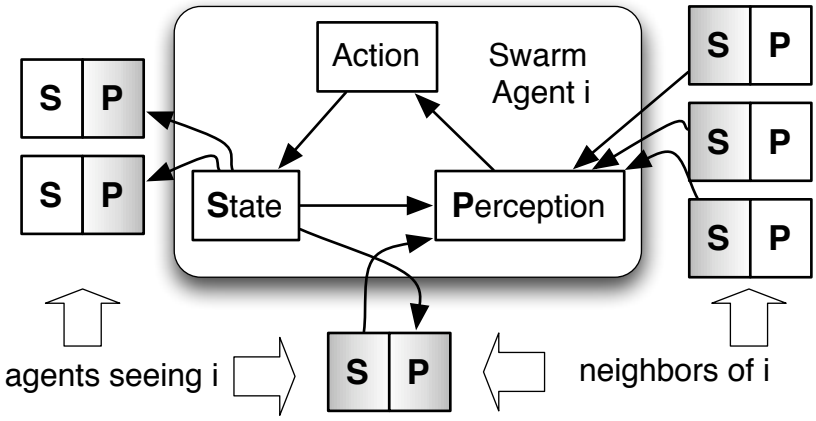


Figure 6.1: The slim arrows in the upper box show the direction of influence between perception, action and state of a swarm agent i. The S-P tuples stand for the state and perception modules of other agents that interact with i.

$$\bar{n}(t) = \frac{1}{M} \sum_{i=0}^{N} n_i(t). \tag{6.1}$$

Finally, the evolution of $\bar{n}(t)$ over the course of time helps to analyze and describe the dynamics of the (swarm) system. Based on this approach we investigate various flocking formations of boid swarms in the next section.

6.2 Analysis of flock formations

Jacob and Kwong have shown that diverse flocking behaviors of boids can be evolved with different parameter sets for Equations 3.4 to 3.5 (Sections 2.6.2, 2.6.5 and 3.1). We utilize four sets of flocking parameters from their work (Table 6.1) to analyze 'choreographic' line formations and figure-eight formations based on $\bar{n}(t)$. Two different swarm configurations are provided for each formation type. The following analysis links several phases of swarm interactions and the occurrences of desired formations to the development of the neighborhood function $\bar{n}(t)$. The presented results are all produced by 50 swarm agents with a perception

radius $l = 3.5$ and viewing angle $\alpha = 2.0$. As above, we normalize all $\bar{n}(t)$ values by the number of active swarm agents.

Line formations

	w_{align}	w_{coh}	w_{sep}	w_{world}	w_{rand}	max_{accel}	max_{vel}	d_{min}
(i)	7	8	5	5	5	38	13	0.14
(ii)	7	8	4	10	4	40	9	0.01

Figure-eight formations

	w_{align}	w_{coh}	w_{sep}	w_{world}	w_{rand}	max_{accel}	max_{vel}	d_{min}
(i)	3	10	1	5	2	38	6	0.01
(ii)	5	10	2	12	1	35	6	0.34

Table 6.1: Evolved parameter sets for 'choreographically' flocking swarms [83].

6.2.1 Line formations

Figure 6.2 shows the development of $\bar{n}(t)$ with the line formation parameters in row (i) of Table 6.1. The graph shows the average number of neighbors perceived by each agent over time. The plot can be partitioned into five distinct phases. In phase I, the average neighborhood perception $\bar{n}(t)$ is rising rapidly. Mainly the urge towards the world center $\vec{w} = (0, 0, 0)^T$ accelerates the initially stationary agents towards each other (Fig. 6.3(a) to (c)), ending up much closer than before (Fig. 6.3(d)).

During cluster formation the agents gain momentum bypassing many other agents. This leads to the decreasing average neighborhood perception in Phase II. As a result, several smaller flocks emerge after these two initial phases (Figure 6.4). The cohesion urge is now strong enough to keep subgroups of agents together that gather in the same vicinities (Fig. 6.4(a)). The alignment urge transforms these subgroups into flocks that exhibit increasingly homogeneous flight patterns (Fig. 6.4(b)).

In phase III of Figure 6.2, a line formation emerges (Fig. 6.5(a)), yielding relatively small values of $\bar{n}(t)$. Steadily, the agents are drawn closer to each other and $\bar{n}(t)$ increases accord-

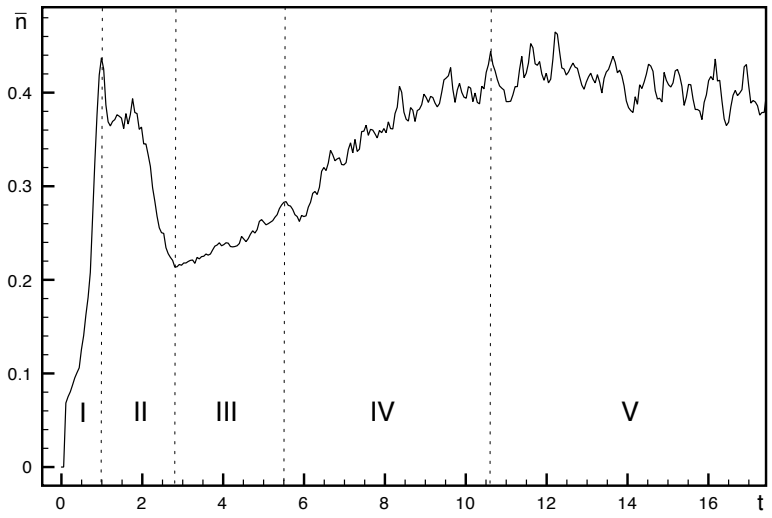


Figure 6.2: Developments of the average neighborhood perception $\bar{n}(t)$ matches several phases of agent interactions and flock formations.

ingly. In phase IV a dense agglomeration of agents emerges at the head of the line formation (Fig. 6.5(b)). Eventually, the line formation is destroyed and substituted by a tight cluster formation (Fig. 6.5(c)). After reaching phase V, the flock remains in a quasi steady state that is subject to only minor fluctuations (Fig. 6.5(d)).

Changing the simulation to the line formation (ii) parameters in Table 6.1 results in increased randomness of the agents' acceleration. The swarm looses its tight, cohesive constraints and thereby allows for the sporadic escape of agents. Figure 6.6 shows the corresponding neighborhood function. The neighbors of a fleeing agent may try to catch up and break out of the cluster as well. Consequently, the swarm's flight is dominated by tight cluster formations but is frequently interrupted by line formations (Figure 6.7). Another consequence is that single agents or even whole flocks can leave the parent flock, so that eventually all agents are dispersed and unable to interact.

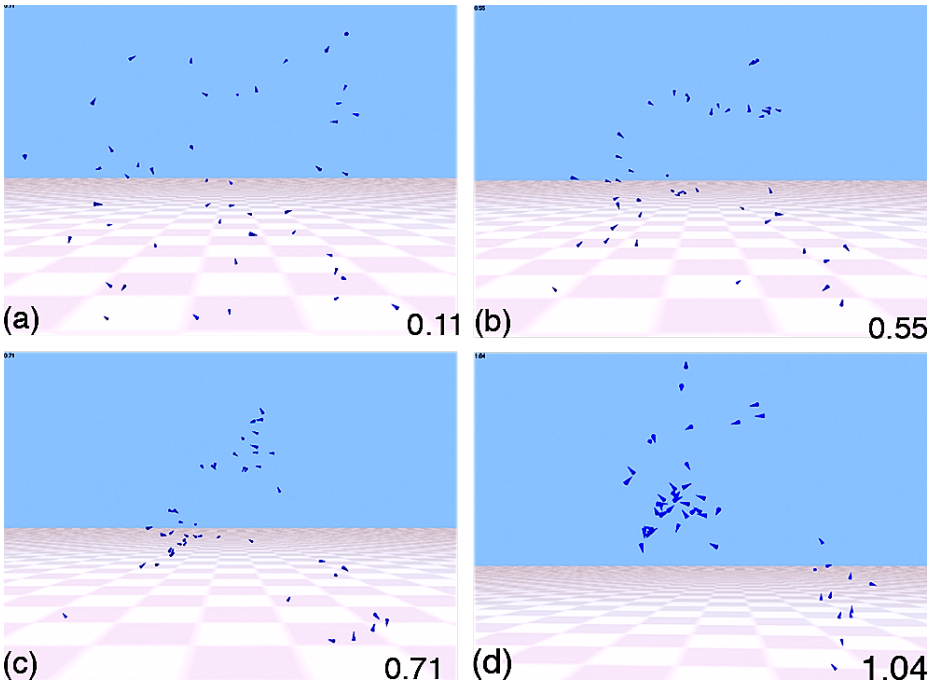


Figure 6.3: In phase I initially stationary agents are drawn together by the urge towards the world center.

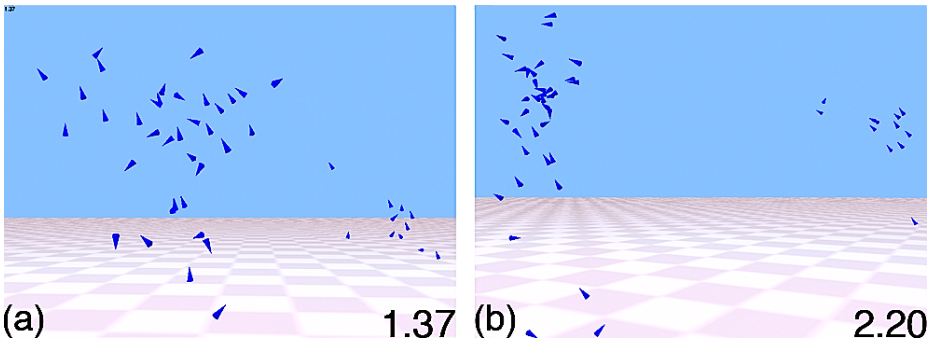


Figure 6.4: In phase II subgroups align as separate flock formations.

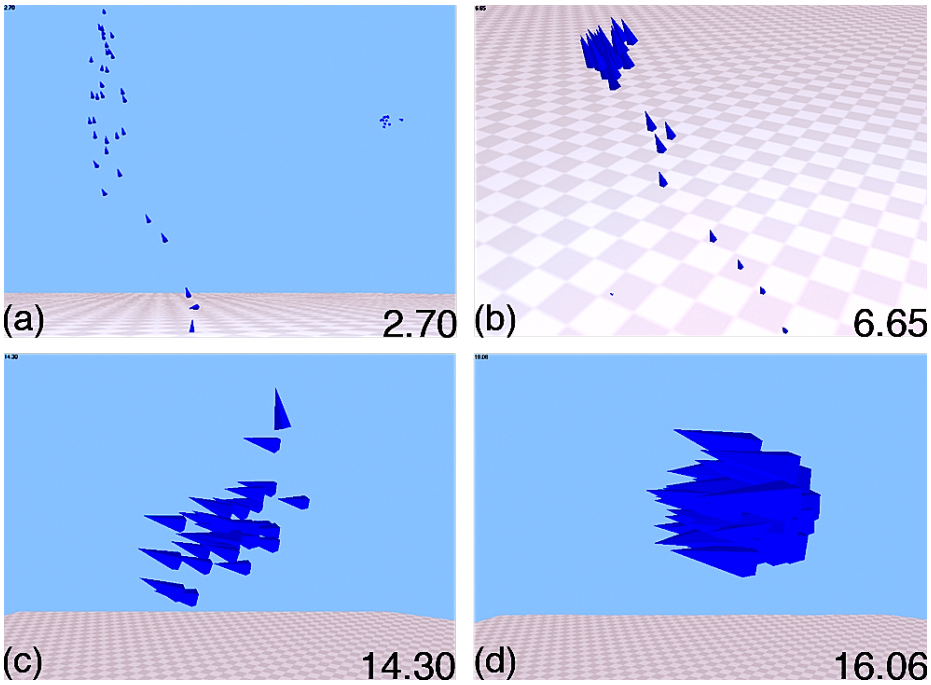


Figure 6.5: (a) Phase III: agents of single flocks follow each other in a line formation. (b) Phase IV: agents gather into dense clusters at the heads of the line formations. (c) and (d) Phase V: a tight cluster has formed that is robust enough against sporadic attempts of separation.

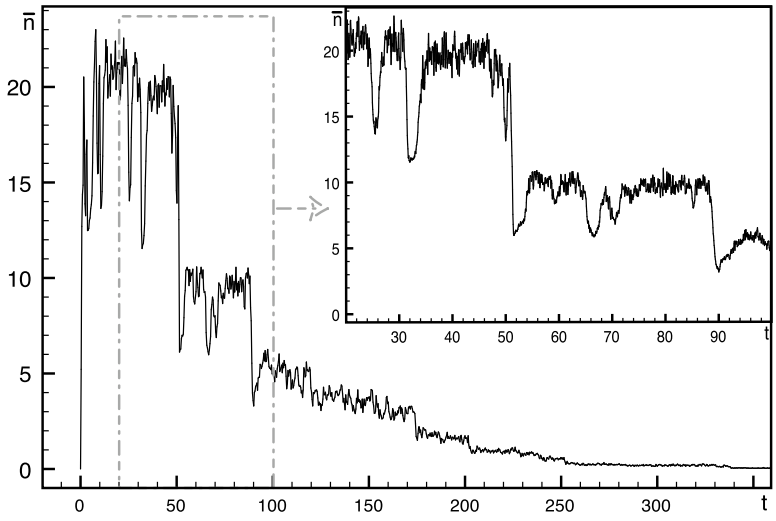


Figure 6.6: In the simulation of line formation (ii) of Table 1 agents break out of tightly formed clusters and take the lead of long line formations. During such events $\bar{n}(t)$ drops temporarily (e.g. at $t = 25$ and $t = 32$). Frequently the line formations break up (as in Fig. 6.5) and the parting flocks do not interact anymore ($t = 50$). As a consequence, $\bar{n}(t)$ reaches a value of zero at about $t = 400$.

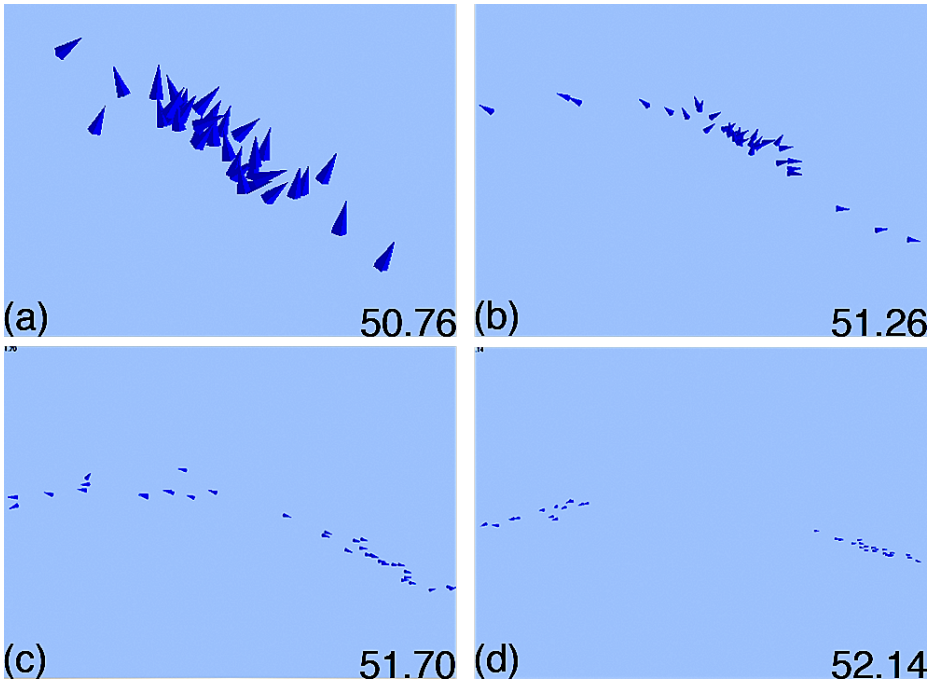


Figure 6.7: An agent cluster is breaking up into two line formations, one urging upwards, one towards the floor.

6.2.2 Figure-eight formations

In analogy to the discussed line formation examples, we investigate two boid configurations that exhibit figure-eight flight patterns with respect to $\bar{n}(t)$. As we can see in Figure 6.8, parameter configuration (i) from Table 6.1 reaches a steady state, whereas with setting (ii) agents repeatedly wander through different phases to eventually spread all agents far enough from each other to prevent further interaction—exactly as in line formation (ii) and in Figure 6.6. Configuration (i) rapidly swings into a figure-eight formation traced by small clusters of six to ten agents (Fig. 6.9(b)). Here, the swarm constantly traverses through a limit cycle of global states as indicated by the fast oscillating values of $\bar{n}(t)$ (Fig. 6.8(i)). In comparison to the line formation experiments, the oscillation of $\bar{n}(t)$ is characteristic for figure-eight formations. Furthermore, for configuration (i) the oscillation reached at about $t = 20$ marks a quasi steady state. High values of $\bar{n}(t)$ correspond with a high value of perceived neighbors. Therefore, we suggest that the oscillation occurs because alternating numbers of swarm individuals meet at the centering knot of the figure-eight formation.

In figure-eight configuration (ii) the neighborhood perception converges towards zero, and the subgroups of swarm agents may break away during intermediary line formations. This is similar to the second line formation experiment. Line formations, such as illustrated in Figure 6.10(a) are reflected by the steep drops of $\bar{n}(t)$ in Figure 6.8(ii). In general, line formations fold back quickly into figure-eight formations, as is shown in Figure 6.10(b). In contrast to the line experiments, the difference between a configuration that quickly results in a stable equilibrium and a swarm that exhibits long periods of drastic changes cannot be directly inferred from the according flocking parameters in Table 6.1. We assume that in complex figure-eight flight patterns it becomes more difficult to identify a parameter, such as the random weight r_{rand} in Equation 3.5, as crucial for spontaneous behaviors.

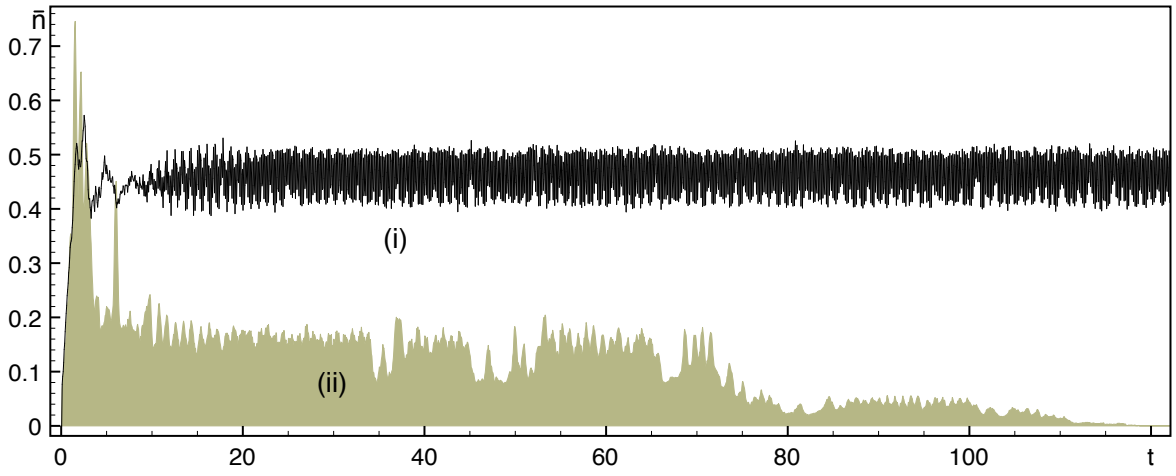


Figure 6.8: The development of $\bar{n}(t)$ of figure-eight formations (i) and (ii) based on the parameter sets in Table 6.1.

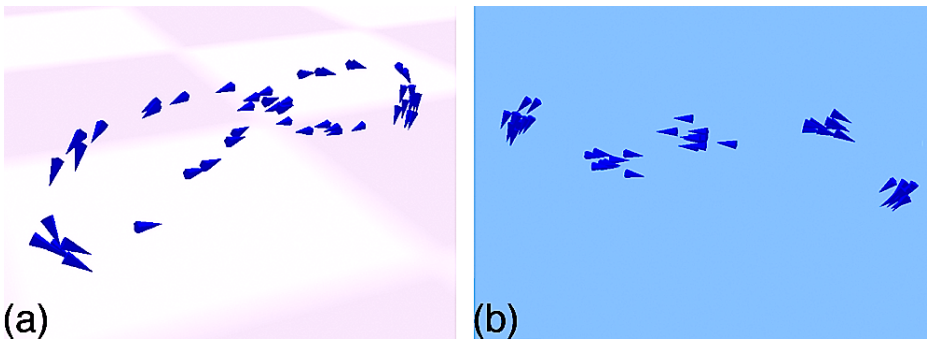


Figure 6.9: (a) Agents in figure-eight formation. (b) Tight agent flocks (of six to ten agents) in figure-eight formation.

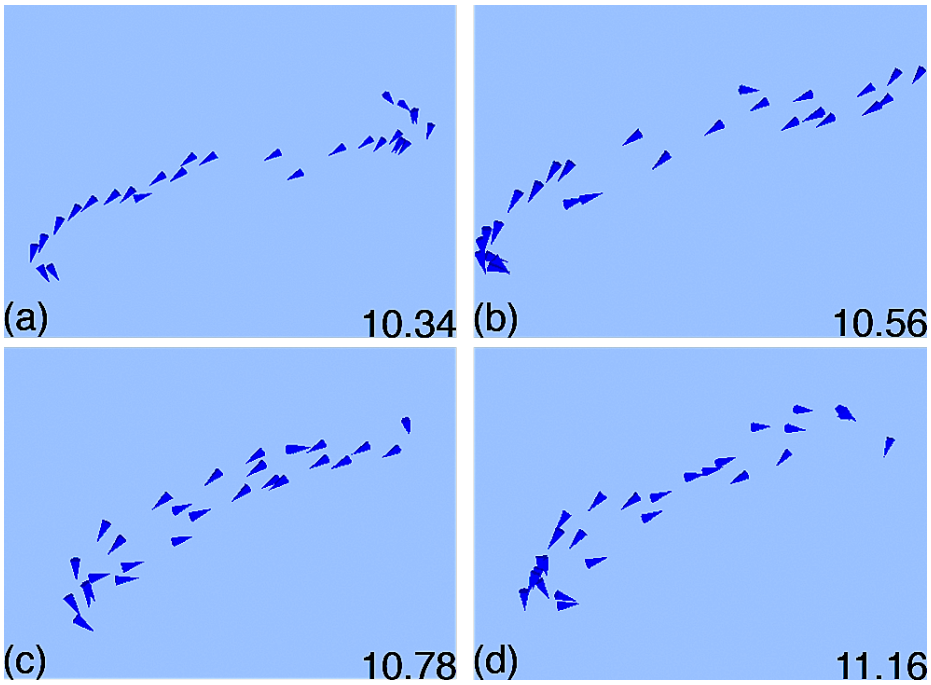


Figure 6.10: A line formation is about to collapse into a figure-eight pattern.

6.3 Reverse engineering of $\bar{n}(t)$

The examples in the previous section demonstrate how measuring the neighborhood dynamics over time can help to describe and analyze emergent flocking formations. Now, we utilize this association to approximate neighborhood dynamics as they might contribute to the coordination of naturally occurring phenomena, such as biological switches and clocks or timers. The research objective that we address is phrased by the following questions:

> "Could an average neighbourhood function that follows a step function result in a swarm that exhibits a bi-state, switch-like behavior?"[3]

And, secondly, "Could an average neighborhood function that follows a sinusoidal function result in a swarm that exhibits an oscillating flocking behavior?". This section explores both these questions by evolving swarms to follow according average neighborhood functions.

The neighborhood value $n_i(t)$ of a single agent may change rapidly, remain fixed or oscillate. It is also easy to discover a whole flock of agents entering an equilibrium of a specific average neighbor value $\bar{n}(t)$. The results in the previous section also show that even a whole swarm can change dynamically, or, put differently, can follow specific evolutions of $\bar{n}(t)$.

If it was not for its contextual evolution, the average neighborhood $\bar{n}(t)$ would mainly characterize the concurrent spread, or density, of boid flocks, thereby corresponding much to molecular concentrations. In fact, neighborhood fluctuations indicate changes in the structure of swarms. Immediately, the question arises which patterns of movement could one expect when looking at evolutions of $\bar{n}(t)$ that correspond to the development of molecular concentrations in biological measurements.[4]

Even though the expression of genes happens stochastically, the levels of expression can differ greatly which promotes the idea of a genetic switch [10]. By the approximation of a step

[3] This motivational question was suggested by Una-May O'Reilly in the course of the evaluation of my submitted dissertation.

[4] Such measured molecular concentrations may come from microarray experiments that approximately capture the number of (reporter) proteins over time.

function for $\bar{n}(t)$, we intend to show that even a homogeneous swarm could exhibit bi-stable switching behavior. Oscillations occur in natural systems as timers, such as circadian clocks. As a second option, we therefore explore which swarm behaviors can be evolved that follow a sinusoidal neighborhood function. For both endeavors we utilize a genetic algorithm that operates on populations of swarms as described in the following paragraphs.

6.3.1 Evolutionary experiments

A homogeneous boid swarm, consisting of agents that share the same control parameters, is represented by a genotype vector $\vec{b} = (w_{align}, w_{coh}, w_{sep}, w_{world}, w_{rand}, max_{vel}, max_{accel}, d, \alpha)^T$. We also want to modify the starting positions, initial accelerations and initial velocities of the swarm agents: $init_0, init_1, ..., init_M$. The extended swarm genotype is therefore $\bar{g} = (\vec{b}, init_0, init_1, ..., init_M)$.

In the following experiments we provide a desired target function $x(t)$ for $\bar{n}(t)$ and reward its approximation with a fitness value $f = 1/\sum_{t=1}^{40} |\bar{n}(t) - x(t)|$. We rely on the genetic operators of fitness proportionate selection, incremental mutation and multi-point crossover on all numeric values. In fact, the applied crossover operator $c3$ is in line with $c1$ and $c2$ that are introduced in Section 5.2.1. During the investigations of complex flocking formations, however, one homogeneous flock of boids is solely determined by one configuration module, C_{boid}. The values stored in this configuration module are exemplarily depicted in Figure 5.4(b) and they are the genes exchanged between pairs of flock specimen by operator $c3$. The values were represented as $64Bit$ floating point numbers, whereas each number was treated as a gene during the recombination processes. A population counts 30 swarms, with each swarm consisting of 30 agents. The genetic algorithm was run up to 300 generations.

6.3.2 Step function

As the computation of the genotype was limited to 40 simulated seconds, we decided to trigger the switch at about half of the overall time-frame. A difference of 0.5 units in a system with values $\bar{n}(t) \in [0; 1)$ denotes an obvious leap, whereas a lower boundary $\bar{n}_{min} = 0.25$ is large enough to allow for further swarm interactions (as opposed to $\bar{n}_{min} = 0.0$ that rules out the possibility for local interactions).

$$x(t) = \begin{cases} 0.25 & t < 22 \\ 0.75 & t \geq 22 \end{cases}$$

Figure 6.11 displays the step function approximation of a boid configuration that appears after 200 generations of the outlined genetic algorithm. In Table 6.2 we list the parameters of the best evolved swarm genotype, which reveals two surprising values: $w_{sep} = 1.0$ and $max_{vel} = 0.0$. In fact, the velocity of the swarm individuals is greater than zero—the integration step of the simulation increases the velocity in accordance with the provided acceleration $\vec{a}$ that is limited to $max_{accel} = 12.15$. The limitation of $max_{vel} = 0.0$ means that the agent is stopped after each iteration, resulting in a very small velocity value, yet ensuring the orientation and alignment according to its flocking urges. As a consequence, starting from their initial positions, the agents are slowly converging towards each other—nicely timed with the target function. The large weight for the separation urge $w_{sep} = 1.0$ prevents the agents from getting too close and exceeding the target value of $x = 0.75$. The swarm is trained to approximate the step function for 40 simulated seconds. The flight behavior of the swarm after 40 seconds is not taken into account by the fitness function. For that reason, the flight parameters are delicately balanced within this time frame. In the given example the swarm reaches an equilibrium shortly afterwards, at about 60 seconds. At this point in time the flock is clustered at the center of the simulation. As the individuals still try to avoid each other and because of a relatively great random impact on their orientation, they frequently look past each other causing $\bar{n}(t)$ to

drop into a quasi steady state at about $t_{qs} \approx 62$ and $\bar{n}(t) \approx 0.37$, with $t > t_{qs}$.

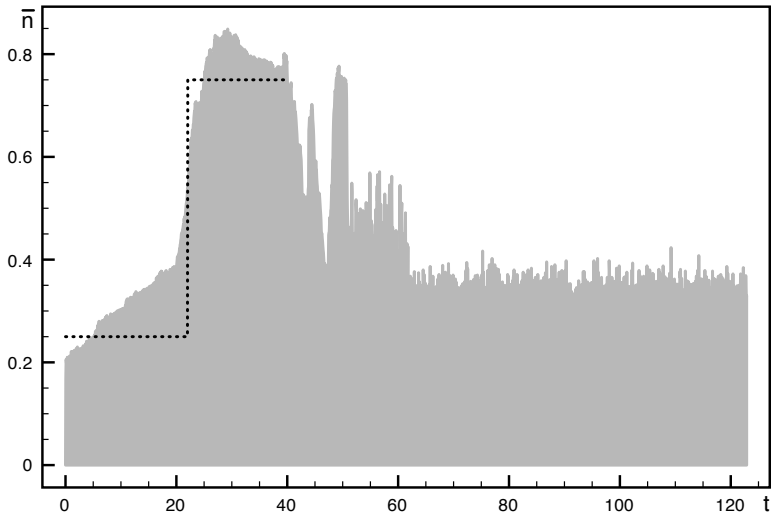


Figure 6.11: A neighborhood function $\bar{n}(t)$ of a boid configuration, bred by an evolutionary algorithm, approximates a step function as $\bar{x}(t)$. Afterwards, outside the evaluation window the swarm drops into an equilibrium with $\bar{n}(t) \in [0.35; 0.45]$.

Boid configuration for $\bar{n}(t)$ step function approximation

w_{align}	w_{coh}	w_{sep}	w_{world}	w_{rand}	max_{accel}	max_{vel}	d_{min}
0.37	0.86	1.0	0.44	0.48	12.15	0.0	4.89

Table 6.2: Evolved swarm parameters that result in the neighborhood function of Figure 6.11, implementing a switch in $\bar{n}(t)$ ($\alpha = 2.09$, $d = 9.32$).

6.3.3 Sine function

Two periods of a sine function are provided as target function $x(t)$ for time frame of 40 simulated seconds. As in the step function approximation, $\bar{n}(t)$ is not forced to drop below 0.25 to guarantee a minimal space for interactions.

$$x(t) = sin(4\pi * t/40.0) * 0.25 + 0.5$$

Figure 6.12 shows the neighborhood function $\bar{n}(t)$ for the evolved swarm configuration as listed in Table 6.3. Eventually, at $t = 1244$ in Figure 6.13, the oscillation ends; this is when the agents form a tight cluster orbiting around the world center $\vec{w}$.

Since complex interactions render it difficult to identify certain flocking patterns, we activated motion blurring to better capture the pattern formations of the swarm. We determined that the oscillation happens as the biggest flock repeatedly expands (Figure 6.14) and contracts (Figure 6.15). Leaps from a plateau to a local maximum, as seen at $t = 100$ in Figure 6.12, occur when formerly separated flocks rejoin (Figure 6.16). In Figure 6.17 several screenshots with activated motion blur illustrate intermediary flight formations.

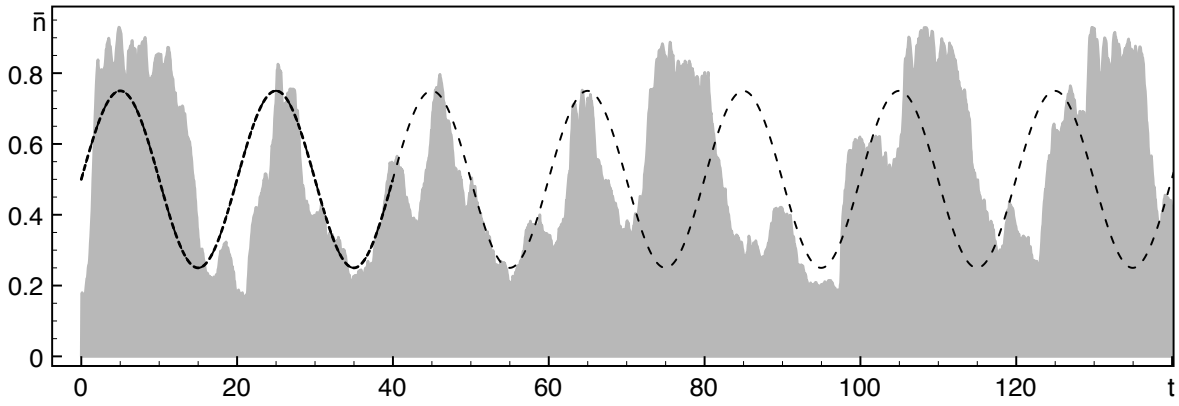


Figure 6.12: A boid configuration bred by an evolutionary algorithm approximates two periods of a sinusoidal neighborhood function. As shown, the oscillations sustain even afterwards.

Boid configuration for $\bar{n}(t)$ sine approximation

w_{align}	w_{coh}	w_{sep}	w_{world}	w_{rand}	max_{accel}	max_{vel}	d_{min}
0.76	0.95	0.53	0.36	0.76	12.15	7.16	4.12

Table 6.3: Evolved swarm parameters that result in the neighborhood function $\bar{n}(t)$ of Figure 6.12. The corresponding swarms display oscillating behaviors ($\alpha = 2.64$, $d = 7.86$).

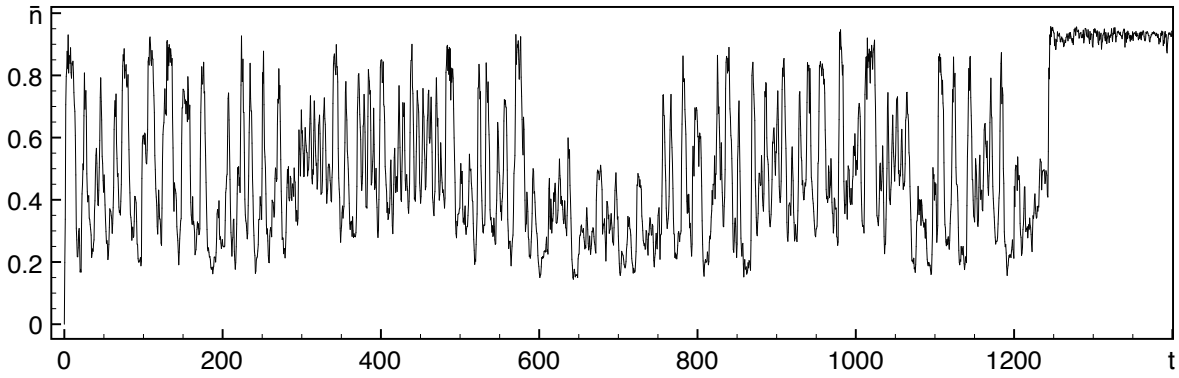


Figure 6.13: After about 1200 simulated seconds the oscillating swarm transitions into a steady state.

6.4 From investigations into the complex toward a new swarm model

Agent states and neighborhood relations are inseparable in swarm systems. Therefore, the dynamics of a swarm can be measured as the fluctuations in perceived neighbors. Based on this approach we are able to characterize the dynamics of boid swarms that exhibit various flocking formations. Hereby, we also identify phases and phase transitions of the boid system, including limit cycles and stead states.

Vice-versa, we evolve boid configurations to approximate characteristic neighborhood functions. Here we show that non-linear and oscillating neighborhood developments can emerge in spatially organized homogeneous swarms that solely rely on linearly scaled flight acceleration based on repulsion and attraction.

As in studies of complex pattern formations in two dimensional CA [135], experimental data suggests that (1) Varying initial conditions and noise influence the evolution of boid flocking patterns locally. The general characteristics of the emerging patterns, however, are mainly based on the flocking parameters of the swarm. (2) Different swarm configurations can lead to very different pattern formations. (3) Chaotic behavior—unexpected, chance-based phase

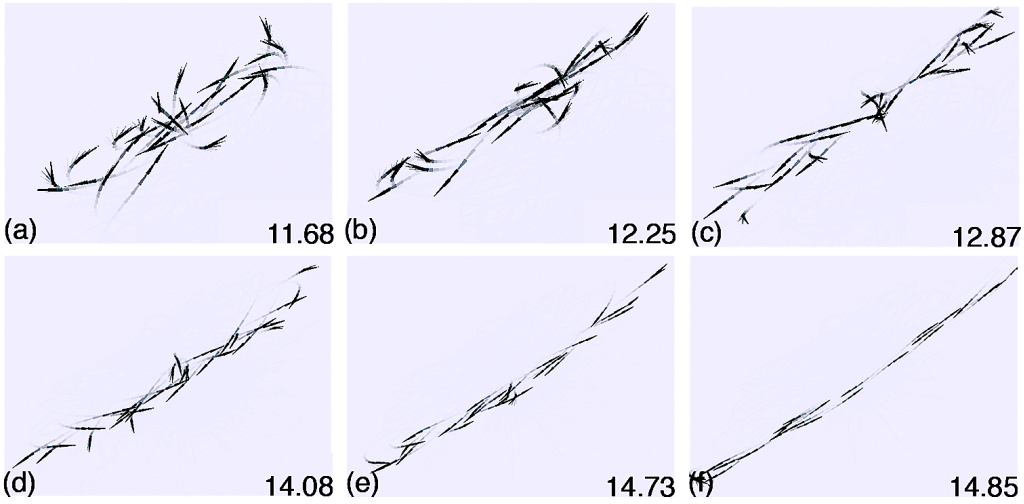


Figure 6.14: The flock extends in two directions.

transitions—can occur in systems that initially show orderly, periodic patterns.

We published the results presented in this chapter in [7]. Therein, we also suggested the creation of an abstract swarm model for further investigation into swarm complexity. The envisioned model should maintain the link between state and neighborhood, but should be reducible to any high-dimensional space. It should be desirable to find operators that comply with the amalgamation of time and space, or respectively structure and state, without realization of physics. With a generalized set of operators that change states and neighborhood relations concurrently, a systematic classification of swarm dynamics might be possible. Thereby, we hypothesized, swarms could become an important model for the dynamics of complex systems in general. In the next chapter, we present our first attempt to realize this objective.

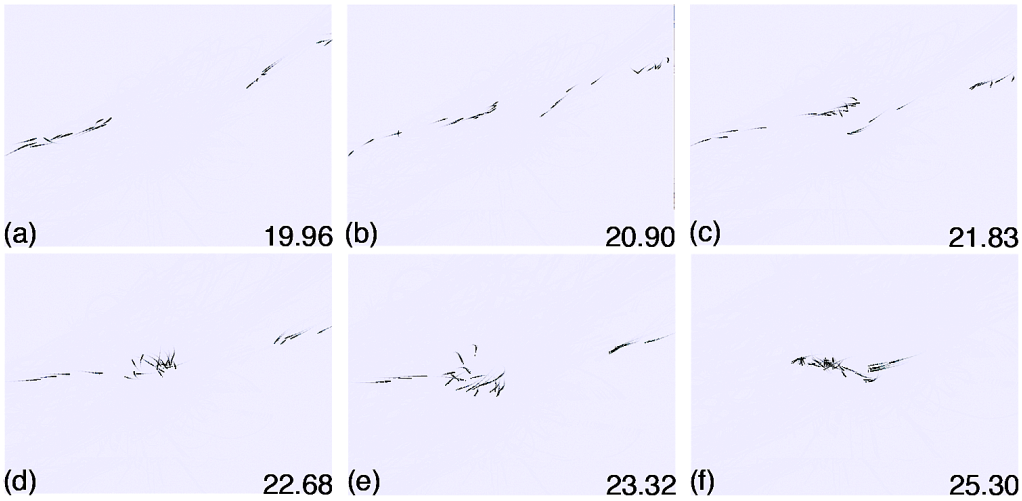


Figure 6.15: The previously extended flock from Figure 6.14 contracts again.

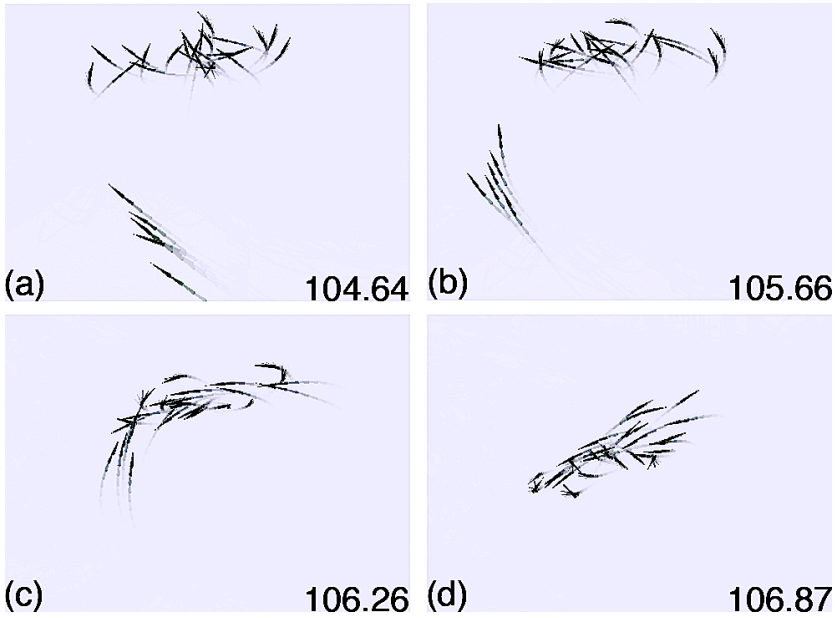


Figure 6.16: A second flock approaches and joins the other one.

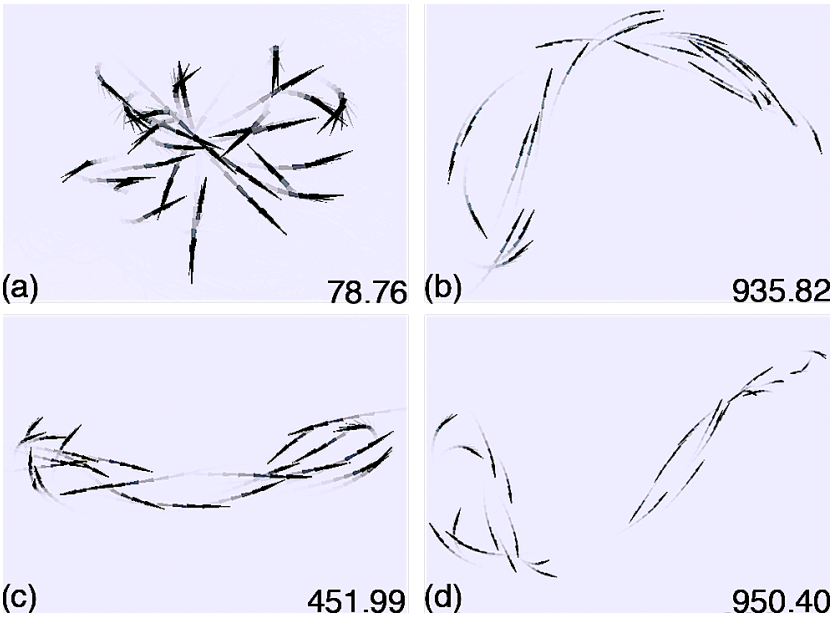


Figure 6.17: Motion blurring renders some of the more complex flight patterns identifiable: (a) Spherical formation, (b) a U-bent figure-eight, (c) and (d) extended figure-eights.

Chapter 7

Swarm graph grammars

Computational developmental models (CDM) have shed light on important phenomena in the development of living organisms (Section 2.2). In *artificial chemistries*, sustainable, autocatalytic networks of interdependent molecules have emerged [111]. Stable structures resembling the membranes of cells have been shown to evolve [48], and concepts for the development of increasingly complex hierarchies of living structures have been presented [138] (Section 2.5.3). However, each step towards understanding the miracles of nature fosters our acknowledgement of their incredible inherent complexities (Section 2.5).

The basis for complexity in CDMs lies in the interaction dynamics of their constituents. In addition, the topologies of their constituents might change, thereby introducing another level of dynamics (Section 2.5.2). The developmental swarm grammar system implements implicit topological changes through a formal grammar and the individuals' flocking behaviors [2]. In this chapter, we introduce *swarm graph grammars* (SGGs) that extend SGs by an explicit topological, graphical representation of interaction processes. SGGs also unify the previously independent encodings of parametric flocking behavior and rule-based agent activities (e.g. reproduction and construction).

Section 7.1 details swarm graph grammars (SGGs) and their constituents, i.e. swarm individuals, graph grammatical rules, and an according SGG algorithm. Section 7.2 shows how the SGG formalism is applied step by step to retrace and to extend the original swarm grammar and the underlying boids model.

7.1 A swarm graph grammar system

A swarm graph grammar $SGG = (Gen, \Xi, \mathcal{G}_{pred}, \mathcal{G}_{perf}, P)$ is a quintuple, where Gen contains a set of genotypes for generating new swarm individuals. At the beginning of the simulation, a set Ξ of axioms, in the form of initialization algorithms, is executed by first selecting and expressing a number of genotypes from Gen, and secondly, by assigning initial states to the newly created individuals. For a homogeneous boid flock, for instance, Gen only has to comprise a single genotype. Having created a sufficient number of boids based on this single genotype, the axioms would assign each boid contextual information such as their initial location in the simulation space and their initial velocities.

After the initialization routine (Algorithm 4), the main loop of the swarm graph grammar algorithm is entered (Algorithm 5). The SGG algorithm maintains two graphs, $G_{pred} \in \mathcal{G}_{pred}$ and $G_{perf} \in \mathcal{G}_{perf}$, where the swarm individuals are represented as nodes, and where edges denote their interrelationships. $\mathcal{G}_{pred}$ represents the set of possible graphs of agents interconnected through unary and binary predicative relations. $\mathcal{G}_{perf}$ hosts all possible graphs that impose performance relations onto swarm individuals. Predicative and performance relations take one or two swarm individuals as parameters. Relational chains among sets of swarm agents create semantic topologies for global graph structures that describe the situational context, or respectively, the activity in an SGG system. In fact, the alternating update of the graph instances G_{pred} and G_{pref} based on the individual swarm agents' behaviors drives the SGG simulation.

Algorithm 4 Initialization Routine

Require: Gen, Ξ
Ensure: added nodes to G_{pred}
express a number of swarm individuals relying on Gen
initialize expressed individuals in the simulation context
add the initialized individuals as nodes to G_{pred}
start the main loop

In order to compute G_{pred} and G_{perf} the following procedures are repeatedly carried out.

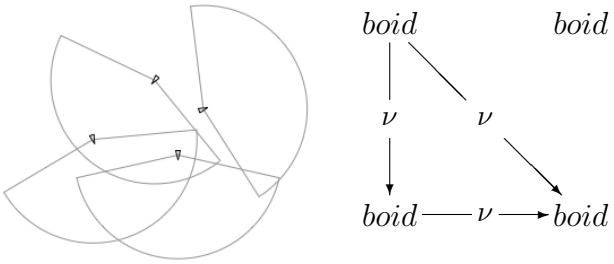
Algorithm 5 Main Loop

Require: G_{pred}, optional: P
Ensure: alternating computation of G_{pred} and G_{perf}
  repeat
    compute predicative graph G_{pred}
    compute performance graph G_{perf} based on G_{pred}
    apply order policy P to G_{perf}
    execute ordered performance relations of G_{perf}
  until simulation is terminated

First, predicates among the swarm individuals (si) are identified. For example, imagine the situation in Figure 7.1. The binary *neighbor* predicate $\nu(si_0, si_1)$ states that si_0 sees si_1, as si_1 sojourns within si_0's field of view. Secondly, the individuals react to their current (predicative) situations through the application of a number of behavioral rules. These graph grammar rules consist of a condition-querying head graph and a change-inducing body graph. If the topology and the labels of the head graph's constituents can be matched in the given situation, the rule is applied. If a rule is applied, its body graph, which maintains references to the head's node matches and a number of *performance relations*, is incorporated in G_{perf}. An execution order policy P can be applied, when the construction of G_{perf} is completed, i.e. when each swarm individual has incorporated all the performance relations of all its applicable rules. This third step of the main loop sorts the performance relations between pairs of individuals according to P. In the fourth and last step of the main loop, the performance relations of G_{perf} are executed in the given order. As a consequence, the swarm individuals obtain new states. Hence, when the predicative graph G_{pref} is computed in the next iteration, different predicates may occur. Detailed examples are provided in Section 7.2.

7.1.1 Swarm individuals

As outlined in Section 2.5.2, a broad agent definition is provided by the quadruple $Ag = (Sit, Act, Dat, f_{Ag})$. Sit refers to the agent's situation and describes what the agent could possibly perceive. Act is the set of possible actions of the agent, whereas Dat describes the space of



(a) (b)

Figure 7.1: (a) Four boids and their respective fields of view. (b) Edges in the graph G_{pred} indicate *neighbor* relations between two boids.

required internal data cells. f_{Ag} is the agent's decision function which determines the next action according to Sit and Act: $f_{Ag} : Sit \times Dat \rightarrow Act$. In the following paragraphs, this agent definition is customized for an SGG agent or swarm individual (si).

Situation, Sit

Being part of the simulation, a swarm individual si may find itself situated within any subgraph of G_{pred}. However, si cannot react to other individuals si_i that it is not immediately related to, i.e. without a predicative edge from itself to si_i. Thus, we define Sit as a set of all those subgraphs of a graph instance $G_{pred} \in \mathcal{G}_{pref}$ that include si, all predicative relations $rel(si, si_i)$ and the corresponding nodes si_i.

Although queries for subgraph isomorphisms are considered NP-complete [185], searching for Sit in G_{pred} is trivial, as only immediate relations to swarm individuals are considered. When querying the situation Sit in the attempt to match rule conditions, the problem size remains considerably small. For deviating SGG implementations, we suggest non-deterministic but efficient subgraph query methods as, for instance, provided through *closure-trees* [186].

Action, Act

Both predicative and the performing relations of a swarm individual are provided by its set of possible actions Act. The implementation of these relations is decoupled from the graphical

representation scheme. Primarily, the above Algorithms 4 and 5 provide the structural framework for process execution. On the second level, predicative and performance relations are processed that only access Dat and Sit information provided by the involved agents[1].

The terms *predicative* and *performance relation* underline the relational connections of the involved agents and the meaning of the simulation graphs G_{pred} and G_{perf}: Contextual interdependencies drive the agent interactions at each computational step. The connections of individuals are described by graphs which may continuously change—the interwoven, ever-changing web of temporary graph structures is a complex, dynamic network process. We deploy predicative relations only to query the states of the involved individuals and to compute and return a boolean value. Performance relations on the other hand may access and change the situation and the internal data of one or both of the involved agents. Thus, performance relations are responsible for any changes in the simulation. Of course, predicative relations could initiate some side-effects on the simulation as well but, ideally, apart from temporary values that are computed based on the predicative configuration of the system, any impact on the simulation should stem from the realization and execution of performance relations.

Internal data, Dat

Three distinct kinds of information are stored as a subset of Dat: (1) The values of a swarm individual's relative state in the simulation, (2) its genotype, and (3) the state of expression of the genotype and expressed data structures with accompanying values.

Part of a swarm individual's genotype is a set of behavioral rules $Bhvr$ which occupies a special area in Dat as it ties the different model components together. Application of the behavioral rules is the main computational step in a SGG system. It guides the computational process from querying certain predicative relations to the execution of arrays of directives. Figure 7.2 shows a rule for a boid agent $boid_{ref}$ to update its acceleration depending on $boid_0$.

[1]Our SGG implementations are written in Java. The core libraries of the prototyping framework *processing* are used for threading, GUI and visualization [157].

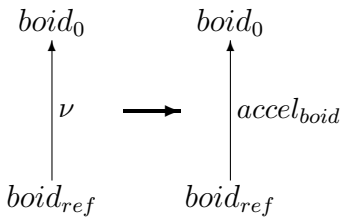


Figure 7.2: The neighbor predicate ν is substituted by an acceleration performance $accel_{boid}$.

Nodes with indices ref reference the instances that own the corresponding rules. We also refer to these nodes as *reference nodes*. The given rule can be read as: If the reference node has a neighbor $boid_0$, it updates its state through $accel_{boid}$ in regard to $boid_0$. The indices of the nodes are embedded in the formalism of the rules to correlate matches of the rule's head with its body[2]. The interaction partners' names, e.g. "boid", are actually taken into consideration when finding a node in G_{pred} to complement a rule's head node. Classifying names provide a shortcut to find relevant interaction partners in the simulation: Instead of testing relevant predicates among all nodes, an individual might, for instance, only be responsive to nodes labeled with "boid". Generally, a node in G_{pred} matches a rule node, if their names are identical and if they share an identical set of relations with the reference node.

Behavioral rules also express removal and creation of an additional swarm individual or particle. If a node occurs in the rule's head, but not in its body, it will be removed. Numerous swarm individuals could attempt to operate on the considered node simultaneously. However, the removal of a node has absolute priority and renders other accessing attempts futile. Figure 7.3 displays a removal rule: Upon perception of si_0 and si_1, si_{ref} removes si_0. At the same time, si_1 might also perceive si_0 and attempt to change its state, e.g. its location. This attempt would fail because removal has priority.

[2]A variable index in a rule node, e.g. $boid_i$, represents any number of matching nodes in G_{pred}. Consequently, the edges from the reference node to such a variable node are hyper edges which allow to match subgraphs in G_{pred} with arbitrary numbers of branches.

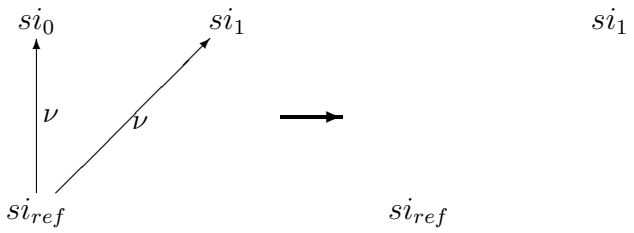


Figure 7.3: Example rule: If the swarm individual si_{ref} sees two neighbors (predicate ν) si_0 and si_1, it removes one of them (si_0).

Mutual exclusion does not arise when new nodes are introduced into the system. Syntactically, such a case is covered by nodes appearing in a rule's body that are not present in its head—exactly the opposite of a removal rule. It is important to note that a new node needs to be initialized by its ancestor. Thereby, it obtains a genotype, a state of expression resulting in its initial phenotype and a clear state in the simulation. As the according initialization performance determines the type, state and behavior of the new node, there is no a-priori distinction between new nodes in G_{perf}. For instance, they may become swarm individuals, static particles or pheromones (Chapter 3).

The rules of an individual si_{ref} might not respond to all the implemented predicates. Only those predicates have to be tested between si_{ref} and the other swarm individuals si_i that are relevant to si_{ref}.

7.1.2 Computational complexity

Computation overhead is minimized for predicate testing, if only relevant interaction partners (by name) and relevant predicates are considered for each of the active nodes. An average complexity of $\Theta(m \cdot n \cdot o)$ results from m as the average number of relevant interaction partners, n as the average number of relevant predicates and o as the number of active nodes in the simulation. In the worst case, all nodes have to be tested among themselves with respect to all

p implemented predicates. Then, the complexity for predicate testing rises to $\mathcal{O}(o^2 \cdot p)$.

There are two features of the SGG algorithm that could be immediately utilized to improve the runtime of a simulation. First, the computational effort for executing a relation might be reduced by recycling intermediately computed results as, for instance, implemented for neighborhood testing in [8]. Second, with a strict read-only policy for predicative relations, computing the SGG simulation could be distributed to a parallel computing system with shared memory. Synchronization would be required after each of the four steps in the main loop (Algorithm 5).

7.2 SGG examples

In the previous section, SGGs were introduced as a graph-based, developmental multi-agent model. SGGs do not qualify as a programming language, as they do not require the involved relations to be implemented in any particular syntax. They present, however, a powerful, unifying modeling framework. In this section we utilize SGGs to successively model boids, basic swarm grammars and extended swarm grammars.

7.2.1 Boids with SGGs

For standard boid flocking [13], the sole genotype $g_{boid} \in Gen$ contains several weights for flocking urges, parameters to determine a field of perception, as well as boundaries for the maximal flight acceleration max_{accel} and velocity max_{vel}. Ξ generates a homogeneous set of swarm individuals that are initialized with a random position $\overrightarrow{p}$ and velocity $\overrightarrow{v}$.

Having initialized the boid simulation, a first predicative graph G_{pred} can be computed. Figure 7.1(a) visualizes the swarm individuals' states, i.e. the swarm individuals' relative locations $\overrightarrow{p}$, their orientations according to $\overrightarrow{v}$ and their parametrically determined perceptional fields. Figure 7.1(b) shows G_{pred} with the only predicative relation ν among the swarm individuals.

$\nu(u, v)$ means that the location of v is within the perceptional field of u.

As the agent simulation proceeds with yielding the system's detailed global state in G_{pred}, the swarm individuals are prompted to act. Boids rely on two behavioral rules. The first rule is depicted in Figure 7.4(a). It continuously updates a swarm individual's position in accordance with its velocity. The second rule, Figure 7.4(b), substitutes $\nu(u, v)$ by $accel_{boid}(u, v)$. The latter relation considers the difference between u and v's states, including their locations and velocities, and accelerates u accordingly.

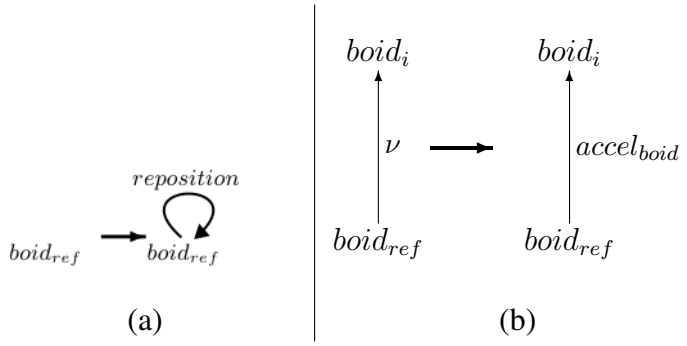


Figure 7.4: (a) Unconditionally, the individual $boid_{ref}$ repositions itself. (b) The reference node updates its state dependent on any neighbor $boid_i$.

In particular, $accel_{boid}(u, v)$ calculates an acceleration vector $\overrightarrow{a^u}$ for the individual u according to the following equation:

$$\overrightarrow{a^u} = w^u_{align}(\overrightarrow{v^v} - \overrightarrow{v^u}) + \tag{7.1}$$

$$w^u_{coh}(\overrightarrow{p^v} - \overrightarrow{p^u}) + \tag{7.2}$$

$$w^u_{sep}(\overrightarrow{v^u} \otimes (-\angle(\overrightarrow{v^u}, \overrightarrow{p^v} - \overrightarrow{p^u}))) + \tag{7.3}$$

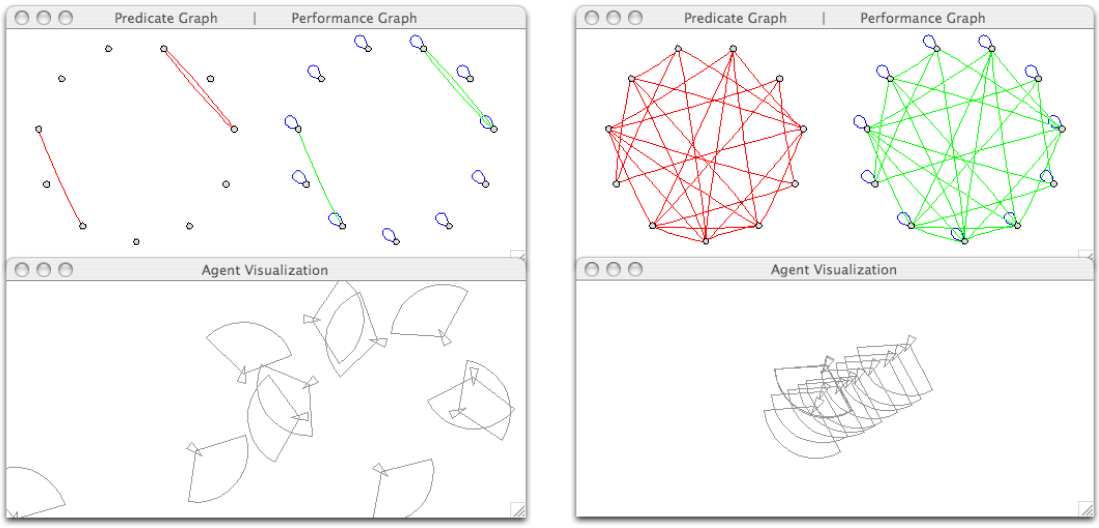
$$w^u_{rand}||\overrightarrow{v}_r|| \tag{7.4}$$

In analogy to the boids implementation of previous SG systems (Section 3.1.2), the weights w^u_x are encoded in u's genotype. Term 7.1 is responsible for u's alignment to v: The velocity difference, and thereby the angle between the two individuals, is reduced. Term 7.2 keeps the individuals together by decreasing their distance. In Term 7.3 u's orientation is rotated, $\otimes$, by

its deflection vector from v, thus realizing an urge for separation. Eventually, Term 7.4 adds an accordingly weighted, normalized random vector $\overrightarrow{v}_r$, resulting in some nondeterminism in the boid's flight. The calculated acceleration vector may not exceed the absolute value of max_{accel}. Furthermore, when $accel_{boid}$ is eventually enforced, the resulting velocity is kept at or below max_{vel}. Note that different from previous boid implementations, the urges for separation (Term 7.3) and cohesion (Term 7.2) are not simply diametrically exercising forces—instead, separation effects a deflection of an agent. Thus, no differentiation between the perceptional range d and the crowding constant d_{min} has to be made to switch the separation urge on or off. Instead, the separation urge continuously impacts the agents' flight like the other basic boid urges alignment and cohesion.

In the following example, g_{boid} provides the weights $w_{align} = 1.0$, $w_{coh} = 0.3$, and $w_{sep} = 0.2$, the parameters $d = 50$ and angle $\alpha = 2.0$ for a conic field of perception, and the restraining flight parameters $max_{vel} = 5$ and $max_{accel} = 15$. In Figure 7.5(a) and (b) as well as in the subsequent figures of SGG simulations, the predicative graph G_{pred}, the performance graph G_{perf} and a visualization of the simulation are shown. In accordance with the boid rule in Figure 7.4(a), all boids apply a reposition procedure, indicated by the self-referencing edge. The second boid rule, displayed in Figure 7.4(b), substitutes neighborhood relations ν in G_{pred} with $accel_{boid}$ relations in G_{perf}. The neighborhood relations can be verified qualitatively, when observing the simulation visualization at the bottom of Figure 7.5. Agents that are close to each other fulfill the neighborhood relation, as indicated by the edges.

Running the boid simulation for several hundred iterations, the boids cluster. This effect cannot only be seen when looking at the actual movement of the agents, but also when looking at the degree of connectedness of G_{pred} and G_{perf} in Figure 7.5(b).



(a) (b)

Figure 7.5: (a) G_{pred}, G_{perf} and a two-dimensional visualization of a SGG-driven boids simulation. (b) After several hundred iterations of the SGG main loop (Algorithm 5), the boid agents have clustered.

7.2.2 Swarm grammars with SGGs

Swarm grammars extend the boid model with (1) the agents leaving traces in space and (2) being able to reproduce. Additionally, the extended swarm grammar model equips the agents for (3) event-triggered reproduction and construction. In the following paragraphs, behavioral rules are added to the boid individuals, in order to consecutively extend the model and to retrace the three outlined aspects of basic and extended swarm grammars.

Particle Traces

Adding the two rules displayed in Figure 7.6 to g_{boid} yields a new genotype g_{constr} which results in continuously constructing swarm individuals. Without precondition, $boid_{ref}$ applies the performance relation $timer_{++}$ to itself, increasing an internal variable $time \in Dat$ with a value set $T = [0, \phi]$ (Fig. 7.6(a)). $timer_{>\phi}$, on the other hand, tests whether the internal variable $time$ of $boid_{ref}$ is greater than a given threshold ϕ. If the predicate $timer_{>\phi}$ is true, $boid_{ref}$ creates and initializes a new node called "particle" with the performance relation $init_P$.

The initialization performance $init_P$ assigns a particle genotype $g_{particle}$ to the newly created node, which determines its visual properties and renders it motionless and indifferent to its environment. It also positions $particle_0$ just behind $boid_{ref}$. Figure 7.7 (a) shows the first simultaneous construction process of a group of agents. In G_{perf} the new nodes are added next to their creator nodes. Figure 7.7 (b) depicts a later stage of the constructive swarm simulation, where some swarm individuals perceive several particles in their neighborhood but do not react to them.

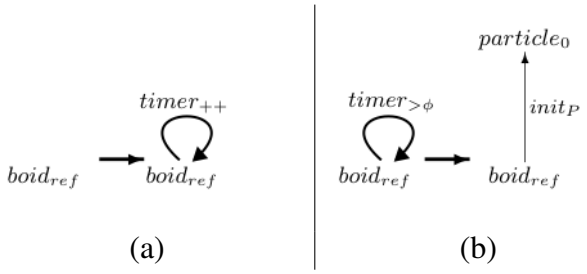
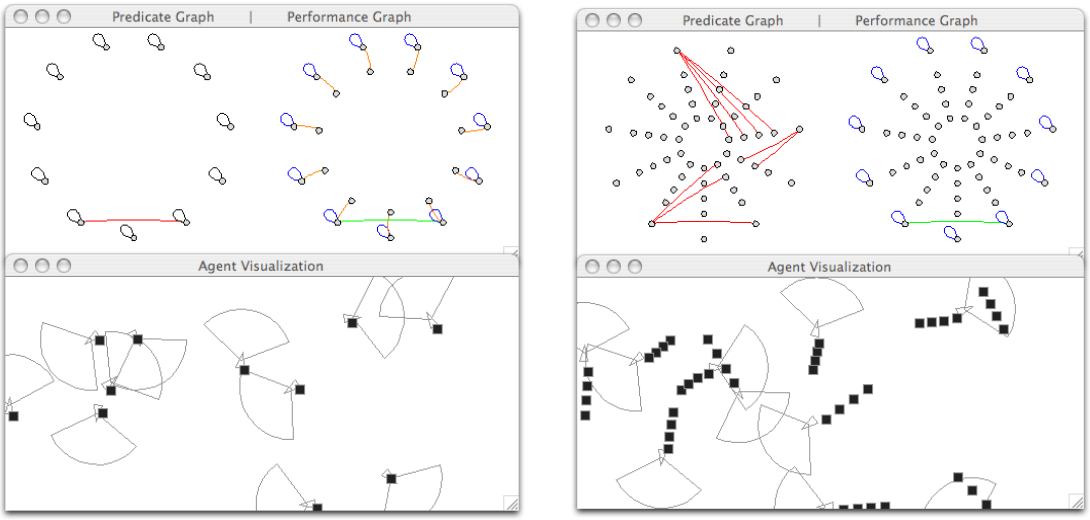


Figure 7.6: (a) $boid_{ref}$ counts up an internal variable via $timer_{++}$. (b) When its internal variable exceeds a certain threshold ϕ, it creates a new node and initializes it as a "particle".



(a) (b)

Figure 7.7: (a) Constructive swarm individuals simultaneously create and initialize new particle nodes (black squares in the visualization). (b) Several iterations into the constructive swarm simulation, some swarm individuals perceive particles in their neighborhoods.

Agent reproduction

When the swarm individuals create new nodes and pass on their own genotypes instead of $g_{particle}$, the number of agents quickly increases and the particle traces branch out. Adding two corresponding rules (Figure 7.8) leads to the evolution of the simulation displayed in Figure 7.9. As the interval of the timer that is introduced to trigger the reproduction overlaps with the construction timer, two self-referencing predicative relations can be seen, when the agents proliferate. New swarm individuals are added to the perimeter of the graphs G_{pred} and G_{perf}.

The emerging structure in the visualization window in Figure 7.9(b) is typical for many grammatical developmental systems. The number of nodes close to the center of the graphs show how productive their builder agents in the perimeter have been.

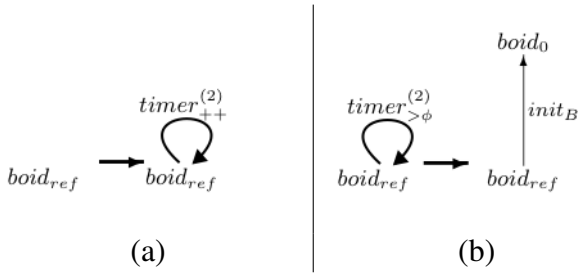
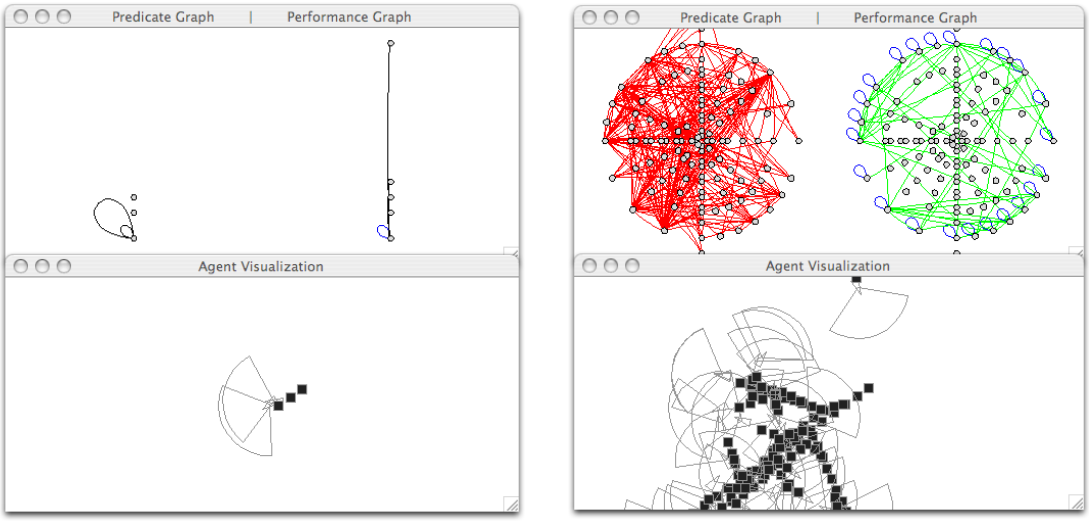


Figure 7.8: (a) A second timer is maintained through $timer^{(2)}_{++}$. (b) The newly introduced timer triggers proliferation.



(a) (b)

Figure 7.9: (a) $timer_{>\phi}$ tiggers the construction of a particle while $timer^{(2)}_{>\phi}$ causes the agent to proliferate. (b) A branching structure emerges.

Stigmergic interactions (extended SGs)

As a stigmergic interaction example, the swarm individuals are instructed to remove particles on sight. The genotype g_{constr} is extended with an according rule (Figure 7.10). The resulting simulation is shown in Figure 7.11. Note the seemingly random regression of particle nodes in G_{perf} and G_{pred}, respectively.

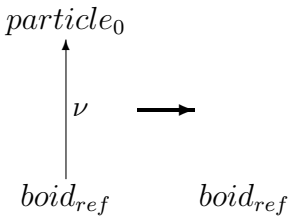
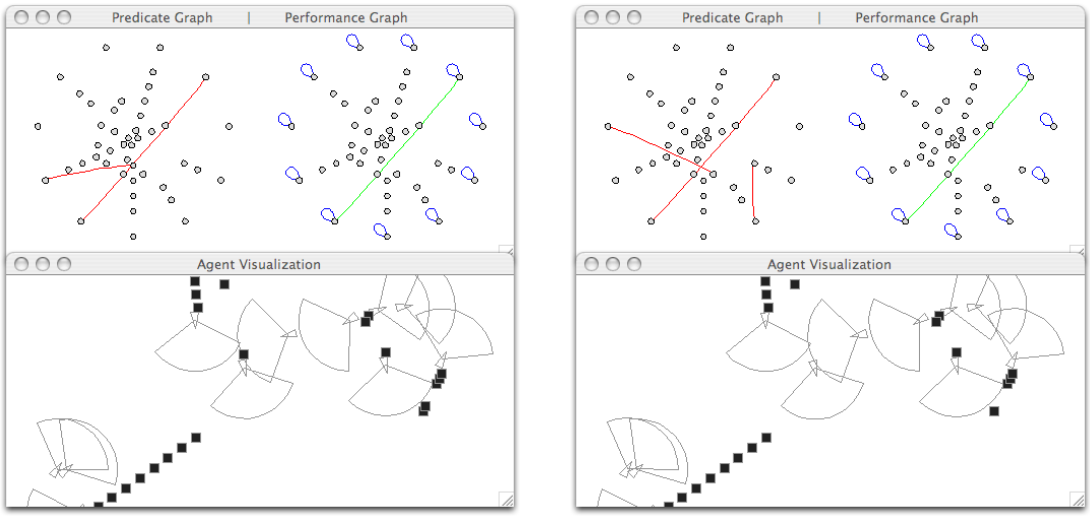


Figure 7.10: A particle is removed on sight.



(a) (b)

Figure 7.11: (a) One of the individuals on the perimeter of G_{pred} perceives a particle, situated closer to the center of the graph. (b) The particle has disappeared from the visualization at the bottom.

7.3 Status quo of swarm graph grammars

The presented algorithmic concepts and implementations show how dynamic swarm networks can be simulated beyond the scope of the spatially staged interplay of boid agents as investigated in Chapter 6. Instead, by means of graphical rule representation and the visualization of the emergent interaction networks, any kinds of relationships can be perceived and analyzed as contributors to the systems' dynamics.

Swarm graph grammars are a framework that drives computational processes through node-oriented substitution of subgraphs. The sum of local relations results in global networks of predicative and performance relations. SGGs incorporate the basic mechanisms of relational queries and performances, node creation and deletion. However, actual problem-specific implementations of the required predicative and performance relations have to be provided. They are not part of the algorithmic framework. For illustrative purposes of the functionality of swarm graph grammars, movement, perception, reproduction and construction as well as an according set of node configurations are introduced to retrace boids, basic swarm grammars and extended swarm grammars in a concise and unified algorithmic manner.

Chapter 8

Summary & future work

The contributions of this thesis touch upon numerous and diverse fields of research. Some of them are general statements or suggest certain perspectives on well-developed concepts. Others bring about new methodologies and create novel spaces for further investigation. However, an intrinsically motivated search for understanding complex phenomena and the possibility to harness artificial productivity and creativity drove the sequence of investigations and explorations. Although several gaps needed patching and some of the chapters required the re-structuring of contents, the presented order of the conducted projects corresponds well with their timely succession.

In analogy to the chapter on related work that started out with the impressive architectural examples built by social insects, the contribution of this thesis began with the formulation of an abstract swarm-inspired CDM. Subsequently, the related work chapter touched upon developmental systems in general, computational creativity, the role of computational evolution, and ended with investigations of complexity in evo-devo, CDM and swarm systems. The stepwise development of swarm grammars (basic, extended and graph-based) retraced this chain of arguments. Eventually, we arrived at the presentation of SGGs, a powerful representation of CDMs that focusses on interwoven interaction processes. In addition to its expressiveness, it provides a means for modeling and analyzing highly dynamic, complex systems.

Firstly, we are going to summarize the contributions of each chapter and explain their underlying motivations. We show how the train of thought that guided this work confirms its resolution and elucidates its purpose. Possible future research projects come naturally when following the explanations of the presented work. Secondly, we present several short- and long-term goals that directly build upon this thesis' contributions.

8.1 Chapter-based résumé

Chapter 2 shed light on the outstanding constructions of social insect orders. In this context, computational developmental systems were presented that strongly abstract from any natural developmental systems, yet integrate important aspects such as reproduction and neighborhood relationships. Even though CDMs are much simpler than any biological developmental models, they were soon applied to promote artificial creativity and productivity. Evolutionary computation has often provided the means to drive these constructive processes, to develop CDMs so to speak. In this context, we mentioned several examples that generate art, design and architecture and we led over to the category of specially designed computational evo-devo systems. These lend themselves as example systems that easily reach complexities that are hardly possible to engineer. The swarm metaphor of endless, recursively interwoven interaction patterns provides an image that conveys a notion about these complexities; This insight finalized the related work chapter—a loop was knotted from the motivating chain of arguments. This loop in turn provided the starting point for the remainder of this thesis, whereas all the previously mentioned topics were investigated and explored in respect to the swarm metaphor.

In order to conduct any studies in the realm of complex, developmental swarm systems, a principal representation had to be created. One that integrates swarm dynamics with the productivity exhibited by other CDMs. Hence, we formulated swarm grammars (Chapter 3). Entering new territory, we cautiously explored the capabilities, the behaviors and the productivity of the novel CDM. The first swarm grammar representation was kept simple and solely merged the boids model with L-systems by interpreting one symbol of a generative L-system grammar as one flocking agent. As soon as we felt comfortable to observe and hone the systems' configurations, we added expressiveness to the representation by considering communication mechanisms exhibited by social insects. This concluded the extended swarm grammar model.

Before reaching this point, we had already started to rely on computational evolution to

breed aesthetically appealing SG structures (Chapter 4). We employed interactive breeding techniques as we were still exploring the growth and development of SGs, without considering any hard structural or functional constraints. However, we also explored new means to boost the search for visually engaging SGs. The outcome was an immersive breeding scenario that allowed an external breeder to tinker with SGs like a gardener with his plants. While exploring the means of computational evolution, the discoveries made during our breeding experiments propelled new exploratory excursions in collaborative arts projects. But the extension of the SG model by means of stigmergic communication rendered an automatic breeding approach inevitable.

This necessity heralded a formal task description for SGs. The appreciation of architecture and the inspirational capability of construction of social insects let us promote swarm-driven architecture (Chapter 5). Obviously, the first automatically evolved SGs should not be overwhelmed by a multitude of functional requirements. Therefore, we aimed for the relatively simple task to build non-functional architectural idea models, three-dimensional structures that are created at an early stage of the architectural design process. We successfully fostered creative and diverse results by assigning high fitness values to SGs based on their levels of diversity, collaboration and communication during the construction processes.

This aspect of our implemented fitness evaluation triggered deeper investigations into the realm of swarm complexity—investigations of the interaction networks that emerge during swarm simulations (Chapter 6). Here, we observed the progression of neighborhood densities in boid flocks. This complexity measure allowed us to identify phase transitions and quasi stable states of previously evolved boid configurations. When we provided continuous graphs of an average density function, boid configurations evolved that retraced the corresponding values. Our results showed how interaction complexity can arise in 3D space and cause global emergent phenomena such as one-way phase transitions or oscillations.

As seen in the examples of Chapter 7, swarm graph grammars are a successful attempt to

design one unifying representation that supports various artificial swarm models. Rule execution advances to the principal computational step. In basic swarm grammars for instance, explicit grammatical substitution rules are only applied to introduce the aspects of construction and proliferation and are considered independently of the agents' flocking behavior. SGGs neither differentiate between the creation of new particles or agents: Instead, it depends on the constructor, or ancestor respectively, to determine the kind of node it introduces into the system. This is especially important as the level of abstraction—whether a plant cell should, for instance, grow in size, or whether it simply serves as a volumetric unit in a graphical structure—is completely assigned to the behavioral model.

The boid model only considers neighborhood dependencies and induces local changes in the agents' accelerations. In constructive swarms, in contrast, a diverse set of relations governs the interaction processes. For this reason we designed SGGs. By means of SGGs not only spatial interrelations (neighborhood relationships) may guide a swarm simulation but any kind of predicative relation. Similarly, the agents are not limited to adjusting their velocities but they may affect any of their internal data and their global states. Together, the formulation of predicative and performance relations is a generalization of the basic boid system that solely acts based on location changes and in a three-dimensional space. Yet, the relationship dynamics observed in boid systems is still applicable—in a multi-dimensional space of dependencies that cannot be easily visualized. However, SGGs provide the means to model interdependencies graphically and to systematically analyze the relationship graphs that emerge during the simulation processes.

8.2 Future work

In this section we present several ideas for future research projects that directly surfaced from the scientific outcomes of this thesis. Partially, we dedicate separate paragraphs to distin-

guishable topics as before. Additionally, however, we also draw images that necessitate a comprehensive and well-coordinated research approach.

8.2.1 Swarm-driven architecture

To foster our work on swarm-driven architecture, the following steps could be immediately considered.

1. Integration of Chapters 5 and 6: Investigation of the temporal development of the perceived neighborhood ratio r_n, or respectively the neighborhood density $\bar{n}(t)$, to further elucidate the impact on the built constructions.

2. Exploration of the impact of alternative pre-defined shapes (e.g. convex geometries) on the diversity and the design of emerging architectures.

3. Protocols of our evolutionary experiments underline the importance of an effective fitness function and effective genetic operators. Here, too, further investigation is necessary to find an optimum for diverse, appealing, and fit constructions that further facilitate the exploration of architectural idea spaces.

4. Too great evolutionary pressures could result in very narrow assortments of bred designs. To counter this effect of evolutionary over-fitting, we suggest the deployment of scalable complexity measures that link the SG genotypes with the built architectures [153].

5. Evolution of specialist builders that fulfill certain construction tasks (also in respect of their flocking behavior). Step-wise construction of a heterogeneous SG that incorporates a hierarchically evolved differentiation policy. For these experiments, we could rely on the hierarchical breeding scenarios presented in [121].

6. Possible improvements regarding the quality and diversity of individually bred SGs could be achieved through changing the global conditions of the evolutionary framework, e.g.

through cyclic promotion of generalist and specialist individuals [187, 188], or through improving the evolutionary operators that work on the genetic codes, e.g. through the automatic adjustment of a genotype's length [189].

7. Swarms that operate on 3D meshes instead of building freely in space. The vertex-vertex system presented in [129] would directly support this idea.

The need for functional interior design

Evolutionary swarm design of architectural idea models works. However, in order to render this technology applicable for architects it has to be fitted according to their needs. Hereby, the main goals are an interactive way to promote the development of compelling designs as well as the introduction of stronger constructional elaborations. The need to incorporate architectural and constructional functionality could be addressed by a novel kind of artificial swarms that we refer to as *consumer swarms* as opposed to constructive swarms. Christian Jacob expressed the idea of such swarms that follow certain movements based on a regular routine. During their flight they could assign fitness values to the exposed structures, thus driving the evolution of the interior design of SG architecture. Here, the sequence of movements would have to reflect the ontology of certain activities. Hybrids of consumer and constructive swarms (CC hybrids) could actively change and improve the constructional design while utilizing the buildings according to their routines. Instead of constructing the whole building, these hybrids could work on pre-defined architectures and 'carve them out'. Again, swarm agents that manipulate 3D meshes would be well-suited for this task. In this context, Gerald Hushlak underlined the importance of the destruction of old structures to create the space for novel developments. Although the possibility of the removal of construction elements in developmental swarm processes has been implemented and tested, comprehensive studies on its impact on emergent designs need to be conducted.

Introducing ecological features

Regarding the coarse constructional layout of a building, the evolution of SG architecture could be driven by measures that consider the ecological and economic features of a building [103, 104, 107]. There are measures as simple as the reduction of required construction material but also those that demand for complex physical simulations such as the streams of ventilation or the demand for thermal energy. Jörg Denzinger was the first to suggest using swarms for these processes as well—instead of extensive numerical simulations (e.g. grid-based diffusion), the swarm agents' perceptional capabilities and high-level behaviors could drastically reduce the involved computational cost. For the development of course-grained architectural models, this approach might bear great potential.

Building in context

Interior design and ecological features are very important corner stones of architecture. However, a building is also placed into an environmental context comprising landscape and climate, neighboring buildings and infrastructure and cultural, traditional expectations. In order to consider these factors, we suggest the following three steps: (1) Modeling the existing environment as closely as necessary. (2) Entrain CC hybrid swarms to mimic the locally established behaviors of movement. (3) Evolve SG architectures and deploy the CC hybrids to adapt the evolved architecture.

Additionally, if there is a great similarity among the existing buildings, one might consider extracting their basic features to create a pre-defined prototype architecture that CC hybrid swarms should learn to decorate and optimize.

8.2.2 Working with swarm graph grammars

SGGs have the potential to evolve complex structures. As in [58], graph-based genetic operators can be introduced to drive evolutionary processes. Based on comprehensive analyses of the network dynamics, an evolutionary development of desired interaction cycles can be

achieved [6]. Evolutionary pressure can be exercised based on a diverse set of complexity measures. In SGGs the behavioral rules, the emerging graphs, the interaction networks over periods of time, or fixed generated structures could be objected to these measures. Especially the observation of interaction dynamics over time might reveal important information to breed systems of higher-order complexity [138].

Building hierarchies through functional subnetworks

When functional subsystems of an SGG's emergent network are discovered, machine learning techniques [118] could be applied to identify their dependencies over time. The resulting knowledge about required influx and produced output of an according subsystem would then allow to disguise its inner workings as a *black box*. Obviously, complex spatial and functional bottom-up hierarchies could be built through repetition of this process of abstraction. In case the requirements that drive a subsystem are not met anymore, its modularization could be revoked and the impact of the changed environmental conditions would be computed from the top to the bottom of the hierarchy of dependencies. We believe such an automatic bottom-up/top-down approach bears the potential to scale the simulation of complex computational systems. From a modeling perspective, the outlined approach bridges between SGGs that incorporate locally defined agent interactions and emerging interaction cycles and the compartmentalized formal modeling approach of *membrane computing* (MC) [190, 191].

Opening dimensions through graphical representation

In developmental SGGs, mapping the systems' predicative and performance graphs on two dimensions quickly becomes confusing. Therefore, we advocate the transition to a three-dimensional graph-visualization with the ability to move and browse through virtual space. In such an immersive environment, user-interference could be promoted through highlighting schemes for visual and automatic stochastic analysis of certain areas of the graphs.

Drawing the relational graphs of an SGG precisely illustrates the computational interde-

pendencies and interactions of a simulation, but on an abstract level. When isolating certain relations, completely new perspectives on the simulation and on its results could be gained. Non-spatial relations would be rendered spatially and therefore become accessible to our minds prone to everyday life.

8.2.3 Virtual walk-through of an interactive swarm design process

Imagine the following scenario [192]. An interdisciplinary team of researchers has gathered around a table. A holographic, three-dimensional still image hovers several inches above the desktop. Before the researchers debate how to tackle the presented problem, each tries to capture the problem alone. After several minutes have passed, the experts describe what they see through their own eyes facing the challenge. They explain what an ideal solution to the problem might look like, point out the differences to the visualization in front of them and suggest ways to get from the status quo to the desired goal. Some members of the group realize that the spoken word defines and confines meaning within the parameters of language, others point out the need for additional semantics and offer visual cues to their unconscious that are outside of language. After listening to each others opinions, the meeting is adjourned.

For the following session the team members prepare visual representations of the problem and ideal solutions defined in a pragmatic way. A resulting three-dimensional graph is blended into the projection that again occupies the space above the table. The most relevant units in the scope of the problem are represented as spherical nodes. The nodes are connected through spatial edges that indicate relationships among the involved units. While explaining the model, the original three-dimensional image of the problem fades out slightly in order to highlight the experts theoretical, graphical model of the situation and its solution. Fading is good because thought becomes more abstract and is not confined to the constraints of the display. The experts begin to discuss the tangibles of the teams understanding of the design task. The team agrees to consider those corner stones that provide models that deviate from the normative expectations.

During the next meeting, sets of solutions are reviewed that were discovered in the implicitly defined space of possible solutions. The solutions that the team prefers require a readjustment from first principles. This process of model improvements may be repeated several times. In the meantime, small changes could be manually or automatically introduced into the model on various scales to analyse the robustness of the interaction networks of the solution. New aspects become obvious throughout the inspection and revision sessions. For instance, unbearable costs and implementation times of the suggested solutions might call for improvements. Eventually, the team chooses among the presented solutions. Details on the transition from the status quo to the result are revealed by the projective computational engine and studied by the team. Knowing what has to be done to master the challenge, the team maps the theoretical solution to a project plan and takes the appropriate actions.

The expert team can address anything in Anyville: The combination of swarm grammars as bio-inspired generative representation and evolutionary exploration of innovative designs opens up an array of possibilities to develop dynamic processes, like gene regulatory processes [193, 194], immune systems [10, 195], swarm choreographies [83], interactive SwarmArt installations [86, 196] and (post-)modern ecological architectural designs. Of course, the outlined design process describes but one methodological projective scenario that integrates interdisciplinary expertise, teamwork, graphical modelling, complex simulation and bio-inspired learning techniques. The presented computer-supported team-play is a viable and potentially fruitful scenario for tackling complex problems. It outlines one visionary and productive way to integrate the concepts of swarm intelligence and evolutionary computation that were presented in this thesis.

Bibliography

[1] S. von Mammen, "Swarm grammars - a new approach to dynamic growth," technical report, University of Calgary, Calgary, Canada, May 2006.

[2] C. Jacob and S. von Mammen, "Swarm grammars: growing dynamic structures in 3d agent spaces," *Digital Creativity: Special issue on Computational Models of Creativity in the Arts*, vol. 18, pp. 54–64, March 2007.

[3] S. von Mammen and C. Jacob, "Genetic swarm grammar programming: Ecological breeding like a gardener," in *2007 IEEE Congress on Evolutionary Computation* (D. Srinivasan and L. Wang, eds.), IEEE Press, pp. 851–858, 2007.

[4] S. von Mammen and C. Jacob, "Evolutionary swarm design of architectural idea models," in *Genetic and Evolutionary Computation Conference (GECCO) 2008*, (New York, NY, USA), pp. 143–150, ACM Press, 2008.

[5] S. von Mammen, J. Wong, and C. Jacob, "Virtual constructive swarms: Compositions and inspirations," in *Applications of Evolutionary Computing, Proceedings of EvoWorkshops 2008*, vol. 4974 of *Lecture Notes in Computer Science*, (Berlin-Heidelberg), pp. 491–496, Springer-Verlag, 2008.

[6] S. von Mammen and C. Jacob, "Swarm-driven idea models - from insect nests to modern architecture," in *Eco-Architecture 2008, Second International Conference on Harmonisation Between Architecture and Nature* (C. Brebbia, ed.), (Winchester, UK), pp. 117–126, WIT Press, 2008.

[7] S. von Mammen and C. Jacob, "The spatiality of swarms — quantitative analysis of dynamic interaction networks," in *Proceedings of Artificial Life XI*, pp. 662–669, MIT Press, 2008.

[8] J. Klein, "breve: a 3d simulation environment for multi-agent simulations and artificial life.." `http://www.spiderland.org/`, October 2008.

[9] J. Schneider, "Website of Jürgen Schneider." `http://www2.informatik.uni-erlangen.de/Personen/schneide/?language=en`, March 2009.

[10] C. Jacob, J. Litorco, and L. Lee, "Immunity through swarms: Agent-based simulations of the human immune system," in *Artificial Immune Systems, ICARIS 2004, Third International Conference*, (Catania, Italy), LNCS 3239, Springer, 2004.

[11] D. Dasgupta, "Advances in artificial immune systems," *Computational Intelligence Magazine, IEEE*, vol. 1, no. 4, pp. 40–49, Nov. 2006.

[12] M. Resnick, *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds*. Complex Adaptive Systems, Cambridge, MA: MIT Press, 1997.

[13] C. W. Reynolds, "Flocks, herds, and schools: A distributed behavioral model," in *SIGGRAPH '87 Conference Proceedings*, vol. 4, pp. 25–34, 1987.

[14] G. Theraulaz and E. Bonabeau, "Modelling the collective building of complex architectures in social insects with lattice swarms," *Journal of Theoretical Biology*, vol. 177, no. 4, pp. 381–400, 1995.

[15] J. Glancey, *Story of Architecture*. Dorling Kindersley, 2001.

[16] B. Hölldobler and E. O. Wilson, *The Ants*. Berlin-Heidelberg: Springer-Verlag, 1990.

[17] S. Levy, *Artificial Life: A Report from the Frontier where Computers Meet Biology*. Vintage Books, A Division of Random House, Inc., 1993.

[18] S. Kauffman, *The origins of order*. Oxford Univ. Press New York, 1993.

[19] K. von Frisch, *Animal Architecture*. Harcout Brace Jovanovich, New York, 1974.

[20] B. Hall and W. Olson, *Keywords and concepts in evolutionary developmental biology*. Harvard University Press, 2003.

[21] H. Honour and J. Fleming, *Weltgeschichte der Kunst*. Munich, Germany: Prestel, 2007.

[22] I. Flagge, R. Schneider, and D. Architekturmuseum, *Die Revision Der Postmoderne: Post-modernism Revisited:[in Memoriam Heinrich Klotz]*. DAM, Deutsches Architekturmuseum, 2004.

[23] W. Thaler and B. Hölldobler, "Ants - nature's secret power." Documentary Film. ORF - Natural History Unit. Vienna, Austria., 2004.

[24] I. Karsai and Z. Penzes, "Comb building in social wasps: Self-organization and stigmergic script," *Journal of Theoretical Biology*, vol. 161, no. 4, pp. 505–525, 1993.

[25] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*. Santa Fe Institute Studies in the Sciences of Complexity, New York: Oxford University Press, 1999.

[26] G. Theraulaz and E. Bonabeau, "Coordination in Distributed Building," *Science*, vol. 269, no. 5224, pp. 686–688, 1995.

[27] S. Camazine, J.-L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau, *Self-Organization in Biological Systems*. Princeton Studies in Complexity, Princeton: Princeton University Press, 2003.

[28] P.-P. Grassé, "La reconstruction du nid et les coordinations interindividuelles chezbellicositermes natalensis etcubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs," *Insectes Sociaux*, vol. 6, no. 1, pp. 41–80, 1959.

[29] M. Hansell, *Built by animals: the natural history of animal architecture*. Oxford University Press, USA, 2007.

[30] R. Jeanne, "The Adaptiveness of Social Wasp Nest Architecture," *The Quarterly Review of Biology*, vol. 50, no. 3, pp. 267–287, 1975.

[31] A. Smith, S. O'Donnell, and R. Jeanne, "Correlated evolution of colony defence and social structure: A comparative analysis in eusocial wasps(Hymenoptera: Vespidae)," *Evolutionary Ecology Research*, vol. 3, no. 3, pp. 331–344, 2001.

[32] D. Ladley and S. Bullock, "Logistic constraints on 3d termite construction," in *Fourth International Workshop on Ant Colony* (M. Dorigo, M. Birattari, L. M. Blum, F. Mondada, and T. Stutzle, eds.), pp. 178–189, Springer, Berlin, 2004.

[33] F. Roces and J. Núñez, "Thermal sensitivity during brood care in workers of two camponotus ant species: Circadian variation and its ecological correlates," *Journal of Insect Physiology*, vol. 41, no. 8, pp. 659 – 669, 1995.

[34] F. Roces and C. Kleineidam, "Humidity preference for fungus culturing by workers of the leaf-cutting ant atta sexdens rubropilosa," *Insectes Sociaux*, vol. 47, no. 4, pp. 348–350, 2000.

[35] C. Kleineidam and F. Roces, "Carbon dioxide concentrations and nest ventilation in nests of the leaf-cutting ant atta vollenweideri," *Insectes Sociaux*, vol. 47, no. 3, pp. 241–248, 2000.

[36] C. Kleineidam, R. Ernst, and F. Roces, "Wind-induced ventilation of the giant nests of the leaf-cutting ant atta vollenweideri," *Naturwissenschaften*, vol. 88, no. 7, pp. 301–305, 2001.

[37] J. Korb, "Thermoregulation and ventilation of termite mounds," *Naturwissenschaften*, vol. 90, no. 5, pp. 212–219, 2003.

[38] S. Kumar and P. J. Bentley, "Biologically inspired evolutionary development," *Evolvable Systems: From Biology to Hardware*, pp. 99–106, 2003.

[39] A. R. Smith, "Plants, fractals, and formal languages," *SIGGRAPH Comput. Graph.*, vol. 18, no. 3, pp. 1–10, 1984.

[40] J. von Neumann and A. W. Burks, *Theory of self-reproducing automata*. Urbana and London: University of Illinois Press, 1966.

[41] W. A. Beyer, P. H. Sellers, and M. S. Waterman, "Stanislaw m. ulam's contributions to theoretical theory," *Letters in Mathematical Physics*, vol. 10, pp. 231–242, 1985.

[42] E. F. Codd, *Cellular Automata*. New York, NY, USA: Academic Press, 1968.

[43] C. G. Langton, "Self-reproduction in cellular automata," *Physica D: Nonlinear Phenomena*, vol. 10, no. 1-2, pp. 135–144, 1984.

[44] J. Reggia, J. Lohn, and H. Chou, "Self-replicating structures: evolution, emergence, and computation," *Artificial Life*, vol. 4, no. 3, pp. 283–302, 1998.

[45] S. L. Miller, "A production of amino acids under possible primitive earth conditions," *Science*, vol. 117, no. 3046, pp. 528–529, 1953.

[46] W. Banzhaf, "Artificial chemistries - towards constructive dynamical systems," *Solid State Phenomena*, pp. 43 – 50, 2004.

[47] O. Patry, "Organic builder: An artificial chemistry simulation." `http://organicbuilder.sourceforge.net`, March 2009.

[48] J. Z. Peter Dittrich and W. Banzhaf, *Artificial Chemistries - A Review*, pp. 225 – 275. MIT Press, 2001.

[49] A. Lindenmayer, "Developmental systems without cellular interactions, their languages and grammars," *Journal of Theoretical Biology*, vol. 30, no. 3, pp. 455–484, 1971.

[50] P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants*. Springer-Verlag, 1996.

[51] M. Frame and Ginger Booth, "Online l-system simulator." `http://classes.yale.edu/fractals/Software/lsystem.html`, March 2009.

[52] C. Jacob, *Illustrating Evolutionary Computation with Mathematica*. San Francisco, CA: Morgan Kaufmann Publishers, 2001.

[53] M. J. M. d. Boer and M. d. Does, "The relationship between cell division pattern and global shape of young fern gametophytes. i. a model study," *Botanical Gazette*, vol. 151, no. 4, pp. 423–434, 1990.

[54] J.-L. Giavitto, C. Godin, O. Michel, and P. Prusinkiewicz, *Modelling and Simulation of biological processes in the context of genomics*, ch. Computational Models for Integrative and Developmental Biology. Hermes, 2002.

[55] J.-L. Giavitto and O. Michel, "Modeling the topological organization of cellular processes," *Biosystems*, vol. 70, no. 2, pp. 149–163, 2003.

[56] A. Spicher, O. Michel, and J.-L. Giavitto, "A topological framework for the specification and the simulation of discrete dynamical systems," *Cellular Automata*, pp. 238–247, 2004.

[57] J.-L. Giavitto and O. Michel, "Data structure as topological spaces," *Unconventional Models of Computation*, pp. 137–150, 2002.

[58] O. Kniemeyer, G. H. Buck-Sorlin, and W. Kurth, "A graph grammar approach to artificial life," *Artificial Life*, vol. 10, no. 4, pp. 413–431, 2004.

[59] O. Kniemeyer, G. Buck-Sorlin, and W. Kurth, "Groimp as a platform for functional-structural modelling of plants," in *Functional-Structural Plant Modelling in Crop Production* (J. Vos, L. F. M. Marcelis, P. H. B. deVisser, P. C. Struik, and J. B. Evers, eds.), pp. 43–52, Springer, March 2006.

[60] O. Kniemeyer, G. Barczik, R. Hemmerling, and W. Kurth, "Relational Growth Grammars—A Parallel Graph Transformation Approach with Applications in Biology and Architecture," *Lecture Notes In Computer Science*, pp. 152–167, 2008.

[61] W. Kurth, G. Buck-Sorlin, and O. Kniemeyer, "Relationale wachstumsgrammatiken: Ein formalismus zur spezifikation multiskalierter struktur-funktions-modelle von pflanzen," *Modellierung pflanzlicher Systeme aus historischer und aktueller Sicht. Symposium zu Ehren von Prof. Dr. Dr. h.c. Eilhard Alfred Mitscherlich*, no. 7, pp. 36–45, 2006.

[62] K. Culik and A. Lindenmayer, "Parallel graph generating and graph recurrence systems for multicellular development," *International Journal of General Systems*, vol. 3, no. 1, pp. 53–66, 1976.

[63] M. Nagl, "On the relation between graph grammars and graph l-systems," *Fundamentals of Computation Theory*, pp. 142–151, 1977.

[64] A. Lindenmayer, "An introduction to parallel map generating systems," *Graph-Grammars and Their Application to Computer Science*, pp. 27–40, 1987.

[65] N. Chomsky, "Three models for the description of language," *Information Theory, IRE Transactions on*, vol. 2, no. 3, pp. 113–124, 1956.

[66] R. Kirsch, "Computer interpretation of English text and picture patterns," *IEEE Transactions on Electronic Computers*, pp. 363–376, 1964.

[67] W. Watt, "Morphology of the Nevada cattle brands and their blazons" report 9050 (out of print) National Bureau of Standards," *Washington, DC*, 1966.

[68] G. Stiny and W. Mitchell, "The palladian grammar," *Environment and Planning B*, vol. 5, no. 1, pp. 5–18, 1978.

[69] L. Sass, "A palladian construction grammar-design reasoning with shape grammars and rapid prototyping," *Environment and Planning B: Planning and Design*, vol. 34, pp. 87–106, 2007.

[70] D. Y. Kwon, M. D. Gross, and E. Yi-Luen Do, "Archidna: An interactive system for creating 2d and 3d conceptual drawings in architectural design," *Computer-Aided Design*, vol. In Press, 2008.

[71] J. Kirsch and R. Kirsch, "The structure of paintings: formal grammar and design," *Environment and Planning B: Planning and Design*, vol. 13, no. 2, pp. 163–176, 1986.

[72] M. Whitelaw, *Metacreation: art and artificial life*. MIT Press, 2004.

[73] S. Todd and W. Latham, *Evolutionary Art and Computers*. Academic Press, Inc. Orlando, FL, USA, 1994.

[74] J. Romero and P. Machado, *The art of artificial evolution: A handbook on evolutionary art and music*. Springer-Verlag New York Inc, 2007.

[75] J. McCormack, J. Bird, A. Dorin, and A. Jonson, *Impossible Nature: The Art of John McCormack*. Australian Centre for the Moving Image, 2004.

[76] P. Bentley and D. Corne, eds., *Creative Evolutionary Systems*. Artificial Intelligence, San Francisco, CA: Morgan Kaufmann, 2001.

[77] M. King, "Programmed graphics in computer art and animation," *Leonardo*, vol. 28, no. 2, pp. 113–121, 1995.

[78] J. Yu, "Evolutionary design of 2d fractals and 3d plant structures for computer graphics," master's thesis, Department of Computer Science, University of Calgary, 2004.

[79] O. Deussen, P. Hanrahan, B. Lintermann, R. Mech, M. Pharr, and P. Prusinkiewicz, "Realistic modeling and rendering of plant ecosystems," in *SIGGRAPH 98, Computer Graphics, Annual Conference Series*, pp. 275–286, ACM SIGGRAPH, 1998.

[80] J. McCormack, "Art and the mirror of nature," *Digital Creativity*, vol. 14, pp. 3–22, 2003.

[81] R. Dawkins, *The Blind Watchmaker*. Harlow: Longman Scientific and Technical, 1987.

[82] K. Sims, "Artificial evolution for computer graphics," in *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, vol. 25(4), (New York), pp. 319–328, ACM Press, 1991.

[83] H. Kwong and C. Jacob, "Evolutionary exploration of dynamic swarm behaviour," in *Congress on Evolutionary Computation*, (Canberra, Australia), IEEE Press, 2003.

[84] H. Kwong, "Evolutionary design of implicit surfaces and swarm dynamics," Master's thesis, University of Calgary, Canada, 2003.

[85] A. C. Nardella, "Website of Anna C. Nardella." `http://www.annanardella.it/`, March 2009.

[86] C. Jacob, G. Hushlak, J. Boyd, P. Nuytten, M. Sayles, and M. Pilat, "Swarmart: Interactive art from swarm intelligence," *Leonardo*, vol. 40, no. 3, 2007.

[87] M. Whitelaw, *Breeding Aesthetic Objects: Art and Artificial Evolution*, pp. 129–145. San Francisco: Morgan Kaufmann, 2001.

[88] D. Thomas, "Aesthetic selection of developmental art forms," in *Artificial Life VIII, The 8th International Conference on the Simulation and Synthesis of Living Systems*, (Cambridge), pp. 157–163, MIT Press, 2002.

[89] M. Hemberg, "Genr8 - a design tool for surface generation," Master's thesis, MIT, June 2001.

[90] M. Hemberg, U.-M. O'Reilly, A. Menges, K. Jonas, M. da Costa Gonçalves, and S. R. Fuchs, *The Art of Artificial Life: A Handbook on Evolutionary Art and Music*, ch. Genr8: Architects' Experience with an Emergent Design Tool, pp. 167–188. Natural Computing Series, Springer, 2008.

[91] J. Rügemer, "From digital to real: Theoretical-digital architectural concepts and the realization of complex spatial forms," in *Education and research in computer aided architectural design in europe (eCAADe)*, (Warsaw, Poland), 2002.

[92] R. Saleri, "Urban and architectural 3D fast processing," in *9th International conference on generative art* (S. C., ed.), (Milano), 2006.

[93] J. Romero, P. Machado, A. Santos, and A. Cardoso, "On the Development of Critics in Evolutionary Computation Artists," *LECTURE NOTES IN COMPUTER SCIENCE*, pp. 559–569, 2003.

[94] P. Machado, J. Romero, and B. Manaris, "Experiments in computational aesthetics," in *The Art of Artificial Evolution* (P. Machado and J. Romero, eds.), Natural Computing Series, Springer, 2007.

[95] H.-P. Schwefel, *Numerische Optimierung von Computer–Modellen mittels der Evolutionsstrategie*, vol. 26 of *Interdisciplinary Systems Research*. Basle: Birkhäuser, 1977.

[96] J. Koza, *Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems*. Department of Computer Science, Stanford University, 1990.

[97] J. R. Koza, M. A. Keane, and M. J. Streeter, "Routine high-return human-competitive evolvable hardware," in *2004 NASA/DoD Conference on Evolvable Hardware*, pp. 3–17, 2004.

[98] M. McQuaid, *Envisioning Architecture: Drawings from the Museum of Modern Art*. New York, NY, USA: The Museum of Modern Art, 2002.

[99] H. Frichot, "On the death of architectural theory and other spectres," *Design Principles and Practices: An International Journal*, To appear in 2009.

[100] M. Speaks, "Which way avant-garde?," *Assemblage*, no. 41, p. 78, 2000.

[101] R. Somol and S. Whiting, "Notes around the doppler effect and other moods of modernism," *Perspecta*, vol. 33, pp. 72–77, 2002.

[102] W. Knoll and M. Hechinger, *Architektur-Modelle: Anregungen zu Ihrem Bau*. Munich, Germany: Deutsche Verlangs-Anstalt, 2006.

[103] S. Bergen, S. Bolton, and J. L. Fridley, "Design principles for ecological engineering," *Ecological Engineering*, vol. 18, no. 2, pp. 201–210, 2001.

[104] S. Van der Ryn and S. Cowan, *Ecological Design*. Island Press, 2007.

[105] D. Pearson, *New Organic Architecture: The Breaking Wave*. University of California Press, 2001.

[106] A. J. Anselm, "Developing designs in balance with nature," in *Eco-Architecture: Harmonisation between Architecture and Nature* (G. Broadbent and C. Brebbia, eds.), Trans-

actions on the Built Environment, pp. 195–204, Wessex Institute of Technology, WIT Press, 2006.

[107] K. Gowri, "Green building rating systems: An overview.," *ASHRAE Journal*, vol. 46, no. 11, pp. 56–60, 2004.

[108] E. Coen, *The art of genes*. Oxford University Press New York, 1999.

[109] W. Banzhaf, J. Koza, C. Ryan, L. Spector, and C. Jacob, "Genetic programming," *IEEE Intelligent Systems and Their Applications*, vol. 15, no. 3, pp. 74–84, 2000.

[110] P. E. Griffiths and R. D. Gray, "Developmental systems and evolutionary explanation," *The Journal of Philosophy*, vol. 91, no. 6, pp. 277–304, 1994.

[111] S. Kauffman, *At Home in the Universe: The Search for the Laws of Self-Organization and Complexity*. Oxford University Press, 1995.

[112] L. Margulis and S. Dorion, *What is Life?* University of California Press, 2000.

[113] R. Dawkins, "Selfish genes and selfish memes," *The Mind's I: Fantasies and Reflections on Self and Soul*, pp. 124–144, 1981.

[114] M. Best, "How Culture Can Guide Evolution: An Inquiry into Gene/Meme Enhancement and Oppostion," *Adaptive Behavior*, vol. 7, no. 3/4, pp. 289–306, 1999.

[115] G. Witzany, "Natural history of life: History of communication logics and dynamics," *SEED Journal*, vol. 5, no. 1, pp. 27–55, 2005.

[116] R. L. Chisholm and R. A. Firtel, "Insights into morphogenesis from a simple developmental system," *Nat Rev Mol Cell Biol*, vol. 5, no. 7, pp. 531–541, 2004.

[117] R. Rojas, *Neural Networks - A Systematic Introduction*. Berlin, New York: Springer-Verlag, 1996.

[118] T. Mitchell, *Introduction to Machine Learning*. Boston, Massachusettes: McGraw Hill, 1997.

[119] J. C. Astor and C. Adami, "A developmental model for the evolution of artificial neural networks," *Artif. Life*, vol. 6, no. 3, pp. 189–218, 2000.

[120] K. Sims, "Evolving 3D morphology and behaviour by competition," in *Artificial Life IV Proceedings* (R. Brooks and P. Maes, eds.), (MIT, Cambridge, MA, USA), pp. 28–39, MIT Press, 6-8July 1994.

[121] M. Pilat, *Morphid Academy: A Virtual Laboratory for Evolution of Form and Function*. PhD thesis, University of Calgary, 2009.

[122] F. Dellaert and R. Beer, "A developmental model for the evolution of complete autonomous agents," in *SAB '96*, 1996.

[123] G. S. Hornby and J. B. Pollack, "Creating high-level components with a generative representation for body-brain evolution," *Artif. Life*, vol. 8, no. 3, pp. 223–246, 2002.

[124] J. M. Denu, J. A. Stuckey, M. A. Saper, and J. E. Dixon, "Form and function in protein dephosphorylation," *Cell*, vol. 87, pp. 361–364, 11 1996/11/1.

[125] P. Watson, "Function follows form: generation of intracellular signals by cell deformation," *The FASEB Journal*, vol. 5, no. 7, pp. 2013–2019, 1991.

[126] D. Ingber, "The architecture of life," *Scientific American*, vol. 278, no. 1, pp. 48–57, 1998.

[127] C. Paul, H. Lipson, and F. J. V. Cuevas, "Evolutionary form-finding of tensegrity structures," in *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, (New York, NY, USA), pp. 3–10, ACM Press, 2005.

[128] N. Khemka, C. Jacob, and G. Cole, "Making soccer kicks better: a study in particle swarm optimization," in *GECCO '05: Proceedings of the 2005 workshops on Genetic and evolutionary computation*, (New York, NY, USA), pp. 382–385, ACM, 2005.

[129] C. Smith, *On Vertex-Vertex Systems and Their Use in Geometric and Biological Modelling*. PhD thesis, University of Calgary, 2006.

[130] E. Coen, A. Rolland-Lagan, M. Matthews, J. Bangham, and P. Prusinkiewicz, "The genetics of geometry," *Proceedings of the National Academy of Sciences*, vol. 101, no. 14, pp. 4728–4735, 2004.

[131] A. Lisi, "An exceptionally simple theory of everything," *arxiv*, vol. 711, no. 6, 2007.

[132] S. Lloyd, *Programming the universe: a quantum computer scientist takes on the cosmos*. Vintage Books, 2007.

[133] M. Gardner, "Mathematical games: The fantastic combinations of john conway's new solitaire game "life"," *Scientific American*, vol. 223, pp. 120–123, October 1970.

[134] S. Wolfram, *A new kind of science*. Champaign, Ilinois, US, United States: Wolfram Media Inc., 2002.

[135] S. Wolfram, "Cellular automata as models of complexity," *Nature*, vol. 311, pp. 419–424, October 1984.

[136] S. Wolfram, "Stephen Wolfram: A New Kind of Science – Reference Material." `http://www.wolframscience.com/reference/`, March 2009.

[137] R. Bagley and J. Farmer, "Spontaneous emergence of a metabolism," in *Artificial Life II*, (Cambridge, MA, USA), MIT Press, 1990.

[138] P. Schuster, "How does complexity arise in evolution," *Complex.*, vol. 2, no. 1, pp. 22–30, 1996.

[139] J. Holland, *Hidden order: How adaptation builds complexity*. Addison Wesley Publishing Company, 1996.

[140] J. Bader, "The drosophila protein interaction network may be neither power-law nor scale-free," *Power Laws, Scale-Free Networks and Genome Biology*, pp. 53–64, 2006.

[141] J. Buhl, J. Gautrais, R. Solé, P. Kuntz, S. Valverde, J. Deneubourg, and G. Theraulaz, "Efficiency and robustness in ant networks of galleries," *The European Physical Journal B-Condensed Matter*, vol. 42, no. 1, pp. 123–129, 2004.

[142] C. A. Hidalgo and A.-L. Barabasi, "Scale-free networks," *Scholarpedia: The free peer reviewed encyclopedia*, 2006.

[143] J. Travers and S. Milgram, "An experimental study of the small world problem," *Sociometry*, vol. 32, no. 4, pp. 425–443, 1969.

[144] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks.," *Nature*, vol. 393, pp. 440–442, June 1998.

[145] C. Moore and M. E. J. Newman, "Epidemics and percolation in small-world networks," *Physical Review E*, vol. 61, p. 5678, 2000.

[146] V. Latora and M. Marchiori, "Economic small-world behavior in weighted networks," 2002.

[147] R. Albert, "Boolean modeling of genetic regulatory networks," *Complex Networks*, pp. 459–481, 2004.

[148] R. Serra, M. Villani, and L. Agostini, "On the dynamics of scale-free boolean networks," *Neural Nets*, pp. 43–49, 2003.

[149] M. Afsharchi, B. Far, and J. Denzinger, "Ontology-guided learning to improve communication between groups of agents," in *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pp. 923–930, ACM New York, NY, USA, 2006.

[150] S. Johnson, *Emergence: The Connected Lives of Ants, Brains, Cities, and Software*. New York: Scribner, 2001.

[151] J. H. Holland, *Emergence: From Chaos to Order*. New York: Oxford University Press, 1998.

[152] G. S. Hornby, "Generative representations for evolving families of designs," in *GECCO* (E. Cantú-Paz, J. A. Foster, K. Deb, L. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. K. Standish, G. Kendall, S. W. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. A. Dowsland, N. Jonoska, and J. F. Miller, eds.), vol. 2724 of *Lecture Notes in Computer Science*, pp. 1678–1689, Springer, 2003.

[153] G. S. Hornby, "Measuring complexity by measuring structure and organization," in *2007 IEEE Congress on Evolutionary Computation* (D. Srinivasan and L. Wang, eds.), (Singapore), pp. 2017–2024, IEEE Press, 2007.

[154] M. Wooldridge, *An Introduction to MultiAgent Systems*. Chichester, England: John Wiley and Sons, February 2002.

[155] C. W. Reynolds, "Website of Craig W. Reynolds." `http://www.red3d.com/cwr/`, March 2009.

[156] I. Burleigh, "Vigo::3d: A framework for simulating and visualizing of three-dimensional scenes.." `http://vigo.sourceforge.net/docs/`, October 2008.

[157] Aesthetics and Computation Group at the MIT Media Lab, "Processing language and programming environment." http://processing.org/, October 2008.

[158] U. Wilensky and the CCL at Northwestern University, "Netlogo: A cross-platform multi-agent programmable modeling environment." http://ccl.northwestern.edu/netlogo/, October 2008.

[159] T. Vicsek, A. Czirok, E. Ben-Jacob, I. Cohen, and O. Shochet, "Novel type of phase transition in a system of self-driven particles," *J. Exp. Mar. Biol. Ecol Phys Rev Lett*, vol. 75, p. 1226, 1989.

[160] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet, "Novel Type of Phase Transition in a System of Self-Driven Particles," *Physical Review Letters*, vol. 75, no. 6, pp. 1226–1229, 1995.

[161] A. Czirok and T. Vicsek, "Collective behavior of interacting self-propelled particles," *Arxiv preprint cond-mat/0611742*, 2006.

[162] I. Derényi and T. Vicsek, "Cooperative transport of Brownian particles," *J. Phys. I (France) Phys Rev Lett*, vol. 75, p. 374, 1994.

[163] C. Huepe and M. Aldana, "New tools for characterizing swarming systems: A comparison of minimal models," *Physica A: Statistical Mechanics and its Applications*, vol. 387, no. 12, pp. 2809 – 2822, 2008.

[164] K. Kaneko, "Overview of coupled map lattices," *Chaos*, vol. 2, p. 279, 07 1992.

[165] K. Kaneko, "Spatiotemporal chaos is one-and two-dimensional coupled map lattices," in *CNLS conference on advances in fluid turbulence*, vol. 16, 1988.

[166] J. Jost and M. Joy, "Spectral properties and synchronization in coupled map lattices," *Science Phys Rev E*, vol. 65, p. 016201, 1999.

[167] A. Lemaître and H. Chaté, "Phase Ordering and Onset of Collective Behavior in Chaotic Coupled Map Lattices," *Physical Review Letters*, vol. 82, no. 6, pp. 1140–1143, 1999.

[168] H. Tanner, A. Jadbabaie, and G. Pappas, "Flocking in fixed and switching networks," in *IEEE Conference on Decision and Control*, vol. 1, p. 2, 2005.

[169] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, 2006.

[170] L. Moura, "Website of leonel moura." `http://www.lxxl.pt`, December 2007.

[171] T. Blackwell, "Swarming and music," *Evolutionary Computer Music*, pp. 194–217, 2007.

[172] N. Khemka, S. Novakowski, G. Hushlak, and C. Jacob, "Evolutionary design of dynamic SwarmScapes," in *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pp. 827–834, ACM New York, NY, USA, 2008.

[173] L. Spector, J. Klein, C. Perry, and M. Feinstein, "Emergence of collective behavior in evolving populations of flying agents," *Genetic Programming and Evolvable Machines*, vol. 6, no. 1, pp. 111–125, 2005.

[174] W. Wright, R. Smith, M. Danek, and P. Greenway, "A generalisable measure of self-organisation and emergence," *Lecture notes in computer science*, pp. 857–864, 2001.

[175] M. Pilat, "Wasp-inspired construction algortihms," tech. rep., University of Calgary, 2004.

[176] S. von Mammen, C. Jacob, and G. Kókai, "Evolving swarms that build 3d structures," in *CEC 2005, IEEE Congress on Evolutionary Computation*, (Edinburgh, UK), IEEE Press, 2005.

[177] S. von Mammen, *Evolving artificial constructive swarms - Experimental models and methodologies*. Saarbrücken, Germany: VDM-Verlag, 2008.

[178] Y. Zeng, P. B. Dennis, and C. H. Jorge, "Multiagent based construction for human-like architecture," in *AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, (New York, NY, USA), pp. 1–3, ACM, 2007.

[179] Y. Zeng, C. H. Jorge, and P. B. Dennis, "Swarmarchitect: a swarm framework for collaborative construction," in *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, (New York, NY, USA), pp. 186–186, ACM, 2007.

[180] C. W. Reynolds, "Flocks, herds, and schools: A distributed behavioral model," *Computer Graphics*, vol. 21, no. 4, pp. 25–34, 1987.

[181] G. W. Litman, J. P. Cannon, and L. J. Dishaw, "Reconstructing immune phylogeny: new perspectives," *Nat Rev Immunol*, vol. 5, no. 11, pp. 866–879, 2005.

[182] M. Oilek and P. Klein, "Stochastic model of the immune response," *Modelling and Optimization of Complex System*, pp. 15–25, 1979.

[183] F. Neelamkavil, *Computer Simulation and Modelling*. John Wiley and Sons, 1994.

[184] S. von Mammen and C. Jacob, "Swarming for games: Immersion in complex systems," in *Applications of Evolutionary Computing, Proceedings of EvoWorkshops 2009*, Lecture Notes in Computer Science, (Berlin-Heidelberg), Springer-Verlag, 2009.

[185] X. Yan, P. Yu, and J. Han, "Graph indexing: a frequent structure-based approach," in *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pp. 335–346, ACM New York, NY, USA, 2004.

[186] H. He and A. Singh, "Closure-tree: An index structure for graph queries," in *Proceedings of the 22nd International Conference on Data Engineering*, p. 38, IEEE Computer Society Washington, DC, USA, 2006.

[187] A. Eldridge, A. Dorin, and J. McCormack, "Manipulating artificial ecosystems," *Applications of Evolutionary Computing*, pp. 392–401, 2008.

[188] J. McCormack and O. Bown, "Life's what you make: Niche construction and evolutionary art," *Applications of Evolutionary Computing*, pp. 528–537, 2009///.

[189] S. Silva and S. Dignum, "Extending operator equalisation: Fitness based self adaptive length distribution for bloat free gp," *Genetic Programming*, pp. 159–170, 2009///.

[190] J. Giavitto and O. Michel, "The topological structures of membrane computing," *Fundamenta Informaticae*, vol. 49, no. 1, pp. 123–145, 2002.

[191] G. Paun and G. Rozenberg, "A guide to membrane computing," *Theoretical Computer Science*, vol. 287, pp. 73–100, 9 2002/9/25.

[192] S. von Mammen, G. Hushlak, S. Novakowski, and C. Jacob, "Evolutionary swarm design," *Design Principles and Practices: An International Journal*, In press. 2009.

[193] C. Jacob and I. Burleigh, "Biomolecular swarms: An agent-based model of the lactose operon," *Natural Computing*, vol. 3, pp. 361–376, December 2004.

[194] C. Jacob, A. Barbasiewicz, and G. Tsui, "Swarms and genes: Exploring λ-switch gene regulation through swarm intelligence," in *CEC 2006, IEEE Congress on Evolutionary Computation*, 2006.

[195] C. Jacob, S. Steil, and K. Bergmann, "The swarming body: Simulating the decentralized defenses of immunity," in *Artificial Immune Systems, ICARIS 2006, 5th International Conference*, (Oeiras, Portugal), Springer, September 2006.

[196] J. E. Boyd, G. Hushlak, and C. Jacob, "Swarmart: Interactive art from swarm intelligence," in *Proceedings of the 12th annual ACM international conference on Multimedia*, (New York, NY, USA), pp. 628–635, ACM Press, 2004.