Chapter 18
Swarm-Based Computational Development

Sebastian von Mammen, David Phillips, Timothy Davison, Heather Jamniczky, Benedikt Hallgrímsson and Christian Jacob

Abstract Swarms are a metaphor for complex dynamic systems. In swarms, large numbers of individuals locally interact and form non-linear, dynamic interaction networks. Ants, wasps and termites, for instance, are natural swarms whose individual and group behaviors have been evolving over millions of years. In their intricate nest constructions, the emergent effectiveness of their behaviors becomes apparent. Swarm-based computational simulations capture the corresponding principles of agent-based, decentralized, self-organizing models. In this work, we present ideas around swarm-based developmental systems, in particular *swarm grammars*, a swarm-based generative representation, and our efforts towards the unification of this methodology and the improvement of its accessibility.

18.1 Introduction

Arithmetic operations drive computational processes by updating existing variables or by infering new ones. The selection of operands determines a topology of dependencies among data which may change over the course of a computational process. In particular, intermediate computational results may control the flow of directives,

S. von Mammen (✉)
Departments of Computer Science,
University of Calgary, Calgary, Canada
e-mail: s.vonmammen@ucalgary.ca

D. Phillips · T. Davison · H. Jamniczky · B. Hallgrímsson
Departments of Cell Biology,
University of Calgary, Calgary, Canada

C. Jacob
Departments of Computer Science and Biochemistry and Molecular Biology,
University of Calgary, Calgary, Canada
e-mail: cjacob@ucalgary.ca

R. Doursat et al. (eds.), *Morphogenetic Engineering*, Understanding Complex Systems,
DOI: 10.1007/978-3-642-33902-8_18,

their sources and their targets. Ultimately, the purpose of a computational process is to create or change information in accordance with goals such as data storage and retrieval, communication, content creation, or data validation, visualization and prediction (simulation).

Computational *representations* are abstraction layers which connect particular models from a scientific domain (domain models) with corresponding models for a provided simulation platform (platform models) [46]. Similar to *knowledge representations* [50], special representations have been studied and designed in the context of simulating developmental processes such as the formation of molecular structures [1], growth and proliferation of cell populations, and structural developments at the organismal level [48].

Not only do these respective computational developmental representations serve different modeling domains but they also rely on various mechanisms of abstraction. L-systems [48], for instance, emphasize the formation of structure based on the generation of symbolic sequences by means of grammatical substitution [7]. Cellular automata (CAs), on the other hand, which are also considered one of the first developmental representations, focus on pattern generation through state changes [67]. Although mitosis and cell differentiation provide a scientifically adequate discretization that bridges the gap from domain model to platform model, CAs and L-systems primarily target development at the cellular level.

Focus areas of developmental models have been simulations of the life cycle of cells as well as molecular and intercellular communication—touched upon by CAs and further explored as random boolean networks (RBNs) [34]. Spatial reconfiguration of cell colonies, e.g., through polarization and migration, and thereby changes in the interaction topologies, also play significant roles in developmental systems [51].

In this chapter, we present our work on *swarm grammars* (SGs), a developmental representation that we have introduced to explicitly combine the ideas of established developmental representations with those of artificial swarm systems. In particular, production rules drive the life cycle of agents (representative of molecules and cells), while the agents' reactivity and motility continuously change the interaction topology of the system. In the next section, we briefly outline work that relates to SGs. Section 18.3 presents various swarm grammar representations that we have designed over the years, as well as means to breed SG configurations through evolutionary computation [32, 60, 62, 64]. In this context, we also present SG examples in the domains of art, architecture and biology [61, 65, 66]. Section 18.4 summarizes and concludes our work.

18.2 Related Work

In the following, we want to build a terminological hierarchy based on *patterns*, *structures*, and *morphologies*. A pattern is commonly defined as "an arrangement or sequence regularly found in comparable objects or events". A structure is "the arrangement of and relations between the parts or elements of something complex".

A morphology "[...] deals with the form of living organisms, and with relationships between their structures" [41]. Morphologies describe the structures of organisms, whereas the recognition of their structural patterns reveals insight into the complex arrangements of the parts of an organism. Consequently, the borders blur between patterns and complex structures when facing the challenges of morphological engineering. Reductionist [23] and quantitative [18] analyses of morphological processes necessitate identification, measurement [20] and simulation of complex, emergent processes [46], and integrating, multi-scale models [68]. Exploring these scientific paths is exciting and important. In this chapter, however, we present our findings that are mainly concerned with swarm grammars as a unified swarm-based developmental representation.

18.2.1 Complex Patterns Through State Changes

In cellular automata (CAs) lattice grids are populated with cells that change their states—frequently represented as a binary digit—in accordance with their neighbors [67]. As underlined by Giavitto et al.'s categorization [16], CAs are dynamic with respect to their states, but not in regard to their interaction topologies. Thus, the development in CAs is limited to state-based pattern formation. These patterns, however, may be seen as structure formation nevertheless. Wolfram introduced four classes of complexity for patterns generated by the state evolution of one-dimensional CAs [69]: Those that converge (1) to a homogeneous state (corresponding to limit points), (2) to a heterogeneous state (corresponding to limit cycles), those that exhibit (3) chaotic behavior (corresponding to chaotic attractors), and those that are (4) self-organizing, reaching attractors of arbitrary complexity from random initial conditions.

18.2.2 Complexity Measures

Several aspects pose starting points to reveal the complexity of a (computational) model, or the lack thereof. Shedding light on the formation and evolution of high-order life forms, Schuster summarizes various approaches to measure complexity [52]: (1) Ecological diversity can lead to systems occupying niches and yield involved food webs. (2) Construction processes can add to the complexity of a system by providing additional functionality such as providing shelter. (3) Formally, logical depth can also measure a system's complexity. Schuster relates systemic hierarchies (from genes over cells to organisms) to logical depth and emphasizes its relevance for biological systems. Hornby aims at the very same idea, introducing the scalable metrics of modularity, reuse and hierarchy (MR&H), which he applies to measure structure and organization [20]. In the given context, scalability implies that the complexity of a system increases with its size. In a series of experiments, Hornby was

able to show that multiplication of the MR&H metrics and normalization by either the design size or by the algorithmic information content (AIC), which accounts for the shortest program to produce a given outcome, yielded the desired scaling effect in complexity.[1]

18.2.3 From Life-Cycles to Structure

Although we have introduced the notion of categories of complexity in the context of CAs (Sect. 18.2.1), L-systems, too, can be subjected to structural measures as described in Sect. 18.2.2). Lindenmayer and Prusinkiewicz designed L-systems in order to retrace the growth of bacterial colonies and plants [48]. L-systems are a formal paradigm that combines the productivity of formal grammars with geometrical information to direct and translate simulated development into three dimensions. In particular, symbols that encode a geometrical command such as *L* (left), *R* (right) or *F* (forward) are substituted in parallel in accordance with a set of production rules. This is supposed to retrace the developmental stages of a naturally growing structure. Special characters such as brackets, [and], which are part of the substitution strings, introduce compartmentalization, i.e., hierarchical information, into the structural outcome. Generally speaking, L-systems loose the appeal of universality that can be claimed for CAs by introducing specialized operators that conduct structural development in an iterative fashion.

At the same time, these geometrical directives render L-systems convenient for describing structural development of natural systems such as plants [47, 70], including simulated plants that interact with their environment [12, 42, 43]. Original work in genetic programming of L-systems [24–26] has led to several platforms for L-system evolution [27, 44, 45] and the breeding of virtual plants in a coevolutionary scenario, which even displays competitive arms-race situations [14]. Beyond plants, L-systems have also been used to evolve virtual creatures and their control networks [21, 22] as well as for the reconstruction of retina and blood vessel structures [35, 36].

The appeal of CAs is that the interaction topologies remain fixed, while patterns develop based on state changes over time. In L-systems, the substitution of existing symbols effectively results in cell differentiation (state changes), the creation of new or the deletion of existing symbols. Thus the neighborhood topology can be altered as well as the next production step in the case of context-sensitive L-systems. However, changes in the interaction topology in L-systems are limited to the symbols' immediate neighbors. When modeling molecular diffusion or cell locomotion and migration in an agent-based manner, interaction topologies undergo great dynamics

[1] R_a, the average reuse of symbols during program execution works well as a structural measure when normalized against the design size, whereas R_m, the average reuse of modules, yields a scalable measure when divided by the system's algorithmic information content [20].

(e.g., [51]). Giovatti et al. termed D^2S such systems in which both the states as well as the topologies are dynamic [16].

18.2.4 Breeding Solutions

The great degree of freedom with a D^2S system brings about the challenge of a largely extended configuration space. Evolutionary algorithms are a means to find sets of diverse, good solutions in such large search spaces. We applied genetic programming techniques to breed swarm grammar systems interactively [32] through the *EVOLVICA* genetic programming framework [27]. We bred swarm grammars like a gardener in a 3D, immersive environment [62]—watering, weeding and recombining individual specimens that grew in a shared environment. Most recently, we let swarm grammars evolve to generate diverse and interesting architectural models [60]. We describe these three evolutionary approaches in detail as part of the following section, which presents different extensions of swarm grammars.

18.3 Swarm Grammars

Our first swarm grammar systems were composed of two parts: (1) a set of *rewrite rules*, which determined the composition of agent types over time, and (2) a set of *agent specifications*, which defined agent-type specific parameters that governed the agents' interactions [32]. Next, we assigned the required genotypical information and the rewrite rules to individual agents, which allowed for co-existing and co-evolving swarm grammars [62]. At that point, we identified the distinction between agent behaviors and rewrite rules as an artificially created artifact which we had to overcome [60]. As a result, the individuals' rewrite rules were extended and turned into general agent rules [10, 11], including the special ability to create new agents or construction elements and to remove existing ones. In analogy to biochemical processes of secretion and diffusion [68], we refer to these abilities as *metabolic* operations. Lastly, in order to make swarm-based modeling accessible to non-computer scientists, we have been pushing toward a standardized swarm-based modeling and simulation framework [64]. In the latter representation, the relationships among swarm individuals were emphasized and the swarm agents' behavioral rules were streamlined and expressed in graphical notation. In this section, we present these different stages of swarm grammars and illustrate their respective features. A brief overview of these stages is shown in Table 18.1.

Table 18.1 Four evolutionary stages of swarm grammars

Representation	Motivation	Example	Section
Basic swarm grammar	Swarm dynamics + growth	Agent-agent and agent-environment interactions, artistic exploration through interactive evolution	18.3.1
Decentralized SG	Individual behaviors	Artistic exploration through interactive evolution	18.3.2
Decentralized, rule-based SG	Event-based interactions	Breeding architecture through automatic evolution	18.3.3
Swarm graph grammar	Improve accessibility and standardize simulation	Simulation of biological developmental processes	18.3.4

18.3.1 Swarm Grammars with Centralized Population Control

A basic swarm grammar system $SG = (SL, \Delta)$ consists of a rewrite system $SL = (\alpha, P)$ and a set of agent specifications $\Delta = \{\Delta_{a_1}, \Delta_{a_2}, ..., \Delta_{a_n}\}$ for n types of agents a_i. The rewrite system SL is a probabilistic L-system with axiom α and production rules P, as described in [48] and [27]. In the simplest form of context-free 0L-systems, each rule has the form:

$$p \xrightarrow{\theta} s,$$

where $p \in \Omega$ is a single symbol over an alphabet Ω, and $s \in \Omega^*$ is either the empty symbol (λ) or a word over Ω. The replacement rule is applied with probability θ. Each agent a_i is characterized by a set of attributes, Δ_{a_i}, which can include its geometrical shape, color, mass, vision range, radius of perception and other parameters that determine its overall dynamics and interaction behavior.

18.3.1.1 The Swarm Agents' Interactions

Figure 18.1 depicts a common view of a swarm agent in 3D space. We configure the local flight behavior of an agent according to a simple "boids" model as it comprises the general ideas of local and global attracting and repelling forces [49]. More specifically, at each simulation step, an agent's acceleration vector $\mathbf{a}$ is set to a weighted sum $\sum_{i=0}^{4} c_i \mathbf{v}_i$, with $c_i \in [0, 1]$ being the weights for normalized vectors $\mathbf{v}_0$ to $\mathbf{v}_4$ that result from the computation of separation, cohesion and alignment urges among local agents, as well as from the individuals' drive towards a global target and the consideration of some noise. The weights c_i are part of the individuals' genotypes as they determine their flight behaviors.

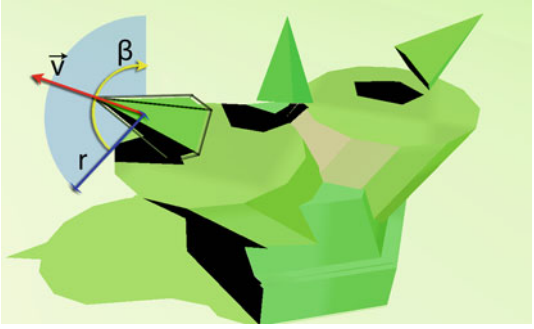

Fig. 18.1 The swarm agents are typically represented as pyramidal cones oriented towards their velocity. An agent's field of perception is determined by a radius r and an angle β

As soon as it has run out of energy, an agent stops acting and is not considered by the SL-system rules any longer. Energy levels are inherited through replication. The energy level also influences certain properties of the built 3D structures such as, for example, their size.

Several values characterize the construction elements or building blocks that are placed in space by the swarm agent after it has flown for a certain number of iterations I_d. The shorter these intervals are, the smoother the appearance of the emerging construction. The color and numbers of edges define the design of the cylindrical shapes.

For example, a swarm grammar $SG_a = (SL_a, \Delta_a)$ with

$$SL_a = (\alpha = A, P = \{A \rightarrow BBB, B \rightarrow A\}), \tag{18.1}$$

$$\Delta_a = \{\Delta_A, \Delta_B\} \tag{18.2}$$

will generate a sequence of *swarm composition strings* A, BBB, AAA, $BBBBBBBBB$, etc. At each iteration step, either each type-A agent is replicated into three B agents, or agents change from type B to type A. If A agents have no separation urge ($c_1 = 0$), and B-type agents do separate ($c_1 = 1.0$), the generated swarm of agents creates a tree-like structure as in Fig. 18.2a. Note that here and in the following examples we assume $\theta = 1$, that is a matching rule is always applied.

In particular, Fig. 18.2a displays 243 agents—which are visualized as pyramidal shapes at the branch tips. Both occurring agent types A and B have an upward urge. Since B-agents repel each other, a bushy crown emerges. Figure 18.2b shows a similar set of swarm grammar agents that are forced to climb up a wall, which they cannot penetrate. Once the agents reach to the top of the wall, they are drawn towards a fixed point above and behind the wall. The small flock of agents is visible just ahead of the top branches. In Fig. 18.2c, agents are attracted towards a rotating "sun" object, which makes them follow a spiral during their upward path. The structure on the right is constructed by a single agent, whereas the left structure involves 20 agents repelling each other.

Each step applying the production rules (in parallel) represents a decision point for all agents within the system. Contrary to L-systems [48], where only a single "turtle"

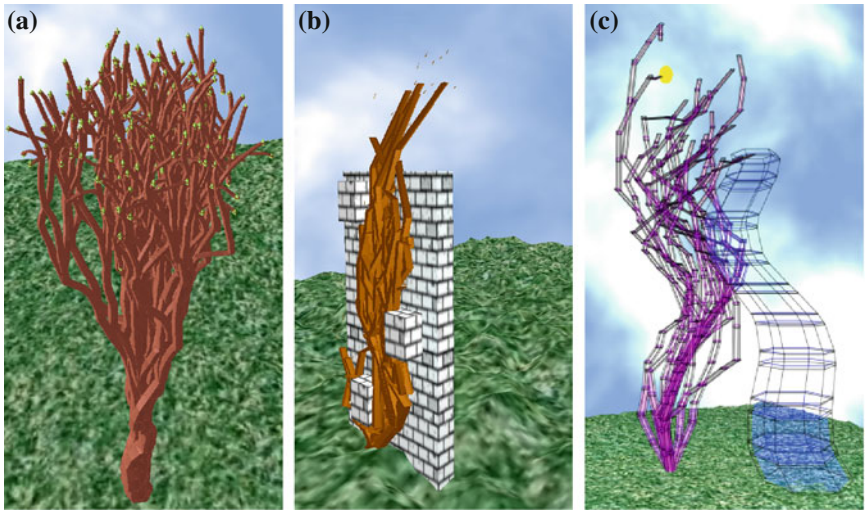

Fig. 18.2 Swarm grammar agents interacting with their environment and their corresponding swarm rewrite systems

is used to interpret a string, we employ a swarm of interacting agents. We do not need to add navigational commands for the turtles within the grammar strings, because the swarm agents navigate by themselves, determined by the agent specifications as part of the SG system. More detailed examples of swarm grammar rewriting that demonstrate further applicative aspects are given in [32].

18.3.1.2 Interactive Exploration of Swarm Grammar Spaces

Combining swarm systems with evolutionary computing has to our knowledge only been considered in the context of particle swarm optimization (PSO, e.g., [53, 54]) and in swarm-based music generating systems (e.g., [4, 33]). Emergence of collective behavior has been investigated for agents within a 3D, static world [57], but this did not involve interactive evolution. Our *Genetic Swarm Grammar Programming* (GSGP) approach incorporates both interactive, user-guided evolution and the use of emergent properties from interactions among a large number of agents.

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. This follows our framework for evolutionary programming, *EVOLVICA* [27], where all rewrite rules and agent parameters are encoded as symbolic expressions [39]. 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 and up to three different types of swarm individuals per SG genotype.

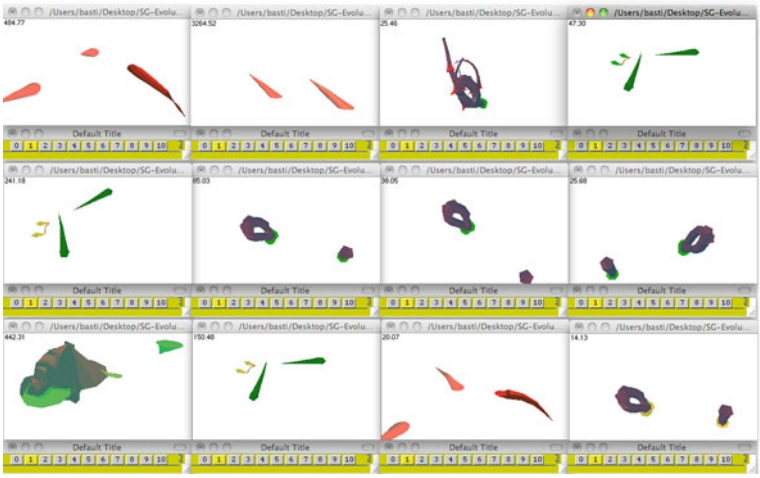

Fig. 18.3 Screenshot of the *Inspirica* GUI that enables interactive evolution based on Mathematica in combination with its genetic programming extension *EVOLVICA*

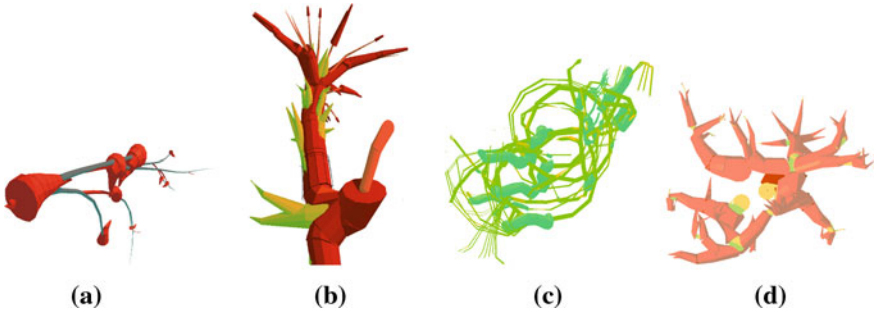

Fig. 18.4 Examples of evolved swarm grammar phenotypes: **a** Pointy yet smooth nodes connect by long thin branches. **b** A flower-like structure created by a single mutation. **c** Spinning and whirling groups of swarm agents create a woven 3D pattern. **d** An organismic structure with growing tips

In our evolutionary swarm grammar experiments standard GP tree-crossover and subtree mutations are the only applied genetic operators [27]. We use an extension of *Inspirica* [39], one of our interactive evolutionary design tools, to explore the potential of the described swarm grammar systems.

Figure 18.3 displays a screenshot of the *Inspirica* user interface that helps to interactively evolve swarm grammars. All windows display the construction process as it occurs. All designs are true objects in 3D space, hence can be rotated, zoomed and inspected in various ways. After assessment of the presented (twelve) structures, the swarm designer assigns fitness values between 0 and 10 to each solution, and proceeds to the next generation. By means of this approach, one can easily—within only a few generations—create structures as illustrated in Fig. 18.4.

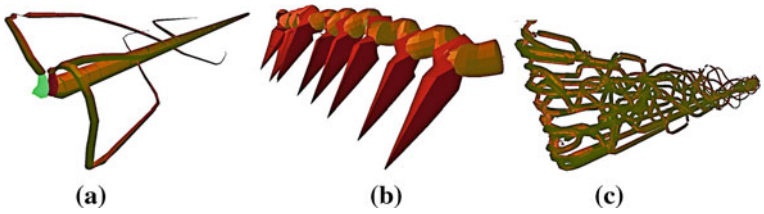

Fig. 18.5 Examples of the impact of interactive breeding: **a** and **b** show two phenotypes that were interbred and whose offspring **c** successfully acquired characteristics of both parent structures. Investigation of the genotypes confirms that a recombinational transfer of a recursively applicable grammatical rule leads to the complex mesh structure in **c**

The impact of the inter-breeding process, accomplished through crossovers of the SL-system grammars and their associated agent parameters, is illustrated in Fig. 18.5. The replication of a swarm agent (as determined by the grammar) and its associated constructions cease as soon as it runs out of energy. Since the energy level of an agent is linked to the radius of the built cylindrical shape, the structures tend to look like naturally grown, with smaller tips at the ends. If the agents' energy loss I_e is very low, however, the radii of the cylindrical objects hardly decrease. Since the energy level is one possible termination criterion, constructions that keep their radii approximately constant often appear in tandem with vivid growth. These effects are illustrated in Figs. 18.2 and 18.4.

18.3.2 SG Individuals with Complete Genotypes

In the previous examples (Sect. 18.3.1.2) swarm grammars were simulated within separate spaces. In an immersive design ecology, however, one could grow large numbers of swarm grammar structures in a co-existing and co-evolutionary fashion. The encountered phenotypes can then result from massive interactions among heterogeneous swarms. For this to happen, each swarm agent has to carry the complete genetic information of a swarm grammar $SG = (SL, \Delta)$, which also allows for real-time mutations and crossbreeding of specimens in the virtual environment. This extension of SGs is not unlike in multicellular organisms, where the complete genetic information is passed from parent to daughter cells, and where the differentiation of a cell is performed through reading and expressing specific genetic information.

18.3.2.1 Spatial Breeding Operators

Our immersive user interface integrates two aspects: visual representation and intuitive manipulation by an external breeder or designer. The latter mechanism is realized by the already mentioned spatial breeding operators, or *breeder volumes*. Figure 18.6

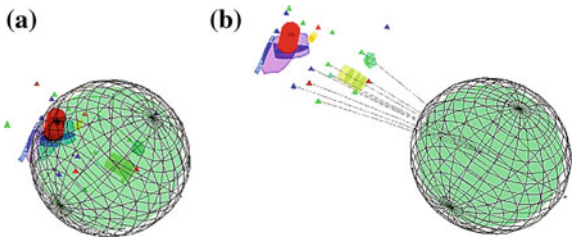

Fig. 18.6 **a** By means of volumetric tools the immersed breeder can manually select and tinker with the present specimens. **b** Visual cues such as connecting specimens to breeder volumes via *dashed lines* allow the breeder to keep track of sets of selected agents

shows a breeder volume that encloses several swarm grammar agents. 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 in a garden, fitness evaluations are only given implicitly by providing more energy to selected groups of agents. In order to facilitate selective evolutionary intervention, breeder volumes can be placed at fixed positions to perform operations on temporary visitors with predefined frequencies. Additional visuals allow to keep track of previous agent selections. Figure 18.6b depicts how previously enclosed agents remain associated with the appropriate breeder volume. This relationship is visualized by the connecting lines.

The visualization interface enables moving, rotating, and zooming the camera, or saving and restoring specific views and scenario settings (Fig. 18.7). Most of these procedures are already incorporated into the agent software environment *BREVE*, which we use as our display and simulation engine [57]. In addition to aspects of visualization, the supervising breeder is equipped with tools to select, group, copy, and move swarm grammar agents. This enables the breeder (designer) 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.

18.3.2.2 The Swarm Grammar Gardener

Figure 18.7 illustrates how a breeder can influence the emerging building processes within a simple ecology of swarms. In Fig. 18.7a, two swarm agents have built a cylindrical structure with a small 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. 18.7b), 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 farther to the right (Fig. 18.7c).

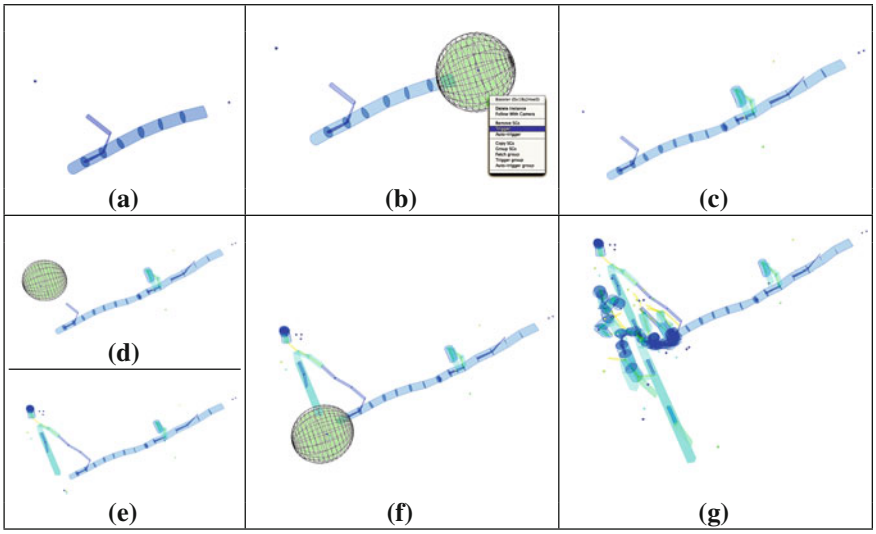

Fig. 18.7 Illustration of interactive manipulation of swarm grammar agents by an external breeder. **a** Two agents create an initial structure. **b** A breeder sphere locally infuses energy. **c** Further growth is initiated by the additional energy. **d**, **e** Replication of an agent triggers further parallel construction. **f**, **g** Expansion of the structure is continued after another energy influx

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, then resume construction (Fig. 18.7d, e). This generates further expansions of the structures and—after more energy boosts (Fig. 18.7f)—gives rise to the structure depicted in Fig. 18.7g. The pattern continues to grow until the agents run out of energy again.

This is only an example of how external manipulation by a breeder, the "gardener", can influence the agent behaviors, thus the building or developmental processes. The evolution of 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. This can be accomplished either automatically or through direct influence from the gardener. Figure 18.8 gives a few examples of evolved swarm grammar ecologies and extracted structures at different stages during their evolution.

18.3.2.3 Swarm Constructions in the Arts

In the examples above, the aesthetic judgement of a breeder drove the evolution of swarm grammars. This works particularly well when an artist searches for innovative expressions of certain artistic themes. Swarm grammar constructions are special in that the dynamics of their construction processes are captured within the emerging

Fig. 18.8 Collage of designs generated by swarm grammars. The centerpiece illustrates a swarm grammar garden ecology, within which the surrounding designs were created

structures. Local interactions determine the placement of construction elements and the flight formations of the swarm. Inherent in any swarm system, the agents' actions and reactions result in a feedback loop of interdependencies [63]. The diagram in Fig. 18.9 hints at the complex relationships that arise in boid systems [49]. Here we do not even consider indirect communication beyond the ever changing neighborhood relations among the swarm individuals. A swarm agent i directly perceives a set of neighbors that determine its acceleration. Its changed location, in turn, affects those swarm mates that perceive i as a neighbor. The emerging dynamics is captured in structures that exhibit liveliness and spontaneity, contrasting themes, rhythmic movements, tension, organic looks, and rigid forms.

Consequently, the artistic interpretation of SG structures can support artistic work in several ways. For example, we composed pieces of computer-generated SG structures and traditionally painted motives [65], and looked at inspiring themes and concepts of artistic works as a whole [66]. Within these explorations—inspired by the architectural potential of swarm grammars—the artist (S.v.M.) combined a collection of swarm structures to create surreal, artificial worlds (Fig. 18.10a, b). In about

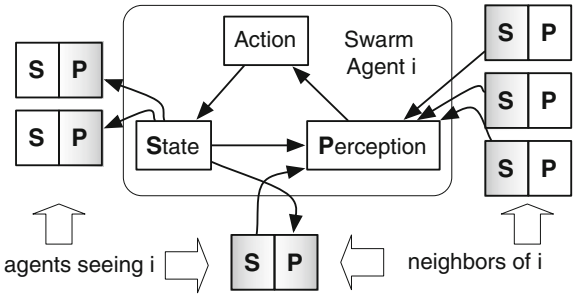

Fig. 18.9 The *black arrows* in the *upper box* show the direction of influence between perception, action and state of a swarm agent *i*. The S-P tuples stand for the state and perception modules of other agents that interact with agent *i*

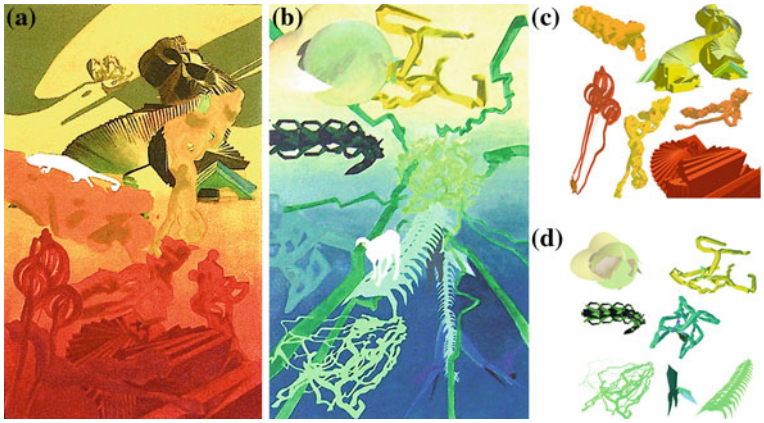

Fig. 18.10 Diptych of the two pieces **a** *caméléon* and **b** *bighorn sheep*. Acrylic medium on canvas, 23" × 38". Selections of swarm grammar structures bred for the diptych are displayed in **c** and **d**, respectively (S.v.M., 2008)

40 interactive evolutionary experiments, the artist bred the sets of swarm grammar structures displayed in Fig. 18.10c, d.

During the evolutionary runs, the artist followed two main objectives. First, robust looking beams should emerge to form a structural mesh, thus opening vast spaces. Secondly, fuzziness, continuity and the resemblance to organic forms should warrant the authenticity of the generated virtual worlds. The color gradients in the background emphasize the wholesome, fluent structural architecture in Fig. 18.10a and the liveliness and dynamics caught in the erratic structures of Fig. 18.10b with warm and cold color palettes, respectively.

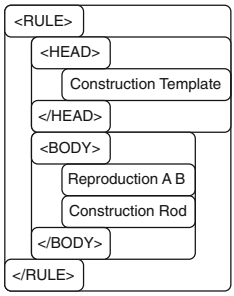

Fig. 18.11 A behavioral rule of a swarm grammar agent

18.3.3 Rule-Based Swarm Grammars

As a second generalization of our SGs, we wanted to go beyond fixed parameters, such as the regularly timed application of reproduction rules or the continuous placement of construction elements. SG rules are now expanded by conditions that each individual agent would relate to, e.g., specific internal states or perception events. An example of such a rule-based genotype is illustrated in Fig. 18.11. Instead of continuous construction and regularly timed reproduction, this rule triggers the reproduction of two agents (types *A* and *B*) and the construction of a rod whenever the acting individual perceives a construction template.

18.3.3.1 Breeding Architecture

Perception-induced rule execution allows for indirect, so-called *stigmergic* communication by which social insects, such as ants, termites and some wasp and bee species, are assumed to coordinate large parts of their construction behaviors [5, 6, 19]. Stigmergy can then be harnessed to create assortments of innovative architectural SG designs by means of computational evolution [60, 61]. In order to automatically drive the evolutionary processes, we need a way to assign fitnesses to SG specimens. There are several aspects that should be taken into account when it comes to measuring structural complexity, as we have outlined in Sect. 18.2.2.

In addition to the analysis of the genotype of a swarm grammar, two aspects can be incorporated into the fitness assignment of an evolutionary algorithm: (1) the construction processes and (2) the emerging structures. Structural analysis is either very coarse-grained, considering for example the overall volume and the proportions, or computationally very costly, for instance when attempting to identify hierarchies and reoccurring modules. Therefore, we put an emphasis on the observation and classification of the construction processes.

In particular, in a series of SG breeding experiments for architectural design, we promoted productivity, diversity and collaboration, and prevented computational outgrowth of the generated structure. Our detailed evolutionary approach to automatic SG evolution is presented in [60]. In order to reward *productivity*, the SG construc-

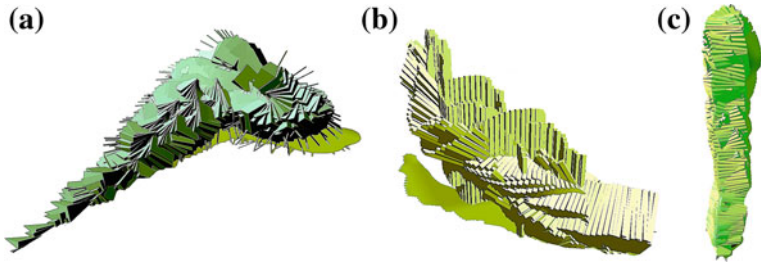

Fig. 18.12 Architectural swarm-built idea models: **a** conical, flat expansion that emerged from continuously reproducing agents, **b** solidified gyration of agents chasing one another, **c** a typical specimen formed from stacked layer elements

tions were compared with (simple) pre-defined structures. More specifically, construction elements built inside a pre-defined cubic shape contributed positively to an SG fitness, whereas constructions outside the cube decreased it. This is similar to an approach we used in [59]. *Diversity* was traced as the total number of expressed agent genotypes, as well as the number of deployed construction materials or construction mechanisms. In order to foster *collaboration* between the SG agents, we observed the numbers of perceived neighbors averaged over all active agents and over time. Low values of perceived neighbors implied that no direct interactions were taking place, whereas large values meant that the agents were trapped within small spaces. Randomly initialized swarm grammar systems can quickly exhaust the provided computing power: Fast, possibly unconditional sequences of SG rule applications may result in exponential agent reproduction. Temporarily, such explosions of activity could be beneficial, for example in designs that produce large numbers of ramifications. In the long run, however, overwhelming demands on computing requirements have to be avoided. As a simple means to prevent prohibitive outgrowth, yet allowing for temporary leaps of activity, we set a time limit for the computation of one specimen. Thus, we filtered inefficient SGs during the evolutionary experiments.

Three examples of architectural SG models are displayed in Fig. 18.12. The flowing and organic shapes built by the bio-inspired, generative SG representation promise to support the design efforts of architects [58, 61]. Utilizing the extended swarm grammar model to breed architectural designs is not only interesting from a creative and innovative perspective on aesthetics, but it also bears the potential for optimizing architectural designs in respect to ecological and economical aspects. Such ecological criteria could be temperature regulation and ventilation [15], adaptation of building structures to the surrounding landscape, utilization of sun exposed structures for electric power generation, and other evolvable and measurable features [17].

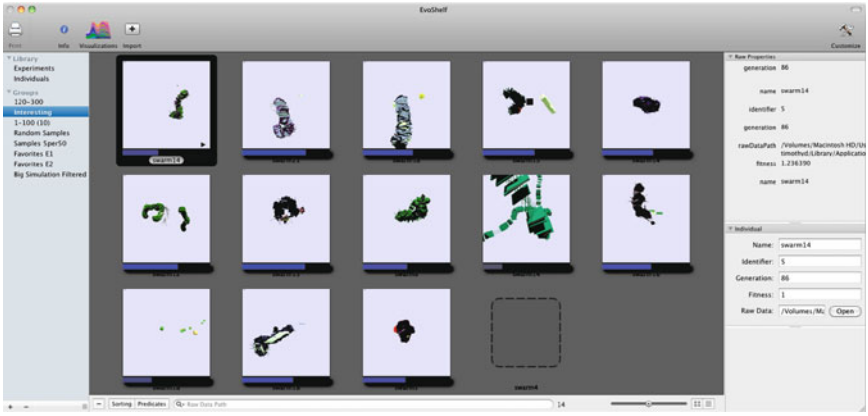

Fig. 18.13 We were able to improve our architectural SG experiments by means of our management and analysis software *EvoShelf*

18.3.3.2 Driving Evolution with *EvoShelf*

In order to further the elaboration and analysis of evolutionary design, we have developed *EvoShelf* [8], a reliable storage/retrieval system for computational evolutionary experiments, and a fast browser for genotype and phenotype visualization and evaluation (Fig. 18.13). With *EvoShelf*, we were able to discover that our preliminary breeding experiments (Sect. 18.3.3.1) tended to produce overfitting SG populations and predominantly promoted the variety of deployed construction elements.

Figure 18.14a depicts a corresponding, representative *FitnessRiver* plot that was created by *EvoShelf*. Our *FitnessRiver* visualization method stacks the fitness values of individuals on top of each other. The fitness of an individual is proportional to the width of its current. Different colors are used to distinguish between successive individuals. Discontinuing currents indicate the removal of an individual from the evolutionary process. In the *FitnessRiver* visualization, the x-axis represents the sequence of generations. The shown plot exposes stagnating and fluctuating fitness development after about 100 generations. A bias towards specific construction materials deployed by the SG specimens could be identified in star plots representing phenotypic features as seen in the corners of the SG visualizations in Fig. 18.14b. Based on these investigations, we are able to adjust our fitness functions and the configuration of our breeding experiments [8].

18.3.4 A Streamlined, Accessible Swarm Simulation Framework

Rule-based swarm systems seem to be a good fit to capture biological models [28–31]. However, there are several hurdles that make it hard to deploy swarm models in fields outside of computer science:

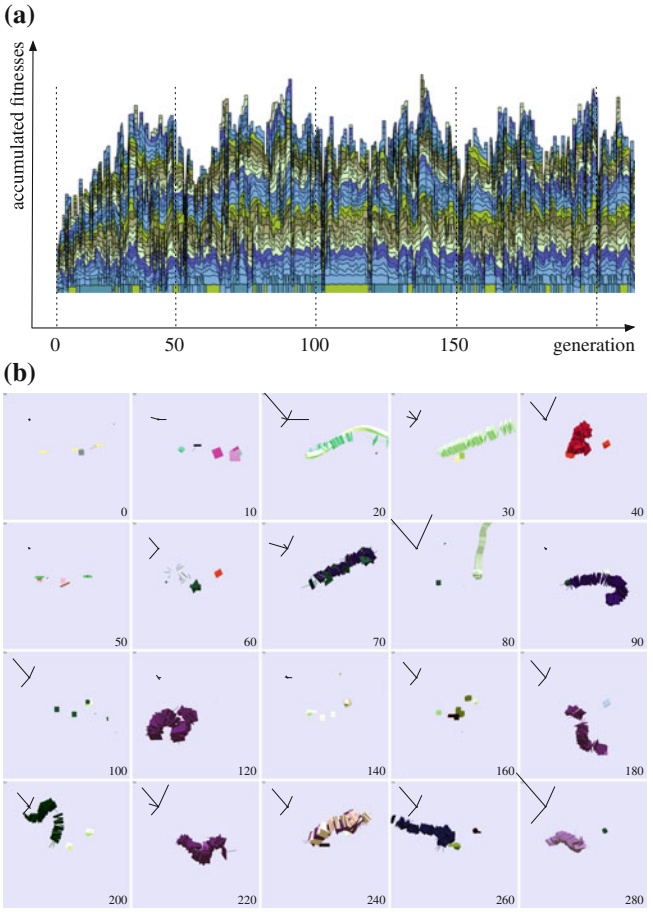

Fig. 18.14 *EvoShelf* provides the user with quick visualization methods for global fitness trends and local comparisons, as in **a** the *FitnessRiver* plot and **b** star plots of the specimens' features, respectively

1. The predicates and actions that drive the simulations—e.g., the detection of a chemical signal or the deposition of a particle—depend on the modeling domains and usually have to be re-implemented for different experiments. Still, many of these operations can be abstracted, parametrically adjusted and reused in different contexts. The integration of these operations into a rule-based formalism also makes it possible to utilize functionality from various computational engines such as physics engines or general differential equation solvers within a single modeling framework.
2. Depending on the degree of specificity of a rule's condition and its associated actions, a theoretically simple interaction can result in an over-complicated rep-

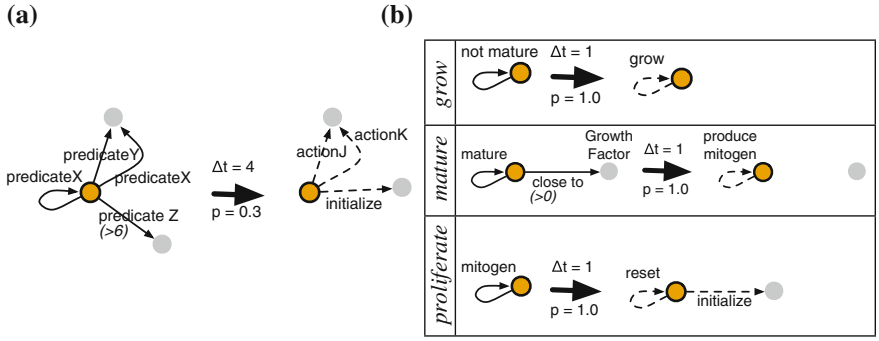

Fig. 18.15 **a** An SGG rule that queries the reference node itself (*orange*), other individuals (*grey*) and sets of interaction candidates. The consequence of the rule defines the interactions, such as deletion of nodes and initialization of a new node. **b** Three rules to describe a simple developmental process model

resentation. A graphical description of the predicates and the associated actions can amend this issue.

3. As swarm simulations often exhibit complex behaviors, small details—for example the order of execution and the discretization steps in a simulation—can greatly influence the outcome. Therefore, we think it is crucial to design models based on a unified algorithmic scheme.

We have devised *swarm graph grammars* (SGGs) to alleviate some of the challenges discussed above [64]. SGGs provide a graphical, rule-based description language to specify swarm agents and a generalized algorithmic framework for the simulation of complex systems. Fundamental operations such as the creation or deletion of programmatic objects, as provided by formal grammars, are part of the SGG syntax. Through SGGs we can capture (metabolic) functions at multiple biological scales. We can capture processes of secretion and diffusion [68] as well as consumption/removal and production/construction [37]. As a consequence of the graph-based syntax, SGGs capture the simulation state in a global graph at each computational step. Thereby, the continuous re-shaping of an interaction topology of a dynamic system is traced and interdependencies that emerge over the course of a simulation can be represented graphically.

18.3.4.1 SGG Rule Description

An SGG agent's behavior is described by a set of rules (Fig. 18.15). Each rule tests a set of predicates (solid edges on the left-hand side) and executes a set of actions (dashed edges on the right-hand side) in respect to the acting agent itself (reference node) or other agents. Nodes represent individual agents or sets of agents. In Fig. 18.15, the acting agent is displayed as an orange node with a black border. Other

agents or agent groups are depicted as grey nodes. The application of the rule is associated with a frequency and a probability. Sets of predicates can attempt to identify an arbitrary number of agents. The relative location, i.e., the 2D coordinates, of the node on the left-hand side of the rule is matched with its appearance on the right-hand side of the rule. If a node does not reappear on the right-hand side, it implies that its corresponding agent has been removed. If a node appears at a location that is unoccupied on the left-hand side, a new node is created. Figure 18.15a shows an example rule. This rule is applied with a probability of $p = 0.3$ at every fourth time step ($\Delta t = 4$) of the agent simulation.

One (arbitrarily chosen) node that fulfills *predicateX* and *predicateY* is affected by *actionJ* and *actionK*. Also note that a new node is created and is initialized, for which no reference had existed before. In case there are at least 6 nodes that fulfill *predicateZ*, they will all be removed.

18.3.4.2 Swarm-Based Embryogeny and Morphological Development

Swarm graph grammars enable us to closely collaborate with researchers from other disciplines such as architecture, biology or medical sciences. Following the footsteps of previous works in artificial embryogeny and morphogenetic engineering [2, 3, 9, 13, 38], we have begun to investigate simulations of biological developmental processes. In a series of (at this point naive) experiments, we integrated high-level SGG agent behaviors (growth, maturation and proliferation) with physical mechanics (collision and impulse resolution). Figure 18.15b shows the applied SGG rules which configure cells to *grow* until they reach maturity (predicates: *not mature* and *mature*). Mature cells that are *close to* a *Growth Factor* increase their internal *mitogen* concentration which in turn instigates proliferation (modeled as *reset* of the acting cell and *initialization* of a second cell).

Accordingly, in the simulation shown in Fig. 18.16, tissue cells (blue: not mature; red: mature) within the vicinity of a signaling molecule (green) start to proliferate. Collision resolution through an embedded physics engine allows the cells to assemble[2]. The emerging protuberance is slanted to the right in accordance with the initial distribution of signaling molecules.

Initially, we were surprised to see that the protuberance in Fig. 18.16 turned out symmetrical, despite its one-sided development. We speculated that this was due to a lack of simulated cell polarization. However, after a series of systematic simulations, we found out that the effects of polarization, in combination with proliferation, would still be overturned by the physics interactions and again result in spherically distributed, aggregated cells (Fig. 18.17).

Using this same agent-based approach, we have begun tracing embryogenic developments in mice. Volumetric embryo data provides a basis to populate initial tissue layers with cells (Fig. 18.18). Basic intra- and inter-cellular interactions then dic-

[2] In the given experiment we rely on the Bullet physics engine, http://bulletphysics.org.

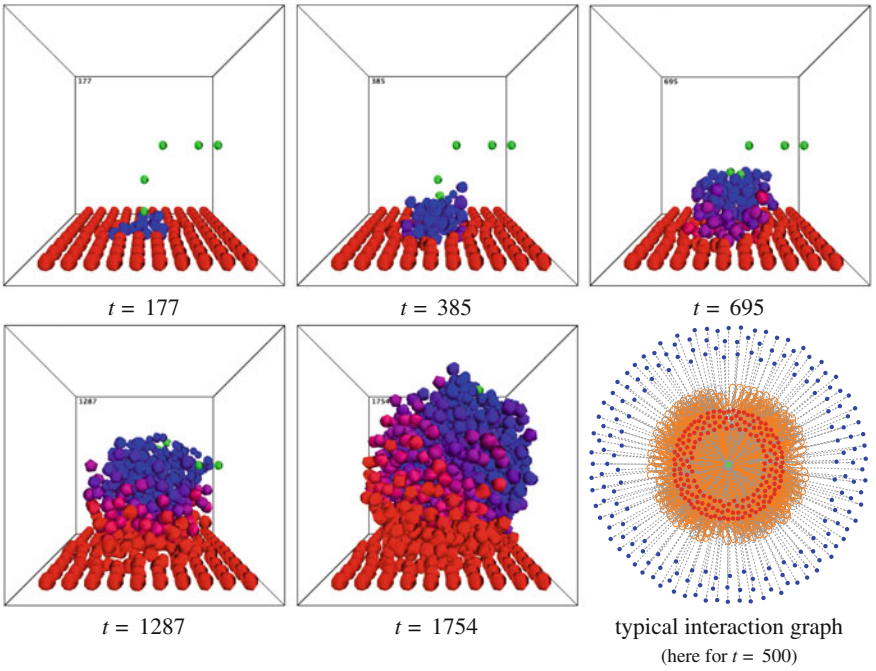

Fig. 18.16 The proliferation of mature cells (*blue* premature; *red* mature) is dependent on the proximity to growth factors (*green*). At any time of the simulation, large numbers of agents are informed by growth factors leading to typically dense but homogeneous graphs that reflect their interactions

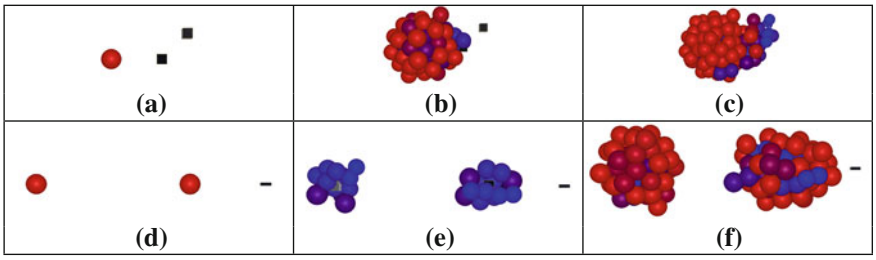

Fig. 18.17 **a–c** Show the same proliferation process as in Fig. 18.16 but with only one initial cell (growth factors are illustrated as *black cubes*); **d–f** Show two simultaneously growing protuberances, whereas the cells on the right-hand side obtain a polarization aligned towards the polarization signal to the right (*black box*)

tate the shaping of existing and the creation of new tissue layers as shown in the mesh-deformation in Fig. 18.19.

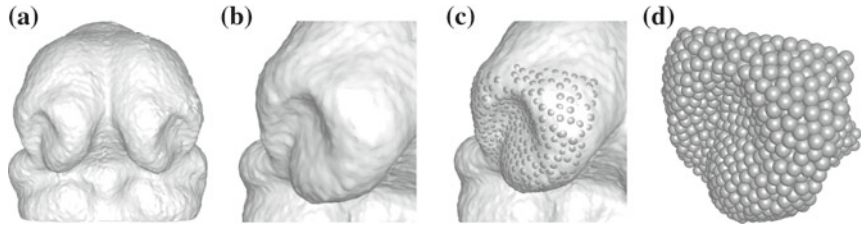

Fig. 18.18 **a** We start with a volumetric scan of a mouse embryo, **b** zoom into the region of interest and **c**, **d** populate it with swarm agents

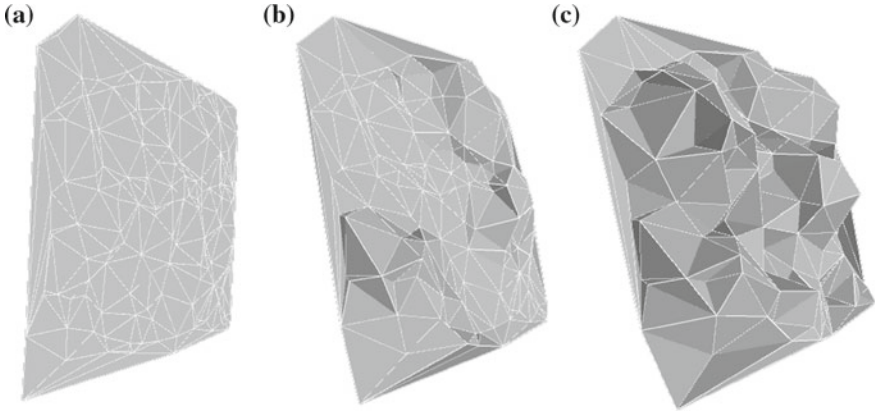

Fig. 18.19 Spatial configuration **a** post, **b** during, and c post interaction

Here, sentient swarm agents play the role of vertices on a graphical surface. In this context, dynamic mesh generation and manipulation could become part of the agents' sets of possible actions [56].

18.4 Summary and Conclusion

Inspired by the construction abilities of social insects, we started investigations into virtual constructive swarms [32]. We designed swarm grammars (SGs) as a computational developmental representation that combines the ideas of artificial swarm simulations[3] with the compositional regulation expressed by formal grammars [7]. L-systems are a prominent approach to translate formal grammars (rewrite rules) into the realm of developmental models [40] (Sect. 18.3). SGs allow for completely

[3] Artificial swarms can be considered a special case of agent-based modeling with a focus on large numbers of locally interacting individuals and the potential of emergent phenomena which cannot be inferred from the individuals' abilities.

unrestrained interaction topologies and provide a simple way to integrate interactions beyond population control and fixed local neighborhood relationships, which represents an expansion from the more constrained L-systems.

In an iterative process of unification and extension of the initial swarm grammar representation, we first incorporated a complete swarm grammar genotype into each swarm agent (Sect. 18.3.2), then started describing its behavior as a set of perception-reaction rules (Sect. 18.3.3). The original idea of using grammatical production to determine the composition of the swarm population became part of a more generic agent-based representation [10, 11]. To even further the modeling capacity of swarm-based simulations, we designed swarm graph grammars (SGGs) as a means to graphically represent swarm agent interactions and to explicitly model inter-agent relationships that might influence the dynamics of the simulations (Sect. 18.3.4). Swarm graph grammars provide a modeling language that can be used for interdisciplinary investigations. In a collaborative project, we have begun tracing complex developmental processes in mice (Sect. 18.3.4.2).

An expanded degree of freedom in SG representations required systematic exploration of configuration spaces. We addressed this challenge by means of computational evolution [32, 62, 60]. In particular, we relied on interactive evolution to explore structural spaces (Sect. 18.3.1.2) that inspired artistic works [65, 66] (Sect. 18.3.2.3). We furthered this approach by the possibility to breed large swarm grammar ecologies in virtual spaces (Sects. 18.3.2.1 and 18.3.2.2). We promoted structural complexity by considering the frequency and diversity of interaction processes among swarm agents in order to generate interesting architectural designs (Sect. 18.3.3.1). More systematic investigations in accordance with scalable complexity measures as outlined in Sect. 18.2.2 might yield a better performance in the context of breeding innovative designs.

With the evolution and exploration of swarm grammars, we have been building methodologies and toolkits that support modeling and simulation of developmental systems in a multitude of domains. Evolutionary computation techniques enable us to find swarm system configurations to trace more or less desired or innovative outcomes for artistic or scientific simulations. However, we are aware that there are several major obstacles to be addressed before our methodologies can become instrumental for broad application. Currently, we focus on two issues. First, we attempt to reduce the computational complexity that inevitably arises from offering a very generous and expressive representation [55]. Second, we are constantly engaged in improving the usability and accessibility of our modeling representation itself.

References

1. Banzhaf, W.: Artificial chemistries—towards constructive dynamical systems. Solid State Phenom. **97**, 43–50 (2004)
2. Bentley, P.J., Kumar, S.: Three ways to grow designs: a comparison of embryogenies for an evolutionary design problem. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999) (1999)

3. Beurier, G., Michel, F., Ferber, J.: A morphogenesis model for multiagent embryogeny. In: Proceedings of the 10th International Conference on the Simulation and Synthesis of Living Systems (ALIFE X) (2006)
4. Blackwell, T.: Swarming and music. In: Miranda, E.R., Biles, J.A. (eds.) Evolutionary Computer Music, pp. 194–217. Springer, London (2007)
5. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, New York (1999)
6. Camazine, S., Deneubourg, J.L., Franks, N.R., Sneyd, J., Theraulaz, G., Bonabeau, E.: Self-Organization in Biological Systems. Princeton Studies in Complexity. Princeton University Press, Princeton (2003)
7. Chomsky, N.: Three models for the description of language. IRE Trans. Inf. Theory **2**(3), 113–124 (1956)
8. Davison, T., von Mammen, S., Jacob, C.: Evoshelf: a system for managing and exploring evolutionary data. In: Proceedings of Parallel Problem Solving from Nature (PPSN) (2010)
9. De Garis, H.: Artificial embryology: the genetic programming of an artificial embryo. In: Soucek, B. (ed.) Dynamic, Genetic, and Chaotic Programming. Wiley, New York (1992)
10. Denzinger, J., Kordt, M.: Evolutionary on-line learning of cooperative behavior with situation-action-pairs. In: ICMAS, pp. 103–110. IEEE Computer Society (2000)
11. Denzinger, J., Winder, C.: Combining coaching and learning to create cooperative character behavior. In: CIG. IEEE (2005)
12. Deussen, O., Hanrahan, P., Lintermann, B., Mech, R., Pharr, M., Prusinkiewicz, P.: Realistic modeling and rendering of plant ecosystems. In: SIGGRAPH 98, Computer Graphics, Annual Conference Series, pp. 275–286. ACM SIGGRAPH (1998)
13. Doursat, R.: Organically grown architectures: creating decentralized, autonomous systems by embryomorphic engineering. In: Würtz, R.P. (ed.) Organic Computing. Springer, Berlin (2007)
14. Ebner, M.: Coevolution and the red queen effect shape virtual plants. Genet. Program Evolvable Mach. **7**(1), 103–123 (2006)
15. Farmer, G., Guy, S.: Visions of ventilation: pathways to sustainable architecture. Department of Architecture, University of Newcastle upon Tyne, Newcastle upon Tyne (2002)
16. Giavitto, J.L., Michel, O.: Modeling the topological organization of cellular processes. Biosystems **70**(2), 149–163 (2003)
17. Gowri, K.: Green building rating systems: an overview. ASHRAE J. **46**(11), 56–60 (2004)
18. Hallgrímsson, B., Boughner, J.C., Turinsky, A., Parsons, T.E., Logan, C., Sensen, C.W.: Geometric morphometrics and the study of development. In: Sensen, C.W. (ed.) Advanced Imaging in Biology and Medicine, pp. 319–336. Springer, Heidelberg (2009)
19. Hölldobler, B., Wilson, E.O.: The Ants. Springer, Berlin (1990)
20. Hornby, G.S.: Measuring complexity by measuring structure and organization. In: Srinivasan, D., Wang, L. (eds.) 2007 IEEE Congress on Evolutionary Computation, pp. 2017–2024. IEEE Press, Singapore (2007)
21. Hornby, G.S., Pollack, J.B.: Body-brain co-evolution using L-systems as a generative encoding. In: Spector, L., Goodman, E.D., Wu, A., Langdon, W.B., Voigt, H.M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M.H., Burke, E. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), pp. 868–875. Morgan Kaufmann, San Francisco (2001)
22. Hornby, G.S., Pollack, J.B.: Evolving l-systems to generate virtual creatures. Comput. Graph. **25**, 1041–1048 (2001)
23. Hu, D., Marcucio, R.: A SHH-responsive signaling center in the forebrain regulates craniofacial morphogenesis via the facial ectoderm. Development **136**(1), 107 (2009)
24. Jacob, C.: Genetic l-system programming. In: PPSN III–Parallel Problem Solving from Nature, Lecture Notes in Computer Science, vol. 866, pp. 334–343. Springer, Jerusalem (1994)
25. Jacob, C.: Evolving evolution programs: Genetic programming and l-systems. In: Koza, J.R., Goldberg, D.E., Fogel, D.B., Riolo, R. (eds.) Genetic Programming 1996: First Annual Conference, pp. 107–115. MIT Press, Cambridge, Stanford University, Palo Alto (1996)

26. Jacob, C.: Evolution and co-evolution of developmental programs. Computer Physics Communications, Special Issue, Modeling Collective Phenomena in the Sciences (1999) **121–122**, 46–50
27. Jacob, C.: Illustrating Evolutionary Computation with Mathematica. Morgan Kaufmann Publishers, San Francisco (2001)
28. Jacob, C., Barbasiewicz, A., Tsui, G.: Swarms and genes: Exploring λ-switch gene regulation through swarm intelligence. In: CEC 2006, IEEE Congress on Evolutionary Computation (2006)
29. Jacob, C., Burleigh, I.: Biomolecular swarms: an agent-based model of the lactose operon. Nat. Comput. **3**(4), 361–376 (2004)
30. Jacob, C., Litorco, J., Lee, L.: Immunity through swarms: Agent-based simulations of the human immune system. In: Artificial Immune Systems, ICARIS 2004, Third International Conference. LNCS 3239, Springer, Catania (2004)
31. Jacob, C., Steil, S., Bergmann, K.: The swarming body: Simulating the decentralized defenses of immunity. In: Artificial Immune Systems, ICARIS 2006, 5th International Conference. Springer, Oeiras (2006)
32. Jacob, C., von Mammen, S.: Swarm grammars: growing dynamic structures in 3d agent spaces. Digital Creativity: Special issue on Computational Models of Creativity in the Arts **18**(1), 54–64 (2007)
33. Jones, D.: Atomswarm: a framework for swarm improvisation. In: Giacobini, M., et al. (eds.) Applications of Evolutionary Computing, pp. 423–432. Springer, Heidelberg (2008)
34. Kauffman, S.: The Origins of Order. Oxford University Press, New York (1993)
35. Kókai, G., Tóth, Z., Ványi, R.: Modelling blood vessel of the eye with parametric l-systems using evolutionary algorithms. In: Horn, W., Shahar, Y., Lindberg, G., Andreassen, S., Wyatt, J.C. (eds.) Artificial Intelligence in Medicine, Proceedings of the Joint European Conference on Artificial Intelligence in Medicine and, Medical Decision Making, AIMDM'99, 1620, pp. 433–443 (1999)
36. Kókai, G., Ványi, R., Tóth, Z.: Parametric l-system description of the retina with combined evolutionary operators. In: Genetic and Evolutionary Computation Conference, GECCO-99. Orlando, Florida, USA (1999)
37. Kumar, S., Bentley, P. (eds.): On Growth. Form and Computers. Elsevier Academic Press, London (2003)
38. Kumar, S., Bentley, P.: Biologically inspired evolutionary development. Evolvable Systems: From Biology to Hardware, pp. 99–106. Springer, Heidelberg (2003)
39. Kwong, H., Jacob, C.: Evolutionary exploration of dynamic swarm behaviour. In: Congress on Evolutionary Computation. IEEE Press, Canberra (2003)
40. Lindenmayer, A.: Developmental systems without cellular interactions, their languages and grammars. J. Theor. Biol. **30**(3), 455–484 (1971)
41. McKean, E. (ed.): The New Oxford American Dictionary. Oxford University Press, Oxford (2005)
42. Mech, R., Prusinkiewicz, P.: Visual models of plants interacting with their environment. In: SIGGRAPH'96, pp. 397–410. ACM SIGGRAPH, New York (1996)
43. Michalewicz, M.T. (ed.): Plants to Ecosystems: Advances in Computational Life Sciences. CSIRO Publishing, Collingwood (1997)
44. Michel, F., Beurier, G., Ferber, J.: The turtlekit simulation platform: application to complex systems. In: Proceedings of Workshop Sessions at the 1st International Conference on Signal & Image Technology and Internet-Based Systems (IEEE SITIS05), pp. 122–128. IEEE Press, Canberra (2005)
45. Mock, K.J.: Wildwood: the evolution of l-system plants for virtual environments. In: IEEE Conference on Evolutionary Computation, pp. 476–480. IEEE Press, New York (1998)
46. Polack, F.A.C., Andrews, P.S., Ghetiu, T., Read, M., Stepney, S., Timmis, J., Sampson, A.T.: Reflections on the simulation of complex systems for science. In: ICECCS 2010: Fifteenth IEEE International Conference on Engineering of Complex Computer Systems, pp. 276–285. IEEE Press, Canberra (2010)

47. Prusinkiewicz, P., Hammel, M., Hanan, J., Mech, R.: Visual models of plant development. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages. Springer, New York (1997)
48. Prusinkiewicz, P., Lindenmayer, A.: The Algorithmic Beauty of Plants. Springer, Heidelberg (1996)
49. Reynolds, C.W.: Flocks, herds, and schools: a distributed behavioral model. Comput. Graph. **21**(4), 25–34 (1987)
50. Russel, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Pearson Education, Upper Saddle River (2010)
51. Salazar-Ciudad, I.: Tooth Morphogenesis in vivo, in vitro, and in silico. Curr. Top. Dev. Biol. **81**, 342 (2008)
52. Schuster, P.: How does complexity arise in evolution. Complex **2**(1), 22–30 (1996)
53. Settles, M., Nathan, P., Soule, T.: Breeding swarms: a new approach to recurrent neural network training. In: GECCO '05: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, pp. 185–192. ACM Press, New York (2005)
54. Settles, M., Soule, T.: Breeding swarms: a ga/pso hybrid. In: GECCO '05: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, pp. 161–168. ACM Press, New York (2005)
55. Shirazi, A.S., von Mammen, S., Jacob, C.: Adaptive modularization of the mapk signaling pathway using the multiagent paradigm. In: Proceedings of Parallel Problem Solving from Nature (PPSN) (2010)
56. Smith, C.: On vertex-vertex systems and their use in geometric and biological modelling. Ph.D. thesis, University of Calgary (2006)
57. Spector, L., Klein, J., Perry, C., Feinstein, M.: Emergence of collective behavior in evolving populations of flying agents. In: Genetic and Evolutionary Computation Conference (GECCO-2003), pp. 61–73. Springer, Chicago (2003)
58. Van der Ryn, S., Cowan, S.: Ecological Design. Island Press, Washington (2007)
59. von Mammen, S., Jacob, C., Kókai, G.: Evolving swarms that build 3d structures. In: CEC 2005, IEEE Congress on Evolutionary Computation, pp. 1434–1441. IEEE Press, Edinburgh (2005)
60. von Mammen, S., Jacob, C.: Evolutionary swarm design of architectural idea models. In: Genetic and Evolutionary Computation Conference (GECCO) 2008, pp. 143–150. ACM Press, New York (2008)
61. von Mammen, S., Jacob, C.: Swarm-driven idea models—from insect nests to modern architecture. In: Brebbia, C. (ed.) Eco-Architecture 2008, Second International Conference on Harmonisation Between Architecture and Nature, pp. 117–126. WIT Press, Winchester (2008)
62. von Mammen, S., Jacob, C.: Genetic swarm grammar programming: Ecological breeding like a gardener. In: Srinivasan, D., Wang, L. (eds.) 2007 IEEE Congress on Evolutionary Computation, pp. 851–858. IEEE Press, Canberra (2007)
63. von Mammen, S., Jacob, C.: The spatiality of swarms—quantitative analysis of dynamic interaction networks. In: Proceedings of Artificial Life XI, pp. 662–669. MIT Press, Massachusetts (2008)
64. von Mammen, S., Phillips, D., Davison, T., Jacob, C.: A graph-based developmental swarm representation & algorithm. In: ANTS 2010: Seventh International Conference on Swarm Intelligence. Springer, Heidelberg (2010)
65. von Mammen, S., Wissmeier, T., Wong, J., Jacob, C.: Artistic exploration of the worlds of digital developmental swarms. LEONARDO (2010)
66. von Mammen, S., Wong, J., Jacob, C.: Virtual constructive swarms: compositions and inspirations. In: Applications of Evolutionary Computing, Proceedings of EvoWorkshops 2008, Lecture Notes in Computer Science, vol. 4974, pp. 491–496. Springer, Berlin (2008)
67. von Neumann, J., Burks, A.W.: Theory of self-reproducing automata. University of Illinois Press, Urbana and London (1966)

68. Walker, D.C., Southgate, J.: The virtual cell-a candidate co-ordinator for 'middle-out' modelling of biological systems. Briefings in Bioinformatics **10**(4), 450–461 (2009)
69. Wolfram, S.: Cellular automata as models of complexity. Nature **311**, 419–424 (1984)
70. Yu, J.: Evolutionary design of 2d fractals and 3d plant structures for computer graphics. Master's thesis, Department of Computer Science, University of Calgary (2004)