Genetic Swarm Grammar Programming: Ecological Breeding like a Gardener

Sebastian von Mammen
Dept. of Computer Science

University of Calgary
Calgary, Alberta T2N 1N4 Canada
Email: s.vonmammen@ucalgary.ca

Christian Jacob
Dept. of Computer Science
Dept. of Biochemistry & Molecular Biology
University of Calgary
Calgary, Alberta T2N 1N4 Canada
Email: cjacob@ucalgary.ca

***Abstract*— We recently introduced swarm grammars as an extension of Lindenmayer systems to model dynamic growth processes in 3D space through a large number of interacting (swarm) agents. Grammatical rewrite rules define different types of agents and their evolution over time. Sets of parameters determine specific interaction behaviors among the generated swarms.**

As we will show, swarm grammars lend themselves to creating an ecology of interacting entities and dynamic structures that are built by a multitude of agents. In addition to a rather traditional approach of evolving swarm grammars through interactive genetic programming, we explore new ways of designing ecologies of swarm agents by immersing the breeder into the growth and evolution processes. The system designer takes on the role of a 'tinkerer' or 'gardener', who is equipped with tools to influence and shape the on-going growth, evolutionary, and other dynamic processes within the swarm grammar ecology. Spatial genetic operators can be directed to specific locations within the evolving swarms. This enables the breeder to overview large numbers of phenotypic developmental processes and implicitly direct their evolution.

I. Introduction: A Garden of Swarms

Evolution in nature is a competitive and distributed process. Agents have to compete for resources to secure their survival. Natural evolution also occurs in physical space, that is organisms as well as developmental and evolutionary processes are constrained by physical laws. Furthermore, evolution utilizes physical properties which naturally constrain the number of possible solutions. Our Swarm Grammar (SG) system incorporates several of these factors. The co-evolutionary SG system works with a multitude of agents, which we subdivide into swarms that exhibit certain properties shared by specific types of agents [1]. The swarming agents act as self-organizing builders that compose three-dimensional structures, while they are interacting with each other, similar to termites or ants building their nests [2], [3]. This creates the scenario of an emergent garden ecology, in which a gardener arranges plants, takes care of them, and breeds or re-seeds plants over time. The ecology has its own dynamics, dependent on the physical properties of the simulated world and determined by the swarm agents' attributes, such as their speed, interaction dynamics, or separation and cohesion urges. Consequently, the breeder or designer is not in complete control of the overall evolutionary dynamics (which structures are built and where?), but can influence both the interaction processes as well as the evolutionary processes at any time and any location within the ecology. This is not unlike PolyWorld, a co-evolutionary virtual 2D world, in which agents, controlled by neural networks, evolve [4].

Parallel rewrite systems as a grammatical paradigm provide beneficial models to study and capture the formation of complex systems [5]. We specify agent interactions through swarm grammars [6] that are an extension of Lindenmayer systems [7], [8], which incorporate aspects of developmental design and morphogenesis. Both the rewrite rules and agent parameters are evolvable over time and help to breed structures in 3D space.

Christian Jacob is a faculty member in the Department of Computer Science and the Department of Biochemistry & Molecular Biology, University of Calgary, Calgary, Alberta, T2N 1N4, Canada (email: cjacob@ucalgary.ca).

Sebastian von Mammen is a Ph.D. student in the Department of Computer Science, University of Calgary, Calgary, Alberta, T2N 1N4, Canada.

II. Related Work

Our Swarm Grammar approach incorporates principles of morphogenesis, multi-agent systems, co-evolution, and interactive design. Therefore, we give a brief overview of related work in these respective areas.

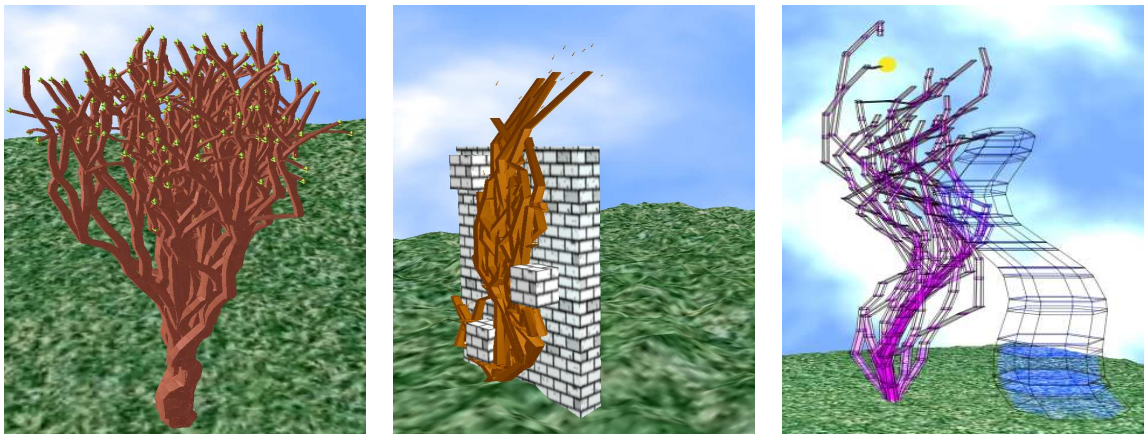
A. Design through Development

Embryogenic and developmental approaches have been investigated for some time in the context of how designs can be grown instead of built [9], or how growth processes facilitate evolution [10]. The creativity that is facilitated by evolutionary systems to create forms and functional designs [11], [12] has led to interesting bridges between simulated design worlds and the automated manufacturing of physical and functional objects [13], [14]. None of these approaches has employed swarm intelligence to create designs such as in [6].

B. Design through Multi-agents

More recently, promising multi-agent systems have been investigated to build and evolve virtual organisms [15]. It has also been shown that swarms of agents can be evolved

1-4244-1340-0/07$25.00 ©2007 IEEE



$SL_a = \{A, \{A \rightarrow BBB, B \rightarrow A\}\}$ $SL_b = \{N, \{N \rightarrow NN\}\}$ $SL_c = \{I, \{I \rightarrow I\}\}$

(a) (b) (c)

Fig. 1. Swarm Grammar agents interacting with their environment and their corresponding swarm rewrite systems. (a) 243 agents building a tree-like structure. All agents—which are visualized as pyramidal shapes at the branch tips—have an upward urge, but B-agents repel from each other, which creates the bushy crown. (b) A similar set of swarm grammar agents is forced to climb up a wall. 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. (c) 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 which are repelling from each other.

to perform sorting tasks by arranging similar objects into clusters [16], [17], [18]. Evolutionary algorithms have been used to breed swarms of agents that display choreographed dynamics [19] or build structures in 3D space [20]. Similar reconstruction algorithms for 3D objects were also implemented with models of honey bees [21]. Computational models that combine morphogenesis and multi-agents show interesting analogies to ebryogenetic processes in fruit flies [22]. Many evolutionary multi-agent systems exploit cooperation or competition in a coevolutionary environment, such as [23].

C. Interactive Evolution

Design is an iterative process. Human design, in particular, is an interactive process. Consequently, different techniques for engaging a system-external designer or evaluator into evolutionary computing have been studied. One of the early examples involves interactive evolution of procedural models for the creation of pictures and textures in computer graphics [24]. Especially interesting results were obtained with interactive techniques in the reproduction and analysis of natural evolutionary processes [25], [26]. A more formal representation of interactive evolution was proposed even before interactive evolutionary methodologies became more applicable due to the faster processing power of desktop computers [27]. Today, interactive evolutionary techniques are starting to become more sophisticated by applying machine learning techniques to adjust to user input and preferences [28].

D. Developmental Modeling and Lindenmayer-Systems

Our swarm grammars are extensions of Lindenmayer systems (L-systems) [8], which—quite successfully—have been used for the grammatical encoding of growth processes and generation of structures in two- and three-dimensional space. Plants have been modeled extensively with L-systems [29], [30], including simulated plants that interact with their environment [31], [32], [33]. Original work in genetic programming of L-systems [34], [35], [36] has led to several platforms for L-system evolution [37], [38], [39] and the breeding of virtual plants in a coevolutionary scenario, which even displays competitive arms-race situations [40]. Beyond plants, L-systems have also been used to evolve virtual creatures and their control networks [41], [42] and for the reconstruction of retina and blood vessel structures [43], [44]

III. Swarm Grammars

Following our previous work on swarm-based simulations [45], [6] and evolutionary swarms [19], we define a swarm grammar (SG) system as composed of two parts: (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.

A. SG Rewrite Rules

A 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 [38] and [8]. 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 such as separation or cohesion urges that determine its overall dynamics and interaction behavior as outlined in Table I.

B. Controlling the Swarm Agents' Interactions

Graphically, a swarm agent is represented as a pyramid with its tip pointing in the direction of the agent's velocity vector (Fig. 1). Each agent is only aware of other flock mates (its neighbors) within its radial field of perception which is defined by a radius (r) and an angle (β). The velocity of an agent is constantly updated with an acceleration vector V_{acc} according to a simple 'boids' model [46]:

$$V_{acc} = c_1 V_1(d) + c_2 V_2 + c_3 V_3 + c_4 V_4 + c_5 V_5. \quad (1)$$

Agents change their direction and adjust their speed according to three influential factors: (1) *separation* ($V_1(d)$), where an agent steers away from the collective of neighbors, given the minimum distance to other agents is smaller than a crowding radius d [19]; (2) *cohesion* (V_2), where the agent moves toward the average position of local flock mates; and (3) *alignment* (V_3), where the agent is oriented toward the average direction of its neighbors.

Vector V_4 points to the center of the simulated 3D world and V_5 represents a random unit-length vector to add some noise. The weights $c_1, ..., c_5$ determine how much influence each factor has on the agent. Each of these 'urges', c_j, is specified for an agent type as part of a swarm grammar. An agent stops applying the SL-system rules when it runs out of energy. Energy levels are inherited through replication.

The energy level also influences certain properties of the built 3D structures such as, for example, their size. The type of a swarm individual determines the visual representations of the construction elements it can leave behind on its journey through the simulated 3D scenario as illustrated in Figure 1. Several values characterize these building blocks: each scaled cylindrical object is placed in space at an agent's location after the swarm has flown for a certain number of iterations. The shorter these intervals are, the smoother the appearance of the emerging construction. The color and the numbers of edges define the design of the cylindrical shapes (Table I).

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

$$SL_a = (\alpha = A, P = \{A \rightarrow BBB, B \rightarrow A\}), \quad (2)$$

$$\Delta_a = \{\Delta_A, \Delta_B\} \quad (3)$$

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 Figure 1(a). Note that here and in the following examples we assume $\theta = 1$, that is a matching rule is always applied.

Each step of applying the production rules (in parallel) represents a decision point for all agents within the system. Contrary to L-systems [8], where only a single 'turtle' is used to interpret a string, we employ a swarm of interacting agents. Neither do we 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 application aspects are given in [6].

TABLE I
PARAMETER RANGES FOR A SWARM INDIVIDUAL

Symbol	Variable	Min	Max
r	Perception field radius	50	150
β	Perception field angle	2	6.28
		—	
$w_{x,y,z}$	x-y-z world center coordinates	-1000	1000
c_1, c_2, c_3	separation, cohesion, alignment	-2	2
c_4, c_5	world center attraction, noise	0	1
V_{max}	Maximum velocity	0	25
A_{max}	Maximum acceleration	0	40
I_e	Energy loss per iteration	0	0.25
I_b	Iterations until branching	20	150
I_d	Iterations until drawing	15	30
$Col_{r,g,b}$	Color range (for each r, g and b)	0	1
Cyl_E	Number of cylinder edges	3	13
Cyl_S	Cylinder scaling	0	2

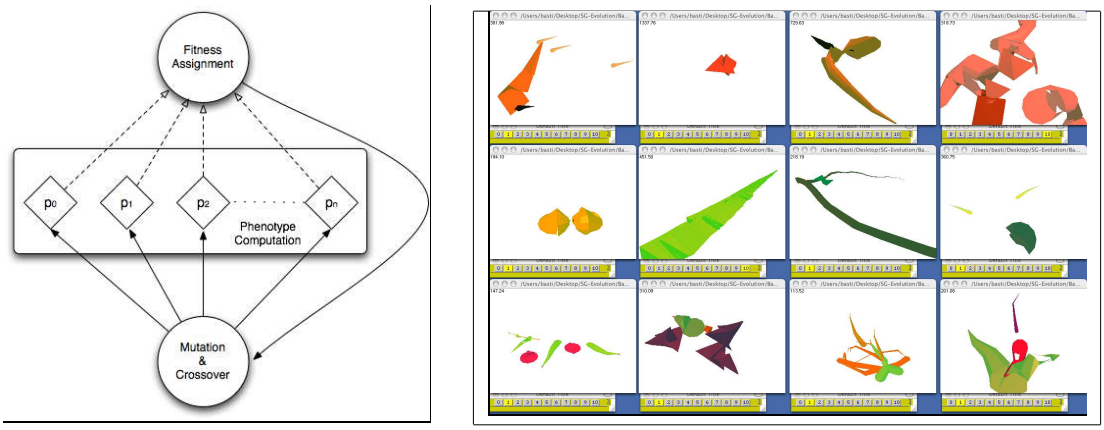
IV. Genetic Swarm Grammar Programming

Combining swarm systems with evolutionary computing has to our knowledge only been considered in the context of particle swarm optimization (e.g., [47], [48]). Emergence of collective behavior has been investigated for agents within a three-dimensional, static world [49], but this did not involve interactive evolution. Our *Genetic Swarm Grammar Programming* (GSGP) approach incorporates both interactive, user-guided evolution as well as the utilization of emergent properties from interactions of a large number of agents.

We will first describe the GP data structures to implement swarm grammars and their associated swarm agents. We introduce new spatial genetic operators through which a designer or breeder can interactively direct the emergent and evolutionary processes within a swarm simulation.

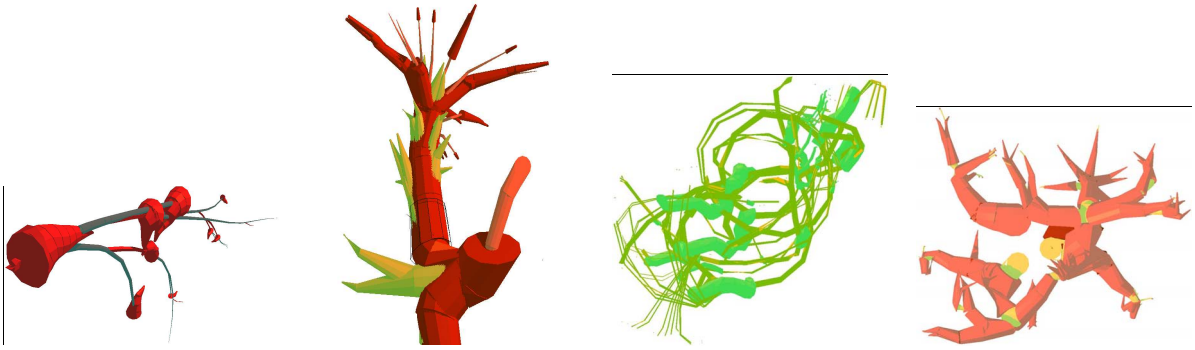
A. Swarm Grammar Genotypes and Genetic Operators

The rewrite rules and agent parameters are represented as symbolic expressions, so that GP can be used to evolve both the set of rules as well as any agent attributes. This follows our *EVOLVICA* framework [38], where all rewrite rules and agent parameters are encoded as symbolic expressions [19]. For the examples we present here, only context-free rules



(a) (b)

Fig. 2. Standard Interactive Evolution: (a) The concept of interactive evolution with manual fitness assignment. The phenotypes (p_0 to p_n) are computed separately, inspected manually, and rated. The assigned fitness values determine the selection probabilities of the genetic operators: mutation and crossover. Arrows illustrate the flow of information. Dashed lines represent visual inspection. (b) The Inspirica [19] user interface helps to 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.



(a) (b) (c) (d)

Fig. 3. Examples of Evolved Swarm Grammar Phenotypes: (a) Pointy yet smooth nodes connect with 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.

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. Again, each agent type is described by the coefficients listed in Table I.

In the context of this paper standard GP tree-crossover and subtree mutations are the only genetic operators used [38]. The replication of an agent (as determined by the grammar) and its associated constructions cease as soon as a swarm agent 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 Figures 1 and 3.

B. Interactive Breeding of Swarm Grammars

We use an extension of Inspirica [19], one of our evolutionary design tools, to explore the potential of the described swarm grammar systems. Following a standard interactive evolutionary approach, as outlined in Figure 2, one can easily—within only a few generations—create structures as illustrated in Figure 3.

V. Immersive Evolution

In the previous example the phenotypes are grown in separate spaces whereas the subsequent fitness assignment is realized through a two-dimensional user interface. The isolated swarm grammar phenotypes, as depicted in Figure 2,

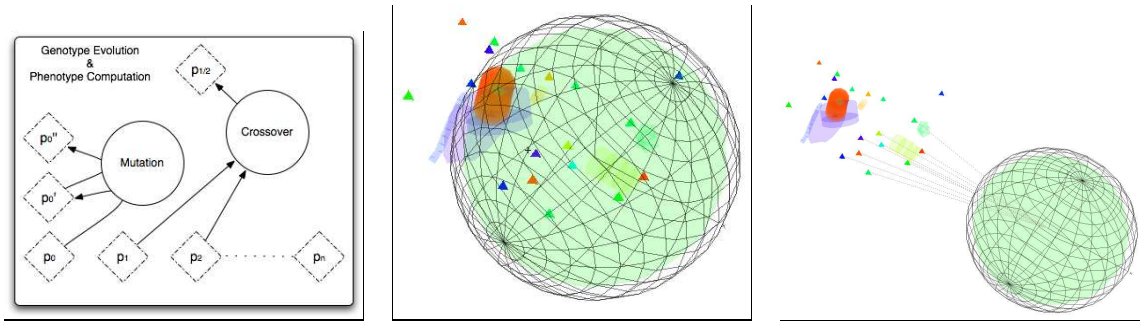


Fig. 4. Immersive Evolution: (a) An immersive evolutionary environment integrates the computation of the phenotypes (p_0 to p_n, etc.) as well as the evolutionary manipulation of the underlying genotypes. Arrows depict the flow of genetic material, induced by spatial breeding operators. (b) A breeder volume that encloses several swarm grammar agents can manipulate the agents' properties and induce mutation operations. (c) Previously enclosed agents from (b) remain associated with the breeder volume. This relationship is visualized by the connecting lines.

are completely independent of each other, that is there is no interaction among the growing structures of different swarm grammars. In a co-existing and co-evolutionary setup, the encountered phenotypes can be the result of massive interactions of swarm agents. On the other hand, one can identify robust swarm grammars that generate stable phenotypes whether they are isolated or put into highly populated environments.

A. Spatial Breeding Operators

Our user interface to this immersive evolutionary scenario integrates two aspects: visual representation and intuitive manipulation by an external breeder or designer. The visualization interface enables the moving, rotating, and zooming of the camera, or the saving and restoring of specific views and scenario settings (Fig. 5). Most of these procedures are already incorporated in the agent software environment *BREVE* which we use as our display and simulation engine [49]. 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.

We facilitate the selection and manipulation of swarm individuals in three-dimensional space through breeder volumes as illustrated in Figure 4. 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.

B. The Swarm Grammar Gardener

Figure 5 illustrates how a breeder can influence the emerging building processes within a simple ecology of swarms. In Figure 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. 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. 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. 5(d,e)). This generates further expansions of the structures and—after further energy boosts (Fig. 5(f))—results in the structure depicted in Figure 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. Figure 6 gives a few examples of evolved swarm grammar ecologies and extracted structures at different stages during their evolution.

VI. Conclusion and Future Work

We presented Swarm Grammars as an extension of Lindenmayer systems. Instead of applying a single ('turtle') agent to convert linear strings into 3D structures, we use a swarm of agents which navigate in 3D space and—as a side effect—place structural building blocks into their environment. The swarm grammars are used to specify how the setup of agent types changes over time. Additional agent parameters determine the agents' behaviors and their interaction dynamics. Both the grammar rules and the agent parameters are evolvable and can change over time—either automatically at replication and collision events among the

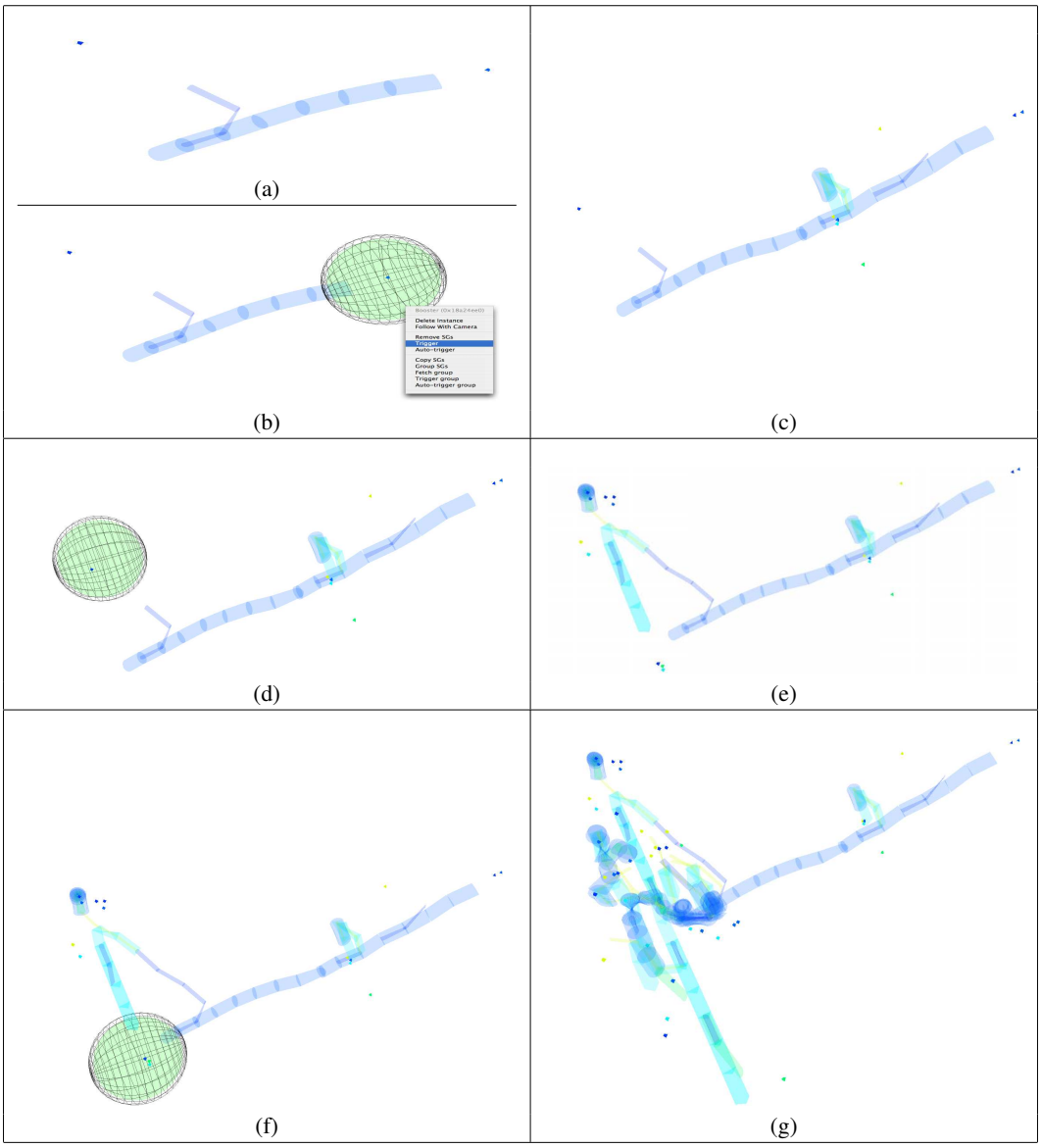


Fig. 5. Illustration of Interactive Manipulation of Swarm Grammar Agents by an External Breeder. (a) Two agents create an initial structure. (b) A breeder sphere locally infuses energy. (c) Further growth is initiated by the additional energy. (d-e) Replication of an agent triggers further parallel construction. (f-g) Expansion of the structure is continued after another energy influx.

agents, or triggered by external 'tinkering' from a supervising breeder.

Construction tasks like the ones illustrated here are, of course, only one example of what swarm agents can do, where we utilize their interaction dynamics to generate structural forms in virtual 3D space. Swarm grammar agents can also be used to investigate and evolve other dynamic processes, such as in gene regulatory processes [45], [50], immune system interactions [51], swarm behavior choreographies [19], or in interactive SwarmArt installations [52].

Further investigations into and expansions of our swarm grammar system will help us to improve the breeder user interfaces and create an efficient evolutionary design system that can be distributed onto computer clusters, which we will utilize for artistic design as well as for evolutionary exploration of swarm phenomena. Up-to-date details about swarm grammars and other agent-based simulation examples from our *Evolutionary & Swarm Design Lab* can be found at: `http://www.swarm-design.org`.

Fig. 6. Collage of Designs Generated by Swarm Grammars. The figure in the centre illustrates a swarm grammar garden ecology, within which the surrounding designs were created.

ACKNOWLEDGMENT

Financial support for this research was provided by NSERC, the Natural Sciences and Engineering Research Council of Canada.

REFERENCES

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

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

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

[4] L. Yaeger, "Computational genetics, physiology, metabolism, neural systems, learning, vision, and behavior or polyworld: Life in a new context," in *Artificial Life III*, C. G. Langton, Ed. Addison-Wesley, 1994.

[5] J. Collado-Vides, "Towards a grammatical paradigm for the study of the regulation of gene expression," in *Theoretical Biology. Epigenetic and Evolutionary Order from Complex Systems*, B. Goodwin and P. Saunders, Eds. Baltimore, ML: Johns Hopkins University Press, 1992, pp. 211–224.

[6] C. Jacob and S. von Mammen, "Swarm grammars: growing dynamic structures in 3d agent spaces," *Digital Creativity*, vol. 18, no. 1, 2007.

[7] P. Prusinkiewicz and J. Hanan, *Lindenmayer Systems, Fractals, and Plants*, ser. Lecture Notes in Biomathematics. New York: Springer, 1989, vol. 79.

[8] P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants*. New York: Springer, 1990.

[9] P. J. Bentley and S. Kumar, "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.

[10] S. Kumar and P. J. Bentley, "Implicit evolvability: An investigation into the evolvability of an embryogeny," in *GECCO 2000: Genetic and Evolutionary Computation Conference — Late Breaking Papers*, 2000.

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

[12] S. Kumar and P. Bentley, Eds., *On Growth, Form and Computers*. London: Elsevier Academic Press, 2003.

[13] H. Lipson and J. B. Pollack, "Automatic design and manufacture of robotic lifeforms," *Nature*, vol. 406, pp. 974–978, August 31 2000.

[14] J. Rieffel and J. Pollack, "Automated assembly as situated development: using artificial ontogenies to evolve buildable 3-d objects," in *GECCO '05: Genetic and evolutionary computation conference*. New York, NY, USA: ACM Press, 2005, pp. 99–106.

[15] G. Beurier, F. Michel, and J. Ferber, "Towards an evolution model of multiagent organisms," in *ECCS'05: European Conference on Complex Systems, Workshop on Multi-Agents for Modeling Complex Systems (MA4CS)*, 2005.

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

[17] V. S. Colella, E. Klopfer, and M. Resnick, *Adventures in Modeling: Exploring Complex, Dynamic Systems with StarLogo*. New York: Teachers College Press, Columbia University, 2001.

[18] V. Hartmann, "Evolving agent swarms for clustering and sorting," in *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*. New York, NY, USA: ACM Press, 2005, pp. 217–224.

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

[20] 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.

[21] G. Olague and C. Puente, "Parisian evolution with honeybees for three-dimensional reconstruction," in *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM Press, 2006, pp. 191–198.

[22] G. Beurier, F. Michel, and J. Ferber, "A morphogenesis model for multiagent embryogeny," in *Proceedings of the 10th International Conference on the Simulation and Synthesis of Living Systems (ALIFE X)*, 2006.

[23] A. Bucci and J. B. Pollack, "On identifying global optima in cooperative coevolution," in *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*. New York, NY, USA: ACM Press, 2005, pp. 539–544.

[24] K. Sims, "Interactive evolution of equations for procedural models," *The Visual Computer*, vol. 9, no. 8, pp. 466–476, 1993.

[25] J. Graf and W. Banzhaf, "Interactive evolution in simulated natural evolution," in *Artificial Evolution*, vol. LNCS 1063, 1995, pp. 259–272.

[26] ——, "An expansion operator for interactive evolution," in *Proceedings of the IEEE International Conference on Evolutionary Computation*. IEEE Press, 1995, pp. 798–802.

[27] W. Banzhaf, "Interactive evolution," in *Handbook of Evolutionary Computation*, T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds. Bristol, New York: Institute of Physics Publishing and Oxford University Press, 1997, pp. C2.9:1–6.

[28] X. Llorà, K. Sastry, F. Alías, D. E. Goldberg, and M. Welge, "Analyzing active interactive genetic algorithms using visual analytics," in *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM Press, 2006, pp. 1417–1418.

[29] P. Prusinkiewicz, M. Hammel, J. Hanan, and R. Mech, "Visual models of plant development," in *Handbook of Formal Languages*, G. Rozenberg and A. Salomaa, Eds. New York: Springer, 1997.

[30] 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.

[31] R. Mech and P. Prusinkiewicz, "Visual models of plants interacting with their environment," in *SIGGRAPH'96*. New Orleans, Louisiana: ACM SIGGRAPH, New York, 1996, pp. 397–410.

[32] M. T. Michalewicz, Ed., *Plants to Ecosystems: Advances in Computational Life Sciences*. Collingwood, VIC, Australia: CSIRO Publishing, 1997.

[33] 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*. ACM SIGGRAPH, 1998, pp. 275–286.

[34] C. Jacob, "Genetic l-system programming," in *PPSN III - Parallel Problem Solving from Nature*, ser. Lecture Notes in Computer Science, vol. 866. Jerusalem, Israel: Springer, 1994, pp. 334–343.

[35] ——, "Evolving evolution programs: Genetic programming and l-systems," in *Genetic Programming 1996: First Annual Conference*, J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. Riolo, Eds. Stanford University, Palo Alto, CA: MIT Press, Cambridge, MA, 1996, pp. 107–115.

[36] ——, "Evolution and co-evolution of developmental programs," *Computer Physics Communications, Special Issue, Modeling Collective Phenomena in the Sciences*, 1999.

[37] K. J. Mock, "Wildwood: The evolution of l-system plants for virtual environments," in *IEEE Conference on Evolutionary Computation*. Anchorage, AL: IEEE Press, New York, 1998, pp. 476–480.

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

[39] F. Michel, G. Beurier, and J. Ferber, "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)*. IEEE Press, 2005, pp. 122–128.

[40] M. Ebner, "Coevolution and the red queen effect shape virtual plants," *Genetic Programming and Evolvable Machines*, vol. 7, no. 1, pp. 103–123, 2006.

[41] G. S. Hornby and J. B. Pollack, "Evolving l-systems to generate virtual creatures," *Computers & Graphics*, vol. 25, pp. 1041–1048, 2001.

[42] ——, "Body-brain co-evolution using L-systems as a generative encoding," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, Eds. San Francisco, California, USA: Morgan Kaufmann, 7-11 2001, pp. 868–875.

[43] G. Kókai, R. Ványi, and Z. Tóth, "Parametric l-system description of the retina with combined evolutionary operators," in *Genetic and Evolutionary Computation Conference, GECCO-99*, Orlando, Florida, USA, 1999.

[44] G. Kókai, Z. Tóth, and R. Ványi, "Modelling blood vessel of the eye with parametric l-systems using evolutionary algorithms," in *Artificial Intelligence in Medicine, Proceedings of the Joint European Conference on Artificial Intelligence in Medicine and Medical Decision Making, AIMDM'99*, W. Horn, Y. Shahar, G. Lindberg, S. Andreassen, and J. C. Wyatt, Eds., vol. 1620, 1999, pp. 433–443.

[45] C. Jacob and I. Burleigh, "Biomolecular swarms: An agent-based model of the lactose operon," *Natural Computing*, vol. 3, no. 4, pp. 361–376, December 2004.

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

[47] M. Settles, P. Nathan, and T. Soule, "Breeding swarms: a new approach to recurrent neural network training," in *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*. New York, NY, USA: ACM Press, 2005, pp. 185–192.

[48] M. Settles and T. Soule, "Breeding swarms: a ga/pso hybrid," in *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*. New York, NY, USA: ACM Press, 2005, pp. 161–168.

[49] L. Spector, J. Klein, C. Perry, and M. Feinstein, "Emergence of collective behavior in evolving populations of flying agents," in *Genetic and Evolutionary Computation Conference (GECCO-2003)*, E. C.-P. e. al., Ed. Chicago, IL: Springer-Verlag, 2003, pp. 61–73.

[50] 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.

[51] 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.

[52] 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.