Evolutionary and Swarm Design in Science, Art, and Music

Christian Jacob[1,2] and Gerald Hushlak[3]

[1] Department of Computer Science, University of Calgary cjacob@ucalgary.ca
[2] Department of Biochemistry & Molecular Biology
[3] Department of Art, University of Calgary hushlak@ucalgary.ca

Evolutionary design can take many forms. In this chapter, we describe how different evolutionary techniques – such as genetic programming and evolution strategies – can be applied to a wide variety of nature-inspired designs. We will show how techniques of *interactive evolutionary breeding* can facilitate the creative processes of design. As practical examples we demonstrate how to use implicit surface modeling to create virtual sculptures, and furniture designs through evolutionary breeding.

Rather than creating variations of blueprints through an evolutionary process, we then focus on the evolution of 'design programs'. That is, instead of a static description (blueprint) of an object, we evolve recipes or algorithms to build objects. This leads to a much wider repertoir of variability on the designer's side and can be implemented in a straight-forward manner using genetic programming. Starting with a simple breeding approach of fractals, we give examples of how to – either automatically or interactively – evolve growth programs for plants with particular characteristics, which we illustrate by a garden of artificial flowers. We use evolvable Lindenmayer systems to capture growth processes.

The evolution of choreographic swarm interactions leads to new ways of 'swarm programming', where changes in control parameters result in emergent agent behaviours. We demonstrate how to use this technique to generate virtual paintings on 2D and 3D canvases. These *SwarmArt* implementations have also been exhibited in various museums as interactive computer installations, which we will use to describe how to integrate music and sound generation into evolutionary swarm systems.

1 Interactive Evolutionary Breeding

Design processes can be seen as iterative and evolutionary. As a designer one has to work within a given particular context – let's say we want to design an everyday household item, such as a new set of containers that can be used to

drink juice. Usually, we will first go through an exploratory 'brain storming' phase, during which we give ourselves the liberty to explore a larger variety of different designs. At that stage, we are not concerned about whether any of these solutions are actually feasible in the sense of whether these designs are easy to fabricate, how expensive they would be, or how practical to use. From this initial set of solutions we might get inspirations regarding specific aspects of our designs: in the case of our intended containers, for example, we might come across an unusual, but attractive shape; an elegant colour scheme; a different way of adding a handle; or an artistic, but hardly practical configuration (see Fig. 5 for a glimpse ahead).

After this inspirational phase, we might then take a more goal-oriented approach. We filter out the practically feasible solutions. We want to take specifically interesting aspects of different solutions and combine them into one design. We then work on the finer details of our designs: change colours, play with different materials, try variations in the proportions of smaller modules, etc.

Ideally, we have found a solution we are satisfied with. But normally design processes are much more time consuming, as the design cycle will start again: creating more variations from the current solutions, re-evaluating the new designs, filtering and identifying promising new design paths and ideas, and then, again, fine-tuning. Some constraints regarding the overall design might have changed on the way as well. Only certain materials are available, which could constrain the variety of shapes (e.g., containers made out of clay have more constraints than using modern, highly flexible and formable plastics), or weight and size could be a factor.

Hence, one might consider design processes to be evolutionary. Solutions have to pass the designer's fitness criteria, surviving solutions get modified – mostly slight modifications are performed, but dramatic diversions from an original solution are welcome as well, as these will act as an 'inspirational pool'. Design modifications are accomplished by variations (mutations) of current solutions or by recombinations of aspects of a number of different solutions into a new design.

With this in mind, one interesting question regarding design practice arises: How many different designs are usually considered or produced in practice? Would the inspirational brain-storming phase benefit from a much expanded pool of suggested designs than current designers do have the time to consider? Would a computer-based evolutionary design system expand a designer's possibilities to explore a wide array of solutions? Would this help to come up with unique design ideas, and then utilize the evolutionary system again to work out the details during a fine-tuning phase?

In collaboration with artists and designers, we have developed *Inspirica* [18], a programming environment and run-time evolutionary design system. We think that *Inspirica* provides enough flexibility from a programmer's perspective as well as ease-of-use from a designer's view. On the one hand, a wide range of design tasks can be tackled with the system through simple plug-in

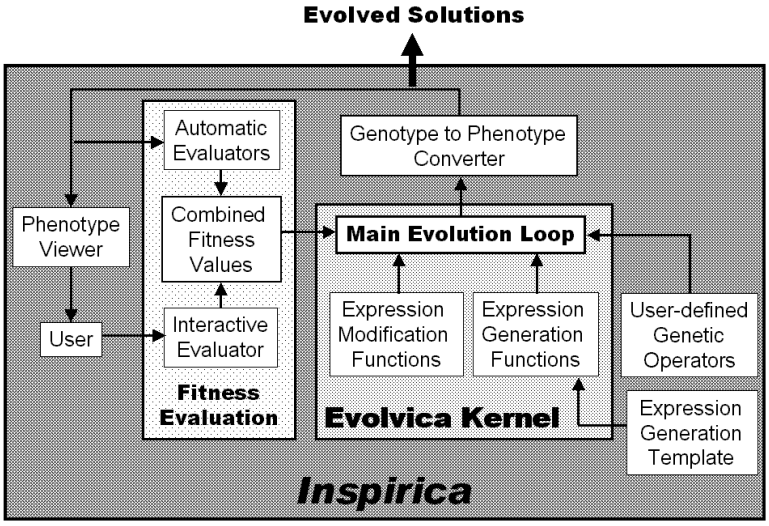


Fig. 1. Overview of the *Inspirica* system [18].

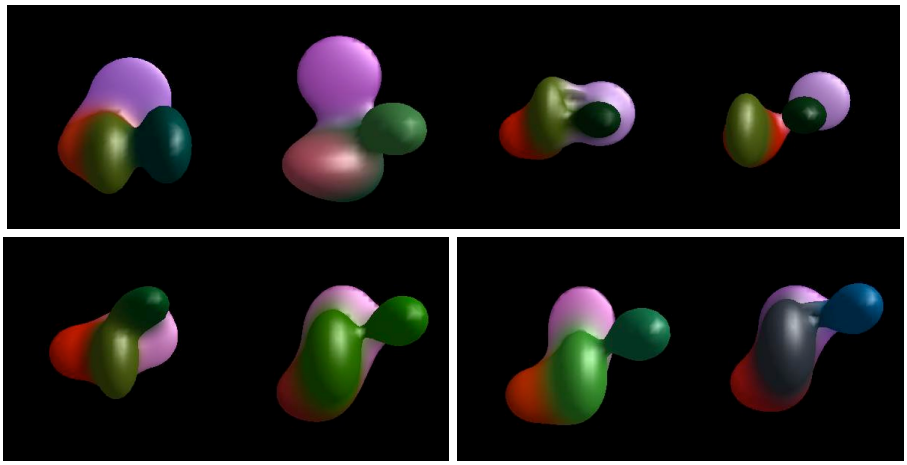
interfaces. On the other hand, the artist/designer does not need to learn how to program or manipulate computer code, in order to use the system. Figure 1 gives an overview of the *Inspirica* architecture. An evolutionary kernel (*Evolvica*) sits at the center and provides the basic functionality of a genetic programming system (solution generation, modification, and selection based on fitness) [9]. Depending on the problem domain, the user selects from a set of standard genetic operators (a number of mutation operators as well as crossover, deletion, duplication, and en-/decapsulation) or adds new, tailored operators. As usual in genetic programming, a problem specific set of building blocks (in the form of symbolic expressions) has to be defined as well. Fitness evaluation can then either be performed automatically (if a fitness function is specified) or interactively through user input. A combination of both is also possible, where the automatic fitness function acts as a pre-filter before the actual evaluation by the user. A genotype-to-phenotype converter will transform the internal design representations (based on symbolic expressions) into visualizations in the form of graphical objects or animations in 2D or 3D space.

In the following sections we will present a variety of design tasks such as the evolution of containers that can hold liquid, furniture in the form of chairs, and virtual sculptures, for all of which we have used the *Inspirica* system.

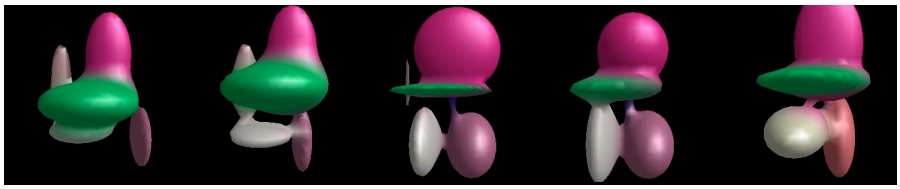
1.1 Virtual Sculptures: Evolutionary Exploration

Systems for evolving engineering designs – such as tables, stairs, heat sinks, optical prisms, boat bows and hulls, and car bodies – have been created in the

past [2, 3]. However, none have used implicit surfaces, which allow to easily model curved shapes [4, 1, 29]. Another advantage of using implicit surfaces is the ease in which one can create blended objects. Such objects are created by combining a set of skeletal primitives (points and lines) using operations such as blend, union, intersection, difference, and spatial warps [33]. A *BlobTree* is a hierarchy of nodes for primitives, operators, affine transformations, warps, and attributes [6]. The primitives provide sources of field values, similar to a magnetic or electrical field emanating from a point source. Implicit surfaces are then defined as the points in 3D space that share the same field value.



(a) Seven steps of mutation on blob sculptures.



(b) Four mutation steps.

Fig. 2. Example mutations on blob sculptures.

Figures 2 and 3 illustrate what kind of designs are typically achieved from BlobTrees[4]. Looking at the top left form in Fig. 2a, the 'sculpture' consists of four sub-components, all of which have a spherical shape. Each of these 'blobby' components is characterized by its coordinate in 3D space, its colour, and its scaling factor along the x-, y-, and z-axis. Through simple mutations

[4] See Figure 8 for an example of how these blob structures are encoded.

of these defining values one can get from one sculptural form to another, as is illustrated in Fig. 2a for seven and in Fig. 2b for four mutation steps.

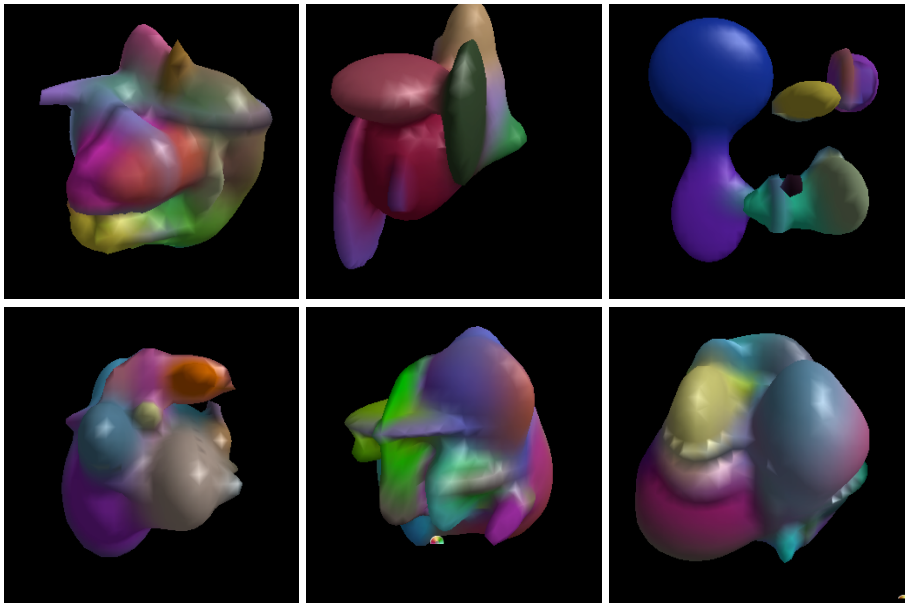


Fig. 3. Examples of evolved blob sculptures.

In addition to mutation operators, we also use recombinations, which merge aspects (colours, coordinates, shapes, and blob points) among two designs. Figure 3 gives a few examples of sculptural forms that were created through a small number (typically 10 to 20 iterations) of interactive evaluations, with a population size of six solutions to be considered at a time. At the time these experiments were conducted, we were not able to also bring these virtual sculptures to real life. Today, however, rapid prototyping printers can turn these into real sculptural forms of various materials that accurately represent the colours as well. Hence, the step from virtual designs to physical realizations has become much more feasible.

1.2 Evolving Functional Designs: Containers

Sculptures usually do not have a particular function. They are artistic objects to be exhibited, but normally are static designs. When it comes to functional design, not only aesthetics play a role, but one also has to incorporate aspects of functionality. Let us consider the design of containers that can hold fluids. If one describes the container forms through implicit surfaces, then the evolved

forms have to be evaluated with respect to their capability of holding a certain amount of fluid.

Figure 5 gives examples of such evolved containers, modeled through implicit surfaces. The actual designs were evaluated both through a manual (the designer inspecting the overall aesthetics of the generated form) and an automatic fitness function, which simulates the dripping of water onto these solutions and measures how much fluid is maintained within the structure. Figure 4 explains for the 2D case how the evaluation is performed.

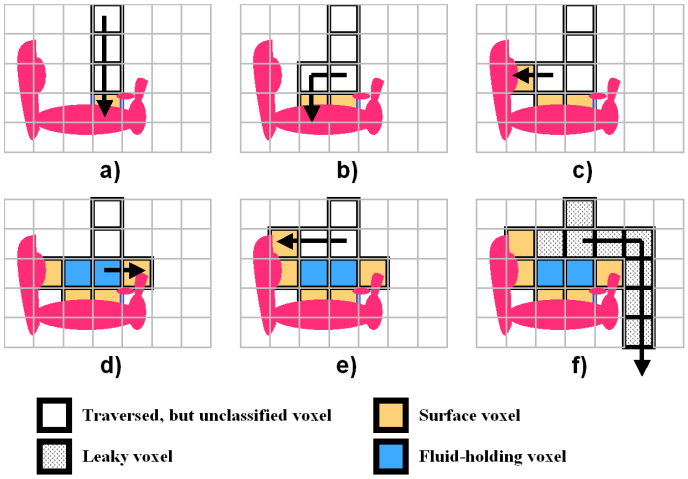


Fig. 4. How much fluid can a design hold? (a) A fluid particle drops from the top until it reaches part of the object, at which point it classifies the voxel as a surface element and returns to the previously visited voxel. (b) The particle recursively attempts to go around the object. (c) The particle is hindered on the left. (d) The particle is constrained on the right and is trapped as a result. All unclassified voxels on its lateral plane are designated as fluid-holding voxels. The particle returns to the voxel on the previous level. (e) The particle bounces off on the left. (f) The particle discovers a path to the bottom of the object's bounding box, at which point all unclassified voxels are considered as leaky [18].

The selection of evolved containers in Fig. 5 gives a sample of the design variety. All these containers can hold fluids, some are quite shallow, but have a unique design, others are combinations of two containers, or come with interesting handles. The 'goblet'-shaped solution (top right) is one example of an obviously practical design (the cup is a little shallow, but this can be rectified by simply changing one parameter by hand).

Fig. 5. A selection of evolved containers for fluids using implicit surface models.

1.3 Evolving Furniture: The Next Generation of Product Design?

Product design, however, may take a different path, where one already has a specific, well-established and well-tested design solution as a starting point. In the case of furniture design, let us look at a chair. Chairs have certain characteristics that have – in most cases – proven useful: a chair has four legs for stability, there is a surface to sit on, and a back part. In this case, it seems reasonable to start with a prototypical chair design, and then see which variations of that scheme can be achieved. This is the approach illustrated in Fig. 6, where we created an implicit surface model of a chair, for which we show different evolutionary paths over three iterations, where genetic operators of colour, scale, and position mutation are applied. Most of the interesting design outcomes in these chair evolutions are the result of mutations as exemplified in Fig. 8a, where we show the symbolic expression data structure and a set of typical mutations together with their effects on the phenotypical chair representations.

Figure 7 shows a few more examples of innovative ideas for chair designs. Here we did not add a fitness function that would filter out non-functional designs. In fact, it turns out to be rather difficult to write such an evaluation function. It is easy to check for connectivity of all components (otherwise the chair would fall apart), but it is more challenging to decide whether only four-leg designs are acceptable (some of the interesting designs do not have four legs), enforce a back part, or require other aspects of structural stability (components wide enough to support weight or make the chair stand level). Indeed, the system is more useful if less constraints are applied to the solutions, as this enables the creation of a wider variety of designs. The designer, who inspects the proposed solutions, can then easily pick out the interesting aspects of each design or dismiss some completely.

In Fig. 8b we illustrate such a case. Three chair designs are the result of taking two non-functional chairs and combining some of their features. This recombination can either be done by changing the underlying genotypical expressions (Fig. 8a) or by directly working on the phenotypical chair repre-

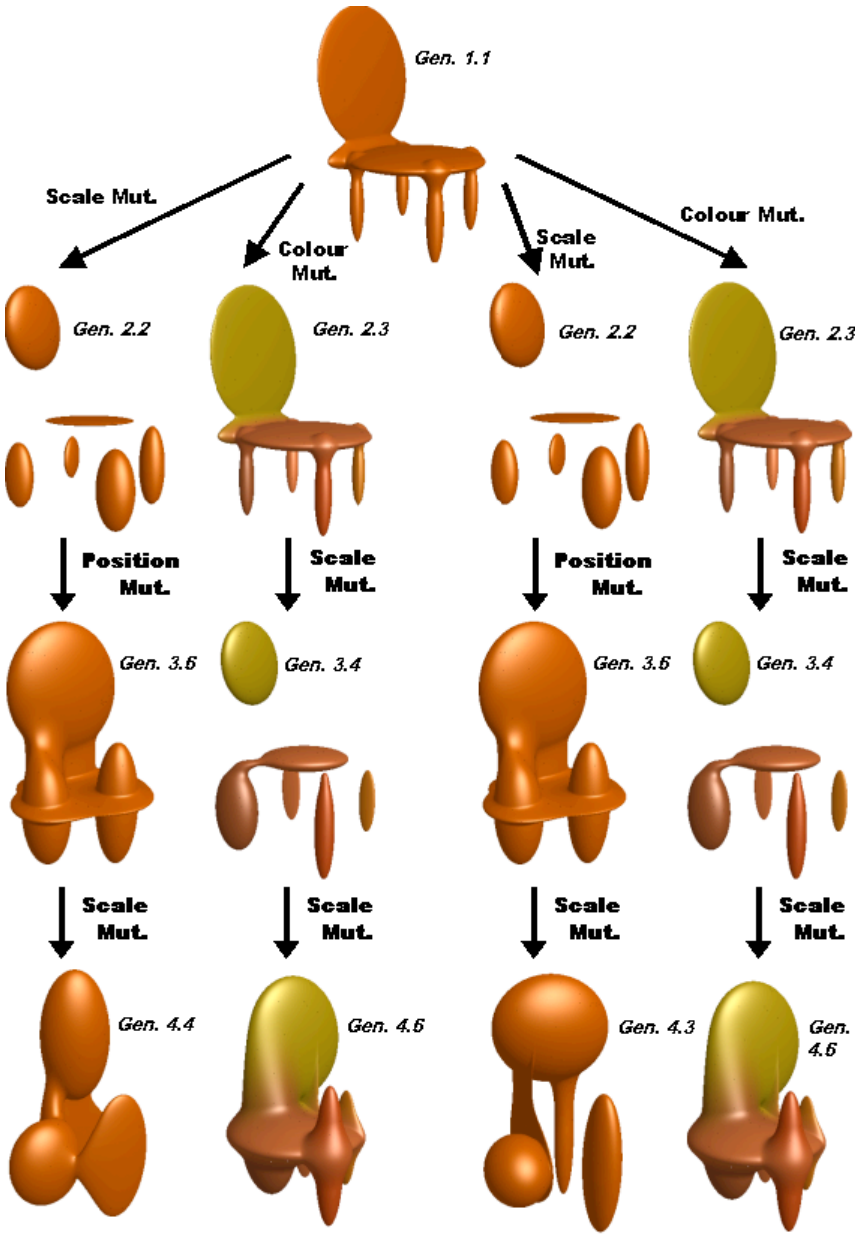


Fig. 6. Examples of evolutionary variations starting from a manually designed chair.

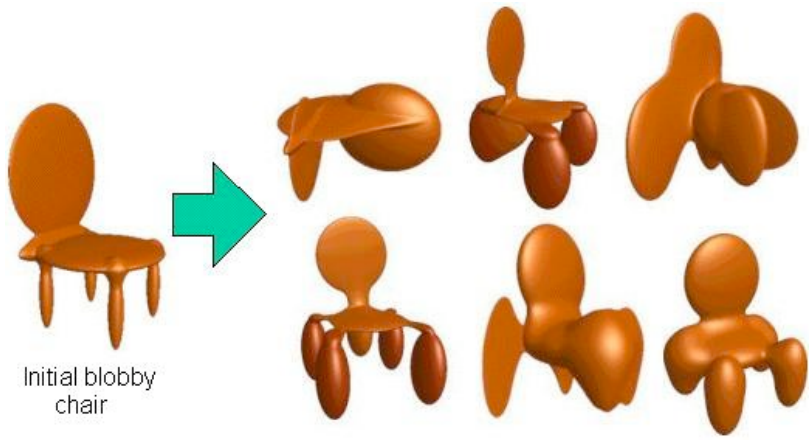
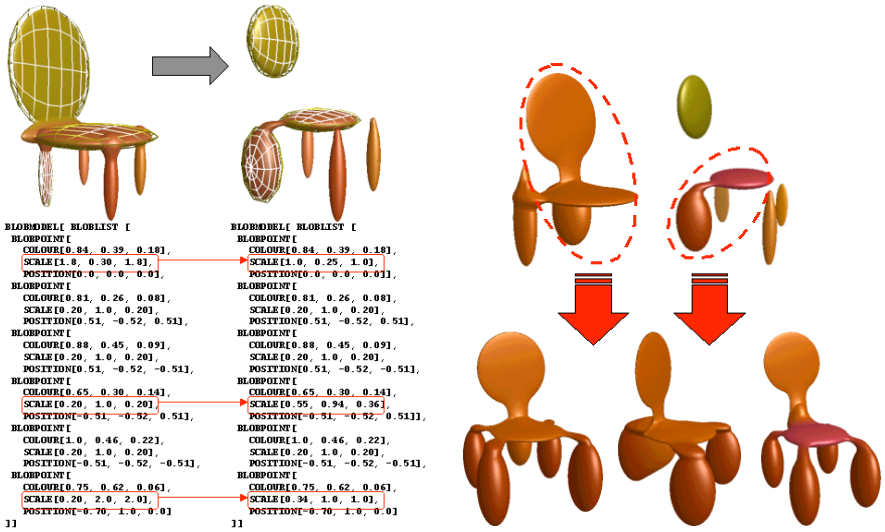


Fig. 7. Further examples of evolved Blob Furniture



(a) (b)

Fig. 8. (a) Encoding of the blob chairs and mutation effects. (b) The chair forms at the bottom are a composition of different parts (indicated by the dotted circles) from the purely evolved, non-functional chair forms above.

sentations. Of course, these 'designs' themselves can only serve as inspirations for the then following actual design process, in which the selection of material, cost, producibility etc. play essential roles. However, from our experience in working with designers and artists using our system, having an exploratory tool such as *Inspirica* at their disposal can definitely help expand their world of creativity. In many cases new tools and new technology drive innovations; this is no different for artistic design.

2 Evolution of Developmental Programs

Artistic design may be considered as a process. To a certain extent, processes can be described and captured through algorithms, procedures, or recipes. Rather than creating variations of design blueprints through an evolutionary technique (as we did in the examples discussed so far), we will now focus on the evolution of 'design programs'. That is, instead of a static description (blueprint) of an object, we evolve recipes or algorithms to build objects. The designer can therefore explore a much wider repertoir of variability. Furthermore, genetic programming techniques can be utilized for the evolution of design programs, encoded as tree-based or symbolic expressions [17, 9]. For the following examples we have used *Evolvica*, a programming environment for evolutionary optimization, design and exploration of developmental processes. *Evolvica* is described in detail in [9]; it is based on *Mathematica* [32] and can be downloaded from the *Swarm-Design.org* web site listed at the end of this chapter.

2.1 Evolution of Fractals: Advantages of Rewriting Systems

In order to illustrate the power of design programs to specify models of developmental processes, we choose a simple version of so-called Lindenmayer systems (L-systems) as examples of rewriting systems that have been successfully used for specifying a wide range of growth processes, mainly in the area of botanical branching structures [20, 25, 26, 9].

The bottom of Fig. 9 shows a simple rewrite system. Starting from an *axiom* of `FLFLFLF`. We iteratively apply the replacement rule

$$\texttt{F} \longrightarrow \texttt{FLFRFRFFLFLFRF},$$

which means that each symbol `F` in the axiom is replaced by the symbols on the right-hand side of the rule. We can then repeat this rewriting on the resulting strings several times, which creates a larger and larger, recursively constructed string. Each symbol `F`, `L`, and `R` represents a command for a drawing device, called a *turtle*: move forward, turn left, and turn right (90 degrees), respectively. This is a straight-forward translation from a one-dimensional string into a 2-dimensional graphical representation. Now imagine we create an evolutionary system (in fact, we used a pre-cursor of *Inspirica*) that can mutate

both the axiom and the replacement rule [34]. What kind of 2-dimensional structures are we going to get? Figure 9 gives a few examples. These fractal structures are the result of a few evolution runs, with interactive selection where we evaluated the aesthetics of the fractals. Starting from an axiom, each associated replacement rule was applied five times, which creates truly recursive structures. The designs range from symmetric to curly line-like structures, from geometric (squared and circular) to more natural looking shapes, and from intricately detailed to rather straight-forward fractal patterns.

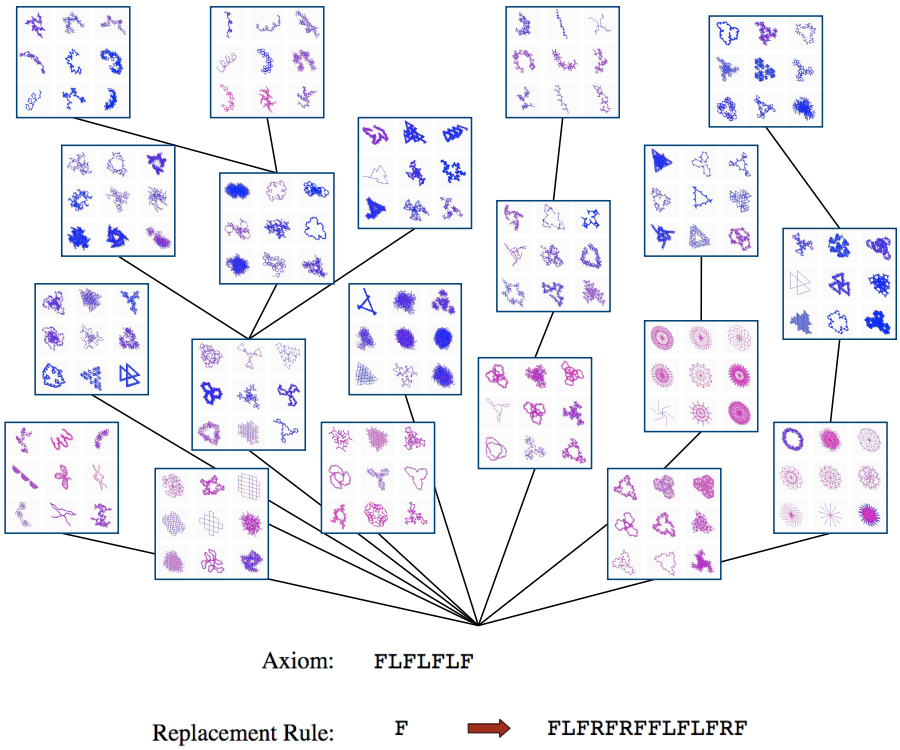


Fig. 9. Fractal structures evolved from an axiom and a single replacement rule.

2.2 Evolving ArtFlowers: A Variety of Botanical Structures

The same principles apply when we look at rewrite systems (L-systems) to which we apply a turtle interpretation in three dimensions. Now commands like pitch, yaw, and roll – similar to maneuvering an airplane in 3D space – come into play. In order to create plant-like structures, we can also add macro commands to add flowers, leaves, and other attributes that relate to plant modules (e.g., thickness, colour, and texture of stem sections).

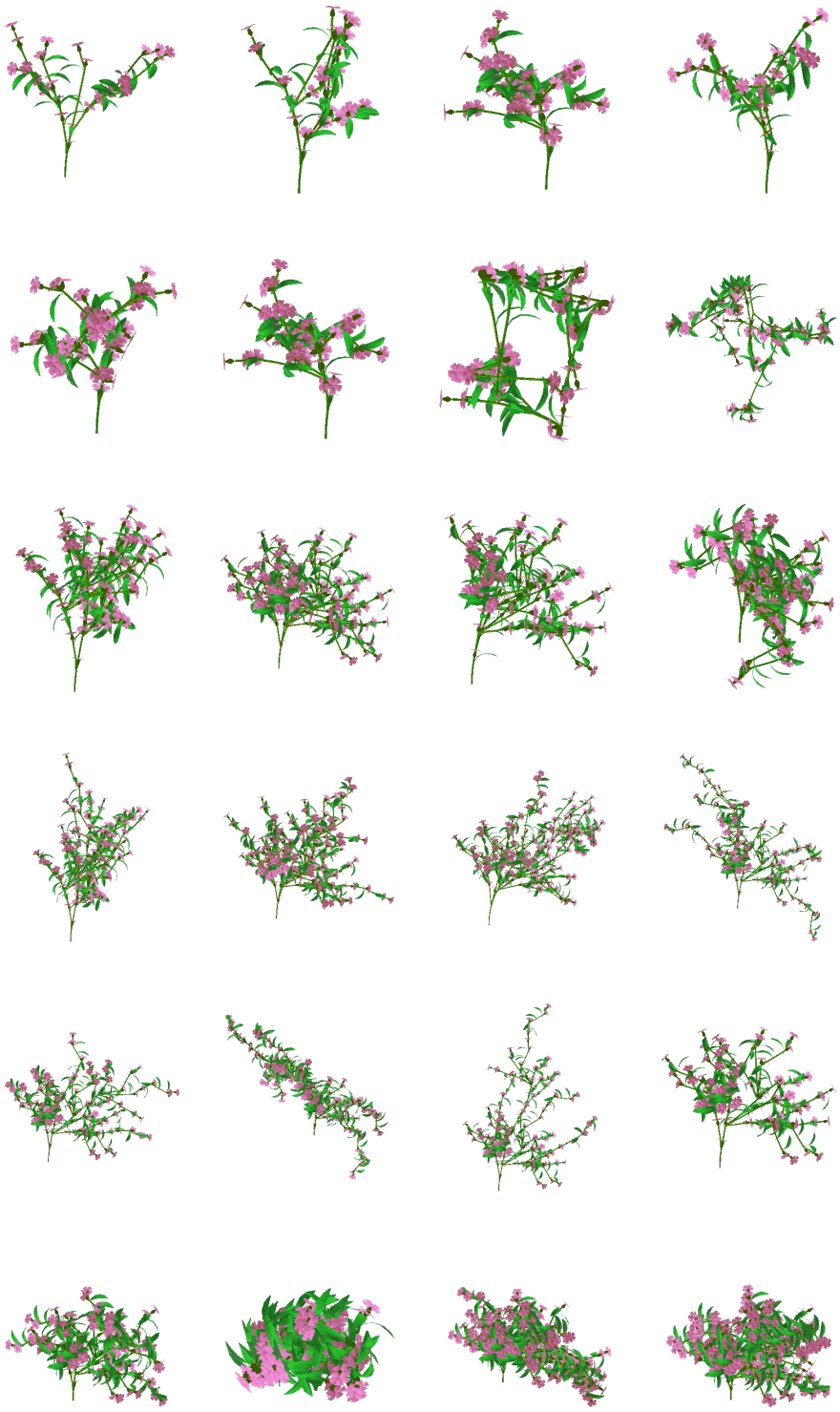


Fig. 10. A selection of evolved ArtFlowers. All L-systems are interpreted and visualized using CPFG [26].

Figure 10 shows a collection of evolved blooming plants. The plant structure on the top-left is a model of the rose champion *Lychnis coronaria*. All other plants were evolved from this *Lychnis* L-system, similar to the chair evolution as discussed previously. Fitness evaluation is performed both automatically and by visual inspection [9]. The automatic fitness function measures the size of the plant (in x-, y- and z-direction) and counts its leaves and blooms. The more leaves and blooms and the more expansive a plant is, the higher its fitness. The second row in Fig. 10 shows some strange growing patterns, which are nevertheless also found in nature. The third row plants carry considerably more flowers. The next two rows are plants that were tuned to grow high and wide, instead of deep. Hence, they are similar to wall-climbing plants. The last row of plants is particularly interesting as these have a 'bushy' appearance. This is a typical example where it is not obvious at all how to incorporate certain characteristics of plants, such as 'bushiness'. Only after evolving examples of bush-like structures one is able to analyze what makes a plant look bushy. It turns out that this appearance depends not only on a single variable or replacement rule. It is a combination of various rules and parameter settings, as, for example, rather large leaves combined with a higher degree of branching. But all these factors are relative to the overall proportion of the plant. Hence, there are some great lessons to learn, regarding features that are obvious to any human, but are extremely hard to incorporate into an automatic design system. Consequently, we now see the human designer as an essential part of any evolutionary design process; atrributes like 'bushiness' are easy to detect by human observers, and the combination of interactive evaluation and automatic fitness calculations makes an evolutionary system even more powerful and useful. The next section will give yet another example of this human-enhanced approach towards evolutionary design.

3 Evolved Swarm Choreography

Let us, once again, expand our view of the systems we want to design. In the last section we introduced growth programs to capture the actual process of developing an object. Many patterns and morphogenetic (structure forming) processes in nature are the emergent result of interactions among a relatively large number of 'agents'. Take a flock of birds, for example. From a single bird's perspective, we can define rules of interaction with other birds in its immediate neighbourhood. Following Craig Reynold's 'Boids' model [27], a bird agent should have rules for *cohesion* (the tendency to keep close to other flock mates), *separation* (avoiding collisions with other birds), and *alignment* (moving in the average direction and with a similar speed to neighbouring birds). These rules are already enough to realistically simulate bird flocking patterns. The same principle applies to other 'swarm intelligence' systems, such as ant colonies [28], pedestrian and vehicular traffic [7, 23], and interactions among bio-molecules [11, 10], cells [14], and bacterial colonies [24, 8].

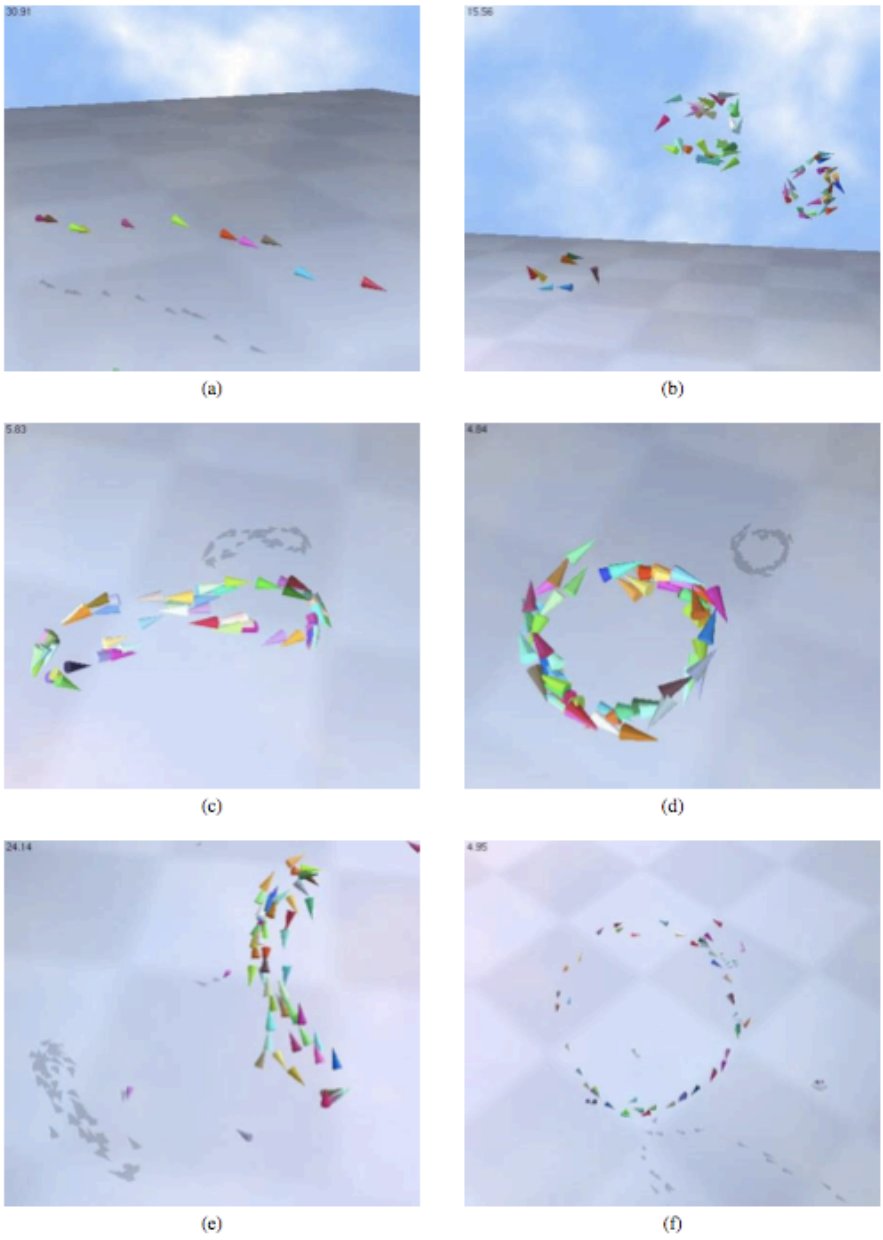


Fig. 11. (a) A simple evolved line formation. (b) Evolved loose swirling ring formations. (c) A figure-eight formation that evolved as a recombination of (a) and (b). (d) A ring formation that evolved from (a) and (b). In this formation, the agents fly in both clockwise and counterclockwise directions, like a figure-eight formation where the two end-loops have folded in on one another. Note that the shape of these ring formations are more defined than those in (b). (e) A messy figure-eight that evolved from (c). The figure-eight shape is not as well defined as the one in (c). (f) A larger ring that evolved from (d) with agents more spread out.

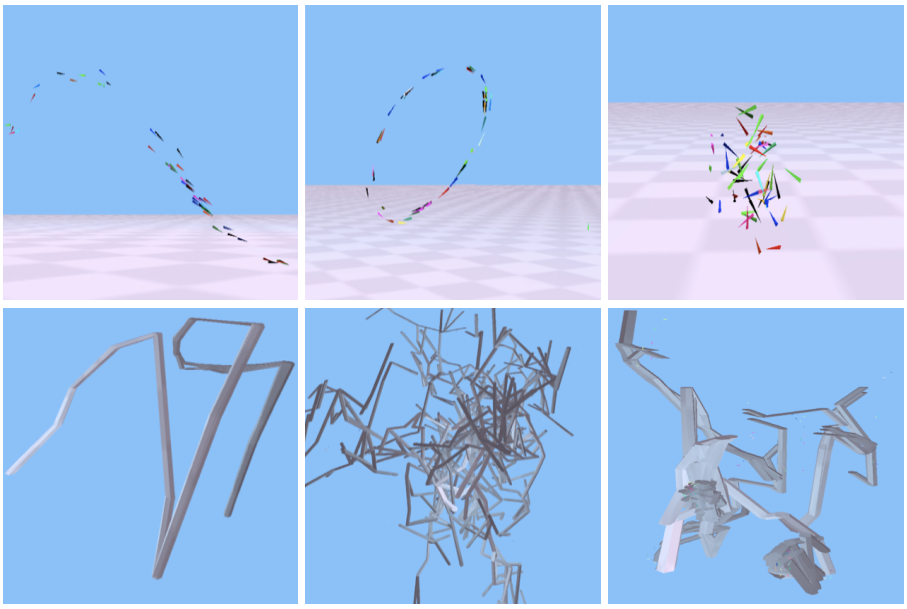


Fig. 12. Choreographed swarm behaviours and their corresponding swarm grammar representations [31, 16].

Simulating swarm intelligence systems, an interesting question arises: To what extent can global properties – such as birds flying in a particular arrangement (e.g., V-shape) – be controlled either through the agent interaction rules or through fine tuning of simulation control parameters? Can one use an evolutionary approach to breed or design choreographed swarm behaviours?

This is what we were trying to explore with the following experiments. Figure 11 shows a selection of examples of evolved swarm choreographies of bird-like agents flying through a 3-dimensional world. Using an interactive evolutionary approach, we were able to breed specific flight patterns, such as line, figure-eight or ring formations. In this case, only swarm control parameters were manipulated by the evolutionary system (for details see [19, 5]).

Figure 12 shows an artistic approach to these choreographed swarms, where we applied concurrent turtle interpretations through swarm grammars [31], which are a swarm-based extension of the rewrite systems discussed in Section 2. This results in artistic arrangements representing the areas covered by the swarm agents. In this case, the swarm builder agents leave trails of cylindrical elements within the 3D canvas; the resulting sculptures represent static snapshots of the dynamic construction and interaction processes among the swarm particles. This system relies on a swarm-based variation of the single-turtle interpretation used in L-systems and is described in detail in [16].

There is another interesting aspect to this 'genetic swarm programming' approach. Imagine that we have constructed a simulation of bird flocking, as described above. A biologist, interested in social insects, might want to study the behaviours of bees. Can the bird model help to develop a bees model? Intuitively, it can. Let us assume that an expert in social insects would use the evolutionary breeder system, starting from the bird simulations, and interactively evaluate the population of simulations on the computer screen. Looking for specific aspects in the flight and interaction patterns that distinguish bees from birds would then guide the step-by-step evolutionary selection process towards more bee-like behaviours. This can be accomplished by the biologist acting as a 'swarm designer' — without any programming knowledge.

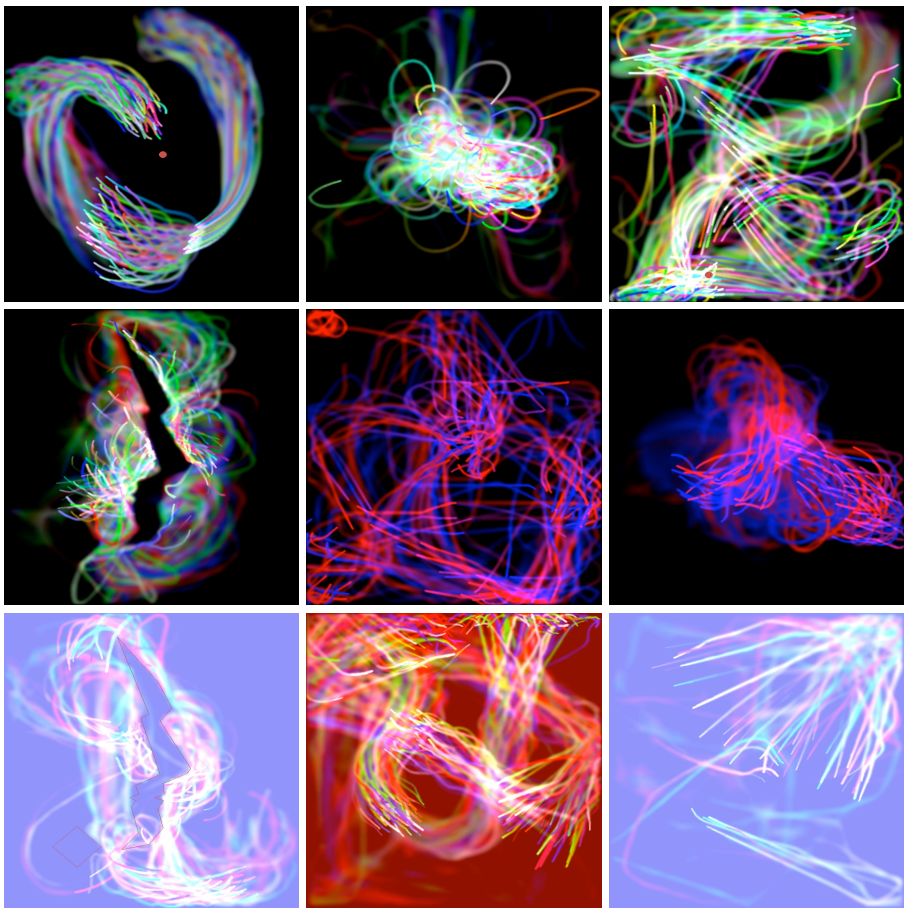


Fig. 13. BoidLand swarm drawings by *SwarmArt.com*.

4 Evolutionary SwarmArt Installations

Our team is enjoying a long-term collaboration between the Faculty of Fine Arts (Department of Art), the Faculty of Science (Department of Computer Science) and the Faculty of Medicine (Department of Biochemistry & Molecular Biology). From our inter-disciplinary exchanges we have learnt how to form close connections between art and science. In our case, this is mainly facilitated by the use of state-of-the-art techniques and methodologies in computer science and engineering, such as multi-media processing, video and image manipulation, computer graphics, computer-aided design, and collective intelligence algorithms. Our investigations of biological and medical systems through simulations and computer modeling [24, 14, 11, 12, 10, 15] provide an ideal vehicle for incorporation of scientific concepts and results into artistic computer exhibitions. This, in turn, leads to a welcome side effect: many visitors to an art gallery become indirectly engaged in scientific explorations, simply by playfully interacting with our *SwarmArt* installations (`www.swarmart.com`). We will briefly describe some of our installations in this section (for more details see [5, 21, 13]).

4.1 Creative Interaction with Swarm Art

The swarm drawing examples in Fig. 13 illustrate a similar point. Here the swarm agents follow a user-directed cursor (red dot), which can influence the movement of the swarm. Each agent has a particular colour assigned and leaves a trail behind while it is moving through the 2D- or 3D-canvas space. Over time, the trails are fading, which introduces depth and a sense of temporal dynamics into the pictures. Any obstacle, such as the outline of a specific shape, can be put inside the simulation space, with the agents floating around it. Several versions of this swarm drawing interface were installed in the Nickle Arts Museum in Calgary; the evolution of this and other *SwarmArt* installations is described in more detail in [5].

4.2 The Sounds of Swarms

Creating a seamless interface for controlling swarm behaviours, especially in a museum or gallery setting, is crucial as complicated technical setups are hard to maintain by museum personnel. Visitors should be able to interact with the installation without having to use specific devices, such as a mouse or sensor gloves. We therefore opted for a video-based interface, as illustrated in Fig. 14. This video interface was implemented for a museum installation in 2005 that incorporated a set of swarms which would move over live video streams taken within the exhibition space. An iSight camera is connected to an Apple Mac mini^{TM} computer. Subsequent image frames from the camera are analyzed to identify any moving objects (similar to [30, 22]). Visitors entering the space would activate the movement detector. Depending on where the movements

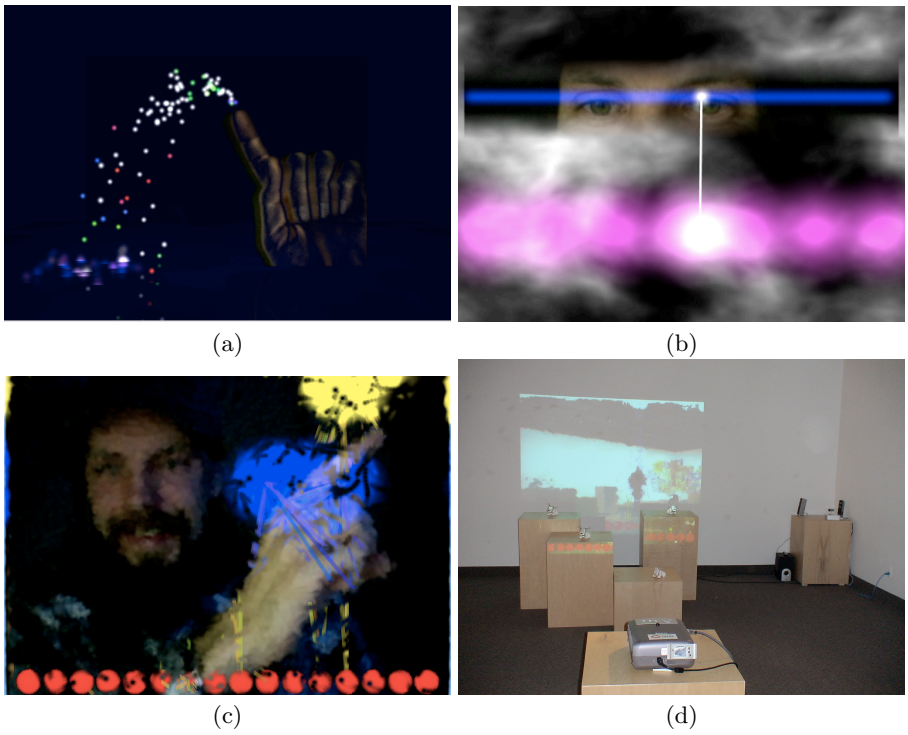


Fig. 14. Controlling swarms through a video interface. (a) A swarm of particles follows the movements of the finger tip. (b) Blinking with one eye triggers an action; the bright spot represents a musical instrument that plays a particular key. (c) Larger movements, such as moving a hand, identifies movement regions; the video image is subdivided into areas within which instruments are triggered. The bottom row of red bulbs represents a percussion set. The middle area plays strings (blue), whereas the top area triggers piano sounds (yellow). (d) The actual setup of the installation in the Nickle Arts Museum, Calgary.

are identified within the video image, different musical instruments would start to play (keyboard, piano, choir, drum set, etc). As the swarms that move over the image also trigger instruments, the viewer establishes an immediate musical communication with the swarm system. The user acts as a composer, who, however, is not in complete control of the actual composition, but rather influences the evolutionary musical paths that the swarms decide to follow.

Figure 15 shows the installation in action. Several visitors are interacting with the system and triggering different tunes by entering the camera space. Any movement detected by the camera is projected into the swarm simulation space, where the agents gravitate towards the coordinates of the detected motion points.

Fig. 15. Close-up of the 2005 SwarmMusic installation at the Nickle Arts Museum in Calgary.

As an additional element, we also included i-dogs^{TM}, three of which are sitting on the boxes in front of the projection screen. These little robot dogs have built-in microphones; depending on the musical patterns they receive, the dogs start moving their heads and ears. Having the i-dogs sit in front of the camera creates an interesting feedback loop: the swarms play musical tunes, the i-dogs listen to the music and start to move after a short while; these movements, in turn, trigger more music. Hence, the i-dogs have started and will maintain an interaction cycle — without any human intervention.

5 Conclusion

By example of our *Inspirica* system we have demonstrated how automatic evolutionary design in combination with interactive selection by a human designer can provide a wider variety of design solutions (blob sculptures, chairs, and containers). As an expansion to static design blueprints, using developmental approaches can help to capture dynamic processes that lead to designed outcomes. We have shown two examples of these emergent designs (fractals and plants) generated through genetic L-system programming. Adding swarms of agents to a virtual design space expands the range of possible structures even further. However, keeping in tight control of these swarm interaction

processes is more challenging, as we have illustrated through the evolution of choreographed swarm behaviours and their swarm grammar expansion. Applying these nature-inspired design principles in artistic settings and bringing them to life in art gallery exhibitions is highly gratifying and establishes a fruitful interaction between science, art, and an engaged audience.

As video-based user interfaces will become more and more easy to use, the incorporation of audience interactions into evolutionary design processes will be a further expansion to explore. This would truly utilize the 'wisdom of crowds' through collective intelligence [**?**], which we have already started to explore [21]. Future evolutionary design systems will become hybrids of interactive, human-directed evolution and design 'ecologies', where design solutions compete for survival in an ecosystem that implicitly defines the constraints of the design spaces. In our *Evolutionary & Swarm Design Lab* we will use swarm grammars to further explore this avenue of emergent design ecologies.

The examples discussed in this chapter as well as further material, such as videos and sample source code, are available online at:

`http://www.swarm-design.org` and `http://www.swarmart.com`.

Acknowledgements

The described projects were funded by NSERC, the Natural Sciences and Engineering Research Council of Canada, and the Canada Council for the Arts. We thank the following students and colleagues at the University of Calgary for their contributions to these projects: Euan Forrester (BoidLand), Henry Kwong (Inspirica), Sebastian von Mammen (Swarm Grammars), Scott Novakowski (musical video swarms), Dr. Przemyslaw Prusinkiewicz (CPFG), Ryan Schmidt (Blob Sculptures), Dr. Brian Wyvill (BlobTree), and Jing Yu (evolved fractals).

References

1. E. Bedwell and D. Ebert. Artificial evolution of implicit surfaces. In *ACM SIGGRAPH Technical Sketch*, July 1998.
2. Peter Bentley. From coffee tables to hospitals: Generic evolutionary design. In Peter Bentley, editor, *Evolutionary Design by Computers*. Morgan Kaufmann, San Francisco, CA, 1999.
3. Peter Bentley. *Generic Evolutionary Design of Solid Objects using a Genetic Algorithm*. PhD thesis, University of Huddersfield, Queensgate, Huddersfield, UK, 2001.
4. J. Bloomenthal, C. Bajaj, J. Blinn, M. Cani-Gasuel, A. Rockwook, and B. Wyvill. *Introduction to Implicit Surfaces*. Morgan Kaufmann, San Francisco, CA, 1998.

5. Jeffrey Boyd, Gerald Hushlak, and Christian Jacob. Swarmart: Interactive art from swarm intelligence. In *ACM Multimedia*. ACM Multimedia, ACM, October 2004.
6. M. Fox. Animated blobtrees for the impatient. Master's thesis, University of Calgary, Calgary, AB, Canada, 2001.
7. Ricardo Hoar, Joanne Penner, and Christian Jacob. Evolutionary swarm traffic: If ant roads had traffic lights. In *IEEE World Congress on Computational Intelligence*, Honolulu, Hawaii, 2002. IEEE Press. VirtualBacteria-Refs (2004).
8. Ricardo Hoar, Joanne Penner, and Christian Jacob. Transcription and evolution of a virtual bacteria culture. In *IEEE Congress on Evolutionary Computation*, Canberra, Australia, 2003. IEEE Press. VirtualBacteria-Refs (2004).
9. C. Jacob. *Illustrating Evolutionary Computation with Mathematica*. Morgan Kaufmann Publishers, 2001.
10. Christian Jacob, Anna Barbasiewicz, and Glorious Tsui. Swarms and genes: Exploring λ-switch gene regulation through swarm intelligence. In *CEC 2006, IEEE Congress on Evolutionary Computation*, 2006.
11. Christian Jacob and Ian Burleigh. Biomolecular swarms: An agent-based model of the lactose operon. *Natural Computing*, 3(4):361–376, December 2004.
12. Christian Jacob and Ian Burleigh. Genetic programming inside a cell. In Tina Yu, Rick L. Riolo, and Bill Worzel, editors, *Genetic Programming Theory and Practice III*. Springer Verlag, 2005.
13. Christian Jacob, Gerald Hushlak, Jeffrey Boyd, Paul Nuytten, and Maxwell Sayles. Swarmart: Interactive art from swarm intelligence. *Leonardo (to appear)*, 2007.
14. Christian Jacob, Julius Litorco, and Leo Lee. Immunity through swarms: Agent-based simulations of the human immune system. In *Artificial Immune Systems, ICARIS 2004, Third International Conference*, Catania, Italy, 2004. LNCS 3239, Springer.
15. Christian Jacob, Scott Steil, and Karel Bergmann. The swarming body: Simulating the decentralized defenses of immunity. In *Artificial Immune Systems, ICARIS 2006, 5th International Conference*, Oeiras, Portugal, September 2006. Springer.
16. Christian Jacob and Sebastian von Mammen. Swarm grammars: growing dynamic structures in 3d agent spaces. *Digital Creativity*, 2007 (submitted).
17. J. R. Koza, M. A. Keane, M. J Streeter, W. Mydlowec, J. Yu, and G. Lanza. *Genetic Programming IV – Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers, Norwell, MA, 2003.
18. Henry Kwong. Evolutionary design of implicit surfaces and swarm dynamics. Master's thesis, Department of Computer Science, University of Calgary, Calgary, AB, Canada, June 2003.
19. Henry Kwong and Christian Jacob. Evolutionary exploration of dynamic swarm behaviour. In *Congress on Evolutionary Computation*, Canberra, Australia, 2003. IEEE Press. emergence.
20. Aristid Lindenmayer. Mathematical models for cellular interaction in development, parts i and ii. *Journal of Theoretical Biology*, 18:280–315, 1968. reference from Principia Evolvica (1998),.
21. Quoc Nguyen, Scott Novakowski, Jeffrey Boyd, Christian Jacob, and Gerald Hushlak. Motion swarms: Video interaction for art in complex environments. In *ACM Multimedia*. ACM, ACM, October 2006.

22. Scott Novakowski. Interactive swarm music. CPSC 503 Project Report, December 2005.
23. Joanne Penner, Ricardo Hoar, and Christian Jacob. Swarm-based traffic simulation with evolutionary traffic light adaptation. In L. Ubertini, editor, *Applied Simulation and Modelling*, International Association of Science and Technology for Development - IASTED, pages 289–294, Crete, Greece, 2002. ACTA Press, Zurich. Proceedings of the IASTED International Conference.
24. Joanne Penner, Ricardo Hoar, and Christian Jacob. Bacterial chemotaxis in silico. In *ACAL 2003, First Australian Conference on Artificial Life*, Canberra, Australia, 2003.
25. P. Prusinkiewicz and J. Hanan. *Lindenmayer Systems, Fractals, and Plants*, volume 79 of *Lecture Notes in Biomathematics*. Springer, New York, 1989. reference from Principia Evolvica (1998),.
26. P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer, New York, 1990. reference from Principia Evolvica (1998),.
27. Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Int. Conference on Computer Graphics and Interactive Techniques, SIGGRAPH*, pages 25–34. ACM, 1987.
28. Garret Suen. Modelling and simulating army ant raids. Master's thesis, University of Calgary, Dept. of Computer Science, Calgary, Canada, April 2004.
29. M. Tigges and B. Wyvill. Python for scene and model description for computer graphics. In *Proc. IPC 2000*, January 2000.
30. Tatsuo Unemi and Daniel Bisig. Playing music by conducting boid agents. In J. et al. Pollack, editor, *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems*, pages 546–550, Boston, MA, 2004. MIT Press.
31. Sebastian von Mammen. Swarm grammars. a new approach to dynamic growth. Technical Report 2006-835-28, Department of Computer Science, University of Calgary, Calgary, AB, Canada, 2006.
32. Stephen Wolfram. *The Mathematica Book*. Cambridge University Press, Cambridge, MA, 3 edition, 1996.
33. B. Wyvill, E. Galin, and A. Guy. Extending the csg tree. warping, blending and boolean operations in an implicit surface modeling system. *Computer Graphics Forum*, 18(2):149–158, June 1999.
34. Jing Yu. Evolutionary design of 2d fractals and 3d plant structures for computer graphics. Master's thesis, Department of Computer Science, University of Calgary, 2004.