Evolutionary Exploration of Dynamic Swarm Behaviour

Henry Kwong
Evolutionary & Swarm Design Group
Dept. of Computer Science
University of Calgary
2500 University Drive NW
Calgary, AB, Canada T2N1N4
kwongh@cpsc.ucalgary.ca

Christian Jacob
Evolutionary & Swarm Design Group
Dept. of Computer Science
University of Calgary
2500 University Drive NW
Calgary, AB, Canada T2N1N4
jacob@cpsc.ucalgary.ca

Abstract- Dynamic swarm behaviour is difficult to design manually. Multiple, simultaneous interactions among a large number of agents, make the relationship between a parameter change and the corresponding change in global behaviour not obvious. This paper presents breeding experiments of dynamic swarm behaviour patterns using an interactive evolutionary algorithm. Specifically, eight scalar parameters that influence swarm behaviour dynamics in a 3D swarm simulation are interactively evolved to produce agents that collectively fly in line, ring, and figure-eight formations.

1 Introduction

The power of the unaided mind is highly overrated. Without external aids, memory, thought, and reasoning are all constrained. But human intelligence is highly flexible and adaptive, superb at inventing procedures and objects that overcome its own limits.

— Donald A. Norman [Nor93], p. 43

This quote elegantly and boldly states the vital, often overlooked, role external aids (such as pencil and paper, calculators, computers, etc.) play in human cognition. We will present such an external aid as an evolution-based tool to 'program' swarm behaviours through interactive breeding. Evolutionary algorithms have been successfully applied to problem domains where the resulting complex solutions are often difficult to comprehend or counter-intuitive, and consequently have resisted discovery by human reasoning. Such problem domains include the construction of neural networks [Sim94, SM02, Gru95], cellular automaton rules [Sip97, dG92], quantum algorithms [SBBS99], and novel electronic circuits [KAea99]. Designing swarm dynamics is another such problem domain.

In [Kwo03], a flexible evolutionary design system called *Inspirica* is used to investigate the potential for exploring complex design spaces. *Inspirica* is an extension of *Evolvica* [Jac01] for automatically and interactively evolving solutions—in the form of computer programs encoded as symbolic expressions—from a variety of static and dynamic domains. These domains include implicit surface (soft object) models of fluid containers, chairs, and facemasks [BBB+98], and swarm behaviour dynamics in 3-dimensional space.

This paper demonstrates how evolutionary algorithms can be utilized to explore complex pattern formations of swarm systems in 3D space, which exhibit a high level of self-organization. It is often challenging to predict the resulting swarm behaviours from a set of control parameters without actually running the simulation. The stochastic, distributed nature of the multiple, simultaneous interactions among a large number of agents make the relationship between a parameter change and the corresponding change in the global behaviour dynamics difficult to assess. For the same reason, it is particularly difficult to manually craft a set of parameters such that a particular behaviour pattern 'emerges.'

However, evolutionary algorithms are an appropriate choice to explore swarm behaviours, because evolution-based approaches have been effective in other computationally irreducible systems, such as cellular automata [Sip97], Lindenmayer systems [Jac96, Koz93] or metabolic pathways [KML+00]. Without going through their actual execution, these systems often defy any predictability of how they will behave [Wol01].

This paper is organized as follows. Section 2 presents the swarm simulation used to visualize the agent behaviours. Section 3 explains the genetic operators, and the representation of the control parameters (genotypes) that produce a variety of swarming behaviours. Section 4 describes the evolution and discovery of various swarming patterns. Finally, Section 5 highlights future work for this research.

2 The Swarm Simulation

For our evolutionary swarm experiments we use BREVE, a simulation package designed for the modeling of decentralized systems and artificial life in both continuous time and continuous, 3-dimensional space [SK02]. The BREVE swarm simulation we use is an implementation of Craig Reynolds' classic "boids" model [Rey91]. Our simulation is typically composed of 50 individual agents that fly through a virtual 3D world (Figure 1). The agents are confined within a bounded volume of space, and steer away from the boundaries when they get too close. Agents can collide with one another and the ground plane, on which they can land.

The local environment and each agent's behavioural parameters determine the direction and speed of their flights. An agent's instantaneous acceleration vector $\vec{A}$ at each time step of the simulation is calculated as follows [SK02]:

$$\vec{A} = A_{max}\frac{\vec{V}}{|\vec{V}|}$$

$$\vec{V} = c_1\vec{V}_1(d) + c_2\vec{V}_2 + c_3\vec{V}_3 + c_4\vec{V}_4 + c_5\vec{V}_5$$

where

c_1 to c_5	are the weightings for the corresponding 'urge' vectors ($\vec{V}_1$ to $\vec{V}_5$) described below.
d	is the preferred distance an agent tries to maintain with its neighbours (crowding radius),
$\vec{V}_1$	is a vector that points away from neighbours that are within d,
$\vec{V}_2$	is a vector toward the center of the world,
$\vec{V}_3$	is the average of the agent's neighbours' velocity vectors,
$\vec{V}_4$	is a vector toward the center of gravity of the agent's neighbours
$\vec{V}_5$	is a random unit-length vector,
$\vec{V}$	is the weighted summation of behavioural urges,
$\vec{A}$	is an agent's instantaneous acceleration vector at each time step of the swarm simulation with $|\vec{A}| \leq A_{max}$,
A_{max}	is the maximum acceleration of an agent.

Any of these listed parameters influence the emergent dynamics of the simulation. Not all parameters, however, affect the behaviours at the same scale, as we are going to demonstrate in Section 4. For example, the crowding radius (d) has a more immediate influence on the overall dynamics than the agents' maximum acceleration (A_{max}).

3 Representation and Genetic Operators

The genotype (behavioural parameters) consists of a fixed-length list containing the weightings (c_1 to c_5), the crowding radius (d), the maximum acceleration (A_{max}), and the maximum velocity of the agents (V_{max}). Hence, a 'chromosome' $\bar{g}$ has the form $\bar{g} = (c_1, ..., c_5, d, A_{max}, V_{max})$.

In our swarm evolution, *mutation* occurs by first choosing a genotype $\bar{g}_a$ from the current population with fitness proportionate selection [Jac01]:

$$\bar{g}_a = (x_1, x_2, ..., x_i, ..., x_8).$$

A random parameter $x_i, i \in \{1, ..., 8\}$ is then chosen, and a random value δ, following a uniform distribution within a user-defined range, is added or subtracted to this parameter to produce a new individual $\bar{g}'_a$.

$$x'_i = \min(max_{x_i}, \max(min_{x_i}, x_i \pm \delta))$$

$$\bar{g}'_a = (x_1, x_2, ..., x'_i, ..., x_8)$$

Note that min_{x_i} and max_{x_i} are the minimum and maximum allowable values for parameter x_i, respectively. The user may choose to repeat this process a number of times to mutate several parameters in one mutation operation.

As a *recombination* operator we use one-point crossover among two individuals $\bar{g}_a$ and $\bar{g}_b$ from the population using fitness proportionate selection, where $a \neq b$.

$$\bar{g}_a = (x_1, x_2, ..., x_i, x_{i+1}, ..., x_8)$$

$$\bar{g}_b = (y_1, y_2, ..., y_i, y_{i+1}, ..., y_8)$$

A random point $i \in \{1, ..., 8\}$ in the two genotypes is chosen, and the two halves at that point are exchanged between the two selected individuals to produce two new individuals $\bar{g}'_a$ and $\bar{g}'_b$.

$$\bar{g}'_a = (x_1, x_2, ..., x_i, y_{i+1}, ..., y_8)$$

$$\bar{g}'_b = (y_1, y_2, ..., y_i, x_{i+1}, ..., x_8)$$

For the experiments reported in this paper, these two operators are the only ones used, as these already sufficed to produce complex enough swarming behaviours. However, as both *Inspirica* [Kwo03] and *Evolvica* [Jac01] provide a much wider set of genetic operators (such as encapsulation, decapsulation, deletion, duplication, multi-point recombination), it is easy to extend the repertoire of mutation operators.

4 Evolutionary Discovery of Swarm Formations

We used *Inspirica* [Kwo03] to explore swarm behaviours, controlled by the parameters described in Section 2, through evolution with interactive fitness assignment. In each evolution experiment, a collection of swarm simulation windows are simultaneously presented to the user who assigns a fitness value between 0 and 10 to each swarm simulation (Fig. 2). Each of these windows shows the swarm behaviour patterns for a particular set of control parameters generated by the evolutionary algorithm. Initially, these parameter lists are randomly generated.

We observe the swarming behaviour in each window for about 300 simulation time units, or until the agents settle into a stable behaviour. This stable behaviour can be a repeated sequence of swarming patterns, or it can be a swarming pattern that never changes. Occasionally, an evolved swarm never settles into an observable swarming pattern. In these cases, we wait for about 300 time units, and then

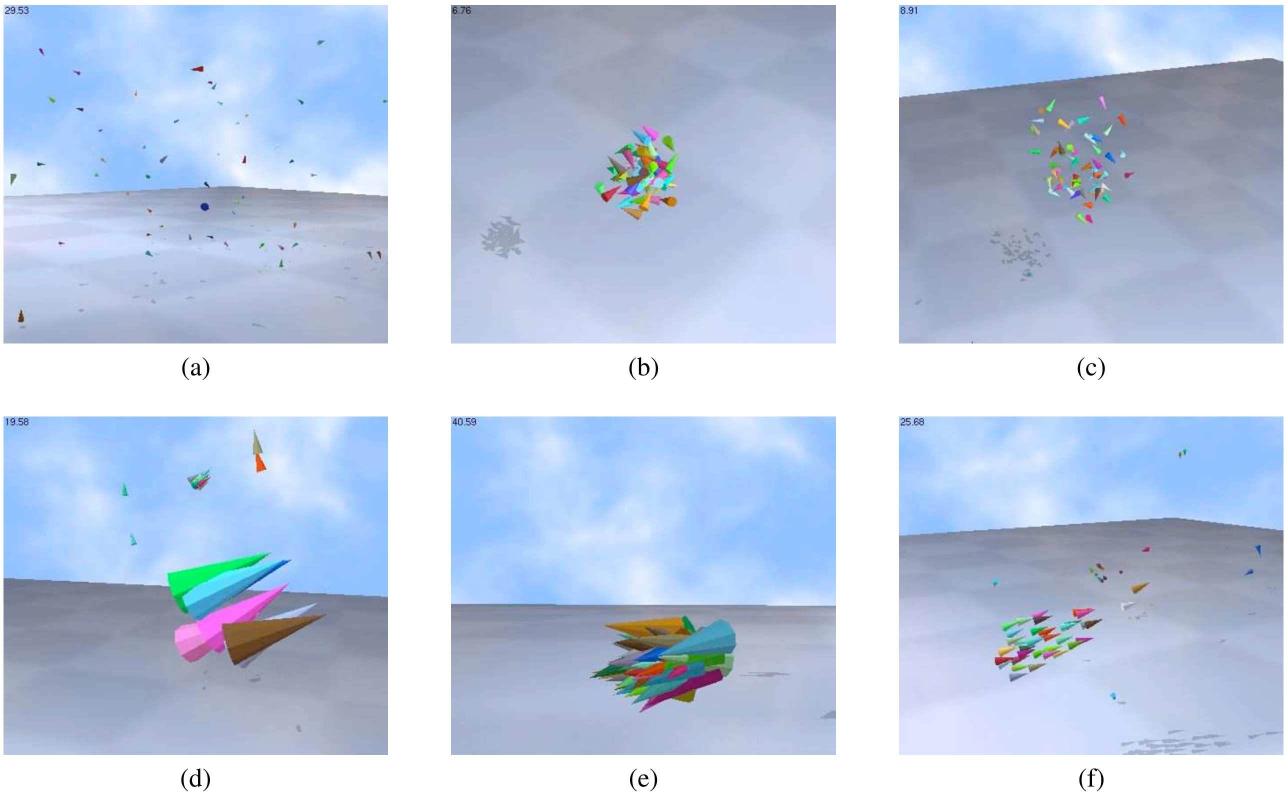


Figure 1: Examples of the most common behaviours that appeared during the evolution experiment. (a) The agents are all scattered and never form clusters or coherent groups (swarms or flocks). (b) The agents form tight stationary clusters. (c) The agents form loose stationary clusters. (d) The agents form small, scattered, mobile flocks. (e) The agents form large, tight, mobile flocks. (f) The agents form large, loose, mobile flocks. Note that the number in the top left corner of each screenshot denotes the simulation time step t when the image was captured. The time steps are: $t = 29, 6, 8, 19, 40, 25$.

assign a low fitness value to the corresponding swarm parameter list. When the agents settle into a stable behaviour, we rank each phenotype based on either how closely they resemble a desired behaviour (e.g., flying in a line formation), or how 'interesting' a newly discovered swarming pattern appears to be. Based on these interactively assigned fitness values, a new generation is created by applying mutation and crossover to the parameter lists of the current generation that were chosen by fitness proportionate selection. This evolutionary process, following a standard genetic programming scheme [Jac01], is usually repeated for about 20 generations. All the parameter values used to produce the behaviours presented in this section are listed in Table 1. For all experiments we used a mutation rate of 15% and a crossover rate of 85%.

4.1 Flocking Behaviour

Figures 1(a) through (d) show screenshots of some of the most common behaviours that were most commonly observed during the breeding experiments. These behaviours appeared most frequently in the early stages of an experiment. One of the first interesting behaviours that emerged was flocking. This was considered an improvement over the previous behaviours because the agents fly in large coherent groups, where about one third or more of the agents stay in the same flock, with seemingly orchestrated in their direction and velocity. Figures 1(e) and (f) show two examples of this flocking behaviour.

4.2 Line Formations and Swirling Behaviour

Figure 4(a) shows the first line formation that was evolved from this experiment. This line only contains a few agents, does not form very often, and is not very stable[1] either. Figure 4(b) shows an interesting swirling ring formation that was serendipitously discovered in a different individual within the same generation.

4.3 Figure-Eight and Ring Formations

Six generations after the line and swirling formation patterns appeared (Figure 4), a swarm control list was generated that causes agents to gather either in a stable figure-

[1]In a stable formation, agents do not break out of the formation unless they encounter the ground plane.

Figure	Behaviour Description	c_5	c_2	c_4	c_3	c_1	V_{max}	A_{max}	d
1 (a)	Scattered Agents	8	9	5	1	6	36	11	0.59
1 (b)	Tight Stationary	8	10	7	8	7	8	38	0.46
1 (c)	Loose Stationary	6	7	5	2	5	6	40	0.23
1 (d)	Several Small Flocks	4	5	7	7	3	13	37	0.4
1 (e)	Tightly Packed Flock	7	15	8	10	6	5	38	0.14
1 (f)	Loosely Packed Flock	10	18	6	7	8	24	6	0.44
4 (a)	Line Formation	5	8	8	7	5	13	38	0.14
4 (b)	Loose Swirling Ring	3	5	7	1	2	5	38	0.43
4 (c) (d)	Figure-Eight and Small Ring	2	5	10	3	1	6	38	0.01
4 (e)	Messy Figure-Eight	3	6	7	3	2	6	40	0.01
4 (f)	Large Ring	1	14	10	5	1	9	39	0.14
3 (b)	Elongated Figure-Eight	1	12	10	5	2	6	35	0.35
5	Long Line Formation	4	10	8	7	4	9	40	0.01

Table 1: The evolved behaviour parameter values used to produce the swarming behaviours described in Section 4.

eight formation (Figure 4(c)) or a stable ring formation (Figure 4(d)). These figure-eight and ring formations emerge from a large range of different initial swarm configurations (i.e., where each simulation run is started with a different random number seed).

This figure-eight formation demonstrates the discovery of self-organized, decentralized, complex behaviour. No single parameter is responsible for the figure-eight formation. The proper proportion of each parameter with every other parameter allows for the emergence of this behaviour. Slight changes in the parameters, that alter their relative proportions, result in less stable figure-eight formations. This is illustrated in Figure 4(e), which shows a slightly mutated offspring of the parent in Figure 4(c). This offspring still forms figure-eight constellations, however these formations break apart more frequently, and the shape of the figure-eight is not as well-defined.

Some parameters are more important to the emergence of the figure-eight behaviour than others. Specifically, small changes in the maximum acceleration (A_{max}) make the behaviour pattern less stable (the formation would break apart more often; compare 1). On the other hand, changes in the weighting for agents to steer toward the center of the world (c_2) has less effect on the stability of the formation.

4.4 Variations of the Figure-Eight and Ring Formations

Figure 4(f) shows a mutated offspring of Figure 4(d). The agents in this offspring form large ring formations, compared to the smaller ring formed in Figure 4(d). However, these large ring formations are not very stable, but they usually form again later after they have broken apart.

Another interesting behaviour pattern that was derived from the figure-eight pattern in Figure 4(c) is shown in Figure 3. The agents in this simulation form very narrow figure-eights, which break into long curvy lines that eventually form back into narrow figure-eights. These behaviours repeatedly cycle through these three formations.

4.5 Line Formations

In addition to all the less expected behaviours that we discovered from these evolution experiments, we also were able to breed a variety of line formations. Starting from the most common behaviours as seen in Figure 1, after 16 generations a swarm parameter listwas evolved that produces long line formations which contain a large number of agents, are fairly stable, and form frequently during the simulation.

5 Future Work

The next step in our research is to evolve actual simulation code, instead of just static parameter lists. This will allow us to evolve an even larger repertoire of more complex global behaviour patterns and will provide the basis for a programming environment for swarm systems. Animations of the results described in this paper, along with other designs evolved with *Inspirica* and other swarm-based evolutionary systems can be found at our *Evolutionary & Swarm Design* website:

`http://www.cpsc.ucalgary.ca/∼jacob/ESD.`

Starting from the examples discusses in this paper and presented on the ESD website, our ultimate objective is to provide research tools for computational biologists to investigate swarming behaviour patterns observed in the natural world, with particular focus on systems studied in proteomics and genomics [BSJ03, PHJ03].

Bibliography

[BBB+98] J. Bloomenthal, C. Bajaj, J. Blinn, M. Cani-Gasuel, A. Rockwook, B. Wyvill, and G. Wyvill. *Introduction to Implicit Surfaces*. Morgan Kaufmann, San Francisco, CA, 1998.

[BSJ03] I. Burleigh, G. Suen, and C. Jacob. Dna in action! a 3d swarm-based model of a gene regulatory system. In *ACAL 2003, First Australian Conference on Artificial Life*, Canberra, Australia, 2003. (submitted).

[dG92] H. de Garis. *Genetic Programming: GenNets, Artificial Nervous Systems, Artificial Embryos*. PhD thesis, Brussels University, 1992.

[Gru95] F. Gruau. Automatic definition of modular neural networks. *Adaptive Behaviour*, 3(2):151–183, 1995.

[Jac96] Christian Jacob. Evolving evolution programs: Genetic programming and l-systems. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. Riolo, editors, *Genetic Programming 1996: First Annual Conference*, pages 107–115, Stanford University, Palo Alto, CA, 1996. MIT Press, Cambridge, MA. reference from SSM (1999) article.

[Jac01] C. Jacob. *Illustrating Evolutionary Computation with Mathematica*. Morgan Kaufmann, San Francisco, CA, 2001.

[KAea99] J. Koza, D. Andre, and et al. *Genetic Programming III: Darwinian Invention and Problem Solving*. Morgan Kaufmann, San Francisco, CA, 1999.

[KML$^+$00] J. R. Koza, W. Mydlowec, G. Lanza, J. Yu, and M. A. Keane. Reverse engineering and automatic synthesis of metabolic pathways from observed data using genetic programming. Technical Report SMI-2000-0851, Stanford University, 2000.

[Koz93] John R. Koza. Discovery of rewrite rules in lindenmayer systems and state transition rules in cellular automata via genetic programming. In *SPF-93, Symposium on Pattern Formation*, Claremont, California, 1993. Dept. of Computer Science, Stanford University, CA.

[Kwo03] H. Kwong. Evolutionary design of implicit surfaces and swarm dynamics. Master's thesis, University of Calgary, Calgary, AB, Canada, 2003.

[Nor93] D. Norman. *Things That Make Us Smarter*. Perseus Books, Reading, Massachusetts, 1993.

[PHJ03] J. Penner, R. Hoar, and C. Jacob. Bacterial chemotaxis in silico. In *ACAL 2003, First Australian Conference on Artificial Life*, Canberra, Australia, 2003. (submitted).

[Rey91] C. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4), July 1991.

[SBBS99] L. Spector, H. Barnum, H. Bernstein, and N. Swamy. Quantum computing applications of genetic programming. In L. Spector, W. Langdon, U. O'Reilly, and P. Angeline, editors, *Advances in Genetic Programming 3*, chapter 7, pages 135–160. MIT Press, Cambridge, MA, USA, June 1999.

[Sim94] K. Sims. Evolving 3D Morphology and Behavior by Competiton. *Artificial Life IV*, pages 28–39, 1994.

[Sip97] M. Sipper. *Evolution of Parallel Cellular Machines: The Cellular Programming Approach*. Springer, Berlin, 1997.

[SK02] L. Spector and J. Klein. Evolutionary dynamics discovered via visualization in the breve simulation environment. In *Artificial Life VIII*, Reading, MA, 2002. Addison-Wesley.

[SM02] K. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

[Wol01] S. Wolfram. *A New Kind of Science*. Wolfram Media, Champaign, IL, 2001.

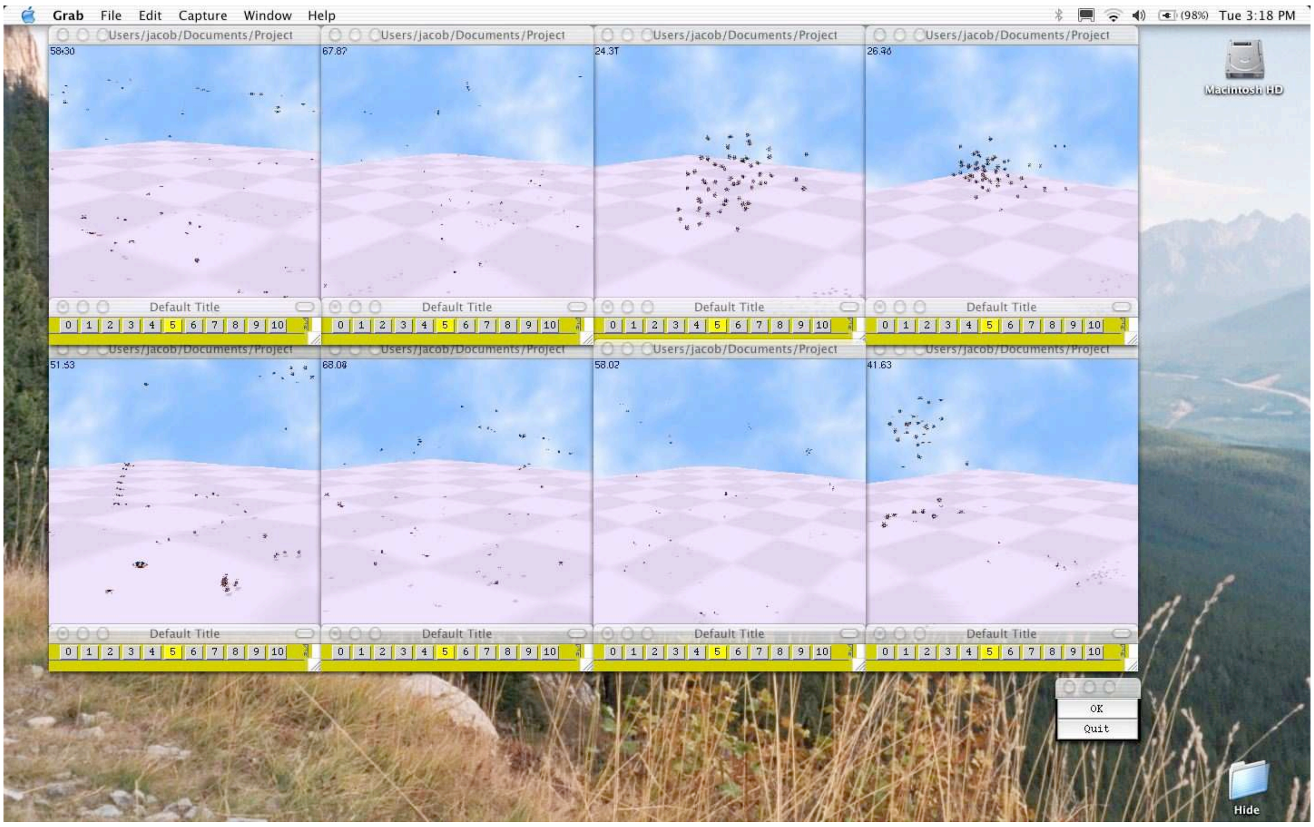
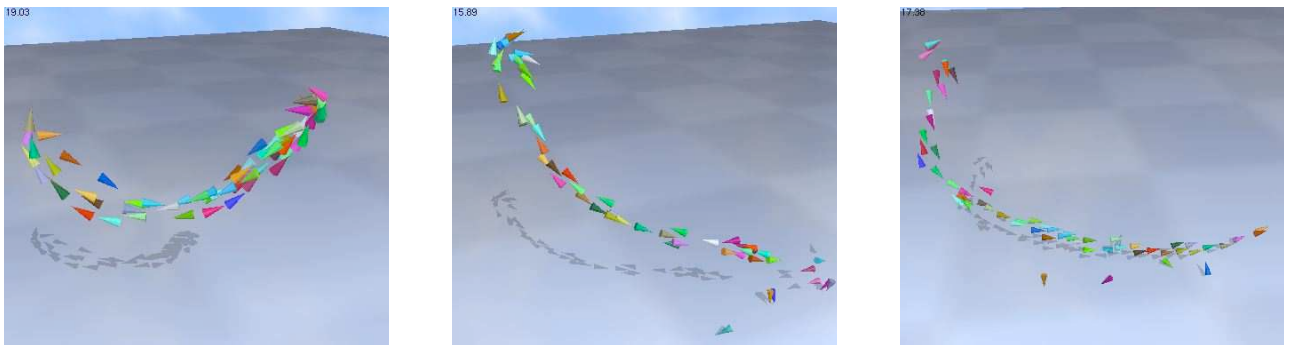


Figure 2: A screenshot of the evolutionary design interface through which we interactively 'breed' swarm behaviours. In this particular case, eight swarm simulation windows—each controlled by a different set of parameters (as described in Section 2)—are simultaneously presented on the screen. Below each simulation window is an evaluation scale, which allows the user to evaluate the corresponding swarm behaviour on a scale from 1 to 10. Assigning a fitness of zero to a simulation will cause it to be removed from the window set in the next evaluation iteration. After all behaviours have been evaluated (the default fitness is set to 5), the next set of simulations is generated and displayed, so that the user can continue with the interactive evaluation. All the results discussed in this paper were evolved through no more than 20 evaluation cycles.



(a) (b) (c)

Figure 3: The agents cycle between a narrow figure-eight formation (a), an elongated figure-eight formation (b) that breaks into a long curved line formation (c), and then eventually repeats from (a).

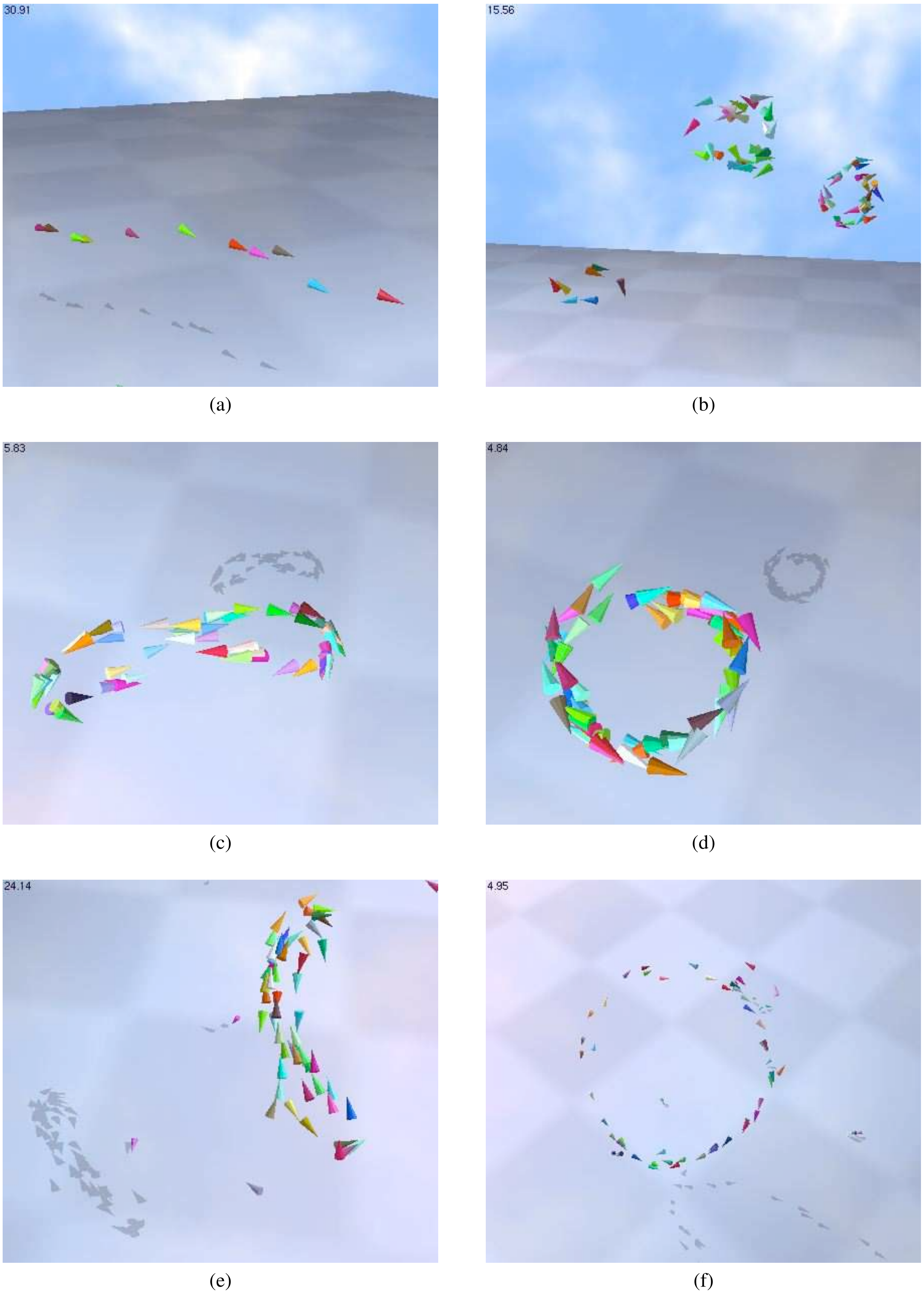


Figure 4: (a) A simple evolved line formation. (b) Evolved loose, swirling, ring formations. (c) A figure-eight formation that evolved from (a) and (b). (d) A ring formation that evolved from (a) and (b). In this formation, the agents fly in both clockwise and counterclockwise directions. It is like 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 ring that evolved from (d). This ring is similar to (d) except that this one is larger and the agents are more spread out.

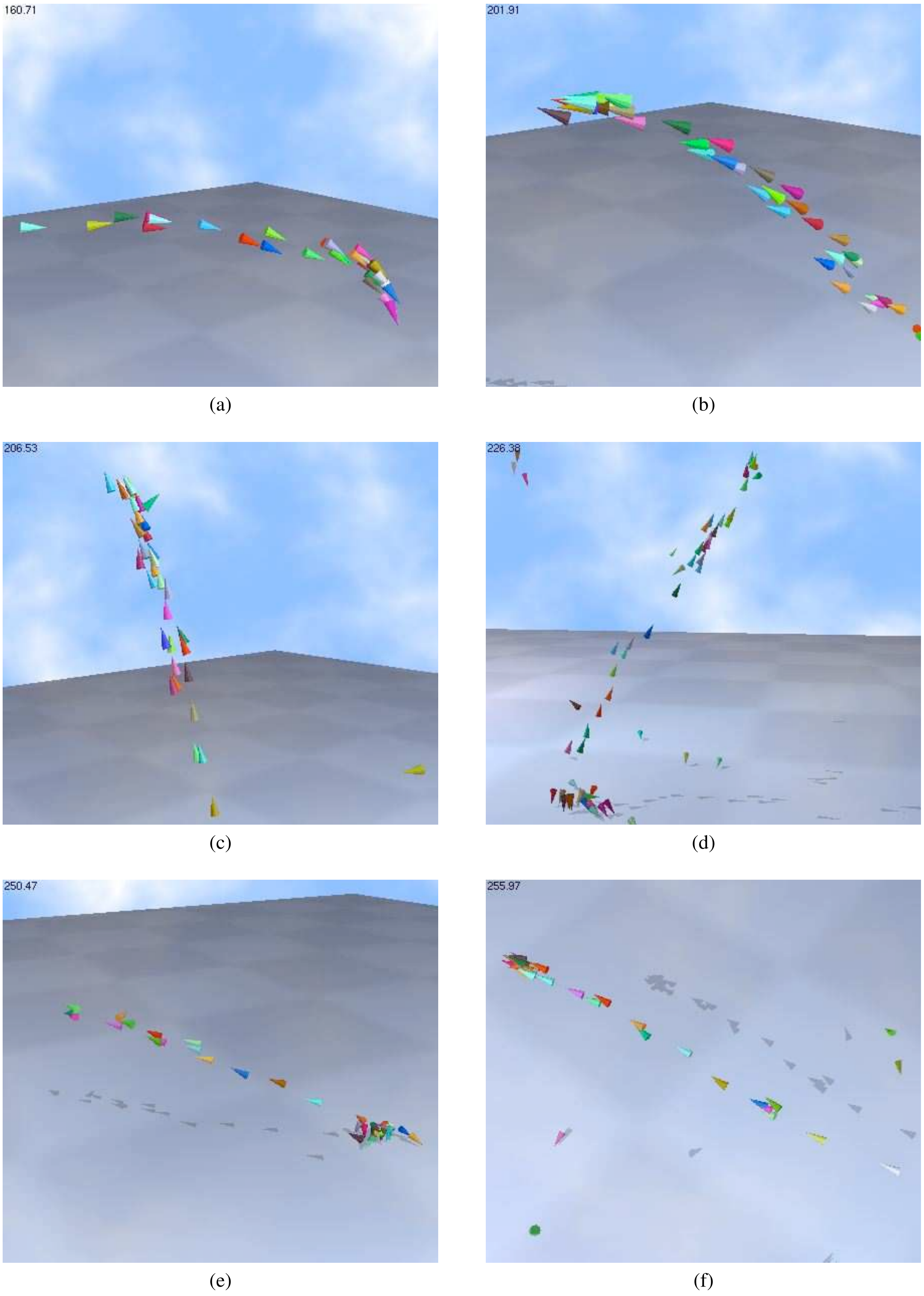


Figure 5: In this simulation, agents constantly form long line formations involving a large number of agents. This figure shows screenshots of several line formation patterns that occur at different times during the simulation.