Swarming For Games: Immersion in Complex Systems

Sebastian von Mammen[1] and Christian Jacob[1,2]

[1] University of Calgary, Dept. of Computer Science, Calgary, Canada
[2] University of Calgary, Dept. of Biochemistry & Molecular Biology, Calgary, Canada

Abstract. The swarm metaphor stands for dynamic, complex interaction networks with the possibility of emergent phenomena. In this work, we present two games that challenge the video player with the task to indirectly guide a complex swarm system. First, the player takes control of one swarm individual to herd the remainder of the flock. Second, the player changes the interaction parameters that determine the emergent flight formations, and thereby the flock's success in the game. Additionally, a user-friendly interface for evolutionary computation is embedded to support the player's search for well-performing swarm configurations.

1 Introduction

AI algorithms have been successfully applied in computer games for pattern detection in human-computer interfaces [5] or for non-player character optimization, i.e. shortest path search [10] and planning [11]. We propose that bio-inspired methodologies might directly offer new approaches to game principles.

In this work, artificial swarms confront the player with a novel perspective: (1) When partaking in a flocking collective, and (2) when conjuring and utilizing emergent flocking formations by regulating a swarm's underlying interaction parameters. We present game concepts including mission statements, level design and user-interaction elements that support both approaches. The interaction patterns in artificial swarms can be highly dynamic and complex. In order to handle this complexity, we provide evolution-based mechanisms to breed swarm configurations in an intuitive manner. To our knowledge, no similar swarm-based games have been presented in the past.

In the next section, we briefly introduce the scientific background for the presented work and related games. In the two subsequent sections we draw a comprehensive picture of two fully-featured, swarm-based games. Finally, our results are briefly summarized, and we suggest future steps for expanding our swarm-based gaming concepts.

2 Swarms & Emergent Gaming

In 1987 Reynolds presented a simulation concept that mirrors the flocking dynamics of birds [12]. In his model, a simulated virtual bird i, also called *boid*, is

represented as a pyramidal graphical object oriented towards its velocity $\overrightarrow{v_i}$. In order to be able to react to its neighbors, a boid possesses a conic field of perception that is determined by a radius r and an angle α. Rather loosely, Reynolds explained which accelerating urges are triggered by the flocking mates perceived in their neighborhood. *Alignment* adjusts the direction to the neighbors, *cohesion* draws the individual towards its neighbors, and an urge for *separation* prevents the boids from bumping into each other. Combined with some randomness, these rudimentary urges suffice to let large crowds of agents flock smoothly in virtual 3D spaces.

Many AI development frameworks provide a basic boid implementation [4, 8, 1, 19]. Our present boid simulation mainly follows the implementation discussed in [9, 17]. In a first step, the neighborhood of each individual is re-calculated based on position updates of the last iteration. The set of neighbors is used to calculate the urges of separation, alignment, and cohesion, as well as an urge towards the world center and a random vector. The classic boid urges are normalized through division by the number of considered neighbors, the world center and the random urges renormalized to unit-vectors. An individual's acceleration is computed by the weighted sum of urges, whereas the weights are part of the swarm agent's configuration. Both acceleration and velocity are limited by a maximal value (again part of the configuration). In a second step, each boid's location is (simultaneously) updated.

As repulsion from the neighboring mates can lead to sudden flock dispersions, we calculate the separation urge as the sum of deflection vectors from the mates. The deflection is calculated by reflecting the vector from the individual to its mate in respect to the individual's velocity. Alignment is simply implemented as the sum of the neighbors' velocities. Cohesion is calculated as the sum of distance vectors. Detailed formulas are provided, for instance, in [17].

Knowing the swarm individuals' positions and orientations, as well as the dimensions of their fields of perception, the interdependencies of a given swarm can be inferred. However, we implemented and tested two visualization methods in order to enhance the gaming experience. Figure 1(a) shows a set of boids flocking in a loose cluster formation. The individuals' orientation is indicated by their rotation. In Figure 1(b) the fields of perception are illustrated. The visualization methods in Fig. 1(a) and 1(b) are adapted from Reynold's original boids publication [12]. When connecting all neighboring boid agents as in Fig. 1(c), the flock appears as a continuously changing interaction network, as suggested in [17]. The field of view visualization (Fig. 1(b)) allows the player to better understand the swarm's movements. The network visualization (Fig. 1(c)) can facilitate the game play, especially since dependency chains are depicted that determine the success of herding or dragging tasks in the game, as we explain in Section 3.

The idea to use artificial swarms in video games becomes obvious when experiencing interactive swarm art installations. Jacob et al., for instance, have demonstrated the interactive potential of swarm simulations steered by video-captures of the audience's movements [6]. The natural feel to its smooth motions,

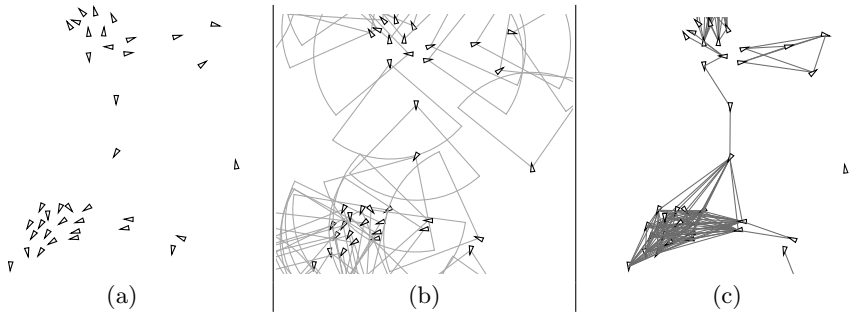


Fig. 1. (a) A boid flock in a loose formation. (b) The geometric fields of perception are illustrated. (c) Edges depict the qualitative interaction network.

the perceived liveliness of the swarm motivates the audience to explore its responsiveness and to enjoy swarm-generated melodies or virtual paintings drawn during the flock's movements. The interactivity of artificial music-generating swarms even supports live performing musicians [3, 2]. An example for an autonomous model is given in *AtomSwarms* where music generating agents evolve and self-organize in a complex eco-system [7]. Within *AtomSwarms* and during live performances a conductor can change the population size of the interactions, but is not provided with the means to directly interfere in the simulation. In other contexts, artificial swarms were bred in well-directed evolutionary runs. The characterization of boid flocks based on the average neighborhood perception allowed to track the changes within the boid interaction network and breed, for instance, an oscillating flock formation [17]. On the other hand, boid swarms were bred by means of interactive evolution to exhibit various flocking formations [9]. Interactive evolution was also applied to explore the capabilities of swarm systems called *Swarm Grammars* that build three-dimensional structures [16]. In these experiments, so-called *breeder volumes* were introduced that apply the genetic operators mutation and recombination to swarm individuals that are passing through.

3 Herding Cattle

In our first swarm-based game prototype, the player is directly immersed in the process of flocking formation. In particular, the player's task is to influence the overall flock, while only navigating one swarm individual. In order to manage this task, one quickly has to gain an intuitive idea of how to control an individual's flight pattern, while taking into account neighborhood relations and affects on overall acceleration behaviors. These skills, in tandem with the player's fast identification of, for example, appearing obstacles and reaction abilities are the ingredients of this basic game concept.

A simple game setup challenges the player to get a number of swarm agents from a starting area to a goal location. Figure 2 shows a screenshot of an ac-

cording implementation[3] in the programming environment Processing [1]. In the following paragraphs, the various elements of the screenshot are magnified and explained in detail.

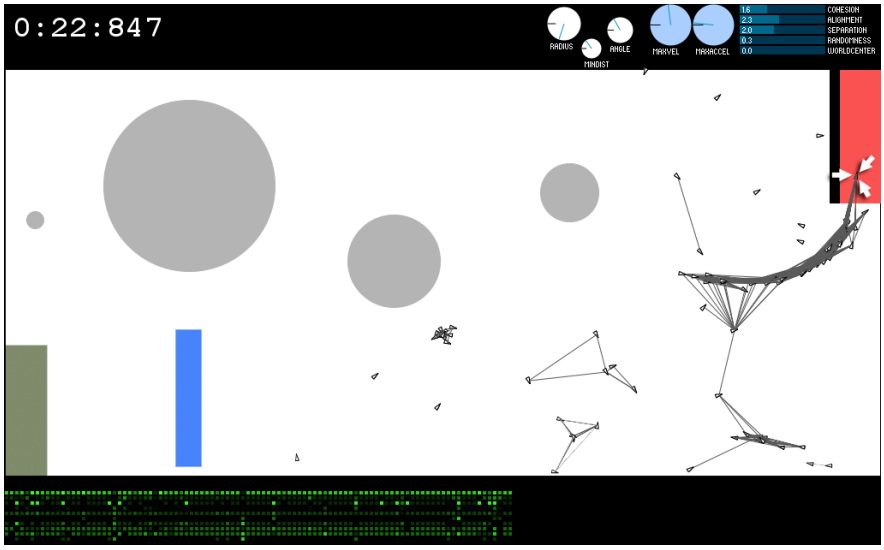


Fig. 2. Screenshot of a 2D boid herding game. The player's swarm individual (highlighted by three white arrows) drags a "tail" of associated agents into the goal.

3.1 Playground Interactions

At the beginning of the game, the swarm individuals are positioned in the bottom left corner of the playground (Fig. 3(a)). The player has to herd the flock into the upper right corner (Fig. 3(b)) only by steering one individual that follows the coordinates of the mouse. When the player steers its individual beyond the perceptional range of its peers, they fall behind. If the guiding individual moves too swiftly, it does not exercise a great impact on its mates' flight momentum. On the way to the goal, there are several obstacles denoted as circles and rectangles. The swarm agents deflect from the four circles (Fig. 3(c)), bounce off the dark bar next to the goal ((Fig. 3(d)), and change their flocking configurations, if they pass through the stripe below the biggest circle. The latter effect is, in fact, implemented as a genetic mutation operator [16]. Analogously, the swarm agents' configurations can be understood as their genotypes, their geometric representation and exhibited behaviors as phenotypes.

[3] Its development started on the basis of Daniel Shiffman's boid implementation "Flocking" that is included in Processing's examples library.

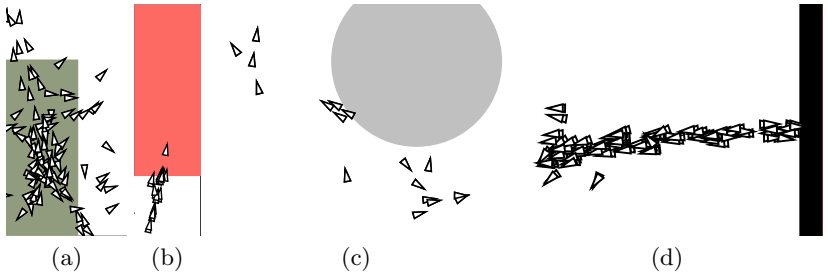


Fig. 3. Level design constituents of a herding game prototype.

3.2 GUI for Genetic Manipulation

Different information about the game is displayed on the black stripes at the bottom and the top. In the upper left corner, the time is shown that has elapsed since the game started. As in many other games, timing is a simple means to measure the efficiency of the player's mastery. On the right-hand side of the upper stripe are three knobs to adjust the swarm agents' fields of perception (Fig. 4(a)) and two more to set the maximal values for acceleration and velocity (Fig. 4(b)). They are followed by five sliders that control the weights of the agents' flocking urges (Fig. 4(c)). The last slider is colored in a different shade. It does not change the swarm's properties, but changes the integration step size of the simulation. Changes through this graphical user interface[4] are applied either to all swarm agents or to selected individuals (Fig. 4(d)). Differences among the individuals are depicted in the bottom graphic in Figures 2 and 4(d). The parameter values are mapped to color values between 0 and 255 corresponding to black and bright green, respectively. One column of pixels represents the genotype of one individual.

4 Crows in the Cornfield

In a second swarm game, the player does not interact with the swarm spatially, but exclusively changes its genotype. The metaphor of the game is a flock of crows that feeds on corn. The same GUI as in the previous game is utilized to configure and thereby direct the flock towards the corn and to maximize the harvest. At the same time, the swarm individuals interact with various objects (as in Section 3), and are even endangered to fly into deathtraps. The latter ones are depicted as the bright rectangular areas in Figure 5. The even brighter, circular spots represent growing corn, the black areas repelling objects. The game ends when all swarm individuals have died. The objects on the playground along with the crops are constantly scrolling towards a random direction. Thus, the flock is confronted with a changing configuration, and the player has to be quick-witted to ensure the flock's survival and a successful harvest. Two GUI elements are different from the previously described herding game: (1) The harvesting

[4] We rely on the controlP5 GUI library by Andreas Schlegel (http://www.sojamo.de).

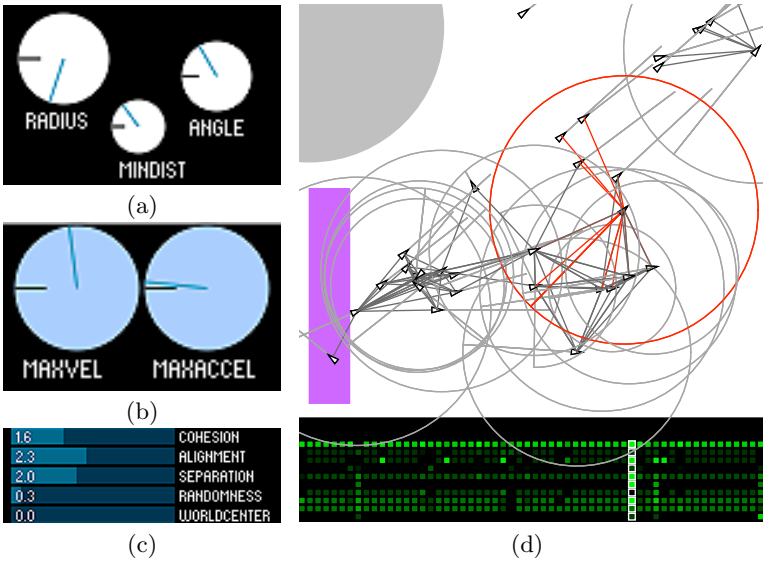


Fig. 4. Graphical user elements allow to change (a) an agent's field of perception, (b) its maximal velocity and acceleration, and (c) its flocking weights. (d) A selected agent is rendered in a different color.

score is displayed in the upper-left corner. (2) An additional slider is depicted left of the knob for radius adjustments. By means of this slider, the player can choose one of a set of previously stored swarm genotypes. The selected swarm genotype is immediately deployed in the game, thus facilitating the player's task of parameter-based, indirect swarm navigation.

4.1 Evolving Swarm-Macros

As part of the game, but before the player is actually urged to control the flock, two ways are offered to create an assortment of swarm configurations. As seen in Section 3.2, swarm configurations can be manually designed and tested. Alternatively, the player may rely on computational evolution to automatically breed a swarm genotype. By placing a number of tiles on the playground, the player determines where the swarm individuals should be flocking (Figure 6). This alternative evolutionary way to give rise to swarm configurations with specific trajectories is quite important. Manual adjustment of a swarm's flocking parameters can be a very challenging task, that could easily frustrate the player. The implemented breeding approach is similar to the evolution of constructive swarms based on predefined shapes [18]. This implements interactive guidance of the evolutionary development not unlike the immersive "gardening" techniques presented in [16]. The idea to train agents and deploy them during a game has been brought about on numerous occasions, e.g. in [14] or in Creatures (most recent release in 2003, Creature Labs).

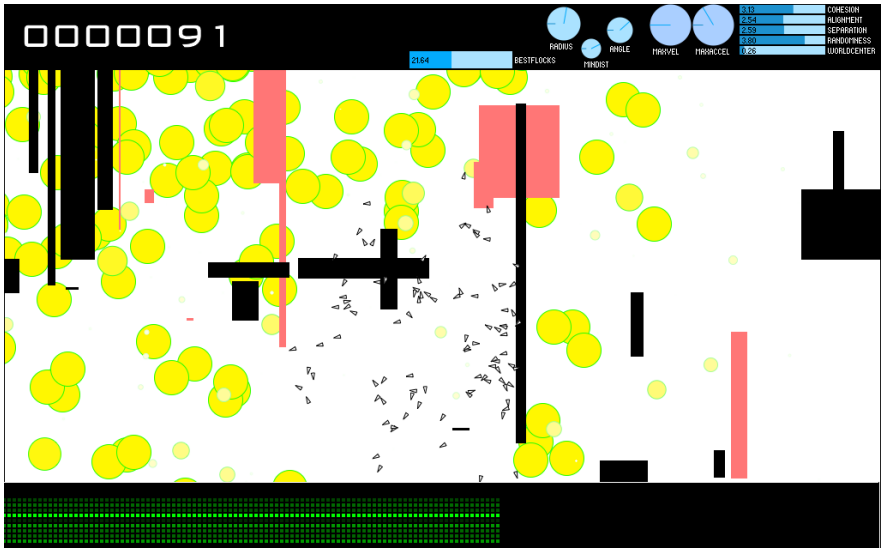
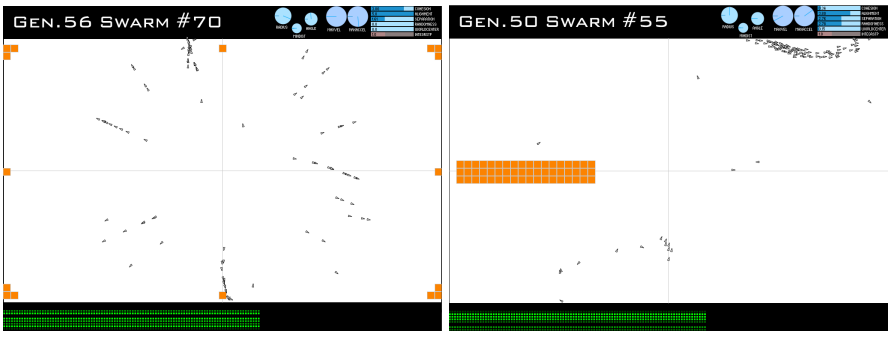


Fig. 5. Screenshot of the game "Crows in the Cornfield" in which the player maximizes the flock's harvest by changing its configuration.



(a) (b)

Fig. 6. (a) A flock has learned to swarm to the edges of the playground. (b) The flight in formation of a broad stripe maximizes the flock's reward when hitting the tiles.

$$f_{swarm} = \frac{1}{ia} \sum_{i} \sum_{t} \min(c(t,i), c_{max}), \ \ with \ c_{max} = \frac{a}{t} \tag{1}$$

When entraining a swarm, the fitness function given in Equation 1 is applied to evaluate its performance. With a as the number of swarm agents, t the number of tiles, i the number of iterations and $c(t,i)$ as the number of collisions between swarm agents and a tile t at iteration i, the function can be read as follows: Over all iterations of a flocking simulation, each collision between an agent and a tile is considered for reward. However, in the case of a perfect distribution over the

given tiles, there should only be up to c_{max} agents on one tile. The final sum is normalized by the product of iterations and the number of swarm agents.

With the next generation composed of 50% recombined and 50% mutated material (with a mutation probability $p = 0.2$ on single flocking parameters), we successfully bred exemplary flocks. In Fig. 6(a) a swarm breaks up into several groups that are heading towards the edges of the playground, where they hit the tiles placed by the user. In another experiment, the flocking formation shown in Fig. 6(b) achieved a high fitness value. The similarity between the shape of the swarm formation in upper-right corner and the tiled area is notable. Another specimen of the same evolutionary experiment is presented in Figure 7. With the world center as the sole geometric point of reference, a solution to the given, non-symmetrical task is promoted that utilizes the general setting and a great degree of randomness. In Fig. 7(a) the flock radially expands from its origin. When repelled from the edges, it breaks into four parts (Fig. 7(b)). To the left and the right of the playground, two swarm groups re-emerge heading back towards the world center (Fig. 7(c)). This strategy results in an effective pass over the tiles to the left center of the playground (Fig. 7(c) and (d)).

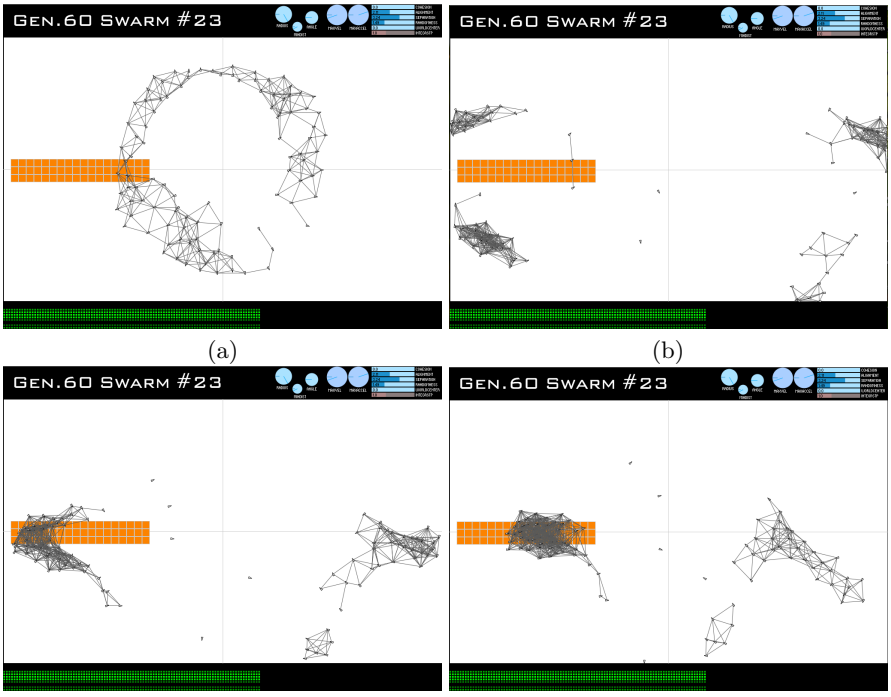


Fig. 7. An evolved swarm relies on interactions with the environment in order to hit a non-symmetrically placed tiled area.

5 Results

We have presented two games that rely on swarm-dynamics. A herding game is introduced in which the player effects the directed flocking of a swarm by taking control of only one individual. This prototype tests the idea of player immersion in a dynamic, complex swarm system. With an obediently following swarm configuration the herding game poses a task as unambitious as finding the path through a maze from a top view. However, the challenge is increased by changing the swarm individuals' genotypes when flying over a mutation area. An intuitively operable GUI lends itself to the player to limit the damage. Control of the flock can be regained by patching single flocking parameters, by reconfiguring a group of swarm individuals to out-balance the maladjustment, or by canny flight maneuvers of the single individual that is manually steered by the player.

Second, a flock of virtual crows is controlled solely through manipulation of their interaction parameters in order to maximize their corn harvest. In addition to real-time adjustments of the swarm individuals as in the herding game, the player may breed assortments of swarm configurations upfront and deploy them afterwards in the game. This evolutionary component is especially important as designing specific flocking configurations can be a frustrating and tedious task.

6 Conclusion and Future Work

The fact that both presented games are embellished with additional game elements—for instance repelling blocks, deathtraps, time and score measures—cannot hide their prototypic character. Emphasis was put on testing ideas of player immersion in an artificial swarm. Both games live on an intuitive understanding of the swarms' flocking dynamics and on the players' skilled control. We believe that they have great potential to grow mature and to seemlessly incorporate many other gaming concepts, for instance realizing the swarm as a cooperative multi-player game.

As a next step, however, it would help to measure, rate and eventually improve the games in regards to basic aspects such as the means to control, (re-) configure and breed the swarms and the clarity of the immediate goals [15].

Swarms serve as a metaphor for highly dynamic complex systems [17]. As such, an improvement in understanding their intrinsic dynamics and in their control supports endeavors in all kinds of undertakings, whether they are economically or ecologically motivated. Therefore, we hope that by promoting the swarm metaphor in computer games, we foster the comprehension and mastery of other complex systems around us.

References

1. Aesthetics and Computation Group at the MIT Media Lab. Processing language and programming environment, http://processing.org/, October 2008.
2. T. Blackwell. Swarming and music. In Edurado Reck Miranda and John Al Biles, editors, *Evolutionary Computer Music*, pages 194–217. Springer London, 2007.
3. TM Blackwell and P. Bentley. Improvised music with swarms. In *Proceedings of IEEE Congress on Evolutionary Computation*, volume 2, pages 1462–1467, 2002.
4. Ian Burleigh. Vigo::3d: A framework for simulating and visualizing of three-dimensional scenes. http://vigo.sourceforge.net/docs/, October 2008.
5. W.T. Freeman, P.A. Beardsley, H. Kage, K.I. Tanaka, K. Kyuma, and C.D. Weissman. Computer vision for computer interaction. *ACM SIGGRAPH Computer Graphics*, 33(4):65–68, 1999.
6. Christian Jacob, Gerald Hushlak, Jeffrey Boyd, Paul Nuytten, Maxwell Sayles, and Marcin Pilat. Swarmart: Interactive art from swarm intelligence. *Leonardo*, 40(3), 2007.
7. Daniel Jones. Atomswarm: A framework for swarm improvisation. *Applications of Evolutionary Computing*, pages 423–432, 2008.
8. Jon Klein. breve: a 3d simulation environment for multi-agent simulations and artificial life. http://www.spiderland.org/, October 2008.
9. Henry Kwong and Christian Jacob. Evolutionary exploration of dynamic swarm behaviour. In *Congress on Evolutionary Computation*, Canberra, Australia, 2003. IEEE Press.
10. Alexander Nareyek. Ai in computer games. *Queue*, 1(10):58–65, 2004.
11. J. Orkin. Agent Architecture Considerations for Real-Time Planning in Games. *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment*, 2005.
12. Craig W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In *SIGGRAPH '87 Conference Proceedings*, volume 4, pages 25–34, 1987.
13. Lee Spector, Jon Klein, Chris Perry, and Mark Feinstein. Emergence of collective behavior in evolving populations of flying agents. *Genetic Programming and Evolvable Machines*, 6(1):111–125, 2005.
14. Kenneth O. Stanley, Bobby D. Bryant, and Risto Miikkulainen. Real-time neuroevolution in the nero video game. *IEEE Transactions on Evolutionary Computation*, 9:653–668, 2005.
15. Penelope Sweetser and Peta Wyeth. Gameflow: a model for evaluating player enjoyment in games. *Comput. Entertain.*, 3(3):3–3, 2005.
16. Sebastian von Mammen and Christian Jacob. Genetic swarm grammar programming: Ecological breeding like a gardener. In Dipti Srinivasan and Lipo Wang, editors, *2007 IEEE Congress on Evolutionary Computation*, IEEE Press, pages 851–858, 2007.
17. Sebastian von Mammen and Christian Jacob. The spatiality of swarms — quantitative analysis of dynamic interaction networks. In *Proceedings of Artificial Life XI*, pages 662–669. MIT Press, 2008.
18. Sebastian von Mammen, Christian Jacob, and Gabriella Kókai. Evolving swarms that build 3d structures. In *CEC 2005, IEEE Congress on Evolutionary Computation*, Edinburgh, UK, 2005. IEEE Press.
19. Uri Wilensky and the CCL at Northwestern University. Netlogo: A cross-platform multi-agent programmable modeling environment. http://ccl.northwestern.edu/netlogo/, October 2008.