Complex Adaptive Music Systems in the BREVE Simulation Environment

Lee Spector
Cognitive Science
Hampshire College
Amherst, MA 01002, USA
`lspector@hampshire.edu`

Jon Klein
Physical Resource Theory
Chalmers University of Technology and Göteborg University
SE-412 96 Göteborg, Sweden
also: Cognitive Science, Hampshire College, Amherst, MA 01002, USA
`jklein@hampshire.edu`

ABSTRACT

We describe two sets of musical applications of the BREVE simulation system. The first adds musical events and interactions to a system of evolving, goal-directed swarms. The second extends 3D cellular automata to respond to musical input and to generate musical output. Although this paper will print in black and white in the proceedings, color versions of the figures, along with links to audio samples and other materials, can be found on-line at http://hampshire.edu/lspector/alife8-music.html.

Complex Adaptive Music Systems in the BREVE Simulation Environment

Lee Spector

Cognitive Science
Hampshire College
Amherst, MA 01002, USA
`lspector@hampshire.edu`

Jon Klein

Physical Resource Theory, Chalmers U. of Technology
and Göteborg University, SE-412 96 Göteborg, Sweden
also: CS, Hampshire College, Amherst, MA 01002, USA
`jklein@hampshire.edu`

Abstract

We describe two sets of musical applications of the BREVE simulation system. The first adds musical events and interactions to a system of evolving, goal-directed swarms. The second extends 3D cellular automata to respond to musical input and to generate musical output. Although this paper will print in black and white in the proceedings, color versions of the figures, along with links to audio samples and other materials, can be found on-line at http://hampshire.edu/lspector/alife8-music.html.

Music and BREVE

BREVE, a simulation package designed by the second author (Klein) for realistic simulations of decentralized systems and artificial life in 3D worlds, was developed for scientific, not aesthetic, purposes. Its high quality graphic output, however, inspired us to work on aesthetic projects in addition to the several scientific projects currently underway. Because BREVE also includes facilities for sound input and output, these aesthetic projects naturally developed musical dimensions.

BREVE simulations are written by defining the behaviors and interactions of agents using a simple object-oriented programming language called STEVE. BREVE provides facilities for rigid body simulation, collision detection/response, and articulated body simulation. BREVE includes a powerful OpenGL display engine that allows observers to manipulate the perspective in the 3D world and view the agents from any location and angle. With these features BREVE simplifies the rapid construction and visualization of complex multi-agent simulations.

The current distribution of BREVE includes facilities for music output via Apple Computer's QuickTime Musical Instruments (a software MIDI synthesizer) and we have also recently added a parameterized multi-channel sine wave generator. For music input we built a version of the BREVE environment that works as a visual plugin in Apple Computer's iTunes audio player; this provides STEVE commands that access audio waveform and spectrum data in real time.

More information about BREVE can be found in (Klein 2002). The BREVE system itself can be found on-line at http://www.spiderland.org/breve.

In the following sections we describe two sets of musical applications of BREVE. The first consists of extensions to an evolving swarm system that was developed initially for scientific purposes (Spector & Klein 2002); we first describe the simple SWARM system that we began with, then the evolving SWARMEVOLVE system which we designed to study the evolution of multiagent systems, and finally the aesthetically motivated SWARMEVOLVE-MUSIC and its variants. We then describe the second set of applications, a set of musical extensions of a 3D cellular automaton system that was developed from the start for aesthetic ends. We conclude with some brief comments about the promise of music projects based on complex adaptive systems.

SWARM

The standard distribution of BREVE includes, as a demonstration program, a "flocking" simulation called SWARM. In this program, which is modeled on the "boids" work of Craig W. Reynolds (Reynolds 1987), the acceleration vector for each agent is determined at each time step as follows:

$$\mathbf{V} = c_1\mathbf{V}_1 + c_2\mathbf{V}_2 + c_3\mathbf{V}_3 + c_4\mathbf{V}_4 + c_5\mathbf{V}_5$$

$$\mathbf{A} = m(\frac{\mathbf{V}}{|\mathbf{V}|})$$

The c_i are constants while the $\mathbf{V}_i$ are vectors determined from the state of the world (or in one case from the random number generator) which are normalized to length 1. Specifically, $\mathbf{V}_1$ is a vector away from neighbors that are within a "crowding" radius, $\mathbf{V}_2$ is a vector toward the center of the world, $\mathbf{V}_3$ is the average of the agent's neighbors' velocity vectors, $\mathbf{V}_4$ is a vector toward the center of gravity of all agents, and $\mathbf{V}_5$ is a random vector. We normalize the resulting velocity vector to length 1 (assuming its length is not zero) and set the agent's acceleration to the product of this result and m,
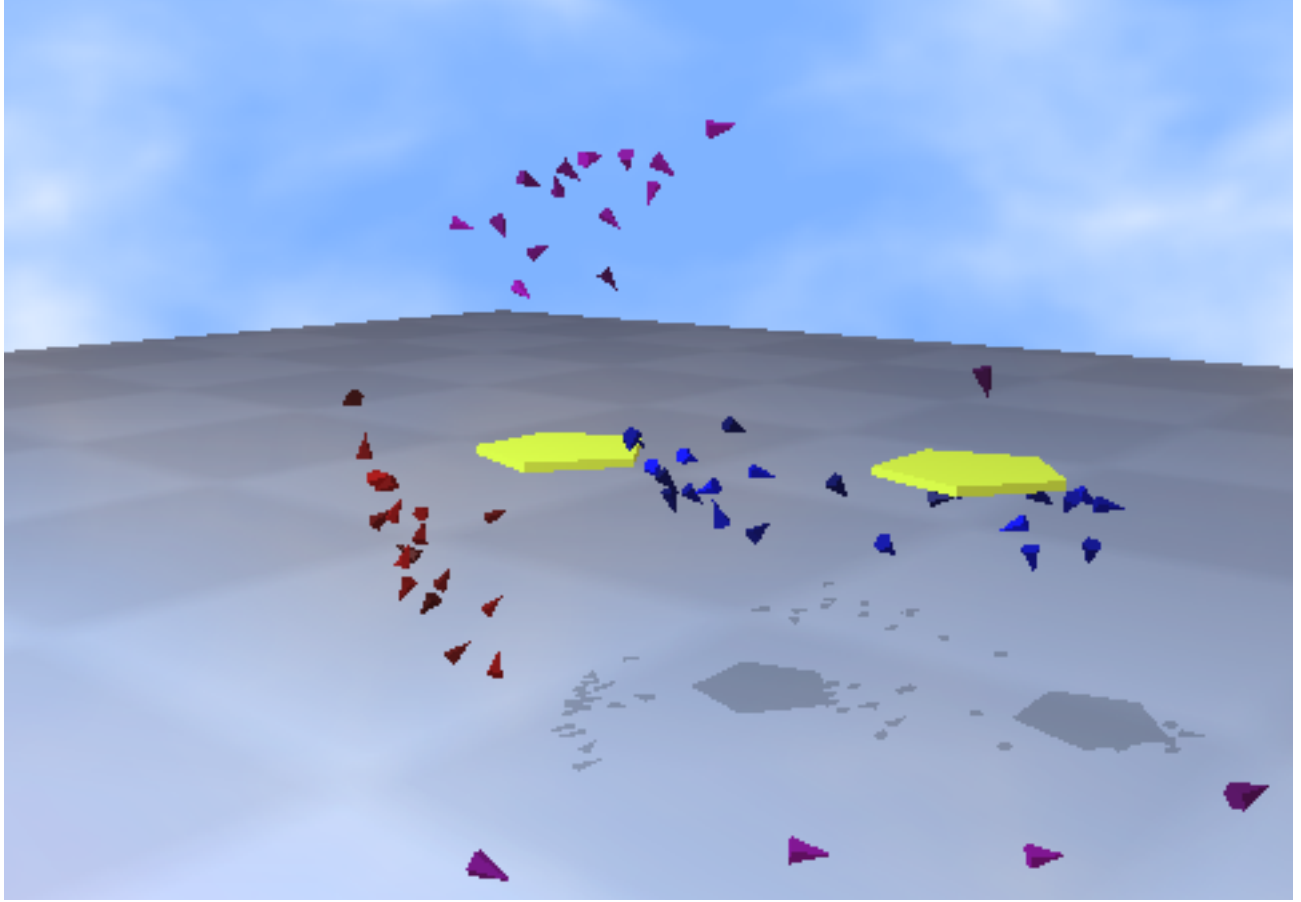
Figure 1: A view of SwarmEvolve. A color version of this figure can be found at http://hampshire.edu/lspector/alife8-music.html. All agents are the same size so relative size on screen indicates distance from the camera.

a constant that determines the agent's maximum acceleration.

If an agent comes sufficiently close to the floor then it "lands," which just means that it is positioned on the floor with zero velocity. After a small random interval of time a landed agent will take off in a new random direction.

By setting different values for system's various parameters (including the c_i and m constants, the "crowding" distance, and the number of agents) one can obtain a range of different flocking behaviors; many researchers have explored the space of these behaviors since Reynolds's pioneering work.

SwarmEvolve

As part of our research program on the evolution of multiagent systems we sought to explore the evolution of controllers for heterogeneous swarms of goal-directed agents. To do this we made a number of enhancements to the SWARM program, producing a program that we call SwarmEvolve. This is a continuing project and we will not describe all of its elements here; we will de-

scribe only the fairly simple version of SwarmEvolve which served as the basis of our aesthetic experiments. For more detail on this and related work see (Spector & Klein 2002).

The first enhancement that we made to produce SwarmEvolve was to create three distinct species of agents, each designated by a different color. As part of this enhancement we added a new term, $c_6\mathbf{V}_6$, to the motion formula, where $\mathbf{V}_6$ is a vector away from neighbors *of other species* that are within a "crowding" radius.

Goal-orientation was introduced by adding randomly moving "energy" sources to the environment and imposing an energy dynamics. Part of this enhancement was to add an additional term, $c_7\mathbf{V}_7$, to the motion formula, where $\mathbf{V}_7$ is a vector toward the nearest energy source. Each time an agent collides with an energy source it receives an energy boost (up to a maximum), while each of the following bears an energy cost:

- Survival for a simulation time step (a small "cost of living").

- Collision with another agent.

- Being in a neighborhood (bounded by a pre-set radius) in which representatives of the agent's species are outnumbered by representatives of other species.

- Giving birth (see below).

The numerical values for the energy costs (along with those for "crowding" radii and other parameters) can be adjusted arbitrarily.

The final enhancement to the version of SwarmEvolve used here was to construct a "fitness" function based on the energy dynamics and to genetically encode the control constants to allow for evolution. Each agent has its own set of c_i constants that control its behavior (via the enhanced motion formula). In the enhanced system the string of c_i constants serves as the agent's genotype. When an agent's energy falls to zero the agent "dies" and is "reborn" in the same location; when this happens the agent receives a new genotype and an infusion of energy. Birth energies are typically chosen to be random numbers in the vicinity of half of the maximum. The agent's new genotype is taken, with possible mutation (small perturbation of each constant) from the "best" current individual of the agent's species. The concept of "best" used here is the *product* of energy and age (in simulation time steps)—one is fit if one has a lot of energy and one is also fit if one is old; the fittest are those that are both old and energetic.

The visualization system presents an automatically scaled and targeted 3D view of the geometry of the world and all of the agents in real time. Each agent is a cone with a pentagonal base and a hue determined by the agent's species (red, blue, or purple). The color of an agent is dimmed in inverse proportion to its energy — agents with nearly maximal energy glow brightly while those with nearly zero energy are almost black. "Rebirth" events are clearly visible as flashes from black to bright colors.

Agent cones are oriented in the direction of their velocity vectors. This often produces an appearance akin to swimming or to "swooping" birds, particularly when agents are moving quickly.

Energy sources are flat, bright yellow pentagonal disks that hover at a fixed distance above the floor and occasionally glide to new, random positions within a fixed distance from the center of the world. An automatic camera control algorithm adjusts camera zoom and targeting continuously in an attempt to keep most of the action in view.

Commonly available hardware is sufficient for fluid action and animation—most of our development work is conducted on Apple Macintosh G3 and G4 computers with stock video hardware.

Figure 1 shows a snapshot of a typical view of the SwarmEvolve world. Animations showing typical action sequences can be found on-line at http:// hampshire.edu/lspector/alife8-visualization.html.

SwarmEvolveMusic

For our first experiments with sound in SwarmEvolveMusic we used the QuickTime Musical Instruments interface to create musical notes whenever certain events occurred.[1] For example, we generate a note for all "feeding" events (when an agent collides with an energy source).

We assign a different instrument to each of the three species, with one producing a clearly tonal musical timbre (the "Synth Voice" instrument), one producing a chirping sound ("Bird Tweet"), and one producing a more complex timbre ("Guitar Fret Noise"). The pitch of each note is determined by the energy level of the agent, with higher energy agents producing lower pitches.

Because agents often settle into an energy source and feed continuously for an interval of time, and because this would otherwise produce an unmanageably large number of nearly simultaneously sounding notes, we also introduced a short "refractory period" — when an agent sounds a note a timer is started and subsequent feeding events will not sound a new note until the refractory period has elapsed. We also produce a note (using the "Woodblock " instrument) for each "landing" event.

The musical result is complex and engaging. As the agents evolve they adopt a variety of flocking and feeding behaviors that produce distinct musical effects. For example, when a species is flocking tightly and is strongly directed toward the energy sources its agents will feed in nearly simultaneous bursts, producing dense clouds of notes. Transitions between different musical patterns are often pleasantly gradual, mirroring the gradual evolutionary dynamics of the underlying simulation. Because the three species produce clearly distinguishable sounds, the overall balance of timbres shifts gradually, with different moods established when one or another species is more successfully feeding. The tonal species ("Synth Voice"), when feeding intensively, produces thick, harmonious chords, while the "Guitar Fret Noise" species, in similar circumstances, produces complex percussive rhythms. The "Bird Tweet" species produces a range of bird-like sounds that are appropriate for the bird-like flying motions of all of the agents.

In additional experiments we have used the sine wave generator plugin to render the actual trajectories of the agents in audio. We have experimented with several variations of this idea. For example we have tried having each species generate a sine wave and we also tried having each agent generate a sine wave. We have tried using a variety of features of each species or agent to determine

---

[1] Our inspiration to do this derived from conversations with Tim Blackwell, who has used swarm systems in different ways to produce music (Graham-Rowe 2002).
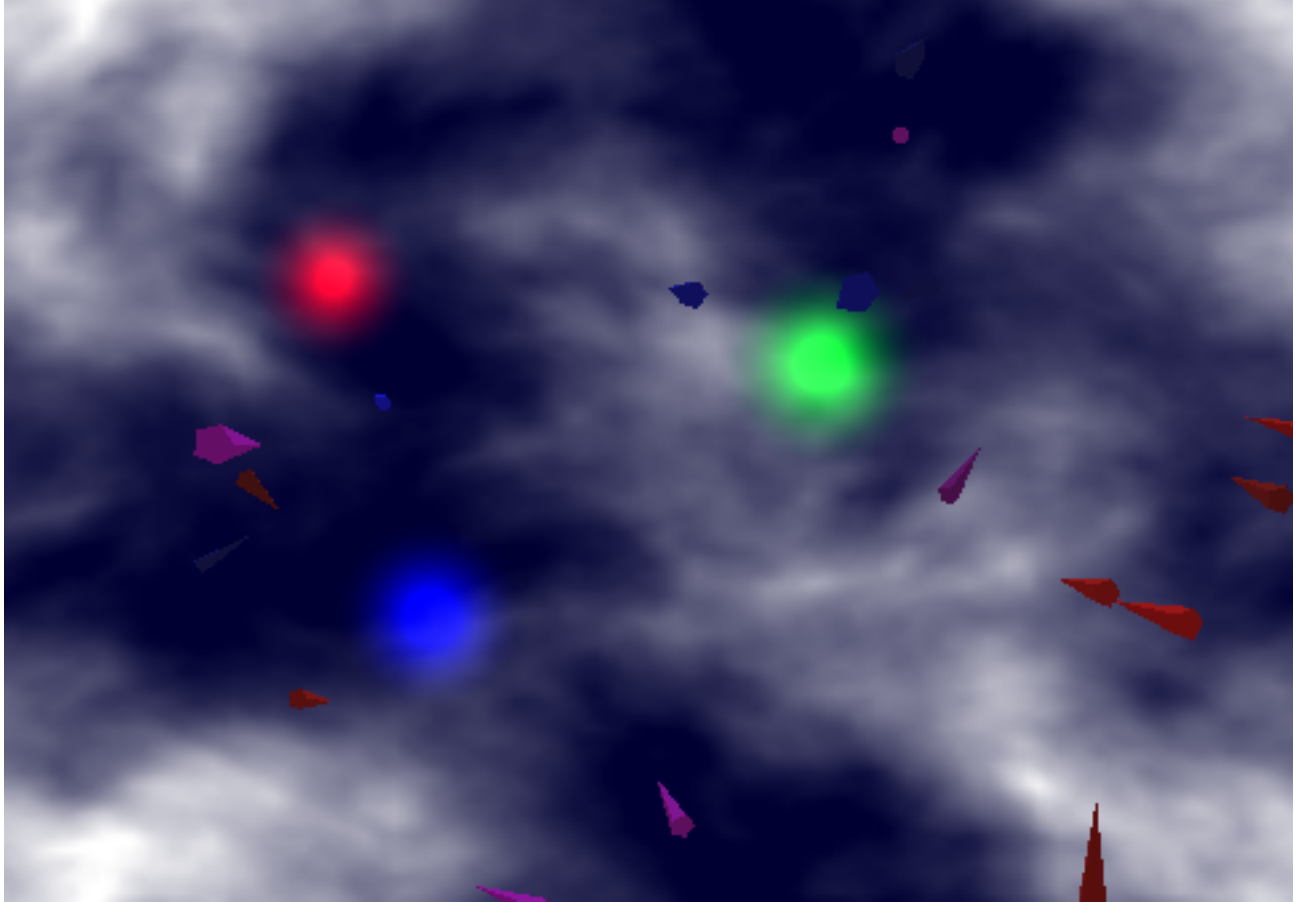
Figure 2: A view of the SWARMEVOLVEMUSICCEILING system, a version of SWARMEVOLVEMUSIC re-designed for projection on the ceiling of a night club. A color version of this figure can be found at http://hampshire.edu/lspector/alife8-music.html.

the frequency of the sine waves, for example height, distance from the center of the world, energy, etc. To date, the most musically satisfying results have resulted from having each agent produce a sine wave and by having the frequency of the sine wave determined by the distance from the agent to the closest energy source; to some extent this renders in audio the visually graceful "swooping" agent trajectories.

In a final set of experiments we have allowed agents to sense (via simulated hearing) the sounds produced by other agents (e.g. when feeding) and we have included new terms in the motion formula that allow for agents to be attracted or repelled by the sounds of others. We intend to use this capability to study the evolution of communication systems, but we will also be alert to musical applications of this enhanced environment—since the sounds will actually play a causal role in the world we expect that they will better reflect the simulation's evolutionary dynamics.

BREVE's object-oriented foundation makes it relatively easy to re-engineer the aesthetics of a piece while re-taining the underlying simulation dynamics. As an example of this, we were asked to adapt a version of SWARMEVOLVEMUSIC for projection on the ceiling of a night club. Because the angled view of the floor (as in Figure 1) was disorienting when projected on the ceiling, we redesigned it to have a transparent floor and placed the camera beneath the floor looking up. The effect is as though one were looking at the scene through a sky-light, upon which the agents can land. We simultaneously adjusted other features of the simulation (the color and shape of the energy sources, the color of the background, etc.) to make them more appropriate for the application. Figure 2 shows a view of the resulting system, which we call SWARMEVOLVEMUSICCEILING.

CA3D

Purely as an aesthetic exercise the first author (Spector) has developed a series of 3-dimensional cellular automata that use BREVE's ability to draw 3-dimensional grids of
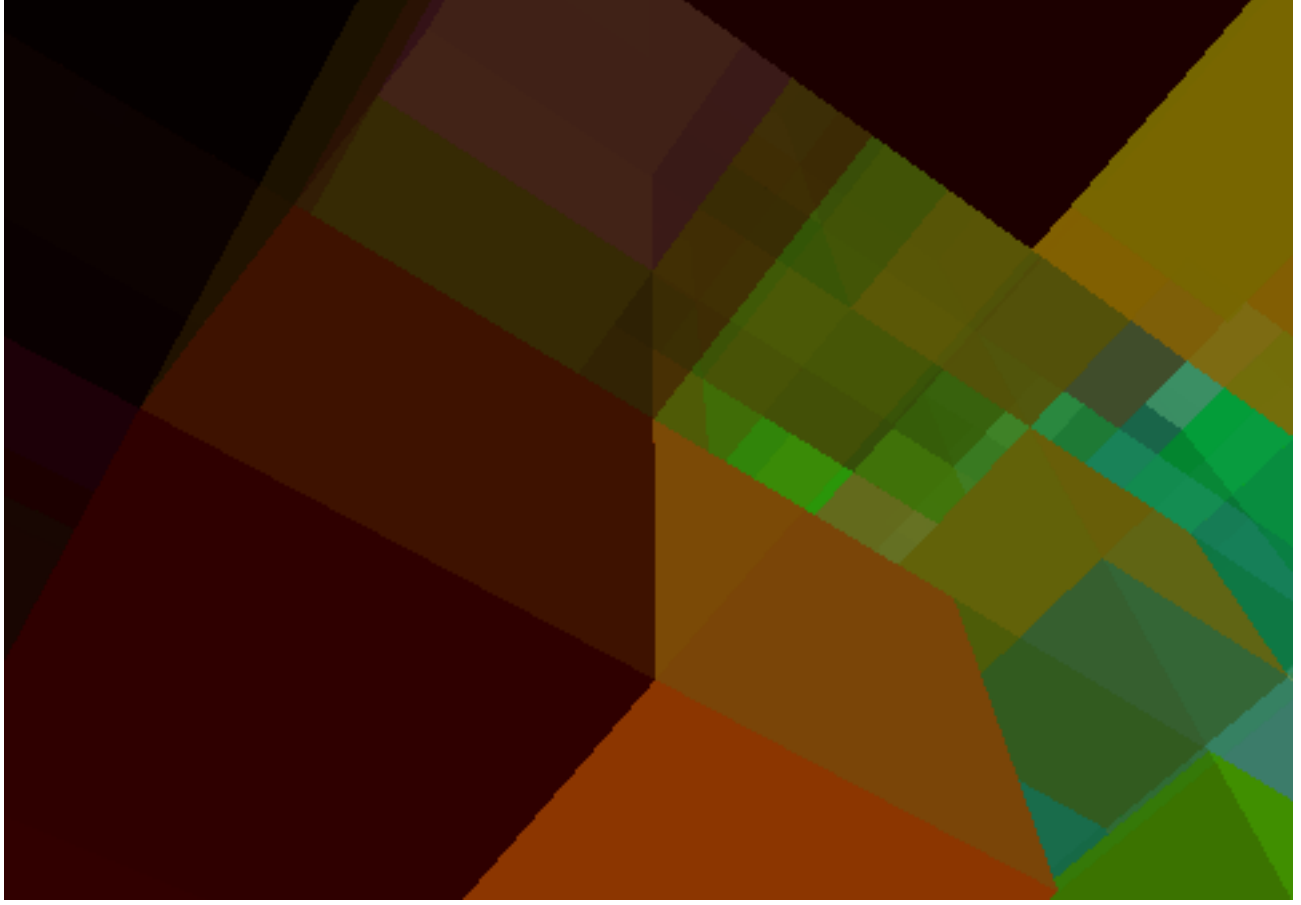
Figure 3: A view of CA3D. A color version can be found at http://hampshire.edu/lspector/alife8-music.html.

colored, partially transparent cube-shaped "patches."[2] These automata extend 2-dimensional work (called "autonomous color studies") done by Spector in 1999. In the new, 3-dimensional cellular automata (called CA3D) the color and transparency of each patch is determined at each time step from the color and transparency of the patch's six face-sharing neighbors.[3] In the simplest case a single formula, f, determines the values of the patch's four components (r = red, g = green, b = blue, and t = transparency) at time step i as follows (where c is a constant between 0 and 1, and R, G, B, and T are the average values of r, g, b, and t, respectively, for the patch's six neighbors):

$$r_i = cr_{i-1} + (1-c)f(G_{i-1}, B_{i-1}, T_{i-1})$$

$$g_i = cg_{i-1} + (1-c)f(B_{i-1}, T_{i-1}, R_{i-1})$$

$$b_i = cb_{i-1} + (1-c)f(T_{i-1}, R_{i-1}, G_{i-1})$$

$$t_i = ct_{i-1} + (1-c)f(R_{i-1}, G_{i-1}, B_{i-1})$$

In addition, the camera position and zoom are changed gradually and randomly as the system runs. In some cases we also use multiple f functions, with randomly occurring transitions from one f to another.

The CA3D system is capable of producing a range of visually compelling dynamic displays, the details of which depend on the choice of f and c. A snapshot of one version of the system is shown in Figure 3.

CA3D Music Projects

To experiment with musical *input* to complex adaptive systems we created a version of CA3D that works as an iTunes plugin.[4] At each time step we grab audio spec-

---

[2]For more information on cellular automata in general see (Wolfram 2002). The literature on cellular automata is large and includes many previous works both on 3D cellular automata and on artistic/musical applications of cellular automata—a web search using the terms "cellular automata," "art," "music," and "3d" should return a large number of relevant resources.

[3]The grid is considered to be torroidal so that every patch has six face-sharing neighbors.

[4]iTunes is audio software that ships with Apple Macintosh computers; see http://www.apple.com/itunes/.
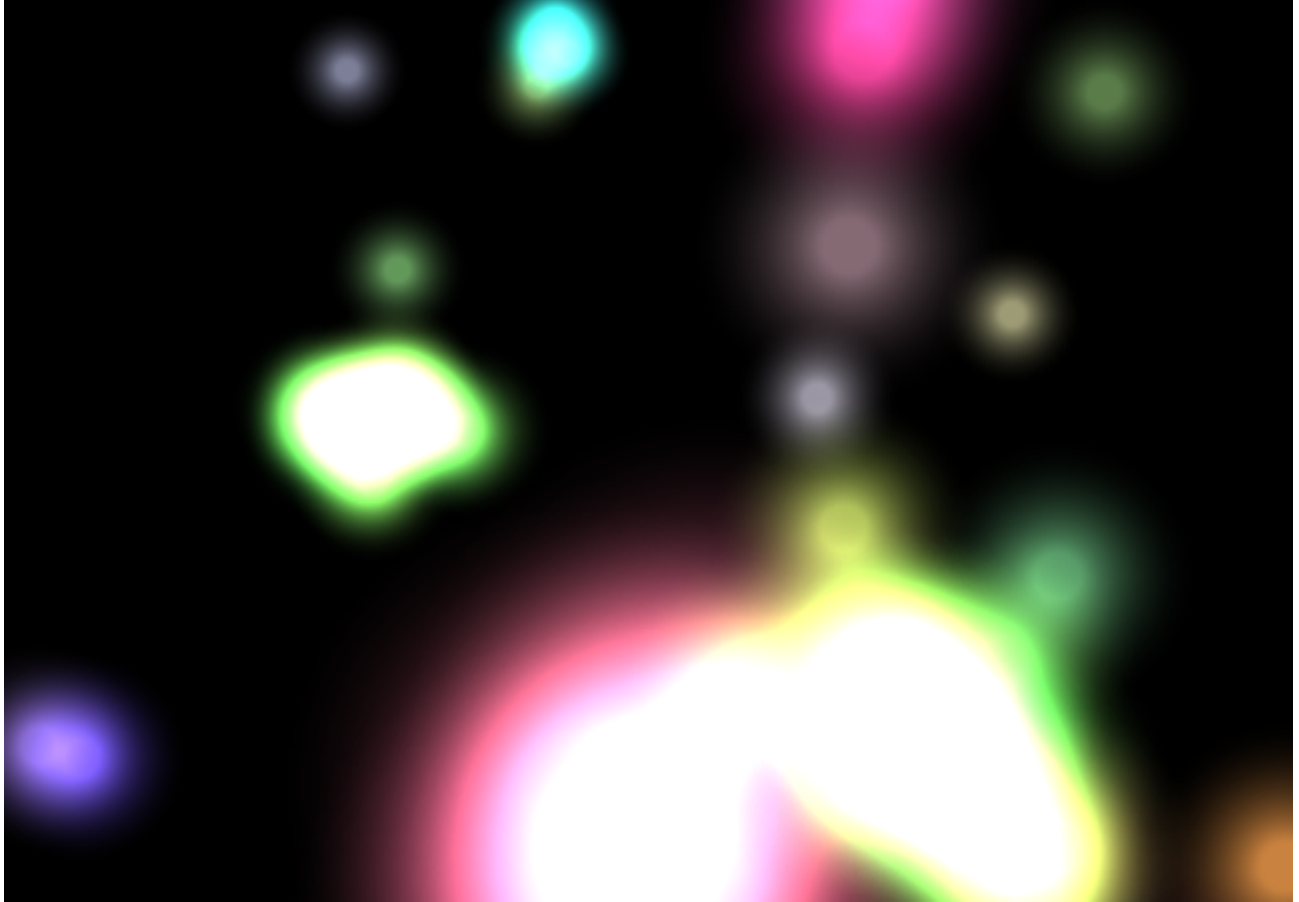
Figure 4: A view of SWARMCA. A color version can be found at http://hampshire.edu/lspector/alife8-music.html.

trum data from the iTunes interface and use the spectrum values to set the transparencies of the patches — each patch is assigned to a particular band of the spectrum. Because of the way in which transparency values serve as arguments to f for the calculation of color values, input to the transparency channel has a clear (though slightly delayed) impact on patch colors.

The result is a display that is coupled to the musical input but which nonetheless has a dynamics of its own. The musical input provides a basic pulse to the display, which also changes in general appearance when major transitions in the harmonic content of the music occur. But the effects of the underlying cellular automaton dynamics are still quite visible, and the visuals never appear to be direct transcriptions of the music into graphics.

We have also experimented with the use of CA3D as a music generator. In our initial experiments we have used only the BREVE sine wave generator, with the frequency of each sine wave component determined at each time step from a feature of a patch or set of patches. Interesting effects result from several variations that we have tried, for example from using patch transparency

values to set the frequencies. We intend to continue to experiment with variations on this theme.

Combined SWARM and CA Projects

The components of the various systems described above can be combined in a variety of ways. As one example, we began with a simple (non-evolutionary) SWARM system and used BREVE's "lightmap" feature to add visual interest—this causes the agents to appear as fuzzy blobs that seem to fuse when they are close to one another (with their colors combining additively, producing white if there are enough components). We then added a cellular automaton component to the color dynamics, determining the color of each agent from those of its neighbors using the methods described above for CA3D, with the exception that in this case the neighbors are dynamically determined. The result is strikingly organic and includes interesting color dynamics. A snapshot of this system, which we call SWARMCA, is shown in figure 4.

Using the iTunes plugin architecture described above, we have also made versions of SWARMCA that respond to music by adjusting color components and/or constants in the swarm motion formula in response to musical dy-

namics and spectrum information.

Conclusions

We believe that the experiments described in this paper are aesthetically successful and we plan to continue our work in this area. Our sense is that the success of these experiments is due in part to the use of complex adaptive systems which have intrinsically interesting dynamics. In some sense the pieces that we have produced "have a life of their own" and the life-like behavior adds a compelling dimension to the visual and audio output.

References

Graham-Rowe, D. 2002. Insects make a buzz on the music scene. *New Scientist* 174:24.

Klein, J. 2002. BREVE: a 3d environment for the simulation of decentralized systems and artificial life. In *Proceedings of Artificial Life VIII, The 8th International Conference on the Simulation and Synthesis of Living Systems.* The MIT Press.

Reynolds, C. W. 1987. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics* 21(4):25–34.

Spector, L., and Klein, J. 2002. Evolutionary dynamics discovered via visualization in the BREVE simulation environment. In *Proceedings of the Workshop 'Beyond Fitness: Visualising Evolution' at The 8th International Conference on the Simulation and Synthesis of Living Systems, Artificial Life VIII.*

Wolfram, S. 2002. *A New Kind of Science.* Champaign, IL: Wolfram Media, Inc.