The Electric Sheep and their Dreams in High Fidelity

Scott Draves

Abstract

Electric Sheep is a distributed screen-saver that harnesses idle computers into a render farm with the purpose of animating and evolving artificial life-forms known as *sheep*. The votes of the users form the basis for the fitness function for a genetic algorithm on a space of abstract animations. Users also may design sheep by hand for inclusion in the gene pool.

Dreams in High Fidelity is a new manifestation of the sheep. Physically it consists of a small computer driving a framed liquid crystal display. It plays sheep selected, edited, and sequenced by the artist at triple the resolution and more stately motion than the screen-saver. Sales of this fine art will support the free network, forming a symbiotic relationship.

1 Introduction

Electric Sheep [Draves 1999] was inspired by SETI@Home [Anderson et al. 2002] and has a similar design. Both are distributed systems with client/server architecture where the client is a screen-saver installable on an ordinary PC. Electric Sheep distributes the rendering of animations, each one 128 frames long and known as a sheep.

Besides rendering frames, the client also downloads completed sheep and displays them to the user. The user may vote for the currently displayed sheep by pressing the up arrow key. Each sheep is specified by a genetic code comprised of about 240 floating-point numbers. The codes are generated by the server according to a genetic algorithm where the fitness is determined by the collective votes of the users. This is a form of aesthetic evolution, a concept first realized by Karl Sims [1991].

All the software is open source and users may participate in the network freely and anonymously. Anyone can download and install it. When their computer is idle and goes to sleep, the sheep animations appear, and in parallel the computer goes to work creating new sheep and sharing its results with all other users.

One can download additional software to manually design genomes and post them to the server where they join the flock [Townsend 2004]. These human designers collaborate and compete with the genetic algorithm (GA). The voting on 6000 sheep from five months of 2005 was about equally divided between human and GA designed sheep, more precisely the *creative amplification factor* of the GA was measured at 2.1 [Draves 2005]. See Figure 1 for two fragments of families of sheep from this time.

[0]Delivered as the keynote at the International Symposium on Non-Photorealistic Animation and Rendering, Annecy France 2006/06.

2 Dreams in High Fidelity

Physically *Dreams In High Fidelity* consists of a small computer driving a large 1280×720 liquid crystal display. The computer creates a continuously morphing, non-repeating, abstract animation.

The animations are rendered by the Electric Sheep, but at triple the resolution and six times more frames per sheep. The image quality is striking on a large display. It requires 20 times the computation to make a high fidelity sheep.

The artist selects his favorite sheep from the archives and public flock, and sends them back to be re-rendered at high fidelity: heaven for an electric sheep. This new flock has grown to 50GB, totaling 8 hours if played end-to-end, and requiring over 1 million CPU hours to render.

Each *Dreams* made has a slightly different flock and includes sheep unique to it. Each flock resonates differently on playback so its sheep have a unique frequency distribution. The artist tunes each flock by adding and removing transitions between sheep to balance the distribution.

The *Dreams* has a symbiotic relationship to the free screen-saver. The free version provides the design laboratory and gene pool from which the best sheep are extracted. It also provides the distributed supercomputer needed to realize the high fidelity content. On the flip side, the hifi version brings the income required to keep the whole project in operation, and develop it further.

3 The Genetic Code

Each image produced by Electric Sheep is a fractal flame [Draves 1992], a generalization and refinement of the Iterated Function System (IFS) category of fractals [Barnsley 1988]. The genetic code used by Electric Sheep is just the parameter set for these fractals. It consists of about 240 floating-point numbers.

A classic IFS consists of a recursive set-equation on the plane:

$$S = \bigcup_{i=0}^{n-1} F_i(S)$$

The solution S is a subset of the plane (and hence a two-tone image). The F_i are a small collection of n affine transforms of the plane.

The equation is normally solved with the chaos game, which generates a large set of particles (610M for the screen-saver and 2.4G for *Dreams*) by starting anywhere then iteratively picking i at random and applying F_i.

A fractal flame is based on the same recursive equation, but the transforms may be non-linear and the solution algorithm produces a full-color image. For how the particles are rendered into the image, see section 4.

The transforms are linear blends of a set of 31 basis functions known as *variations*. The variations are composed with an affine matrix, like in classic IFS. So each transform F_i is:

$$F_i(x,y) = \sum_j v_{ij} V_j(a_i x + b_i y + c_i, d_i x + e_i y + f_i)$$

where v_{ij} are the 31 blending coefficients for F_i, and a_i through f_i are 6 affine matrix coefficients. The V_j are the variations, here is a partial list:

$$\begin{aligned} V_0(x,y) &= (x,y) \\ V_1(x,y) &= (\sin x, \sin y) \\ V_2(x,y) &= (x/r^2, y/r^2) \\ V_3(x,y) &= (r\cos(\theta+r), r\sin(\theta+r)) \\ V_4(x,y) &= (r\cos(2\theta), r\sin(2\theta)) \\ V_5(x,y) &= (\theta/\pi, r-1) \end{aligned}$$

where r and θ are the polar coordinates for the point (x,y) in rectangular coordinates. V_0 is the identity function so this space of non-linear functions is a superset of the space of linear functions. See [Draves 2004] for the complete list.

There are 3 additional parameters: color (see below), and weight and symmetry, not covered here. Together these 40 (31 for v_{ij} plus 6 for a_i to f_i plus 3 is 40 total) parameters make up one transform, and are roughly equivalent to a gene in biological genetics. The order of the transforms in the genome does not effect the solution image. Many transforms have visually identifiable effects on the solution, for example particular shapes, structures, angles, or locations.

Normally there are up to 6 transforms in the function system, making for 240 (6×40) floating-point numbers in the genome. Note however that most sheep have most variational coefficients set to zero, which reduces the effective dimensionality of the space.

3.1 Animation and Transitions

The previous section described how a single image rather than an animation is defined by the genome. To create animations, Electric Sheep rotates over time the 2×2 matrix part (a_i, b_i, d_i, and e_i) of each of the transforms. After a full circle, the solution image returns to the first frame, so sheep animations loop smoothly. Sheep are 128 frames long, and by default are played back at 23 frames per second making them 5.5 seconds long.

The client does not just cut from one looping animation to another. It displays a continuously morphing sequence. To do this the system renders transitions between sheep in addition to the sheep themselves. The transitions are genetic crossfades based on pairwise linear interpolation, but using a spline to maintain C^1 continuity with the endpoints.

4 Rendering

How are the particles rendered into an image? Note that unlike a traditional particle system, there is no frame-to-frame coherence among these particles, or interaction between them. They are all generated fresh from a random number seed each frame. They are really samples of a stochastic system.

Normally in IFS the particles are drawn by simple accumulation: the particles are plotted and their color is added into an accumulation buffer.

This kind of linear mapping of density to gray values is unsatisfying because of the large dynamic range of densities. Like many natural systems, the densities are often distributed according to a power law (frequency is proportional to an exponent of the value). A few pixels are much brighter than the rest.

The display of high dynamic range images like these is called *tone mapping*. Flames use a logarithm followed by a gamma factor for a tone map.

This log-density mapping is the source of a 3D illusion. On sight people often guess that the sheep are rendered in 3D, but the algorithm works strictly with a 2D buffer. However, where one branch of the IFS crosses another, one may appear to occlude the other if their densities are different enough because the lesser density is inconsequential in sum. For example branches of densities 1000 and 100 might have brightnesses of 30 and 20. Where they cross the density is 1100, whose brightness is 30.4, which is hardly distinguishable from 30.

Another problem with just plotting particles is that at low densities (less than 1 particle per pixel) noise is quite visible. The latest version of the flame algorithm uses a density estimator that increases the filter size dynamically to compensate [Suykens and Willems 2000]. It also makes motion blur much cheaper by approximating and running it before the estimator, instead of running the estimator once for each temporal sample. The renderer also uses 2×2 spatial oversampling.

5 Coloring

Naturally we want to use a palette or color-map which we define as a function from [0,1] to (r,g,b) where r, g, and b are in [0,1]. A palette is classically specified with an array of 256 triples of bytes.

To achieve this we assign a color coordinate k_i to each function F_i and add an independent coordinate k to the (x,y) of the chaos game. The formula for the color coordinate is very simple: for each iteration, $k = (k + k_i)/2$.

This has the important property that colors are continuous in the final image: particles that are from the same part of the attractor have the same color. Particles are colored according to the history of the choices made for the functions in the chaos game.

Coloring is normally not a problem with particle systems because particles have state from frame to frame.

6 Conclusion

I look forward to many more generations of sheep at ever higher resolutions, with more expressive genetic codes, in three dimensions, responding to music, performing feats not yet imagined.

References

ANDERSON, D., ET AL. 2002. Seti@home: An experiment in public-resource computing. *Communications of the ACM 45*, 56–61.

BARNSLEY, M. 1988. *Fractals Everywhere*. Academic Press.

DRAVES, S., 1992. The fractal flames. Open source release on `http://flam3.com`.

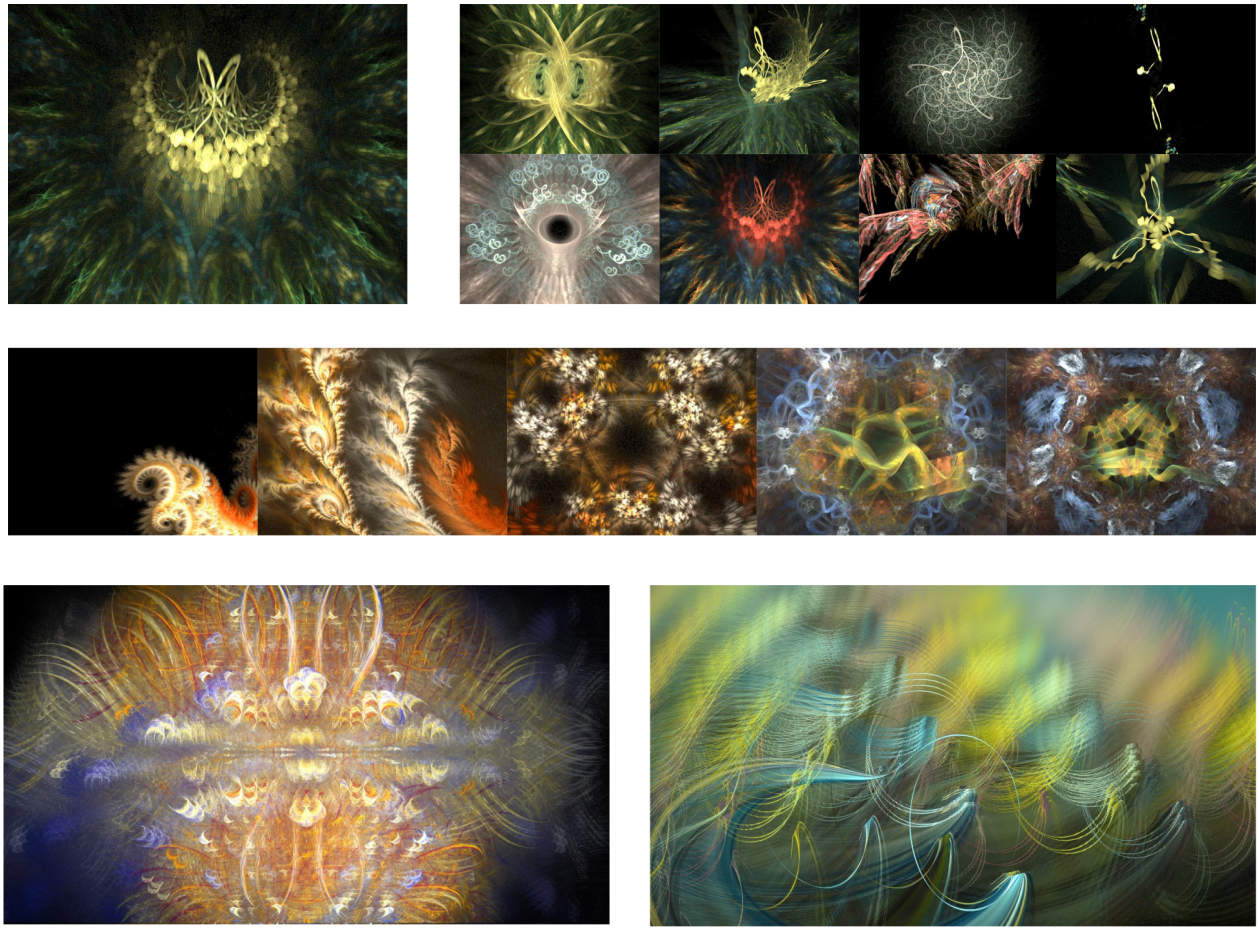

Figure 1: Sheep 15875 of generation 165, on the top-left, was born on August 16 and died 24 hours later after receiving one vote. It was one of 42 siblings. It was reincarnated on October 28 as sheep 29140, received a peak rating of 29, lived 7 days, and had 26 children, 8 of which appear to its right. In the middle are five generations of sheep in order starting on the left. Their numbers are 1751, 1903, 2313, 2772, and 2975. The last is a result of mutation, the previous three of crossover, the first was posted by etomchek. Bottom left is *Dream 191.21054* and bottom right is *Dream 198.19616*.

DRAVES, S., 1999. The electric sheep screen-saver. Open source release on `http://electricsheep.org`.

DRAVES, S., 2004. The fractal flame algorithm. `http://flam3.com/flame.pdf`.

DRAVES, S. 2005. The electric sheep screen-saver: A case study in aesthetic evolution. In *Applications of Evolutionary Computing, LNCS 3449*, Springer Verlag.

SIMS, K. 1991. Artificial evolution for computer graphics. In *Proceedings of SIGGRAPH*, ACM.

SUYKENS, F., AND WILLEMS, Y. 2000. Adaptive filtering for progressive monte carlo image rendering. In *Eighth International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media*.

TOWNSEND, M., 2004. Apophysis. `http://apophysis.org`.