The Interpretation of Dreams: An Explanation of the Electric Sheep Distributed Screen-Saver

by Scott Draves, Ph.D.
DreamWorks SKG, San Francisco, USA

Intro

The name *Electric Sheep* comes from Philip K. Dick's novel *Do Androids Dream of Electric Sheep*. It realizes the collective dream of sleeping computers from all over the Internet. *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. The project is an attention vortex. It illustrates the process by which the longer and closer one studies something, the more detail and structure appears.

1. The Client

When the software is activated, the screen goes black and an animated 'sheep' appears. In parallel, the screen-saver client contacts the server and joins the distributed computation of new sheep, an idea inspired by the SETI@home project. [1]

The screen-saver is a window into a visual space shared among all users. Clients render JPEG frames and upload them to the server. When all the frames are ready, the server compresses them into an MPEG animation. Each animation is the phenotype of an artificial organism, an 'electric sheep.' Clients download the MPEG sheep and display them one after another in a continuous, ever-changing sequence.

About once every five minutes a new sheep is born and distributed to all active clients. Each sheep is an animated fractal flame. [2] Example still frames appear in Figure 1. The shape is specified by a string of 120 real numbers -- a genetic code of sorts. Some of the codes are chosen at random by the server with heuristics to avoid malformed sheep, somewhat like spontaneous abortion. The rest are derived from the current population according to a genetic algorithm with mutation and cross-over.

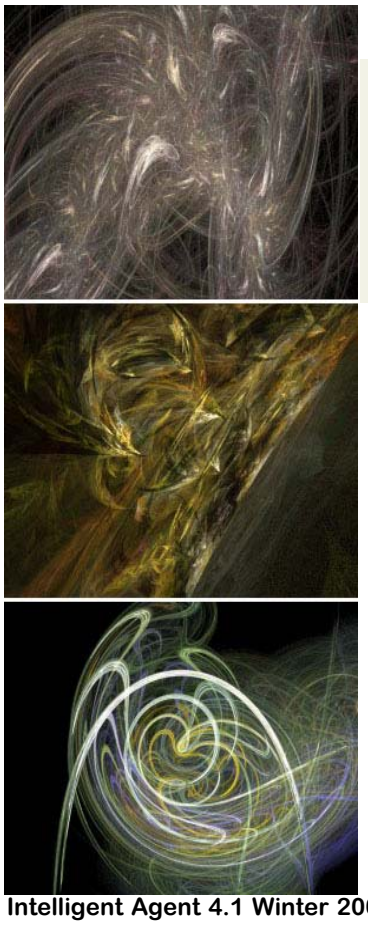

Figure 1. Example still frames

Electric Sheep...realizes the collective dream of sleeping computers from all over the Internet; a distributed screen-saver that harnesses idle computers into a render farm with the purpose of animating and evolving artificial life-forms

2. The Server

The server has a web interface for people in addition to the one used by clients. It allows users to see and download the currently living sheep as well as monitor the rendering of new ones.

Clients can identify themselves with a nickname and URL and see exactly which frames are theirs. The server generates rankings of nicknames and IP addresses by the number of frames contributed. Users can visit each other's web pages and find out who else is in the community.

Normally, *Electric Sheep* is very reliable and runs for weeks without assistance, but with new versions come new bugs, and at these times, the ability to tweak the server live and online is essential to keeping the flock healthy. By entering a password, a user can become an administrator and delete bad frames, entire sheep, or block clients by address. An administrator can also inject a particular genetic code into the system, for example, to resurrect a sheep from the code stored in a previously captured MPEG file.

3. Life, Death, and Interpolation

A sheep's life is finite. I only have enough disk quota to keep about thirty alive on the server. Old sheep are deleted without a trace. Users may vote for a sheep by pressing the up arrow key when that sheep is displayed on their screen. [3] Popular sheep live longer, and are more likely to reproduce. Hence, the users' preferences provide the fitness function for an aesthetic evolutionary algorithm, an idea first realized by Karl Sims. [4]

The fractal flame algorithm takes the genetic code and produces a still image, the first frame of the animation. The genetic code contains blending coefficients and 2D affine transformations. The animation of each sheep is produced by rotating all its transforms by 360 degrees. As a result, the shape has returned to its original state by the end of the rotation, and hence each sheep is an animation that loops.

The parameter space of sheep is continuous, and the server generates smooth transitions between sheep by interpolating in the genetic space. The original interpolation method resulted in C1 discontinuities (angles or jerks in the motion) at the beginning and end of each transition because it was pair-wise linear, and the direction of rotation differed from the direction to the next sheep. Cassidy Curtis suggested the method used to solve this problem: use pair-wise linear interpolation, but make the end-points rotating sheep instead of fixed.

The set of animations on the client form a graph, as illustrated by the diagram. This kind of diagram is used on the web server to represent the state of the flock. Each arrow represents an animation. The nodes represent key-frames. A sheep animation is an arrow with the same keyframe at its head and its tail, because sheep are loops. The client plays the animations by following the arrows head to tail and branching and to seek out new territory.

4. Measurements and Statistics

Clients typically store about 100 sheep totaling 9 minutes of animation and taking 250 megabytes of disk space. The server uses a free MPEG2 video encoder at a resolution of 640 by 480 pixels and 5 megabits per second. Clients typically take between 20 and 80 seconds to render each frame.

The high resolution sheep available from the web pages and in the video documentary were born on the sheep server, then the parameters were tweaked to increase quality, and finally they were re-rendered and compressed off-line to avoid MPEG compression artifacts.

In ten days, at the end of October 2001, clients from 650 unique IP addresses contributed frames to the server. Multiple users may share an address, and no attempt is made to uniquely identify clients, so the real user count is unknown. At that time about 150 clients were participating in the render farm at any given time. In the first 12 days of March 2003, clients from 4900 unique IP addresses downloaded animations from the two operational sheep servers (the second server supported legacy clients). In November 2003, there were 135 clients simultaneously rendering frames, and 1700 workers within one week. User growth is currently slow but steady. When the OSX and Windows versions leave beta, I expect a surge in clients again.

The user base is limited to those with high-bandwidth, always-on connections to the Internet such as DSL, cable-modem, or university or corporate networks. Because the client uses only the http protocol on port 80 and it supports web proxies (via the underlying curl library), it can generally be used from behind firewalls and NAT boxes.

5. Development

From August 1999 (when the client was created) until October 2001, the *Electric Sheep* client only ran on Linux. At that time, Matt Reda released a Mac OS X client, and the number of clients quickly doubled. Despite many requests, several promises, and one near miss, no working Microsoft Windows version appeared until Nicholas Long delivered a beta version in May 2003.

Linux v2.4 included a substantial upgrade to the core Fractal Flame code, including symmetries, and new variational equations contributed by Ronald Hordijk, as well as the new interpolation technique. The Macintosh client was not updated and its users remained cut off from the server until Mathew provided the crucial updates in October 2003.

In October 2002, the domain name was hijacked by a competing 'electric sheep' site. Fortunately, after a hacking and legal scuffle, the domain has been returned and the site is back in operation, though the user base suffered a setback. Both clients and the server are open source and there is a developer community as well as a user community. The whole system, centered around the electricsheep.org web site, has its own buzz. The users and developers exchange messages through the discussion forum and email, and clients and servers exchange images and animations. There is an evolving ecology of agents, codes, and protocols.

6. The Vortex

Electric Sheep investigates the role of 'experiencers' in creating the experience. If nobody ran the client, there would be nothing to see. Eons ago, tiny irregularities in our universe became centers of accretion and eventually grew into stars. A parallel process unfolds in cyberspace.

It starts with an idea.

The sheep system exhibits increasing returns on each of its levels. As more clients join, more computational muscle becomes available, and the resolution of the graphics may be increased, either by making the sheep longer, larger, or sharper. As more people participate in the system, the appearance of the images improves. Likewise, as developers focus more of their attention on the source code, the client and server themselves become more efficient, grow new features, and are ported into new habitats. The project gains momentum, and attracts more developers. And as more users vote for their favorite sheep, the evolutionary algorithm more quickly distills randomness into eye candy. Perhaps attention acts on information the same way gravity acts on mass: attraction begets attraction and a positive feedback loop is formed.

7. The Future

Electric Sheep is open-ended and very much a work in progress. For example, the server is currently a bottleneck because it must compress and deliver large MPEGs to so many clients. But if clients act as servers and become a true peer-to-peer network, the compression and bandwidth load could be distributed as much as the computational load already is. This project is currently underway, and I have selected gnutella [5] as the P2P protocol because it has mature open-source implementations. A central server will still be used by clients to find each other and to coordinate basic animation parameters such as resolution and quality. The rendering, compression, evolution, and voting can all be fully distributed.

The architecture is not specific to fractal flames, and the protocol should support multiple alternate renderers. I am seeking collaborators to contribute their own generative animation software.

I believe the free flow of code is an increasingly important social and artistic force. The proliferation of powerful computers with high-bandwidth network connections forms the substrate of an expanding universe. The electric sheep and we their shepherds are colonizing this new frontier.

Acknowledgements

I would like to thank Mike Kuniavsky, Nick Thompson, Katherine Mills, and especially Maribeth Back for their input on this paper. Thanks to Dean Gaudet for hosting the web sites and Carnegie Mellon University School of Computer Science for providing the bandwidth for the heavy lifting. I'd also like to send Kudos out to everyone who sent me a patch, bug report, or even just one vote.

References

[1] SETI@home searches for a signal from extra-terrestrials in radio-telescope data. It consists of a screen-saver client that is downloaded and installed by users all over the world, and a server that divides-up the data among the clients and collects the results. It puts idle computers to work. SETI@home is the original distributed screen-saver, and its architecture is the inspiration for *Electric Sheep's*. See http://setiathome.ssl.berkeley.edu.

[2] Fractal flames are the output of a particular Iterated Function System (IFS) fractal rendering algorithm created by the author in 1992. Each image is a histogram of a two-dimensional strange attractor. The flame algorithm contains three innovations: (a) It uses a collection of special functions that are composed with the usual affine matrices. (b) The intensity of each pixel is proportional to the logarithm of the density of the attractor rather than a linear relationship. (c) The color is determined by appending a third coordinate to the chaotic system and looking it up in a palette. Great care is taken to correctly anti-alias the image, both spatially and temporally (with motion blur). Flame is designed to produce images without artifacts, and to reveal as much of the information contained in the attractor as is possible. For more information, see http://flam3.com and the unpublished paper "The Fractal Flame Algorithm," available there.

[3] Pressing the up or down arrow key transmits a vote for or against the currently displayed sheep. The server's web interface also has voting controls. In Linux, voting by key-press requires a special version of xscreensaver (part of the gnome desktop interface) to work, so it is not widely (if at all) deployed. Voting works correctly in the Mac OSX and Windows versions. The next version of the Linux client will include the modified version of xscreensaver.

[4] "Artificial Evolution for Computer Graphics," Karl Sims, Computer Graphics (Siggraph proceedings), July 1991, available from http://www.genarts.com/karl/genetic-images.html.

[5] Gnutella was initially developed by Justin Frankel of Nullsoft. AOL (the parent company) pulled the plug in early 2000 and ordered Nullsoft to cease all development. The source code of Gnutella was intended to be eventually released under the GPL (thus the "GNU" in its name), but those plans were crushed by AOL's early intervention. Despite the crackdown, the protocol was reverse-engineered and numerous clients appeared. Today it is by far the most popular file-sharing network.