The Electric Sheep Screen-Saver: A Case Study in Aesthetic Evolution

Scott Draves

Spotworks, San Francisco CA, USA

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 fractal animations. Users also may design sheep by hand for inclusion in the gene pool. This paper describes the system and its algorithms, and reports statistics from 11 weeks of operation. The data indicate that Electric Sheep functions more as an amplifier of its human collaborators' creativity rather than as a traditional genetic algorithm that optimizes a fitness function.

1 Introduction

Electric Sheep [5] [6] was inspired by SETI@Home [1] 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 fractal animations. Each animation is 128 frames long and is 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 160 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 [9] and analyzed by Alan Dorin [3].

This is how Electric Sheep worked until March 2004, when a new source of genomes appeared: Apophysis [10]. Apophysis is a traditional, stand-alone Windows GUI to the sheep genetic code primarily intended for still-image design, but useful for key-frame animation. Besides a traditional direct manipulation interface where the user drags sliders and types numbers into labeled fields, it includes a Sims-style mutation explorer.

In March, Townsend and Draves connected this application to the Electric Sheep server. A simple menu command causes the current genome to be posted to the server, rendered, and distributed to all active clients. If the resulting animation receives votes it may reproduce and interbreed with the artificially evolved population.

Not surprisingly, these posted genomes proved much more popular than the purely random ones. And as they are subject to mutation and crossover, the genetic algorithm creates variants of them. One can compare the total amount of quality animation to the amount that was human designed. This ratio is the *creative amplification factor* of the system, as discussed in Section 6.1.

The rest of this paper is structured as follows: Section 2 describes the architecture and implementation of Electric Sheep. Section 3 briefly explains the concept and artistic goals of the project. Section 4 surveys the genetic code on which all sheep rendering and evolution is based, and Section 5 explains the genetic operators and the specifics of the evolutionary algorithm. Section 6 reports empirical results of running this system for over 11 weeks during which time more than 6000 sheep were born. Section 7 puts this work in context of past research, and Section 8 concludes.

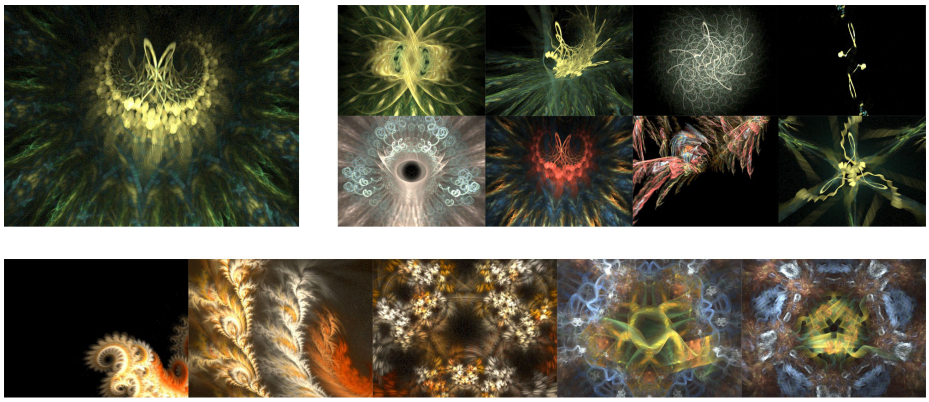

Fig. 1. Sheep 15875, on the top-left, was born on August 16 and died 24 hours later after receiving one vote. It was one of 42 siblings. It was reincarnated on October 28 as sheep 29140, received a peak rating of 29, lived 7 days, and had 26 children, 8 of which appear to its right. Below are five generations 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.

2 Architecture and Implementation

Electric Sheep has a client/server architecture. The client initiates all communication between them, and if no client were running the server would not run at all.

The client has three main threads. One thread downloads sheep animations from the server to a local disk cache. It downloads those with highest rating first. The default size of the cache is 300Mbytes (enough for 65 animations) but the user may change it. Another thread reads the sheep from the cache and displays them in a continuous sequence on the screen. The third thread contacts the server, receives a genome specifying a frame to render, renders the frame, then uploads the resulting JPEG file.

The server maintains several collections of sheep. Sheep are numbered as they are created and are identified by this sequence number. Freshly conceived genomes start out in the render queue. When all the frames of a sheep have been uploaded, they are compressed into MPEG and deleted, and the sheep is made available for download and

voting. Sheep average 4.6Mbytes each. Eventually the sheep dies (Section 5.2 explains when) and the MPEG file is deleted.

All these sheep are referred to collectively as a generation. Each time the server is reset the database is wiped, all sheep are deleted from the server and from all client caches, the generation number is incremented, and evolution starts fresh. The sheep that are the subject of this paper are members of generation 165.

The server is implemented with two machines in separate colocation facilities. Both are commodity Linux x86 servers running Apache. One runs the evolutionary algorithm, collects frames and votes, compresses frames, and sends genomes to clients for rendering. The other only serves the completed MPEGs. The first server receives 221Kbit/s from the clients and transmits 263Kbit/s to them (measured average of July to October 2004). The MPEG server's bandwidth allocation has varied from 15 to 20Mbit/s, and it uses all of it.

The MPEG server is currently the bottleneck in the system. A future version of Electric Sheep will use a P2P network to distribute this bandwidth load much as the computation load already is.

The client runs on Linux, OSX, and Windows. It uses only the HTTP protocol on port 80 and it supports proxies. However, it does require a broadband, always-on connection to the internet. When the server is not reachable the client's sheep display still works but no new sheep appear.

All the code is open source and is licensed under the GPL (General Public License). The fractal flame utilities are written in C and the server is written in Perl. The clients are written in C, C++, and Objective-C.

3 Concept and Motivation

Electric Sheep is an attention vortex. It illustrates the process by which the longer and closer one studies something, the more detail and structure appears. Electric sheep investigates the role of experiencers in creating the experience. If nobody ran the client, there would be nothing to see. 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. The more people who participate, the better the graphics look. These adjustments are made manually on the server or with new client releases.
- 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.

The votes tell the server which sheep are receiving the most attention. Those sheep are elaborated, expanding the variety and detail of those parts of the fractal space that are most interesting.

There is a deeper motivation however: 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.

4 The Genetic Code

Each image produced by Electric Sheep is a fractal flame [4], a generalization and refinement of the Iterated Function System (IFS) category of fractals [2]. The genetic code used by Electric Sheep is just the parameter set for these fractals. It consists of about 160 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.

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. The transforms are linear blends of a set of 18 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 18 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:

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

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 [4] for the complete list.

There are 3 additional parameters for density, color, and symmetry, not covered here. Together these 27 (18 for v_{ij} plus 6 for a_i to f_i plus 3 is 27 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 162 (6×27) floating-point numbers in the genome. Note however that most sheep have most variational coeffients set to zero, which reduces the effective dimensionality of the space.

4.1 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 pair-wise linear interpolation, but using a spline to maintain C^1 continuity with the endpoints.

Transitions are also 128 frames long. For each sheep created, three transitions are also created: one from another random flock member to the new sheep, one from the new sheep to a random flock member, and another one between two other random members. Most of the rendering effort is spent on transitions.

5 The Genetic Algorithm

There are three parts of the genetic algorithm: the rating system that collects the votes and computes the fitness of individual sheep, the genetic operators used to create new genomes, and the main loop that controls which live and die.

As already mentioned, users can vote for a sheep they like by pressing the up arrow key. If the sheep is alive its rating is incremented. Pressing the down arrow key decrements the rating. Votes for dead sheep are discarded. Users may also vote for or against a sheep by pressing buttons on its web page.

The ratings decay over time. Each day the ratings are divided by four with integer arithmetic rounding down.

5.1 Genetic Operators

There are four sources of genomes for new sheep: random, mutation, crossover and posts from Apophysis. The parents for mutation and crossover operators are randomly picked from the current population weighted by rating. The probability of being selected is proportional to the rating. Sheep that have received no votes have rating zero and so cannot be selected.

random The affine matrix coefficients are chosen with uniform distribution from [-1, 1]. The variational coefficients are set to zero except for one variation chosen at random that is set to one.

crossover The crossover operation has two methods chosen equally often. One method creates a genome by alternating transforms (genes) from the parents. The other method does pair-wise linear interpolation between the two parent genomes where the blend factor is chosen uniformly from [0, 1].

mutation The mutation operator has several different methods: randomizing just the variational coefficients, randomizing just the matrix coefficients of one transform,

adding noise (-10 decibels, or numbers from [-0.1, 0.1]) to all the matrix coefficients, changing just the colors, and adding symmetry.

When applying these three automatic operators, the server renders a low-resolution frame and tests if the image is too dark or too bright. The operator is iterated until the resulting genome passes. For random genomes, 43% are rejected (in a test run 177 tries were required to get 100 passing genomes).

post Human designers may post genomes to the server with Apophysis. The designer is required to submit a password with the genome, but its value is shared on a public email list and is common knowledge there. The genome is checked for syntactic correctness, but the image it creates is not tested.

The server has a queue of sheep and transitions that are currently being rendered. Posted genomes go into this queue. When the queue is left with fewer than 12 sheep, it is filled with genomes derived with one of the three automatic operators. 1/4 of these genomes are random and have no ancestor. The remaining 3/4 are divided equally between mutation and crossover.

5.2 The Main Loop

The server maintains a single flock of sheep and continuously updates their ratings, creates new sheep, and kills off old ones. The server has 510MB of disk space for storing sheep animations, enough for 28 sheep and 83 transitions. Each time a sheep is born, the sheep with the lowest rating is killed to make room. If several sheep are tied for worst, then the oldest is taken (usually several sheep have received no votes and are tied with a rating of zero).

Killing a sheep removes the animation file from the server, but not from clients who may have allocated more disk space to their caches. The other records, including the peak rating, parentage, genome, 16 thumbnails, and the first frame are kept. This archive may be browsed on the server either sorted by peak rating, or as extended family trees.

This on-line or steady-state approach contrasts with the more traditional genetic algorithm's off-line main loop that divides the population into generations and alternates between rating all the individuals in a generation and then deriving the next generation from the ratings. Note that Electric Sheep does have 'generations', but it means something else, see Section 2.

6 Empirical Results

Three main datasets are analyzed below. The voting and posting data are from the web server log files from August 7th to November 4th 2004, 90 days later. Another dataset has daily aggregate usage reports from the server June 18th to October 31st (139 days).

The primary dataset was collected from the server's database starting May 13th until October 13th, 153 days later. May 13th is when version 2.5 became operational. Previous versions of the server did not keep a record of the sheep: when they died they were completely deleted from the server. And though there are large collections of sheep collected by clients from before May 13th, they are not complete and they lack fields for ratings and parentage.

During this time the server was subject to performance optimization but the basic algorithm remained fixed with one exception: until July 13th there was no limit on how many votes a user could make. An increasing incidence of users voting many times in rapid succession instigated a limit of 10 votes per user per day. The numbers below are from after that change.

The data we have collected are a starting point to understanding the system and its behavior. However, they are somewhat confounded:

- IP addresses are equated with users, but because of the prevalence of Network Address Translation (NAT), several or many users may appear as the same IP address. Worse, some computers are assigned an IP address dynamically, so one user may appear under many addresses.
- The client uses the ratings to prioritize downloading. Now that the server is busy enough that some clients cannot download all the sheep, this causes a snowball effect where a high rating itself causes more votes.
- The audience is fickle: sheep with identical genomes regularly receive completely different ratings (See Figure 1). Presumably the audience becomes fatigued by repeated exposure to variations of a successful genome, and stops voting for them. Even once popular sheep reintroduced much later do not necessarily fare well.
- Designers may vote for their own sheep, post many similar sheep, or post alterations of the results of automatic evolution. The administrator occasionally kills sheep, explicitly directs mating, mutation, reincarnation, and votes without limit.

There are many ways of measuring the number of users. Figure 2 shows four of them, and a graph of the rate of new users trying the system. The daily downloaders linear growth rate is 9 users per day, far fewer than the hundreds of new addresses per day. When Electric Sheep is first installed and run it takes quite some time (often about ten minutes but according to some reports hours or even days) to download and display the first sheep. Perhaps many people decide the software is broken and remove it. Or it could result from miscounting dynamic IP addresses.

There were on average 166 downloads of the client installers per day from the homepage during the first 8 days of December. But the installers are distributed on CD-ROM and mirrored on high-volume sites such as nonags.com, debian.org, and freebsd.org from which no statistics are available. These secondary sites also lack preview graphics and explanation so users from them may be likely to remove it.

Figure 3 shows the distribution of how many clients had a given number of days of activity (at least one attempted download). 1839 clients were active half the days or more, and 65 were active every day. Using data collected from November 2 to November 7 2004, there were 626 votes made, 76% with the arrow keys on client, and 24% with the web site.

The average number of valid votes per day is 111. During the 90 day period votes were received from a total of 1682 different client IP addresses. The average number of posts per day is 10.8 from a total of 64 different client IP addresses.

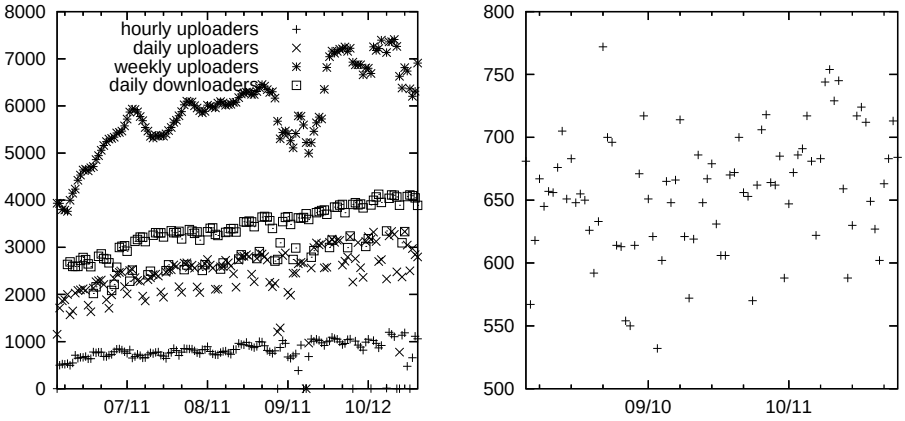

Fig. 2. On the left is a graph of number of users over time. Downloaders refers to clients downloading sheep, uploaders refers to clients uploading rendered frames. The dip from 9/2 to 9/19 coincides with server outages. On the right is a graph of the number of new client addresses over time. The total number of unique downloaders is 62000.

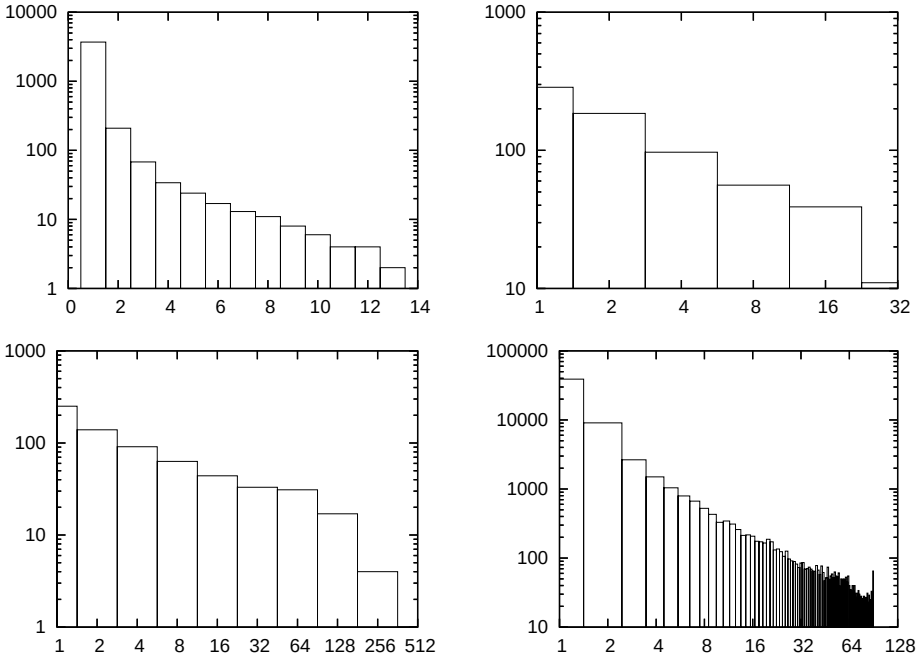

Fig. 3. On the top-left is a histogram of lengths of lineages, and on the top-right is a histogram of the ratings of the sheep. On bottom-left is a histogram of the sum of ratings of all descendents. Children of two parents contribute half of their sum to each parent. On the bottom-right is a histogram of number of days of activity out of 90 total possible by each IP address. 1839 clients were active half the days or more, and 65 were active every day (the upturn on the far right).

6.1 Amplification of Creativity

In a system with human-computer collaboration, the creative amplification is the ratio of total content divided by the human-created content. If we compare the posted genomes with their evolved descendents we can measure how much creative amplification Electric Sheep provides.

In the primary dataset there were 21% hand-designed, posted sheep and 79% evolved sheep. If the sum is weighted by rating, then we get 48% to 52%, for an amplification factor of 2.08 (1+52/48). One could say the genetic algorithm is doubling the output of the human posters.

Of the 79% evolved sheep, 42% of them result from the totally random genetic operator. Their fraction of total ratings is only 3.8%.

There are some caveats to this metric. For example, if the genetic algorithm just copied the posted genomes, it might receive some votes for its 'creativity'. Or if it ignored the posted genomes and evolved on its own, it would receive some votes but they would not represent 'amplification'.

Figure 3 shows the distribution of lengths of lineages of the sheep. The lineage length of a sheep is the maximum number of generations of children that issue from it. Sheep with no children are assigned one, and sheep with children are assigned one plus the maximum of the lineage lengths of those children. Instead of fitness increasing along lineage, we find it dying out: the rating of the average parent is 6.7 but the average maximum rating of direct siblings is only 3.8.

The decay in ratings may result from the audience losing interest in a lineage because it fails to change fast enough, rather than a decay of absolute quality of those sheep. The viewpoint of watching the screensaver and seeing sheep sequentially is different from the viewpoint of browsing the archive and comparing all the sheep. Neither can be called definitive.

Genetic algorithms normally run for many tens to hundreds or thousands of generations. In contrast, the lineages (number of generations) of the sheep are very short: the longest is 13.

7 Related Research

There are now many distributed screen-savers. Most are scientific (SETI@Home, climateprediction.net), cryptographic (distributed.net) or mathematical (zetagrid.net), rather than graphical or artistic. The Golem@Home project [7] has a evolutionary algorithm for evolving locomotion in electro-mechanical assemblages.

The aesthetic selection used by Electric Sheep is inspired by Karl Sims [9]. His supercomputer is replaced by internet-connected commodity PCs. The sheep voting community is much larger but much less focused than his user-base. Sims used Lisp expressions on pixel coordinates, but also included a primitive for IFS.

The International Genetic Art IV project [8] uses a web-based Java client to evolve images following the technique of Sims. Since the images are rendered on the clients, the computation is distributed, but users do not share votes or the gene pool. Its previous incarnation, International Genetic Art II, ran on a single server with a web interface so

the computation was not distributed but the voting and gene pool were shared by all users.

8 Summary and Conclusion

Electric Sheep is a distributed screen-saver for animating and evolving fractal flames, a kind of iterated function system. The animations are shared among the clients and displayed in parallel with rendering. The evolution is guided by the will of the audience. The genetic code is made of geometric transforms, each one containing 6 coefficients for an affine transform of the plane and 18 coefficients for blending the variational functions.

The genetic algorithm employed does not converge on an optimal solution, but follows the attention of the users. And while the genetic algorithm is not competitive with the human designs, it does serve to effectively elaborate and amplify human designs.

Acknowledgements

Many thanks to: The anonymous reviewers, Tamara Munzner, and Matthew Stone for providing feedback on drafts of this paper, Dean Gaudet for help with performance optimization and server hosting, Tristan Horn for hosting the high-bandwidth server, Matt Reda and Nick Long for porting the client to OSX and Windows, respectively. And to all those who sent in patches, reported bugs, voted for their favorite sheep, or just ran the software once.

References

1. David Anderson et al: SETI@home: An Experiment in Public-Resource Computing. Communications of the ACM **45** (2002) 56–61.
2. Michael Barnsley: Fractals Everywhere. Academic Press, San Diego, 1988.
3. Alan Dorin: Aesthetic Fitness and Artificial Evolution for the Selection of Imagery from The Mythical Infinite Library. Advances in Artificial Life, Proc. 6th European Conference on Artificial Life, (2001) 659–668.
4. Scott Draves: The Fractal Flame Algorithm. Forthcoming, available from `http://flam3.com/flame.pdf`.
5. Scott Draves: The Interpretation of Dreams. YLEM Journal **23;6** (2003) 10–14.
6. Scott Draves: The Electric Sheep web site at `http://electricsheep.org`.
7. Hod Lipson and Jordan B. Pollack: Automatic Design and Manufacture of Robotic Lifeforms, Nature **406** (6799), 2000 August 31, 945–947.
8. John Mount, Scott Reilly, Michael Witbrock: International Genetic Art IV. Published on the web at `http://mzlabs.com/gart`.
9. Karl Sims: Artificial Evolution for Computer Graphics. Proceedings of SIGGRAPH 1991, ACM Press, New York.
10. Mark Townsend: Apophysis v2 software distributed from `http://apophysis.org`. 2004.