ALife Meets Web: Lessons Learned

Luigi Pagliarini Ariel Dolan Filippo Menczer Henrik Hautop Lund

The Danish National Centre for IT-Research
University of Aarhus, Ny Munkegade bldg. 540, 8000 Aarhus C., Denmark

Institute of Psychology
National Research Council, 15 Viale Marx, 00137 Rome, Italy

Computer Science & Engr. Dept.
Cognitive Science Dept., University of California, San Diego, La Jolla, CA 92093–0114
USA

The Danish National Centre for IT-Research
University of Aarhus, Ny Munkegade bldg. 540, 8000 Aarhus C., Denmark

luigi@caio.irmkant.rm.cnr.it aridolan@netvision.net.il
fil@cs.ucsd.edu hhl@daimi.aau.dk

Abstract Artificial life might come to play important roles for the World Wide Web, both as a source of new algorithmic paradigms and as a source of inspiration for its future development. New Web searching and managing techniques, based on artificial life principles, have been elicited by the striking similarities between the Web and natural environments. New Web interface designs, based on artificial life techniques, have resulted in increased aesthetic appeal, smart animations, and clever interactive rules. In this paper we exemplify these observations by surveying a number of meeting points between artificial life and the Web. We touch on a few implementation issues and attempt to draw some lessons to be learned from these early experiences.

1 Introduction

In recent years, the Internet and its hypertextual and graphical World Wide Web subset have developed very rapidly. Even though new techniques — based on human-computer interaction, information retrieval, and network routing methodologies — have been applied to a range of Web problems, the emergence of suitable Web techniques has not been as rapid as the growth in size and complexity of the Web.

Can the field of artificial life (ALife) provide the growing Web community with useful inspiration? What new paradigms, methodologies, and algorithms does ALife offer? Is the Web a suitable artificial environment to foster new, useful artificial life forms? Should we bother to embark on the enterprise of populating the Web with intelligent, interactive, autonomous agents with life-like behaviors? These are important questions that deserve the attention of the ALife community. To stimulate this discussion, we start in this paper with a survey of some projects in which ALife has already, to variable degrees, met the Web.

We will focus on some Web-related issues where ALife, in our opinion, may come to play a positive role. These issues include aspects of distributed information search and management, such as scaling and user-adaptability (described in the Applications to Web Search and Management section); and aspects of Web interface design,such as aesthetic appeal, animation, and interactivity (surveyed in the Applications to Web Interface Design section). Throughout the paper, we discuss some lessons learned through these early experiences.

2 Applications to Web Search and Management

One of the central goals of ALife, in our opinion, is to apply algorithms inspired by natural systems to practical applications. The applications that best lend themselves to ALife approaches are those whose operating environments share important characteristics with natural environments. With the explosion of the Internet, it would have been difficult not to notice the striking similarities between this artificial environment and those in which real creatures evolve, adapt, strive, compete, collaborate, learn, grow, reproduce, and die. Like natural environments, the Web is very large; it is dynamic, with documents being added, removed, and moved all the time; it is heterogeneous — even considering text alone — with context-dependent languages, formats, and styles; it is noisy, with lots of irrelevant, outdated, or incorrect information; and it is distributed, so that actions have costs (for example the network latency associated with accessing a certain page).

Innumerable paradigms have been brought fourth to tame the Web into fitting some pattern familiar to users. Even the emerging field of autonomous software agents owes much of its success to the ready availability of virtually infinite information environments, and the difficulties encountered by users attempting to cope with such environments. The InfoSpiders project was born as an exploration of the possible matches between some of the features that make the Web a complex environment, and some of the mechanisms that allow ecologies of organisms to adapt to different — but at least equally complex — natural environments.

In particular, the large, dynamic, and distributed flavor of the Web leads to scaling problems when traditional information retrieval methods are employed to build search engines. Information often becomes stale before a search engine's database can be updated. The heterogeneous organization of information makes any single, global exploration strategy less than optimal — no matter how clever. What can nature teach us about ways to complement these methods with less traditional approaches? In nature, no single species is "optimal" with respect to the whole world; each is adapted to some niche in which members of that species have evolved. Populations of situated, mobile agents afford decisions based on local context, continuous adaptation, and robustness — in natural and artificial environments alike. Different subpopulations can specialize in dealing with the peculiar characteristics of the local environments they experience.

2.1 InfoSpiders

The aim of the InfoSpiders system is to apply and test several machine learning techniques, inspired by natural adaptation, for problems posed by searching and managing information on the Web. Different methods have been explored, extended, and integrated for this task. In the project, we experimented with versions of the genetic algorithm employing localized selection schemes, to overcome the problem of premature convergence and allow for distributed implementations; with different agent representations, to enable agents to internalize local textual features into their evolving behaviors; with reinforcement learning, to adapt individual strategies over short-term time and space scales, based on local context; and with relevance feedback, to permit the user to bias the search process, seamlessly and on-line, based on previous and current performance.

A detailed description of the implementation of the InfoSpiders system is out of the scope of this paper. Interested readers can find such details, as well as reports on preliminary experiments, elsewhere [14–16] or on-line [1]. Here we limit ourselves to outline the general ideas behind the model. The first step is to identify the crucial resource, the "food" of the artificial environment. For the task at hand, it is easy to equate resources with relevant information. Since relevance is subjective (actual

relevance depends on the user, and must be estimated by agents), information must be transformed into a higher-entropy quantity; this single currency by which agents in a population survive, reproduce, and die is called "energy." Energy must be positively correlated with performance as defined by user and environment.

Agents asynchronously go through a simple cycle in which they receive input from the environment as well as internal state, perform some computation, and execute actions. Actions have an energy cost but may result in energy intake. Energy is used up and accumulated internally throughout an agent's life; its internal level automatically determines reproduction and death, events in which energy is conserved. Agents that perform the task better than average reproduce more and colonize the population. Indirect interaction among agents occurs without the need of expensive communication, via competition for the shared, finite environmental resources. Mutations and crossover afford the changes necessary for the evolution of dynamically adapted agents. This paradigm enforces density-dependent selection: the expected population size is determined by the carrying capacity of the environment. Associating high energy costs with expensive actions intrinsically enforces a balanced network load by limiting inefficient uses of bandwidth.

Collective Behavior Adaptation means for agents to concentrate in high energy areas of the Web, where many documents are relevant. Each agent's survival will be ensured by exchanging an adequate flow of information for energy. The situation is illustrated by the snapshots in Figure 1, illustrating a typical collective behavior of InfoSpiders in response to a query, and limited to a well defined chunk of the Web. After a while, blue agents have found what the user wanted; they prosper and multiply in this relevant niche, while other agents continue to explore the world in search of alternatives.

Search Efficiency One of the central results of the InfoSpiders project is to have shown that it is useful for information search algorithms to view the Web as a "natural" environment and characterize it in such terms. What are the environmental assets from a learning agent's perspective? Statistical features such as word frequencies are of course crucial dimensions. It has been argued that the "link topology" structure imposed by information providers upon the organization of documents is another important resource. Even in unstructured information environments, authors tend to cluster documents about related topics by letting them point to each other. This creates a landscape that agents can explore making use of correlation of relevance across links.

Preliminary results, based on theoretical analysis, simulations, and actual experiments on Web-based corpora, are very encouraging [14]. It has been shown that link topology can indeed be detected and exploited by distributed agents, outperforming exhaustive search by an order of magnitude. Moreover, the synergy between evolution, learning, and relevance feedback induces an additional, four-fold boost in performance [15].

3 Applications to Web Interface Design

Web Interface Design (WID) is drastically changing in its aspect and functionality. Yet it is unclear in what direction it is moving. There are at least three quality factors in WID: (a) clever interactive rules; (b) smart animation; and (c) aesthetic appeal.[1] Until recently, WID interactivity and smart animation, handmade or aided

[1] Here we will not take into account other important, but secondary factors, such as user-oriented interface personalization, interface legibility, multilevel functionality, etc.

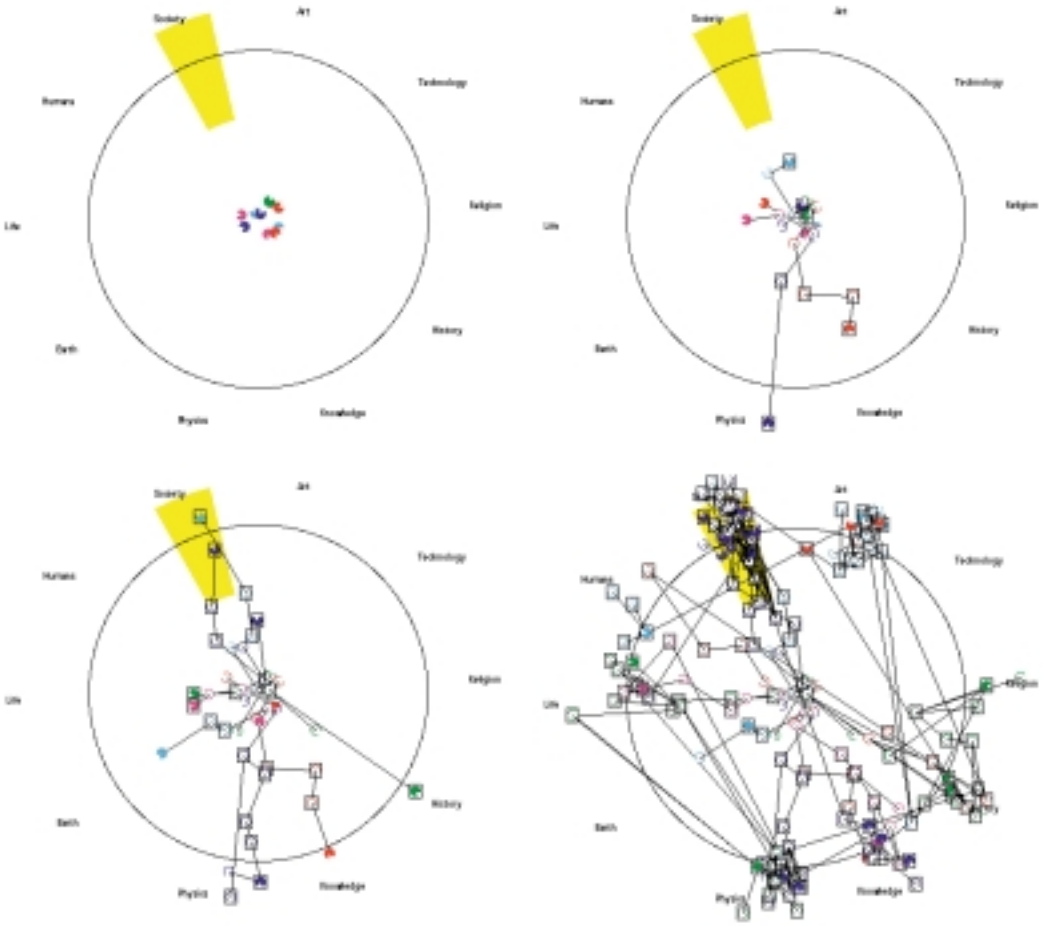

Figure1. Snapshots of InfoSpiders searching the EB Web space for documents relevant to the query "Laws governing relations among sovereign states." The document's ontology is represented with more specific topics farther from the center of the circle, and actual articles outside of the circle. The relevant documents are represented by the yellow area, unbeknownst to the agents. A document is marked with a rectangle if it is estimated as relevant by the first agent visiting it. The color of an agent represents its lineage, so that all agents sharing a color descend from the same ancestor. (EB data is ©1997 Encyclopaedia Britannica, Inc. [2]; the visual representation is the author's.)

by artificial intelligence techniques, relied only on animators' or programmers' abilities; and the aesthetic appeal of a WID relied uniquely on graphic designers abilities. However, since users want to get out of this unexciting landscape, there is a demand for more appealing or artistic aesthetics, much smarter — if not life-like — animations, and more clever — if not human-like — interactions. ALife techniques appear among the earliest and most plausible candidates to address some of these needs. Indeed, in the remainder of this section we show some examples in which ALife helps us to fulfill these needs by providing for new ways to interact (a), by making animations richer and more "natural" (b), and by extending the human capabilities to build aesthetic artifacts (c).

3.1 Web Interactive Cellular Automata

WICA stands for "Web Interactive Cellular Automata." It is a Game of Life [13] (or any other) cellular automaton [20] that is controlled by free flowing text on an HTML page. The free flowing text is some active text that, when clicked, dynamically changes the state of the cellular automata. It was initially created to support Web-design and provide pleasing, changing and unpredictable animations for Web page logos. It was then extended to support dynamic computer art and other kinds of applications. Some examples of WICA can be seen on the WICA homepage [3]. Animation seems to be almost a requirement in the current fashion of Web pages; something must always move. If the page is "static," it becomes boring. Therefore we see a lot of animations on the net. Most of them are GIF animations, and the others are usually simple Java animations. GIF animations are not very pleasing: monotonous, nagging, tiresome — sometime irritating! Java animations are just a little better. The endless repetition of exactly the same patterns is hardly inviting. WICA animations are different. Although they can be used for the same purpose as GIF animations (e.g., animated logos), they are far from monotonous. Their ALife flavor makes them always unpredictable, conveying a feeling of life.

A WICA-based Homepage As an example, consider a WICA dynamic website logo shown in Figure 2. This usage of WICA was implemented in the homepage of Prof. Domenico Parisi of CNR [4]. Two CA's of different colors are superimposed on the page logo, and a new pattern is released into one of them each time another page is selected by clicking a hypertext link. Since the CA applet resides on a separate HTML frame, it is always visible, even when other frames are changed. Although the CA is played over the logo, the logo remains clear and is not hidden by it. The resulting animation is much more interesting, unpredictable, and "alive" than any animated GIF or pre-designed animation can be. We attribute this effect to the use of ALife techniques.

Figure2. The WICA dynamic website logo.

The specific WICA implementation was made so that the cellular automaton applet uses Conway's classic Game of Life rules. However, the CA world (imple-

mented here as an arbitrary background picture) is shared by two parallel, independent, overlapping CA's. It is an interactive applet, where control is achieved through clicking on textual links rather than by the orthodox use of buttons and menus. The actual dimensions of the CA grid are arbitrary, but since WICA usually goes with a picture background, the size is dictated by the picture.

The CA algorithm implements three methods for making its calculations efficiently:[2]

1. While the current generation is displayed on the screen, the next generation is prepared on an off-screen buffer, and then replaces the current screen in a single step.
2. Only cells that changed their status are re-painted.
3. When calculating the number of live neighbors of each cell, knowledge from previous cell calculations is used: e.g., since the left column of the current cell is the middle column of the previous cell, the status of the left neighbors is known and need not be recalculated.

The dynamic character of WICA is based on two parallel, independent CAs playing simultaneously in the same world. The CA world is controlled by Javascript statements that activate a Java routine. When this routine is activated, a new pattern of cells is inserted into one of the CAs. The CA that receives the patterns is selected at random. Also selected at random are three parameters of the new pattern: the pattern itself, its color, and its location within the CA frame. The pattern itself is selected from a list of three interesting and well-known unstable patterns: the *R-Pentomino*, the *H-Heptomino* and the *Pi-Heptomino*. The color is selected at random from a list defined in the HTML file.

This implementation results in having exactly two colors (and two populations) at any given moment, but the colors may change giving the illusion of seeing new populations. It also gives a somewhat unexpected behavior, as one of the CA populations suddenly changes color and takes a new pattern.

Text Oriented Interactivity WICA animations can be integrated with a Text Oriented Interface (TOI) [17] so that they always renew themselves. JcaToi [6], shown in Figure 3, is such an extension of WICA and another example of value-added communication. JcaToi provides the user with an interactive Game of Life implementation that is controlled by free-flowing text. WICA was designed to react in parallel to regular HTML hypertext events, being a sort of by-product to regular Web page requests, and always producing the same (albeit unpredictable) behavior, namely adding a new CA pattern to the CA world. JcaToi, on the other hand, is designed to perform various predefined actions. In JcaToi, various hyperlinks mean different things and perform different dedicated actions. This is done by using a more diverse Java-Javascript interface, where multiple Java routines can be activated by the Javascript commands.

JcaToi makes use of three tools: HTML frames, Java, and Javascript. HTML frames are used for creating pages where one part of the page (the program or the picture) is static and constantly displayed in a predefined location, while the rest of the page can be scrolled or change content. This makes the size of the "user interface" practically unlimited. Java is used for the program itself, Javascript for manipulating the hypertext links, and the Java-Javascript interface is used to allow the text to control the program.

The text describes the CA applet that is displayed on the HTML page, and allows the browsing user to play and learn with it, by clicking on the appropriate words. The JcaToi application thus demonstrates that ALife techniques can add

[2] The basic algorithm is due to Andreas Ehrencrona [5].

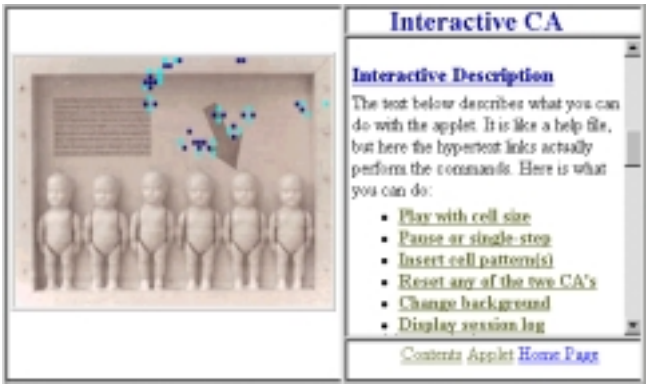

Figure3. The JcaToi website.

educational value to the Web, by making reading an interactive and "alive" process. For example, you can imagine an Artificial Life Interactive Handbook where, by using this technique, you can actually transform your reading into an active process. An example would be: if you do not know what a 'hidden unit' in a neural network is, you just click on the 'hidden unit' text and you see that the corresponding part of the neural network picture is flashing. That is what we mean by reading as an interactive process. Of course, this is the most static part of it. You can also think of 'introducing food' free flowing text into a running simulation, etc.

3.2 Flocking Web Creatures

Floys [7] are another example in which ALife issues cross over with Web interface issues. Floys (shown in Figure 4) belong to the flocking ALife creatures variety [19], sharing with them the social tendency to stick together, and the life-like emergent behavior which is based on a few simple, local rules. They differ from most other ALife flocking (boids-type) implementations by being territorial animats that defend their territory against intruders. In the WID system they are implemented as Java applets. The more advanced applets allow individual Floys to change traits and personality (iFloys and eFloys), and population of Floys to breed and evolve (eFloys). One can observe various applications of this technique. The boids-based algorithm has been used in the entertainment industry (e.g., in the "Jurassic Park" movie for animating a storm of prehistoric birds). For an example of its application to WID, see [8]. Animation realized with this kind of technique is more life-like, visually pleasing, and smarter than traditional Web animation techniques; it is also easy to realize — once one has learned to program the boids or simply use their applet.

Before further discussing Floys' implications for WID, let us go over a few details about their implementation. In general, the Floys behavior is governed by two rules:

- a rule specifying how to relate to one's own kind;
- a rule specifying how to relate to strangers.

Unlike most other flocking algorithms, a Floy does not relate to all members of its community, but only to its two closest neighbors. This idea was borrowed from Alex Vulliamy's Flies code [9], of which Floys is a descendant. Each Floy is "emotionally" attached to two other members (its neighbors) of the flock, and tries to stay close to them.[3] The Floy relates to the neighboring Floys by identifying whether they are

[3] An approximate neighbor detection algorithm is used for efficiency, which is of course a crucial aspect in Web design.

Figure4. The Floys website.

of its own kind or strangers. It will chase a stranger away while in its own territory, or flee if chased by a stranger while outside of its own territory.

In the basic Floys applet, behavior parameters are common to the whole population. In iFloys, instead, each Floy can be assigned different traits. Among the parameters that can be assigned, we find acceleration, adhesion, collision distance and maximum speed. For example, an iFloy with higher acceleration and speed traits will tend to be faster, more abrupt, and will appear more nervous, spontaneous and individualistic; a Floy with higher adhesion value will tend to cling more to the community and will look like a conformist. When one iFloy is assigned high speed or acceleration traits, the whole group becomes a little confused and disorganized. An iFloy with slow and lazy traits will be left behind. Color codes for stranger vs. local Floys. Local iFloys have the aggressive behavior, while non-local iFloys have the flight behavior. When iFloys are transformed to the stranger color, they are attacked, even if they are the majority. However, when many strangers are present, the local iFloys appear confused and do not fight them effectively.

The most advanced of the three versions is the eFloys applet, which supports evolution. This setup allows manipulation of more than 30 different parameters. In addition, several pre-defined behavior styles can be assigned to the current population. The applet also supports the option of attaching a number graphically to each Floy in order to track individual Floys easily. Although applets do not support file access for security reasons, current status and results can always be displayed in a dedicated information screen. Where non-evolving Floys live eternally (or until their energy drops to zero), eFloys live one generation, and a generation ends when the stranger is finally killed. eFloys evolve sexually, where each eFloy is the descendent of two parents. The two parents are selected according to two fitness attributes, energy and safety. An eFloy can gain or lose these during its lifetime, and the more it has, the fitter it is. Energy is lost proportionally to speed, and safety is gained from proximity to neighbors. With this set-up, one can easily make experiments with populations of eFloys having predefined behaviors, or perform simulations of the evolutionary process tuning the relative weights of safety and energy in the fitness function.

3.3 Interactive Computer Art

Floys have exemplified how ALife models can be implemented to run across the Web. But do such systems afford anything useful from a WID point of view? A positive answer may be obtained by considering interactive computer art, where

the user interacts with graphical objects rather than code. The Artificial Painter program [18, 21, 10] is one of several applications of ALife to computer graphic design. It is based on artificial neural networks and evolutionary algorithms. At the beginning of each evolutionary session it generates 16 random images which are a sort of "seed-images" for the painter to start its work. The user can select four of them as parents, which then produce sixteen new individuals constituting a new generation. The user can zoom out each of the images, carefully look at them, and observe changes, trends, and interactions in composition, color, texture, form, and perspective. The only parameter the user can control at each generation is mutation rate (which however can be assigned differently for each generation).

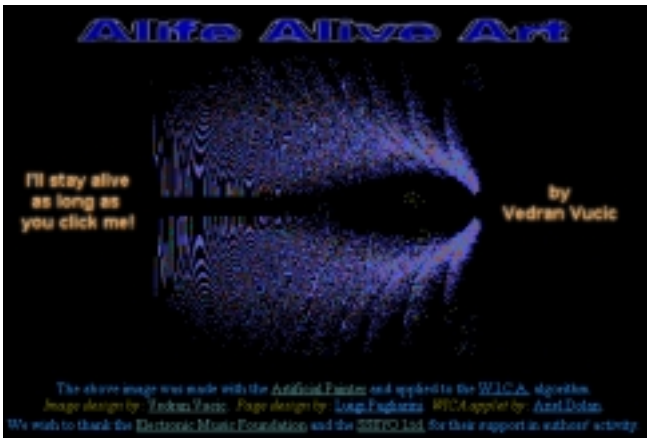

Figure5. The Alife Alive Art website.

Applications of this program can be seen in Vedran Vucic's homepage [11] and the Alife Alive Art site [12] shown in Figure 5. These sites demonstrate a couple of points. First, ALife techniques can be applied to graphic design and therefore to WID, as in the case of Vedran Vucic's site. Second, the user can interact with and influence a picture by controlling its ALife-based evolution. The Alife Alive Art site, derived from the Artificial Painter and Floys, allows the user to do so by interacting with the text that is attached to, and part of, the picture.

4 Conclusion

The projects surveyed in this paper offer a chance to highlight some of the meeting points between ALife and the Web. As we have been showing all along the paper, one can see the web as an enviroment where artificial creatures made up with ALife techniques can move. We also showed that one can imagine these creatures to be very different agents with very different function and properties.

Firstly, as shown by the InfoSpiders, the Web is a fertile ground to apply and explore ALife-inspired algorithms. As electronic information environments grow and become more complex, while the class of Web users extends from computer experts to elementary school students, the need for "soft" computing technologies can only become more stringent in our view. The more the Web looks like a natural, live environment, the more we must look at real, living systems for humbling inspiration.

This also seems to be the case for WID: we have found ALife techniques (inspired by living systems) to provide a number of suitable tools that might facilitate the design of pleasing Web interfaces. We have shown examples of uses of such

techniques to achieve appealing aesthetic (as in the Alife Alive Art site and Vedran Vucic's homepage), smart animations (as in the WICA and Floys sites), and clever interactive rules (as in the JcaToi demonstration).

Finally, one can also view the Web as a shared testing ground for ALife models, as exemplified in the Floys system. The premise of the scientific method is the reproducibility of experiments. ALife techniques and simulators are often difficult to compare due to the fact that each defines a different class of environments. The Web provides ALife practitioners with a global laboratory in which they can examine their theories and do good science.

So, facing such problems as the synthesis of virtual worlds on the web it seems necessary to take into account those kind of approaches we have shown. The most important lesson learned is: think of the web as a natural enviroment where to place organisms and by doing that one quickly realizes that ALife techniques, at the moment, are the most fit ones for the realization of bio-like worlds.

References

1. http://www.cs.ucsd.edu/~fil.
2. http://www.eb.com.
3. http://gracco.irmkant.rm.cnr.it/luigi/wica.
4. http://gracco.irmkant.rm.cnr.it/domenico.
5. http://www.student.nada.kth.se/~d95-aeh/lifeeng.html.
6. http://www.aridolan.com/JcaToi.html.
7. http://www.aridolan.com/JavaFloys.html.
8. http://www.arch.su.edu.au/ thorsten/alife/biomorph.html.
9. http://www.vulliamy.demon.co.uk/alex.html.
10. http://www.daimi.aau.dk/~hhl/ap.html.
11. http://www.rex.opennet.org.
12. http://gracco.irmkant.rm.cnr.it/luigi/vedran.
13. M. Gardner. Mathematical games. *Scientific American*, 233:120–23, 1970.
14. F. Menczer. ARACHNID: Adaptive Retrieval Agents Choosing Heuristic Neighborhoods for Information Discovery. In D. Fisher, editor, *Proceedings 14th International Conference on Machine Learning (ICML97)*. Morgan Kaufmann, 1997.
15. F. Menczer and R. K. Belew. Adaptive Information Agents in Distributed Textual Environments. In *Proceedings 2nd International Conference on Autonomous Agents (Agents 98)*, 1998. To appear.
16. F. Menczer, R. K. Belew, and W. Willuhn. Artificial Life Applied to Adaptive Information Agents. In *Working Notes of the AAAI Spring Symposium on Information Gathering from Distributed, Heterogeneous Environments*, 1995.
17. L. Pagliarini and A. Dolan. Text Oriented Interactivity. Technical report, C.N.R., Rome, 1998. Submitted to Applied Artificial Intelligence (AAI) Journal Special Issue Animated Interface Agents: Making them Intelligent.
18. L. Pagliarini, H. H. Lund, O. Miglino, and D. Parisi. Artificial Life: A New Way to Build Educational and Therapeutic Games. In C. Langton and K. Shimohara, editors, *Proceedings of Artificial Life V*, pages 152–156, Cambridge, MA, 1997. MIT Press.
19. C. W. Reynolds. Flocks, Herds, and Schools: A Distributed Behavioral Model. *Computer Graphics*, 21(4):25–34, 1987.
20. J. von Neumann. *The Theory of Self-Reproducing Automata*. University of Illinois Press, Urbane, 1966. A. W. Burks, ed.
21. V. Vucic and H. H. Lund. Self-Evolving Arts — Organisms versus Fetishes. *Muhely - Hungarian Journal of Modern Art*, 104:69–79, 1997.