Alien Letter Forms

Jason Lewis
Concordia University
1455 de Maisonneuve Blvd. West CB206-14
Montreal QC (CA) H3G 1M8
jason.lewis@concordia.ca

David Bouchard
Independent Artist
2715 rue Dézéry
Montreal QC (CA) H1W 2S6
david@deadpixel.ca

ABSTRACT

In this paper we describe Alien Letter Forms, a software environment for letters to live in and evolve into larger texts. Alien Letter Forms builds on technical and conceptual approaches borrowed from research around artificial life systems to create a digital ecology in which text agents live, reproduce, evolve, mutate and die in response to their virtual environment. Individual letters are treated as autonomous agents, where contact between individuals produce offspring composed of new letter combinations up to and including whole words. Evolution is driven further by how well new combinations of letters fit into a set of pre-existing reference texts. The Alien Letter Forms environment is designed to produce a continuously evolving collection of letters that has local stabilities that produce readable text.

Keywords

Digital text, dynamic typography, evolutionary text, emergent meaning, visual language, recombinant text.

1. INTRODUCTION

The current work is the latest installment in a series of experiments focused on exploring the expressive possibilities of computationally active digital texts. By computationally active, we mean texts which make use of computation to operate on the interactive, dynamic and linguistic dimensions of a text. The goal of these experiments is to discover new ways–both creative and technical–of working with one of our oldest communication technologies.

2. MOTIVATION

This project aims to explore a potential biological metaphor for language. Through evolution, simple life forms gave way to more complex ones; similarly, our characters slowly evolve into more complex strings towards formation of words and sentences. An important question we are hoping to answer with this experiment is whether the biological metaphor is relevant and to what extend it can be applied to language. We also hope to provide a means of exploring the semantic space of words as well as the visual space of typography.

We are not interested in, nor make any claims of, developing a scientific tool that simulates the natural evolution of human language. Rather we approach text from an evolutionary perspective as a source of inspiration and as a means of exploring the semantic space of a particular language as well as the visual space of typography.

3. RELATED WORK

The present work draws technical and creative inspiration from concepts of artificial life and emergent behaviour, agent-based behaviour systems and computational text.

Sim's Galapagos [15] is perhaps one of the most interesting attempts so far at applying genetic algorithms to interactive computer art. Projects like the Virtual Fish Tank [9] have been exploring evolution and emergent behaviour between virtual agents in a creative and visually compelling way.

Lewis and Weyers' work on ActiveText [4] specifies a system for coordinating complex dynamic and interactive behaviours between textual elements. The Visual Language Workshop at the MIT Media Lab has been the source of a number of works that use agent-based behaviour systems to support the visualization of rich behaviour-based texts. Ishizaki's dissertation [4] on "typographical performance" anticipates the use of coordinated behaviors to create a rich visual interplay of text in an email system. Wong's thesis [17] on "temporal typography" uses Soo's object-based, behavior-driven architecture [10] to combine dynamics and semantics.

There are a number of creative works which combine computation and text manipulation to create variations of the Surrealist's Exquisite Corpse [11]. Recombinant works such as Lewis' Breeder [5] and Seaman's World Generator [13] recombine texts from a corpus according to pre-marked grammatical attributes. Generative works that compose new texts according to specifications of a rule-base system are exemplified by include Bulhak's Dada Engine [1]. Evolutionary works, such as Schmitzs' Genotyp [12], make explicit use of genetic algorithms to breed new typefaces.

4. ARCHITECTURE

The architecture of Alien Letter Forms is inspired by agent-based simulations. In such simulations, agents are often modeled after real-world organisms. They are autonomous and independent entities which usually have a limited awareness of their environment.

This translates in our system by defining an agent as the most basic unit of written language--a character. Additionally, our agents carry properties of their own such as a velocity and a typeface, which affect their behaviour on screen and their visual appearance. Their position and their proximity to other agents determine what they know of their

environment and therefore how they can interact with each other and the space around them. (Figure 1).

Agents take advantage of the NextText library (described in section 8) to model different behaviour that can act on properties like velocity and direction to generate movement patterns (linear motion, erratic motion, etc.). Behaviours can also modify the physical appearance of the agents themselves. These behaviours are programmed as independent units and can be inherited from an agent to the next.

The simulation itself is performed on a time-step basis. At the tick of an artificial clock, each agent is updated in a parallel fashion. The method by which each agent is updated depends on its properties and behaviours but also on its interaction with other agents, which consists primarily of the reproductive mechanism described in section 6.
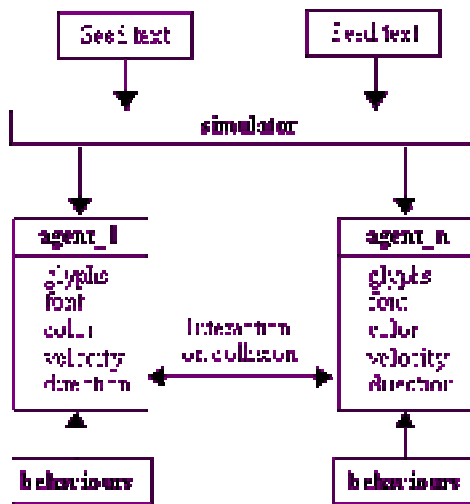

Figure 1. Alien Letter Forms' agent architecture

5. ENVIRONMENT

Agents exist as though they were floating in liquid or in space. They move freely around the environment, but tend to be naturally attracted towards other healthy agents. This movement makes the randomness required for genetic evolution visible to the observer

The environment is seeded with pre-written texts. Each seed text generates a zone of influence which is associated with a region of the screen (Figure 2). Theses zones are dynamic, meaning that seed texts compete for space amongst themselves. Their influence area grows larger as more agents spend time within their boundaries. A future iteration will allow users to submit seed texts to the system in real-time over a network connection.

The seed texts could be texts exhibiting, for instance, a contrast in content, style or even language. The content of the seed texts influences the reproductive process and thus shapes the outcome of the text generation.

Figure 2. Sketch of the environment filled with random agents in their early stages.

6. EVOLUTIONARY MECHANISMS

In order for evolution to occur, we must set up mechanisms that aim to introduce improvements in the population by each successive generation. This is translated in the system by a reproductive function, which has a small chance of being triggered whenever two agents are within physical proximity of one another. To be coherent with the evolutionary metaphor, our reproductive function implements some common operators of genetic algorithms [8].

6.1 Fitness

Usually, fitness is a function that attempts to rate agents based on how "good" they are at performing a given task. Note that this function should be computationally effective because it will be frequently called. It should also be stochastic in order to allow the system to take unexpected directions.

In Alien Letter Forms, the task is for agents to become readable texts. Some strings will be better suited to this than others; in the English language, for instance, there is an identifiable difference in fitness between a string such as "house" and "zkcoe". In order to represent and measure fitness, agents are rated according to the relative probability of occurrence of their character string within the seed texts. The result of the function is a real value from 0 to 1 which indicates a string's fitness. Additionally, fitness is directly reflected by the agent's properties such as speed.

Fitness is computed using a data structure inspired by Decision Learning Trees [7] which conveniently allows for efficient queries at runtime as well as dynamic updates of the reference texts. Every possible combination of up to four characters has its own path in the tree. The four character limit is imposed by memory restriction since the tree grows exponentially at every branch level. When building the tree, a counter is incremented every time a node is reached. Additionally, the maximum counter is stored at each level of the tree, indicating the number of occurrences of the most frequent letter combination out of all possible strings of a certain length (where the length is the depth in the tree). We use this maximum value as a reference to which all other values are compared in order to generate relative frequencies.

Using such a tree, determining the fitness is as straightforward as following the path given by the characters of the string we are trying to evaluate and then dividing the score contained in the terminal node by the maximum score for its depth. Figure 3 illustrates the tree structure for a partial alphabet and how each level of the tree is associated to a string of corresponding length. The actual tree used in this project includes uppercase and lowercase combinations, spaces and a set of common punctuation characters.

For strings larger than four characters we evaluate their overall fitness by randomly picking a sample substring every time, thus ensuring an element of uncertainty. The use of samples in this case accurately estimates the fitness of the string as a whole over a time span because the sampling area is always changing.
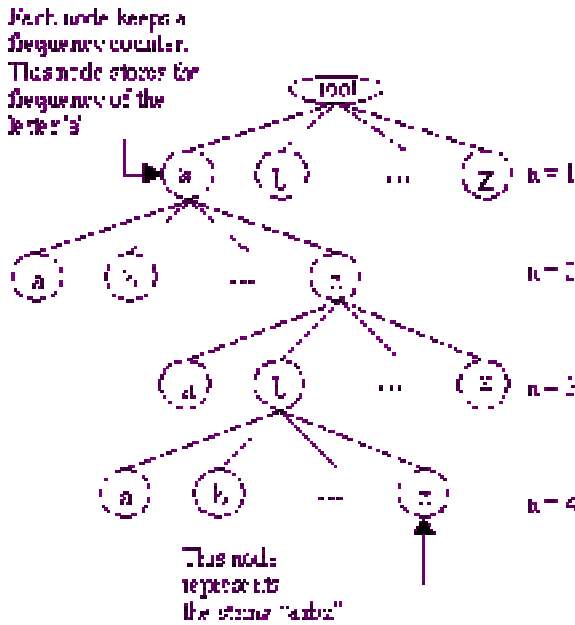

Figure 3. Partial decision tree of maximum depth (n) 4 for an alphabet [a…z].

Fitness acts as a determining factor in the application of the other operators. Whenever a fitness test has to be performed, a random number from 0 to 1 is generated. If it is smaller than the agent's fitness, then the test is a success, otherwise it is a failure. Accordingly, fit agents will have a higher success rate.

6.2 Crossover
The genetic algorithm also implies a crossover operator which simulates reproduction between individual agents. (Figure 4) Traditionally, this operation is performed by taking two agents and randomly recombining them into two children which in turn take over their parent's place in the next generation while preserving some of their properties.

In this case, however, crossover is approached in a different way because we want our organisms to grow larger over time. For this reason we let agents aggregate together during the crossover operation. Therefore, instead of producing two new offspring, agent couples merge together into one larger

agent during crossover. The order in which the two agents are aggregated is determined at random.
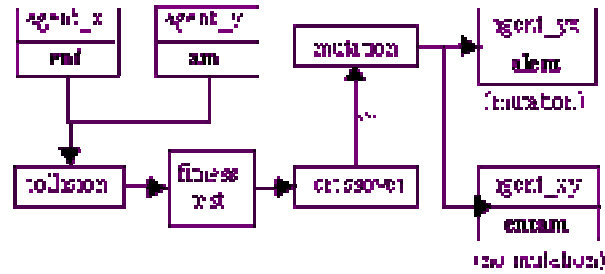

Figure 4. A scenario describing two possible outcomes of a crossover (one with mutation and one without).

6.3 Mutation
Every time two strings aggregate, we introduce a small chance that they might undergo a mutation in the process. All agents are equally likely to be affected by a mutation, regardless of their fitness. A mutation is represented by replacing one of the characters by a new random character. Mutation occurs in order to help prevent the system from stagnation.

6.4 Death
The death operator acts like natural selection. Unlike crossover, which requires two agents to be near each other, death can strike anyone at anytime. Fit agents are less likely to be affected by death than other agents.

To be sure, when struck by death, agents do not disappear entirely but instead break into two smaller strings. This is a consequence of the crossover operator which already reduces the number of agents in the population. Therefore, we chose to avoid destroying large agglomerations of agents at once. Only single-glyph agents can literally die—be permanently removed from the system—as they cannot be further reduced.

6.5 Clone
The last operator, clone, is used to allow fit agents to reproduce themselves by duplication (Figure 5). Like death, clone can happen at any time and is random, although fit agents are more likely to be selected. This operator stimulates the production of good agents in order to improve the overall quality of the population. Also, like during crossover, there is always a slight chance that a mutation might occur in the process.
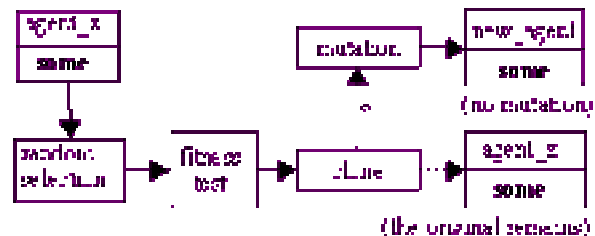

Figure 5. A scenario for a clone operation where no mutation occurs

7. INTERACTION

The Alien Letter Form environment, once it has begun, is largely autonomous and acts without outside intervention. Users can influence the system directly only through the process of agent creation. New agents are introduced to the system from text that users input. In the current system, this is accomplished by typing into a window, from which Alien Letter From pulls characters to turn into agents. The next system will allow users to do this in a more natural fashion by using a microphone and speech recognition engine.

Sentences submitted by users are broken down to letters and born again as fresh agents in the system. In the process, there is an opportunity to parse the input text (before it is taken apart) and extract semantic properties from it, assuming that the user types in or says actual sentences. The occurrence of certain words, for instance, will influence the agent's initial properties such as type of movement behaviour, font or color. Agents can also be given additional properties at birth based on the meaning of the input. An example of this is predator agents.

Predators are integral to most ecosystems yet they have been overlooked in this discussion so far. The threat they pose to other species serves the dual purpose of regulating population growth and stimulating evolutionary change. In order to transpose this phenomenon to language, predator agents are generated whenever a slang word is parsed from the input text. Slang disturbs established linguistic structures. Similarly, our predator agents are given a property that causes them to trigger the death (See 6.4) of other agents upon collision. Note that predators are not affected by death the way regular agents are. Instead, predators have a lifespan dependent on whether or not they can find prey over a given period of time. This mirrors creation of a natural life cycle that will not be dominated by a large population of predators.

8. NEXTTEXT

Alien Letter Forms relies on NextText [9], a Java library which allows manipulation of dynamic and interactive texts. The NextText architecture encourages users to regard text in both its linguistic and visual dimensions simultaneously. Instead of having to shift modes or worse yet, applications, in order to move from one dimension to another and back again, NextText provides an environment in which creators can write as well as paint with words.

NextText employs an object-oriented approach that parses input text into a loosely language-based hierarchy of glyphs, words and phrases. This hierarchy supports the easy applications of behaviors at different levels of the text, allowing the user to specify behavior for an entire passage of text in the same manner as for individual letters. Behaviors can operate on the visual, dynamic, interactive or semantic characteristics of the text. Text components are provided with simple agent-like functionality to allow them to interact with one another as well as with the user.

NextText is the successor to ActiveText, a C++ library with a similar architecture.

9. CONCLUSION AND FUTURE WORK

We have described a software framework created to experiment with genetically evolved texts. We find that the resulting environment generates interesting sub-texts and provides a rich means for re-writing texts. However, the current state of the work represents just the first few steps into this area. Further research and development remains.

One area that will be explored in a future iteration of the project regards the physical appearance of agents. Further work is planned in order to allow evolution of font faces during reproduction. The implementation has been delayed as we research how best to represent glyph outlines such that common characteristics between glyphs can be recognized, extracted and compared. We also need to do additional work on determining how fitness applies to the how fitness applies to the evolution of font faces and how it should be measured. Suggestions include nearness to a historical font class (Modern, Roman, Gothic, etc.) or legibility, although the latter remains rather subjective and therefore difficult to quantify computationally.

In the semantic direction, we are interested in breaking the text generation in two phases, the present one which operates at the character level and a new one which operates at the word level. This approach would require defining a hybrid of grammar-based and genetic models. The presented mechanisms for agent reproduction would be preserved and used to breed readable words. These words could then be assembled together according to a grammar-based system. Once words begin forming, lexical operations could be performed to drive mutations through different lexical spaces such as synonyms, antonyms and heteronyms. The newly formed word agents would employ grammatical operations to link themselves together into phrases and sentences. In order to integrate this new element into the biological metaphor, the grammar itself could be subject to algorithmic evolution.

10. ACKNOWLEDGEMENTS

11. REFERENCES

[1] Bulhak, A., The Dada Engine.
http://dev.null.org/dadaengine/

[2] Ishizaki, S., Multi-agent Model of Dynamic Design: Visualization as an Emergent Behavior of Active Design Agent. In *Proceedings of CHI '96 Human Factors in Computing Systems,* pp. 347 - 354, Vancouver, BC, April 1996.

[3] Leopoldseder, H. and Schopf, C. (eds), Cyberarts 2000. Ars Electronica Center, Linz, Austria, 2001

[4] Lewis, J. and Weyers, A., ActiveText: an architecture for creating dynamic and interactive texts. In the Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology Conference (Asheville, North Carolina, November 1999), ACM Press, 131-140.

[5] Lewis, J., Dynamic Poetry: Introductory remarks to a new medium. M. Phil. thesis, Royal College of Art, 1996.

[6] Lewis, J., NextText Java Library, www.obxlabs.net/research/nexttext/

[7] Mitchell, T., Machine Learning, McGraw-Hill, 1997

[8] Mitchell, M., An Introduction to Genetic Algorithms, MIT Press, 1999

[9] Nearlife, The Virtual Fishtank, http://www.virtualfishtank.com/press/index.html

[10] Pechawis, A., Nation (multimedia performance). http://www.aalab.net/ephemera/resid/1.1Nation/, 2000

[11] Rubin, William S. Dada & Surrealist Art. Harry N. Abrams, Inc., NY, 1968.

[12] Schmitz, Genotyp. http://www.genotyp.com

[13] Seaman, B., Recombinant Poetics: Emergent meaning as examined and explored within a specific generative virtual environment. Doctoral dissertation, Center for the Advanced Inquiry into the Interactive Arts, 1999

[14] Sha, XinWei, Hubbub, http://www.gvu.gatech.edu/people/sha.xinwei/topologicalmedia/hubbub/index.html

[15] Sims, K., Galapagos, http://www.genarts.com/galapagos/

[16] Soo, D., Implementation of a temporal typography system. Masters thesis, Massachusetts Institute of Technology, 1997.

[17] Wong, Yin Yin. Temporal Typography: Characterization of time-varying typographic forms. Masters thesis, Massachusetts Institute of Technology, 1995.