TIERRA

(Spanish for Earth)

Thomas S. Ray, 1995, Evolution, Ecology and Optimization of Digital Organisms: Biology Department, University of Delaware, Newark, Delaware 19716

Extracted and Adapted from http://www.isd.atr.co.jp/~ray/pubs/tierra/
Thomas S. Ray, 1995, Evolution, Ecology and Optimization of Digital Organisms:
Biology Department, University of Delaware, Newark, Delaware 19716

"The intent of this work is to synthesize rather than simulate life"

This approach starts with hand crafted organisms already capable of replication and open-ended evolution, and aims to generate increasing diversity and complexity in a parallel to the Cambrian explosion.

While the origin of life is generally recognized as an event of the first order, there is another event in the history of life that is less well known but of comparable significance: the origin of biological diversity and macroscopic multicellular life during the Cambrian explosion 600 million years ago. This event involved a riotous diversification of life forms. Dozens of phyla appeared suddenly, many existing only fleetingly, as diverse and sometimes bizarre ways of life were explored in a relative ecological void.

The work presented here aims to parallel the second major event in the history of life, the origin of diversity. Rather than attempting to create prebiotic conditions from which life may emerge, this approach involves engineering over the early history of life to design complex evolvable organisms, and then attempting to create the conditions that will set off a spontaneous evolutionary process of increasing diversity and complexity of organisms. This work represents a first step in this direction, creating an artificial world which may roughly parallel the RNA world of self-replicating molecules (still falling far short of the Cambrian explosion).

The approach has generated rapidly diversifying communities of self-replicating organisms exhibiting open-ended evolution by natural selection. From a single rudimentary ancestral creature containing only the code for self-replication, interactions such as parasitism, immunity, hyper-parasitism, sociality and cheating have emerged spontaneously. This paper presents a methodology and some first results.

Other simulations, such as Core Wars programs, computer viruses, and worms are capable of self-replication, but fortunately, not evolution.

What is Core Wars?
http://mcraeclan.com/Graeme/CoreWars.htm

Core Wars is a game played by two or more programs (and vicariously by their authors) written in an assembly language called Redcode and run in a virtual computer called MARS(for Memory Array Redcode Simulator). The object of the game is to cause all processes of the opposing program to terminate, leaving your program in sole possession of the machine.

Popularized in May, 1984 by A. K. Dewdney's column in "Scientific American" magazine, this was actually devised by Victor Vyssotsky, Robert Morris Sr., and Dennis Ritchie in the early 1960s (their original game was called "Darwin" and ran on a PDP-1 at Bell Labs).

- ☞ Most evolutionary simulations are not open-ended. Their potential is limited by the structure of the model, which generally endows each individual with a genome consisting of a set of pre-defined genes, each of which may exist in a pre-defined set of allelic forms.
- ☞ The object being evolved is generally a data structure representing the genome, which the simulator program mutates and/or recombines, selects, and replicates according to criteria designed into the simulator.
- ☞ The data structures do not contain the mechanism for replication, they are simply copied by the simulator if they survive the selection phase.

Tierra is based on a different principle

- ☞ Although the model is limited to the evolution of creatures based on sequences of machine instructions, this may have a potential comparable to evolution based on sequences of organic molecules.
- ☞ Sets of machine instructions similar to those used in the Tierra Simulator have been shown to be capable of ``universal computation" and in this sense are like Turing's Universal Computer.
- ☞ This suggests that evolving machine codes should be able to generate any level of complexity.
- ☞ In other words, what is evolving here is the computer's code, the instructions usually written by the programmer. Only here, once the initial replicator is created it is capable of self-evolving its own code to enhance its own survival.
    - ▸ Because of this the simulation is done in a virtual computer to prevent the evolving code from escaping and continuing to evolve in other computers.

Organic life is viewed as utilizing energy, mostly derived from the sun, to organize matter. By analogy, digital life can be viewed as using CPU (central processing unit) time, to organize memory.

ORGANIC LIFE:	DIGITAL LIFE:
Utilizes energy.	Uses CPU time
to organize matter	to organize memory

- ☞ Organic life evolves through natural selection as individuals compete for resources (light, food, space, etc.) such that genotypes which leave the most descendants increase in frequency.
- ☞ Digital life evolves through the same process, as replicating algorithms compete for CPU time and memory space, and organisms evolve strategies to exploit one another. CPU time is thought of as the analog of the energy resource, and memory as the analog of the spatial resource.

The creature consists of a self-replicating assembler language program.

- ☞ Assembler languages are merely mnemonics for the machine codes that are directly executed by the CPU.

In the biological analogy, the machine instructions are considered to be more like the amino acids than the nucleic acids, because they are ``chemically active".

- ☞ They actively manipulate bits, bytes, CPU registers, and the movements of the instruction pointer (see below).
- ☞ The digital creatures discussed here are entirely constructed of machine instructions.
- ☞ They are considered analogous to creatures of the RNA world, because the same structures bear the ``genetic" information and carry out the ``metabolic" activity.

A block of RAM memory (random access memory, also known as ``main" or ``core" memory) in the computer is designated as a ``soup" which can be inoculated with creatures.

- ☞ The ``genome" of the creatures consists of the sequence of machine instructions that make up the creature's self-replicating algorithm.
- ☞ The prototype creature consists of 80 machine instructions, thus the size of the genome of this creature is 80 instructions, and its ``genotype" is the specific sequence of those 80 instructions.
- ☞ In its present manifestation, the Tierran language consists of 32 instructions, which can be represented by five bits, operands included.

The Tierran Operating System

Memory Allocation:

The Tierran computer operates on a block of RAM of the real computer which is set aside for the purpose. This block of RAM is referred to as the ``soup". In most of the work described here the soup consisted of about 60,000 bytes, which can hold the same number of Tierran machine instructions. Each ``creature" occupies some block of memory in this soup.

Cellularity:

An analog to the cell membrane is needed in digital organisms in order to preserve the integrity of the informational structure from being disrupted easily by the activity of other organisms.

☞ Tierran creatures are considered to be cellular in the sense that they are protected by a ``semi-permeable membrane" of memory allocation.

☞ Each creature has exclusive write privileges within its allocated block of memory. The ``size" of a creature is just the size of its allocated block (e.g., 80 instructions)

☞ This ``membrane" is described as ``semi-permeable" because while write privileges are protected, read and execute privileges are not. A creature may examine the code of another creature, and even execute it, but it can not write over it.

Time Sharing:

The Tierran operating system must be multi-tasking (or parallel) in order for a community of individual creatures to live in the soup
simultaneously.

☞ The Tierran operating system must be multi-tasking (or parallel) in order for a community of individual creatures to live in the soup simultaneously.

☞ The system doles out small slices of CPU time to each creature in the soup in turn.

☞ The creatures have to compete over the use of CPU time, with those who lost being eliminated from memory.

☞ Since there was a competition over CPU time, those creatures with the fewest number of instructions would stand the best chance of surviving (and thus reproducing.)

The system maintains a circular queue called the ``slicer queue". As each creature is born, a virtual CPU is created for it, and it enters the slicer queue just ahead of its mother, which is the active creature at that time. Thus the newborn will be the last creature in the soup to get another time slice after the mother, and the mother will get the next slice after its daughter.

The number of instructions to be executed in each time slice may be set proportional to the size of the genome of the creature
being executed, raised to a power.

☞ If the ``slicer power" is equal to one, then the slicer is size neutral, the probability of an instruction being executed does not depend on the size of the creature in which it occurs.

- ☞ If the power is greater than one, large creatures get more CPU cycles per instruction than small creatures.
- ☞ If the power is less than one, small creatures get more CPU cycles per instruction.

The power determines if selection favors large or small creatures, or is size neutral. A constant slice size selects for small creatures.

MORTALITY - THE REAPER

The Tierran operating system includes a ``reaper'' which begins ``killing'' creatures from a queue when the memory fills to some specified level (e.g., 80%).

- ☞ Creatures are killed by deallocating their memory, and removing them from both the reaper and slicer queues.
- ☞ Their ``dead'' code is not removed from the soup.

In the present system, the reaper uses a linear queue. When a creature is born it enters the bottom of the queue. The reaper always kills the creature at the top of the queue.

- ☞ Individuals may move up or down in the reaper queue according to their success or failure at executing certain instructions.
- ☞ When a creature executes an instruction that generates an error condition, it moves one position up the queue, as long as the individual ahead of it in the queue has not accumulated a greater number of errors.
- ☞ Two of the instructions are somewhat difficult to execute without generating an error, therefore successful execution of these instructions moves the creature down the reaper queue one position, as long as it has not accumulated more errors than the creature below it.

The effect of the reaper queue is to cause algorithms which are fundamentally flawed to rise to the top of the queue and die. Vigorous algorithms have a greater longevity, but in general, the probability of death increases with age.

MUTATION: Mutations occur in two circumstances.

1. At some background rate, bits are randomly selected from the entire soup (e.g., 60,000 instructions totaling 300,000 bits) and flipped.
   - ▸ This has the effect of preventing any creature from being immortal, as it will eventually mutate to death.

2. In addition, while copying instructions during the replication of creatures, bits are randomly flipped at some rate in the copies.
   - ▸ The copy mutation rate is the higher of the two, and results in replication errors. The copy mutation rate has generally been set at about one bit flipped for every 1,000 to 2,500 instructions moved.
   - ▸ In both classes of mutation, the interval between mutations varies randomly within a certain range to avoid possible periodic effects.

It turns out that bit flipping mutations and flaws in instructions are not necessary to generate genetic change and evolution, once the community reaches a certain state of complexity. Genetic parasites evolve which are sloppy replicators, and have the effect of moving pieces of code around between creatures, causing rather massive rearrangements of the genomes. The mechanism of this ad hoc sexuality has not been worked out, but is likely due to the parasites' inability to discriminate between live, dead or embryonic code.

The Tierran Ancestor

The ancestor is a single self-replicating program which is 80 instructions long.

- ☞ The ancestor is a minimal self-replicating algorithm.
- ☞ No functionality was designed into the ancestor beyond the ability to self-replicate, nor was any specific evolutionary potential designed in.

Reproduction

- ☞ The ancestor examines itself to determine where in memory it begins and ends.
- ☞ Having determined the address of its beginning and its end, it subtracts the two to calculate its size, and allocates a block of memory of this size for a daughter cell.
- ☞ It then calls the copy procedure which copies the entire genome into the daughter cell memory, one instruction at a time.
- ☞ When the genome has been copied, it executes the DIVIDE instruction, which causes the creature to lose write privileges on the daughter cell memory, and gives an instruction pointer to the daughter cell.

The Results

Evolutionary runs of the simulator are begun by inoculating the soup of about 60,000 instructions with a single individual of the 80 instruction ancestral genotype.

- ☞ The original ancestral cell executes 839 instructions in its first replication, and 813 for each additional replication.
- ☞ The initial cell and its replicating daughters rapidly fill the soup memory to the threshold level of 80% which starts the reaper.
- ☞ Typically, the system executes about 400,000 instructions in filling up the soup with about 375 individuals of size 80 (and their gestating daughter cells).
- ☞ Once the reaper begins, the memory remains roughly 80% filled with creatures for the remainder of the run.

The arms race described below took place over a period of a billion instructions executed by the system.

- ☞ The remarkable number of other "clever" viruses the sprung up was astounding.
- ☞ Most observations on the diversity of Tierran creatures have been based on the diversity of size classes. Creatures of different sizes are clearly genetically different, as their genomes are of different sizes.

☞ In a run of 526 million instructions, 366 size classes were generated, 93 of which achieved abundances of five or more individuals. In a run of 2.56 billion instructions, 1180 size classes were generated, 367 of which achieved abundances of five or more.

☞ Each size class consists of a number of distinct genotypes which also vary over time. There exists the potential for great genetic diversity within a size class. There are 3280 distinct genotypes of size 80

Long Term Evolutionary Patterns

1. Gradual Drop in Size: the mutations quickly caused the viruses to drop in size with only the smallest ones surviving.
   - They dropped in size to about 65 instructions before the process ended.

2. Parasites: But then something very odd happened - viruses appeared which were 45 instructions in length!
   - Ray considered this impossible at first because he estimated at least 60 instructions were needed for the virus to just be able to replicate itself. Then he figured out what had happened - the new viruses were parasites off the old ones... they would find the portion in memory of the larger virus's reproduction and use that.
   - This caused an evolutionary battle to occur where first the larger viruses would try to hide their locations in memory - until the parasites also developed to a point where they only had to find the location once and could then remember it.
   - Typically, parasites begin to emerge within the first few million instructions of elapsed time in a run.

3. Immunity to parasites:
   - At least some of the size 79 genotypes demonstrate some measure of resistance to parasites.
   - Freely evolving systems have been observed to become dominated by size 79 genotypes for long periods, during which parasitic genotypes repeatedly appear, but fail to invade.

3. Circumventing Immunity to Parasites:
   - Occasionally these evolving systems dominated by size 79 were successfully invaded by parasites of size 51.
   - When the immune genotype 0079aab was tested with 0051aao (a direct, one step, descendant of 0045aaa in which instruction 39 is replaced by an insertion of seven instructions of unknown origin), they were found to enter a stable cycle

4. Hyperparasites: after the parasites started getting strong, a hyper-parasite appeared. It was 77 instructions in length, and could track down any parasites that were feeding off of it, and steal their CPU time.
   1. These hyper-parasites became so powerful, that they quickly dominated the computer, driving the parasites to extinctions.
   2. As time went on, the hyper-parasites actually developed specialization areas and began to share information with one another. This "culture" easily dominated and lasted for millions of cycles.

6. Social Hyperparasites:
   - Hyper-parasites drive the parasites to extinction. This results in a community with a relatively high level of genetic uniformity, and therefore high genetic relationship between individuals in the community.
   - These are the conditions that support the evolution of sociality, and social hyper-parasites soon dominate the community.
   - Social hyper-parasites appear in the 61 instruction size class.

7. Cheaters: Hyper-Hyper Parasites: Then, all of a sudden, another form of parasite appeared.
   - This one was only 27 lines long! What it did was simply intercept all the information being passed around between the hyper-parasites and used it for its own advantage. Ray called these new parasites "cheaters" and they were indeed able to destroy everything that had developed because of the hyper-parasites.

Macro-Evolution

When the simulator is run over long periods of time, hundreds of millions or billions of instructions, various patterns emerge. Under selection for small sizes there is a proliferation of small parasites and a rather interesting ecology.

0045aaa is a ``metabolic parasite''. Its genome does not include the copy procedure, however it executes the copy procedure code of a normal host, such as the ancestor. In an environment favoring small creatures, 0045aaa has a competitive advantage over the ancestor, however, the relationship is density dependent. When the hosts become scarce, most of the parasites are not within the search limit of a copy procedure, and are not able to reproduce. Their calls to the copy procedure fail and generate errors, causing them to rise to the top of the reaper queue and die. When the parasites die off, the host population rebounds. Hosts and parasites cultured together demonstrate Lotka-Volterra population cycling.

A number of experiments have been conducted to explore the factors affecting diversity of size classes in these communities. Competitive exclusion trials were conducted with a series of self-replicating (non-parasitic) genotypes of different size classes. The experimental soups were initially inoculated with one individual of each size. A genotype of size 79 was tested against a genotype of size 80, and then against successively larger size classes. The interactions were observed by plotting the population of the size 79 class on the x axis, and the population of the other size class on the y axis. Sizes 79 and 80 were found to be competitively matched such that neither was eliminated from the soup. They quickly entered a stable cycle, which exactly repeated a small orbit. The same general pattern was found in the interaction between sizes 79 and 81.

When size 79 was tested against size 82, they initially entered a stable cycle, but after about 4 million instructions they shook out of stability and the trajectory became chaotic with an attractor that was symmetric about the diagonal (neither size showed any advantage). This pattern was repeated for the next several size classes, until size 90, where a marked asymmetry of the chaotic

attractor was evident, favoring size 79. The run of size 79 against size 93 showed a brief stable period of about a million
instructions, which then moved to a chaotic phase without an attractor, which spiraled slowly down until size 93 became extinct, after an elapsed time of about 6 million instructions.

Symbiotic relationships are also possible. The ancestor was manually dissected into two creatures, one of size 46 which contained only the code for self-examination and the copy loop, and one of size 64 which contained only the code for self-examination and the copy procedure (Figure 3). Neither could replicate when cultured alone, but when cultured together, they both replicated, forming a stable mutualistic relationship. It is not known if such relationships have evolved spontaneously.

Much of the evolution in the system consists of the creatures discovering ways to exploit one-another. The creatures invent their own fitness functions through adaptation to their biotic (``living") environment. These ecological interactions are not programmed into the system, but emerge spontaneously as the creatures discover each other and invent their own games. In the Tierran world, creatures which initially do not interact, discover means to exploit one another, and in response, means to avoid exploitation.