Biomorphs: Cumulative Selection and Genetic Mutations Form Unique Trees

Michele Degges
Indiana University Bloomington, Computer Science Undergraduate,
Bloomington, Indiana
(mrdegges@umail.iu.edu)

Abstract. This paper will discuss the fundamental aspects of biomorph creation. In Dawkins' original program, he added features that would evolve creatures and inanimate objects. The ultimate goal in this biomorph implementation is to see if Dawkins' program can be modified so that only specific organisms are evolved and displayed to the user. This problem was approached by first finding reliable biomorph pseudo-code, and then by piecing together a unique program in Python to model Dawkins' research. Once a working implementation was created, I went through a lengthy trial and error process to add constraints to the program. The results confirm that a biomorph program of this sort can be implemented. This shows that we can evolve creatures of our choosing, given a strict set of initial conditions and constraints.

Keywords: Dawkins, biomorphs, evolution, cumulative change, artificial selection, asexual reproduction, genetic mutations, genes, arrays, algorithms, decoding genes, artificial trees

1 Introduction

Biomorphs are computer generated families that all share similar genetic structures. Each family structure has one parent and four children. These children are replicas of their parent who reproduces asexually, except that one randomly chosen gene for each of the children is mutated to either increase it's integer value by one or negative one. Through a powerful genetic decoding and unpacking process, the family members are visually represented. The user who runs the program will be the Omniscient god, and will choose only one child to survive and reproduce as the new generation's parent. This parent's children will be produced the same way as in the last generation, and the cycle will continue until the user closes the program. The

members of each family will look very similar to one another at the first generation, but over time, the cumulative changes will be distinct.

Richard Dawkins created the first biomorphs program, and was stunned by the results. His biomorphs ranged from having the appearance of trees, to bats, flowers, beetles, and even inanimate objects, all from following a simple algorithm. Over time, many other biomorph programs have been created that evolve specific creatures and patterns. In this implementation, my goal was to evolve tree-like structures using a strict set of initial values and genetic restrictions. This goal has been reached. As you will see, the biomorph families indeed show the incredible amount of complexity and variation that result from artificial selection and random mutations over time.

2 Biomorphs and Evolution

Biomorphs are not created through the process of Darwinian evolution. They are not formed through natural selection, a key component in Darwinian evolution. In nature there are many environmental factors that directly impact the survival of certain species. Natural selection is a process in which individuals are more likely to survive and reproduce because of certain traits they have (camouflage, speed, skin color). While some species may prosper in certain environments, there must be a certain degree of variation to enable individuals with certain traits to survive and adapt to new, unforeseen environmental changes. The key components of Darwinian evolution (natural selection, variation, random mutations, environmental changes, inherited genes from parents, and adaptation) are not all accounted for in biomophs. Rather, as Dawkins points out, “As the generations go by, the total amount of genetic difference from the original ancestor can become very large, cumulatively, one small step at a time.. although the mutations are random, the cumulative change over the generations is not random” (Dawkins, p.56). The goal of biomorphs is to use asexual reproduction, artificial selection, and genetic mutations to study the effects of *cumulative* evolution.

3 Algorithm

The simple algorithm (modified from Source 1) used to create biomorph families is as follows:

1. Start with random values for the genes of the parent (within the range 0-5).
2. Allow it to asexually reproduce and create 4 children, where each child has exactly one random gene mutated to +1 or -1 its current value.
3. Display these children to the user, and the user will click on the child he fancies.
4. This child will become the parent for the next generation.
5. All the children will be remade according to step 2.
6. Repeat steps 1-5 until the user closes the program.

4 Genes as Arrays

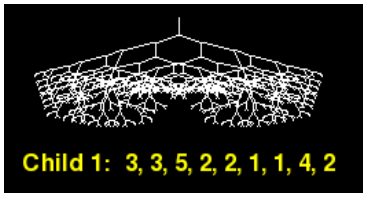
To create biomorph family members, each one must be assigned specific values that the computer can recognized. A simple way of doing this is to create each parent and it's children from genes, which are each represented as integers. Arrays are created to store the unique genetic sequences for each family member. In other words, the values of each individual gene control a particular element of the biomorphs' structure. The genes in this program control the angles of branching, the length of branches, the depth of branching, the number of branchings, etc. These all contribute to the overall shape of the biomorphs.

6 Role of Mutation

Mutation plays a huge role in the formation of biomorphs. My program follows Dawkins' description: each offspring will be an exact replica of the ancestor, with just one random gene (out of nine) being increased or decreased by a value of one. However, to allow the user to experiment with, I made the mutation rate (which is, by default, 1) variable. When this rate is increased, we can see much more variation between the ancestor and its offspring in a smaller amount of time (fewer generations).

5 Decoding the Genes

When we use the same decoding scheme as Source 1 suggests, our biomorphs repeatedly began to look like upside down trees, as shown in the image below:



This decoding scheme followed these rules:

```
#the nine genes are decoded to compute values for dx
#basic dx directions
dx[3] = gene[1]
dx[4] = gene[2]
dx[5] = gene[3]
#ensures y-axis symmetry
dx[1] = -dx[3]
```

```
dx[0] = -dx[4]
dx[7] = -dx[5]
#directions 2 and 6 along y-axis
dx[2] = 0
dx[6] = 0
#the nine genes are decoded to compute values for dy
#the way this is done ensures y-axis symmetry
dy[2] = gene[4]
dy[3] = gene[5]
dy[4] = gene[6]
dy[5] = gene[7]
dy[6] = gene[8]
dy[0] = dy[4]
dy[1] = dy[3]
dy[7] = dy[5]
```

For the parent biomorph, random gene values are chosen within a certain range. Then this array containing nine genes is passed to a setdX and setdY function. These functions produce arrays of various integers based on the gene values they are passed. In other words, these dx and dy arrays define the direction changes set up by the genes. Then these arrays are passed to the tree drawing function, and the parent biomorph is actually drawn. The offspring's dx and dy arrays are computed in a similar way. First, the offspring's genes are computed by replicating the parents genes, and then one random gene will be mutated. Then the dx and dy directions are created by passing each child's genetic array (containing nine genes) through the setdX and setdY functions. The result is then passed to the draw tree function. For more information about the specific computations performed in the tree drawing function, see section 8.

- The two arrays dx and dy define 8 directions

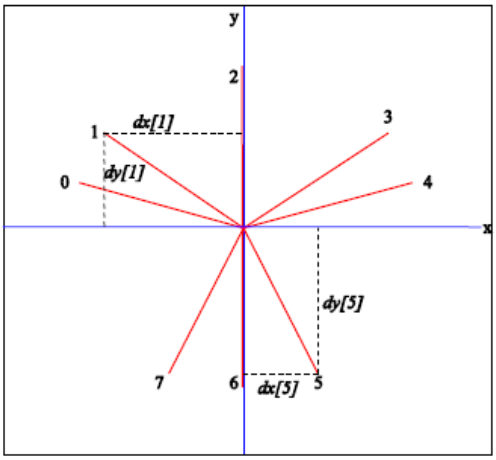


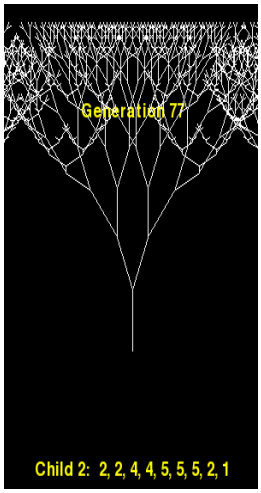
Fig. 1. This image shows the directions that the dx and dy values control. (Source 1)

Some of these dx and dy definitions had to be changed from the above decoding scheme to ensure that the biomorphs took on a tree-like appearance. After playing around with the numbers a bit, I found that changing these rules to the following reversed the “downward” effect:

```
dx[3] = gene[1]*scale
dx[4] = gene[2]*scale
dx[5] = gene[3]*scale

dy[2] = -gene[4]*scale
dy[3] = -gene[5]*scale
dy[4] = -gene[6]*scale
dy[5] = -gene[7]*scale
dy[6] = -gene[8]*scale
```

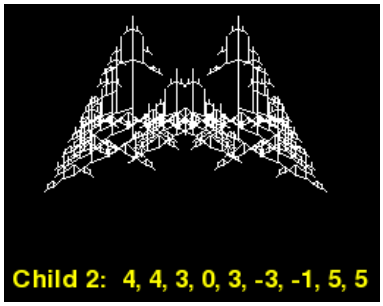
To keep the overall shape of the biomorphs contained, I used a scaling factor of 0.6. Notice that the trees can quickly become very tall and wide when this scaling effect is commented out of the code:



This is why it's important to have the scale down those genes that affect these traits. Similarly, consider an offspring's array of genes. The dx and dy arrays show significant changes when just one randomly chosen gene increases by one after a mutation. When these values are not contained, the offspring can quickly evolve into organism-like biomorphs.

7 Scaling the Genes

After choosing a random gene within the possible index ranges and mutating it (by adding one or negative one), the values are scaled. This is done by performing checks on the gene value at the mutated index. The values in Dawkins' basic program ranged from 0-9, but unless we scale down the trees as they grow larger, we will not be able to see their growth as they will be too large and grow past the width and height of the pygame board. To remedy this problem, I initialized a check that makes the sure the values of the genes do not go below 0 or above 5. It took quite a bit of trial and error to figure this out, but with different ranges you might be able to evolve all sorts of interesting creatures. The user is free to change these numbers in the mutate function and experiment with different ranges. For example, the biomorph below was evolved with constraints of -5 and 5. One could argue that this biomorph resembles a beetle:



8 Drawing the Trees

To draw the biomorphs, I chose to use Python's pygame library. I followed “The Tree Drawing Algorithm” from Source 1. This recursive algorithm takes six input values:

1. Initial x coordinate
2. Initial y coordinate
3. Initial length (which is the number of branches).
The higher the number, the longer it will take to compute and draw the biomorphs.
4. Initial direction (default is 2, which is up)
5. An array dx of length 8 (0-8)
6. An array dy of length 8 (0-8)

New coordinates are computed by adding the given coordinates to the product of the number of branches and the respective direction. A line is

drawn from the old coordinates to the new coordinates, and then the left and right sub trees are drawn recursively.

9 Unique Features

The differences between my program and Dawkins' are minor, but noticeable. Firstly, this program only implements nine genes instead of sixteen. Dawkins' added the other seven genes to control various other values. With just nine genes, you can see that no organism-like creatures are created. We only see trees, or somewhat more artificial looking trees.

I found it important to limit the range on the pygame board that users click on, for each biomorph. Since clicking on the parent would not do anything, I made that unclickable. I did this by first checking to see if a mouse click event happened, and then by I checking if the click was in a valid position for each of the offspring. Then the chosen child becomes the parent, the children become replicas of the parent with just one randomly mutated gene, and the process is repeated.

The gene values are also limited to positive integers, ranging from 0-5. This prevents large, organism-like biomorphs from being created, since the goal is to evolve trees.

10 Room for Growth

Because of the way this implementation is set up, it would be easy to extend the pygame board dimensions and add more biomorph children. This would give the user more options to choose from and would show more variation. Depending on your monitor size, you can also comment out the scaling and gene restrictions to let the biomorphs grow large and take on new forms. There are also a lot of new features that can be added in future versions, including color choices, viewing evolution history, editing individual genes, and saving certain genes to go back to later on.

13 Version History

13.1 Version 1: November 18

Created initial program using pygame, very similar in form to the standard genetic algorithm. At each iteration, the parent bred with user-selected child (fitness was determined based on mouse clicks). All biomorphs, with the exception of the parent, were created based on recombination mutation rates. This produced both tree-like and organsim-like structures.

13.2 Version 2: November 23

Deleted the breeding function which recombined the parent's genes with the user-selected child's genes. Dawkins' original approach was followed more closely. The parent's genes become the selected child's genes. Then each of it's offspring's genes become replicas, except for one randomly chosen gene in the array. This gene is mutated and is incremented by either one or negative one.

13.3 Version 2.2: December 3

Added text to the pygame board that shows the current generation number, the parent's name and genes, and it's offspring's names and genes at every iteration.

13.4 Version 2.3: December 6

Scaled down the size of the biomorphs and restricted gene values between zero and five. This produces smaller biomorphs that are mostly tree-like in structure.

12 References

1. Lutz, Rudi. "Dawkins Biomorphs." *Sussex.ac.uk*. Web. <http://www.sussex.ac.uk/Users/rudil/FAI_WEB_PAGES/DawkinsBiomorphs.pdf>.

2. Dawkins, Richard. *The Blind Watchmaker: Why the Evidence of Evolution Reveals a Universe without Design*. New York: W.W. Norton &, 1996. Print.

3. "Biomorph." *The Jovial Monk's | Guide to Portland*. Web. <http://www.jovialmonk.com/biomorph/iphone.html>.

4. Gregory, Michael. "Natural Selection." *Clintoncc.suny.edu*. Web. <http://faculty.clintoncc.suny.edu/faculty/michael.gregory/files/bio%20101/bio%20101%20lectures/Natural%20Selection/natural.htm>.

5. Rennard, Jean-Philippe. "Introduction to Biomorphs, Dawkins and the Blind Watchmaker." *Rennard.org*. Web. <http://www.rennard.org/alife/english/biomintrgb.html>.

6. Degges, Michele. "Biomorph Implementation in Python." *Stackoverflow.com*. 28 Nov. 2011. Web. <http://stackoverflow.com/questions/8299966/biomorph-implementation-in-python>.