SURVIVAL MACHINE

new music for flute and computer derived from the gene neuroligin 4, Y-linked
composed by David Morneau

Concept

In his book *The Selfish Gene*, Richard Dawkins proposes that the primary agent of evolutionary development and selection is the gene. Our genes began their existence as simple chemical compounds with the improbable ability to replicate themselves. These replicators gradually became more complex and their chemical resources became scarce, forcing them to compete for their survival. Dawkins argues that the two primary survival mechanisms became the development of protective barriers and the eventual cooperation between replicators, which led to the development of colonies of replicators working together for their survival.

Whatever became of the replicators? Where are they now?

> Now they swarm in huge colonies, safe inside gigantic lumbering robots, sealed off from the outside world, communicating with it by tortuous indirect routes, manipulating it by remote control. They are in you and in me; they created us, body and mind; and their preservation is the ultimate rationale for our existence. They have come a long way, those replicators. Now they go by the name of genes, and we are their **survival machines**. [Richard Dawkins]

DNA carries the codes for life in long series of chemical bases, of which there are only four types: adenine, guanine, thymine, and cytosine. How the coding actually works and the processes by which life results are the topics of much study and debate. The purpose of *Survival Machine* is to illustrate the process in a general and artistic manner. Also, it is an attempt to reconcile the friction between the absolute rigidity of the genetic code and the fluid, unpredictable experience of human existence.

Beginning with the raw data from the gene neuroligin 4, Y-linked, I developed a set of rules that would create sounds from the data coded in the DNA. These rules, once encased in an algorithm, generated sounds in three categories: quick moving, angular melodies; sustained tones that provide harmonic support; and percussive rhythms. The actions of the algorithm is a metaphor for protein synthesis.

In order to make these sounds audible, the algorithm—as equivalent to protein—had to interact with the electronic music equivalents of oxygen, carbon, and hydrogen; namely sine wave oscillators and a white noise generator. The actions of the algorithm to shape these raw elements into musical timbres was guided at all times by the raw nucleotide data from the gene.

Once these sounds were created I set about composing additional music for flute to be performed alongside the computer music. The materials created by the algorithm shaped and guided the composition of the flute part; some melodic ideas coming straight from the computer music. There are also passages where the flute struggles to break from the generated music, including a freely improvised passage in the last section of the piece.

We do not perceive and experience others as a collection of genetic codes and inherited traits. Instead it is the often intangible aspects of personality and patterns of behavior—both in others and in ourselves—that are the basis of our relationships. In the same way, this piece is more than the simple sonic animation of genetic code. It is an attempt to create music that is appealing for reasons other than the method of its genesis.

Data to Sound

This entire piece is being built from the data found on one gene on the Y chromosome: neuroligin 4 Y-linked (NLGN4-Y). I chose this gene since there is another version of it on the X chromosome (NLGN4-X), leaving open the possibility of a companion work.

I began by downloading raw nucleotide data from Ensembl Human at the Sanger Institute [NLGN4-Y here], one of several genome browsers available for researchers. This site displays the data with introns (non-coding segments) removed and with the exon divisions highlighted.

This is what the raw data looks like (exons alternate blue and black):

GAGACGAAGCAGGGAGAGAGTGAACTTCAGCCCCGTCCCCTCCCCACTGCCACGGCTGGG
GCAACCCAACCCGCGCCTGAAGCGGCTTGGCTTGACCTGCGGAAGCGCGGGCCGGGATGG
CGTGGGGAGAGGGAGGTAGGTGCCACTGGGCTGCAGATGACGAGTGGGTTGGGGGCTTGC
TGTGGGACAAGAGGTTCAGGTTCCGGCCTGCGCCTTCCACTCCGCGGTGGCGCTCTCTGC
CTGCGGTTTTCCAGGAGGCCGATCTACCCCAGGGACACTCTCATCCTTCAGGCGGTCTCC
TGGACGCCCTTTCCTCCCCTTGCCTCCCAGCCTGACCTGGCTCTTTCGCCCCTCGGAGAA
CCGGTTGCATTGGAGTTTTCGAAAGACTTATCTTTCTGCAGGCTCGCCTCTGAGCTTTGT
CTCCTTGGAGCCACCTCACTTAGACAGCTTCGGATGTGGATGCAGATTTGAACCATGTTG
CGTCCCCAGGGACTGCTATGGCTCCCTTTGTTGTTCACCTCTGTCTGTGTCATGTTAAAC
TCCAATGTTCTTCTGTGGATAACTGCTCTTGCCATCAAGTTCACCCTCATTGACAGCCAA
GCACAGTATCCAGTTGTCAACACAAATTATGGTAAAATCCAGGGCCTAAGAACACCATTA
CCCAGTGAGATCTTGGGTCCAGTGGAGCAGTACTTAGGGGTCCCCTATGCCTCACCCCCA
ACTGGAGAGAGGCGGTTTCAGCCACCAGAATCCCCATCCTCCTGGACTGGCATCCGAAAT
GCTACTCAGTTTTCTGCTGTGTGCCCCCAGCACCTGGATGAAAGATTCTTATTGCATGAC
ATGCTGCCCATCTGGTTTACCACCAGTTTGGATACTTTGATGACCTATGTTCAAGATCAA
AATGAAGACTGCCTTTACTTAAACATCTATGTGCCCATGGAAGATGATATTCATGAACAG
AACAGTAAGAAGCCTGTTATGGTCTATATCCATGGGGGATCTTACATGGAGGGAACCGGT
AACATGATTGATGGCAGCATTTTGGCCAGCTATGGGAACGTCATCGTTATCACCATTAAC
TACCGTCTGGGAATACTAGGGTTTTTAAGTACCGGTGACCAGGCAGCAAAAGGCAACTAT
GGGCTCCTGGATCAGATTCAAGCACTGAGGTGGATTGAGGAGAATGTCGGAGCCTTTGGC
GGGGACCCCAAGAGAGTGACTATCTTTGGCTCGGGGGCTGGGGCCTCCTGTGTCAGCCTG
TTGACCCTGTCCCACTACTCAGAAGGTCTCTTCCAGAAGGCCATCATTCAGAGCGGCACT
GCCCTGTCCAGCTGGGCAGTGAACTACCAGCCGGCCAAGTACACTCGGATATTGGCAGAC
AAGGTCGGCTGCAACATGCTGGACACCACGGACATGGTAGAATGTCTGAAGAACAAGAAC
TACAAGGAGCTCATCCAGCAGACCATCACCCCGGCCACCTACCACATAGCCTTTGGGCCG
GTGATCGACGGCGACGTCATCCCAGACGACCCCCAGATCCTGATGGAGCAAGGCGAGTTC
CTCAACTACGACATCATGCTGGGCGTCAACCAAGGGGAAGGCCTGAAGTTCGTGGACGGC
ATCGTGGATAACGAGGACGGTGTGACGCCCAACGACTTTGACTTCTCCGTGTCCAACTTC
GTGGACAACCTTTACGGCTACCCTGAAGGGAAAGACACTTTGCGGGAGACTATCAAGTTC
ATGTACACAGACTGGGCCGATAAGGAAAACCCGGAGACGCGGCGGAAAACCCTGGTGGCT
CTCTTTACTGACCATCAGTGGGTGGCCCCCGCCGTGGCCACCGCCGACCTGCACGCGCAG
TACGGCTCCCCCACCTACTTCTATGCCTTCTATCATCACTGCCAAAGCGAAATGAAGCCC
AGCTGGGCAGATTCGGCCCATGGCGATGAAGTCCCCTATGTCTTCGGCATCCCCATGATC
GGTCCCACAGAGCTCTTCAGTTGTAATTTCTCCAAGAACGACGTCATGCTCAGTGCCGTG
GTGATGACCTACTGGACGAACTTCGCCAAAACTGGTGATCCAAACCAACCAGTTCCTCAG
GATACCAAGTTCATTCATACAAAACCCAATCGCTTTGAAGAAGTGGCCTGGTCCAAGTAT
AATCCCAAAGACCAGCTCTATCTGCATATTGGCTTGAAACCCAGAGTGAGAGATCACTAC
CGGGCAACGAAAGTGGCTTTCTGGTTGGAATTGGTTCCTCATTTGCACAACTTGAACGAG

ATATTCCAGTATGTTTCAACAACCACAAAGGTTCCTCCACCAGACATGACATCATTTCCC
TATGGCACCCGGCGATCTCCCGCCAAGATATGGCCAACCACCAAACGCCCAGCAATCACT
CCTGCCAACAATCCCAAACACTCTAAGGACCCTCACAAAACAGGGCCCGAGGACACAACT
GTCCTCATTGAAACCAAACGAGATTATTCCACCGAATTAAGTGTCACCATTGCCGTCGGG
GCGTCGCTCCTCTTCCTCAACATCTTAGCCTTTGCGGCGCTGTACTACAAAAAGGACAAG
AGGCGCCATGAGACTCACAGGCACCCCAGTCCCCAGAGAAACACCACAAATGATATCACT
CACATCCAGAACGAAGAGATCATGTCTCTGCAGATGAAGCAGCTGGAACACGATCACGAG
TGTGAGTCGCTGCAGGCACACGACACGCTGAGGCTCACCTGCCCTCCAGACTACACCCTC
ACGCTGCGCCGGTCGCCGGATGACATCCCATTTATGACGCCAAACACCATCACCATGATT
CCAAACACATTGATGGGGATGCAGCCTTTACACACTTTTAAAACCTTCAGTGGAGGACAA
AACAGTACAAATTTACCCCACGGACATTCCACCACTAGAGTATAGCTTTTCCCTATTTCC
CCTCCTATCCCTCTGCCCCTACTGCTCAGCAATGTAAAAGAGACAAATAAGGAGAAAGAA
AATCTCCAAACCAGGAATGTTTTTGTGCCACTGACTTTAGATAAAAATGCAAAAGGGCAG
TCATCCTGTCCCAGCAGACCCTTCTCATTGGCATTTTCCAGTATTGTGAGATCAATTTCT
GACCATATGAAATGTGAAAAGTATATGTTTCTGTTACAATACTGCTTTAAGATCTAAACC
ATGCCAACAGATGTTTCGTGTGACTAGGACATCACCATTTCAAGGAACTGTGTGTTTCCA
ACATCATGGTAGCAGCACACACTTCCAAAGCTCAGCCAGGGACACTTAATATTTTTTAAT
TACAATGGAAATTTAAACATTTTTATGTGGGCTACACAATGGATGGCTCTTCTTAAGTGA
AGAAAGACTCTATAGGCTTTTACACAGCACATGAAGCAGTAATCCAGAAAGAAGGAAATG
CAGAATTTTATTATCAAAGTAAGCGAATTGACTGTGCAGAAAAATTGTAGGGTTCTGTGG
AAGGAGGTATTCTGCCAGCCTGAACTATATTTAAGAAACTTTGTAAAAAATAAAAATGTA
TATAGCTGTGAGCTCAAACAAAAACTGCAGACAAACAAAAAAGAGAAAAGCTTTTATTTG
TGTTTTCAGTTTGAAAGAACTTTTAGCAAGGTTGTGCTTTCAAACACATATTAGTCCTAC
CACCTTAGTTCCTCTACAGCAAAAGAGGCTTTTCTTCTTAATTACATGTAAACAAAGACA
TGGGATTTTCTGACGTAAGATTTTCATTTGTAGGAATATGTGATGTCAAATGGAAGACTC
AGAAGTTTTGTGTGGCCTATTTCTCCCTGTCAGGTTGCACAGATGCATGTAGAGCATTCT
TAGGAGACCATTGTTTTAGAAAACTTTGATTTGTACATGTTAGTTTTCATGAAATTGCAA
CACAGAGATAGGTCCTAAAAGTGGAATGTATTTAAAACTTGTTGAATTAGACACACACAC
ACAGACACACACAAAGAATCAGCAGAGAAAACAAAATACAAGTCCTGTTCTGTAGTTCTT
GCCCTTTGAATATATTTGGGAAGAGTTGCTTCCTATTTCAGGACCCTGCCAAAAAAGAAG
AAAGCTTGCCTTTGGTGGGGCTATGCCCCTTGGAGTAAATACAGCTCTGTGTTCCCTAGC
AGCTGCCGGAGGATTTGGCTGATGAAGTACCTGCTCAGCTTAGCTAATCAGATTAAAGGA
AGACATGTATGTCTTTTGTTTAAGCACCTAGTCCCTTATGTATCAGTAAACAGGTTTTTA
AAAATCTTTTATGTCATTTATAGGATAAAACATATGCTTGTCTGAAAATATCACCTTTTG
TGGATTTATCTGATCACCAAATAATAAATATTAAGAAGAATGGGGGAAAAAGGATAGAAT
ATTAAAACTGCTTTGCATAGGTTTTTGGGGAAATTAGGATATCTTCACTGACAAGACACT
GAATGGAATTTATTCACCCATTTTAAATTGGTTACTTGGGGATCAGAGATTTGTCTCTCC
AACAGCTTGTGGTTTTCTTATTACTCATTTTCAGGAAAGTTTGTAGTATTACAAGGCAGA
AGGAAACACAGTAGCAATGGTTGCTCTATATTTTGTCTTTCAAAGATTACTGCATTACCA
AGAAACAGTAGCCAAAGATGTTTGAAGATCATGTCCCTTAGCTGCATTGTGGGTTATTCT
AGAAATCCAATGTTAAATGCCTCTACTAAAGTGGGGATTCCCCATAAAAATTGTCCAGCT
ACCTGACTCTTTTGCAATAACAACTTTGATTACTGAATCCATACACTCAAACTATAGTGA
TATATCAGTGTTTGGGAGTGACCTCTAGAAAAAAGAAAACTGTTTTTAGAAATACATAAA
ATCACTTCCAAATCCTGTTGCTTATGTTGGGTTAAATTTGAAAGCAATTCTCTATATATA
AATATGTGAAATATTATGATCTGAACTTAGCACACATGAAGCAACATTTCTTTGCTACAC
AGAGGTGTCTTGGAAAGATTTCATTCCCAATTCATTTTTCATAGATCTATAATCAGGCAA
TTTCTGCAAGCAATGTATGACCCCACCTGAGCAACCACAAATAGGCTCTCCATGAAACTG
CAAAGGAACTGATGTGTGGCATCCATGCTGGTTTTGTCTGTCTATAATATGAATTCAAGT
ATCTGTTCATATTTCCAATTGTCTCCTGCTAGCAATATGTGCCACAACATGACAGTCTTG

TGACATCTTAAGGAAAAGAAGAGTTCCTGTTAAATGAATAGCTTTAGCTTTTACAGGGGA
TTATGATTAAAAGTGATTTAGTACATCTT

The first step in translating the data was to separate it by exon, which allowed me to keep track of where the sectional divisions occur. This is how I will be generating the form for the music.

Next, the nucleotides were converted to codons, which is to say they were split into groups of three: ACA, TCA, TGC, etc. Each codon was then converted to a number from 1 – 64 (numbers are necessary for the Max/MSP algorithm, as will be seen). Here is a version of the chart I used. It includes my numbering system as well as the amino acid that each codon represents. (The amino acid data is used occasionally my the synthesis process.)

TTT = 1 Phenylalanine (Phe)	TCT = 2 Serine (Ser)
TAT = 3 Tyrosine (Tyr)	TGT = 4 Cysteine (Cys)
TTC = 5 Phe	TCC = 6 Ser
TAC = 7 Tyr	TGC = 8 Cys
TTA = 9 Leucine (Leu)	TCA = 10 Ser
TAA = 11 STOP	TGA = 12 STOP
TTG = 13 Leu	TCG = 14 Ser
TAG = 15 STOP	TGG = 16 Tryptophan (Trp)
CTT = 17 Leu	CCT = 18 Proline (Pro)
CAT = 19 Histidine (His)	CGT = 20 Arginine (Arg)
CTC = 21 Leu	CCC = 22 Pro
CAC = 23 His	CGC = 24 Arg
CTA = 25 Leu	CCA = 26 Pro
CAA = 27 Glutamine (Gln)	CGA = 28 Arg
CTG = 29 Leu	CCG = 30 Pro
CAG = 31 Gln	CGG = 32 Arg
ATT = 33 Isoleucine (Ile)	ACT = 34 Threonine (Thr)
AAT = 35 Asparagine (Asn)	AGT = 36 Serine (Ser)
ATC = 37 Ile	ACC = 38 Thr
AAC = 39 Asn	AGC = 40 Ser
ATA = 41 Ile	ACA = 42 Thr
AAA = 43 Lysine (Lys)	AGA = 44 Arg
ATG = 45 Methionine (Met)	ACG = 46 Thr
AAG = 47 Lys	AGG = 48 Arg
GTT = 49 Valine(Val)	GCT = 50 Alanine (Ala)
GAT = 51 Aspartic acid (Asp)	GGT = 52 Glycine (Gly)
GTC = 53 Val	GCC = 54 Ala
GAC = 55 Asp	GGC = 56 Gly
GTA = 57 Val	GCA = 58 Ala
GAA = 59 Glutamic acid (Glu)	GGA = 60 Gly
GTG = 61 Val	GCG = 62 Ala
GAG = 63 Glu	GGG = 64 Gly

The last step in preparing the data for the algorithm was to "fold" it into lists, mimicking the folding that occurs naturally in protein molecules. The three "stop" codons (# 11, 12, & 15) were used to separate the large list of data into smaller units. Each stop codon was placed at the beginning of the following list, creating four types: three that begin with a stop codon (11, 12, 15) and one that does not (i.e. the first list in each exon). The data for neuroligin 4 Y-linked now looks like this

exon 1

• 63 46 47 31 60 63 36 59 17 31 22 20 22 21 22 34 54 46 50 64 58 38 27 22 62 18 59 62 50 16 17 55 29 32 47 24 64 30 60 16 20 64 63 48 63 57 52 54 34 64 29 31 45 46 36 64 13 64 50 8 4 64 42 44 52 10 52 6 56 29 24 17 26 21 24 52 56 50 21 8 29 32 1 6 48 48 30 37 7 22 48 55 34 21 37 17 31 62 53 6 16 46 22 1 18 22 17 54 6 31 18 55 29 50 17 14 22 21 60 59 30;

exon 2

• 49 58 13 63 1 14 43 55 9 2 5 8 48 21 54 2 63 17 4 21 17 60 54 38 10 17 44 31 17 32 45 16 45 31 33;
• 12 38 45 13 20 22 31 60 29 25 16 21 18 13 13 5 38 2 53 4 53 45 9 39 6 35 49 17 29 16 41 34 50 17 54 37 47 5 38 21 33 55 40 27 58 31 3 26 49 53 39 42 35 3 52 43 37 31 56 25 44 42 26 9 22 36 63 37 13 52 26 61 63 31 7 9 64 53 22 3 54 10 22 26 34 60 63 48 32 1 31 26 26 59 6 26 6 6 16 34 56 37 28 35 50 34 31 1 2 50 61 8 22 31 23 29 51 59 44 5 9 13 19 55 45 29 22 37 16 1 38 38 36 13 51 34 13 45 38 3 49 27 51 27 35 59 55 8 17 7 9 39 37 3 61 22 45 59 51;

exon 3

• 41 5 45 39 39 36 47 47 18 49 45 53 3 37 19 64 60 2 7 45 63 60 38 52 39 45 33 51 56 40 33 13 54 40 3 64 39 53 37 49 37 38 33 39 7 20 29 60 41 25;

exon 4

• 52 1;
• 11 57 30 61 38 48 31 27 47 58 34 64 21 29 51 31 33 27 58 29 48 16 33 63 63 35 53 60 54 1 56 64 55 22 47 44 61 34 37 1 56 14 64 50 64 54 6 4 53 40 29 13 38 29 6 23 7 10 59;

exon 5

• 53 2 6 44 48 26 10 5 44 62 58 54 29 6 40 16 58 61 39 7 31 30 54 47 7 34 32 41 13 58 55 47 53 56 8 39 45 29 55 38 46 55 45 57 59 4 29 47 39 47 39 7 47 63 21 37 31 31 38 37 38 30 54 38 7 23 41 54 1 64 30 61 37 55 56 55 53 37 26 55 55 22 31 37 29 45 63 27 56 63 5 21 39 7 55 37 45 29 56 53 39 27 64 59 56 29 47 5 61 55 56 37 61 51 39 63 55 52 61 46 22 39 55 1 55 5 6 61 6 39 5 61 55 39 17 7 56 7 18 59 64 43 55 34 13 32 63 34 37 47 5 45 7 42 55 16 54 51 47 59 39 30 63 46 32 32 43 38 29 61 50 21 1 34 55 19 31 16 61 54 22 54 61 54 38 54 55 29 23 62 31 7 56 6 22 38 7 5 3 54 5 3 19 23 8 27 40 59 45 47 22 40 16 58 51 14 54 19 56 51 59 53 22 3 53 5 56 37 22 45 37 52 22 42 63 21 5 36 4 35 5 6 47 39 55 53 45 21 36 54 61 61 45 38 7 16 46 39 5 54 43 34;

exon 6

• 60 44 21 44 36 1 61 16 18 33 2 22 4 31 49 58 31 45 19 57 63 19 2;
• 15 63 38 33 49 9 59 39 1 51 13 7 45 9 49 5 45 43 13 27 23 44 51 48 6;
• 11 47 16 35 57 1 43 34 4;
• 12 33 44 23 42 23 42 55 42 23 43 59 10 58 63 43 42 43 7 47 6 4 2 57 49 17 54 17;
• 12 41 3 13 60 44 49 50 6 3 5 48 38 29 26 43 47 47 43 50 8 17 16 16 56 3 54 18 16 36 43 7 40 2 61 5 18 40 40 8 32 48 33 16 29 45 47 7 29 21 40 9 50 35 31 33 43 60 44 19 57 4 17 13 1 47 23 25 53 18 3 57 10 57 39 48 1 9 43 37 1 3 53 33 3 48 41 43 19 45 17 53;
• 12 43 3 23 17 13 16 33 3 29 37 38 43;
• 11;
• 11 41 9 44 44 45 64 59 43 60;

• 15 35 33 43 34 50 13 19 48 1 13 64 43 9 60 3 17 23;
• 12 27 55 34 59 16 35 9 5 38 19 1 43 13 49 34 16 60 10 63 33 4 21 6 39 40 13 16 1 2 3 7 10 1 10 60 47 1 57 57 9 27 56 44 48 43 23 36 40 35 52 8 2 41 1 4 17 10 47 33 34 58 9 26 44 39 36 40 27 44 4 13 47 37 45 6 17 40 8 33 61 52 3 2 44 35 26 45 9 35 54 2 34 43 61 64 33 22 19 43 35 4 26 50 38;
• 12 21 1 8 35 39 39 1 51 7;
• 12 37 19 42 21 43 25;
• 15;
• 12 3 37 36 49 16 63;
• 12 18 25 59 43 44 43 29 1 9 59 41 19 43 37 34 6 43 6 4 8 9 4 16 49 43 1 59 40 35 2 25 3 41 35 45;
• 12 35 33 45 37;
• 12 34;
• 15 23 42;
• 12 40 39 33 2 13 25 23 44 52 53 13 59 44 1 19 6 27 5 33 1 19 44 2 41 37 48 27 1 29 27 58 45 3 55 22 38;
• 12 58 38 42 35 48 21 6 45 43 29 27 48 39;
• 12 4 61 58 6 45 29 49 13 2 53 3 35 45 35 10 36 37 4 10 3 5 27 13 2 18 50 40 35 45 8 23 39 45 42 53 13;
• 12 19 17 47 59 47 47 36 6 4;
• 11 45 35 40 1 40 1 7 48 60 9;
• 12 9 43 61 33;
• 15 7 37;

Each type of list was used to generate sounds in a different manner. Where possible, the codons were used in combination rather than one per note or event. This allowed for a wider variety in the sounds and more closely mimics the function of DNA where codons—and even genes—work together to create single traits for their survival machine.

List type 0 (first list in each exon)
The codons in these lists are grouped in fours and are used to determine pitch and timbre for non-melodic tones. These tones appear in the music as accompaniment figures for the flute as well as for other lists. Also, some the the timbres are used for pitches in subsequent lists from the same exon.

The data fed through the patch is grouped into sets of four. The last number in the list will be duplicated to fill out the last group if the list is not able to be evenly divided. For the first exon the data now looks like this:

63 46 47 31;
60 63 36 59;
17 31 22 20;
22 21 22 34;
54 46 50 64;
58 38 27 22;
62 18 59 62;
50 16 17 55;
29 32 47 24;
64 30 60 16;
20 64 63 48;
63 57 52 54;
34 64 29 31;
45 46 36 64;
13 64 50 08;
04 64 42 44;

52 10 52 06;
56 29 24 17;
26 21 24 52;
56 50 21 08;
29 32 01 06;
48 48 30 37;
07 22 48 55;
34 21 37 17;
31 62 53 06;
16 46 22 01;
18 22 17 54;
06 31 18 55;
29 50 17 14;
22 21 60 59;
30 30 30 30;

Each group becomes one note with the first number determining pitch and the other three determining timbre. The patch used a basic Frequency Modulation (FM) synthesizer where the second and third numbers determine the amount of modulation applied to the note. This way, each note has a different sound—the amount of difference can be subtle though.

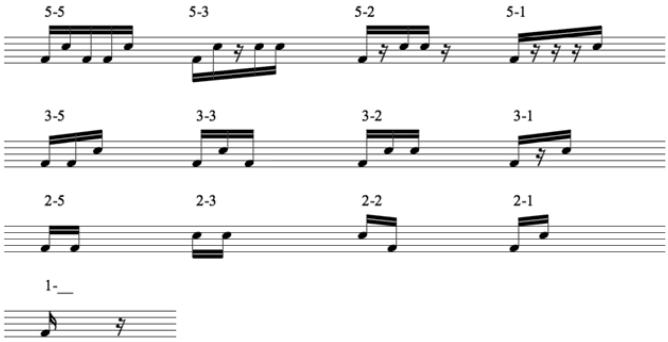
The fourth number selects an envelope shape for the modulation. The envelope acts as a volume controller that determines how much of the modulated sound gets mixed with the pure tone, making the sound dynamic rather than static. In the main patch (shown on the left above) the last three numbers in the groups also determine the overall amplitude envelope shape for the sound, choosing from sets of attack, sustain, and release shapes (I have chosen to include decay in the attack and sustain shapes). The process of choosing an envelope shape in all cases is determined by the amino acid that each codon represents (allowing for fewer total envelope shapes).

List type 11

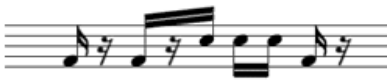
This list uses pairs of codons to select one of thirteen rhythmic cells. These cells are then combined to create longer rhythmic passages. On its own, this list will feed the rhythms through a filter that is attached to a noise generator. In the presence of list type 12, the results will also be used for the generated melodic tones.

The data in these lists determines rhythmic patterns. These patterns are realized by this patch and also in conjunction with list type 12. To determine the rhythm, each number is first converted to either 1, 2, 3, or 5. If the number is prime it becomes 1, if it can be divided by 2 then it becomes 2, likewise for 3 and 5. In the case of numbers that can be divided by more than one of 2, 3, and 5 (such as 12, 15, and 30) the number from the list is converted to the largest number it can be divided by—for example, 30 can be divided by 2, 3, and 5 so it is converted to 5.

Next, these numbers are paired, and each pair determines one rhythmic cell. The length of the cell is determined by the first of the pair and the number and type of articulations is determined by the second of the pair. I've prepared a chart to show each of the cells.



Turning for a moment to the second list on the fourth exon, we can see that the first eight numbers are: 11 57 30 61 38 48 31 27. Using the method detailed above this list is translated as: 1 3 3 1 2 3 1 3, which gives us this rhythm:



Depending on its context this rhythm is either applied to a melody generated by list type 12 or it is articulated on its own. The rhythms are articulated by the manipulation of a band-pass filter on the output of a noise generator. This process is commonly known as subtractive synthesis. By setting the filter to two different frequencies, a primitive electronic percussion sound is produced.

List type 12
The melodic list in the set, this list generates pitch for melodic figures. Of all the lists, this one is the most dependent on the others. Its timbre is taken from the first pitch in the list type 0 in the same exon. When there is a list type 11 in the same exon, rhythmic information will come from that list. Otherwise the rhythm for this list will be an undifferentiated string of 16th notes.

This list is least like the others as it translates every codon to its own note. Unless it is combined with the output from a list type 11 the notes occur at a constant rate (16th notes). Often the resulting melodic lines provide material for the flute part.

The patch for this list type uses Frequency Modulation (FM) synthesis like list type 0. This patch implements a simpler version of the synthesis, resulting in an even timbre from one note to the next.

List type 15
Each occurrence of this list will mimic the function of list occurring directly before it in the exon. This all-purpose list type models the interdependence of genetic material in the construction of a living being.

Sound to Music

Once the sounds were created I set to composing music for the flute part. In order to allow for a certain level of flexibility and expression I allowed myself to adjust the DNA sounds as follows:

- The rhythms for all sounds generated by list type 0 were freely composed, but no pitches could be repeated or reordered.
- Sounds from different lists could overlap freely within the same exon.
- List type 11 can repeat
- In the last section (exon 6) list type 12 can repeat, and the repetitions can be freely reordered, but not shortened.

While working with the material I was surprised—and pleased—to see pitch patterns emerge. For example, in the first section the pitches F, Bb, F#, appear twice in that order. Many other times a pitch will repeat (usually in a different octave). These patterns were all helpful when trying to create continuity within sections, as well as overall through the piece.

The flute music always relates to the DNA music in some way. In the open, recitative-like passages the flute lines use the computer pitches points or arrival and departure. I was also able to draw motifs from the list type 12 music that also provide abundant material for the flute.

In the last part (exon 6) a section of improvisation is included. This is natural in the context of the loop based music (due to repetitions of lists 11 and 12). And it provides one last element of tension between the strict coding the the DNA music and spur of the moment creativity.

In performance, the player will be provided with a foot pedal connected to a computer, controlling the playback of the DNA sounds. At certain points in the piece the computer will pause playback, allowing the player to catch up. Or more accurately: allowing the player to be expressive without running the risk of losing synch with the playback. Giving the player this control is appropriate, not only in general performance terms, but also in keeping with my metaphor. As Dawkins puts it:

> The genes too control the behavior of their survival machines, not directly with their fingers on puppet strings, but indirectly like the computer programmer. All they do is set it up beforehand; then the survival machine is on it's own and the genes can only sit passively inside… Like the chess programmer the genes have to 'instruct' their survival machines not in specifics, but in the general strategies and tricks of the living trade.
>
> Richard Dawkins. The Selfish Gene (Oxford University Press, 1976), pg. 56, 59.