First Contact -CM Adams

Background:

"First Contact" is a work for Violin and Electronics. The piece was developed over the course of a 3 day workshop at the Universitat der Kunste Berlin as part of the "Sound of Contagion" project, a multimedia collaborative endeavor exploring the interaction between AI and the arts. "First Contact" itself has no permanent score. Instead, the piece utilizes an evolutionary algorithm to develop procedurally unique material in a minimalistic, cell structure format. Due to the nature of the piece, in lieu of a score, several examples of score material and recordings are provided. If you wish to run the piece, the final section of the abstract contains instructions; for a more in depth look at the code in detail, see the video file "First_Contact_Demo".

Aesthetic Description:

The piece itself has a variable time frame, lasting as long as the performer wants. Over the course of the piece, the algorithm changes and develops as the algorithm listens to the audio it receives. These developments are meant to be aurally unpredictable, but computationally consistent, so that the piece always sounds new. The piece is played back and forth in sections where the performer plays what is generated, the computer responds according to what was played while new material is created. This pattern continues on and on like a rotating game of telephone where the end result is something you might guess but something you didn't. Fundamentally, the piece will always be sight-read by the performer.

Regarding musical aesthetics, the piece finds itself rather oddly between Cage and Stockhausen. The procedural design it implies draws from the two, combining elements of precision and entropy in unique ways. In general, I perceive the work as a very large fixed set of possibilities. Certain elements in the set tend to occur more than others, but all possibilities may occur. Much like a jazz lead sheet it may be hard to explain what is a correct or incorrect version of the work, but an experienced jazzer will nevertheless perform something appropriate. My philosophy of procedural design refines this method of composition. The material is apparent in the structure, but the algorithm allows the piece to maintain the unexpected feeling used with aleatoric composition.

Technical Description:

The work was developed in Ableton utilizing Max for Live, the BACH package for Max/MSP, and javascript. Ableton serves as the performance environment for the work with sends for effects that are employed as needed by the algorithm. Max/MSP, as used within Ableton as Max for Live, employs the logical component of the piece; Javascript assists by handling some computation for certain

numeric functions. The BACH package for Max/MSP handles the UI for the player; Much like score writers such as Sibelius, Finale, or MuseScore, BACH allows notation to be created easily for a staff. What makes BACH so useful is its capability to work in a generative environment, utilizing a LISP like syntax to represent the important information within a measure.

The algorithm starts by first randomly calibrating a series of weights that determine the prominence of certain properties within candidate solutions. Pitch content and rhythmic content are represented as simple integers. The work itself utilizes an evolutionary algorithm; candidate solutions are initially created by several generative functions. These solutions are then evolved according to the calibration of certain parameters. Within the algorithm, these certain calibrated parameters are known as ISAPs, or Incredibly Specific Arbitrary Parameters. ISAPs are the key root of generation; they provide the prototype that all candidate solutions strive to appease. As the piece continues, the weights associated with the ISAPs are adjusted. With each new generation of material, the piece develops, and each overarching version of the piece is itself determined by the initial generation.

- **The following section is the unedited READ_ME file used for implementing the algorithm by the performer or technician for the piece. If you don't wish to implement the algorithm or can't because you lack the software needed; Don't feel compelled to read this section.**

Implementation Instructions:
"First Contact" -CM Adams

_____

"First Contact" utilizes an evolutionary algorithm, a form of machine learning to generate score material for the violinist.
The piece itself explores the concept of moment form in a radical new way by not having any consistent score.
Instead, the piece creates a new score for the violinist several times throughout a performance.

The piece has been workshopped with a violinist, and given some small performance demos to a small crowd, but the algorithm
itself isn't perfect or entirely free from bugs. However, with that said it does work and this help file is intended to
troubleshoot any potential problems that may be encountered within the current version of the piece.

_____

The work itself is created as a Max for Live device using Ableton 10.
The patch utilizes Ableton mainly as an easier software for routing sends then building them entirely in Max.
The patch itself requires the embedded javascript files listed below:
-Lydian_Melody.js
-Dorian_Melody.js
-Major_Minored_Pent_Melody.js
-Major_Pent_Melody.js
-Note_Mutation.js
-Rhythmic_Generator.js
-Rhythmic_Mutation.js
-Manifestor.js
-Markov_Color.js
-Markov_Ambience.js
-Rhythm_MeanDifference_ISAP.js
-Rhythm_LengthModuloSum_ISAP.js
-Rhythm_Modulo_ISAP.ks
-ISAP_Pitch_Product_Difference.js

To include these files find the gui button on the bottom side of the max patch labeled "Show Containing Project"

This will open a separate gui. If the files listed above are NOT present in the project.
Navigate to the bottom left of this window, click add file to project, and click add existing file.
Navigate to where you have downloaded the js files and add them to the project.
Note that without any of these files the patch may not run or may not run as intended.

The other required component of the patch is the BACH package which is a Max package designed to incorporate score features into Max/Msp.
To see if this is installed check the package manager on the left side of the screen and navigate to installed packages. If it isn't installed,
install it using the package manager.

Now that you have checked that the files have been added to the patch. The algorithm is capable of generating material.
In presentation mode, to the left of the "s beginGeneration" object is a "bang" or button like object which will send a message to start
generation before you set this. I recommend incrementing the values marked below as "Pitch Number of Generations" and
"Rhythm Number of Generations". These values determine the number of generations that each subroutine of the algorithm will run through
the evolutionary algorithm.

A message box in the gui will then move from "Standby" to "Generating" and finally to "Ready to Play".
To reset the algorithm for another generation, click electronic phase then violin phase, then click the button next to "s beginGeneration"
as before.

Note that this is designed to work within context as a piece, and it has before. If you are struggling to reset the algorithm for further generation. I recommended simply closing and reopening the patch. This will work as well.

Note that algorithm also works using a set of weights that are initialized randomly and adjusted based on received audio from the violinist.
If results appear to be similar from generation to generation without closing the patch, it is because they most likely are, since the

weights of the algorithm have not been adjusted. Close the patch and reopen it to reinitialize the weights. The weights are randomly
initialized, so there is a chance that the material may still be similar but it is statistically improbable.

If the algorithm fails to generate material, double check that BACH has been installed and that all of the JS files have been added to the
patch. Furthermore, my submission includes screenshots of example generations to show you what the material looks like if all else fails.