A Gaussian Canon

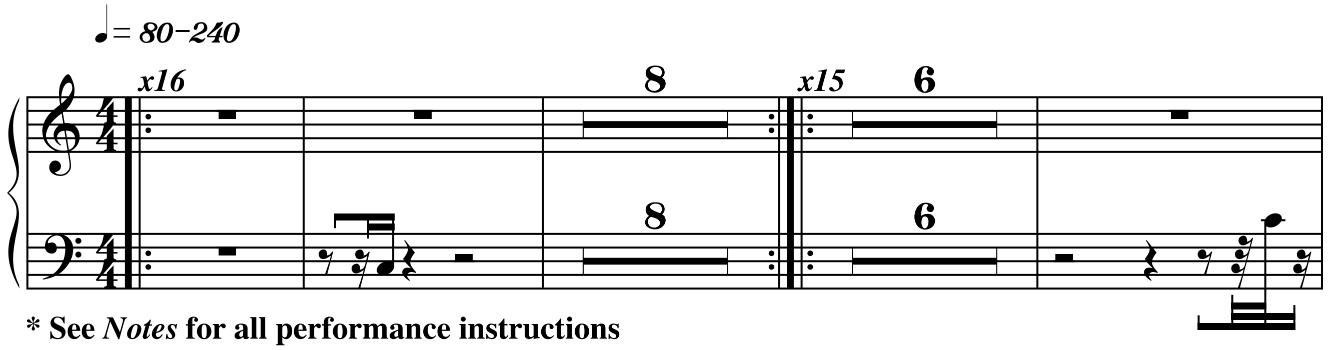
for Kathryn Anne Lutzner

Michael Benjamin Winter (2005)

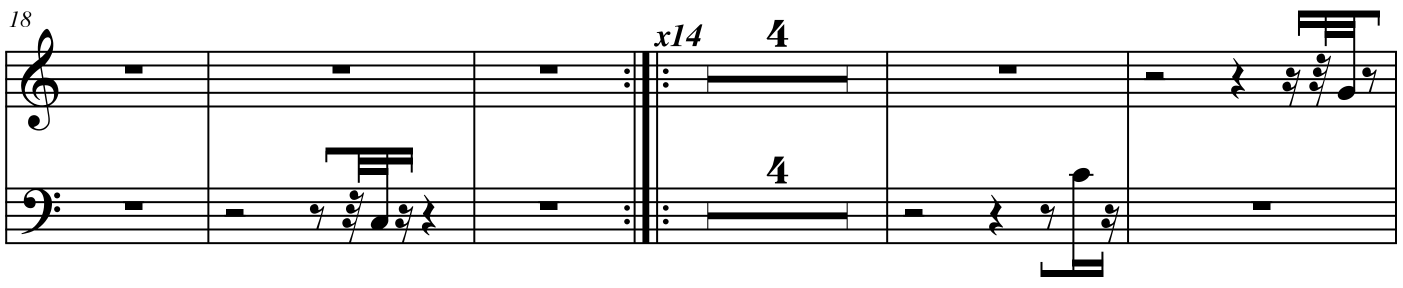
A Gaussian Canon

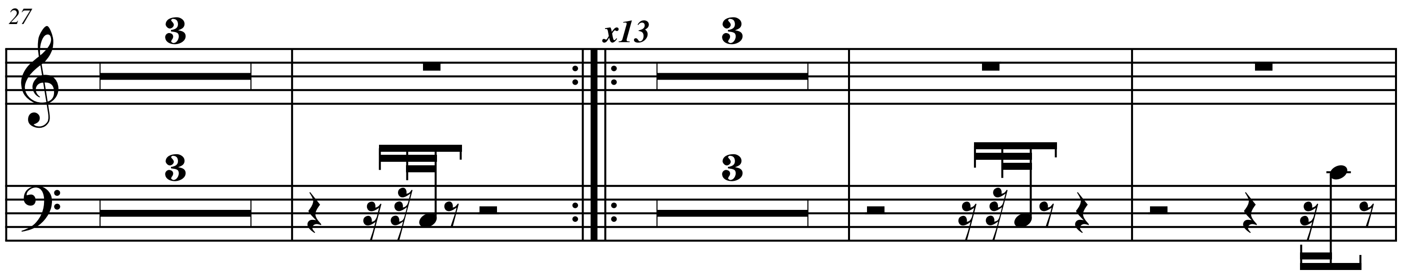
for Kathryn Anne Lutzner

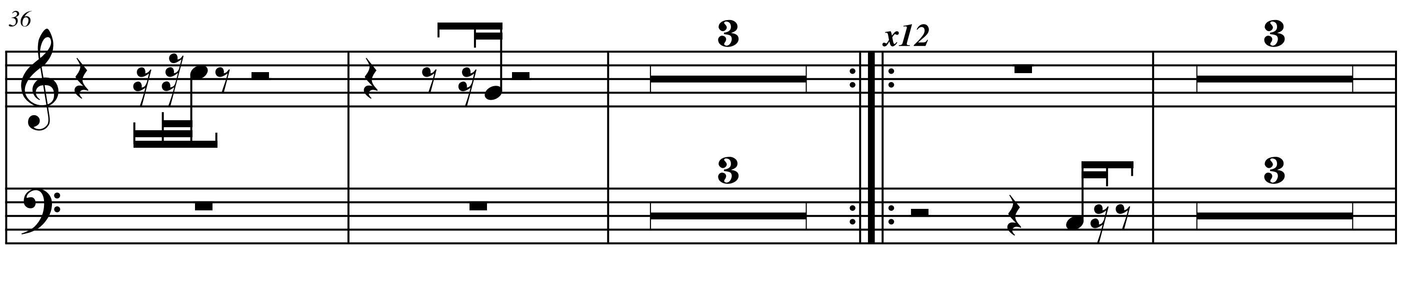
Michael Benjamin Winter(2005)

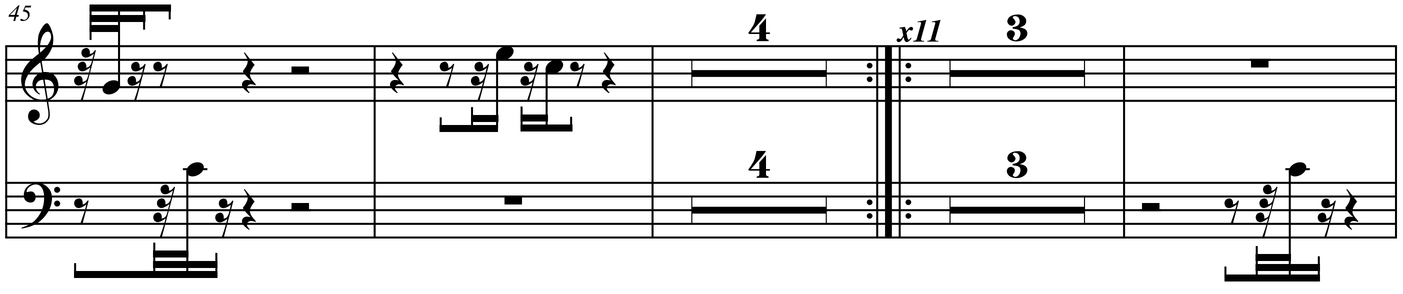


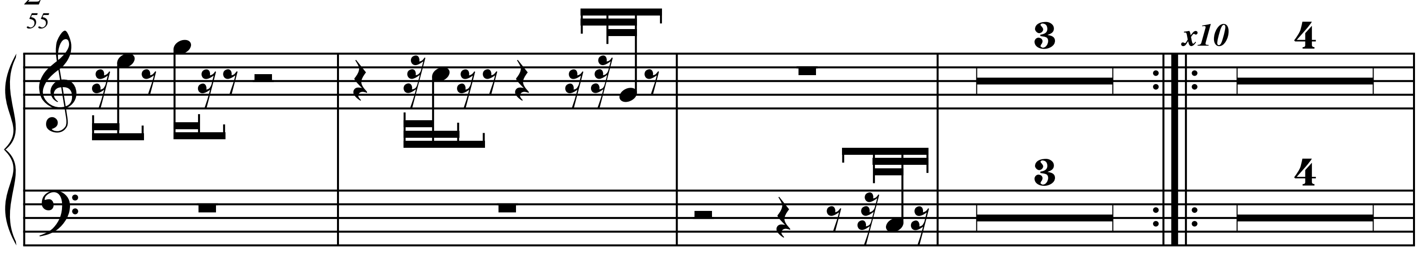
*** See *Notes* for all performance instructions**

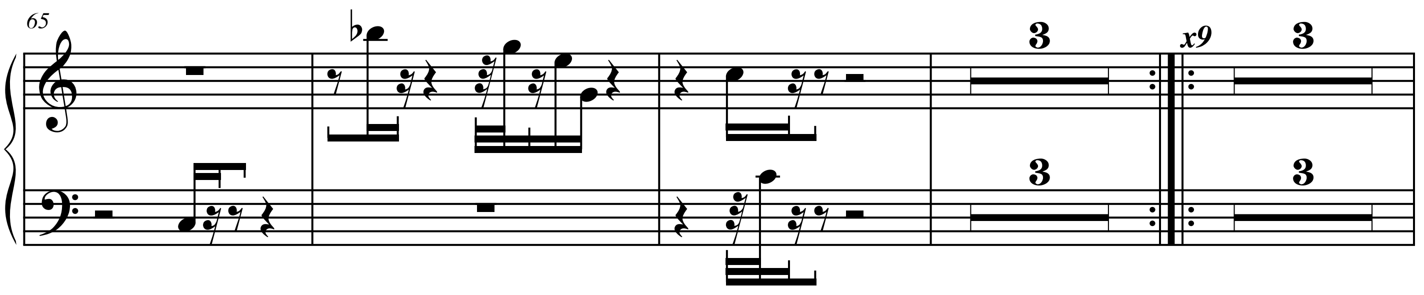


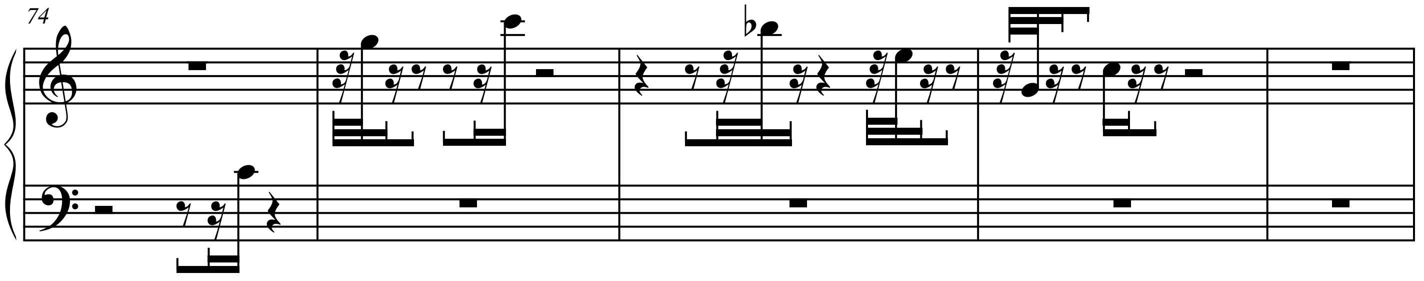


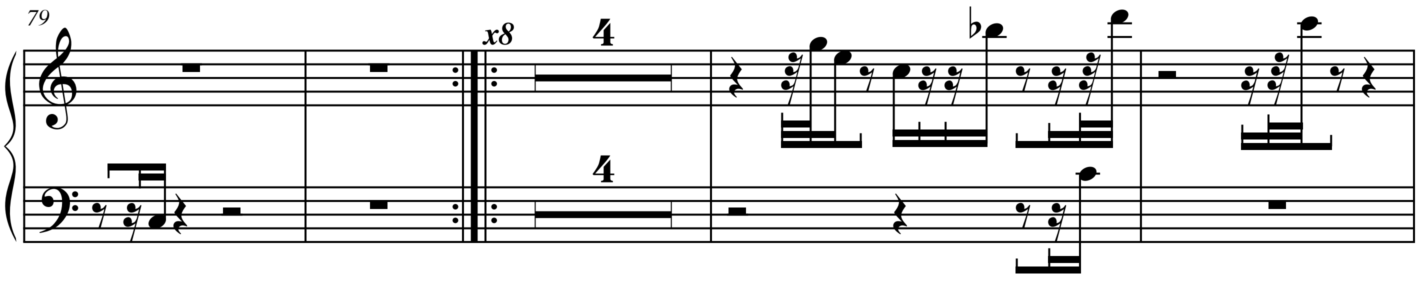


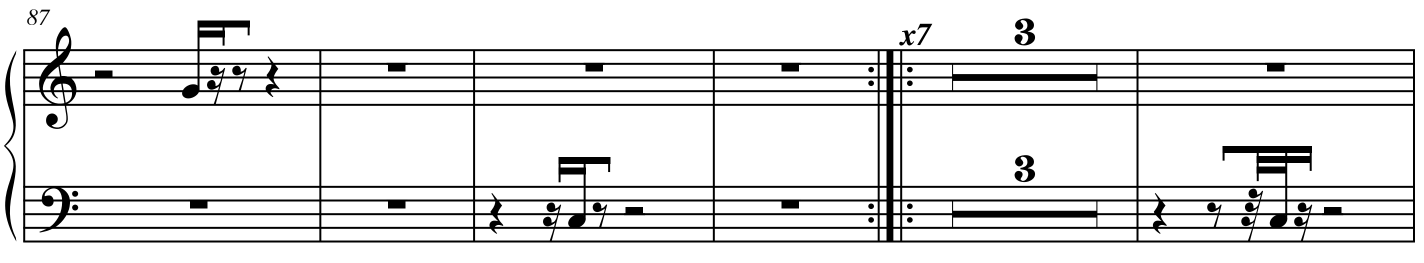


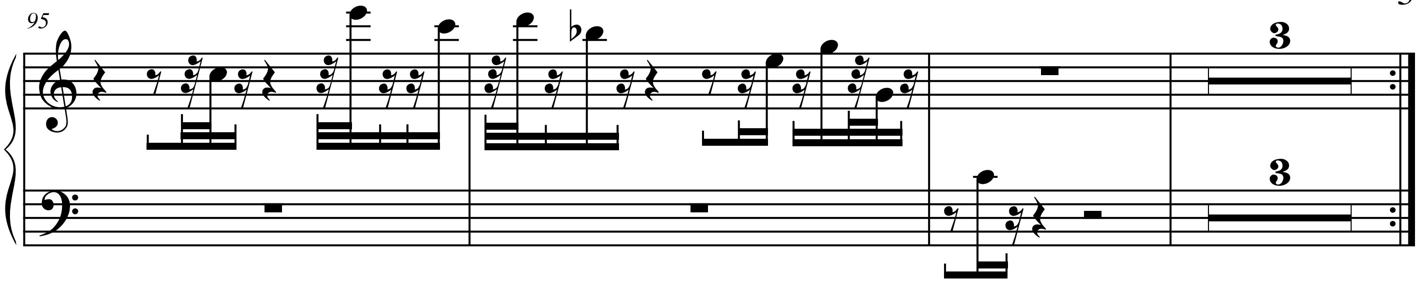

55
3
x10
4

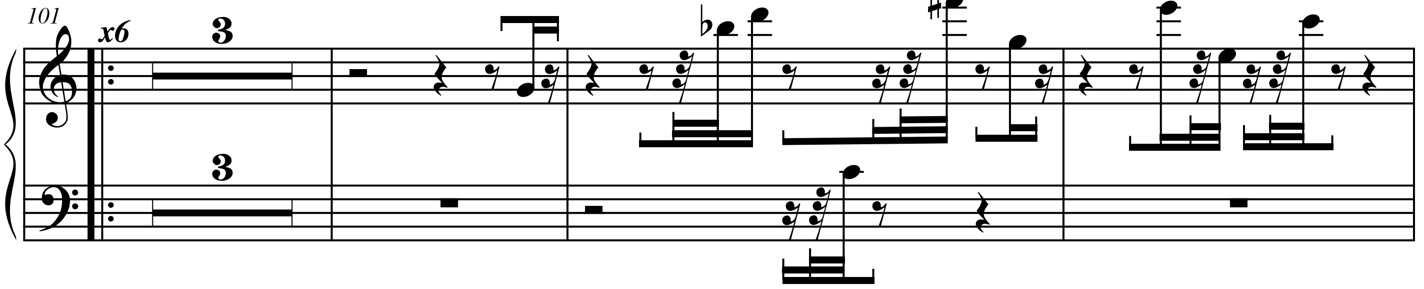

65
3
x9
3

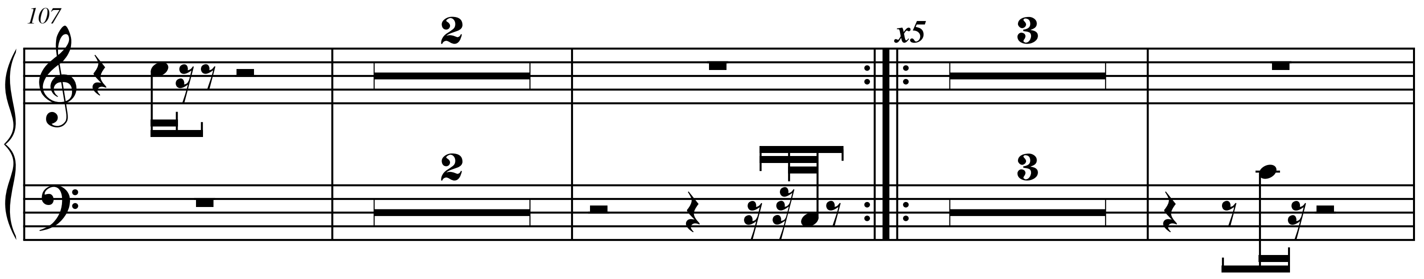

74

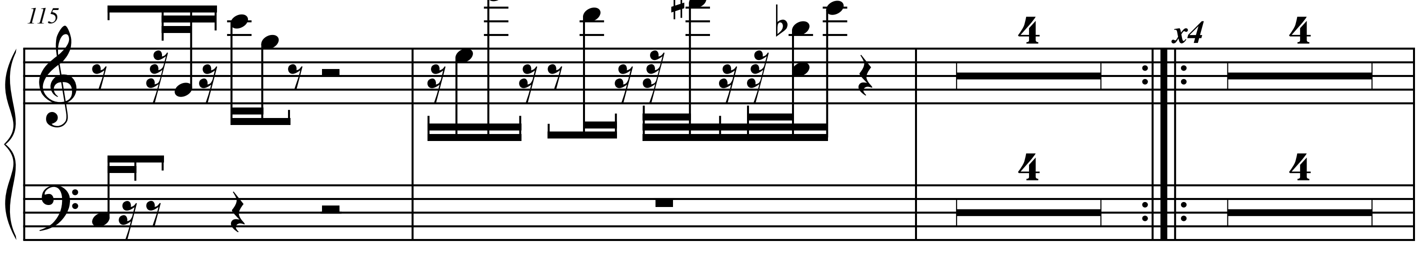

79
x8
4

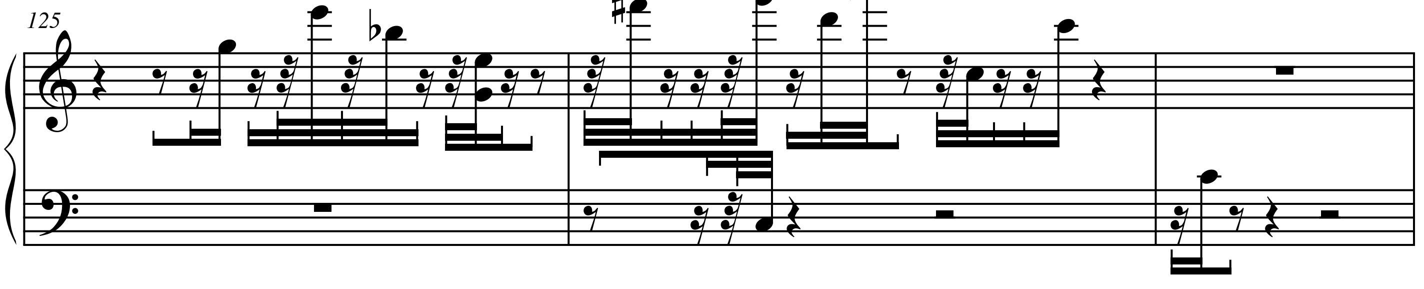

87
x7
3

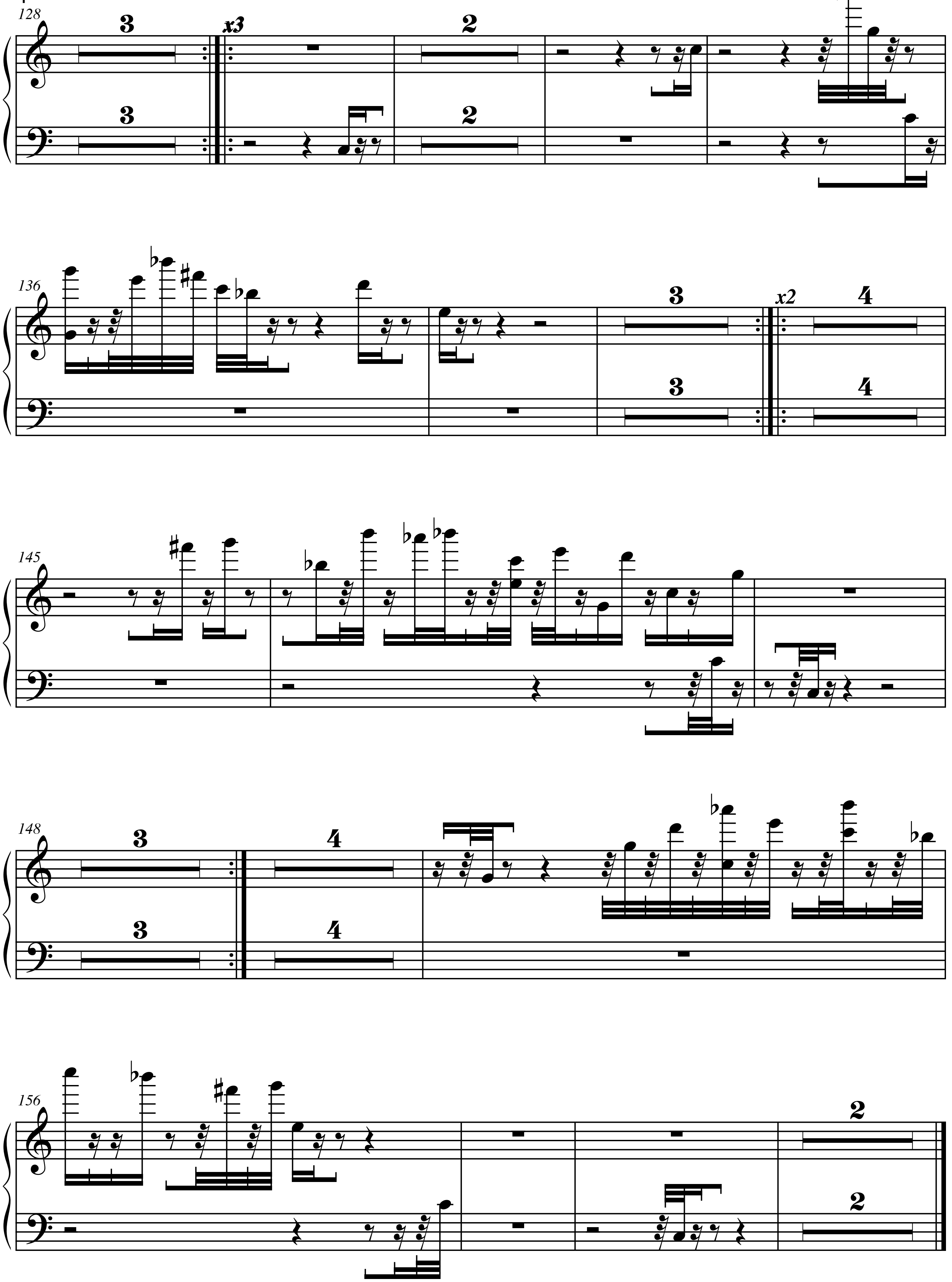

95
3
3


101
x6
3
3


107
2
2
x5
3
3


115
4
4
x4
4
4


125


128
3
3
x3
2
2
136
3
3
x2
4
4
145
148
3
3
4
4
156
2
2

A Gaussian Canon

for Kathryn Anne Lutzner

Michael Benjamin Winter (2005)

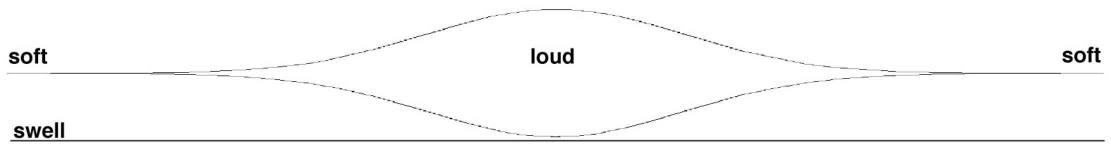
Notes

Performance Instructions/Realization

*Included is the score of a realization that I generated. This particular realization was conceived for piano or vibraphone and prerecorded piano/vibraphone or disklavier with a possible accompaniment of any other pitched instruments. However, most of the following information can and should apply to any realization.

Dynamics

*The dynamic for each tone within a swell should be based on a Gaussian function like so:



Durations

*All durations for percussion or piano should be as long as the natural decay of that tone, i.e. **pedal should be down throughout**. I shortened all the tones for sake of readability in my realization. For wind instruments or voice, each tone should be the length of one comfortable breath. And for strings, the tone should be the length of one down bow-stroke.

Other

*Each swell is bracketed by repeat markings with a corresponding number of repeats written above. This does not mean that one plays that many cycles before moving onto the next swell. On the contrary (see *Meta-Score*), this simply indicates the number of repetitions in the piece, i.e. the second swell starts ten measures into the piece and repeats 15 times, the third starts twenty measures into the piece and repeats 14 times, and so on. I suggest that the live performer play the entire piece linearly with no repeats against a recording or disklavier playback of all the swells that repeat one less time then written (thus omitting the last swell). Then, the tape part or disklavier would be triggered to start ten measures into the performance and the pianist will end with the disklavier/playback. I am currently working on providing a midi file and prerecorded version. Contact me for more information.

*In this realization, any other instrument can play any notated tone, so long as it is available. In other realizations, the same should apply. It is mandatory though that no tone is missing, i.e. they may be doubled but not omitted. Also, the entire realization may be transposed by any amount.

Tuning for this Realization

*Though it is not mandatory, it is preferred that this piece be played with the proper cent deviations. When tuning the piano, it is advised to bring the A down 59 cents, and have the player replace all Ab's in the score with A's. The tuning is as follows:
C+0, D+4, E-14, F#-49, G+2, A-59, Bb-31, B-12

The Meta-Score

First a series of frequency ratios in reference to a fundamental should be chosen. Each resulting pitch in the series should be higher than the previous one. In general, though not always necessary, the harmonic complexity in reference to the fundamental should increase and the successive interval sizes in the series should decrease as on moves up the series.

Based on the number of pitches in the series there will be an equal number of swells. The first swell will include only the first pitch in the series and repeat as many times as there are pitches. The second will contain the first and second pitch, start after one playing of the first swell, and repeat pitches-1 times. The third will contain the first, second, and third pitches, start after the second repeat of the first swell, and repeat pitches-2 times. This process is continued until the final swell that contains all the pitches and does not repeat at all. Even with the repeats, due to the overlaps, the resulting number of swells will always equal the number of pitches in the series.

A duration between ten and thirty seconds should be chosen as a common length for all the swells. To determine the location in time of a particular pitch in a swell the following process must be implemented.

1) Calculate the interval size, in cents (one hundredth of a tempered semitone), in reference to the fundamental.
2) Choose a random number, x, between 0 and 1.
3) Calculate f(x) for the Gaussian function:

$$f(x) = e^{-(6x-3)^2}$$

4) Multiply *f*(*x*) by the interval size of the last/highest pitch in the series.
5) If *f*(*x*) >= interval size in cents of that particular pitch, accept and multiple x by swell duration.
   Else, throw and try again.

This should work for all pitches except the last one in the series, which can only occur at the exact halfway point of the swell.

Included is the source code of the above process implemented in Java for the realization I generated. The program will print out the point in time of each pitch with swell lengths of twenty seconds. It already offsets the swells by the appropriate amount but does not take into account the repetitions. If one were to transcribe the information they would simply have to put repeat bars around every twenty-second segment.

Tone location in time may be quantized to the nearest 32nd note.

For duration and dynamic information see *Performance Instructions/Realization.*

A poem may be recited over this piece.

```
/*Gaussian Canon pitch generator. When transcribed, one will have to put
repeats around every 20 second section.
Mike Winter*/

package main;

import java.util.Arrays;

public class Gaussian_Canon {

    public static void main(String[] args) {

        /*Enter number of pitches in the series.*/
        int noPitches = 16;

        /*Calculate total number of pitches in the piece.*/
        int arraySize = 0;
        for (int i = 1; i <= noPitches; i++) {
            arraySize = arraySize + i;
        }

        /*Declare Arrays*/
        double[] xArray= new double[arraySize];
        double[] xArrayCompare= new double[arraySize];
        double[] yArray= new double[arraySize];
        int[] sorted= new int[arraySize];

        /*This is a bit backwards. It calculate the placement of a particular
        pitch for all the swells it occurs in.*/
        int currentNote = 0;
        for (int arrayIndex = 0; arrayIndex<16; arrayIndex++){
            for (int swell = arrayIndex; swell<16; swell++){

                /*If last pitch in series, then place in middle of final
                swell.*/
                if (arrayIndex == noPitches-1) {
                    int xLast = 10000+swell*20000;
                    xArray[currentNote] = xLast;
                    xArrayCompare[currentNote] = xLast;
                    /*this is the midi note of the last pitch*/
                    int yLast = 48+24;
                    yArray[currentNote] = yLast;
                }
                else {
                    for (int index = 0; index<1; index++){
                        /*Enter frequency ratios of series in order*/
                    int[] harmonic = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
                            12, 13, 14, 15, 16};
                    double x = 0;
                    double y = 0;
                    /*Choose random x value between 0 and 1.*/
                    x = Math.random();
                    /*Calculte interval size, in cents, in reference to
                    fundamental*/
                    y = 1200/(Math.log(2)/Math.log(10)) *
                        (Math.log(harmonic[arrayIndex])/Math.log(10));
                    /*Calculate midi note*/
                    int midiNote = (int) Math.round(y/100)+24;
                    /*Gaussian Function for test. The last number is the
                    interval size of the last pitch in the series in
                    reference to the fundamental*/
                    double funcUp = Math.exp(Math.pow(x*6-3,2)*-1)*4800;
                    /*Test generated pitch.*/
                    if (y<funcUp && y>=0){
                        /*place pitch in appropriate swell.*/
                        x = x*20000+swell*20000;
                        xArray[currentNote] = x;
                        xArrayCompare[currentNote] = x;
                        y = midiNote;
                        yArray[currentNote] = y;

                        currentNote++;
                    }
                    /*Throw if out of range.*/
                    else {
                        index--;
                    }
                }
            }
        }
    }

    /*Sort arrays by starting point of tone.*/
    Arrays.sort(xArray);
    for (int i = 0; i<xArray.length; i++){
        for (int j = 0; j<xArray.length; j++){
            if(xArray[i]==xArrayCompare[j]){
                sorted[i]=j;
            }
        }
    }
```

```
        /*Print out a sorted list of x values in milliseconds and y values in
        midi note numbers.*/
        for (int i = 0; i<xArray.length; i++){
            int x = (int) xArray[i];
            int y = (int) yArray[sorted[i]];
            System.out.println(x+" "+y);
        }
    }
}
```