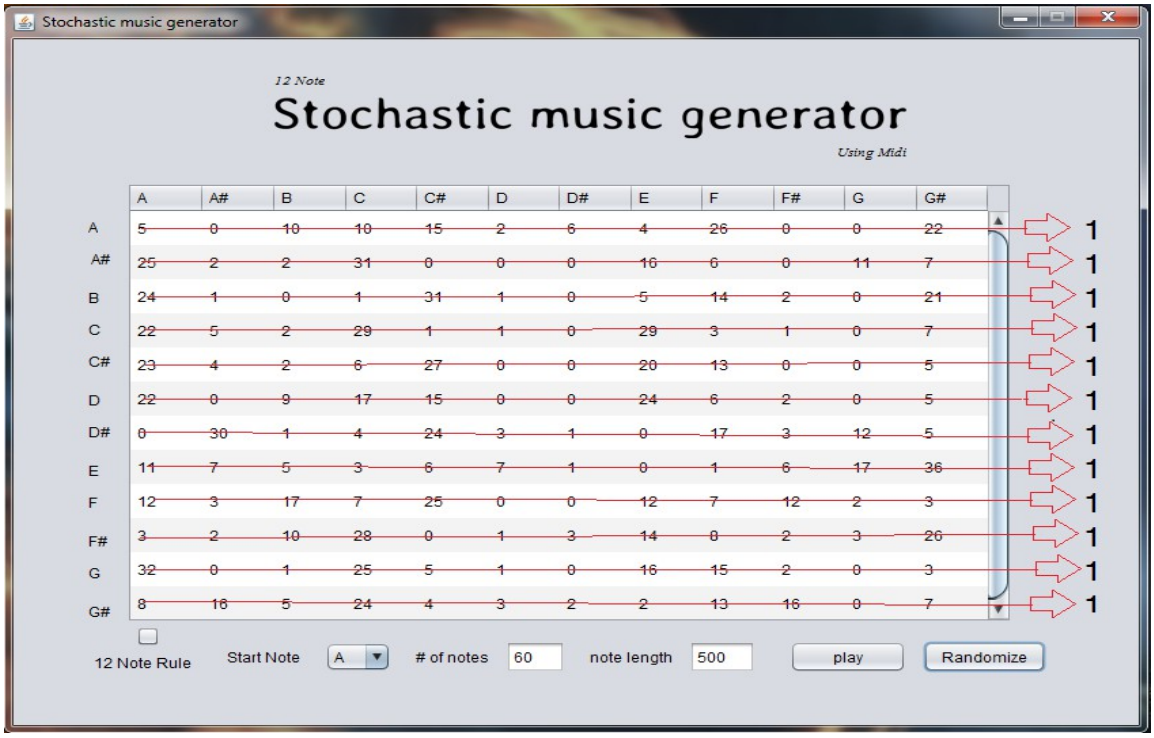
Twelve Note Stochastic Music generator using MIDI
By Andrew MacPhail

Background information

Based on the work of Iannis Xenakis, stochastic music generation is another way of saying probabilistic music generation. This is where a node leads to another node based on a given probability, and the overall result will always be non-deterministic (meaning there will always be that element of randomness). A node can be anything from a grain or midi note to even a sample.

To represent these nodes and their probabilities, either a Markov chain can be used, where it is structured like a linked list and each node would contain n variables for the probabilities, or a stochastic matrix can be used where the matrix would be of size nxn.



With a stochastic matrix each row adds up to 1, In my implementation my nodes are midi notes, and there are12 of them and I made each row add up to 100 for simplicity. In my implementation we see that if A is played the largest probability is to go to g# next with a 22% chance.

How to use it

To Start off the matrix is actually a jTabel and it must be populated to begin. You can either populate it manually, or use the randomize button I added. The table starts out empty and will not work unless all items are integers (So you can not leave anything blank). Another thing to point out is to make sure your rows add up to 100 because otherwise it will not work properly (it will still run and play notes, but probably not the correct notes).

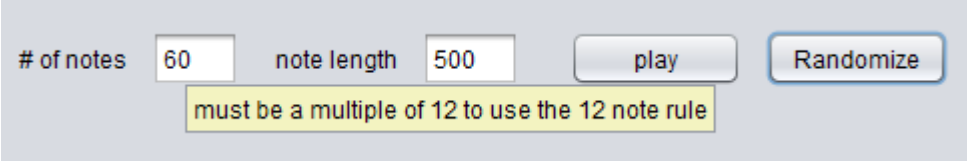
Once the matrix is populated all you have to do is hit the “play” button and you will be generating your composition. The fields “note length” is the length of each note in milliseconds, meaning the time between the MIDI note on and the MIDI note off. “# of notes” is straight forward, which means how many notes will be in your composition. Start note is the node you will start with, and in the above example we start with A.

How it selects the next Node

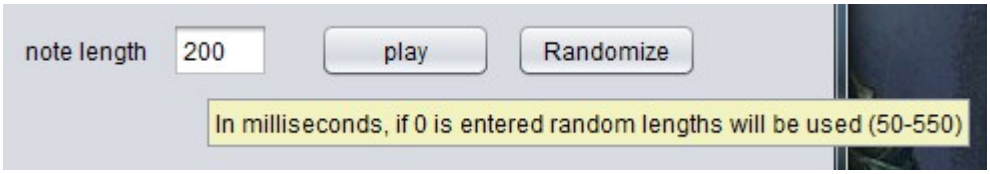
We have a set of 12 numbers that add up to 100, so to select one of these nodes, we use a random number between 1 and 100, which would then select one of the nodes. This is done by checking to see if our random number is less than or equal to the first number in the row, if it is not we move to the next node while accumulating the node percentages and check again, this is done until we reach a node we are less than or equal to. A node will always be found at some point because by the time we reach the final node in the row the accumulated value will be 100 and our random number is definitely less than or equal to 100.

Help

Tool tips have been added for each button and textfeilds, for example if I hover over the “# of notes” textfield we see

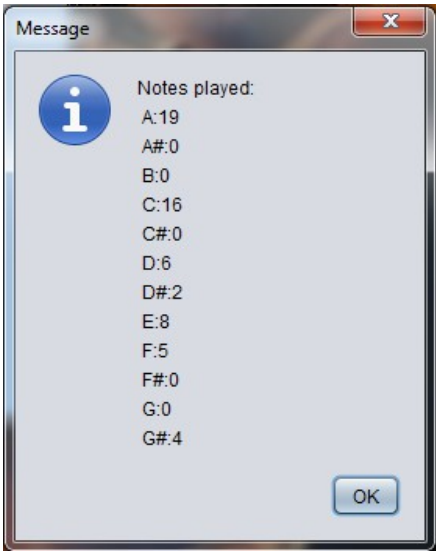


These tooltips will explain any requirements, and further explain what they do, the above tooltip is directed towards the 12 note rule which will be explained later.



This tooltip explains that the length of notes is to be inputted in milliseconds, and that an option for random lengths has been added. If you input 0 for the note length a random length will be generated for each note in the range of 50 milliseconds to 550 milliseconds.

Lastly After your composition has finished, a message box will appear and show you how many times each note has played.



You can use this to verify the probabilities you entered. For example if you entered A to go to A 100% of the time you will see just A played x amount of times assuming it was also your start note.

The twelve note rule

Iannis xenakis was known for adding rules into his composition to constantly change the stochastic matrix. Some examples of rules could be something as simple as re-randomizing the matrix every 4 notes, or change the values of the matrix after every note using a fractal. In his most well known work Metastasis, he used a rule called the 12 note rule which I thought was quite interesting.

The twelve note rule is when no emphasis is put on any note, meaning each note is played an equal amount of times. Now when we combine this with a stochastic matrix we get something that resembles Bayes theorem from AI (3p71) where probabilities change based on newly given information and input.

A quick explanation of the algorithm I used to do this:

For example say we have a 3x3 matrix for A, A# and B

75	0	25
33	33	34
33	33	34

At first glance without the 12 note rule, A has the most chance of being played.
With the 12 note rule each note has to played evenly meaning the number of notes has to be a multiple of 3. So if we have 3 each note is played once, if we have 6 each note is played twice and so on.

The process

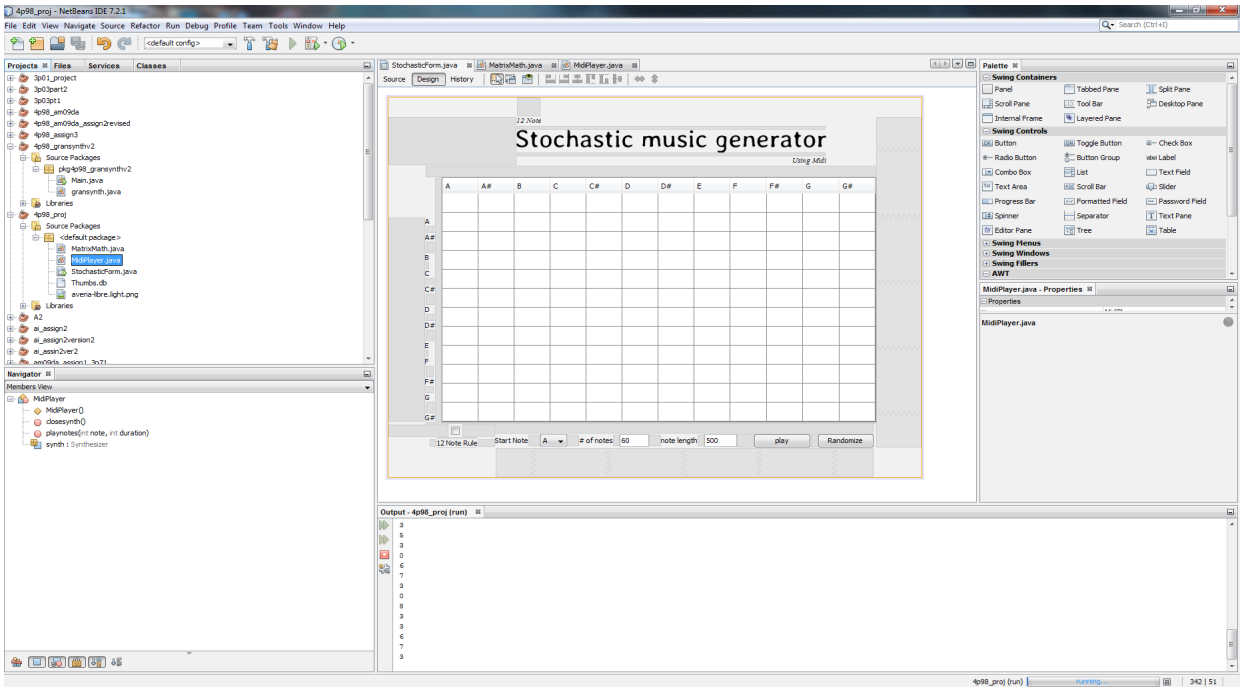
In this case say we want to play 9 notes, each note has to be played 3 times

The formula used is:
-probability of note just played (p) – (p * times played (tp) / times can be played (3))
so p – (p*tp/3)
-So for each row we now must reduce the probability of the note just played.
-say we started with A (we just played an A)
-A is the first column so for the A row we get
75 – (75 * 1/3) = 50 and with that 25% we subtracted it should now be evenly distributed to the other nodes in that row (assuming the other nodes have not been played 3 times) so A# is now 12.5 and B is 37.5
-the same is done with the next 2 rows (the 33% of row 2 and the 33% of row 3 would be reduced).
If A is played again we get
50 – (50* 2/3) = 16.666 and 33.333 would be distributed to the other nodes.
So A# is now 29.1666 and B is 54.16666
And if A gets played for a third time in a row we get
16.666 – (16.6666 * 3/3) = 0 with 16.6666 being distributed over the other 2 nodes of that row and A can no longer be played because its probability is 0 in all of the rows.

-We see by adding this rule the notes played will always be exact but the sequence still holds the non-determinist quality.

A bit on the design

The application was made in netbeans, and I used the designer, which would then generate the code of the jframe and all of its components. This is a lot like the design view in visual studio for RAD applications only instead of generating the code in a second class and combining the 2 classes netbeans just generates it in the single source file you are working on.



Executing the application

I compiled the application into a jar, so all you have to do is grab the jar file, put it in which ever directory you wish to execute it from and double click it. If you run the jar file on Linux the command is java -jar 4p98_proj.

I also included the netbeans project file in case you want to do any further inspection, add new features or change anything. For a quick note on changing the design; if you want to change the design code manually and not in the designer you must open the class in another program like notepad++ to edit the generated code because netbeans will not let you.

Side notes

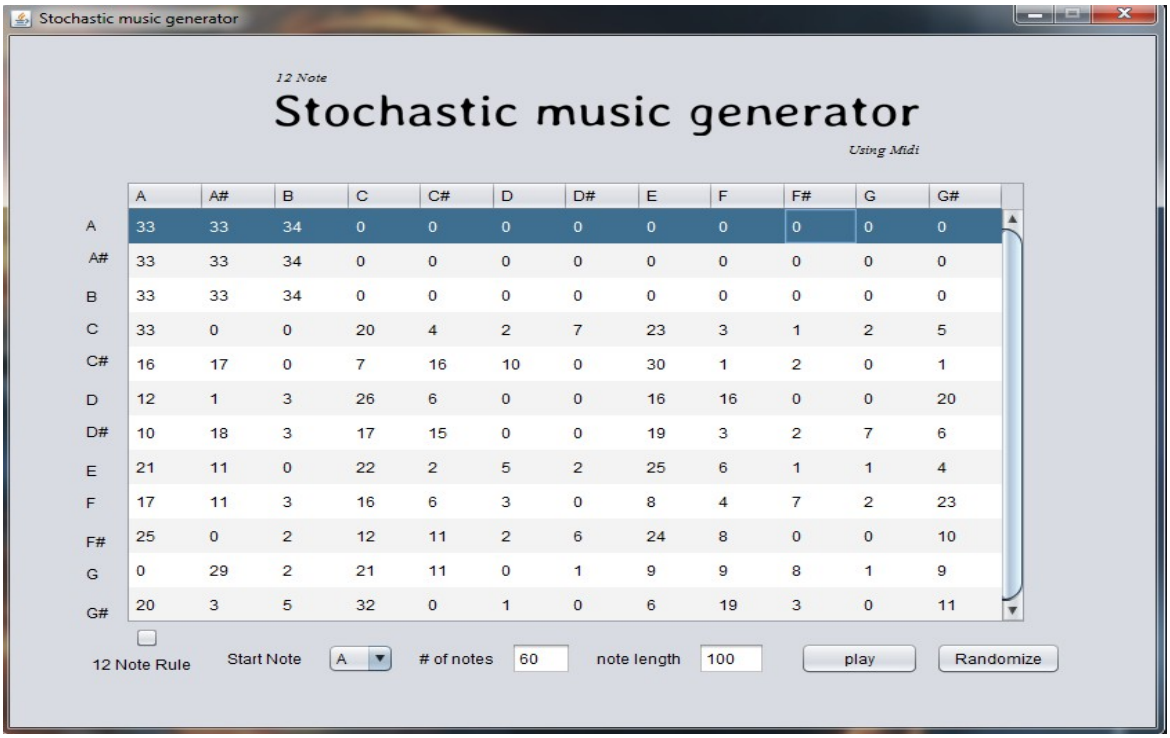
If you wish to see the sequence of the notes played I did a system.out.println which you can see in the I/O window in netbeans in the above screenshot.

Another thing Iannis Xenakis did was also allow the length of notes to be stochastic, but unfortunately I did not have the room on the jframe to include this. I was thinking of using a fractal for this called the midpoint selection algorithm for determining the note lengths but I did not have the time.

Something to point out is that online you may see that some stochastic matrix's columns add up to 1 instead of the rows, I believe this is just a matter of opinion, and so I went with rows.

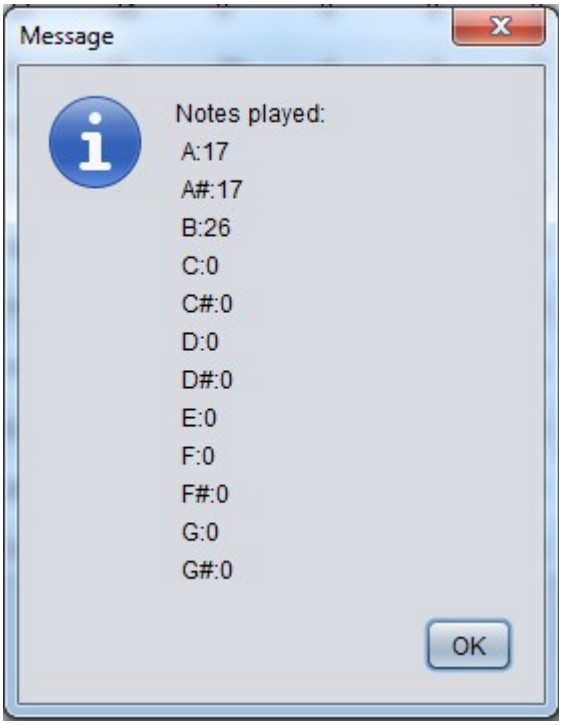
Final note – a quick example of adding your own values to the stochastic matrix

Say I want to cycle between 3 notes I would input something like



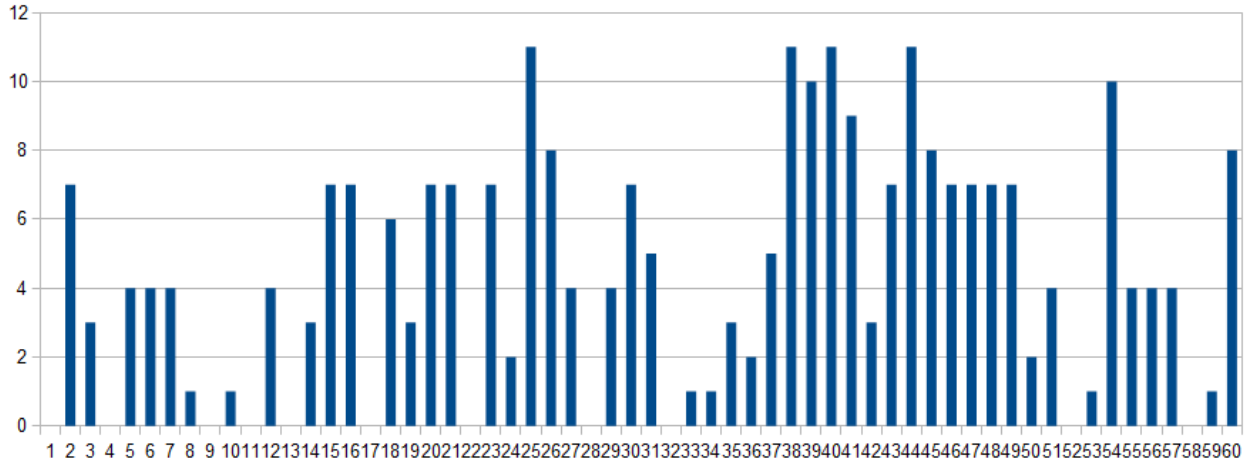
	A	A#	B	C	C#	D	D#	E	F	F#	G	G#
A	33	33	34	0	0	0	0	0	0	0	0	0
A#	33	33	34	0	0	0	0	0	0	0	0	0
B	33	33	34	0	0	0	0	0	0	0	0	0
C	33	0	0	20	4	2	7	23	3	1	2	5
C#	16	17	0	7	16	10	0	30	1	2	0	1
D	12	1	3	26	6	0	0	16	16	0	0	20
D#	10	18	3	17	15	0	0	19	3	2	7	6
E	21	11	0	22	2	5	2	25	6	1	1	4
F	17	11	3	16	6	3	0	8	4	7	2	23
F#	25	0	2	12	11	2	6	24	8	0	0	10
G	0	29	2	21	11	0	1	9	9	8	1	9
G#	20	3	5	32	0	1	0	6	19	3	0	11

And when the composition is finished I can verify it worked by checking the notes played.

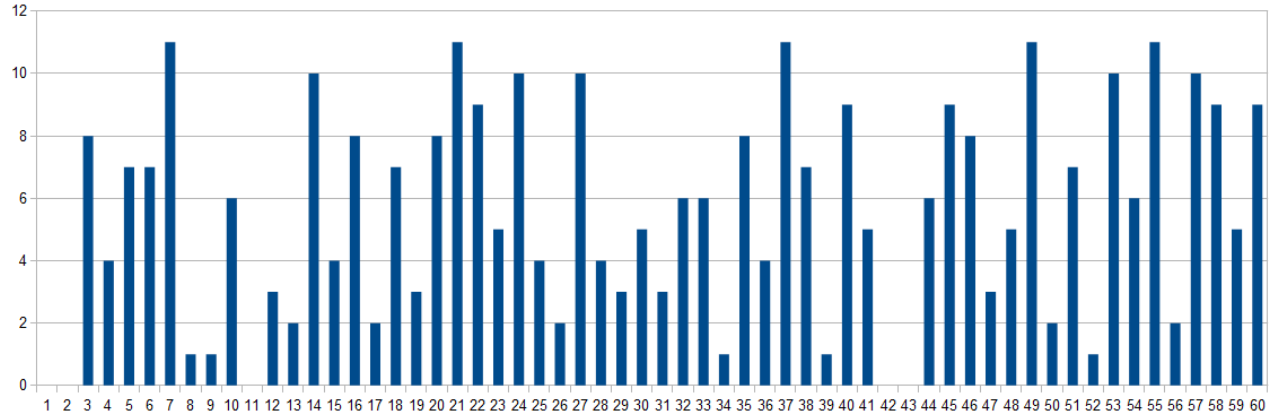


Diagrams

To show the difference of notes played between 12 note and non 12 note I ran both with 60 note lengths and plotted them.



This is the non 12 note



This is the 12 note

In the non 12 note you can see that there is a lot more repetition, and looks less evenly distributed. It is also good to note that the 0's in the graphs are also a note (the A note).

Bibliography (same as my presentation)

Formalized music – Iannis Xenakis

http://music.columbia.edu/cmc/courses/g6610/fall2012/week4/Gendy3.pdf

http://www.theguardian.com/music/tomserviceblog/2013/apr/23/contemporary-music-guide-xenakis

http://earsketch.gatech.edu/category/learning/randomness/introduction-to-randomness-chance-and-stochastic-composition

http://books.google.ca/books?id=y6lL3I0vmMwC&printsec=frontcover&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=true