Kellie Banzon, Alex Gravenor, Jeffery Ho, Steven Pineda Team "I feel like a tater tot"
CPE 123-03

Final Project Write-Up

Description

(Briefly describe the idea behind your product, the look of the program, and its basic controls. Essentially, "here's what the program will look like when it's finished.")

A Markov chain produces a sequence based on probabilities. Our project intends to create audio through the analysis of Markov chains. We first analyze existing music files to generate the probability of each note occurring, then we feed those probabilities into our program to synthesize music. Initially, our computer-generated music will sound extremely similar to our sample music file; however, as we analyze more and more sample files, our synthesized audio will sound more unique. Ultimately, we hope that the music produced randomly by our program will be indistinguishable to the human ear from music composed by hand by professional composers.

Once the program launches, the Markov audio generation automatically begins. The notes (represented as nodes) are circles labeled with the appropriate note name, arranged in a circle around the center of the window. The program launches with six nodes, but users can add more by clicking the "+" button or pressing the "." key. Similarly, they can remove a selected node by clicking the "-" button or pressing the "," key. There is no limit on the amount of nodes that can be added (other than the amount of computer memory available).

The original six nodes are pre-programmed with probabilities based on probabilities calculated from imported audio files. We wrote both a Java and a Racket program that are functionally identical: they accept MIDI files as parameters and analyze them to generate an initial Markov chain that mirrors the MIDI file. These programs work by extracting sequencing information from the MIDI files and using a hash map to get total frequencies of one note following another. The programs then sort the nodes by pitch and present them as a normalized lists of connections from one node to the next that the core program can use.

When users add a new node, the connection strengths of this node are randomly generated. (In other words, these new nodes do not have probabilities based on our MIDI analysis.) The connection strengths of all the pre-existing nodes adjust to accommodate the new value by reducing their other connection strengths. No probability can ever fall below 0%, and when one probability changes (whether due to the addition of another node or the user altering the connection strengths with the arrow keys), the probabilities of all the other connections adjust appropriately (so that the total probability still sums to 100%).

Each node is connected to every other node and itself by a line that represents the probability of that particular connection. The weight of the line gets heavier as the probability increases and lighter as the probability decreases.

A connections box on the left also lists the connection strengths of a selected node. The connections box only displays the connections from one node at a time, so users can click on another node to change which connections are delineated. Users can press the up and down arrow keys to scroll between these probabilities. The currently selected node and the currently selected connection will light up red on the map of nodes. Pressing the right arrow key increases the probability of the connection selected and pressing the left arrow key decreases the probability of the connection selected.

In the top right corner, a slider can either be clicked or dragged to adjust the volume of the audio. Below it, a mute button can be clicked to mute or unmute the audio. Pressing "M" achieves this same mute functionality. If the volume is muted, the mute button appears as red; otherwise, it appears as gray. Clicking the pause button or pressing "P" pauses the audio.

Users can also load pre-programmed Markov chains based on known songs. "A" loads "Frosty the Snowman," "S" loads Jingle Bells, "D" loads "Misirlou," and "F" loads the "Ghostbusters" theme song. Buttons along the right side also trigger these same chains. Ultimately, more songs will be added. (Note: because these songs are all fairly complex, this program loads and plays simplified versions of the most commonly occurring notes for pragmatic reasons.) To reset to the original, pre-loaded Markov chain, users can click the button on the right or press "R."

Along this same right panel, users can select from a series of background tracks to play under the Markov chain melody. (After these background tracks have been selected, corresponding keyboard shortcuts will also be implemented.)

On launch, the program automatically displays a help / about screen. After reading these instructions and background information, users can click anywhere to advance to the program itself. While they are using the program, if they wish to see the help screen again, they can click the "?" button in the bottom right or press "H".

Potentially, a Markov chain could also be developed that assigns note lengths in addition to a MIDI note value. If this chain is generated using a piece of music, the rhythm will tend to imitate that original piece of music. This Markov chain doesn't have nodes that correspond to MIDI notes, but rather note lengths. So a rhythmic Markov chain might have whole note, half notes, quarter, and eighth notes as the values of each node. These are connected in the same way as the original Markov chain and both work simultaneously to create a simulacrum of the original music.

Prototype

(Describe what features and rules you have implemented in your prototype. Essentially, "here are the parts of the program that we're still working on.")

The GUI of our program has yet to be fleshed out. Ultimately, the whole program will be more aesthetically appealing. The background and nodes will have "textured" backgrounds, the buttons will have the appearance of depth, and the help menu will be stylized to be consistent with the look of the rest of the program. In place of the minimal instructions the help menu currently contains, there will be a more thorough background about the program's conception and purpose as well as more in-depth instructions about how to use the program. Also, the node currently delineating its probabilities in the connections box on the left will also light up a different color on the Markov map on the right.

As to functional aspects of the program, the polyphony has yet to be implemented. We would like to select some background tracks to run under the melody of the Markov chain. Users will be able to select one of the background tracks or turn them all off using buttons on the right or keyboard shortcuts. We also plan to add buttons and keyboard shortcuts to offer the ability to pause the Markov chain audio, remove a selected node, and select from more preset Markov chains based on well-known songs.

If time allows, our program will also implement a Markov modeling rhythm. The code would run a similar process as it did with the note generation to create rhythms by varying note length. This new Markov rhythm chain would run simultaneously with the Markov melody chain to produce varying notes of varying lengths.

Testing

(Describe the process the users went through in testing your program. How did you find your testers? How much time did you spend explaining how it worked? How long did they use it? How did you gather information from them? What questions did you ask them?)

For testing, our group first designed a Google Forms survey with questions about our program. Primarily, we wanted to find out how difficult our program was to use, how it looked, what users liked and disliked, what parts users had trouble with, and what improvements could be made.

For our testers, we targeted people outside our class because we wanted to know if people without prior knowledge of the DrRacket shell and Racket language could understand our program. We got 10 responses from roommates and friends of varying technical backgrounds, and one collective response from an entire group in our CPE section. When running the program, we did not explain anything to the testers. Our program initially launches a directions screen that outlines the purpose of our program and the features. Using this, testers had to figure out everything by themselves. We observed how they used the program and noted any confusion they had. We only interfered to ensure they explored all aspects of the program. Testing generally took 5-10 minutes per person. After the testers finished, we asked them to fill out the survey.

On our survey, questions included:

- On a scale of 1-5, how difficult was it to use the program? (1: Easy - 5: Hard)
- On a scale of 1-5, how did the program look? (1: Horrendous - 5: Exceptional)
- Did the program achieve what you expected?
- What did you like about the program?
- What did you dislike about the program? What areas do you think need improvement?
- What questions did you have while you were using the program?
- What questions do you have after using the program?

Analysis

(Analyze the data. Start by summarizing the measurements you made. Then: do you see clear common elements in your responses? Are there any patterns that you see?)

The distribution of responses for the difficulty of our program was pretty symmetric with most selecting 3 (the middle). Ideally, we want most responses in the 1-2 range. Those who had trouble were overwhelmed by the directions since it includes so much information. They eventually figured out all the controls, but it took some time and experimentation. To decrease the difficulty of our program, we can edit the direction screen to be more concise and only explain the most important controls.

On the other hand, many testers were a little confused with what the program did at first. Although most figured the instructions out later, they said they were a little too hard to understand and provided too many things all at once.

When asked how the program looked, many testers (about 65%) rated it a 4. We are pleasantly surprised by this response because people liked the program’s presentation even though we are not finished with the GUI. We have more elements to add and plan to keep improving the organization.

Testers really enjoyed how they could manipulate the probabilities between the nodes. They said they had fun playing around with the music. All of them also liked the different colors and how the program looked when you added more nodes. They appreciated the numerous options they could control throughout the program.

The tempo is generated randomly, so sometimes our program produces faster rhythms than at other times. When notes were played in quick succession, users had more trouble figuring out what was happening. While using the program, many testers also didn’t understand what node was selected and what connections were being shown. They later realized that the connections box shows the percentages for one specific node.

For improvement, they recommended highlighting the currently selected node (the one displayed in the connections box) to make it more clear which node they were looking at and what the percentages mean. They also wanted to opportunity to pause the program so they could figure out things at a slower pace. Many also looked for a functionality that allowed them to remove certain nodes. Users liked the idea of having song options, but wanted more they could choose from. Many also expressed confusion that the songs generated by our program did not sound like the recognizable songs they were familiar with. A few also wished there was a constant background track to give the sound more depth.

Refinement

(Given this feedback, what should you change about your program? You should include the prioritized list of improvements discussed above.)

There are a few aspects of our program that we have yet to implement: we still need to design the GUI and the background tracks. Based on the feedback from our beta testing, though, we now also plan to highlight the currently selected node, flesh out the help screen with background “about this program” info and clearer instructions, add more song options, create a pausing ability, and enable users to remove selected nodes.

Below is our list of planned improvements. They are sorted by a combined expectation of difficulty and importance.

Improvements (Prioritized):

- Highlight currently-selected node
- About screen / finish the help menu
- GUI
- Polyphony (background tracks)
- More songs
- Pausing
- Remove a node?
- Markov modeling rhythm?