Project 5: Report

Eu Jing Chua, eujingc

Criteria

- Creating Serpent data-structures for representing transition matrices.
- Implementing of methods for combining Markov transition matrices to represent fusion.
- Generating sequences of notes from the new Markov chain, and applying note constraints to the generation.
- Using generated notes for playback via MIDI by incorporating timing and sound envelope information, derived from user input.
- Testing the generation and MIDI playback.
- Creating user manual to detail the controls of the program.

What I did

- TransitionMatrix class in transition_matrix.srp for:
    - Matrix data structure that is a 2D list, supporting a state space that is (pitch class x duration). Each row represents a probability distribution over all states given we are in the current state.
    - Since MIDI durations are arbitrary and not always in strict powers of 2 i.e. whole note, half, quarter etc. I round every fractional duration to the nearest power of 2. For simplicity, I also ignore rests.
    - The class stores the duration information separately in a list so we can sparsely represent durations instead of enumerating all possible combinations of the state space.
    - Easy serialization and deserialization using CSV. Since the state space is sparsely represented, the relevant states are parsed from the header row in the CSV.
    - Weighted, linear combination of two transition matrices. Since state spaces are sparsely represented, the function creates a new state space using a union of the two being combined and combines transition probabilities in a weighted manner.

- Generating transition matrices:
    - Reading Standard MIDI Files into Allegro data structures in generate/generate_tm.srp.
    - User selects which track to use and turn into a TransitionMatrix.

    - User then selects where to store the resulting CSV file that contains the serialized TransitionMatrix.

- play_a_measure():
    - Use a TransitionMatrix object to generate a measure of music as a first order Markov chain, where the next pitch and note duration depends on the current ones.
    - Constraints selected pitches to be within the current scale.
    - Instrument is set with a MIDI program change at the start of the measure only so the whole measure will be in the same instrument.
    - Selected note durations are multiplied by a dragging factor from 1.0 to 2.0, selected via TouchOSC. This allows us to control how much notes overlapped, as without overlaps note durations sounded very artificial and exact.
    - Note velocities are modeled with a normal distribution with mean 100 and standard deviation chosen from TouchOSC, ranging from 15 to 30. Downbeats however, are set to exactly 100 for consistency. This created some variation to break up the monotonous volume.

Assessment

I felt that given our very limited knowledge of music theory, we managed to come up with a generator that mixed well with the others in the laptop orchestra. I think I initially faced much difficulty putting the Markov chain concept into practice; the initial jazz MIDI files I used were full of irregular note durations, and I found it extremely difficult to identify the keys the songs were in. However, I managed to make some progress by using simpler MIDI files instead, like national anthems and game soundtracks, with the help of my partner Jacky. When I managed to come up with the first version of the generator for the preview, we realized it did sound very boring and monotonous.

After working on some dynamics to the sound, such as dragging out note durations abit to create small overlaps and introducing variation in the velocity, it sounded much more natural. We also experimented with different instrument sounds to find one that was a good fit for the Markov chains that we were generating, and settled on more airy sounding vocal “Doos” instead.

The concept of mixing transition matrices with a weight must have worked to some extent, as we could audibly hear a corresponding spectrum of styles as we increased and decreased the mixing weight. However, despite the occasional resemblance, the first order Markov chain simply could not generate a convincing variation of the samples it was trained on

Project 5: Report

Eu Jing Chua, eujingc

as it is a relatively simple model. One big limitation was that generated pitches were all in the same octave, a limitation of our state representation. Although we could have adopted a more sparse state representation in which we modelled the exact pitch number instead of pitch classes, the sparsity of the state space would make it extremely difficult to generate measures in arbitrary keys.

Given more time, we could experiment with training higher order Markov chains with more data to balance out the sparsity problem. However, in the end maybe Markov chains as is just are not able to characterize music well with the strong assumptions of the Markov property - there is much more to music than just what was played beforehand.