Project 5 Design Document

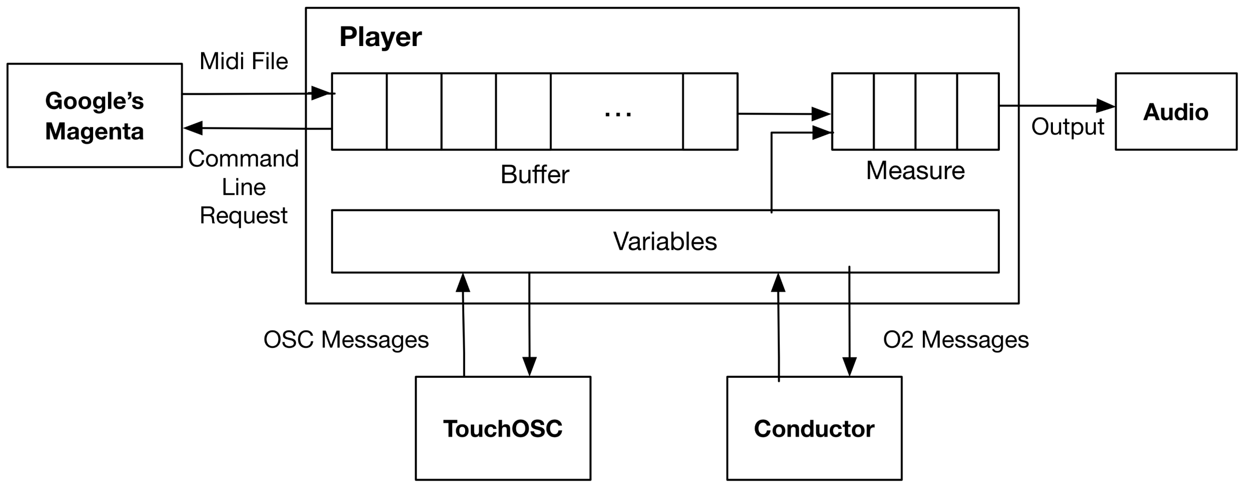
Group Members

Mengjin Yan (myan1) and Yinan Wang (yinanw1)

Summery

We performed melody on a certain instrument (which is configurable). The RNN model from Google's Magenta is used for music learning and generation. We created a music generation system that integrates the Conductor's pattern requirements, TouchOSC parameter configurations and the RNN algorithm to play music in real-time.

System Design Detail



Above is our system module graph. There are three main components in our system. First is **Parameter Modification** with communication with TouchOSC as well as the Conductor. The second is **Notes Generation** with communication with Google's music generation tool Magenta using RNN algorithm. The third is **Note Processing** and scheduling using hierarchical buffers. Below is detailed description of the three components. The parameters given by the conductor are fitted in real-time before dispatch in the **Melody Process and Fit** module, which will be discussed below.

Parameters Modification

This component is the central control of the system. Multiple different system parameters are maintained in our player which can be modified through the control messages between TouchOSC as well as the Conductor.

Our player communicates with TouchOSC through OSC messages. The parameters to be controlled include volume, types of instrument, etc. Multiple handlers also need to be implemented to for all different kinds of messages. Our player communicates with the Conductor through O2 messages. All the messages related to melody also need to be handled and the corresponding variables need to be modified.

Thus, this component achieves the interaction and real time control over our music generation system.

Notes Generation

Google's Magenta will be used for notes generation in this project. The Magenta algorithm learns from Google's NSynth dataset, which contains thousands of music clips on different instruments. The learning and generation tasks are done using a RNN algorithm. In this project, we will pre-train the network and use it to run the generation algorithm. Since hacking the algorithm itself requires a huge amount of work, we will just use the algorithm as it is.

The RNN music generator outputs a music clip that starts with a primer clip of music (which is the input of the algorithm). The algorithm will generate a music clip with the style of the primer clip. We will embed the chord and genre information in the primer clip to make the output follows the style information from the Conductor. Since the generation process is offline, we need to generate the music clip by clip in order to make the system run in real-time. To preserve the coherency of clips as a whole, we will use the last several notes of the previous clip as the primer clip for the clip to be generated.

Since the Magenta is written in Python, the system and Magenta communicate with system shell and file system as the figure shows. When a clip of music is required to be generated (there is less than 120 notes in the buffer), the system send the request as a system command to run the generator. When finished, Magenta outputs the result as a midi file in local filesystem. The system will periodically (every 0.5 second) check the filesystem to see if the generator finished. If so, we will roughly parse the notes from the clip and push them in a buffer queue.

Notes Scheduling

A set of hierarchical buffers are maintained in the player. The generated notes pitch as well as its duration and genre information are put into a buffer which maintained in our player. At the head of the buffer, a measure of notes will be pulled into the measure buffer using the time signature information from the Conductor's message.

Since there might be cases that the genre of the music changes during the generation of the buffer, the measure in the buffer might not be consistent with the genre during output. In this case, several rhythm patterns are pre-stored in the player and the measure will be fit into one of the corresponding rhythm patterns before output.

The final processed measure will then be scheduled to output at once. In each output function, the corresponding notes will send a midi not_on message with the corresponding volume and pitch.

Melody Process and Fit

For the processing in general, we extract exactly one measure of the notes from the Magenta output queue and fit the notes as well as rhythm according to the configuration (bpm, scale, chord) provided by the conductor as well as TouchOSC (volume, range, instrument). Then the whole measure is scheduled and played. The processed measure is also fed into Magenta as the prior melody to ensure the coherence of the music.

Specifically, for fitting the rhythm, two methods are tried. The first is the have some rhythm patterns predefined in the source code and fit the pitches according to a certain rhythm pattern when extract the measure. Another method is just feed certain rhythm into Magenta as the prior melody the first time the generator is called and just use whatever rhythm Magenta generated when output. In practice, we found the first method can produce more pleasant music and thus the first method is chosen.

For fitting the scale, we check each note in the measure. If the measure is not in the scale pitch class, we add one to the pitch and check again. The pitch will be continued incremented by one until the pitch is in the scale and the new pitch is stored back to the measure.

For fitting the chord, a variable indicating the possibility of a note in the chord pitch class is created and set to 0.5 initially. We also check each note in the measure. If the note is in the chord pitch class, no change is made to the pitch and possibility variable is set to 0.1. Otherwise, we first decide if the note should be in the pitch class under the possibility indicated by the variable. If the note doesn't need to be in the pitch class, the original pitch is preserved and the possibility variable increments by 0.1. If not, the pitch will be fit into the chord pitch class using the same method in fitting scale and the possibility variable will change to 0.1.

Configuration provided by TouchOSC is the volume, the range of the notes as well as instruments. We support 12 types of instruments in total which is 6 melody instruments and 6 percussion instruments respectively. For each instrument, the highest and the lowest pitches are found. When fitting the pitches, we also make sure that all the pitches are in the range for the instrument by adding or subtracting 12 on the pitches.

There is also a slider controlling the pitch ranges for the output. As it selects the left one third portion, the output will be fit to the highest two octaves of the instrument. As it selects the middle one third portion, the output will be fit to from one octave higher from the lowest pitch to one octave lower from the highest pitch of the instrument. As it selects the right one third portion, the output will be fit to the lowest two octaves of the instrument.