LUNCZ: PLAYING DATA

Final Project by Dror Ayalon
Course: Software Synthesis
Lecturer: Jean-Luc Cohen
December 22nd, 2016

For full Csound and Python code, see the project's GitHub repository.

OVERVIEW: MUSIC AS A TOOL

"Luncz: Playing Data" was my first attempt to generate audio related data from an analog recording, and to use this data to play new music.
The purpose of this project was to build a tool (or an installation) that will create an endless conversation between a musician and a machine. The software allows musicians to record short snippets of music, and uses these recordings to change the background music in the room. The recorded music is being analyzed, and using the result of the audio analysis, beats and changes in notes (frequencies) are being introduced on the background music.
The background music was not designed for listening. Instead, the background music was designed to support the musician, and to leave room for further playing and recording (more about that on the 'SOUND DESIGN' section).

HOW TO PLAY: INSTRUCTIONS

Luncz is being run by two Python scripts:

1. 'input.py' - Responsible for recoding live music using PyAudio, analyzing the recording using librosa, and dumping a WAV file with the recorded audio, a TXT file with the estimated tempo of the recording, and a CSV file with the beats onset data.

2. ‘output.py’ - Responsible for initiating Csound (‘Luncz.csd’), collecting the files, reading the data, and sending score events to the Csound engine using Csound’s the Python API and ctcsound, a Python wrapper for the C Csound API.

The following steps should be taken in order to experience Luncz:

1. If you don’t have Homebrew installed, please open terminal and run:
   ```
   $ /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
   ```

2. If you don’t have portaudio installed, please run the following command on your terminal:
   ```
   $ brew install portaudio
   ```

3. While on the project main directory, using virtualenv, create a virtual environment for the project:
   ```
   $ virtualenv env
   ```

4. While on the project main directory, activate your virtual environment by running :
   ```
   $ source env/bin/activate
   ```

5. While on the project main directory, install Python dependencies - Using virtualenv, install the dependencies specified in ‘requirements.txt’. Run:
   ```
   $ pip install -r requirements.txt
   ```

6. Add ‘ctcsound’ to you site packages:
   1. Copy ‘ctcsound.py’ and ‘csoundSession.py’ files from the ‘ctcsound’ folder.
   2. Paste the files on ‘<project_main_folder>/env/lib/python2.7/site-packages

7. While on the project main directory, run ‘output.py’ to start the background music:
   ```
   $ python output.py
   ```

8. Open a new terminal window, and go to the project main folder. While on the project main directory, run ‘input.py’ to record analog audio signal:

   ```
   $ python input.py
   ```

9. Hear the background music changes based on the analysis of the recorded audio.

10. Repeat step 3 as many times as you like.

TROUBLESHOOTING

- In case where you’re getting the following error:

  “RuntimeError: Python is not installed as a framework. The Mac OS X backend will not be able to function correctly if Python is not installed as a framework. See the Python documentation for more information on installing Python as a framework on Mac OS X. Please either reinstall Python as a framework, or try one of the other backends. If you are Working with Matplotlib in a virtual enviroment see 'Working with Matplotlib in Virtual environments' in the Matplotlib FAQ”

  Do the following actions:

  In you terminal, type:

  ```
  $ nano matplotlibrc
  ```

  Add the following line to the matplotlibrc file:

  ```
  $ backend: TkAgg
  ```

TOOLS AND WORK PROCESS

AUDIO ANALYSIS AND FINDING THE RELEVANT DATA

- PyAudio (Python package) – Used to capture live audio recording using the computer audio input source.
- librosa (Python package) – Used to analyze the recorded sound. After a long research on librosa, I was able to analyze and unlock the estimated tempo or the recording, and the most dominant notes in the recording. This data is being used to generate score events in Csound. Beats are being added to the music. The beats

are being played in the tempo of the recorded music, and their frequency is based on the dominant notes of the recorded music.

SENDING THE DATA TO CSOUND

A vast variety of ways and tools, including Csound's OCS and 'csnd6' (Csound's Python package, which was deprecated on Csound v6.08) were tested, in order to find the best way to initiate Csound, and add new score events while the Csound is up and running.
Eventually, the communication between with Csound was done using the 'ctcsound' Python package.

SOUND DESIGN

INITIAL BACKGROUND MUSIC

The initial music (before any recording was done) was designed to emphasize 'the sound of a room'. The music should be present, but to provide a sense of emptiness, that will potentially invite users to add (play) their own music.

BEATS

The sound of the beats, which are being heard after a successful recording has been done, was designed to be used as a metronome, on one hand, and to allow the musician to continue and improvise around his/her musical idea, on the other hand.
In a way, the beats sound like an electronic base player who gives a reminder about the tempo once in a while, and plays notes according based on the original recording.

MODIFIED BACKGROUND MUSIC

Once a score event is being sent to Csound, the background music changes its timbre to allow more space for live music playing, and its notes, based on the original recording.

ENDLESS COMPOSITION

To ensure that the music played by Csound will be played continuously, I used if statements and the schedule opcode to generate new score line based on the lines that were played.

FURTHER DEVELOPMENT

POSSIBLE ADDITIONS

- Timbre - Retrieving data about the timbre of the recorded piece will allow me to change the timbre of the background synthesized music.
- Physical controller - Pressing a physical controller (pedal) Will allow the musician to focus on his/her playing, and will free him/her up from clicking to computer.

THE BIG PICTURE

This project is the first experiment within a series of experiments. The vision for these experiments is to setup a tool that will allow musicians to record musical ideas, instead of 3-5 minutes songs. In order to that, a future system will use the data retrieved from a recorded piece of music to help the musician develop the a musical idea. The outcome could be an endless piece of music, generated from a series of recordings, and a set of configurations, that could be dynamic and change according to different real-world and virtual events.
More about this vision can be found in this blog post -
http://www.itp.drorayalon.com/2016/12/15/the-musicsystem-explained/

Thank you for giving the opportunity to participate in this class, and the inspiration to take on new challenges that seemed to be beyond the horizon just a few months ago.

— Dror Ayalon