CMSC 12200
Winter 2016

PROJECT PROPOSAL

Team Name

WIRE - Web Interfaced Reverb Engine

Team Members

Jonathan Kyl (jkyl@uchicago.edu)
Mayukh Sen (mayukhsen@uchicago.edu)
Esteban Zacharzewski (ezachar@uchicago.edu)

Goal

To create a Web interfaced sound engine that accepts user audio input, transforms the audio file based on user specified parameters, visualizes such transformations and generates an output audio file.

Project description

The purpose of this project is to create a sound engine that can be accessed and used through a Web based interface.

The user can provide his/her own audio file (in a range of specified audio formats) as input to the program or use a sound from within the program's sound bank. The user can specify the program with various available sound transformations to generate a modified audio file.

The sound engine itself (i.e. the backend) is based off different algorithms for signal processing such as FFTs (Fast Fourier Transforms), signal convolution, ring modulation, etc. The primary transformation of the sound engine is to create a simulated reverb effect by convolving two sounds together. One of the sounds can be given to the program as input, the other can be chosen from the program's sound bank. Alternatively, both sounds can be chosen from the program's sound bank.

The program's sound bank is composed of two sources of audio files. The first one is a massive collection of field recordings produced by *radio :: aporee* which is stored in *archive.org* under a Creative Commons License. The second one is a much smaller collection of recordings produced by this team across campus and Chicago and stored locally.

The Web interface will not only allow the user to specify input and output parameters but it will provide visualizations for different output stages of the sound engine such as waveform of the audio file, frequency spectrum, etc. These visualizations will be made using matplotlib.

We believe the end product will be a fun product which is open-source for anybody to use. Users may be attracted to its educational qualities --understanding sound transformations--, its artistic potential as a tool for songwriting, or simply because it is fun to play with.

<u>Sources of data</u>

- *radio :: aporee* collection of field recordings from around the globe.
- Recordings of impulse response from spaces within campus and Chicago.

<u>Project timeline</u>

1. Metadata scraping / Sound recording (Week 5, Week 6)
    a. Sound recording spaces within campus, Chicago (Week 5 weekend)
    b. Run metadata scraping scripts on *radio :: aporee* using *archive.org* API
    c. Process metadata into SQL database
    d. Develop Python backend to query database

2. Sound engine framework and algorithms (Week 4, Week 7)
    a. Develop framework design
    b. Develop algorithm for a range of sound transformations
    c. Develop sound visualization scripts

3. Django framework (Week 8, Week 9)
    a. Develop the Django Web interface
    b. Pipe interface to sound engine framework

4. Ironing out (Week 10)
    a. Test program within CS 122 VM.
    b. Make sure all framework pipes work properly.
    c. Final additions to documentation.
    d. Final version of README.md

<u>New data structure, algorithm, programming technology</u>

- Django:
    - The Web interface will be based on Django. Users will be able to input audio files, specify parameters, see visualizations, play output audio and save output to disk all within the browser interface.
- archive.org API:
    - Scrape metadata of *aporee :: radio* audio collection using the *archive.org* API. Store metadata in a SQL database. Access audio file download from API.
- Signal processing algorithms using scipy and working with audio files using pydub:

  - Our code will make heavy use of signal processing modules within the scipy library.
  - In-program representation of audio files will be accomplished using pydub and various file format transformations make use of codec libraries.
