**Computer Science and Engineering, University of Nevada, Reno**

**MelodyBot**

**Team #19**

**Nicholas Harris**

**David Neilsen**

**Joseph San Nicolas**

**Stefan Stukelja**

**Instructors: Sergiu Dascalu & Devrin Lee**

**External advisor(s): Benjamin Brown**

**David Feil-Seifer**

**May 7th, 2018**

Project Summary

Functionality Implemented

Use Cases:

* use case 1: a user will be able to import files.
* use case 2: a user will be able to remove files.
* use case 3: a user will be able to play music.
* use case 4: a user will be able to pause music.
* use case 5: a user will be able to resume music.
* use case 6: a user will be able to change the volume of the music.
* use case 7: a user will be able to access the project website.
* use case 8: a user will be able to select a genre.
* use case 9: a user will be able to select a tempo.
* use case 10: a user will be able to select an output instrumentation.
* use case 11: a user will be able to select a seed.
* use case 12: a user will be able to select a length for the sample.
* use case 13: a user will be able to generate samples.
* use case 14: a user will be able to download samples.
* use case 15: a user will be able to exit the application.

Functional Requirements:

* FR\_01: MelodyBot shall allow users to select a type of music genre.
* FR\_02: MelodyBot shall have a functioning audio player that can play, pause, resume, fast-forward, and rewind music.
* FR\_03: MelodyBot shall allow the user to download the generated music sample created by it.
* FR\_04: MelodyBot shall allow the user to change the audio settings, i.e. volume, of the music it is playing.
* FR\_05: MelodyBot shall allow the user to change the music settings, i.e. tempo, of the music that it is playing/generating.
* FR\_06: MelodyBot shall allow the user to generate a new random “seed”, upload a new seed, and save a seed.
* FR\_07: MelodyBot should include information about its use for the user to read.

Functionality Not Implemented

Non-Functional Requirements:

* (Level 1) User manual to be accessed through the “help” tab. This would be a simple guide, possibly with FAQs, to assist any users in learning how to operate the software. It has not been implemented until the software is fully completed to ensure it has all the functionality accurately and completely covered. This was not completed as the functional requirements took priority over non-functional requirements.

Contribution of Team Members:

* **Nicholas Harris**
  + Responsible for all neural network code
  + Oversaw all neural network training
  + Scripts for converting midi data to text characters and back
  + Acquired all data for machine learning
  + Approx. 45 hours total work
* **David Neilsen**
  + UI Design and aesthetics
  + Implemented main UI window and settings window
  + 100 Lines of code
  + Approx. 25 hours total work
* **Joseph San Nicolas**
  + Integration of neural network into UI
    - Including the implementation of different options (genre, instrumentation, etc.)
  + About 150 lines of code
  + Updating and maintaining project website
  + Approx. 30 hours of work
* **Stefan Stukelja**
  + 3 Prototypes for UI (Python, .NET, Qt)
  + 400 lines of code
  + Configured deployable software package
  + 35 hours total work

Software Overview

**Module 1 - RNN\_Trainer.py**

This 296-line python programmodified an RNN implementation provided by Andrej Karpathy under BSD License. The implementation makes use of the library Numpy. This program invokes the midiscript programs (covered under their own modules) to strip collected MIDI data into text files for learning. The RNN then engages in learning for a set number of iterations, samples are produced along the way, and the trained model is saved so that it can be invoked in use of the application to produce music.

**Module 2 - RNN\_Sampler.py**

This 381-line python program is very similar to the previous module, RNN\_trainer. In this case, the program receives a number of command-line arguments that specify the model, tempo, and other characteristics of the music the user has requested. The program then loads in the desired pre-trained neural network, applies the parameters supplied to it, and uses the neural network to generate a midi sample of the desired nature.

**Module 3 - MidiScript**

This executable, originally written in C++, converts a comma-separated-value representation of midi data into a custom text script where each note or pause is represented by a character. This character data is then learned on by the RNN.

**Module 4 - MidiScript 2**

This executable, originally written in C++, converts the custom text script representation of music back into a CSV format. This is done so that the music that the RNN produces in the custom character format can be eventually converted back into a true MIDI file so that it can be listened to by the user.

**Module 5 - Midicsv**

This executable converts a MIDI data file into a CSV file that fully represents the same data, i.e. no information is lost. This is the first step in the conversion pipeline as this CSV file is then given to MidiScript to be converted once more into the custom character script for learning.

**Module 6 - Csvmidi**

This executable performs the opposite function as Midicsv. Namely, it converts a CSV representation of MIDI data back into true MIDI format so that the music can actually be played by an audio player. This is routinely done on the songs that the AI produces so that the user can hear them.

**Module 7 - Main Menu UI (Form 1)**

The current UI is designed and written in Visual Studio; the first form of the UI is the main window form. The main window form contains the media player that plays the midi files that the neural network generates. The main window form also contains buttons that access the setting menu form as well as invokes the neural network to generate a sample. The form is also able to adjust the volume of the sample that is playing. In addition, samples can be imported, removed, and saved from the menu bar. The project website can also be accessed from the menu bar.

**Module 8 - Setting Menu UI (Preferences)**

This form is in charge of choosing the settings that will be passed into the neural network. Theses settings include: the genre, the output instrument, the length of the sample, the tempo of the sample, and the seed/melody of the sample. Each setting can be changed via sliders, list boxes, and buttons.