DATA533 - Project Doc

Noman Mohammad & Nyx Zhang

Introduce

- Music Player - Noman
  - Users can randomly choose three notes of their choice
  - The notes will be sent to the music generator to be parsed into playable music
  - The user can then select play in order to play the music generated from their note selection
- Jam - Nyx
  - Analyze the input three notes.
  - Generate chord regression based on the analysis results.
  - Generate rhythm and beats.
  - Generate a melody based on the three notes and chords.
  - Mix the results into an audio file with the metadata.

Sub-Packages1 - Music Player

Module 1 - Interface generation

The class name is **MusicInterface**. Its main purpose is to create the underlying GUI for our program. It inherits the Frame class from the tkinter library.

Functions

- Function 1: Init function

  Calls the initialize_interface function which is designed to create the widgets within our interface

- Function 2: initialize_interface

  This function will create all the widgets for our interface

- Function 3 - 7 : value_C - value_B

  These functions act upon a piano key click. They call a function from out notes.py module and set values for our notes class

- Function 8: generate

  Will act upon the generate button. Calls our MusicJam.py module to generate music based on user input from selected keys. Also calls our notes.py module to clear notes object upon music generation and set status on the main label

- Function 9: play

  Will act upon the play button and play the generated ‘mid’ file from our MusicJam.py module. Also calls our notes.py module to clear notes object upon music generation and set status on the main label

Module 2 - Note generation

The class name is **Notes**. Its main purpose is to parse the data coming from the user. This includes duration, speed and error checking for expected note count.

Functions

- Function 1: Init function

  Sets empty lists for notes and time

- Function 2: getDuration

  This function will return a ratio of the elapsed time between note selection approximated to the nearest set value

- Function 3: addNote

  This function appends selected note to our initialized list along with time note was clicked so that calculation of speed and duration is possible

- Function 4: getSpeed

  This function calculates and returns the total elapsed time from first to last note click

- Function 5: getNote

  Returns list of notes for label update in Music player interface

- Function 6: clearNotes

  Clears the current attributes set for class from user input. Called when music is generated/played

- Function 7: getData

  Packages data in form accepted by MusicJam.py module to be parsed and turned into music. This returns a list of notes, speed and duration.

Module2 - Note generation

- SetNote will assign note values to the object
- GetNote will retrieve the note values from the object
- Duration will return a ratio of the elapsed time between note selection approximated to the nearest set value

Sub-Packages2 - musicJam

Module1 - music_analyzor

- Class Introduction

```
class MusicAnalyzor:
    def __init__(self, note_objects, num_bars):
        return _
    def key_analyze(self):
        return count_note_ipt
    def rhythm_analyze(self):
        return rhythm_start
    def rhythm_setting(self):
        return rtm_beats
    def chord_setting(self):
        return chords_lst
```

The class name is **MusicAnalyzor**. It analyzes the input notes Object and sets the key and rhythm of the music.

- Store the input notes from the music player as initial.
- Analyze and Extract the features.
- Using based information to set key and rhythm.

- Functions

  - Function 1: key_analyze

    Analyze possible music keys and determine the main key of the music by using empirical probability and music theory.

  - Function 2: rhythm_analyze

    Determine the start rhythm of the music based on the input rhythm analyzing result

  - Function 3: rhythm_setting

    Based on the analyzed-first-rhythm result to generate the whole rhythm by using empirical probability and music theory.

  - Function 4: chord_setting

    Based on the analyzed-first-rhythm result to generate the whole chord progression by using empirical probability and music theory.

Module2 - music_generator

- Class Introduction

```
class MusicGenerator(MysicAnalyzor):
    def __init__(self, note_object, num_bars):
        return _
    def chords_generate(self):
        return chords_lst
    def melody_generate(self):
        return melody_lst
    def melody_str2notes(self,melody_notes):
        return melody_lst
    def chords_f3notes(self,melody_lst):
        return melody_lst
    def mix_melody_chords(self)
        return stream
```

The class name is **MusicGenerator**. It inherits the **MusicAnalyzor** class

- Set all the parent class features as initial.
- The primary model that generates the music.
- Output the music audio, music score picture, and other metadata to the music player.

Functions

- Function 1: chords_generate

  Transform the generated chords to music21 objects.

- Function 2: melody_generate

  generate notes object(pitches and durations) stream based on the beats setting result and generated chords progression by using empirical probability and music theory.

- Function 3: melody_str2notes

  a tool for melody_generate to transform chords objects from roman numerals.

- Function 4: melody_f3notes

  a tool for melody_generate to replace the first several notes to the inputs.

- Function 5: mix_melody_chords

  Mix the stream of notes and stream of chords.

  Mix the score of the music.

  Return the chords, score, and other metadata.

references of music-related metadata

```
metadata_sample =
{
   // beats per minute
   // default and flexible
   "bpm":111,
   //  the major or minor scale around which a piece of music revolves
   //  default and fixed
   "Key":"C",
   // rhythmic pattern constituted by the grouping of basic temporal
units, called beats, into regular measures, or bars
   // default and fixed
```

```
"meter":"4/4",
// music genre: such as Blues, Jazz, Metal.
// default and fixed
"genre":"pop",
// chords of music melody in this key
// model generated the whole melody chords divided by the bars.
"chords":[
   {
         // chords
      "chords":[
         "C",
         "G",
         "Am",
         "B-"
      ],
      // Offset of the chord in the unit beat
      "chordRythm":[
         "1",
         "2",
         "3",
         "4"
      ]
   },
   {
      "chords":[
         "G",
         "A",
         "C",
         "D"
      ],
      "chordRythm":[
         "1",
         "2",
         "3",
         "4"
```

```
         ]
      }
   ],
   // notes of music melody in this key
   // user random input several notes, then model generated the whole
melody notes divided by the bars.
   "notes":[
      {
            // notes
         "notes":[
            "C",
            "E",
            "A",
            "G",
            "C",
            "D",
            "F",
            "D"
         ],
         // Offset of the notes in the unit beat
         "notesRythm":[
            "0.25",
            "0.25",
            "0.125",
            "0.75",
            "1",
            "0.25",
            "0.5",
            "1"
         ]
      },
      {
         "notes":[
            "G",
            "A",
```

```
            "F",
            "G",
            "C",
            "D",
            "G",
            "D"
         ],
         "notesRythm":[
            "0.75",
            "1",
            "0.5",
            "0.25",
            "0.5",
            "1",
            "0.5",
            "1"
         ]
      }
   ]
}
```