Joseph Lee and Bryan Werth
SoftDes Mini-Project 4
Interactive Algorithmic Music Composition

Project Overview:

For this mini project, we were developing an interactive algorithmic music composition program. In essence, the program would gather information from the user such as chord progressions, the key of the piece, the general mood, etc. and then use that information to generate a melody within those parameters. Due to time constraints, the project in its current form prompts the user for command line input regarding the chord progression and rhythm. The program then generates a chord tone melody.

Results:

We successfully produced an interactive command line program that can take information about a chord progression and use it to generate a chord tone melody with harmonies to support it.

The figure above shows an example output generated using our program.

Implementation:

The three major classes designed during the production of this program are Note, Chord, and ChordProgression (figure 1). Functionally, we wanted the program to take a chord progression as an input and produce a melody made by randomly chooses a note from each chord in the chord progression. To accomplish this, the chord progression had to be broken down into more simple classes with related methods that could support them. The ChordProgression class also had to have a series of related methods to support the goal of the program.

The note class had to effectively describe the note as a whole and be able to describe the notes surrounding it. The producenote() method takes an integer number of half steps as an

input and returns the note that is that distance from the original note. This method is used in the createchord() method within the Chord class to define a list of notes that make up a chord. This list of notes is used by the createchordprogression() method to create a list of lists of notes. Finally, this output is used by the createmelody() method to randomly pick a note from each of the lists of notes.

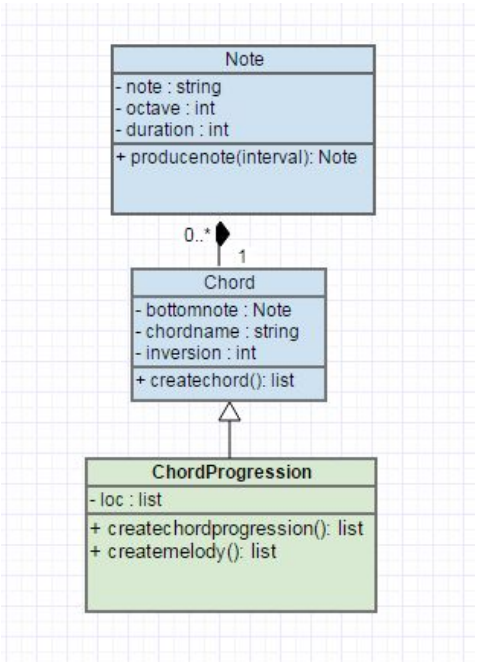

Figure 1. The UML class diagram portraying the relationships of the three classes associated with our program

Reflection:

In this mini-project, one thing that went particularly well was that we worked well together and found the project itself extremely engaging and fascinating. We felt like we really learned a lot in the project, despite its slightly less interactive nature as compared to a project like making a game or making interactive data visualizations. That being said, our project has the full potential to expand to those more interactive areas in many directions - for instance, we want to implement a GUI and in the future, may even begin to incorporate elements of facial recognition and use that to influence the way that our algorithm composes music. We both plan on continuing work on the project even after the mini-project is officially over. Things that maybe didn't go as well - due to time constraints, we were not able to implement as many of our stretch goals as we wanted to. The way that we handled the team aspect of the project was that we both started two simultaneous implementations with slightly different focus. One of us began creating all the necessary classes, for instance, the classes for notes, chords, chord progressions etc. while the other used pre-existing libraries to hammer down the actual melody generation algorithm and the MIDI file interface. After a brief period, we took the algorithmic aspects from the code written for the music21 library and adapted it to work with the classes and objects that we had defined. From there we transitioned into pair programming and

simultaneous programming (using floobits) to finalize the integration of the two scripts and to work on the user interface. I think perhaps the only thing we would do differently would have been to realize that our time was more limited than we first anticipated and it probably would have been more efficient to have a clear picture of which stretch goals were reasonable and which ones were not before starting.

chordprogression.chord.note.createinterval()