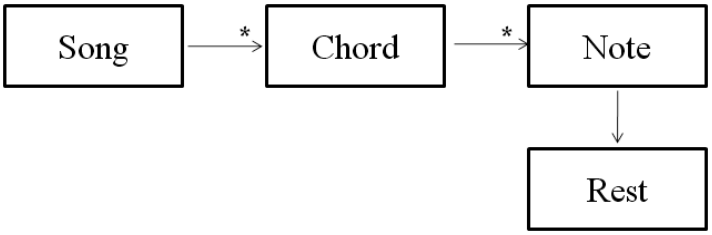
Project Design Proposal

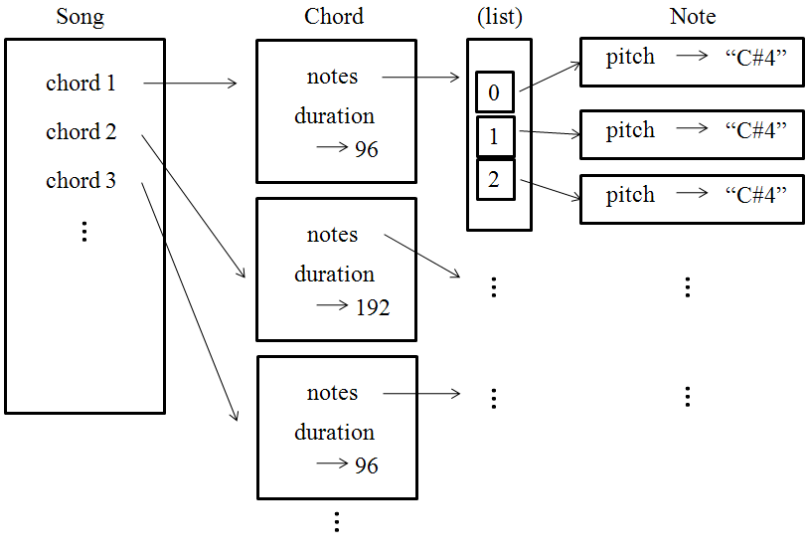
1. The Story

In developing the project design, we researched and ideated to find a way to easily generate music with the program that would follow musical theory to a degree. We wanted the program to work with any MIDI file so that the user could put any seed in and create a new musical work.

We initially considered quasi-random generation of music where the program would understand some musical theory (such as common chord progressions) and generate music based off of a short seed file. The downside to this method is that the program would be required to be trained in a significant amount of music theory, which would not be an interesting use of time from a software design perspective. Rather, we decided to implement a Markov analysis scheme, which wouldn't need to understand music theory and would merely generate rational output based off of the structure of the seed musical piece. As Markov can create sentence output that makes some sense, we hope that a similar method can be used to create music that sounds plausible. In order to implement a Markov method, we needed to create a way to represent notes as object that we can manipulate. We considered changing the pitch of notes based on Markov chains and maintaining the duration of notes from the original piece, or changing the duration but maintaining the pitches, or determining both through Markov analysis. As none of these methods would be particularly harder or easier to implement, and that we don't know how they will sound in the final design, we decided to give the user choice over generation of pitch, duration, or both.



<Figure 1. Class Diagram>



<Figure 2. Object Diagram>

2. Data Structures

Our first data structure is a Note. A Note class is based on the musical definition of note, as it represents the pitch of a given note. However, our definition is broader, as it also considers a Rest a type of Note (Rest isa Note).

Our next class is a Chord. Also based off the musical definition, a Chord stores all the notes that are being played at a given time. Our definition of Chord is broader than the musical definition in two ways: First, we consider a single Note to be a Chord. Second, that single Note may be a Rest, if no notes are being played then. A chord consisting of a Rest and a Note other than a Rest is an invalid data structure.

In addition, Chords store their duration. Storing duration in each Chord rather than in each Note was a design decision made for simplicity. This means that no Chord may overlap with another Chord. This makes Markov generation easier, since we can represent music as simply a sequence of Chords. The tradeoff is that our code will handle a certain musical pattern incorrectly. If multiple notes start at the same time, but end at different times, our data structures will fail to represent this correctly. When this pattern occurs, our code will pretend that all notes that begin simultaneously end as soon as any one of them end.

Our finally data structure is a Song. A Song stores a list of all the Chords that make up a song. It also stores peripheral information, such as the time signature and instrumentation.

These data structures were designed to facilitate Markov generation, and we expect them to be very adequate for meeting this minimal goal. Our reach goal, of using music theory to impose constraints on the Markov generation, will likely necessitate modifications to the class structure. For example, a Chord containing the notes C E G could also store that it is a C major chord. We will examine this issue more thoroughly at the appropriate time, but we expect that out class structure will require very little modification.

3. Development Plan

(1) What have been done

We wrote the basic classes that are involved in the design. Using these classes, we have written a simple program that reads a midi file, stores the pitch information of each note and performs Markov analysis on the notes.

(2) Next phase

We are going to perfect the Markov analysis program. Next, we want to be able to perform controlled Markov analysis where the user can control certain aspects of the music generation (e.g. chords/ pitch range/etc.)

We also want to write a GUI that allows the user to interact with the program. The user can select which style of music generation that they want to use, view the generated music on a musical stave, and perhaps interact with the music staff during music generation.

(3) Work division

There are two sub-teams each composing of two people. Shane and Jeff are working on music generation using different generation methods. Tom and Rui are working on GUI implementation.

(4) Communication

The group decided upon the classes, the music generation methods and the general components of the Gui together. The actual codes are written individually. The group meets frequently to update each other on what have been done. Before splitting up, we agreed on the basic class structure of all the classes involved in the program, the common understanding of the class structure acts like a reference point between the two subgroups.