Taking Note of Computerized Improvisation

The automation of piano improvisation using Python

Abstract

Musical improvisation, despite its apparent randomness, has recognizable patterns. All songs have a predetermined key, which gives musicians a set of notes that they can play during an improvisation. These notes can be grouped together to form chords that also exist within the song's key. Most songs consist of four-chord loops, or progressions, that can often be predicted and automated. After various chord progressions and melodies were tested, a program was designed to randomize a key, a progression, and a harmonious melody. In addition, an interface was created so that users could play an on-screen piano and click a button to generate an improvisation. Although these improvisations are rather short in length, they demonstrate the increasingly growing importance of technology in musical production. The significance of programs such as these will continue to grow as producers and performers look for ways in which their music writing can be expedited and digitized.

Introduction

All music consists of specific characteristics that can be predicted, repeated, and automated. Every song has a predetermined key, which creates a scale of eight pitches. These pitches can then be combined in chords of three pitches to form chords, which can eventually be accompanied by a melody. Whereas jazz music often strays from a song's key (and contains "accidentals," or pitches that do not exist in the scale), classical and pop songs are usually built entirely off of the scale.

For the purpose of this project, the complexity of automated chords remained relatively simple to ensure a harmonious final product. If the eight pitches in a scale are labeled numerically, chords can be created in an organized fashion. For instance, if the key of a song were F# (read as "F sharp"), the scale would consist of the pitches F#, G#, A#, B, C#, D#, E#, and, once again, F#. These pitches are viewed numerically, so that $x = 1$ at F#, $x = 2$ at G#, and so on. Basic chords are built by choosing a starting pitch, assigning it a variable x, and combined with pitches $x + 2$ and $x + 4$. In the key of F#, possible basic chords are listed as follows:

- I: F#, A#, C# (F# major)
- ii: G#, B, D# (G# minor)
- iii: A#, C#, E# (A# minor)
- IV: B, D#, F# (B major)
- V: C#, E#, G# (C# major)
- vi: D#, F#, A# (D# minor)
- vii: E#, G#, B (E# diminished)

Chords are named by the location of the first pitch of the chord in relation to the scale. Musicians capitalize the Roman numerals (I, IV, and V) if the chord is major (creating a "happy" sound) and lowercase them if the chord is minor (creating a "sad" sound). By the same logic, a chord written as EM (or simply E) is read as “E major,” whereas a chord written as Em is read as “E minor.”

Once the importance of chords is recognized, chord progressions—groups of four chords that all exist within the given key—can be analyzed within music. Analysis of pop songs revealed that a wide variety of chord progressions sound objectively ”good,” and can therefore be used in an improvisation. For example, the progression I-V-vi-IV is a system that the human ear is comfortable with, due to its prevalence in pop music. This, along with dozens of other chord progressions, were placed into a master list from which a program selected from in order to create a foundation for a short eight-bar improvisation.

However, music does not only consist of chords. Chords simply serve as the basis for songs; they provide musicians with a set of pitches that can be played or sung as a melody. Once the key of the song or improvisation is determined, a computer can use the major scale to create a list of melodies that will sound pleasing to the ear when paired with the aforementioned progression.

In its essence, this program was designed to determine the extent to which improvisation can be entirely automated, using simple and accessible tools. As music technology is a growing field, the combination of computer science and music has been seen as an increasingly useful alternative to human DJs and music producers. With artificial intelligence designs such as Siri at the disposal of smartphone users, song recognition can be paired with improvisation technology to produce a virtual DJ. This would completely revamp the structure of the music industry, in both production and performance.

Materials and Methods

In its simplest form, every song can be categorized into two distinct sections: a background and a foreground. The background (which, in pop music, usually consists of percussion, bass, backup vocals, and, most importantly, a chord progression laid out on piano or guitar) sets a foundation on which the foreground can build. My program was designed to generate chord progressions on piano. The foreground is a single voice in the form of a melody. Whereas the background gives music a mood, melodies are what gives music a texture; they are essential in order to give songs a catchy, memorable feel. It is essential to focus on these two pieces that coexist within a song—or, in this case, a thirty-second long improvisation. For ease of the reader, the thought processes behind each piece are individually explained in the space that follows.

1. Before the program was written, extensive research of modern-day music was completed so that common chord progressions and melodies could be analyzed and stored as applicable for an improvisation. Songs such as Pachelbel's "Canon in D" (the easily recognizable song played at weddings, written in the late 1600s) contain sets of chords that are still implemented into pop songs today. Analysis of both classical and modern music shows that the first of four chords in a progression is often the I, IV, or vi chord (refer to the chord labels in the Introduction section). After testing all possible progressions following the I, IV, or vi chord, a series of if-then statements was created in a Python script that randomizes a first chord, uses this chord to choose from a list of possible second chords, and continues until a series of four pleasing-sounding chords has been created. Each of the four chords is stored into the variable "chord1," "chord2," "chord3," or "chord4." Then, a list with the name "progression" is created so that:

$$progression = [chord1,\ chord2,\ chord3,\ chord4].$$

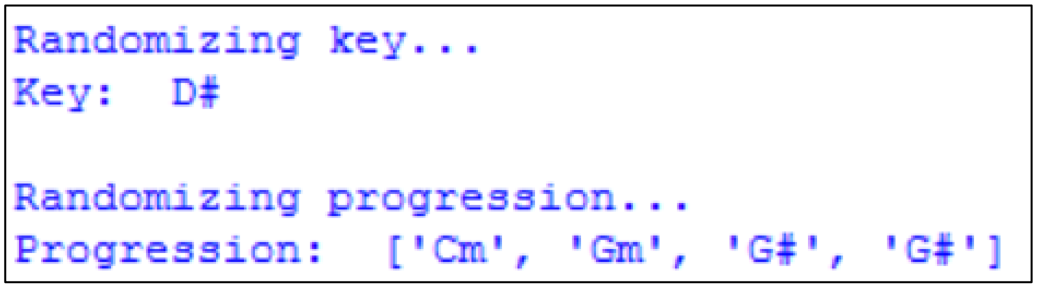
   At this point, the program completed one task: choose one chord progression from the 2208 individually-tested progressions. A few possible results of the program could include progressions such as [E, B, C#m, B], [D, Bm, G, D], and [Am, G, C, Em]. Note that, at this point, the program only produced a list of chords; no sound was emitted by the computer.

2. Recall that songs are built with both chords and a melody. In both classical and modern music, melodies consist of notes that rarely stray from the song's key (and therefore do not contain "accidentals"). Therefore, the decision was made to only allow the computer to select notes that fall in the song's key. Specifically, the three notes that build up each chord (as demonstrated in the Introduction section) are placed into a list, and, one by one, randomized pitches are selected to play over the chords.

   In an effort to create a variety of sound, I concluded that multiple rhythms should be incorporated into each improvisation. As rhythm is mathematically organized into subsections (whereas one quarter note fills a beat, two eighth notes fill the same time... as do four sixteenth notes, eighth thirty-second notes, etc.), the pause time between each note in the melody is adjusted so that the final improvisation consists of varying rhythms. As with the other components of this program, rhythm is almost entirely randomized; users cannot predict the rhythm of any given improvisation. However, the number of times that the rhythm can change in one period of time is limited so that overall structure is maintained.

By this point the project, the program did not actually produce sound; it only created lists of chords and notes. Although these lists would be helpful to a human musician, computers cannot read and play chords through a simple Python function—third-party modules must be used. So, Python's Pygame module was incorporated into the program in order to allow the creation of sound. An audio recorder was used to record .wav files of two octaves (25 notes) of a piano. Then, the program was adjusted so that, when prompted, the computer took the randomized progression as an argument, analyzed each element as a chord, and determined

Illustrations



```
Randomizing key...
Key:  D#

Randomizing progression...
Progression:  ['Cm', 'Gm', 'G#', 'G#']
```

Figure 1: This is an example of an output for the progression randomizer that is built within the program. Both the key and a progression in this key are generated and printed into the Python shell. The list displayed after "Progression:" is further analyzed by the program so that the progression is played through the computer's speakers.

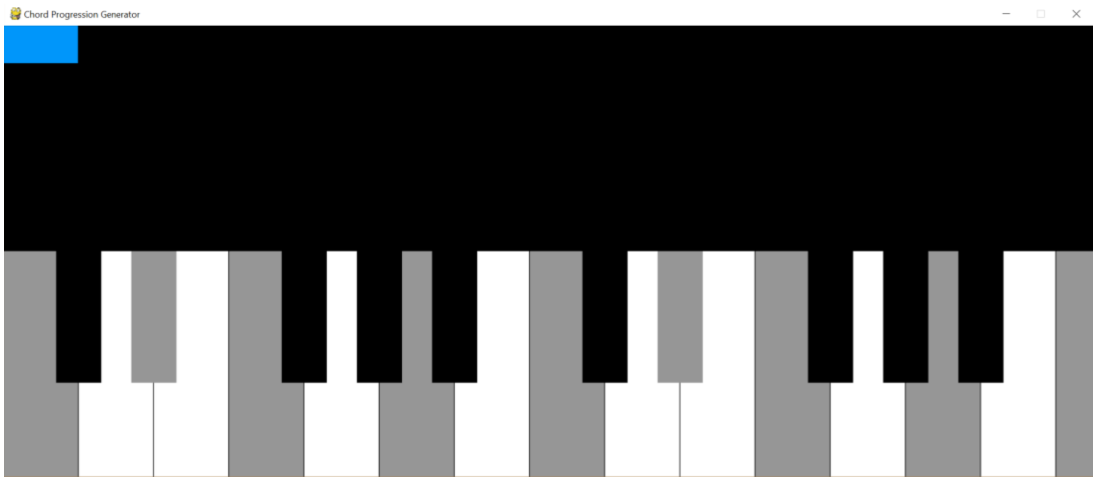
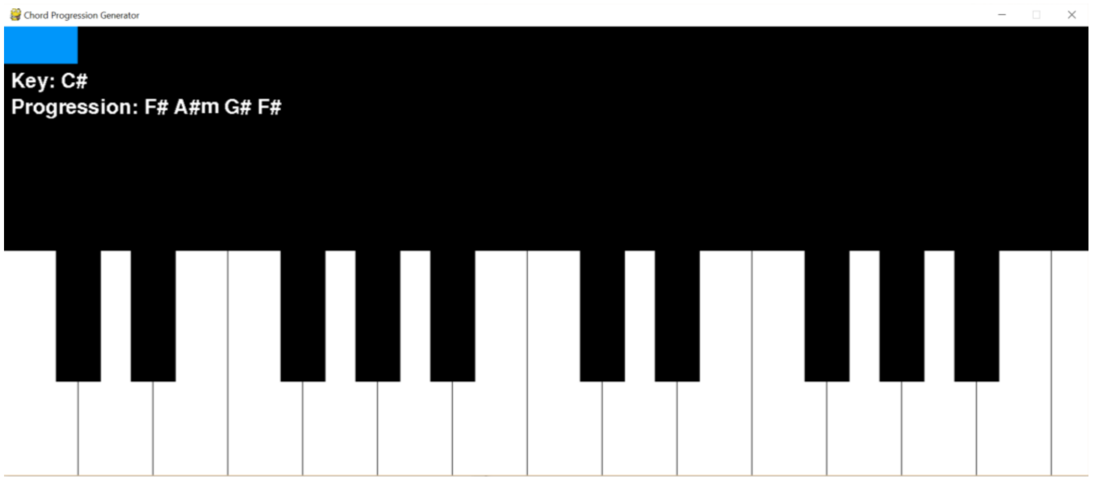


Figure 2: This is the piano interface, in which users can interact with an on-screen piano. In this screenshot, I am playing an F$_7$ chord using the keys on my keyboard. The chord can be heard through the speakers of the computer.



Key: C#
Progression: F# A#m G# F#

Figure 3: This is the result shown after the blue button in the upper-left hand corner is clicked. The key and progression of the generated improvisation are shown, and the improvisation plays.

Results and Discussion

Because of the extensiveness of computer science, it is difficult to collect “final results” from a project such as this. Just as electronic devices and applications require constant updates, this automated improvisation program will certainly be improved in the future. However, for the time being, it is essential to recognize the implications that this program has on music technology:

1. This program focused on the characteristics of music that humans are accustomed to hearing. From common chord progressions to patterns in melodies, most pop songs (and many classical songs) have similar characteristics that were thoroughly analyzed before the creation of this program. By analyzing and finding patterns in music, it was discovered that, if these patterns are recorded and implemented into code, **improvisational music can indeed be randomized, while the comfort of organized music remains**.

2. Python is a relatively simplistic programming language; less than 1500 lines of code were written to create the entire piano interface (which is displayed in the Illustrations section). Despite its seemingly basic nature, Python is extremely powerful in importing sounds (especially when using the Pygame module), which allows it to accurately time sounds and keep the improvisation on a constant beat. In other words, the conclusion can be drawn that, **in the future, Python can allow computers to perform more sophisticated improvisations, in terms of the complexities of the progressions and melodies**.

Noting these conclusions, further action will be taken in order to improve the program. Surprisingly, despite the complexities of modern-day computer science and artificial intelligence, music technology is still in quite rudimentary stages. With the promotion of programs such as this, music technology will likely advance in the next few years. Writing, recording, and performing of music will start to become automated at an increasingly rapid pace.

Until very recently, most music-related technology has only been related to recording and editing human-made sound. Although the achievements brought about by music editing software is notable, it inhibits the automated creation that is encouraged by computer scientists. Improvisation is a natural component of a musician's experiences as a performer; technology should begin to reflect this love for spontaneity in performance.

Conclusions and Future Work

The influence of this program on the field of music technology has been touched upon in previous sections. This section will explore the features that this program will take on in the future, as a response to the recent surge of both artificial intelligence and music technology. It will also analyze the importance that programs such as these will have in the entertainment industry.

The program itself will be made much more sophisticated in the future, in order to mimic the varied nature of music. As mentioned before, melodies sometimes contain “accidentals,” or pitches that are not in the song’s key. They are quite common in jazz pieces and certain classical songs. By incorporating accidentals into the program’s foundation, the complexity of each improvisation will be increased.

Additionally, it is important to realize that most songs are built up of more instruments than simply piano. A second feature that will be added to this script will be the ability to replace the piano with guitar and other string instruments. The melody could be played on a string instrument (such as piano, guitar, violin, etc.), a woodwind instrument (flute, clarinet, saxophone, etc.), or a brass instrument (trumpet, trombone, tuba, etc.). The possibilities are endless—sounds of each instrument will simply be downloaded and applied to the randomization portion of the program.

A third addition to the program will be the implementation of different moods into the improvisations generated. Multiple types of keys exist (major, harmonic minor, natural minor, etc.), and each one has an individual mood associated with it: songs in major keys are generally happy, songs in harmonic minor keys are eerie, and songs in natural minor keys are generally sad. By including a drop-down feature that allows users to choose a mood (happy, sad, or eerie), the program will be able to choose a scale and generate an improvisation that applies to the mood selected.

The application of machine learning must be explored in any modern music technology program. With song recognition programs (such as those included in intelligent personal assistants, including Cortana and Siri) so readily available, the entertainment industry is only a step away from designing automated DJs. In other words, the same tools that are used to identify the name of a song (which involve breaking down portions of the song into individual frequencies, and then plotting these frequencies on a graph) can be used to categorize a list of songs by tempo, key, and overall dynamics. Especially in the case of fast-paced dance music, tempo can easily be found by following the pulse of a bass drum. Therefore, a program can be designed that organizes a list of songs based on similar tempos and keys, performing the same jobs that DJs complete. Then, transitions can be added between each song so that the music does not stop during a party or other event. In other words, this program can help create an efficient and inexpensive way to control the flow of music from one song to another.

This project has proved that a future of entirely automated music is on the horizon. As human reactions to music are further studied, comfortable patterns within improvisations will continue to be implemented into scripts. Eventually, DJs will be able to simply input a list of songs into a program, and the computer will organize it by mood and connect them with seamless transitions. As music technology rises in importance, I will continue to strive for a perfect computerization of musical improvisation.

References

"Chord Progressions." The Music Theory Academy. Music Theory Academy, n.d. Web.

Tymoczko, Dmitri. "The Geometry of Musical Chords." Science 313 (2006): 72-74. Dmitri.Tymoczko.com. American Association for the Advancement of Science, 7 July 2006. Web. 17 Aug. 2017.

Van Nieuwenhuizen, Heinrich A. "The Study and Implementation of Shazam's Audio Fingerprinting Algorithm for Advertisement Identification." (n.d.): n. pag. North-West University. Web. 27 Aug. 2017.