# Procedural Content Generation of Music Melodies using Music 21

Gustavo Kang Shim

*Computing Science Department, Thompson Rivers University*
*805 University Drive, Kamloops, BC, Canada*
gkangshim@gmail.com

***Abstract*— Outside of the standard make of music as human-made sometimes, they are randomly generated based on the idea that we can input a ruleset through an evaluator that can run multiple times to develop a music that is musically appealing based on the rules themselves. Being a musician who struggles to find inspiration to create melodies that pertain to people's minds. Melodies generated through a well-written piece of code are what musicians can use to personalize their own wills into their procedurally generated music. In this paper, I introduce a proof-of-concept procedurally generated melody that explores the relationship between music theory and melodies across many different theories and ideas. For that purpose, I have created a Python script using the music21 module where musicians can generate melodies from a based length, key, beats per minute, and chord progressions. By engaging an evaluator in the program, I am able to create music that is better aligned with some of the music theory topics I am including in this project. These melodies are both interesting to play with in a music-making software but also create musically appealing melodies.**

## I. Introduction

Melodies are generally perceived as a fundamental part of music. Usually, they are the heart of a song, where a chord progression will follow the ways of the melody to generate a memorable sound using Music theory concepts such as Melody Contour [1]. Following practices in music productions, musicians have created melodies in various ways to amplify a melody's perception to the audience. As the music theory behind the creation of a song becomes more complex, studies such as ()() [2] have tried to augment music training through complex theories and tests. Moreso, many of those attempting to create new music struggle to create the music's melody rather than a chord progression itself. eg. (EXAMPLE HERE). Within a melody, some musicians have already tried to implement an automatic creation of different melodies through coding and a fitness function, but none have really implemented a 10-check evaluator when it comes to defining what a good melody is.

In this paper, I created a procedurally generated melody program that outputs a "good" sounding - defined by and considered by the evaluator function as nice sounding - which can later on be transported into Music-making Software. With the power of the library music21[5] and Music Theory concepts such as Dynamics, Melodic Contour, Rhythmic Diversity and others. Not only that, the music generator itself has a certain set of rules it also has to follow which can influence the outcome of the melody itself.

## II. The Procedurally Generated Melody

The "generate_melody" function initializes a musical stream using the music21 library and sets initial parameters, including the starting note, melodic direction, and the maximum allowed step size.

### *A.* Dynamic and Articulation Elements

Within the code, lists of dynamic and articulation values are defined, which are selected randomly for each note in the melody. This introduces diversity in the generated melody's expression.

### *B.* Melody Generation Loop

A loop iterates over a specified "length" to generate the melody. Within this loop, the code checks whether the next note will be a rest or a musical note. Rests provide silent intervals within the melody.

### *C.* Note Generation

For non-rest notes, the code randomly selects note duration, dynamics, and articulation values. These parameters are used to create a "note.Note" object representing the musical note.

### *D.* Melodic Progression

Depending on the availability of a "chord_progression," the code either selects the next note based on the provided chord progression or employs melodic contour logic to choose the next note within the scale. This aspect of the code contributes to the harmonic or melodic structure of the generated melody.

### *E.* Maximum Step Control and Direction Variability

To ensure that the melody does not exhibit overly large jumps, the code restricts the maximum step size, which can be adjusted through the "max_jump" parameter. Additionally, the direction of the melody may change with a 20% chance, introducing variety and unpredictability.

### *F.* Conclusion

The "generate_melody" function is a versatile tool for automated melody generation, capable of producing diverse musical sequences by taking into account key, rhythm, and optional chord progression. Its adaptability and capacity for creating expressive and dynamic melodies make it a valuable component in various music generation applications.

In summary, this code segment represents an essential component in the automated generation of musical compositions, demonstrating the integration of musical theory with computer programming to create expressive and dynamic melodies.

## III. The Evaluator of The Melody

The evaluate_melody that evaluates the quality of a given musical melody is represented as a "melody_stream" with respect to a given "chord_progression." The function calculates a score based on various musical criteria, such as chord transitions, consonance/dissonance, melodic contour, rhythmic diversity, stepwise motion, repeated motifs, use of dynamics, articulation variety, range analysis, and phrase length. Below is an explanation of each part of the code:

*A.* Chord Transition Analysis: This section assesses the melody's compatibility with the underlying chord progression. It analyzes the melody's transition to chord tones and rewards the melody when it aligns with the chords.

*B.* Consonance and Dissonance: For each note in the melody, this part checks whether it harmonizes with the current chord. Consonant notes are rewarded, while dissonant notes are penalized.

*C.* Melodic Contour: The code evaluates the melodic contour of the melody. It penalizes large jumps (larger than a major third) to encourage smoother melodic lines.

*D.* Rhythmic Diversity: It calculates the diversity of rhythmic values used in the melody. Melodies with a wider range of rhythmic values receive a higher score.

*E.* Stepwise Motion: The function counts the occurrences of stepwise motion (consecutive notes that are a second apart). A melody with more stepwise motion is rewarded.

*F.* Repeated Motifs: This section identifies and counts repeated motifs (sequences of three notes) in the melody. The count of repeated motifs is added to the score.

*G.* Use of Dynamics: The code rewards melodies that use dynamics (changes in loudness) in more than one-fourth of the notes.

*H.* Articulation Variety: This part evaluates the variety of articulations used in the melody. Melodies with more diverse articulations receive a higher score.

*I.* Range Analysis: The function calculates the pitch range of the melody and rewards melodies that stay within a 2-octave range (melody_range < 24).

*J.* Phrase Length: It examines the length of musical phrases within the melody. Melodies with moderate phrase lengths between 3 and 10 notes are rewarded.

*K.* Final Score Calculation: The individual scores for each criterion are summed up to determine the overall score of the melody. A higher score indicates a melody that adheres to the defined musical criteria.

This function serves as a tool for assessing the quality of generated musical melodies by quantifying their adherence to established musical principles. It can be valuable in the context of algorithmic music composition, where the quality of generated compositions needs to be evaluated objectively.

## IV. Conclusion

In summary, this paper presents a novel approach to automated music composition by integrating computer programming and music theory. The "generate_melody" function offers musicians a versatile tool for creating expressive melodies while adhering to musical principles. The "evaluate_melody" function provides a comprehensive method for assessing the quality of generated melodies. This research offers valuable tools to inspire and enhance music composition, making it easier for musicians to infuse their compositions with melodious and harmonically pleasing elements. It marks a significant step in bridging the gap between creativity and computational capabilities in the field of music composition.

## References

[1] I will add references i promise ;-;