# Mucify.js - Procedural Music Generator

Nikolay Troshkov, Innopolis University
https://github.com/ariser/mucifyjs

***Abstract*—This article describes a JavaScript library for music generation. With *Mucify.js* user can generate melodic lines, that, when combined, can result in a musical composition. The article briefly describes the algorithm for generation and input parameters that can be changed in order to generate different melodies. The output can be used in games as well as in any other environment that requires musical arrangement.**

***Keywords*—PCG, procedural generation, music**

## I. Algorithm Description

Melody generation in *Mucify.js* is based on music theory and author's experience in improvisation.

### A. Overall principle

The algorithm creates a sequence of bars that follows a certain pattern. Human brain appreciates patterns. Music can be pleasant because of expectations that we have and because those expectations are being satisfied. Thus, each melodic phrase should start with *tonic*, which is the main note in a scale, and should end with it as well. Constructed in this way, melodic phrase seems complete and proper. In the most simple approach, which is demonstrated here, any other note from a scale can be played in a phrase between the first and the last one. The *Mucify.js* sticks to this approach.

### B. Phrases generation

Each phrase consists of 4 bars. Each bar is a sequence of notes of different values. Combined values of all notes in a bar equals 1. Each bar has its key note. This means that the first note in a bar will be this note, while the remaining notes could be randomly picked from a scale. The first bar always has *tonic* as the key note. The last one can have any key note, but the last note will always be *tonic* as well.

### C. Octaves

A scale is simply a sequence of intervals that describes notes that can be used within this scale. However, notes pitch can be different. In the *Mucify.js* algorithm notes generated for a bar remain within a single octave that should be explicitly specified. This enables user to create melody lines for different purposes: bass line, melody line, etc.

## II. Taxonomy

### A. Online vs. Offline

**Online.** *Mucify.js* can generate melodies in real time.

### B. Necessary vs. Optional

**Optional.** While music can make a big difference for atmosphere, it is certainly not necessary element for a game to be playable.

### C. Degree and Dimensions of Control

**A lot of control.** *Mucify.js* has a lot of properties that can be adjusted. From instruments and tempo to probability of certain note to be played. Fine tuning can result in different melodies suitable for different game situations.

### D. Generic vs. Adaptive

**Generic.** The algorithm does not process user input. All calculations are based on music theory and random factor.

### E. Stochastic vs. Deterministic

**Deterministic.** Huge amount on control on the generated output makes the algorithm predictable. The same input parameters will result in more or less the same melody. However, each time it will be slightly different.

### F. Constructive vs. Generate and Test

**Constructive.** As an online algorithm, *Mucify.js* melodies do not have a privilege to be tested beforehand.

### G. Automatic vs. Mixed Construction

**Automatic.** While generated melodies can be used as a reference for manually created ones, *Mucify.js* is intended to create assets that are ready for use right away.

## III. Generator Tuning

This section describes possible adjustments that can be made in order to achieve different outputs from the generator.

### A. Scales

One of the most basic things that can be changed is scale. *Mucify.js* provides seven basic scales: Ionian, Dorian, Phrygian, Lydian, Mixolydian, Aeolian and Locrian, - along with Altered scale, which is commonly used is jazz, and minor (Aeolian) pentatonic, which is a common blues scale. Switching between these scales can result in changing of resulting melody sound.

### *B. Degrees sequences*

The other thing that influences the sound of a melody is order in which bars' key notes are being placed. A common blues progression, called blues square, is the following sequence of degrees: tonic, tonic, subdominant, tonic, dominant, subdominant, tonic, dominant. The blues square is also provided in *Mucify.js*. Along with it there is a tool for generation random sequence of degrees. Each of degrees can be selected with its own probability. These probabilities can also be adjusted.

Also, user can create their own determined sequence of degrees.

### *C. Tempo*

Another basic property that strongly affects the sound is tempo. In *Mucify.js* we can set tempo for a melody in bmp (beats per minute). The default value is the default tempo in music which is 120 bpm.

### *D. Note values*

One more property that can be adjusted is note values. While it is not possible to set notes values in the resulting melody directly, we can influence the probability of one or another value to be assigned to a note. This case is similar to the degrees sequence case.

### *E. Instrument*

This section is not a part of a generation algorithm, but of a playback algorithm. In *Mucify.js* playback is performed using a MIDI player. This enables usage of different instruments for generated melodies, which also strongly affects the resulting sound.

## IV. Conclusion

*Mucify.js* provides a method for music generation in real time. Being highly customisable, the algorithm can produce a variety of different melodies that can be used in different game situations. Further development of the project can enable the algorithm to generate even more diverse music. The nearest plans are to develop an ability to change time signature, and to create a chords generator.

## Acknowledgement

The author is grateful to professor Joseph Brown who made this project possible. The idea of the project wouldn't appear without his course of Procedural Content Generation in Games.

The author also wants to reference the JavaScript library for creating and playing MIDI: MIDI.js. The GitHub repository of the project: https://github.com/mudcube/MIDI.js. Special thanks to Benjamin Gleitzman who created a set of pre-rendered sound fonts that were used in the project. The GitHub repository for the sound fonts: https://github.com/gleitz/midi-js-soundfonts.

## Materials Used in Development

Diatonic Modes - http://guitarpages.narod.ru/Harmony.html (Rus)

List of Musical Scales and Modes - https://en.wikipedia.org /wiki/List_of_musical_scales_and_modes

Degree in music - https://en.wikipedia.org/wiki/Degree_(music)