# Task 8 - Switch From Three to Two Melodies + Add to Constraint System

In the previous task, it was determined that having three melodies sounded like a mess and the third melody did not add anything of substance to the project, so I decided to remove melody3 from the music sample and add the bassline generation as one of the options for the bassline of melody2. This task also  adds some extra features to the constraint system to generate more interesting results for the secondary melody. These include “octave harmonization,” or harmonizing by taking melody1 and moving it up or down an octave, and stepwise generation of the bassline, so the pitches jump less often.

## Melodies Clean-up

Bassline has now been incorporated into an option for melody2, instead of being a separate melody altogether. Code throughout the program has been cleaned up to match the removal of melody3 from the *music* object.

```
; Method to generate melody2 as either a harmonization,
; permutation, or random melody
( defmethod generate-melody2 ( ( m1-notes list ) )
    ( setf choice ( random 5 ) )
    (cond
        (( = choice 0 )
            ( create-harmonization m1-notes )
        )
        (( = choice 1 )
            ; permutation
            ( list-permutation m1-notes )
        )
        (( = choice 2 )
            ;random melody
            ( generate-melody1 )
        )
        (( = choice 3 )
            ( generate-bassline )
        )
        (( = choice 4 )
            ( create-octave-harmonization m1-notes )
```

```
        )
    )

)
```

## Octave Harmonization Demo

```
[1=2]> ( demo--octave-harmonization )
----------------Individual 2----------------
X:1
T:Individual 2
C:Dystopian Tuesday
M:4/4
L:1/4
Q:1/4=120
V:S clef=treble name=S
V:A clef=treble name=A
%%score [ ( S ) ( A ) ]
K:C
V:S
%%MIDI program 0
F' C2' F/2' G' E/2' E/2' C2' D/2' G/2' E2' E2' F' F/2' D2' D' G'
C2' A' E' C' B'
V:A
F C2 F/2 G E/2 E/2 C2 D/2 G/2 E2 E2 F F/2 D2 D G C2 A E C B
NIL
[1=3]>
```

## Octave Harmonization Demo Code

```
; Method to demo the create-octave-harmonization method.
( defmethod demo--octave-harmonization ()
    ( setf melody1 ( generate-melody1 ) )
    ( setf melody2 ( create-octave-harmonization melody1 ) )

    ( setf m
        ( make-instance 'music
```

```
            :melody1 melody1
            :melody2 melody2
            :rank 0
            :melody1-rank 0
            :melody2-rank 0
            :num 2
        )

    )

    ( easyabc-display m )

)
```

## Octave Harmonization Code

```
; Method to create a harmonization of melody1 by using the same
notes/duration
; in a different octave from melody1
( defmethod create-octave-harmonization ( ( notes-list list ) )
    (cond
        (( = 0 ( length notes-list ) )
            '()
        )
        (t
            ( setf new-octave ( car ( remove ( note-octave ( car
notes-list ) ) *melody-octaves* ) ) )
            ( setf new-note
                ( make-instance 'note
                    :pitch ( note-pitch ( car notes-list ) )
                    :duration ( note-duration ( car notes-list ) )
                    :octave new-octave
                    :str-representation ( generate-str-representation
                                          ( note-pitch ( car notes-list)
)
                                          ( note-duration ( car
notes-list ) ) )
                                          new-octave
```

```
                                            )
                )
            )
            ( cons new-note ( create-octave-harmonization ( cdr notes-list
) ) )
        )
    )
)
```

## Stepwise Bassline Demo

```
----------------Individual 3----------------
X:1
T:Individual 3
C:Dystopian Tuesday
M:4/4
L:1/4
Q:1/4=120
V:S clef=treble name=S
V:A clef=treble name=A
%%score [ ( S ) ( A ) ]
K:C
V:S
%%MIDI program 0
G/2 D B F/2 G/2 E D2 F2 B2 A E/2 F2 D/2 E2 F2 G B2 A/2 B2
V:A
B, A2, F, A, B2, A, B4, A2, G4, G, F4, E,
NIL
```

## Stepwise Bassline Demo Code

```
; Demo method for generating a bassline with stepwise motion.
( defmethod demo--stepwise-bassline ()
    ( setf melody1 ( generate-melody1 ) )
    ( setf melody2 ( generate-bassline ) )

    ( setf m
```

```
        ( make-instance 'music
            :melody1 melody1
            :melody2 melody2
            :rank 0
            :melody1-rank 0
            :melody2-rank 0
            :num 3
        )

    )

    ( easyabc-display m )
)
```

## Stepwise Pitch Generation Code

```
; Method that generates a list of pitches based on the
; length of duration-list and stepwise movements
( defmethod generate-pitches-stepwise ( ( duration-list list ) previous
    &aux stepwise-l ele previous-position )
    ( setf stepwise-l '( 0 1 1 2 ) )
    ( setf ele ( select-random-arr-element stepwise-l ) )
    ( setf previous-position ( position previous *CMAJOR* ) )
    ( cond
        (( = ( length duration-list) 0 )
            '()
        )
        (( > ( + previous-position ele ) ( - ( length *CMAJOR* ) 1) )
            ( cons ( nth ( - previous-position ele ) *CMAJOR* )
                ( generate-pitches-stepwise ( cdr duration-list ) ( nth (
- previous-position ele ) *CMAJOR* ) ) )
        )
        (t
            ( cons ( nth ( + previous-position ele ) *CMAJOR* )
                ( generate-pitches-stepwise ( cdr duration-list ) ( nth (
+ previous-position ele ) *CMAJOR* ) ) )
        )
```

```
        )


)
```