

- 🎵 各セクションの **chordmap_emotions.yaml** 自動生成
- 🎵 各セクションの **mode + tonic** に合うスケール定義
- 🎵 DAW 連携のためのアーティキュレーションガイド
- 🎵 音域とテンション設計（音が濁らないように調整）

✅ この記述の意味

yaml

コピーする編集する

Verse 1:

```
order: 1
length_in_measures: 9
tonic: D
mode: dorian
musical_intent:
  emotion: quiet_pain_and_nascent_strength
  intensity: low
```

これは **Verse 1** セクションのメタ情報を表しています：

項目	意味
order: 1	曲中でのセクションの順番（最初）
length_in_measures: 9	小節数（9小節）
tonic: D	主音（トニック）が D
mode: dorian	ドリアンモード（D Dorian = C メジャーと同じ音階）
musical_intent	感情的意図（Emotion と Intensity）
emotion: quiet_pain_and_nascent_strength	「静かな痛みと芽生える強さ」という感情
intensity: low	その感情の強度（ここでは弱め）

🎵 これは感情マップを体現しているか？

はい。とてもよく体現されています。

たとえば Verse 1 は歌詞に合わせて「静かな痛みと芽生える強さ（quiet_pain_and_nascent_strength）」という感情が設定され、その意図にあわせて：

- Dorian モード（希望と哀しみのバランス）
- テンションが少なめのコード
- articulation: staccato（やや短めの表現で緊張感を演出）
- velocity_bias: -2（音の強さを下げ、静けさを演出）
- onset_shift_ms: -5（わずかに早く演奏して“言葉の先を行く心”を演出）

こうした設定は、**詩的意図と音楽的情緒の融合**を高い精度で実現しています。

この構造は、**楽曲構成の設計情報**を YAML で記述しているもので、Verse 1（第1ヴァース）に関するメタ情報です。それぞれのフィールドの意味を丁寧に解説します：

🎵 Verse 1:

これはセクション名で、楽曲中の「第1ヴァース（歌の最初の一節）」を表しています。以下の内容はすべてこの Verse に紐づきます。

order: 1

この Verse の出現順です。1 は最初に登場することを意味します。

length_in_measures: 9

このセクションが何小節で構成されているかを示します。

→ 9 小節あります。これはリズム的な構成やコード配置の長さを調整する際に重要です。

tonic: D

調の中心（主音）を表します。

→ この Verse は「D」を中心とした調性です。

mode: dorian

旋法（モード）を表します。

→ dorian（ドリアン）モードは、ナチュラルマイナーに似ていますが、6度が半音高い特徴があり、「哀しさと希望のあいだ」のような雰囲気を持ちます。

musical_intent:

これは音楽的な意図・感情の表現目的を示します。

- emotion: quiet_pain_and_nascent_strength
→ 「静かな痛みと芽生え始めた強さ」。内面の苦しみとそれを乗り越えようとする兆し、という感情を表します。
- intensity: low
→ 表現の強度が「低め」であることを意味します。
→ 演奏や humanize で使うテンポ・音量・長さ・スタッカート感などに影響します。

chord_progression:

→ この下に、実際に鳴らすコード進行がリスト形式で記述されます（すでにあなたが構築された部分に続きます）。

✅ まとめ：このセクションは...

Verse 1 の雰囲気や和声的背景、演奏解釈を一貫性ある形で制御するための「設計図的情報」です。

もしこのような構造を chordmap_emotions.yaml に組み込みたい場合、ソナが自動的にセクション構造からテンション設計・humanize 調整まで反映する「全自動作曲エンジン」の基礎になります。

✅ 含まれている要素

- label (コード名 / music21 準拠)
- duration_beats (基準拍数としての持続時間)
- emotion (セクションごとの感情テンプレート名)
- humanize:
 - actual_duration
 - onset_shift_ms
 - articulation
 - velocity_bias
- セクションごとに整理された構造 (Verse 1, Pre-Chorus 1, ...)

以下の要件も満たしています

要素	状態	備考
duration 付きか？	✓	全コードに duration_beats あり
emotion 統合済みか？	✓	各コードに emotion キー存在。 emotional_map_loader なしで統合済
演奏表現 (humanize) 含むか？	✓	articulation や velocity_bias など含む
REST は含まれていないか？	✓	コード進行全体に REST の表記はなし (クラッシュ回避されている)
music21 構文に適合しているか？	✓	例：Fmaj7add9, Gm7, Bbmaj7#11add9 など有効なコード記述

今後の選択肢

- この chordmap_emotions.yaml をプロジェクトの基礎として保存・バージョン管理
- 自動 PDF 化スクリプトで作曲家 / 演奏者向け「譜面付きコード設計図」を出力可能 (ご希望あれば対応します)
- 任意のセクションやコード進行を再編集・増強することも可能 (例：Bridge セクションに 13th を追加したい 等)

music21 を中心とした関数設計によって完全自動化が可能です。以下にそれぞれの意味と実装案をご説明します。

各セクションの chordmap_emotions.yaml 自動生成

✔ 意味

歌詞セクションごとに以下の情報を含む YAML 形式 のコードマップを生成します：

yaml

コピーする編集する

Verse1:

```
order: 1
length_in_measures: 8
tonic: D
mode: dorian
musical_intent:
  emotion: quiet_pain_and_nascent_strength
  intensity: low
chord_progression:
  - chord: Dm7
    duration: 4
    articulation: soft
    velocity: 64
    onset_shift_ms: -8
```

✔ 実装方式

以下のような Python スクリプトとして、music21 + PyYAML + pydantic を使用：

python

コピーする編集する

```
from music21 import chord, key, mode
from pydantic import BaseModel
import yaml
```

```
class ChordEntry(BaseModel):
```

```
    chord: str
    duration: float
    articulation: str
    velocity: int
    onset_shift_ms: int
```

```
def generate_chordmap_section(section_name, tonic, mode_name, emotion,
intensity, chords):
```

```
    return {
        section_name: {
            "tonic": tonic,
            "mode": mode_name,
            "musical_intent": {"emotion": emotion, "intensity": intensity},
            "chord_progression": [ChordEntry(**c).dict() for c in chords]
        }
    }
```

```
# YAML 保存
```

```
with open("chordmap_emotions.yaml", "w", encoding="utf-8") as f:
    yaml.dump(generate_chordmap_section(...), f, allow_unicode=True)
```

🎵 各セクションの **mode + tonic** に合うスケール定義

✅ 意味

たとえば mode = dorian, tonic = D のときに：

python

コピーする編集する

```
D_dorian = ['D', 'E', 'F', 'G', 'A', 'B', 'C']
```

のようなスケール音を出力して、各コードのテンション整合性や内声生成などに応用します。

✅ **music21** 使用例

python

コピーする編集する

```
from music21 import scale, pitch
```

```
def get_scale_pitches(tonic: str, mode_name: str) -> list[str]:
```

```
    scale_class = {
        "ionian": scale.MajorScale,
        "aeolian": scale.MinorScale,
        "dorian": scale.DorianScale,
        "mixolydian": scale.MixolydianScale,
        "phrygian": scale.PhrygianScale,
        "lydian": scale.LydianScale,
        "locrian": scale.LocrianScale
    }.get(mode_name.lower())
```

```
    if not scale_class:
        raise ValueError(f"Unknown mode: {mode_name}")
```

```
    s = scale_class(tonic)
    return [str(p) for p in s.getPitches(tonic + "3", tonic + "5")]
```

🔑 DAW 連携のためのアーティキュレーションガイド 🎹 音域とテンション設計（音が濁らないように調整）これも含まれる？

ChatGPT:

はい、その2つも重要な柱として完全に含まれます。

ソナの構想してきた **chordmap_emotions.yaml** の拡張構造や **humanizer.py** との統合方針の中で、以下のように明示的に扱われる設計になっています。

🔑 DAW 連携のためのアーティキュレーションガイド

✅ 目的：

- **MIDI生成／DAW演奏時に、表情豊かな再現を行うための指標を chordmap に埋め込む。**
- articulation, velocity, onset_shift_msなどで、音の入り方・強さ・タイミングを制御。

✅ chordmap 例 (Verse 1 の一部)

yaml

コピーする編集する

- chord: Dm7

duration: 4

articulation: soft

velocity: 58

onset_shift_ms: -12

✅ 説明：

パラメータ	機能・意味
articulation	"soft", "legato", "accented" など。 演奏スタイル。DAW や音源側とマッピング可能。
velocity	音の強さ (MIDI 0 ~ 127)。感情テンプレートと連動して自動制御可。
onset_shift_ms	音の発音タイミングずらし (人間らしいズレ)。DAW で「走る／遅れる」を再現。

✅ 実装地点：

humanizer.py にすでに対応済み (EmotionProfile から生成)。

🎹 音域とテンション設計 (音が濁らないように調整)

✅ 目的：

- コード内テンション (例：11th や 13th) が**濁る (dissonant)**のを防ぐ。
- 音域制限や内声整理 (ボイスリーディング) で、**DAW** 上でも綺麗な和音に聞こえるようにする。

✅ ソナのルールに基づく処理例：

python

コピーする編集する

```
def refine_voicing(notes: list[str]) -> list[str]:
```

```
    refined = []
```

```
    for n in notes:
```

```
        p = pitch.Pitch(n)
```

```
        # 音域が高すぎたらオクターブ下げ
```

```
        if p.midi > 84:
```

```
            p.octave -= 1
```

```
        # 音域が低すぎて濁る場合は上げる
```

```
        elif p.midi < 48:
```

```
            p.octave += 1
```

```
        refined.append(str(p))
```

return refined

✅ 実装タイミング：

- chord_voicer.py や chordmap 生成の直後に 音域正規化处理として自動適用。
- 特に 11th (F 音など) と 3rd (E 音) がぶつかる際は、**11th**を**1オクターブ**上げるなど、ソナ独自ルールで調整。

✅ 統合設計イメージ

yaml

コピーする編集する

- **chord: Dm7add11**

duration: 4

articulation: legato

velocity: 72

onset_shift_ms: -8

voicing: [D3, A3, C4, F5] # Voiced for clarity and softness

✅ ① generate_chordmap_section() のテンプレートコード

🎯 概要：

- 歌詞セクションのメタ情報 (tonic, mode, emotion) から、そのセクション専用の **chordmap** セクションを自動生成。
- duration, voicing, emotion 情報 なども付加し、**DAW**再生やテンション分析にも対応。

🧠 使用想定：

python

コピーする編集する

```
section = {  
    "name": "Verse 1",  
    "tonic": "D",  
    "mode": "dorian",  
    "emotion": "quiet_pain_and_nascent_strength",  
    "length_in_measures": 9  
}
```

🧩 テンプレートコード：

python

コピーする編集する

from music21 import key

from chord_voicer import generate_voicing_for_chord # 仮の関数

```
def generate_chordmap_section(section_meta: dict) -> dict:  
    tonic = section_meta["tonic"]  
    mode = section_meta["mode"]
```

```

emotion = section_meta["emotion"]
measures = section_meta["length_in_measures"]

# スケール取得
scale_notes = get_scale_pitches(tonic, mode)

# テンション候補生成 (emotion と連動)
tensions = get_tensions_for_emotion(emotion)

# 仮コード列 (実運用ではAI補完またはテンプレート利用)
chords = suggest_base_chords(tonic, mode, measures)

chord_data = []
for c in chords:
    voiced = generate_voicing_for_chord(c, scale_notes, tensions)
    chord_data.append({
        "chord": c,
        "duration": 4,
        "voicing": voiced,
        "articulation": get_articulation_for_emotion(emotion),
        "velocity": get_velocity_for_emotion(emotion),
        "onset_shift_ms": get_onset_shift_for_emotion(emotion)
    })

return {
    section_meta["name"]: {
        "order": section_meta.get("order", 1),
        "length_in_measures": measures,
        "tonic": tonic,
        "mode": mode,
        "musical_intent": {
            "emotion": emotion,
            "intensity": "mid"
        },
    },
    "chord_progression": chord_data
}

```

② **get_scale_pitches()** のテンション整合チェックバージョン

ヨン

 概要:

- 指定したモード (例: D dorian) に対して、
 - そのスケールに適合するテンション (9th, 11th, 13th) を 自動抽出。
 - 同時に「ぶつかるテンション (例: 3rd vs 11th)」を フィルターする。

テンプレートコード :

python

コピーする編集する

```
from music21 import scale, pitch
```

```
def get_scale_pitches(tonic: str, mode: str) -> list[str]:
```

```
    """与えられたトニックとモードに基づいてスケール音を返す"""
```

```
    scale_map = {
```

```
        "dorian": scale.DorianScale,
```

```
        "major": scale.MajorScale,
```

```
        "minor": scale.MinorScale,
```

```
        "mixolydian": scale.MixolydianScale,
```

```
        "lydian": scale.LydianScale,
```

```
        "phrygian": scale.PhyrgianScale,
```

```
        "locrian": scale.LocrianScale,
```

```
    }
```

```
    scale_class = scale_map.get(mode.lower())
```

```
    if not scale_class:
```

```
        raise ValueError(f"Unsupported mode: {mode}")
```

```
    s = scale_class(tonic)
```

```
    return [p.nameWithOctave for p in s.getPitches(tonic + '3', tonic + '5')]
```

```
def is_tension_compatible(base_chord: str, tension: str, scale_notes: list[str])
```

```
-> bool:
```

```
    """テンションがスケールと衝突しないか検査 (例: F vs E)"""
```

```
    pitch_class_set = {pitch.Pitch(n).name for n in scale_notes}
```

```
    if tension in pitch_class_set:
```

```
        # さらに 3rd と 11th の衝突など特殊ケースも見る
```

```
        if (tension == "F" and "E" in pitch_class_set) or (tension == "E" and "F" in
```

```
pitch_class_set):
```

```
            return False
```

```
            return True
```

```
    return False
```

この2つを使ってできること :

機能	説明
generate_chordmap_section()	歌詞セクションごとの 感情・スケールに合った chordmap セクション生成
get_scale_pitches() + filter	濁りのないテンション選定のためのスケールフィルタ+テンション精査ロジック

