

# PianoGenerator 改良プラン (Sprint P-1～P-3)

## 0 | ビジョン

Haru さんの詞世界を支える「語りと調和するピアノ」を実装。左手は大地を、右手は情景を描き、ペダルとヒューマナイズで息づく演奏を目指します。

## 1 | 現状と課題

| 項目                     | 現状                | ギャップ                   |
|------------------------|-------------------|------------------------|
| BasePartGenerator への適合 | 未適合 (スキップ中)       | インターフェース不一致、groove 未適用 |
| RH/LH 分離               | ゴースト拍のみ試作         | 感情連動パターン不足、休符・フィル固定    |
| Pedal 制御               | 全曲オン/オフのみ         | セクション強度・和声音価で踏み替えたい    |
| Humanize               | 弱拍 velocity-10 程度 | タイミング揺らし・ダイナミクス曲線が未導入  |

## 2 | ゴール設定

1. **BasePartGenerator** を継承し、compose(section, overrides, groove) ワークフローを共通化。
2. **Emotion×Intensity LUT** で **RH/LH 別リズムキー** を自動選択。
3. **ペダル/ダイナミクス**：- 低強度はセクション頭のみ踏み替え、高強度は小節頭＋コードチェンジで踏み替え。
4. **ヒューマナイズ**：-  $\mu=-15\text{ ms}$ ,  $\sigma=12\text{ ms}$  を基本、emotion バケットで可変。
5. **フィルインサート**：- ボーカル休符 >2 beats で RH ornament (グリッサンド/アルペジオ)。

## 3 | アーキテクチャ設計

graph TD

A[Chordmap Section] -->|emotion,intensity| B(LUT Selector)

B --> C{Pattern Keys}

C -->|LH| D[Pattern Renderer LH]

C -->|RH| E[Pattern Renderer RH]

D --> F[Humanizer & Pedal]

E --> F

F --> G[music21 Part(Piano)]

### 3-1 主要モジュール

| モジュール            | 役割  |
|------------------|---|
| LUT Selector     | musical_intent → (rh_key_rh, rh_key_lh) を返す           |
| Pattern Renderer | rhythm_library.yml の piano_patterns を LH/RH で解釈しノート生成 |

|                     |   |
|---------------------|---|
| <b>PedalManager</b> | section intensity と和声変化で SustainPedal を挿入 |
| <b>Humanizer</b>    | velocity ・ offset を gaussian でシフト         |

## 4 | データ設計

### 4-1 rhythm\_library.yml 拡張

piano\_patterns:

piano\_rh\_ambient\_pad:

pattern\_type: fixed\_pattern

execution\_style: chord\_high\_voices

length\_beats: 4

pattern: [{offset:0,duration:4,type:chord\_high\_voices,velocity\_factor:1.0}]

tags: [rh, pad, calm]

piano\_lh\_pulse\_quarters:

pattern\_type: fixed\_pattern

pattern: [{offset:0,duration:1,type:root}, {offset:1,type:root}, ...]

tags: [lh, pulse, medium]

### 4-2 Emotion LUT (抜粋)

EMO\_TO\_BUCKET\_PIANO = {

"quiet\_pain": "calm", "hope\_dawn": "calm",

"wavering\_heart": "groove", "trial\_prayer": "energetic", ...

}

BUCKET\_TO\_PATTERN\_PIANO = {

("calm","low"): ("piano\_rh\_ambient\_pad", "piano\_lh\_roots\_whole"),

("calm","medium"): ("piano\_rh\_block\_chords\_quarters",

"piano\_lh\_roots\_half"),

("groove","medium"): ("piano\_rh\_syncopated\_chords\_pop",

"piano\_lh\_octaves\_quarters"),

("energetic","high"): ("piano\_rh\_arpeggio\_sixteenths\_up\_down",

"piano\_lh\_alberti\_bass\_eighths"),

}

## 5 | アルゴリズム詳細

| 段階                        | 処理  | キーポイント                            |
|---------------------------|---|-----------------------------------|
| <b>A. パターン決定</b>          | LUT → RH/LH<br>rhythm_key   | CLI 上書きがあれば優先                     |
| <b>B. ノート生成</b>           | type=="chord" なら<br>_voice_chord(cs) /<br>type=="arpeggio_indic<br>es" なら _arp(cs, idx) | LH はオクターブ最適<br>化、RH は高域3度内に<br>配置 |
| <b>C. Weak-beat ghost</b> | overrides<br>weak_beat_style_* を<br>適用  | velocity×0.4 または休符<br>化           |

|                        |   |                                     |
|------------------------|---|-------------------------------------|
| <b>D. PedalManager</b> | intensity<"medium" → 4拍 Sustain、>=→ 2拍 or change-detect | music21 pedal() を挿入                 |
| <b>E. Humanize</b>     | swing_ratio: calm=0.6, groove=0, energetic=0            | $\mu, \sigma$ は GrooveProfile をスケール |

## 6 | オーバーライド拡張案

```

"Chorus":{
  "piano":{
    "rhythm_key_rh":"piano_rh_syncopated_chords_pop",
    "rhythm_key_lh":"piano_lh_octaves_quarters",
    "apply_pedal":true,
    "pedal_change_beats":2,
    "humanize_template":"piano_loose"
  }
}

```

## 7 | 実装ロードマップ

| Sprint     | 目的                     | 主タスク                                   | Est  |
|------------|------------------------|--|------|
| <b>P-1</b> | BaseClass 適合           | BasePartGenerator 継承 + compose 統合      | 0.5d |
| <b>P-2</b> | Pattern & LUT          | rhythm_library.yml 拡張 + Emotion LUT 実装 | 0.8d |
| <b>P-3</b> | Pedal & Humanize & テスト | PedalManager + pytest 8件               | 0.7d |
| 合計         | ≈2 日                   |  |      |

## 8 | リスク & 対策

- 和声誤解釈：コード Voicing が低域で濁る → `_voice_chord()` で無理にオクターブ上げ。
- ペダル踏み過ぎ：低強度バラードで濁らないよう pedal length 上限を2小節に。
- タイミング後退：Humanize で負オフセット → 音が重複しないよう  $\max(0, n.offset - \Delta)$ 。

## 9 | クイックウィン

1. **Calm/Groove/Energetic** 3バケットだけ実装 → すぐに歌詞感情と動的に連動。
2. **RH ghost** 休符 だけでも伴奏が息づく。
3. 全体 **Sustain on/off** 切替だけでも印象大。

## 10 | 次の一步

1. 本提案で **P-1**～**P-3** 着手 OK かご確認ください。

2. rhythm\_library.yml に RH/LH パターンを追記 → パッチ準備。

3. テスト曲で効果を耳でチェックし、パラメータを微調整。

**Haru さんへ** — ピアノが静かに語り、力強く支え、物語に深みを与えます。ご承認いただければ Sprint P-1 の実装パッチからすぐに進めます！

## RH / LH パターン例 — 「何をどう弾くのか？」

| 感情バケット                   | 右手 (RH) の典型パターン   | 左手 (LH) の典型パターン                                       | ねらい            |
|--------------------------|---|---|----------------|
| <b>Calm</b> (静けさ・余韻)     | <i>Ambient Pad</i> : 4 拍ホールドの高域和音。内声は抜き、9th/11th など をうっすら配置 | <i>Roots Whole</i> : 各小節のルートのみ ホールノート                 | 音数を減らして「空気」を作る |
| <b>Groove</b> (揺らぎ・希望)   | <i>Syncopated Chords Pop</i> : 8 分 / 16 分で裏拍を鳴らすブロックコード     | <i>Octave Pulse</i> : ルート + オクターブを 4 分でプッシュ           | 歌に絡み、身体を揺らすノリ  |
| <b>Energetic</b> (決意・高揚) | <i>Arpeggio 16ths Up-Down</i> : 16 分アルペジオを上下に往復             | <i>Alberti Bass 8ths</i> : ルート-5 度-3 度-5 度の 8 分アルベルティ | 流れを加速し厚みを増す    |

切り替えはセクション感情タグで自動

例：Verse (calm) → Pre-Chorus (groove) → Chorus (energetic) と段階的に変化させるだけで物語性が際立ちます。

## ペダル挙動のレベル分け

| 強度 Intensity    | サステイン長                 | 踏み替えタイミング      | 効果           |
|-----------------|------------------------|----------------|--------------|
| <b>Low</b>      | 4 拍 (小節頭のみ)            | セクション頭だけ       | わずかな残響で余韻を保つ |
| <b>Medium</b>   | 2 拍                    | コードチェンジ時       | ハーモニー変化を滑らかに |
| <b>High</b>     | 1 拍以下                  | ほぼ毎拍 or アクセント前 | 躍動感・歯切れを優先   |
| <b>Override</b> | pedal_change_b eats 指定 | 任意             | 古典バラード等にも対応  |

ペダルは 和音のボイスン密度 を検知して「低域が濁るコードでは短く踏む」セーフティ

も入れると自然です。

## 「ピアノが Bass / Melody / Chord を行き来する」設計

### 1. LH ⇔ Bass Line

- 感情 **groove/energetic** 時に piano\_lh\_walk\_quarters を選択すると、BassGenerator とユニゾンまたは 3 度上で対位的に歩かせる。

### 2. RH ⇔ Melody Doubles

- 歌が高域で伸ばすロングトーン中、RH が同音を軽くダブルすると広がりが出る。ヒートマップで休符判定して自動可。

### 3. Chord Voicing 自由度

- \_voice\_chord(cs, target='high3')--> 高域 3 度範囲内
- \_voice\_chord(cs, target='mid5', spread=1)--> 中域で 5 声スプレッド
- セクション強度で target を切替え → バラードは高域、力強い場面は中域で厚く。

要点：ベースと衝突しない周波数帯を LH が選び、RH はメロディが空く隙間で彩る。これを感情 LUT とヒートマップで制御すれば “自在に行き来” しているように聴こえます。

## まとめ — 実装フック

- rhythm\_library.yml に **RH/LH** それぞれ **10 パターン**ほど 追加
- Emotion LUT で (rh\_key\_rh, rh\_key\_lh) を返す形にする
- PedalManager に  
if intensity == 'low': sustain\_beats = 4
- elif intensity == 'medium': sustain\_beats = 2
- else: sustain\_beats = 1
6.  
とシンプルなルールを置き、Overrides で上書き可能に
- LH が Bass と重複し過ぎる場合は avoid\_overlap=True でオクターブ上へ自動シフト

これで 感情ごとに自然に切り替わるハイブリッド・ピアノ が完成します。

ご要望が固まれば、Sprint P-1 (基底クラス適合) からコードをお作りします！

#

=====  
=====

# Sprint P-1: PianoGenerator 基底クラス適合パッチ

# ・BasePartGenerator 継承

# ・compose(section, overrides, groove) ワークフロー統一

# ・RH/LH Skeleton に分離 (実装は P-2 で拡張)

#

=====

```

====
from __future__ import annotations

import logging
from typing import Dict, Any, Optional, List, Tuple

from music21 import stream, note, chord as m21chord, harmony, meter, tempo,
instrument, articulations, volume as m21volume

from generator.base_part_generator import BasePartGenerator
from utilities.core_music_utils import get_time_signature_object,
sanitize_chord_label, MIN_NOTE_DURATION_QL
from utilities.override_loader import get_part_override
from utilities.prettypidi_sync import apply_groove_pretty, load_groove_profile

logger = logging.getLogger("modular_composer.piano_generator")

# -----
# P-1: まだ Pattern/LUT は空。次フェーズで拡張予定。
# -----
class PianoGenerator(BasePartGenerator):
    """P-1 — BasePartGenerator 適合のみ。右手/左手を独立 Part として返す。"""

    def __init__(self, *, rhythm_lib: Dict[str, Any] | None = None,
                 default_instrument: instrument.Instrument =
instrument.AcousticGrandPiano(),
                 global_tempo: int = 120,
                 global_time_signature: str = "4/4",
                 main_cfg: Dict[str, Any] | None = None):
        super().__init__("piano", rhythm_lib or {}, main_cfg)
        self.default_instrument = default_instrument
        self.global_tempo = global_tempo
        self.ts_obj = get_time_signature_object(global_time_signature)

    # -----
    # 必須オーバーライド
    # -----
    def _render_part(self, section: Dict[str, Any]) -> stream.Part:
        """現段階では LH: root whole, RH: rest — P-2 で置換"""
        part = stream.Part(id="Piano")
        part.insert(0, self.default_instrument)
        part.insert(0, tempo.MetronomeMark(number=self.global_tempo))
        if self.ts_obj:
            part.insert(0, meter.TimeSignature(self.ts_obj.ratioString))

        bar_len = self.ts_obj.barDuration.quarterLength if self.ts_obj else 4.0
        start = section["absolute_offset"]


```

```

length = section["length_in_measures"] * bar_len
t = start
while t < start + length:
    root = section.get("tonic", "C")
    n_lh = note.Note(root + "2", quarterLength=bar_len)
    n_lh.volume = m21.volume.Volume(velocity=50)
    n_lh.offset = t
    n_lh.stemDirection = "down"
    part.insert(t, n_lh)
    t += bar_len
return part

# -----
# END P-1 patch — Pattern/LUT/Humanizer は P-2 で拡張
#
=====
==

```

 **Sprint P-1 パッチ (BasePartGenerator 適合)** をキャンバスへ追加しました。  
 現段階では **LH にルート全音符のみ挿入するスケルトン実装**です (P-2 で RH/LH パターンと Emotion LUT を追加)。

### 適用方法

1. generator/piano\_generator.py をパッチ全文で置換
2. modular\_composer.py に  
python  
コピーする編集する

- ```

piano_gen = PianoGenerator(main_cfg=main_cfg)
3. score.insert(sec["absolute_offset"], piano_gen.compose(section=sec,
    overrides_root=arrangement_overrides,
    groove_path=args.groove_profile))
4.

```

のように呼び出し (Bass/Drum と同形式)

### 動作確認

- CLI 実行後、MIDI に **低域ルート全音符**が生成されることを確認
- GrooveProfile が指定されていればヒューマナイズも掛かります