“歌に呼応して揺れ・表情を付けるエモーショナル・ドラム” へ進化させること。
Codex には 具体的なフィールド名・関数名・型ヒント・テスト項目 **を与えると精度が大幅アップするので、上のテンプレをベースにカスタマイズして投げてみてください。**

これから出来る “人間越え” ブラッシュアップ案
Codex への指示キーワード

マイクロタイミング
Push/Snag 曲線を感情別に切替（例：悲しい＝後ノリ 80 ms、怒り＝前ノリ 40 ms）
humanizer.py::apply_swing 拡張 or 新 apply_push_pull(part, curve)
“dynamic push-pull curve by emotion”

フラム／ドラッグ／ラフ
"type": "flam" / "drag" / "ruff" を pattern に許可し、挿入時に複数 grace hit を合成
drum_generator.py::_apply_pattern
“insert grace hits 20-40 ms earlier with reduced velocity”

HH アーティキュレータ
"hh_open"／"hh_pedal" を GM_MAP に追加。ボーカルの語尾長さを見てオープン ↔ クローズを自動切替
drum_generator.py::_make_hit, AccentMapper
“open HH when vocal note length < 200 ms”

モチーフ拡張フィル
fill_ins に template: tom_run と length: 1-4 beats などを書ける API。自動でトライアド上昇 or 降下
FillInserter.insert
“generate tom-run ascending to crash”

ブラシ⇄スティック切替
override_loader.PartOverride.drum_brush: bool を参照し BRUSH_MAP にスイッチ
drum_generator.__init__
“if brush mode, use soft samples”

ベロシティ・ランダムウォーク
同一パターン中でも徐々に ±8 程度ランダムに揺らす
AccentMapper.get_velocity
“per-bar rand walk amplitude”

ビッグデータ学習
GroovePool (MIDI ループ集) から n-gram / VAE でゴースト配置を生成
新 groove_sampler.py
“sample next-hat position via Markov”

ボーカル子音同期

Essentia / Librosa で sibilant(s/k/t) を検出 → クローズHHを被せる
新 vocal_sync.py
“align closed-hat to detected consonants”

---

mido・Essentia・Librosa活用アイデア

①mido

リアルタイム試聴／外部 DAW への仮想ポート出力
mido.open_output('DrumPreview') で生成ノートを即時送信し、人手でタイムスタンプを検証

②Librosa

WAV からメロディライン & ボーカル強勢を推定
librosa.pyin → 音高曲線, librosa.beat.beat_track → ビート揺れ解析

③Essentia

ムード/エネルギーレベル推定 → ドラム強度へ反映
essentia.standard.Danceability() で intensity 算出、pattern_base_velocity を係数でスケール

📊 “現状 ⇄ ロードマップ” アップデート

項目	実装状況	実装ファイル / 関数	既に出来ていること	ギャップ & 次アクション
Push / Snag（感情別プッシュプル）	✅ α版	drum_generator._apply_push_pull	感情キー → self.push_pull_map を読んで 1 beat 単位でシフト	128 分音符グリッドへの補間・apply_swing との合成が未実装 → apply_push_pull(part: Part, curve: list[float], subdiv:int=8) を humanizer へ昇格し、swing と加算合成

Flam	✅	_insert_flam() + type:"flam"	Grace + 本打ち、Vel40 % / 100 %	フラム距離を grace_ms フィールドで可変化
Drag / Ruff	✅ *NEW*	_insert_grace_chain() + type:"drag"/"ruff"	drag=2hit, ruff=3hit、Vel 減衰カーブ	grace 距離ランダム化、左右スティック交互でパン振り
HH open / close auto	✅ 簡易	_sync_hihat_with_vocals	4 拍目前後の語尾で chh→ohh 1 発置換	語尾長さしきい値で open↔close レベル段階制御、子音検出との統合
HH edge / pedal	✅	drum_map_registry + _apply_pattern	Vel<50 → edge、pedal:true → pedal	edge⇄tip ランダム交互、pedal ニュアンス (grace 1 発＋短い close)
ブラシ⇄スティック切替	✅ *β*	PartOverride.drum_brush → _apply_pattern	snare→snare_brush、Vel×0.6	Ride/Hat のサンプル切替、スウィープ 32 連打生成
Velocity ランダムウォーク	❌	—	—	AccentMapper.start_bar() で bar ごとに self.walk=randrange(-8,+8) → hit ごと ±1 ステップ減衰
Tom-run DSL フィル	❌	—	—	fill_ins.*.template=="tom_run" なら generate_tom_run(root:str, length:int) で自動展開

GrooveSampler (n-gram)	❌	—	—	groove_sampler.sample_next(offset, instrument) -> dict を作り、空パターン時 fallback
ボーカル子音同期	❌	—	—	vocal_sync.detect_consonants(wav) -> list[float] → _apply_pattern で chh を被せる
ベロシティ曲線プリセット	⭕	resolve_velocity_curve()	`"crescendo"	"decrescendo"

🎯 次の“伸びしろ” 5 ステップ

Step	実装ガイド	型ヒント / 関数シグネチャ	最小テスト例
1. Velocity Random-Walk	1小節冒頭で乱数を発生 → 各打点ごと ±step 減衰して加算	AccentMapper.begin_bar(walk_range:int=8) -> NoneAccentMapper.get_velocity(rel:float, base:int, *, step:bool=True) -> int	1bar=16 発 → max(base±8) 以内に収まるか
2. Push-Pull × Swing 合成	apply_push_pull(part, curve, subdiv) を humanizer へ昇格し、apply_swing 後に加算。curve は拍単位 →128 分音符線形補間。	apply_push_pull(part: stream.Part, curve: list[float], subdiv:int = 8) -> None	感情 “sad” カーブ [-80,...] → Note.offset が平均 +80 ms 以内

3. Tom-run DSL	fill_ins: tom_run_up: {template: "tom_run", direction:"up", length:2} FillInserter.insert で generate_tom_run() 呼び出し	generate_tom_run(root:int, direction:str, length_beats:float) -> list[dict]	DSL 入力 → 連続 tom1→tom2→tom3→crash が生成される
4. HH 子音同期	vocal_sync.py で librosa 片準メソッド：detect_consonants(wav_path, sr=22050, thresh=0.2) -> list[float]	_sync_hihat_with_consonant(part, hits:list[float], window:float=0.05)	d/t/s 音がある位置 ±50 ms に chh が存在
5. GrooveSampler 2-gram	groove_sampler.load_pool(dir)->None で MIDI 読み込み → (offset, inst) シーケンス頻度辞書。sample_next(prev_two:tuple)->dict	GrooveSampler.sample_bar(style:str, length:float) -> list[dict]	GM Pool12 曲 → サンプル生成 → 重複率 <30 %