A GM SYNTH WITHOUT SAMPLES

サンプルを持たない GM 音源

188 音色を、耳だけで合わせないために。

libsonare に内蔵した GM/GS 音源を、

測って近づけ、耳で決めるまでの手順を話します。

libraz ／ Music Tech Meetup / 音楽技術勉強会 2026.10

NO SOUND INSIDE

MIDI ファイルには、音が１つも入っていない

01

入っているのは「何番の音色で、どの鍵盤を、どの強さで」という番号と指示だけです。音を作るのは、それを受け取った音源です。

MIDI ファイル

音色　1
鍵盤　60
強さ　100

→

音源

ふつう：サンプル（録音）を鳴らす

今日の音源：
サンプルを持たず、計算で作る

→

音

ここで
初めて
音になる

同じファイルでも、音は**受け取った音源**で決まります。今日はその音源の側の話です。

#

WHO

書いている人



libraz

Tokyo ／ libraz.net

github.com/libraz

本業

2001 年からソフトウェアエンジニア。現在は **認証認可（OIDC）と決済まわり**。

作ったもの

libsonare ― 音声解析とリアルタイムオーディオ処理を扱う C++ ライブラリ。同じエンジンを、ブラウザ（**WASM**＝WebAssembly）・Node・Python から叩けます。

libcantus ― 和声と音律を扱う TypeScript ライブラリ。コードを組み、調を判定し、声部を書く。音そのものは扱いません。

sonare studio ― 上の二つで組んだ、**ブラウザだけで動く DAW**。インストール不要です。

作り方

コードは **AI と一緒に**書いていて、たくさんのプロジェクトを並行して進めています。

ほかに

アイドル事務所を運営しています。**音楽系 OSS を書いている理由は、だいたいこれです。**

この資料で

GM/GS の音色を物理モデルで鳴らす過程を話します。プラグイン開発の内側ではなく、**ブラウザ・測定・理論の側**から。ピアノ以外の音の出来は話しません。

だから今日話すのは、コードの細部より、**判断と、その確かめ方です。**

LIBSONARE AT A GLANCE

libsonare は、音をデータに、データを音に変える

02

C++17 で書いた 1 つのコアが、解析から鳴らすところまでを受け持ちます。同じコアが、ブラウザ・Python・Node・ネイティブで動きます。

楽曲解析	編集・採譜	ミキシング	マスタリング
MIDI と内蔵音源	リアルタイム再生	DAW の土台	部屋の響き
ブラウザ（WASM）	Python	Node.js	ネイティブ・CLI

この資料の話　libsonare の領域　動く場所

触れる入口は **libsonare.libraz.net** のデモです。ブラウザの中で動き、音声はアップロードされません。

BYTES OR CYCLES

ブラウザでは、サンプルを同梱する代わりに計算で鳴らす

03

音源モジュールも、VST などのプラグイン音源の多くも、サンプルを抱えて音色を増やしてきました。ブラウザでは、同梱したサンプルが聞く前のダウンロードになります。そこで、サンプルを同梱せず、その分を CPU の計算で払う道を選びました。

音源モジュール、VST などのプラグイン

本体やディスクにサンプルを置く

音色 × 強さの段 × 品質で増える

ギガバイト級まで増やせる

ブラウザ

同梱した分が、聞く前のダウンロードに

だから、サンプルは同梱しない

代わりに、計算で鳴らす

物理モデルと波形合成で、音を計算する

サンプルの音色集（SoundFont）が無くても、188 音色すべてが必ず鳴ります。**引き換えに、CPU の時間を使います。**

NAMES TRAVEL AS NUMBERS

ブラウザでは、音の指定もファイルの中の番号で運ぶ

04

鳴らし方が決まっても、「どの音で、どう鳴らすか」を伝える手段が要ります。ブラウザには音源プラグインを持ち込めないので、それをファイルの中の番号で運びます。



GM（「1 番はピアノ」）と、Roland が広げた **GS** は、**番号で音や動きを決めた約束**です。90 年代の曲の多くも、これを前提に書かれています。

#

188 VOICES

応える音色は 188。GM の全部と、よく使われる GS

05

GM が約束するのは、128 の音色とドラム 1 セット。この音源は、その全部と、よく使われる GS の音色に応えます。

188 音色の内訳

＋ ドラム 1 キット ↓

GM　128 音色（全プログラム）

GS　59 バリエーション

物理モデルにしなかった音も入っています
合唱・声 3 ／ 効果音 8（拍手・波など）

独自の音を持つものだけ
ほかの番号は親の音色で鳴る

ドラム：基本の 1 キットを元に、太鼓ごとに音を差し替えて、GS の 26 セットを作る

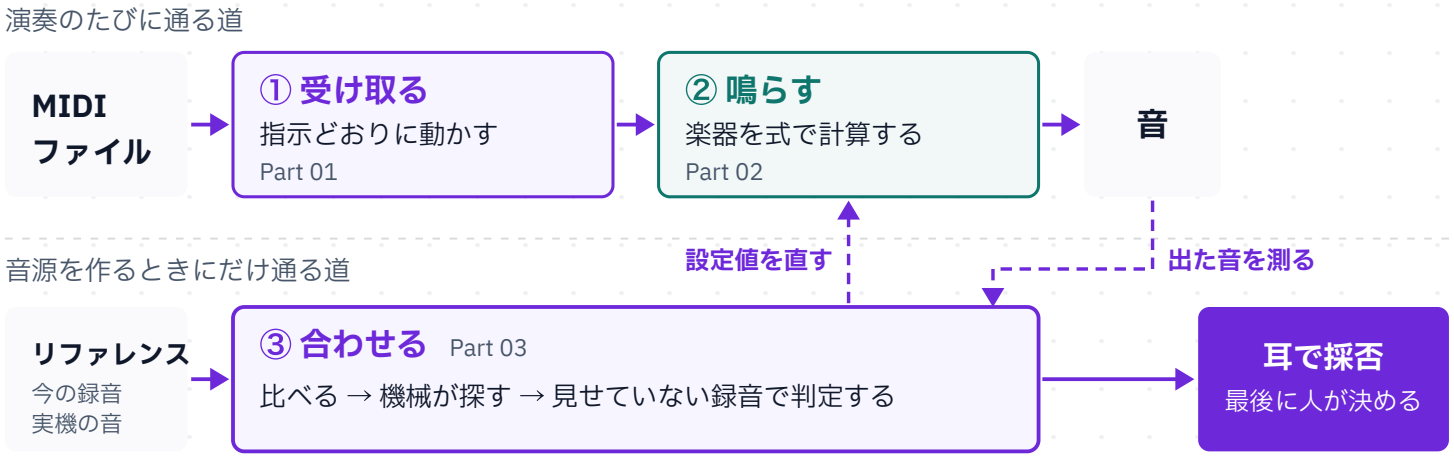
GS の基準にしている実機 SC-8850 は 1,640 音色。4 世代の音色表を丸ごと抱えた数です。
ここでは、GM の 128 全部と、よく使われる GS の音色を合わせた 188 を、計算だけで鳴らします。

THE WHOLE PICTURE

今日の全体像。受け取る、鳴らす、合わせる

06

この音源は、3 つの部分でできています。今日は、この順に 1 つずつ見ていきます。



どの部分も、**耳を使うのは最後だけ**になるように組んでいます。

#

Part 01 · ① 受け取る

GS 互換とは、何を守ることか

MESSAGES ARRIVE · PARAMETERS MOVE · THE MACHINE DECIDES

THE COMPATIBILITY CONTRACT

GS 互換で約束するのは、音ではなく、指示どおりに動くこと

07

たとえば「音を明るく」という指示が届いたら、明るくなる。変わる向きと大きさは、マニュアルに書かれたとおり。
約束するのは、そこまでです。

届いた指示（例）	約束すること	音の似かたは
「音を明るく」 SysEx：番地と値	→ **書かれた向きと量で、明るくなる**	**③ 合わせる（Part 03）で**

対応する仕組みが無い指示も、近い動きに届ける（黙って捨てない）

確かめるのは、値を動かした向きに、おおよそ見合う大きさで音が変わることです。**これは耳ではなく、測って確かめられます。**

#

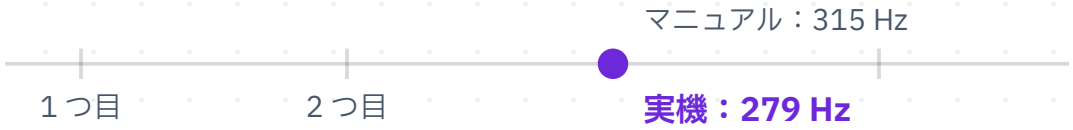
THE MACHINE DECIDES

実機を測ると、マニュアルと違う所、書いていない所があった

08

約束は「マニュアルどおり」。実機のエフェクトを SysEx で動かして測ると、違う所と書かれていない所がありました。どちらも実機に合わせます。

エフェクトの周波数の目盛り（3 分の 1 オクターブ刻み）　40 03 07 ／ Stereo-EQ (01 00) M1 Freq



回転スピーカーの、遅い・速いの切り替え（値は 0〜127）　40 03 0D ／ Rotary (01 22) Speed



- 測ったのは**反応**で、音色ではない。実機は 1 台だけ。
- 事実と読みを分けて **soundings.libraz.net** で公開。

事実と読みを分けておけば、**読みに異論がある人も、測定はそのまま使えます。**

#

Part 02 · ② 鳴らす

楽器を、
どう計算して鳴らすか

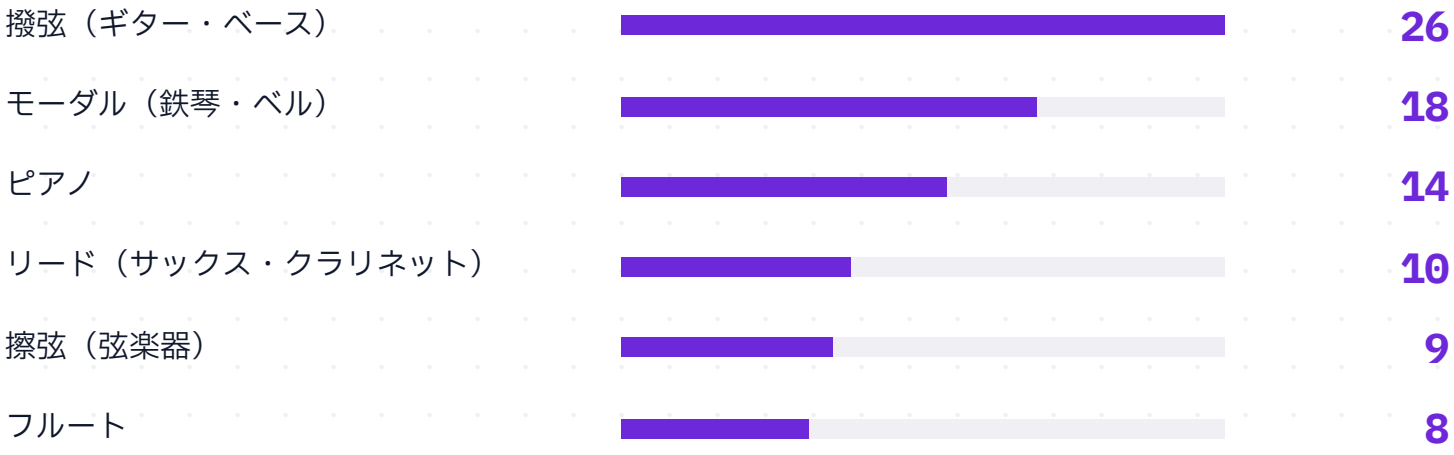
PLAYING INSTRUMENTS BY PHYSICS

PHYSICAL MODELS: 106 VOICES

物理モデルで鳴らすのは 106 音色。楽器の振動を式で計算する

09

楽器の種類ごとに「エンジン」を作り、音色にはそれぞれの設定を渡しています。16 のエンジンのうち、楽器の振動を計算する物理モデルは 10。多い順に 6 つ。



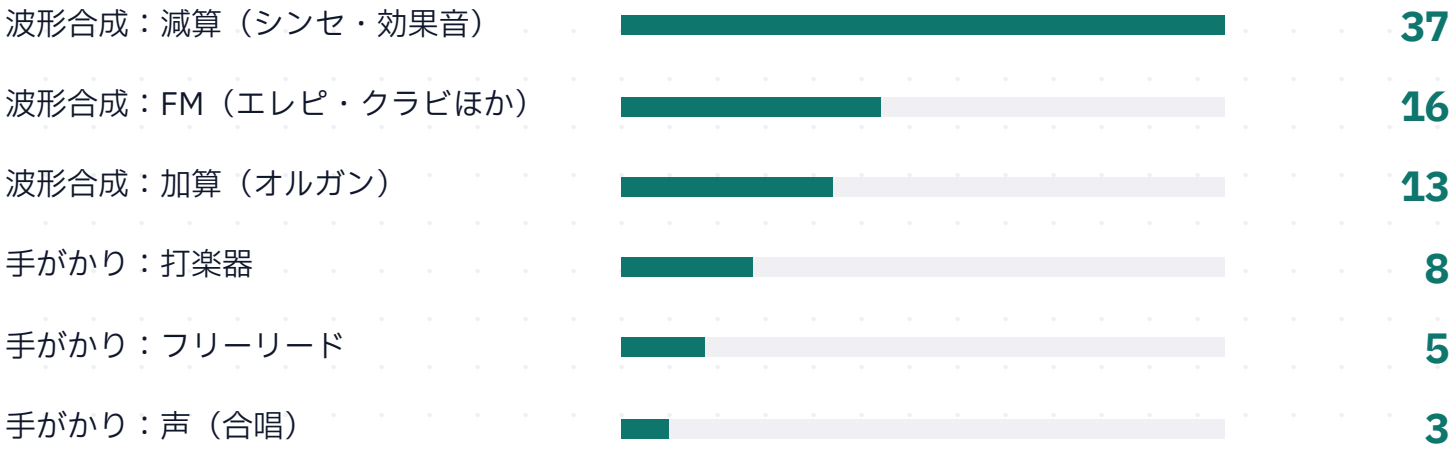
ほかにチェンバロ・金管・パイプオルガン・弾弦。**ピアノの 14 音色は、1 つのエンジンを共有しています。**

THE REST: BUILT, NOT SIMULATED

残りの 82 音色は、振動を追わずに組み立てる

10

振動の往復は計算せず、波形や倍音を直接組み立てる波形合成が 66 音色。打楽器や合唱のように、物理を手がかりにして形だけ作る音が 16 音色です。



188 音色の全部を、物理で計算しているわけではありません。**拍手や合唱は、それらしく作る音です。**

Part 02 · ② 鳴らす · 物理モデルとは

揺らす部分と、響く部分。その動きを、式で追う

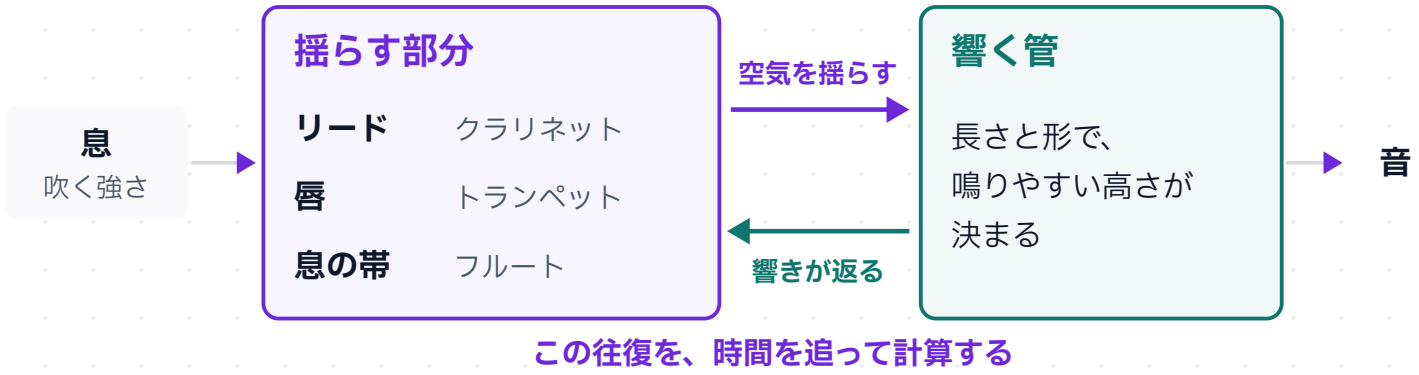
EXCITER × RESONATOR · WIND, BOW, PLUCK, PIANO

WIND: EXCITER AND BORE

管楽器は、揺らす部分と響く管の往復で鳴る

11

まず管楽器。息で揺れる部分が管の空気を揺らし、管の響きがその揺れ方に返ってきます。高さはおもに管で決まり、揺らす部分が少し動かします。



どちらか片方だけでは、管楽器の音になりません。**あとで「足りない仕組み」を足すのも、この往復の中です。**

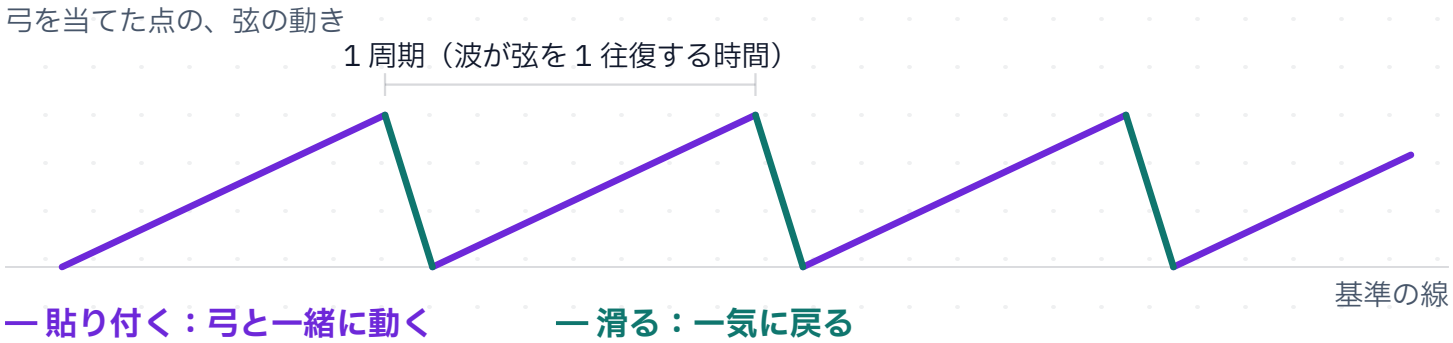
#

BOW: STICK AND SLIP

弓は、弦に貼り付いては滑る。擦弦はその繰り返し

12

次は弦。弓の毛と弦の間は、くっついては滑ります。波が弦を行って戻る時間に合わせて、この切り替わりが繰り返されると、安定した弓の音になります。



- 安定した音になるかどうかは、弓の力・速さ・当てる位置で決まります。
- 弓の毛の幅も扱えます（一部の音色で使用）。

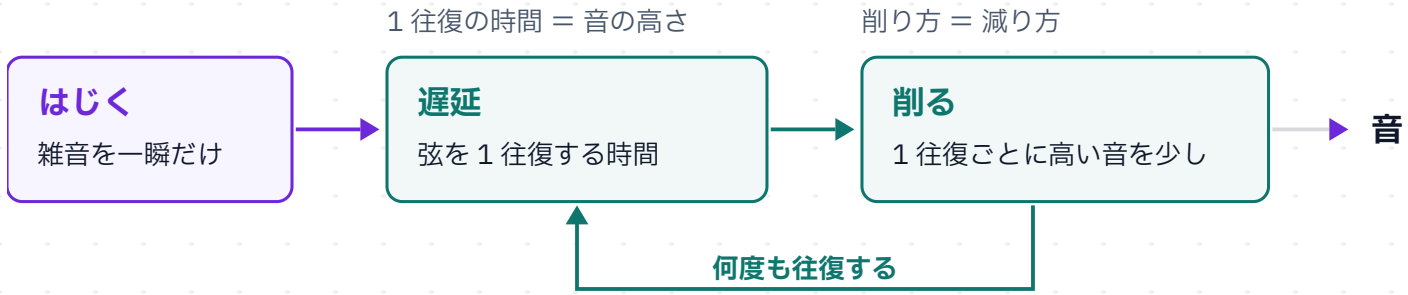
一部の音色では、弓の力をニュートン、速さを m/s で計算しています。**演奏で知られている力の範囲と、同じ単位で比べられます。**

PLUCK: A LOOP OF DELAY

撥弦は、1 回はじいて、弦を行き来する波を追うだけ

13

ギターやベースは、はじいたら指が離れます。あとは弦の上を波が行き来するだけ。その波を計算し、1 往復ごとに高い音を少しだけ削ると、はじいた弦の鳴り方になります。



1 往復の時間で高さが、1 往復ごとの削り方で減り方が決まります。**ピアノの弦も、この仕組みを発展させたものです。**

INSIDE THE PIANO

ピアノは、ハンマーが打って離れたあと、弦と響板が響く

ピアノも撥弦と同じく、ハンマーが弦を 1 回打って離れます。ハンマーが弦と押し合う一瞬も、離れたあとの弦と響板も、時間を追って計算します。

ピアノの 1 つの音の中

ハンマー
弦と押し合って離れる
強いほど短く明るい

→

弦（1〜3 本）
波が行き来する
倍音が高めにずれる

駒でつながる
弦どうしが結びつく
減り方が二段になる

響板
和音全体で 1 枚
共鳴の束と残響の網

接触も毎サンプル計算。戻る波の効果は、接触時間の上限（1 往復）で入れてある

ハンマーは打って、すぐ離れます。**ピアノらしさは、弦の硬さ、弦どうしの結びつき、和音で共有する響板から出ます。**

INSIDE THE PIANO, PART TWO

低音の金属的な鳴りも、ペダルの響きも、仕組みとして入れてある

弦を行き来する波だけでは、ピアノに聞こえない所があります。音域ごとに、足りない仕組みを足しています。



どれも、録音を足すのではなく、**物理の理由がある仕組みとして足しています。**

Part 03 · ③ 合わせる

音色を似せる作業を、誰がやるか

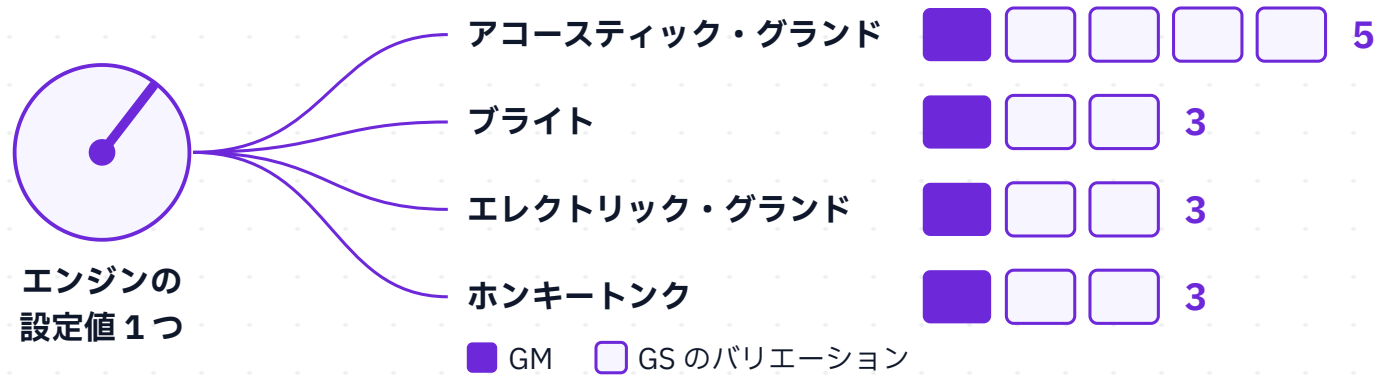
MEASURE, THEN CLOSE THE GAP

ONE SETTING, FOURTEEN VOICES

設定値を 1 つ動かすと、14 の音色が一緒に変わる

14

サンプルを差し替えられないので、音色ごとに計算の中身を合わせる作業が残ります。しかもピアノ系の 14 音色（GM 4 と GS のバリエーション 10）は同じ計算の仕組み（**エンジン**）を共有していて、その設定値を 1 つ動かすと、全部が動きます。



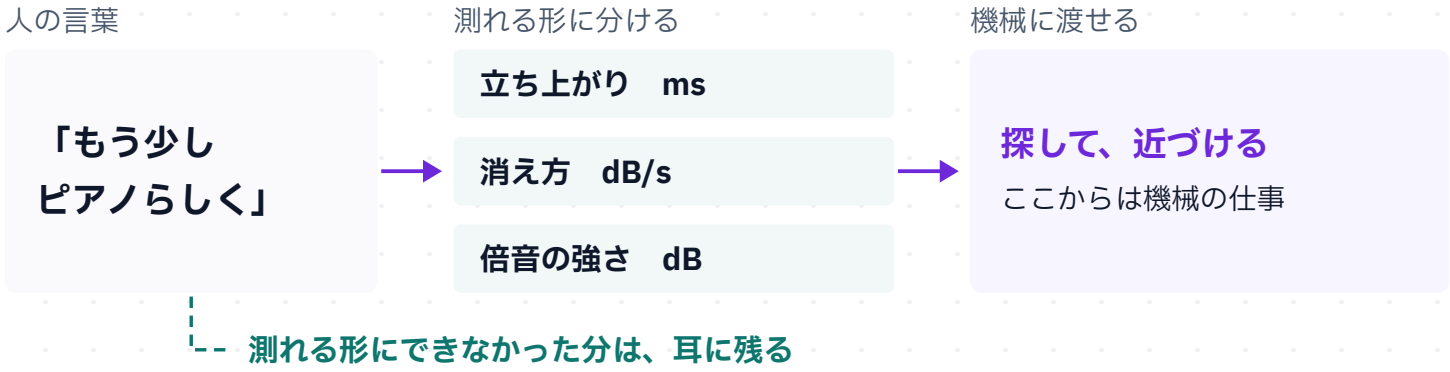
1 音色を良くしたつもりで、ほかの 13 音色を壊すかもしれません。**188 音色を、耳で確かめ続けることはできません。**

SIMILARITY HAS TO BE SAID

「似ている」は、機械が自分では言ってくれない

15

耳で確かめ続けられないなら、合わせる作業は機械に任せたい。ところが物理モデルは式のとおりに鳴るだけで、その音がピアノに似ているかは、どこにも書いてありません。



似せる作業は機械に回し、**似ているかの最後の判断は、耳に残します。**

#

CAPTURING REFERENCES

リファレンスの音は、プラグインを外から鳴らして集める

16

似せるには、まず手本の録音が要ります。aubounce は、Mac のプラグイン音源を、画面を開かずにコマンドで鳴らして録音する道具で、**github.com/libraz/aubounce** で公開しています。ただ、外から鳴らすと、失敗してもエラーになりません。

段階	何もしないと	対策
① **読み込み**	受け付けたのに、ずっと無音	→ **画面側の処理を回して待つ**
② **最初の 1 音**	最初の音だけ、落ち着いていない	→ **1 音鳴らして捨てる**
③ **録音**	オフラインと言わずに速く回すと途中で無音	→ **実時間で録る**
④ **全体**	効いていない設定も、 それらしいファイルになる	→ **全部を読み直して検査**

落とし穴は、エラーにならず、**それらしい別のファイルが返ってくる**こと。検査を録音の手順に入れています。

#

SPLITTING SIMILARITY

「似ている」を、耳が感じる点ごとの物差しに分ける

17

「この音、ピアノっぽい」を、耳が感じる点ごとに分けて、単位のある物差しで測ります。報告と採否では、1 つの点数にまとめません。

耳で感じること		測るもの	単位
鳴り出しの速さ	→	**立ち上がりの時間**	**ms**
消えていく様子	→	**1 秒あたりに下がる量**	**dB/s**
音色のかたち	→	**成分（倍音）ごとの強さ**	**dB**
音程の合い方	→	**基音の高さのずれ**	**cents**
明るさ	→	**高い成分の多さ**	**リファレンスとの比**

この物差しで近いと出ても、**似ているとは言い切りません。**それを決めるのは耳です。

WHO DEPENDS ON WHAT

共通の設定は、複数の音色が同じ値を求めたときだけ動かす

18

さっきの 14 音色の問題への答えです。設定値は 2 種類あり、動かす順番を分けています。

共通の設定値（ピアノのエンジン：133） — 動かすと 14 音色すべてに効く

グランド 5	ブライト 3	エレクトリック 3	ホンキートンク 3
自分だけの設定値	自分だけの設定値	自分だけの設定値	自分だけの設定値

① まず、各音色は自分だけの設定値で合わせる（ほかの音色は動かない）

② 共通の設定値は、複数の音色のリファレンスが同じ値を求めたときだけ動かす

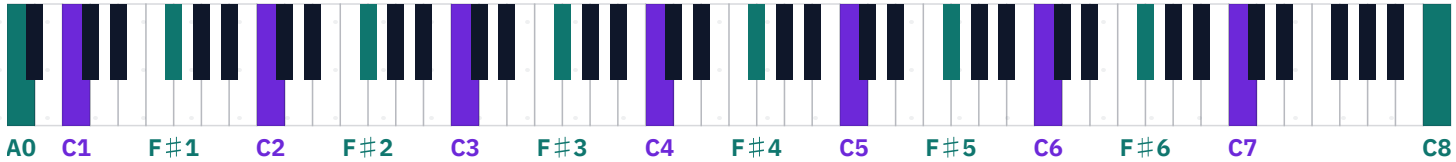
どの設定値が誰に効くかは、人が書き写さず、**ライブラリ自身が一覧にして出します。**

SEARCH AND VERDICT

探索は機械に回させ、採るかどうかは「見せていない録音」で決める

ここでは、ピアノを例に説明します。リファレンスは、本物のピアノを鍵盤ごとに録音したもの。その鍵盤を１つおきに分け、半分の録音で探し、残り半分の録音で採点します。

ピアノの鍵盤（A0〜C8）。録音したのは 15 鍵 × 強さ 4 段（探すのはそのうち 3 段）。合わせる先は 3 台のうち 1 台



① 探す用の 8 鍵で探す
設定値を１つずつ、その軸で
いちばんずれが減る値へ動かす

→

② 採点用の 7 鍵で採点
使わなかった鍵盤の録音と、
項目ごとに比べる

→

良くなり、上限内 → 採る

ならない → 元の値のまま

探すのに使った録音では採点しません。**ただ採点用も同じピアノの隣の鍵盤で、試せるのは鍵盤の間を埋める力までです。**

ADD A MECHANISM

届かないときは、数字合わせではなく、足りない仕組みを足す

20

自動で探しても、届かないことがあります。そのとき、ずれを埋めるためだけの設定値は足しません。

届かない：どの設定値を動かしても、その項目の測定値が動かない

✕ 数字合わせ　ずれを埋める設定値を足す → **数字は合うが、別の条件で壊れる**

○ 仕組みを足す　本物の楽器で起きることを考える → その経路を足して、測る

例（弓）：速く引くほど、押す力も比例して増える
例（フルート）：強く吹くほど、息の帯が速くなる

これは方向の話で、結果の話ではありません。足した仕組みは既定でオフで、**出荷の音色に入れるかは、音色ごとに測って決めます。**

#

WHERE WE ARE

細かく詰めたのはピアノだけ。ほかは、同じ工程に載せたところまで

いま、どこまで来ているか。ピアノは工程をひととおり通しました（採点の式を直したので、いまは合わせ直し待ち）。ほかのエンジンは、自動で探したところまでです。

	集める	測る	探す	判定する	耳で確認	完了
ピアノ	**判定まで通した（いまは合わせ直し待ち）**				**0 音色**	**0 音色**
ほかのエンジン	**自動で探したところまで**					

数字だけでは「完了」にしない決まりです。**だから耳の欄は、いまは 0 です。**

TAKEAWAYS

今日のまとめ。約束は制御に、数字は補助に、不足は仕組みで

22

① 約束は制御に — 受け取る
鳴らし方を替えても、約束は崩れない

② 数字は補助に — 合わせる
近づけるのは機械、最後の採否は耳

③ 不足は仕組みで — 合わせる
ずれ埋めの設定値は足さない

どれも、**耳の時間を最後まで取っておく**ための形です。
最後に、この発表そのものの話を 2 つ。

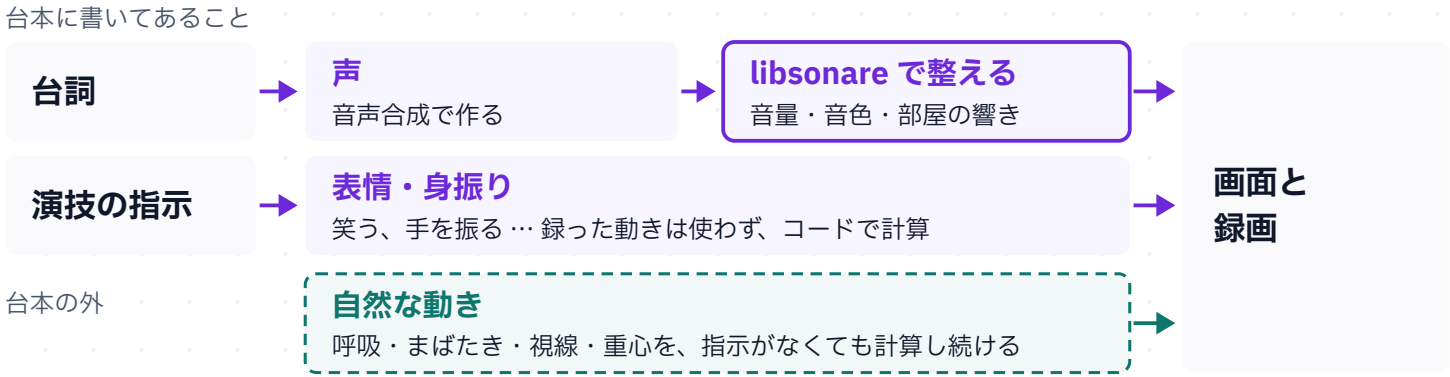
#

HOW THIS TALK IS MADE

この声と動きは、hashidate で作っています

BACKSTAGE

hashidate は、台本を 1 行ずつ受け取り、声と動きを計算で作る仕組みです。作者は同じで、**github.com/libraz/hashidate**（Apache-2.0）で公開しています。



サンプルを持たない音源の話を、**録音ではない声**でお届けしていました。

#

CREDITS

後ろで鳴っていた BGM は、libsonare のピアノです

BGM	しゃろう氏「SUMMER TRIANGLE」（OpenTracks）／編曲：髄雪氏
音色	**libsonare のピアノ**（サンプルを持たず、計算で鳴らしています）
声と動き	hashidate
アバター	デザイン：GPT Image ／ モデリング：GPT-6 Astra

どう聞こえたか、ぜひ教えてください。

耳は、最後に取っておく

ライブラリは libsonare、実機の測定は soundings、この発表を動かす仕組みは hashidate にあります。

https://libsonare.libraz.net

soundings.libraz.net ／ 訂正・質問を歓迎します