

Weaverシリーズ：徹底活用リファレンスガイド

最終更新：2026/04/20

1. Weaverシリーズの「設計コンセプト」と「思想」(Design Philosophy)

1.1 このツールが生まれた理由：独学ゆえの「設計思想」と「試行錯誤」

このWeaverシリーズは、アニメーションの専門教育を受けたわけでもない一人の制作者（作者）が、「たった一人で、短期間に、高品質なアニメを作りたい」という切実な願いを抱え、数え切れないほどの試行錯誤を繰り返して辿り着いた答えです。

私はプロのアニメーターでも、プロのイラストレーターでもありません。専門学校で基礎を学んだ経験もない私が、いざ一人でアニメを作ろうとした時に最大の壁となったのは、「どうすれば動きを美しく付けられるのか」「どうすれば作画崩壊を防げるのか」という、深刻な技術的制約の壁でした。

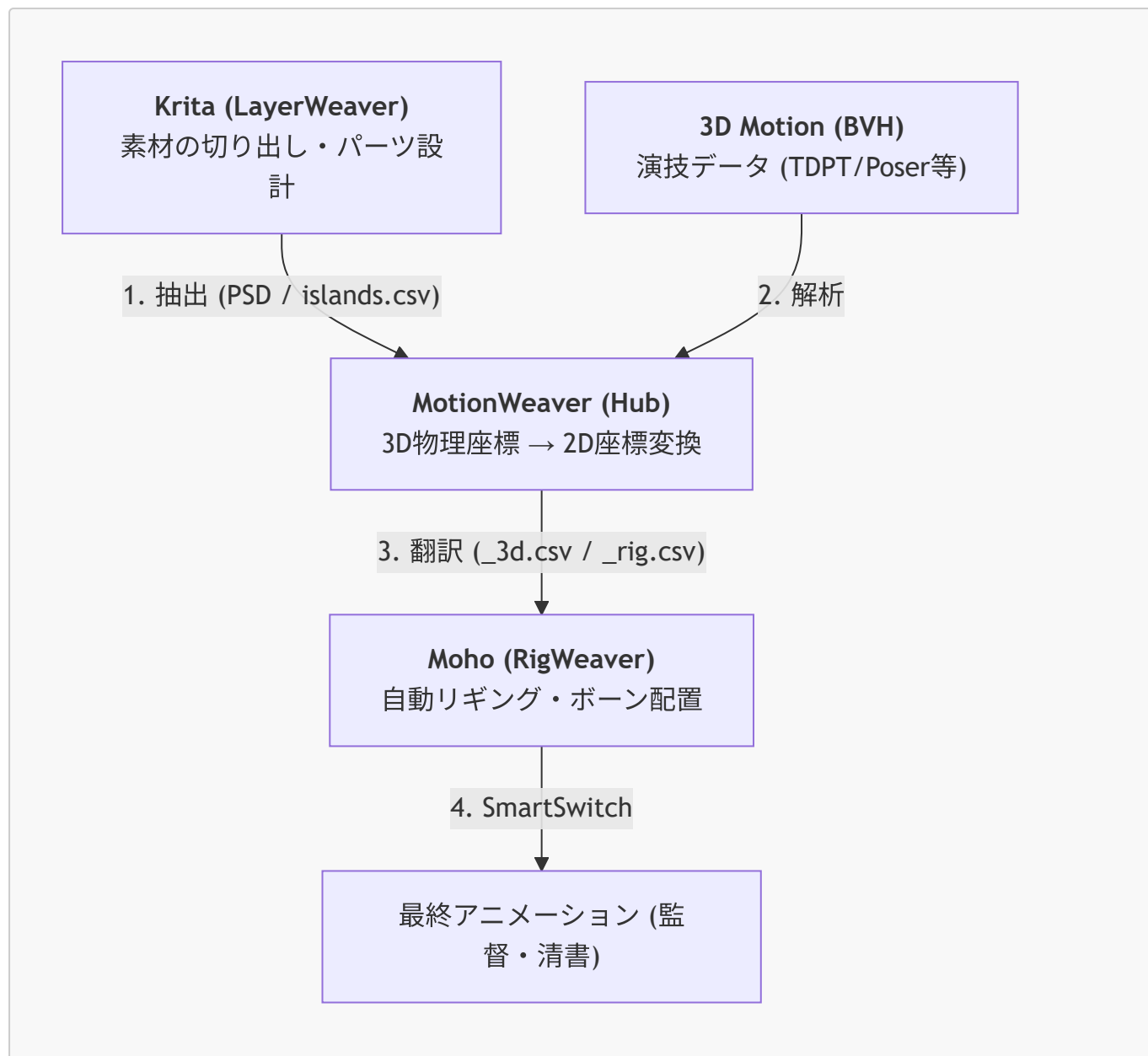
そこで、今私の手元にある素晴らしい既成ツールたち—— **Moho, TDPT, CTA5, Poser, Blender** ——を孤独な作業の中でどうにか「繋ぎ合わせ（Weave）」、もっと楽に、もっと楽しく、創作の情熱をクリエイティブな表現だけに注ぎ込める仕組みは作れないかと考えました。そうして生まれたのが、このWeaver（つむぎ手）シリーズです。

[!NOTE] **G3テンプレート規格の出自について** 本システムが対応している「G3テンプレート規格」は、Cartoon Animator 5 (CTA5) の開発元である **Reallusion社** によって設計された極めて洗練された既存規格です。Weaverシリーズは、この魔法のようなテンプレート規格をゼロから発明したものではなく、あくまで「既存の素晴らしい規格」へと、あなたの絵を正しく導き、折り合わせる（Weave）ための「翻訳機」として機能します。

もしあなたが「一人で作りたいけれど、技術の壁に阻まれて迷っている」のなら、このツールはあなたの味方です。Weaverは、3Dの「幾何学的な整合性」と、2Dの「デフォルメされた表現」の間に架けられた、一本の橋なのです。

1.2 データのバトンリレー：分業型スタジオの思想

Weaverは単一のソフトウェアではなく、複数の専門ツールがバトンを渡す「分業型スタジオ」として設計されています。それぞれが「得意なこと」に専念し、苦手なことはシステムが橋渡しをします。



各ソフトが担う「専門領域」

- **LayerWeaver (Krita Plugin): 【部品工場】** 制作の起点です。イラストを単なる画像ではなく、運動可能な「肉体（島）」と、接続部としての「関節マージン（のりしろ）」に分離します。ここでのフォルダ構成こそが、完成までの全工程を左右する「設計図（Lawbook）」そのものとなります。
- **MotionWeaver (Core Engine): 【翻訳センター・3Dハブ】** システムの心臓部です。3Dのボーン回転情報を、2.5Dキャンバス上の座標と、重なり順を決める奥行き（Z値）へ精密に変換・投影します。Moho専用ではなく、PoserやBlenderでの活用も視野に入れた「ユニバーサル・モーションハブ」として機能します。
- **RigWeaver (Moho Script): 【組み立てライン】** MotionWeaverが算出した「論理的な数値」を、Moho上で視覚的なボーンとして実体化します。画像をボーンへ自動的に「配線（バインド）」し、あなたが描いた絵が即座に同期して動き出す状態を整えます。

1.3 アニメ作りを劇的に楽にする「4つのステップ」

この仕組みの根底には、「先に動き（骨）を固め、後からじっくり肉（絵）を仕上げる」という、独学者の知恵から生まれた効率的なワークフローの提案があります。

1. 動きの「ガイド」を用意する (**MotionWeaver**) : スマホアプリ (TDPT) 等から「3D的に正しい動き (BVH)」を読み込み、作画の揺れを防ぐ強固な土台を作ります。難しい動きをゼロから手で描く必要はありません。
2. 作画の「下準備」を自動化する (**LayerWeaver**) : 3Dのポーズを下絵としてKritaに読み込み、独自のマーカー配置に従って、手作業では膨大な時間がかかる「パーツ切り分け」と「命名」を自動で行います。
3. 自動リギングで「動く絵コンテ」を作る (**RigWeaver**) : Moho上でボーンと画像を自動紐付けし、この段階で「動きのある下書き (ビデオコンテ)」が完成します。ここで演出の良し悪しを判断できるのが最大の強みです。※**Step 1B 「Bind Character」をスキップすると、後の工程 (2B以降) でキャラクターがバラバラに散乱する原因となるため、必ず順番通りに実行してください。**
4. 好きなだけ丁寧に仕上げる (クリーナップ) : 動きが確定したガイドの上をなぞるようにベクターレイヤーで清書します。親グループがすでに動きの鍵を持っているため、描き直した瞬間にその絵はシステムによって完璧にシンクロして動き始めます。

2. 「命名」という名の設計図 (Naming Engineering)

Weaverシステムにおいて、レイヤー名やフォルダ名は、単なるラベルではありません。それは、**「どうリギングし、どう動かすか」というシステムへの直接的な設定情報 (ロジック) **です。

2.1 キャンバス設計：デザイン優先の作画と「輸出用配置」の分離

Weaver 2Dの基本原則は、描かれたパーツを一つの画像として統合する**「1枚絵 (シングル・アイランド) への集約」**ですが、これは「描画段階から離して描かなければならない」という意味ではありません。

- **「重ねて描き、ずらして置く」ワークフロー**: たとえば、スカートやマントをデザインする際、腰の位置に重ねて描くのは当然のことです。全体のバランスを見ながら、キャラクターの一部として魅力的に描いてください。しかし、そのままの状態で解析 (Step 1) を実行してしまうと、画像解析エンジンが「スカート」と「足」を同じ一つの肉体として誤認し、ピクセルが統合されて足の動きが隠れたり、形状が破綻したりします。
- **Kritaのレイヤー機能を活かす**: これを防ぐためのWeaverの作法は、**「描き終わったレイヤーを、キャンバス内の完全に空いている場所へずらす」**ことです。これにより、デザインの整合性を保ちながら、システムには「これは独立した別パーツである」と正しく伝えることができます。キャンバスは「完成予想図を置く場所」ではなく、最終的にはバラバラの部品が並ぶ「部品出荷場」として機能させるのがコツです。
- **論理的バインドの自動化**: Mohoへインポートされた際、パーツは描かれた位置 (ずらした状態) で現れますが、心配はいりません。自動的にボーンとのバインドは完了しているため、Moho上で腰の位置にパーツを戻した瞬間、そのパーツはデザイン通りの場所でシステムによって同期して躍動し始めます。

2.2 特殊スナップ (自動吸着)：作者のリップシンク短縮術

基本的にパーツは手動で位置を合わせますが、「顔 (Face)」と「手 (Hand)」だけは特別です。ここに作者の「一番面倒な作業を一瞬で終わらせたい」という願いが込められています。

- **対象タグ**: @Head (およびネストされた @Mouth等)、@RHand、@LHand
- **自動スナップ機能**: これらはキャンバスのどこに描かれていても、RigWeaver実行時にキャラクター本体の「本来あるべき場所 (腰や手首のボーン位置)」へ自動的に配置 (スナップ) されます。

- **開発の意図**: リップシンク（口パク）や手の差し替えは、アニメ制作で最も頻繁に行われる作業です。これらを自動吸着させることで、制作者は単調な「位置合わせ」から解放され、よりクリエイティブな「表情の演技」に集中できるようになっています。
- **重なり回避（レイヤーの移動）**: 例えばスカートを腰の位置に重ねて描くと、画像解析時に「足」とマージされ、足の動きが見えなくなります。そのため、デザインが固まったら、アクセサリやバリエーションパーツのレイヤーを本体から離れた**「空いている場所」へ移動させてから解析を行う**のがWeaverの正しい作法です。
- **論理的バインドの自動化**: これらをインポートした際、パーツはキャンバス上での位置（バラバラの状態）に配置されますが、ボーン構造とのバインド計算はRigWeaverが裏側ですべて完了させています。
- **自動スナップ機能**: これらはキャンバスのどこに描かれていても、RigWeaver実行時に**キャラクター本体の「本来あるべき場所（腰や手首のボーン位置）」へ自動的に移動（スナップ）**します。
- **レイヤー順序の自由（独立バインド）**: ここが@タグの真骨頂です。これらのレイヤーは、インポートされた時点で個別に必要なバインド情報（ボーンとの設定情報：DNA）を保持しています。
 - **演出の幅**: たとえば「BackHair（後ろ髪）」は頭ボーンの後ろに、「FrontHair（前髪）」は前に配置したい場合、Moho上でこれらを自由にレイヤー移動させることができます。
 - **グループ化の束縛からの解放**: 通常、バインドを維持するために強引にグループ化する必要はありません。独立したレイヤーとして重なり順を自由に入れ替えても、それぞれのパーツは正しくボーンの動きに同期し続けます。
- **全自動化しないという「意志」**: この「重なり順の自由度」を守るため、Weaverシステムはリギングとユニット化（SmartSwitch）をあえて一続きの自動処理にはしていません。ユーザーの手でレイヤーの前後関係を納得いくまで調整し、その後に自分のタイミングでSmartSwitchを実行してパッキングする。この「選べる間（ま）」があるからこそ、Weaverは道具として職人に寄り添えるのです。

B. その他のパーツ（マニュアル組み立て）

スカート（Weary_Acc(0)等）などは自動スナップされません。これらはユーザーがMoho上で手動で正しい位置（腰など）に配置します。位置を合わせた瞬間に、自動完了していたバインドによって躍動し始めます。

2.3 拡張性と手動の「編み込み」：解析テーブルへの登録プロセス

ユーザーがtemplates.jsonを毎回編集しなくて良いのは、システムがリギング開始時に行う「Discovery Phase（探索フェーズ）」のおかげですが、ここにはユーザー自身の「意志」を介在させるための重要なステップがあります。

- **未知のパーツの自動「検知」**: PSD内のレイヤー名に@や>（例：Sword>Hand）などの命名タグが含まれていると、システムは即座にそれを「CSVやテンプレートに記述されていない新しいユニット」としてプレビュー表示します。Viewer上では、それらに対応する「島番号(i)」や「マージン番号(j)」が自動的に割り当てられ、確認できるようになります。
- **解析テーブル（CSV）への手動登録（必須ステップ）**: ここが重要なポイントです。システムは「検知」はしますが、勝手にリギングを完了させるわけではありません。
 1. ** Viewerでの確認**: まず、プレビュー画面でそのパーツの「島番号(i)」と「マージン番号(j)」を確認します。

2. **** 行の追加****: 次に、解析テーブル (`islands.csv` や `rigging_map.csv`) を自分で開き、新しい行を追加します。
 3. **** 情報の手入力****: マーカーにつけた「パーツ名」、確認した「島番号(i)」、そして「マージン番号(j)」を入力します。
- **職人のための最終制御**: この「手入力」の手間があるからこそ、意図しないゴミレイヤーがリギングされるのを防ぎ、職人は自分の狙い通りにパーツを「編み込む (Weave)」ことができます。マージン番号欄では `j23-40` のように範囲指定もサポートしており、これによって一つのパーツに複数のボーンの影響 (屈曲など) を自在に持たせることが可能です。

3. 基幹ファイル仕様：Weaverの知識体系 (The Lawbook)

Weaverシリーズが、多様なキャラクターを個別のプログラム修正なしに動かせるのは、その裏側に「事実」と「ルール」を分離した一貫した設定システム (Lawbook) があるからです。

3.1 密接なファイル連携

システムは以下の3つのファイルが連携することで、一つのキャラクターを完成させます。

1. **`islands.csv` (肉体の部品リスト)** : LayerWeaverが抽出した物理的なパーツ (島) とマーカーの配置情報の「事実記録」です。
2. **`rigging_map.csv` (配線図)** : 「どの島 (パーツ) を、どの骨 (ボーン) に、どのような順番で繋ぐか」を記した「設計図」です。
3. **`templates.json` (マスターカタログ)** : 「人間型ならこの配線図 (`rigging_map_human.csv`) を使え」といったトップレベルの「統治ルール」です。

3.2 `rigging_map.csv` : 骨と肉体を結ぶ「配線」の極意

このファイルは単なるリストではなく、1行の中に「レイヤー名」「島名」「ボーン名チェーン」をセットで記述する、高度な配線図です。

- **どの骨で動かすか (担当決め)** : 1つのパーツ (島) に対して割り当てるボーンを指定し、その動きへの追従を定義します。
- **どの骨をどう繋ぐか (チェーン構築)** : 1行の中に複数のボーンを書くことで (例: `Hip, Torso, Chest`)、システムは自動的に左から右へと親子関係 (チェーン) を構築します。

【重要】枝分かれ (ブランチ) の構築仕組み

`rigging_map.csv` の1行では直列のチェーンしか書けませんが、**「複数の行で同じボーン名を共有する」**ことで、複雑な枝分かれが実現します。

- **例**: 「胴体の行」と「左腕の行」の両方に `LShoulder` というボーン名を含めておきます。するとシステムは「胴体パーツにLShoulderがある」かつ「左腕の起点もLShoulderである」という共通の接合点 (Pivot) を見つけ出し、パズルのピースを繋ぐように、胴体から左右の腕へと分かれる巨大なツリー構造を全自動で完成させます。

A. 強度と物理の制御 (Forces)

作者が「アニメらしい、しなやかな動き」を追求する中で生まれたパラメータです。

- **hip_force (初期値: 0.4)**: 腰のボーンが画像（島）を引っ張る強さです。これが低いと、激しい動きの際に胴体がボーンに置いていかれ、高いとガチガチの硬い動きになります。作者が何十回ものテストを経て辿り着いた「最も人間らしく、かつ破綻しない」黄金比が設定されています。
- **flexi_force (初期値: 0.1)**: フレキシ・バインド（ゴムのような変形）の適用強度です。関節の曲がり角で、肉が不自然に潰れるのを防ぎ、有機的な柔らかさを演出します。

B. 3Dと2Dの架け橋（Virtual Bones ロジック）

Weaverの最も知的な仕組みの一つが「架空の骨」の自動生成ですが、これが存在する真の理由は、BVH（3D）側の欠落ではなく、**「PSD（作画）側の肉体的な制約」**にあります。

- **PSDを“正解”とするがゆえの課題**: WeaverはPSDの島とマーカーの位置を絶対的な正解としてリグを組みます。しかし、2Dイラストでは通常「胴体（Torso）」は1つの大きな塊として描かれます。PSDのマーカー構成（例：腰と胸のみ）に忠実すぎると、胴体には1本の長いボーンしか割り当てられません。
- **2Dならではの「ボーン増設（構造的補完）」**: このままでは、体を反らせた時に1枚の2Dメッシュを柔らかく曲げるための「中間関節」が物理的に足りなくなります。そこで **templates.json** の **virtual_bones** により、**PSD上には描かれていない制御用の架空関節（TorsoやNeckなど）を「増設」**し、3Dデータが持つ細やかな動きを1枚のメッシュに破綻なく適用して変形を安定させます。
- **redirect_children**: 増設した架空ボーンに対して、本来の子供たち（腕など）の接続先を自動的に繋ぎ直す命令です。これにより、作画のシンプルさを維持したまま、アニメーションの美しさを最大化しています。

C. その他の重要パラメータ

- **face_bone_keywords: hair, eye, mouth** 等。これらのキーワードを含むボーンは、身体の激しい物理変形（バインド）から保護され、形状を保つように特別扱いされます。
- **mismatch_indicators**: 3DのBVHボーン名とテンプレート名が一致しない場合に、システムが「これはあの骨のことだな」と推測するためのヒント集です。
- **search_radius_factor (LayerWeaver)**: Krita上でお絵描きソフトが「この画像パーツは、どの骨に属させるべきか」を検索する際の半径の倍率です。

3.3 基幹ルール：templates.json の技術パラメータ辞典

templates.json は、Weaverシステムの基幹ルールであり、プログラムを一切触らずに挙動をカスタマイズするための実用的な設定ガイドです。

パラメータ名	技術的な詳細と「作者の意図」
bind_all_unassigned_layers	true に設定すると、CSVに名前がない浮いたレイヤーも、座標から推測して最も近いボーンに自動バインドします。「塗り忘れた小さな影」などが置いていかれるのを防ぐ安全網です。
strict_match	true の場合、レイヤー名とテンプレート名が1文字でも違うと解析をスキップします。多数のキャラを管理する際、意図しない混同を避けるための厳格なモードです。
search_radius_factor	島（Island）に対してマーカーをどの程度の距離まで探すかの閾値。キャラクターのプロポーション（太り具合）に合わせて微調整し、誤判定を防ぎます。

パラメータ名	技術的な詳細と「作者の意図」
<code>flexi_binds</code>	全身のボーンの中で、どのボーンに「非剛体（しなり）」の物理変形を許可するか。スカートや尻尾など、有機的な動きをさせたい部位をここで定義します。
<code>face_bone_keywords</code>	全身の物理的なバインド計算（変形）から特定のボーンを外すための**「除外キーワード」**。ここに登録された単語（eye, mouth等）を含むボーンは、身体の大きな動きによるメッシュの歪みから保護され、形状を維持するように処理されます。
<code>variation_mouth_maps</code>	口のバリエーション名（音素）と読み込み順序の定義。リップシンクを全自動化し、3Dモーションの言葉のリズムを2Dに直接流し込むための「辞書」です。
<code>joint_maps</code>	3Dモーション（BVH）側のボーン名と、Weaver内部の論理ボーン名を結びつける**「ボーン名翻訳辞書」**です。ソフトごとに異なる3D側の命名を、Weaverが理解できる共通言語に変換するための橋渡しを担います。
<code>redirect_children</code>	<code>virtual_bones</code> （架空関節）を追加した際、本来の子供ボーン（手首など）の接続先を「元の親」から「新設された架空親」へ自動的に繋ぎ直す論理リマップです。
<code>is_accessory</code>	マーカー名（例： <code>@Skirt>Hip</code> ）をトリガーに、パーツを自動検知するフラグ。マーカーが存在する時だけリギングが始動するため、「スカートのあるキャラと無いキャラ」で同じテンプレートをそのまま共用できるという極めて高い柔軟性を持ちます。
<code>reorder_to_template_joints</code>	リギングファイルの書き出し順序をテンプレート通りに整流します。Mohoでのバッチ処理時に、ファイルの構造を完全に一定に保つための安定化機能です。
<code>rig_perspective_base_z</code>	【RigWeaver用】 Moho上のレイヤー倍率からパースの深さを逆算する基準値。値を小さくするほどパースが強くなります（TDPT等では4.0を推奨）。
<code>motion_perspective_default_z</code>	【MotionWeaver用】 モーション変換時にパース強度が指定されていない場合に使用されるデフォルト値（標準 8.0）。

3.4 CSVファイル：物理（Island）と論理（Rigging）の疎結合

- **islands.csv**: 「この描画された島には、この関節マーカーが乗っている」という、**物理的な事実**の記録です。
- **rigging_map.csv**: 「この島（肉）を、MotionWeaverのどの論理ボーン（骨）で動かすか」という、**論理的な命令**の記録です。この2つを分けたことで、3Dの骨の名前が変わっても、ユーザーはCSVの対応表を一行書き換えるだけで、既存のお絵描きデータを再利用できるようになっています。

3.5 ファイルの配置場所：プロジェクト固有設定とシステム標準

Weaver は、作業中のプロジェクトに合わせて最適な設定ファイルを自動的に見つけ出します。基本的には以下のルールに従って配置します。

- **プロジェクト固有のファイル** (`templates.json`, `islands.csv` 等): PSD ファイルや Moho プロジェクトと同じフォルダに置きます。特定のキャラクター専用のカスタマイズを行いたい場合に有効です。Weaver はまずこの場所をチェックします。
- **システム標準のリソース**: Mohoインストール先 / `Scripts` / `ScriptResources` / `rigweaver` フォルダに置きます。複数のプロジェクトで使い回す共通のテンプレート（標準人間型など）はここに集約しておくくと便利です。

[!TIP] **スクリプトフォルダを汚さない運用** Weaver シリーズでは、スクリプト本体 (`scripts/tool`) の中身を書き換える必要はありません。すべての設定は「プロジェクトフォルダ」か「専用のリソースフォルダ」で管理されるため、スクリプトのアップデート時も設定が消える心配がなく、安全に運用できます。

4. MotionWeaver : 3Dモーションハブと「情緒的検証」

MotionWeaverは単なる変換ソフトではありません。3Dの世界と2Dの世界を繋ぐ「司令塔」です。

4.1 ユニバーサル・モーションハブとしての側面

MotionWeaverはMoho専用ツールではありません。BVHを読み込み、アニメ制作に最適な「クリップ」として切り出す機能は、**PoserやBlenderでの3D制作**においても強力な武器となります。

- **3Dワークフローの起点**: 3Dソフトで高度なアクションを作り、それをWeaverで必要な部分だけトリミングして保存。これにより、3Dの確かな制作力を2D/3D両方のプロジェクトに共有できます。

4.2 `_3d.csv` : あなた専用の役者による「演技の記録」

出力される `_3d.csv` は、ただの座標データではありません。それは、3Dの複雑な動きを2.5Dキャンバス (Moho空間) へと丁寧に投影・変換した「完成された演技」そのものです。

- **作画への集中**: このファイルはあえて「ユーザーが中身をいじれない」ように設計されています。難しい数字の調整はシステムに任せ、制作者は「納得のいく絵を描くこと」にすべての心血を注げるようにするためです。長期的なデータの保存や再利用、アセット運用の考え方については、**「第5章：動く資産の運用」**を参照してください。

4.3 視覚的検証：Viewer機能 (`csv_viewer.py`)

MotionWeaverには、出力された座標データを視覚的に確認するための **Viewer (`csv_viewer`)** が搭載されています。

- **リギング前のプレビュー**: Mohoに読み込む前に、3D空間でスケルトンが正しく動いているか、関節の接続が破綻していないかを確認できます。
- **診断ツール (将来の展望)**: 現在開発中の診断機能では、PSDの構造とリギングマップの矛盾を自動で検知し、「このレイヤー名の綴りが間違っています」「このパーツにマーカーが足りません」といったフィードバックを返す予定です。

4.4 情緒的な「デフォルメ」：煽り・俯瞰・アングル制御

現実に忠実な3Dだけでは、アニメ特有の「迫力」は生まれません。

- **アングルの自由**: 手動設定で「見下ろし方向 (Fukan)」や「見上げ方向 (Aori)」を柔軟に調整できます。
- **2Dへの緊急脱出**: 3Dベースで制作していて、「もっと大胆に形を崩したい!」と思ったら、同じBVHから出力された `_3d.csv` を使ってMohoへ移行してください。2Dのベクター操作なら、アニメらしい誇張表現が数秒で実現します。

5. 「動く資産」の運用：データ保存とキャラクターの再利用

Weaverシステムにおいてデータは単なる「記録」から、再利用可能な「資産」へと進化します。ここでは、ツールを横断して効率的な制作を実現するための思想について解説します。

5.1 アニメーション制作におけるデータ保存の考え方

- **中間データと最終アクションの区別**: Moho用の `_3d.csv` や BVH ファイルは、あくまで 3D の演技を 2D へ持ち込むための「途中のアクションデータ (中継地点)」であり、最終的な成果物ではありません。
- **タイムライン上での完成**: 最終的なアクションは、エフェクトの追加や動きのブラッシュアップを含めて、Cartoon Animator 5 (CTA5) や Moho の「タイムライン」上で作り込むべきです。
- **保存形式の選択**: アクションを保存する際は、BVH や `_3d.csv` に書き戻す必要はありません。
- **理由 (エフェクトの保持)**: 中間データ形式 (CSV等) で保存してしまうと、追加したエフェクト情報が保存できないため、各ソフト独自の形式で「アクション」として保存した方が適切に整理・運用できます。

5.2 キャラクターの再利用と効率化 (ツールの使い分け)

本ワークフローで使用するソフトは、単に CSV データで動かすだけのものではなく、動かしたキャラクターを「動く資産」として育てていくことが可能です。

- **キャラクターの清書と拡張 (Moho)**: 動かしたキャラクターをベクターで清書し、さらにアクションを追加して保存することが可能です。
- **キャラクターの入れ替えと流用 (CTA5 / Moho)**: 各セクションの冒頭でキャラクターを変更したい場合、キャラクター自体を入れ替えれば、タイムライン上のアクションはそのまま流用できます。
- **修正の最小化**: キャラクターを入れ替えた際、ベクターデータの再修正が必要になる可能性はありますが、修正作業は最小限で済みます。特に CTA5 の場合、ベクター作成はできませんが、同じアクションを驚くほど簡単に使い回せること自体が極めて強力な武器となります。

5.3 ボーンとエフェクトの継承 (共通ボーンの威力)

- **ボーンの共通性**: キャラクターが異なってもボーンの構造が同じであれば、キャラクターを入れ替えても同じボーンのアクションを使い続けることができます。
- **エフェクトの引き継ぎ**: ボーンに対してエフェクトを設定している場合、ボーンが共通であれば、キャラクターを入れ替えてもそのエフェクト設定はそのまま引き継がれます。
- **アクセサリの動的検知**: アクセサリはマーカーから動的に検知されるため、衣装違いのバリエーションキャラ (例: スカートの有無) であっても、同じボーンセットの設定をそのまま使い回すことが可能です。システムがその都度「そこにあるパーツ」を発見してリグを構築します。

•

5.4 制作コストの最適化：シーン主導のリギング (Scene-Driven Rigging) +Weaverシリーズが推奨するワークフローは、**「動き（3D）が先、リグ（2D）は後」**です。この順序を守することで、制作コストを劇的に抑えることができます。+- **「万能リグ」の罠を避ける**: 従来の2D制作では、最初に「360度どこから見ても破綻しない万能なキャラ」を作ろうとして、リギングだけで数週間を費やし、結局そのほとんどを使わずに終わるという悲劇がよく起こります。+- **必要な角度だけを清書する**: BatchWeaverでアングル（カメラワーク）を先に決めることで、そのシーンで本当に必要な「角度」と「レイヤー数」が明確になります。+- **後戻りのない制作**: 「いざリグを組んだが、使いたいアングルの素材が足りなかった」という後戻りは、Weaverでは起こり得ません。3Dのガイドが、常にあなたの描くべき絵の「最短ルート」を示してくれるからです。

6. プロフェッショナルな自動化：一括量産（Batch Integration）

作者がプロの制作現場に匹敵するスピードを実現するために構築した、Weaverシステムの到達点とも言える自動化の極致です。

6.1 Moho を「ビデオコンテ兼コントローラー」にする

このパイプラインにおいて、Mohoは単なるアニメーションソフトではなく、プロジェクト全体の演出を司る**「デジタル・ディレクターズ・コンソール」**へと変貌します。

1. **動く下絵としてのスケルトン**: 読み込んだ `_3d.csv` のスケルトン（演技データ）をガイドに、Moho上で「このタイミングでこの角度から撮る（カメラ・レイヤーの回転）」という演出プランを組み立てます。
2. **マーカーによる演出の「切り出し」と静止画書き出し（動画・静止画とPose-to-Pose補間）**:
 - タイムライン上にMohoの「マーカー」を打ちます。ラベル（名前）を付けたマーカーはすべて、そのフレームの画像が **PNG形式で自動書き出し** される対象となります。書き出されるPNGは、動画用・静止画用ポーズであることを明示するため、末尾に `_p` が付与されたファイル名（`{ラベル名}_p_f{フレーム数}.png`）で保存されます。
 - 画期的な「**Pose-to-Pose（ポーズ間自動補間）**」ワークフローの実現: マーカーの長さ（幅）をドラッグして設定すると「動画カット（CSV/Batch）」として認識され、幅が0のマーカーは「静止画（1フレーム）カット」として処理されます。さらに、**持続時間0の静止画マーカーは、PNG書き出しと同時に「1フレームのみの3D CSVデータ（サフィックス `_f[frame]_s`）」としても自動出力されるようになりました**。これはアニメーターにとって最大の朗報です。3Dモーションデータから代表的なキースタンスやキーポーズだけをマーカーで切り出して流し込み、各ポーズ間をMohoの強力なベクター・ボーンの自動補間機能で繋ぐことで、**「Pose-to-Pose（ポーズ間補間）による効率的で滑らかなアニメーション制作」**が一瞬で構築できるようになります。これにより、長大なモーションを一からベイクするだけでなく、必要な決めポーズだけを流し込んで手作業で味付けするクリエイティブな作画演出が劇的に容易になります。
 - **【Mode 2 限定：T-Poseマーカーの自動挿入と下絵PNG生成】**: BatchGen（ステップ4）のモード選択にて「**Mode 2: Add T-Pose (Frame 0)**」を選択すると、タイムラインのフレーム0にすでにマーカーが存在するかをシステムが自動チェックします。存在しない場合のみ、フレーム0に **"T-Pose"** というラベルのマーカーを自動で作成・挿入します。これにより、バッチ保存直後の静止画書き出し機能が動作した際、初期姿勢のPNG画像（`T-Pose_f0.png`）が自動で出力されます。この画像は、後続のKrita連携（Layerweaver）における「イラスト部品の配置下絵」として極めて完璧に機能します。なお、すでにフレーム0に他の手動マーカーが存在する場合は、既存の情報を保護するために自動挿入処理を安全にスキップする競合防止ロジックが組み込まれています。

- **レイヤーによるカットの整理**: `3D_Angle_Controller` の中に `story1`, `story2` といったグループレイヤーを作成し、それぞれのレイヤー上にタイムラインマーカー (TimelineLabel) を置くことで、複数のカット割り案を一つのプロジェクト内で整理・比較できます。
- 3. **情報の全自動走査**: スクリプトを実行すると、グローバルタイムラインおよび選択したレイヤー上のマーカーを高速スキャンします。どのレイヤーにマーカーを置いても、アニメーションデータは常に `3D_Angle_Controller` から正確に抽出されるため、安心してレイヤーを分けた演出管理が可能です。

6.2 実行プログラム (RUN.bat) の自動生成と量産

RigWeaver Step 4 (BatchGen) の実行は、単なるデータ出力ではありません。それは、巨大な工場を動かすための「スイッチ」を組み立て、同時に「写真撮影 (PNG)」を行う作業です。

- **瞬時のセットアップ**: Mohoから抽出された演出情報を統合し、一瞬で **RUN.bat** (実行用バッチファイル) と、それに対応する **CSV指示書** がカレントフォルダに作成されます。
- **【完全ポータブル (自己完結) パッケージ機能】**: バッチファイルを元のBVHとは異なる任意の新規フォルダ等に保存した場合、**元のBVHファイル本体** およびアニメーションのベイクに必要な **設定ファイル一式** (`templates.json` など) が保存先へすべて自動的に同封コピー (複製) されます。さらに、バッチ内のBVH指定パスは絶対パスから「同封されたBVHファイル名のみの相対指定」へと自動で書き換えられます。これにより、フォルダ全体を圧縮・共有したり別のPCへ移行させても、絶対パスの依存から完全に解放され、環境変数 `PATH` に `MotionWeaver.exe` が通っていればダブルクリック一つで即座に実行できる完全なポータブル作業パッケージが完成します。 ※バッチを元のBVHと同一フォルダに保存 (Quick Save) した場合は、無駄なファイルロックや二重上書きを防ぐため、このコピー処理は安全に自動スキップされます。
- **【新規フォルダ保存時の手動命名ルール】**: 「No (Select Folder...)」を押して任意のフォルダへ保存する際、Moho ProのLua APIである `LM.GUI.SaveFile` の制約上、保存ダイアログに初期デフォルト名を自動入力することができません。そのため、別フォルダへ初めて保存する際は、ダイアログのファイル名欄に**手動で RUN または RUN.bat** と入力して**保存ボタンを押してください** (これによりCSVも連動して `RUN.csv` として正しく命名・出力されます)。最も手軽な標準保存は、BVHと同じ場所に一瞬で保存される「Yes (Quick Save)」の活用です。
- **静止画の自動書き出し (`we_makepng.lua`)**: ラベル付きマーカーが見つかったと、バッチ生成の直後に「PNGを今すぐ書き出すか？」という確認ダイアログが表示されます。これを承認するだけで、指定した全フレームがプロジェクトフォルダへ一括保存されます。
- **一括指示出しの快感**: 作成された **RUN.bat** をダブルクリックするだけで、MotionWeaverがバックグラウンドで高速起動。指定された何十種類ものアングル、何百フレームものシーンが、あなたの介在を必要とせずに次々と書き出されていきます。
- **BatchWeaver による広域量産**: `BatchWeaver.py` を活用すれば、特定のフォルダ内にある全BVHファイルに対して、一律の処理をかけることも可能です。これにより、数分の間にプロジェクトに必要な数十パターンのモーション素材を自動で用意することさえ現実となります。

これは、「楽をしたい」という一人の独学者の情熱が、結果としてプロフェッショナルな量産体制を個人環境に持ち込むことに成功した、情熱の結晶なのです。

7. アニメーションの「デフォルメ」と「情緒」を深めるテクニック

Weaverが整えるのは「論理」ですが、そこに「豊かな表現 (情緒)」を付与するのはあなたの仕事です。作者が日々の制作で見出した、システムを使いこなすための微調整術を紹介します。

7.1 自動的な仕組み：バインドを継承し、ユニット化する

RigWeaverによって自動吸着された `@Head` や `@Hand`、あるいは `@Skirt` といったパーツ群は、`Weaver_SmartSwitch` スクリプトを通すことで、単なる画像レイヤーから「知的なユニット」へと昇華されます。

- **スイッチ（単一表示）とグループ（一括表示）の使い分け:**
 - **表情や手のポーズ:** これらは「どれか一つを表示する」**スイッチレイヤー**としてまとめられ、瞬時の差し替えを可能にします。
 - **スカートやフリルの「ひらひら」:** 分散して描かれた複数のパーツ群（拡散の塊）を一つの**グループレイヤー**として統合し、すべてのパーツを同時に表示させることができます。
- **核心：バインド情報の「継承」と「昇格」:** ここがWeaverシステムの最も強力な点の一つです。通常、Mohoで既存のレイヤーを単純にグループ化すると、元の個別のバインド情報は失われたり、グループに対して再設定が必要になったりします。しかし、SmartSwitchは元々の**@タグ レイヤーが持っていた「島（Island）のバインド情報」を、新しく生成されたグループ（またはスイッチ）レイヤーそのものに自動で適用します。**
- **制作者への恩恵：ベクター清書と線の強調:** グループ・レイヤー自体が正しくボーンにバインドされるため、その中のレイヤーに後からベクターで清書を加えたり、線画の強調（ディティールアップ）を行ったりしても、**すべての加筆が完璧に関節の動きに追従します。** これにより、「3Dベースの安定した動き」の上に、「2Dならではの緻密な描き込み」を矛盾なく共存させることができます。

【New】ボーン選択からの「即時パーツ作成」機能

SmartSwitch v3.2からは、レイヤーだけでなく**ボーン（特に `_Scale` などのPinBone）を選択した状態での実行にも対応しました。**

- **挙動:** レイヤーを選択せずにボーンを選んで実行すると、そのボーンに紐付いた「空のベクターレイヤー」と「親グループ」を自動生成し、即座にバインドを完了させます。
- **制作者への恩恵:** 胸のワッペンや腕のアクセサリなど、既存のリグに新しいパーツを後付けしたい場合、ボーンを選んでボタンを押すだけで、位置ずれの心配がない「描き込み専用のキャンバス」が正しい階層に用意されます。
- **命名の自動化:** 作成されるレイヤーには選択したボーン名が自動で割り当てられるため、管理の手間も最小限に抑えられています。

7.2 強度（Force）の微調整で「質感」をコントロールする

`templates.json` の強度設定は、キャラクターの「服の硬さ」や「肉体のしなり」を反映する手段です。

- **重い衣装の場合:** `hip_force` を少し上げ、`flexi_force` を下げることで、厚手の服が体に密着して動く質感を表現できます。
- **しなやかなクリーチャーの場合:** 逆に `flexi_force` を高めに設定することで、骨の動き以上に肉体が波打つような、人外の不気味さや優雅さを演出できます。

7.3 開発者が仕込んだ「究極の安全装置（フェイルセーフ）」

技術的な安定性を極限まで高めるため、プログラム（`RigWeaver.lua`）内部には、設定ファイルを無視してでも発動する**「ハードコードされた安全装置」**が存在します。

- **肩の強制結合ロジック:** `ExecuteCreateBones2D` 内の処理において、肩のボーン (`LShoulder` / `RShoulder` 等) は、CSVや3Dデータの構造がどうであれ、**強制的に「Torso (胴体)」の子供として繋ぎ直されます。**
- **目的:** これにより、どれほどトリッキーな3Dモーションを読み込んだとしても、2Dリグの肩が胴体から外れることを許さず、常に安定したメッシュ変形を約束します。これは、作者が数千回のクラッシュと格闘して辿り着いた、Weaverの「見えない守護神」とも呼べる究極の安全装置です。

7.4 3Dと2Dのハイブリッド：いつ「緊急脱出」するか

Weaverの真骨頂は、3Dの安定感と2Dの自由度をいつでもスイッチできる点にあります。

- **3Dに留まるべき時:** 歩行、ダンス、多人数が絡むシーン。これらの「整合性」が問われる動きは、MotionWeaverによる3D投影が最適です。
- **2Dへ飛び出すべき時:** キャラクターの感情が爆発する「決めゴマ」、物理法則を無視した「パースの誇張」。こうしたシーンでは、3Dデータの制約に縛られず、PSDからインポートされた各レイヤー（ラスターかベクターかを問わず）を、メッシュ変形やベクター操作で大胆にいじり倒してください。`_3d.csv` が提供する正確なアクションを土台にしつつ、その上で2Dならではの「嘘」をつくのがWeaverを使いこなすコツです。

付録：トラブルシューティングと作者の知恵袋（FAQ）

独学でツールを使い倒す中で、作者がぶつかった壁とその乗り越え方です。

Q1. 画像がボーンについてこない、または変な方向に伸びる

- **原因:** マーカーの配置が「島 (Island)」の境界線ギリギリすぎませんか？
- **対策:** LayerWeaverが画像を認識する際、`search_radius_factor` の範囲内にマーカーがないと、迷子になってしまいます。マーカーは「そのパーツの肉の中」にしっかりと置いてください。

Q2. 新しい動物や特殊なキャラを作りたい

- **答え:** `templates.json` に新しいセクション（例：`quadruped`）を追加し、`target_hierarchy` を定義するだけで、システムはその骨格を理解します。
- **知恵袋:** 既存の `human_v2` をコピーして、骨の名前だけを差し替えるのが一番の近道です。`virtual_bones` を使えば、どんな歪な骨格でも、Moho上では「まっすぐで扱いやすい骨」として扱うことができます。

Q3. データの正確性に不安がある

- **答え:** 迷わず `csv_viewer.py` を実行してください。Mohoへ飛ばす前の「生の3D変換結果」が見えます。ここでスケルトンが正しく歩いていれば、問題の原因はMoho側のリギング設定（CSVのミスなど）にあることが即座に判断できます。
- **【今後の展望】:** 現在、`csv_viewer.py` にはさらなる**「診断ツール」**の搭載が予定されています。これは、PSD内の構造と `rigging_map.csv` の定義に矛盾がないかを事前にチェックし、「不可能な配線」を警告してくれる機能です（現在は開発中・未完）。完成すれば、独学者の試行錯誤はより一層、確信を持った制作へと変わるはずです。

Q4. タイムラインの途中で複数のモーション（CSV）を配置してベイクしても問題ありませんか？

- **答え:** 「はい、完全にサポートされています。複数のモーションを繋ぎ合わせた連続ベイクも安全に行えます。」
- **理由:** 最新の Step 4B. Bake & Bind ([ExecuteBakeMotion2D](#)) は、タイムライン上のマーカーに記録されたすべてのCSVパスと開始フレーム ([path|start](#)) を自動的にスキャンして一括処理する「マルチアクションベイク」を搭載しています。これにより、各アクション区間ごとに個別のCSVスケールデータと期間が適用され、タイムラインの途中から開始されるモーションであっても、骨格が崩れたりバラバラに散乱したりすることなく、自動的に正しいパースと伸縮・回転が適用されます。
- **作者の助言:** 異なるダンスやアクションをタイムライン上で順番に並べ、マーカーを配置するだけで、一括で綺麗な2Dリグアニメーションへと焼き付けることができます（ポーズ間補間を組み合わせたステージングも容易です）。ただし、リグの基本構造設計 (Step 1A~2B) やイラストの初期ポーズ位置調整 (Rest Pose) 自体は、変わらず「フレーム0 (セッアップモード)」が絶対的な基準点となりますので、セッアップ自体は0フレームで行う必要があります。

Q5. 遠近感（パース）が不自然に強すぎる、あるいは弱すぎる

- **原因:** 使用しているモーションデータ (TDPT等) の深度変化の激しさと、テンプレート側の基準値が一致していない可能性があります。
- **解決策:** [templates.json](#) の [rig_perspective_base_z](#) を調整してください。
 - パースを強くしたい (ダイナミックにしたい) 場合: 値を小さく (例: 4.0) 設定します。
 - パースを弱くしたい (平行投影に近づけたい) 場合: 値を大きく (例: 12.0) 設定します。
- **作者の知恵袋:** 「100と入力したのにあまり変わらない」という場合は、パースが弱まりすぎて変化が視認しにくくなっている可能性があります。まずは 4.0 ~ 10.0 の範囲で微調整するのが、最も効果を実感しやすい「スイートスポット」です。

Q6. Step 4A 実行後に出る警告の意味は何ですか？

- **答え:** 4A (ボーン生成) が完了した直後は、まだ画像がボーンに対して「正しく変形できる状態 (ベイク & バインド)」になっていないことを知らせる安全装置です。
- **解決策:** 引き続き **Step 4B 「Bake & Bind」** を実行してください。これにより、動きがリグに固定 (ベイク) され、画像が正しい重み付けで骨に吸着します。
- **中断する場合:** もし 4B を実行せずに作業を中断したい場合は、Mohoのメニューから **Menu > Bone > Reset All Bone Rigging** を実行してリグを一度リセットしてください。これを怠ると、手動でボーンを動かした際にメッシュが不自然に歪む (あるいは全く動かない) 状態が残ってしまいます。

Q7. タイムラインの途中 (例: 48フレーム等) から差し込んだアクションでバッチ切り出し (BatchGen) を実行しても良いですか？

- **答え:** 「バッチ切り出し (3B. BatchGen) 自体の作業は、必ず0/1フレーム起点 (同期状態) で行う必要があります。」
- **理由:** バッチ生成 ([BatchGen](#)) は、Moho上のマーカーのフレーム位置をそのまま参照して「生のBVHファイル」から指定範囲を切り出すバッチファイル ([RUN.bat](#)) を生成します。そのため、Moho上のタイムラインフレームと元のBVHのフレーム番号が1:1で一致していなければ、切り出される範囲がズレてしまいます。このため、BatchGen実行時には「Timeline Sync Warning」警告が表示され、同期を確認する仕組みになっています。
- **作者の助言:** ただし、バッチ生成によってタイムラインマーカーへ記録された後の「Bake & Bind (4B)」処理については、上述の通りタイムラインの途中のどのフレームからであっても正確に自動ベイクが実行されます。したがって、「アクションの切り出し (BatchGen)」は同期した状態で行い、生

成されてマーカーに紐づけられた後の「動きの焼き付け（Bake）」はタイムライン上の自由な位置に配置して実行する、という使い分けを意識してください。

Q8. 2D平面のキャラクターイラストが、モーション読み込み後に斜めに歪んだり、Hipボーンが真下を向いてしまいます

- **答え:** 以前のバージョンでは、3Dモーション（BVH）のX軸・Y軸の3D回転がそのまま適用されると、2Dイラストが斜めに歪んだり、Hipボーンがジンバルロックや特異点で瞬間反転して暴れる現象が発生していました。しかし、最新のバージョンでは投影角度抽出（行列変換）および二重のアンラップ（折り返し防止）処理が極めて堅牢になったため、暴れや意図しない反転は自動的に解消されています。
- **解決策:** 現在は、RigWeaverパネルの「**2D Mode**」チェックボックスを**OFF**（チェックなし=3Dモード）で実行することが、最も推奨される標準ワークフローとなっています。
 - 「**2D Mode**」が無効（チェックなし=3Dモード）の場合、3Dデータが持つ豊かな回転データ（X/Y/Z軸）が適正に反映され、3Dならではの立体的な伸縮（パース）やしなやかなねじれが2Dリグ上で綺麗に表現されます。
 - 「**2D Mode**」が有効（チェックあり）の場合は、3D的なパース歪みを一切出さず、完全にイラスト原画のフラットなシルエットのまま平面的に動かしたい特殊なケースでのみONにしてください（X軸・Y軸の3D回転が0度固定され、Hipには背骨への2D投影フォールバック角度が適用されます）。

このリファレンスガイドは、あなたの創作と共に進化し続けます。論理を学んだら、あとは思い切り「嘘」をつき、あなただけの物語を紡いでください。

8. プロフェッショナル技術仕様書（Detailed Technical Specifications）

より深いシステムの挙動や設定ファイルの論理を理解したい場合は、以下の各ツール専用の仕様書を参照してください。

- **LayerWeaver**：プロフェッショナル技術仕様書 ～ 島解析、閾値処理、命名工学の深層 ～
- **MotionWeaver**：プロフェッショナル技術仕様書 ～ 3D座標投影、ノイズ除去、マッピング論理 ～
- **RigWeaver**：プロフェッショナル技術仕様書 ～ 5段階パイプライン、自動バインド、階層最適化 ～