

RigWeaver：プロフェッショナル技術仕様書

はじめに：2Dアニメーション制作の「摩擦」を解消するための架け橋

[!IMPORTANT] サポート対象アセットに関する制約 本システム（RigWeaver）が公式にサポートするのは、**LayerWeaver**を用いて作成された比較的単純なレイヤー構造を持つ**PSDファイル**のみです。

他社製品（Cartoon Animator 5等）の**G3キャラクター等の高度に構造化されたテンプレートには対応していません**。それらの製品は独自の優れた設計思想を持っており、RigWeaverはその代替や互換を目指すものではありません。あくまでWeaverシリーズという特定のワークフローを補助するためのツールであるため、異なる設計のデータを読み込んだ場合の動作は保証されません。

RigWeaverは、Moho Pro環境においてWeaverシリーズ（Layer / Motion）の成果物を論理的に繋ぎ合わせ、リギングという煩雑な工程をスムーズに通過させるための「ワークフロー・ブリッジ」である。

本システムは、3Dの数学的な整合性をガイドとして活用しつつ、2Dアニメーション特有の「表現の嘘（パースの誇張、意図的な形状の歪み）」を、破綻なく共存させることを目的としている。従来の2Dリギングが膨大な手作業による試行錯誤に依存していたのに対し、RigWeaverはMotionWeaverから受け取った3D空間の動きを、2Dキャンバスの描画特性に合わせて適合させるための「計算の代行」を行う。これはキャラクターを自動で生み出す魔法の杖ではなく、クリエイターが表現の探求に集中できるよう、技術的なハードルを下げるための精密な「補助エンジン」である。

第1章：アーキテクチャと設計思想

1.1 3D物理座標と2D表現の融和

RigWeaverの核心は、3D空間上のボーン回転データを直接2D画像に適用するのではなく、一度「Viewerボーン」という3DガイドをMohoの3次元空間内に投影し、それを2Dの「コントローラーボーン」へと翻訳する多段マッピング構造にある。

この構造により、3Dデータが持つ正確な運動軌跡をガイドとして維持しつつ、2Dキャライラストに特有の「特定アングルでの腕の短縮」や「関節の意図的な位置移動」といった、3Dの物理制約では不可能な「絵としての美しさ」をコントローラー経由で自由に上書き・付加することが可能となる。システムは3Dデータを絶対的な制約とするのではなく、2D表現の自由度を拡張するための「確実な幾何学的基準」として位置づけている。

1.2 非線形・ステートベース・パイプライン

RigWeaverは、従来の「設計・構築が終わってから動かす」というリギングの線形手順を排し、制作の進捗率やアセットの完成度に合わせて、どの段階からでも介入可能な「ステートベース」の設計を採用している。

具体的には、プロジェクト開始直後に **Step 1 (Setup & Viewer)** を実行するだけで、リギング設定が未完了の状態でも、スケルトンのみを用いた「先行演出（アクションコンテ/プリビズ）」が可能となる。さらに、この段階でラフな画像アセット（PSD）が存在すれば、それをスケルトンと同じ動きをするレイヤーに擬似的に張り合わせることで、ボーン単体よりも視覚情報の多い「キャラクターによる演出検証（擬似プレビズ）」を先行して行うことができる。ただし、この手法で生成される動きはあくまで演出のタイミングや距離

感を確認するための「確認用」であり、正式なリギング（Step 2～4）を経て完成するプロダクション・リグとは区別して運用されるべきものである。

一方で、キャラクターが「棒人間」のような極めてシンプルな構造である場合、あるいは骨格の動きが画像の変形と1対1で対応するような同一のデザインの場合には、この初期段階の設定をそのまま本編のアニメーションとして転用・完結させることも可能である。各工程はMoho内の「ステート」を非破壊的に更新・上書きする形式をとるため、こうした初期検証——あるいは簡易制作——を経た後に、演出のタイミングを維持したまま正規のリギングへと精緻化、あるいは差し替えを完全に同時並行で進められる。

1.3 シンクロマニピュレーション（清書の継承）

RigWeaverにおいて「清書の継承」は、アニメーションの品質を極限まで高めるための基幹技術である。後述する **SmartSwitch** によってグループ化され、適正なバインド設定が行われたリグ階層において、内部の画像レイヤーは親ボーンの変換マトリックスを完全に継承するように設計されている。

この環境下で、Mohoのベクター機能を用いてラスター画像（PSDインポート画像）をなぞる「清書（インキング）」を行った場合、あるいは元のラスター画像そのものをボーンで変形させる場合であっても、描画要素は座標系の再計算を介さず、即座に親の「3D対2D翻訳マトリックス」と同期を開始する。この「描いた（あるいは配置した）瞬間に動きが宿る」性質により、アニメーターは複雑なバインドの再設定や重み付けの調整から解放され、純粋な「作画（Inking）」や「変形演出」の作業を通じて3Dの躍動感を2D表現へとシームレスに引き継ぐことが可能となる。

第2章：基幹データ物理仕様：templates.json

2.1 templates.json：リギング・カーネルとしての定数定義

本プロトコルにおいて、**templates.json** は単なるユーザー設定ファイルではなく、リグの「DNA（物理法則）」を定義するマスター・カーネルとして位置づけられる。RigWeaverは実行時にこのファイルを走査し、後述するモデル検知アルゴリズムによって最適な定数群をメモリ上に展開する。ここでの定義ミスは、座標変換の破綻や階層構造の崩壊に直結する。

2.2 モデル自律検知論理 (**bone_folder_main**)

RigWeaverは、Mohoプロジェクト内のボーングループ名をスキャンし、**bone_folder_main** キーに記述された文字列（例：**RL_Bone_HumanV2**）と完全に一致するグループを探索する。

- **機序**: この一致が確認された瞬間に、システムは対象のキャラクター構造（二足歩行、四足歩行等）を特定し、関連するすべての物理定数（バインド規則、仮想ボーン定義等）を適用する。
- **工学的価値**: これにより、ユーザーが手動でモデルプロファイルを切り替える必要がなくなり、一つのシーン内に複数の異種リグが混在していても、スクリプトが自律的に正しい演算エンジンを選択することが可能となっている。

2.3 空間トポロジーの補完と構築 (**virtual_bones**)

virtual_bones は、PSD（作画）側の構造的な制約を補い、1枚のイラストを3Dの動きに合わせてしなやかに変形させるための機能です。PSD内には **Torso** や **Hip** などのマーカー（関節位置）が配置され、位置も明確に規定されています。しかし、作画レイヤーとしては「胴体」が1枚の繋がった絵（シングルメッシュ）として描かれていることが多いため、物理的にパーツを **Torso** と **Hip** のレイヤーに振り分けることができません。この「1枚絵の胴体」に対して、3D（BVH）データの持つ細やかな屈曲を破綻なく適用するには、Flexi-

Bind（しなやかな結合）を用いて、複数のボーンの影響力で滑らかに変形させる必要があります。

virtual_bones は決して「何もないところから架空の骨を生み出す」わけではありません。PSDで規定された関節位置を手助けとし、このFlexi-Bindによる変形操作をMoho上で成立させるために必要なボーン階層（操作・構造用関節）を、適切に連結・補完する役割を担っています。

- **redirect_children の論理**: この属性は、特定の親ボーンが当初持っていた子供ボーン群を、補完された構造ボーン（例：**Torso**）の配下へと強制的に移譲させます。
- **機序**: Luaスクリプト内の **EnhanceRigHierarchy** 関数が実行される際、このルールに基づき階層ツリーのポインタを動的に書き換えることで、3Dの解剖学的構造を「2Dの1枚絵を動かす操作性」に適したトポロジーへと再編します。

2.4 ハイブリッド・バインド選別・強度定義 (**flexi_binds** / **flexi_force**)

RigWeaverは、レイヤーごとに「柔構造（Flexi）」か「剛構造（Layer）」かを自動判別するハイブリッド・バインド・エンジンを搭載している。

- **選別条件**: **flexi_binds** リストに含まれるレイヤーは、複数のボーンの影響を同時に受けるフレキシブル・バリエーションとしてマークされる。
- **強度勾配 (**flexi_force**)**: フレキシ・バインド対象となったボーン群に対し、**flexi_force**（または **hip_force** 等）で定義された「ボーン強度（Strength）」を自動適用する。これにより、身体の「芯」となる腰部は強く、腕や脚などの末端は適切な柔らかさで曲がるように物理的な影響範囲が自動計算される。

2.5 解剖学的エイリアスとオフセット補正 (**bone_aliases** / **bone_rot_offsets**)

3D空間の多義性と、2D描画時の「正解」のギャップを埋めるための数学的補正機能である。

- **bone_aliases (救済論理)**: 作画者がPSDで **UpperArm** と命名しても **Shoulder** と命名しても、エイリアスリストによって同一の解剖学的パーツとして認識し、**_rig.csv** との紐付けを保証する。
- **bone_rot_offsets (座標変換補正)**: BVHが持つ「腕の上げ方」の初期値と、Mohoキャンバス上での「デフォルトの腕の角度」にズレがある場合、ここで指定された角度（例：**LHand: 90**）を実行時にベクトル加算する。これにより、3Dの回転を2Dキャライラストに最適なポーズへと「融和」させる。

2.6 リソース・ディスカバリーと解決優先順位 (Resource Discovery & Resolution)

RigWeaver は、**templates.json** や **rigging_map.csv** といった基幹リソースを読み込む際、プロジェクトの整合性を担保するために以下の優先順位でディレクトリを自律的に走査する。

1. **プロジェクト・ディレクトリ (PSD フォルダ)**: 現在アクティブなアセット（PSD）が保存されている階層。プロジェクト固有のカスタム定義を最優先する。
2. **モーション・ディレクトリ (CSV フォルダ)**: インポートに使用したモーションデータが保存されている階層。
3. **システム・リソース・ディレクトリ**: **moho:UserAppDir()** / **Scripts** / **ScriptResources** / **rigweaver**。Weaver シリーズの標準設定が格納される最終的なフォールバック先。

この優先順位により、ユーザーはスクリプト本体（scripts/tool フォルダ）を汚染することなく、プロジェクトごとに最適なリギング設定を安全に切り替えることが可能となっている。

第3章：内部論理：幾何学マッピングと座標変換

3.1 階層構造の統計的推論エンジン (Inference Engine)

RigWeaverは、不完全なデータセット（一部の関節マーカーが欠落した PSD 等）からでも、頑健なボーン階層を再構築するための推論エンジンを搭載している。

- **機序**: PSD 内のレイヤー名（@ による帰属、> による入れ子）および `rigging_map.csv` の定義を多重にスキャンし、ボーン間の親子関係を決定論的に導き出す。
- **ループ・ガード**: 推論プロセスには深度優先探索（DFS）が用いられ、50 階層を超える循環参照を検知した場合に処理を強制中断するリミッターが実装されている。これにより、複雑な PSD 構造における Moho のクラッシュを未然に防いでいる。

3.2 ピボット自動探索とフォールバック救済論理

ボーンの回転中心となるピボット（関節点）を特定するため、システムは「階層的マーカー探索」を実行する。

- **探索順序**: まず指定された関節名（例：`LShoulder`）と一致する空レイヤーまたはフォルダを探索し、存在しない場合はその親要素の重心、あるいは `templates.json` 内の `bone_aliases` で定義された代替ボーン（例：`LArm`）の基点を参照する。
- **救済論理**: 主要な関節（例：`Torso`）が物理的に存在しない場合、システムは `Chest` または `Spine` をルート座標として自動的に割り当てる「ピボット・フォールバック」を発動し、リギングの完全な停止を回避する。

3.3 マトリックス・ベーキング (Matrix Baking) と座標系リセット

Step 5 (Finalize) で実行される `ExecuteFlattenRig` は、リギング作業用の中間階層を破壊し、プロダクション向けに座標系をフラット化（正規化）する高度な行列演算プロセスである。

- **ベーキングの機序**: 各レイヤーの「見かけ上のトランスフォーム (World Matrix)」をキャプチャし、親階層を解除した後のルート座標系において、その同一の位置・角度・スケールを維持するように逆算して再配置する。
- **成果**: これにより、複雑な親子関係に依存していた動きがすべてワールド座標（またはルートボーン）基準に「焼き付け」られ、書き出し後のデータ互換性と再生安定性が極めて高いレベルで保証される。

3.4 2D翻訳エンジン：3D角から2D角への空間転写演算

Step 4A において、3D空間上のクォータニオン/オイラー角を 2D の回転角へと転写する「翻訳エンジン」が動作する。

- **転写演算**: 単なる角度のコピーではなく、Step 2B (Align Base Pose) によって同期されたイラスト原画の初期姿勢を基準点（0度）とし、そこからの 3D ボーンの回転偏差に対し `bone_rot_offsets` をベクトル合成する。
- **2.5D表現の正当化**: 3D 空間では Z 軸回りの回転であっても、2D 上では X 軸や Y 軸の投影角度として解釈し直す「空間的な嘘」を数学的に正当化し、イラストの破綻を最小限に抑えた滑らかな動きを実現する。

UIの「2D Mode」チェックボックスによる3Dベイクの自動制御

RigWeaverの操作パネル上に用意された「**2D Mode**」チェックボックス (**is2DMode**) は、インポートおよびリターゲット時における3D回転成分のベイク挙動を決定論的に制御する中心的なUIスイッチである。最新のバージョンでは、空間投影およびアンラップ（角度の折り返し防止）処理が極めて堅牢になったため、「**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る。これにより、ユーザーが設定ファイル (**templates.json**) を手動で書き換えることなく、以下の挙動が自動で切り替わる。

- 「2D Mode」無効時 (チェックなし / 3D Mode) 【標準推奨】**：3Dモーションが持つ豊かな回転データ (X/Y/Z軸、Pitch/Yaw/Roll) がすべて正規にベイクされる。3Dの物理的に正しい動きがそのまま流し込まれ、2Dリグ上での立体的な伸縮（パース）や、しなやかな身体のねじれが極めて自然かつダイナミックに表現される。空間的にどんなに複雑なねじれが発生しても、行列投影と二重アンラップ処理によって、骨◆### 6.1 タイムライン・マーカー：演出の「切り出し」と指示書定義 タイムライン上のマーカーは、演出情報を **BatchWeaver** (CLI) へと繋ぐための「論理的なスイッチ」として機能すると同時に、**Bake & Bind (4B)** における複数アクションの適用タイミングを伝達する情報源としても機能する。
- マーカーの有効条件**: RigWeaver は、ラベル（名称）が付与されたマーカーを解析対象とする。
 - ラベル（名称）の存在**: ファイル名決定やアクションCSVの探索のため、名前のないマーカーは無視される。
 - 出力種別の判定とポーズ間自動補間 (Pose-to-Pose)** :
 - 動画 (CSV/Batch)** : ドラッグして有効な幅 (Duration > 0) を設定したマーカーが対象となる。
 - 静止画 (PNG) と1フレームCSV**: ラベルが存在すれば、幅の有無に関わらず PNG 書き出し候補 (**pendingPNGs**) としてスタックされる。特に持続時間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仕様となりました。これにより、アニメーターは3Dの代表的なキースタンスやキーポーズのみを狙って抽出し、タイムライン上の各時間に配置することで、Moho独自の強力なボーン変形・補間機能に繋いで「**Pose-to-Pose (ポーズ間自動補間) によるキャラクター制御**」を行うことが可能です。膨大なフレームを一括ベイクする負荷を回避しつつ、決めのキーポーズだけを自動流し込みして手動で細部を調整するような、極めて高効率で手戻りのない2Dアニメーション制作が実現します。
- 静止画自動出力機序 (**we_makepng.lua**):**
 - ExecuteBatchGen** の完了直後、有効なラベル付きマーカーが検出されている場合、外部ヘルパー脚本文 **we_makepng.lua** が動的に呼び出される。
 - このスクリプトは、Moho の **fTimelineMarkers** チャンネル（ドキュメントおよび選択レイヤー）を自律的に走査し、指定された全フレームのレンダリング (**FileRender**) をプロジェクトディレクトリに対して一括実行する。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に末尾に **_p** が自動付与されたファイル名 (**{ラベル名}_p_f{フレーム数}.png**) として整理出力される。
- データ取得の一貫性 (Controller への固定)** :
 - マーカーはドキュメントのグローバルタイムラインだけでなく、各レイヤー上（例：**story1**, **story2**）に個別に配置することも可能である。
 - いずれの場合も、アニメーションデータ（カメラ角度およびZ回転）は常に **3D_Angle_Controller** から一貫して取得されるため、管理レイヤーの多層化による不整合を排除している。
- 演出のセグメント化**: マーカーの「ラベル名」はそのまま最終的な出力ファイル名となり、「マーカーの幅（長さ）」がシーンの尺（開始・終了フレーム）として定義される。制作者はタイムライン上で必要な範囲にマーカーを打つだけで、演出プランの「切り出し」が完了する。

- **指示情報の全自動走査:** `ExecuteBatchGen` 関数が実行されると、システムはタイムライン全体を高速スキャンし、全マーカー区間の「開始フレーム」「終了フレーム」「レイヤー回転（アングル）」の情報を抽出し、正規化された CSV 指示書へと変換する。`jectedZAngle` 関数は `layer.GetFullTransform` を経由して演算を行うため、ワークスペースの回転状態（ビュー行列）が計算結果に直接反映される。これは意図しない歪みを防ぐための「安定化」を目的としているが、副次的に 5.4 節で述べる「パース・ベークキング」という演出技法を可能にしている。

3.5 PSD を「グラウンド・トゥールース（地上真実）」とする設計の代償

RigWeaver の設計哲学における最大の特徴は、**「PSD 内のレイヤー（島）とマーカーの配置を、リグ構築における絶対的な正解（グラウンド・トゥールース）」**として扱う点にある。この思想は作画の自由度を保証する一方で、工学的には以下の特殊な課題を発生させる。

- **ボーンの過不足問題 (Bone Starvation):** 2D イラストにおいて「胴体 (Torso)」などは通常一つの巨大な画像として描かれる。PSD のマーカー配置（例：腰と胸のみ）に厳密に従いすぎると、その巨大なメッシュに対してボーンが1本しか割り当てられず、体を反らせる、あるいは捻るといった「有機的な曲がり」に必要な関節の回転軸が物理的に不足する事態に陥る。
- **仮想ボーンによる外科的解決:** この「解剖学的不足」を補うため、`templates.json` の `virtual_bones` を用いた**「ボーンの外科的増築」**が必要となる。これは 3D (BVH) の論理を写し取るためだけでなく、PSD という 2D の肉体が持つ「メッシュ変形リソースの不足」を、システム側で補完・拡張するための工学的必然から生まれた仕組みである。
- **トポロジーの再マッピング:** 増設された仮想ボーンは、3.4 節で述べた行列演算（2D 翻訳）と 3.3 節のベークキングプロセスにおいて「中間関節」として機能し、1枚のメッシュを多段的に屈曲させるための回転ベクトルを提供する。このプロセスにより、作画を複雑にすることなく、プロダクション級の滑らかなアニメーション密度を実現している。

3.6 ワークスペース正規化と初期スケーリングの工学的要請

RigWeaver の演算体系において、**「Moho ワークスペース（高さ約 1.6 ユニットの投影領域）へのスケール正規化」**は、座標系の整合性を維持するための絶対的な前提条件である。

- **空間の投影仕様:** `ExecuteImport3D` 関数は、BVH ソースの絶対的な高さをスキャンし、Moho の標準的な画角に収まるように $POS_SCALE = 1.6 / height$ という定数を動的に算出して Viewer を投影する。これは、3D 空間の運動エネルギーを 2D キャンバスの有効描画範囲へと最適にマッピングするための空間正規化プロセスである。
- **パース計算の正規化:** 上記の正規化（高さ 1.6）を前提として、Camera Z は $(rig_perspective_base_z / Magnify) / POS_SCALE$ の式で算出される。この正規化プロセスにより、キャラクターの物理的な大きさが異なっても、`rig_perspective_base_z` という単一の定数によって一貫した遠近感の制御が可能となっている。
- **基準座標系の合致:** Moho にインポートされた直後の PSD アセットは、元の解像度や DPI に依存した任意（非正規）のサイズで配置される。リギングを開始する前（Step 0）にイラストをワークスペースのサイズに手動で適合させることは、システムの「演算基準座標系」と PSD の「実体座標系」をシンクロさせる重要な儀式となる。
- **不整合のリスク（階層の崩壊）:** この初期スケーリング（正規化）を怠った状態でリギングプロセスを強行した場合、マーカーの検出位置や階層構築時の相対ベクトル計算に累積的な数値誤差が生じる。結果として、「レイヤーの異常な位置への転移 (Layer Shifting)」や「骨の生成位置の致命的なズレ」といった、論理的な不整合によるリグの崩壊を招くことになる。

- **セットアップポーズの保護 (Setup Pose Protection):** システムの演算基準となる「0フレーム（セットアップフレーム）」への意図しない書き込みを防止する物理的なガードが実装されている。万が一、タイムラインが0フレームにある状態でモーションの流し込み（Step 3）を実行しても、内部的に「1フレーム目からの適用」へと自動補正される。これにより、アーティストが設定したイラストの「原型（Setup Pose）」の座標系が、3Dの数学的な運動データによって上書き破壊されるリスクを工学的に回避し、リグの永続的な安定性を保証している。

3.7 1B/2B 同期：3D 投影と 2D リグを繋ぐ数学的ブリッジ

RigWeaver のワークフローにおいて、**Step 1B「Bind Character」**と **Step 2B「Align Base Pose」**の連続性は、リグの物理的な安定性を決定づける極めて重要なフェーズである。

- **1B の役割（論理的接着剤）:** Step 1B は、Viewer ボーンという 3D 空間上のガイドに対し、2D の画像アセットを数学的に紐付けるプロセスである。この段階で、画像は単なる「キャンバス上の絵」から、3D 空間のトランスフォーム・マトリックスに従属する「リグの構成要素」へと昇格する。
- **2B の役割（幾何学的アライメント）:** Step 2B は、1B で紐付けられた画像を、イラストの原画ポーズ（Base Pose）に合わせて最終的なアライメント調整を行う。
- **1B スキップのリスク（座標系の乖離）:** 1B をスキップして 2B を実行した場合、画像は 3D 空間の座標系に属さない「浮いた状態」のままアライメント演算に投入される。この結果、3D 側の運動ベクトルと 2D 側の描画座標が計算上で一致せず、キャラクターがバラバラに散乱する、あるいはリグが崩壊するという致命的な不整合が発生する。
- **T-Pose 例外の理由:** キャラクターが完璧な T-Pose（腕が水平、脚が垂直）である場合、3D Viewer と 2D リグの初期座標系が数学的に一致（ゼロ・オフセット）するため、1B をスキップしても偶然整合性が保たれることがある。しかし、これは「座標のズレがたまたま 0 だった」に過ぎず、ポーズが少しでも異なる場合には通用しない「例外的な状況」である。安定したリギングのためには、T-Pose であったとしても 1B を実行し、論理的なバインドを確立させることが工学的な鉄則である。

3.8 「Frame 0」絶対ルールの工学的背景

Moho Pro における **Frame 0（セットアップモード）**は、単なるタイムラインの開始点ではなく、リグを構成する全要素の「設計図（REST Pose）」が格納される特殊な領域である。

- **不変の基準点:** RigWeaver のすべての構築プロセス（骨の生成、ピボットの決定、バインドの確立）は、この Frame 0 の状態を「絶対的な正解」として参照する。
- **1フレーム以降の影響:** 1 フレーム目以降に記録されるデータは、Frame 0 からの「変化（Delta）」でしかない。もし Frame 0 が 3D データの流し込み等によって汚染・改変された場合、その後に積み上げられるすべてのモーションデータは「狂った基準点」からの偏差となり、リグは永久的に予測不能な挙動を示すことになる。
- **システムによる保護:** RigWeaver はこの致命的なリスクを回避するため、Step 3 以降のモーション適用において 1 フレーム目への自動シフトを強制している。Frame 0 を「聖域（Sanctuary）」として保護し続けることこそが、複雑な 3D 演技を 2D リグで再現するための物理的な根幹である。

[!CAUTION] タイムライン途中へのモーション配置とベイクの安全性、およびバッチ切り出し

（BatchGen）における同期要件 最新のバージョンでは、タイムライン上の任意の位置に配置された複数のモーションを自動スキャンし、各アクションごとの期間や CSV スケールデータに基づいて一括でベイク・バインドする「マルチアクションベイク」機能が Step 4B. Bake & Bind に実装されました。これにより、かつて存在した「タイムラインの途中（1フレーム目以降）でデータを書き込むと

リグの整合性が崩壊する」というリスクは技術的に完全に克服され、タイムライン上のどこからでも安全に異なるアクションを流し込み・ベイクできるようになりました。

ただし、バッチ生成 (**BatchGen**) においては、依然として「**フレームインデックスのオフセット同期**」の制約が存在します。**BatchGen** は Moho のタイムラインマーカー位置を元に、生のBVHファイル (3Dモーション) から該当アクション区間を切り出す命令を生成します。そのため、Moho上のタイムラインフレームと生BVHのフレーム番号が「1:1で完全に同期している (0/1フレーム起点)」ことを前提として計算を行います。もし、タイムラインの途中 (例: 48フレーム目) からretargetしただけのズレた状態でマーカーを打つと、BVHから誤った時間軸のモーションが射出される原因となります。

【安全回路：Timeline Sync Warning の実装】: システムは **BatchGen** 実行時にこの不整合を防ぐため、ファイル選択直後に「**Timeline Sync Warning**」英語警告ダイアログを表示し、ユーザーに同期状態を確認させるガードを設けている。

つまり、「アクションの切り出し (3B. **BatchGen**)」の作業は同期した環境 (または0/1フレーム起点) で行う必要がありますが、切り出されてマーカーにCSVパスが紐づけられた後の「2Dボーンへのベイク・バインド (4B. **Bake**)」については、タイムライン上の自由な位置に配置して一括実行することが可能です。この挙動の使い分けを理解することが、効率的な複数アクションのベイクにおける鉄則である。

3.9 リターゲット時のボーン長不整合に対する自律回避と安定化論理 (胴体崩壊防止とハイブリッド・スケーリング)

3Dモーションデータ (CSV) が持つ骨格比率と、2Dキャラクターイラスト (PSD) から構築された2Dリグの骨格比率 (特に胴体の長さなど) に乖離がある場合でも、RigWeaverはリグを崩壊させることなく高精度なモーション流し込みを完了する「自律回避・安定化論理」を備えている。

- **リターゲット時の中間関節位置 (Translation) のロック機序**: **Step 3. Retarget Motion** の実行時、システムは基準点である「Hip (腰)」、「Root (根元)」、および親ボーンを持たない「孤立ボーン (Orphan)」のみに位置移動 (Translation) の値を流し込む。それ以外の腕、脚、脊椎 (胴体) などの中間ボーン群に対しては、親ボーンからの相対的な位置変化を一切適用せず、接続箇所固定のまま「Z軸周りの回転 (Rotation)」のみを伝達・反映する。これにより、元の3Dデータの胴体や肢体の長さが2Dキャラクターとどれだけ異なっても、2Dキャラクターの肉体 (比率) が強制的に引き伸ばされてバラバラに散乱したり、関節位置が狂って崩壊したりするのを論理的に防止する。
- **初期ポーズ (レストポーズ) を基準としたダイナミック伸縮演算**: **Step 4B. Bake Motion 2D** の工程において、システムはアニメーション中の肢体セグメントに対して動的な伸縮 (スケーリング) を施す。この伸縮率は、3D (CSV) 側の生の位置データから直接算出されるのではなく、まずセットアップフレーム (フレーム0) における2Dリグ上の各ボーンの「初期設計距離 (静止距離)」を計測してキャッシュする。その後、各フレームにおける3Dガイド上の対応するボーン間距離を、この初期設計距離で除算して「伸縮比率 (倍率)」を動的に逆算する。この倍率を2Dリグのボーンスケールに適用するため、CSV側のボーンが元々どれほど長かったり短かったりしても、フレーム0でクリエイターが設計した2Dキャラクターの初期比率が「基準 (等倍 = 1.0)」として厳密に維持される。これにより、モデル全体の崩壊を防ぎながら、3Dアニメーションが本来持っている遠近 (パース) に伴う肢体の動的な伸縮のみが適切にリグに還元される。

第4章：運用工学：多段型検証ワークフロー

RigWeaver は、アセットの完成度に応じて検証の深度を切り替えられる「多段型プロトコル」を採用している。これにより、リギングの完成を待たずに演出を確定させ、手戻りを最小限に抑えることが可能となる。

4.1 フェーズ1：ボーン・プレビズ (Skeleton-Only Previs)

制作の最も初期段階、あるいはキャラクターデザインの確定前に実行されるフェーズである。

- **機序:** 1A. Load 3D Data により `_3d.csv` の動きを Viewer ボーンに直接投影する。
- **目的:** 演技のタイミング、カメラワーク、および空間的な配置の先行確定。
- **工学的価値:** キャラクターアセットが一切存在しない状態でも、実際のモーションデータを元にした「動くビデオコンテ」として活用でき、演出上の致命的なミスを上流工程で排除できる。

4.2 フェーズ2：レイヤー・プロキシ検証 (Layer-Driven Mockup)

キャラクターのラフデザインが完成した段階、あるいは仮のアセットを用いて「画面の厚み」を検証するフェーズである。

- **機序:** スケルトンと同等の動きをする仮のレイヤー（プロキシ画像）を張り合わせ、3D の Z 軸（奥行き）に基づいた重なり順の妥当性を検証する。
- **目的:** シルエットの確認、キャラクター同士の接触判定、および奥行き表現の最終確認。
- **工学的価値:** 正規のリギング（1A～4B）を介さずとも、キャラクターを用いた具体的な画面構成の検討が可能となり、「描画範囲の不足」や「アライメントの不整合」を早期に発見できる。

4.3 フェーズ3：プロダクション・リグと清書シンクロ (Full Production)

全アセットが揃い、最終的なアニメーションを出力する完成フェーズである。

- **機序:** 1A～5 の正規プロセスを完遂し、SmartSwitch によってパッキングされた最終アセットに対してハイブリッド・バインドを適用する。
- **シンクロ・インキング:** 本フェーズの主な利点は、ボーンに紐付けられた画像グループ内で行うベクター清書である。描き込まれたベクター線は、親が持つ 3D 変換マトリックスを「無遅延・無再計算」で継承し、瞬時に 3D モーションの躍動感作画へと転写する。
- **工学的価値:** 技術的なリギング設定が「表現のための透明なインフラ」へと消化され、制作者は最終的な「描き込み」による情緒的な演出に全リソースを集中させることができる。

第5章：演出工学：ビデオコンテとアングル解析

Weaver プロトコルにおける Moho の役割は、単なるアニメーション出力機ではなく、演出を視覚化し、作画を導くための「デジタル・ディレクターズ・コンソール」である。

5.1 3D ポーズによる「動く下絵」の生成とプレビズ

RigWeaver の 1A. Load 3D Data および 3. Retarget Motion を経ることで、3D 空間上の演技データは 2D キャンバス上の「動くスケルトン」として実体化される。

- **実務上の機序:** このスケルトンは解剖学的に正しい 3D の運動軌跡を保持している。ここに 4.2 節で述べたラフ PSD を重ね合わせることで、最終的なキャラクターの量感や動きのタイミングを精緻に検証する「動く下書き（ビデオコンテ）」が完成する。

- **作画へのフィードバック:** アニメーターはこの動くスケルトンを「土台」とすることで、複雑な短縮ポーズや関節の動きに悩まされることなく、演出意図に基づいた確信のある作画に集中できる。

5.2 コントローラによる俯瞰・煽りの物理：空間投影の「嘘」を制御する

RigWeaver は、3D 空間の絶対的な回転から「垂直方向のチルト（煽り・俯瞰）」と「水平方向のアングル」を分離抽出し、レイヤーの回転とコントローラーボーンの角度差分（Delta）として再計算する。

- **空間投影の数学:** MotionWeaver が算出した 3D->2D 投影情報を、RigWeaver は 1 枚の画像の「擬似的なパース変形」として Moho 上に展開する。
- **感性による微調整:** コントローラーボーンを操作することで、3D 的な「正しさ」を維持したまま、2D 画として最も迫力が出る「煽りの角度」を 0.1 度単位で演出的に微調整することが可能である。

5.3 職人の領域：ガイドに基づく「煽り画」の再編成（LayerWeaver への再帰）

Weaver シリーズの最も効果的な運用法は、この動くガイドを基に再び LayerWeaver (Krita) へ戻り、「特定のポーズに最適化された煽り/俯瞰パーツ」を描き直すという再帰的なフローに存在する。

- **プレビュー機能によるガイド抽出:** Moho のプレビュー機能（File > Preview）を実行することで、現在のフレームのレンダリング結果を別ウィンドウで確認できる。このウィンドウのメニューを使用することで、様々な形式での画像保存やクリップボードへのコピーが可能となる。
- **ワークフローの再帰性:** 抽出した画像を Krita のレイヤーとして配置（あるいは貼り付け）し、その骨格を基準として煽りや俯瞰に特化したパーツの清書を行う。完成したパーツは LayerWeaver を通じて再び Moho へインポートされ、自動バインドによって 3D の動きにシンクロする。
- **工学的成果:** これは単なる「自動化」ではなく、3D ガイドという名の「確かな補助線」を作画者に提供し、作画の限界を 3D の論理で押し広げるためのプロセスである。3D の整合性と、人間の手でしか描けない「情緒的なパースの嘘」が、プレビュー画像という媒体を介して高次元で融合する。

5.4 視点依存のパース・ベーキング (Perspective Baking)

RigWeaver の投影アルゴリズムは、単なる 3D 空間の転写ではなく、「Retarget 実行時のワークスペース（オービット）の角度」を 2D リグの基準ポーズとして焼き付ける特殊な特性を持っている。

- **機序と演出効果:**
 - Step 3. Retarget Motion の実行前に、Moho のワークスペースを「少し斜め」に回転させておくと、システムはその時点での投影図を 2D キャンバス上の正面として解釈する。
 - 軌道を 0 に戻した際、リグにはその斜め視点のパース（意図的な歪み）が残る。これにより、Moho のカメラを固定したまま、キャラクターに「常に特定のアングルを向いている」という 2D アニメ的な嘘を内封させることが可能となる。
- **工学的な運用限界と特性:**
 - **適用範囲:** 本機能は「少し斜め」や「緩やかな俯瞰・煽り」において最も安定して動作する。
 - **深度の制約:** 極端な俯瞰（真上からの視点等）には対応していない。これは、3D の垂直ベクトルが画面上で点に収束し、2D 平面への角度投影が幾何学的に破綻するためである。
 - **活用の推奨:** キャラクターの立ち姿に僅かな「奥行き感」を与えたい場合や、アクションのインパクトを強めるための「わずかなパースの誇張」に用いるのが、本システムの設計思想に最も合致した運用法である。
- **パース強度の調整と逆算ロジック:**
 - リギング時、レイヤーの「拡大率 (Magnify)」が Camera Z に変換される際の基準として `rig_perspective_base_z` が参照される。

- **工学的特性:** この値が小さいほど（例: 4.0）、同じレイヤー拡大率でも Camera Z が近く（パースが強く）なり、よりダイナミックな遠近感が得られる。TDPT 等の奥行き変化が激しいソースを使用する場合は、この値を 4.0 に設定することで、Moho 上の見た目と実際の 3D 深度の乖離を最小限に抑え、意図した通りのパース表現をリグに内封させることが可能となる。

5.5 ハイブリッド・スケーリング：短縮効果とパースの分離定義

RigWeaver v1.0.4 以降、リグの各部位に対して以下の 2 種類のスケーリング論理を使い分ける「ハイブリッド・スケーリング」が導入されている。

- **肢体セグメント (Limb Segments):** 腕、脚、脊椎など。これらの中間ボーンには、3D 投影距離から算出される「短縮効果 (Foreshortening)」のみが適用される。これにより、パーツがカメラに近づいても画像が不自然に太く（均等拡大）ならず、パースによる「長さの変化」のみが表現される。
- **先端セグメント (Terminal Segments):** 手、足、頭、および末端ボーンの親。これらには、CSV データに基づく「パース・スケーリング (Perspective Scaling)」が、専用のピンボーン (`_Scale`) を通じて適用される。
- **工学的成果:** この分離により、「腕の幅を変えずに、手先だけをパースで大きく見せる」という、2D アニメーション特有のダイナミックな誇張表現が、3D データの整合性を維持したまま自動化される。

RigWeaver における自動化の活用法は、Moho を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統括する**「デジタル・ディレクターズ・コンソール」**として運用する点にある。これは個人環境において制作スピードを向上させるための量産工学である。

6.1 タイムライン・マーカー：演出の「切り出し」と指示書定義

タイムライン上のマーカーは、演出情報を `BatchWeaver` (CLI) へと繋ぐための「論理的なスイッチ」として機能する。

- **マーカーの有効条件:** RigWeaver は、ラベル（名称）が付与されたマーカーを解析対象とする。
 - **ラベル（名称）の存在:** ファイル名の決定論的命名のため、名前のないマーカーは無視される。
 - **出力種別の判定:**
 - **動画 (CSV/Batch) :** ドラッグして有効な幅 (`Duration > 0`) を設定したマーカーが対象となる。
 - **静止画 (PNG) :** ラベルが存在すれば、幅の有無に関わらず PNG 書き出し候補 (`pendingPNGs`) としてスタックされる。
- **静止画自動出力機序 (`we_makepng.lua`):**
 - `ExecuteBatchGen` の完了直後、有効なラベル付きマーカーが検出されている場合、外部ヘルパースクリプト `we_makepng.lua` が動的に呼び出される。
 - このスクリプトは、Moho の `fTimelineMarkers` チャンネル（ドキュメントおよび選択レイヤー）を自律的に走査し、指定された全フレームのレンダリング (`FileRender`) をプロジェクトディレクトリに対して一括実行する。
- **データ取得の一貫性 (Controller への固定) :**
 - マーカーはドキュメントのグローバルタイムラインだけでなく、各レイヤー上（例：`story1`, `story2`）に個別に配置することも可能である。
 - いずれの場合も、アニメーションデータ（カメラ角度および Z 回転）は常に `3D_Angle_Controller` から一貫して取得されるため、管理レイヤーの多層化による不整合を排除している。

- **演出のセグメント化:** マーカーの「ラベル名」はそのまま最終的な出力ファイル名となり、「マーカーの幅（長さ）」がシーンの尺（開始・終了フレーム）として定義される。制作者はタイムライン上で必要な範囲にマーカーを打つだけで、演出プランの「切り出し」が完了する。
- **指示情報の全自動走査:** `ExecuteBatchGen` 関数が実行されると、システムはタイムライン全体を高速スキャンし、全マーカー区間の「開始フレーム」「終了フレーム」「レイヤー回転（アングル）」の情報を抽出し、正規化された CSV 指示書へと変換する。

6.2 CLI による 100 カット自動射出：RUN.bat 生成の物理

Moho から射出されるのは単なるデータではない。それは巨大な工場を動かすための「点火スイッチ」である。

- **RUN.bat (実行バッチファイル) の自動生成:** 演出情報を統合した `RUN.bat` と CSV 指示書がカレントフォルダに生成される。このバッチファイルは、MotionWeaver に対する一括コマンド命令群を内包している。
- **一括指示出しの工学的成果:** 生成された `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。指定された数十種類以上のアングル、数百フレームにおよぶシーンが、制作者の介在なしに次々と書き出される。
- **量産のパラダイムシフト:** 演出を Moho で固め、書き出しをシステムに完全委譲するこのフローにより、従来の 2D アニメーション制作では不可能だった「物量とクオリティの両立」が、一晩の自動処理によって現実のものとなる。

第7章：Weaver_SmartSwitch：バインド継承の物理

`Weaver_SmartSwitch` は、単なるレイヤーのパッキングツールではなく、RigWeaver で構築された**「バインド情報を上位階層へ昇格し、座標マトリックスを恒久的に伝播させる」**ための知的な変換エンジンである。

7.1 バインド DNA の「昇格」と階層トポロジーの強化

通常、Moho において個別にバインドされたレイヤーを単純なグループに入れると、バインド情報の再設定が必要になるか、構造が不安定になる。SmartSwitch はこの問題を工学的に解決する。

- **バインド情報の自動昇格 (Promotion):** スクリプト実行時、各 `@タグ` レイヤーが保持していたバインド情報（どのボーンに紐付いているかという DNA）をスキャンし、生成された親グループ（Group/Switch）に対してその情報を自動的に再適用（昇格）させる。
- **階層統治 (Governance):** 親階層がボーンに強固に紐付けられることで、その内部は一つの「独立した座標空間」として定義される。これにより、内部のレイヤー群は個別のバインド設定を意識することなく、親が受ける 3D/2D 変換マトリックスを 100% 透過的に継承する。

7.2 スイッチ・レイヤーによる多相的演技制御

SmartSwitch によって生成された Switch レイヤーは、単なる「表示切り替え」以上の役割を担う。

- **多相的なシンクロ論理:** 「前・横・後」のアングル違いや、複雑なリップシンクの各コマが、すべて同一の「親のバインド情報」の下で統治される。これにより、どのコマに切り替わった瞬間でも、解剖学的に完璧な位置でボーンの動きに同期することが数学的に保証される。
- **演出の解放:** アニメーターは「リギングの崩れ」を一切懸念することなく、スイッチの切り替えのみによってキャラクターの多面的な演技を制御できる。

7.3 清書シンクロの極致：なぜ「描き直した瞬間」に動くのか

本システムの最大の工学的成果は、後次的なベクター清書に対する「無遅延の動きの転写」である。

- **マトリックス継承の恩恵:** SmartSwitch で統治されたグループ内に新しいベクターレイヤーを作成し、清書（インキング）を行う。この新しいレイヤーは、作成された瞬間に親グループが持つ「ボーンへの配線パス」を継承する。
- **情緒的な嘘と論理の融合:** 3D ガイド（プレビュー画像等）をなぞって描かれたベクター線は、親階層を通じて 3D モーションの躍動感を直接受け取る。これにより、「3D の確信に満ちた動き」と「2D の職人が描く緻密な線」が、一切の再リギング工程を挟まずに完全融合する。

7.4 ボーン主導の新規パーツ錬成論理 (Bone-Driven Instantiation)

SmartSwitch v3.2 以降、既存レイヤーのパッキングに加え、アクティブなボーンを起点とした新規パーツの動的生成機能が実装された。

- **自律判定ロジック:** スクリプト実行時、レイヤー選択数が 0 またはボーンレイヤーのみである場合、システムは自動的に「ボーン選択モード」へ遷移する。
- **トポロジーの強制正規化:** 生成された新規コンテナ（Group/Switch）は、`moho.layer` ではなく、スケルトンの所有主であるボーンレイヤーの直下に強制的に配置される。これにより、ユーザーによるレイヤー選択の不備をカバーし、ボーン制御が確実に適用される階層構造を維持する。
- **バインド情報の即時注入:** コンテナ生成と同時に、選択されていたボーン ID が `SetLayerParentBone` を通じて即座に適用される。内部に作成される空のベクターレイヤーは、この親コンテナの座標マトリックスを 100% 継承するため、リギング済みの既存モデルに対して、アクセサリ等の後付けパーツを幾何学的な誤差なしに即時追加することが可能となっている。

第8章：技術定義対照リファレンス

UI ボタン名	内部関数	物理的役割・工学的価値
1A. Load 3D Data	<code>ExecuteImport3D</code>	3D 運動データのインポート、および Viewer ボーン（3D ガイド）による空間座標の可視化。
Set Thickness	<code>ExecuteSetThickness</code>	「流し込み」演算における深度（厚み）の適用。Viewer ボーンの Z 軸偏差による空間的実体化。
1B. Bind Character	<code>ExecuteBindImages</code>	生成された Viewer ボーンの座標系に対して、PSD アセットを物理的に紐付け。
2A. Build Hierarchy	<code>ExecuteBuild3D</code>	<code>templates.json</code> の推論規則に基づき、正規の 3D スケルトン階層を決定論的に構築。
2B. Align Base Pose	<code>ExecuteAlignMarkers</code>	イラストの原画姿勢とボーンのレストポーズを同期させ、幾何学的アライメント誤差を較正。
3. Retarget Motion	<code>ExecuteRetarget</code>	3D ガイドの運動エネルギーを正規ボーンへと「転写」。注入位置（開始フレーム）や既存データの消去を制御可能。 ※0フレーム以外での実行はリグ崩壊のリスクあり。

UI ボタン名	内部関数	物理的役割・工学的価値
4A. Build 2D Rig	ExecuteCreateBones2D	3D 空間の回転成分を 2D キャンバスへと「モーション流し込み」し、アニメーション制御用コントローラーを動的に生成。
4B. Bake & Bind	ExecuteBakeMotion2D / BindCleanup	2D 空間への演技の「焼き付け (Baking)」、およびハイブリッド・バインドによる肉体と骨格の物理的同期。肢体セグメントには「短縮効果」を、先端セグメントには <code>_Scale</code> ボーンを介した「パース・スケール」を個別に適用する。
(補足) <code>_Scale</code> ボーン	-	先端パーツのパース変形用ピンボーン。中間パーツ（腕など）の予備ボーンは、時計や装飾品など「形を崩したくない小物」のバインド先として活用可能。

| 5. Flatten Rig | `ExecuteFlattenRig` | リギング作業用の中間階層を破棄し、座標系をワールド基準に正規化（マトリックス・ベークイング）。 || **Smart Switch** | `SmartSwitch:Run` | バインド DNA の昇格・物理伝播、およびプロダクション向けのユニット化（スイッチ化）。 || **Batch Gen** | `ExecuteBatchGen` | タイムライン上のラベル付きマーカーから演出情報を全自動走査し、量産出力用の CLI 指示書（CSV/BAT）および **静止画（PNG）群を自動生成**。※0フレーム同期前提。途中差し込み時のズレ警告ダイアログ搭載。 |

第9章：templates.json 主要パラメータ対照リファレンス（RigWeaver専用）

RigWeaver（Moho スクリプト）は、PSD の読み込みおよびボーン構築の際、`templates.json` の以下の項目を参照してリギングの挙動を制御します。

パラメータ名	工学的役割	設定例
<code>bone_folder_main</code>	モデル識別子: Moho プロジェクト内のグループ名と照合し、このプロファイルを適用するかを決定する。	<code>"RL_Bone_HumanV2"</code>
<code>virtual_bones</code>	トポロジー錬成: 実在しないボーンを生成し、 <code>redirect_children</code> によって既存ボーンの親子関係を強制的に組み替える。	<code>{"name": "Torso", "parent": "Hip", ...}</code>
<code>flexi_binds</code>	柔構造定義: 指定したレイヤーを、リストアップされたボーン群の共同影響下（Flexi-Bind）に置く。	<code>{"layers": ["Hip"], "bones": ["Hip"]}</code>
<code>flexi_force</code>	重み勾配設定: フレキシ・バインド対象ボーンの「強度（影響力）」をミリ単位で規定。	<code>0.1</code>
<code>bone_rot_offsets</code>	座標翻訳補正: 3D の回転軸と 2D 原画のズレを、ベクトル加算によって強制較正する。	<code>{"LHand": 90, "RHand": -90}</code>
<code>face_bone_keywords</code>	除外キーワード: 指定文字列を含むレイヤーを物理変形から除外し、自動的に <code>Head</code> ボーンへバインドする。	<code>["hair", "face", "eye", ...]</code>

パラメータ名	工学的役割	設定例
<code>rig_perspective_base_z</code>	パース逆算基準: Moho 上のレイヤー倍率 (Magnify) から Camera Z を算出するための基準定数。TDPT 等のパースが強いデータには 4.0 前後を推奨。	4.0

第10章：Pinボーン的设计思想と手動IK（逆運動学）調整のテクニック

10.1 全関節へのPinボーン配置の工学的意義

RigWeaver v1.0.4 以降、リグの各関節および中間パーツに対して「長さが0のPinボーン（ポイントボーン）」が標準で自動生成される設計が採用されています。このアプローチには、以下の2つの極めて強力なメリットが存在します。

- 3Dモーション焼き付けにおける無歪み（リジッド）追従:** 通常のくさび形ボーンとは異なり、Pinボーンを親として「直接のレイヤーバインド（Layer Binding）」を行うことで、画像はどれだけ大きく動いても均等に回転・移動するのみであり、伸縮や回転による「2Dイラストの不自然なゴム状変形」が完全に防止されます。これは `_3D.csv` などのモーションデータを直接流し込んで動作させる際に、最も美しく滑らかな結果を保証します。
- アタッチメント（アクセサリ）配置の極めて高い利便性:** 各部位（胸、肩、腰など）に独立したPinボーンが最初から配置されているため、「胸にワッペンを貼る」「腕に時計を着ける」といった後付けパーツの追加が、その部位のPinボーンへ直接レイヤーバインドするだけで、幾何学的な位置ズレや変形を起こさずに全自動で追従します。

10.2 プロユーザー向け：手動IK（逆運動学）追従の切り替えテクニック

MohoのIK（逆運動学）計算エンジンは、ボーンの「長さ（ベクトル）」を基準に関節の曲がりを算出します。そのため、中間に「長さが0のPinボーン」が介在すると、IKチェーンの連続性が物理的に切断され、末端を動かしたときに親ボーンが自動で曲がらなくなります（選択したボーン単体しか動かないFK挙動になります）。

手動で手足をIKで動かしてポーズを調整したいプロユーザーは、以下の簡単な手順でいつでもIK追従を有効化できます。

- 不要な中間Pinボーンの削除:** Mohoのキャンバス上で、手足の関節の途中（例：肘や膝など）にある不要な中間Pinボーンを選択し、`Delete` キーで削除します。
- 自動的なIKチェーンの再接続:** 中間の点（Pinボーン）が消えることで、前後の長さを持つくさび形ボーンの親子関係（IKチェーン）が直接連結されます。これにより、末端のコントローラー（手や足のボーン）を引っ張るだけで、腕全体や脚全体が美しく連動（IK追従）して曲がるようになります。

このハイブリッドな運用法（デフォルトの全部Pinボーンによる安定性 → 必要に応じた手動での中間ボーン間引き）こそが、Weaverプロトコルの柔軟性を最大化するプロフェッショナルなテクニックです。

第11章：周辺ツール連携：CSVビューアーのWindowsエクスプローラー連携と「送る(SendTo)」運用工学

RigWeaverの周辺ユーティリティである `csv_viewer`（3Dモーションデータの3Dプレビュー確認用ツール）をWindows環境で運用する場合、OSが保持するデフォルトのテキスト系拡張子（.csv）の強力な関連付け競合をスマートに解決し、2Dアニメーション制作のデスクトップワークフローと完全共存させるための知的な運用手段が提供されています。

11.1 Windowsエクスプローラーにおける関連付け競合の技術的背景

Windows OSにおいて、`.csv` 拡張子は VSCode や Microsoft Excel などの主要ツールにシステムレベルで極めて強力に関連付けられています。ユーザーが「常にこのアプリで開く」等のエクスプローラー標準手段によって、デフォルトプログラムを `csv_viewer.exe` に上書き変更しようとしても、Windowsのアップデートや競合エディタのアップデートによって自動的に VSCode 等に引き戻され、関連付けが戻ってしまう問題が多発します。

これを回避するため、本ツールはコマンドライン引数（`sys.argv[1]`：ファイルの絶対パス）の自律的な検証および解析ロジックを搭載し、以下の「送る (SendTo)」機能を活用した共存フローを公式設計としています。

11.2 恒久的ソリューション：「送る (SendTo)」メニューへのビューアー登録

普段のCSVデータの編集や閲覧（テキスト編集）はVSCodeで行い、**「3Dアニメーションとして形状や軌道をプレビューしたい瞬間だけ、右クリックから一瞬でビューアーへ送る」** という理想的な共存運用をわずか3ステップで構築します。

1. **「送る」専用フォルダの展開**: キーボードの `Win + R` を押し、ファイル名を指定して実行ダイアログに `shell:sendto` と入力して Enter キーを押します。Windowsユーザー個別の「送る (SendTo)」フォルダが開きます。
2. **ショートカットの配置**: ビルドされた `csv_viewer.exe`（または `csv_viewer.py` への起動ショートカット）をコピーし、開いた「送る」フォルダ内の何もない場所で右クリックし、「ショートカットの貼り付け」を実行します。
3. **識別名の決定**: 作成されたショートカットの名前を `CSV 3D Viewer` などのわかりやすい名前に変更します。

11.3 運用上のメリットと実作業プロセス

- **完全な共存**: ダブルクリックによるテキストエディタ（VSCode等）の迅速な起動を一切損なうことなく、右クリックから瞬時に `csv_viewer` を使って 3D 軌跡を表示・プレビュー可能です。
- **抜群の安定性**: システム全体のレジストリを一切汚さないため、PC管理者権限も不要で、OSアップデート等の競合に破壊されるリスクも物理的にゼロという高い堅牢性を誇ります。

第12章：結論：制作の「あと一步」を支えるための補助装置としての Weaver

RigWeaverは、3Dの数学的な正当性を2Dの情緒的表現へと効率的に繋ぐための補助装置である。制作者をリギングの迷宮から解放し、キャラクターに「命」を宿す清書と演出という、本来の創作工程にリソースを集中させること。このガイドに基づいた論理的な運用こそが、Weaverシリーズが目指す「制作の挫折をなくし、理想 of 表現へ辿り着くための確かな足場」を現実のものとする。