

# RigWeaver: Professional Technical Specifications

---

~ Manual for Rigging Engine Logic, Spatial Topology, and Production Workflow ~

## Introduction: A Bridge to Resolve "Friction" in 2D Animation Production

**[!IMPORTANT] Constraints on Supported Assets** This system (RigWeaver) officially supports only **PSD files with relatively simple layer structures created using LayerWeaver**.

It **does not support highly structured templates such as G3 characters** from other companies' products (Cartoon Animator 5, etc.). Those products have their own excellent design philosophies, and RigWeaver does not aim to substitute or be compatible with them. Since it is a tool to assist specific workflows of the Weaver series, operation when loading data of different designs is not guaranteed.

RigWeaver is a "workflow bridge" in the Moho Pro environment for logically connecting the deliverables of the Weaver series (Layer / Motion) and smoothly passing through the tedious process of rigging.

The system aims to coexist 2D animation-specific "lies of expression" (perspective exaggeration, intentional shape distortion) without collapse while utilizing 3D mathematical consistency as a guide. While traditional 2D rigging relied on massive manual trial and error, RigWeaver performs "calculation on behalf" of the creator to adapt 3D spatial movement received from MotionWeaver to the drawing characteristics of the 2D canvas. It is not a magic wand for automatically creating characters, but a precision "auxiliary engine" for lowering technical hurdles so that creators can concentrate on the exploration of expression.

---

## Chapter 1: Architecture and Design Philosophy

### 1.1 Harmony of 3D Physical Coordinates and 2D Expression

The core of RigWeaver lies in its multi-stage mapping structure, where instead of directly applying bone rotation data in 3D space to 2D images, it projects a 3D guide called "Viewer bones" into Moho's 3D space and translates them into 2D "controller bones."

This structure allows for maintaining accurate motion trajectories of 3D data as guides while freely overwriting and adding "aesthetic beauty as a drawing"—such as "arm shortening at specific angles" or "intentional joint position shifting"—impossible with 3D physical constraints through controllers. The system positions 3D data not as an absolute constraint but as a "reliable geometric standard" for expanding the freedom of 2D expression.

### 1.2 Non-linear, State-based Pipeline

RigWeaver discards the linear rigging procedure of "moving after design and construction are finished" and adopts a "state-based" design that allows intervention from any stage according to production progress and asset completion.

Specifically, simply executing **Step 1 (Setup & Viewer)** immediately after starting a project allows for "advance direction (action storyboard/previs)" using only the skeleton even if rigging settings are incomplete. Furthermore, if rough image assets (PSD) exist at this stage, "directional verification by character (pseudo-

previs)" with more visual information than standalone bones can be performed in advance by pseudo-pasting them to layers that move identically to the skeleton. However, movements generated by this method are strictly for "confirmation" to check timing and sense of distance, and should be operated distinctly from the production rig completed through formal rigging (Step 2–4).

On the other hand, if a character has an extremely simple structure like a "stick figure," or has an identical design where skeletal movement corresponds one-to-one with image deformation, it is possible to use these initial stage settings as the main animation. Since each process takes the form of non-destructively updating and overwriting "states" within Moho, refinement to formal rigging after such initial verification—or simplified production—can proceed completely in parallel while maintaining the timing of direction.

### 1.3 Synchronous Manipulation (Inheritance of Cleanup)

In RigWeaver, "inheritance of cleanup" is a core technology for maximizing animation quality. In rig hierarchies grouped by **SmartSwitch** and given proper binding settings, internal image layers are designed to completely inherit the transformation matrix of parent bones.

In this environment, whether performing "cleanup (inking)" by tracing raster images (PSD imported images) using Moho's vector function or deforming the original raster image itself with bones, drawing elements immediately start synchronizing with the parent's "3D-to-2D translation matrix" without recalculation of coordinate systems. Due to this property of "movement residing the moment you draw (or place)," animators are freed from complex binding resetting and weight adjustment, and can seamlessly inherit 3D dynamism into 2D expression through pure "inking" or "deformation direction."

## Chapter 2: Core Data Physical Specifications: templates.json

### 2.1 templates.json: Constant Definition as a Rigging Kernel

In this protocol, **templates.json** is positioned not just as a user configuration file but as a master kernel defining the "DNA (laws of physics)" of the rig. RigWeaver scans this file at execution and deploys optimal constant groups into memory using the model detection algorithm described later. Definition errors here directly lead to collapse of coordinate transformation and hierarchical structures.

### 2.2 Autonomous Model Detection Logic (**bone\_folder\_main**)

RigWeaver scans bone group names in the Moho project and searches for groups that exactly match the string described in the **bone\_folder\_main** key (e.g., **RL\_Bone\_HumanV2**).

- **Mechanism:** The moment this match is confirmed, the system identifies the target character structure (biped, quadruped, etc.) and applies all related physical constants (binding rules, virtual bone definitions, etc.).
- **Engineering Value:** This eliminates the need for users to manually switch model profiles, allowing the script to autonomously select the correct calculation engine even if multiple heterogeneous rigs coexist in a single scene.

### 2.3 Complementation and Construction of Spatial Topology (**virtual\_bones**)

**virtual\_bones** is a feature designed to compensate for structural constraints on the PSD (drawing) side, enabling a single illustration to deform smoothly and plially in response to 3D movement. Markers (joint

positions) such as **Torso** and **Hip** are placed in the PSD, and their locations are clearly defined. However, as the illustration layers often depict the "torso" as a single connected artwork (single mesh), it is physically impossible to segregate the assets into separate **Torso** and **Hip** layers. To apply the subtle bending details of the 3D (BVH) data to this "single-image torso" without collapse, it is necessary to deform it smoothly under the influence of multiple bones using Flexi-Bind (pliable binding). **virtual\_bones** by no means "creates imaginary bones out of thin air." Guided by the joint positions defined in the PSD, it plays the role of appropriately connecting and complementarily building the bone hierarchy (operational/structural joints) necessary to establish this Flexi-Bind deformation control in Moho.

- **redirect\_children Logic:** This attribute forcibly transfers child bone groups originally held by a specific parent bone to the subordinate of the complemented structural bone (e.g., **Torso**).
- **Mechanism:** When the **EnhanceRigHierarchy** function in the Lua script is executed, it dynamically rewrites the pointers of the hierarchy tree based on this rule, reorganizing the 3D anatomical structure into a topology suitable for the operability of animating a single 2D image.

## 2.4 Hybrid Bind Selection and Strength Definition (**flexi\_binds** / **flexi\_force**)

RigWeaver features a hybrid bind engine that automatically distinguishes whether each layer is "flexible structure (Flexi)" or "rigid structure (Layer)."

- **Selection Condition:** Layers included in the **flexi\_binds** list are marked as flexible variants influenced by multiple bones simultaneously.
- **Strength Gradient (**flexi\_force**):** Automatically applies "bone strength (Strength)" defined by **flexi\_force** (or **hip\_force**, etc.) to the bones targeted for flexi-bind. This automatically calculates physical influence so that the "core" hip part is strong, and terminal parts such as arms and legs bend with appropriate softness.

## 2.5 Anatomical Alias and Offset Correction (**bone\_aliases** / **bone\_rot\_offsets**)

A mathematical correction function to bridge the gap between the ambiguity of 3D space and the "correct answer" during 2D drawing.

- **bone\_aliases (Rescue Logic):** Even if the drawer names a part **UpperArm** or **Shoulder** in the PSD, it is recognized as the same anatomical part by the alias list, guaranteeing linkage with **\_rig.csv**.
- **bone\_rot\_offsets (Coordinate Transformation Correction):** If there is a discrepancy between the initial value of "how the arm is raised" in BVH and the "default arm angle" on the Moho canvas, the angle specified here (e.g., **LHand: 90**) is vector-added at execution. This "harmonizes" 3D rotation into an optimal pose for 2D character illustrations.

## 2.6 Resource Discovery and Resolution

RigWeaver autonomously scans directories in the following order of priority when loading core resources such as **templates.json** and **rigging\_map.csv** to ensure project consistency:

1. **Project Directory (PSD Folder):** The hierarchy where the currently active asset (PSD) is saved. Custom definitions specific to the project take top priority.
2. **Motion Directory (CSV Folder):** The hierarchy where the motion data used for import is saved.
3. **System Resource Directory:** **moho:UserAppDir() / Scripts / ScriptResources / rigweaver.**  
The final fallback destination where standard settings for the Weaver series are stored.

This priority allows users to safely switch optimal rigging settings for each project without polluting the script body (scripts/tool folder).

---

## Chapter 3: Internal Logic: Geometric Mapping and Coordinate Transformation

### 3.1 Statistical Inference Engine for Hierarchical Structure

RigWeaver features an inference engine for reconstructing robust bone hierarchies even from incomplete datasets (e.g., PSDs missing some joint markers).

- **Mechanism:** It multi-scans layer names in the PSD (attribution by @, nesting by >) and definitions in `rigging_map.csv` to deterministically derive parent-child relationships between bones.
- **Loop Guard:** Depth-first search (DFS) is used in the inference process, and a limiter is implemented to forcibly interrupt processing if a circular reference exceeding 50 levels is detected. This prevents Moho from crashing due to complex PSD structures.

### 3.2 Automatic Pivot Search and Fallback Rescue Logic

To identify pivots (joint points) that serve as rotation centers for bones, the system executes "hierarchical marker search."

- **Search Order:** First, it searches for empty layers or folders matching the specified joint name (e.g., `LShoulder`). If none exist, it refers to the center of gravity of the parent element or the base point of an alternative bone (e.g., `LArm`) defined in `bone_aliases` within `templates.json`.
- **Rescue Logic:** If a primary joint (e.g., `Torso`) does not physically exist, the system activates "Pivot Fallback," which automatically assigns `Chest` or `Spine` as root coordinates to avoid a complete stop of rigging.

### 3.3 Matrix Baking and Coordinate System Reset

`ExecuteFlattenRig` executed in **Step 5 (Finalize)** is an advanced matrix calculation process that destroys intermediate hierarchies for rigging work and flattens (normalizes) coordinate systems for production.

- **Mechanism of Baking:** Captures the "apparent transform (World Matrix)" of each layer and repositions it by calculating backward to maintain the same position, angle, and scale in the root coordinate system after releasing the parent hierarchy.
- **Result:** This "bakes" all movements that were dependent on complex parent-child relationships based on world coordinates (or root bones), guaranteeing extremely high levels of data compatibility and playback stability after export.

### 3.4 2D Translation Engine: Spatial Transfer Calculation from 3D to 2D Angles

In **Step 4A**, the "translation engine" that transfers quaternions/Euler angles in 3D space to 2D rotation angles operates.

- **Transfer Calculation:** Not a mere copy of angles; it uses the initial posture of the original illustration synchronized by Step 2B (Align Base Pose) as the reference point (0 degrees) and vector-synthesizes

`bone_rot_offsets` against the rotation deviation of the 3D bone from there.

- **Justification of 2.5D Expression:** Mathematically justifies "spatial lies" that re-interpret rotations around the Z-axis in 3D space as projection angles of the X or Y axes in 2D, achieving smooth movement with minimal illustration collapse.

### Automatic Control of 3D Baking via the UI "2D Mode" Checkbox

The **"2D Mode" checkbox (`is2DMode`)** provided on RigWeaver's control panel is a central UI switch that deterministically controls the baking behavior of 3D rotation components during import and retargeting. In the latest version, since spatial 2D projection and angle unwrap (jump prevention) algorithms are highly robust, **running with the "2D Mode" checkbox disabled (unchecked = 3D Mode) is the strongly recommended standard workflow**. This automatically toggles the following behaviors without requiring the user to manually edit the configuration file (`templates.json`):

- **"2D Mode" Disabled (Unchecked / 3D Mode) [Standard Recommended]:** All three X/Y/Z rotations (Pitch/Yaw/Roll) are formally baked onto the layers. The physically accurate movement of the 3D data is injected directly into the rig, allowing for organic body twisting and visual depth (foreshortening/perspective scale) to be beautifully expressed on the 2D canvas. Even if the 3D motion involves extreme twists, the matrix projection and double unwrap features smoothly round the angles, preventing any abrupt flips or rig instability.
- **"2D Mode" Enabled (Checked):** To preserve 100% of the flat, original silhouette of 2D illustrations, the baking of 3D rotations on the X and Y axes is completely disabled (fixed to 0 degrees), ensuring **zero 3D-like perspective shearing or distortion on the artwork**. In this mode, rather than directly applying raw 3D Z-rotations (e.g., extreme values like 112.19 degrees at frame 350) which would distort parent layers, the parent Hip layer automatically triggers a **2D projection fallback**. This calculates a natural 2D orientation (e.g., approximately -4.10 degrees at the same frame) based on the 2D projected vector from the Hip to the Torso (spine), physically ensuring that the bone follows the spine naturally without breaking the illustration. This mode is useful only for flat assets where you do not want the original art silhouettes to deform or twist.

[Supplementary Note] Inclusion of Viewport Matrix Since the `GetStableProjectedZAngle` function performs calculations via `layer:GetFullTransform`, the rotation state of the workspace (view matrix) is directly reflected in the calculation results. While this is intended for "stabilization" to prevent unintended distortion, it secondarily enables the directorial technique of "Perspective Baking" described in Section 5.4.

### 3.5 The Price of Designing PSD as "Ground Truth"

The most prominent feature in RigWeaver's design philosophy is treating **"the placement of layers (Islands) and markers in the PSD as the absolute correct answer (Ground Truth)"** in rig construction. While this philosophy guarantees freedom of drawing, it creates the following special engineering challenges:

- **Bone Starvation:** In 2D illustrations, parts like the "Torso" are usually drawn as one giant image. Following PSD marker placement (e.g., only hip and chest) too strictly results in only one bone being assigned to that giant mesh, leading to a physical lack of rotation axes necessary for "organic bending" such as arching or twisting the body.
- **Surgical Solution via Virtual Bones:** To compensate for this "anatomical lack," **"surgical addition of bones"** using `virtual_bones` in `templates.json` is necessary. This is not only for mirroring 3D (BVH)

logic but is a mechanism born from engineering necessity to complement and expand the "lack of mesh deformation resources" of the 2D body in PSD on the system side.

- **Topology Remapping:** The added virtual bones function as "intermediate joints" in the matrix calculation (2D translation) mentioned in section 3.4 and the baking process in section 3.3, providing rotation vectors for multi-stage bending of a single mesh. This process achieves production-level smooth animation density without complicating the drawing.
- **Operational Optimization (High-Precision Segmentation):** For more precise part separation, we highly recommend using the **"Digital"** -> **"Pixel Art"** category (e.g., Pixel Art Fill) in Krita for drawing or editing markers. These brushes provide binary opacity (100% Opaque) without anti-aliasing, maximizing the benefit of the internal **alpha\_threshold logic**. This enables the system to reliably recognize even a **1-pixel gap** as a segmentation boundary.

### 3.6 Engineering Requirement for Workspace Normalization and Initial Scaling

In RigWeaver's calculation system, **"scale normalization to the Moho workspace (a projection area of approximately 1.6 units in height)"** is an absolute prerequisite for maintaining the consistency of coordinate systems.

- **Spatial Projection Specifications:** The **ExecuteImport3D** function scans the absolute height of the BVH source and projects the Viewer by dynamically calculating the constant **POS\_SCALE = 1.6 / height** so that it fits within Moho's standard field of view. This is a spatial normalization process for optimally mapping motion energy of 3D space to the effective drawing range of the 2D canvas.
- **Normalization of Perspective Calculation:** Assuming the above normalization (height of 1.6), Camera Z is calculated using the formula **(rig\_perspective\_base\_z / Magnify) / POS\_SCALE**. This normalization process ensures consistent perspective control via a single constant, **rig\_perspective\_base\_z**, regardless of the character's physical size.
- **Alignment of Reference Coordinate Systems:** PSD assets immediately after being imported into Moho are placed in arbitrary (non-normalized) sizes depending on original resolution or DPI. Manually fitting the illustration to the size of the workspace before starting rigging (Step 0) is an important ceremony for synchronizing the system's "calculation reference coordinate system" with the PSD's "entity coordinate system."
- **Risk of Inconsistency (Collapse of Hierarchy):** If the rigging process is forced without performing this initial scaling (normalization), cumulative numerical errors will occur in marker detection positions and relative vector calculations during hierarchy construction. This results in rig collapse due to logical inconsistencies, such as "Layer Shifting" to abnormal positions or "fatal deviation of bone generation positions."
- **Setup Pose Protection:** A physical guard is implemented to prevent unintended writing to frame 0 (setup frame), which serves as the system's calculation reference. Even if motion injection (Step 3) is executed while the timeline is at frame 0, it is automatically corrected internally to "application from frame 1." This engineeredly avoids the risk of the illustration's "Setup Pose" coordinates set by the artist being overwritten and destroyed by 3D mathematical motion data, guaranteeing permanent stability of the rig.

### 3.7 1B/2B Synchronization: The Mathematical Bridge Connecting 3D Projection and 2D Rigs

In the RigWeaver workflow, the continuity of **Step 1B "Bind Character"** and **Step 2B "Align Base Pose"** is a critical phase that determines the physical stability of the rig.



- **The Role of 1B (Logical Adhesive):** Step 1B is the process of mathematically linking 2D image assets to the 3D spatial guides (Viewer bones). At this stage, images are promoted from simple "drawings on a canvas" to "rig components" subordinate to the 3D spatial transform matrix.
- **The Role of 2B (Geometric Alignment):** Step 2B performs final alignment adjustments on the images linked in 1B to match the original illustration pose (Base Pose).
- **Risk of Skipping 1B (Coordinate System Divergence):** If 1B is skipped and 2B is executed, images remain in a "floating state" not belonging to the 3D coordinate system during alignment calculations. Consequently, the 3D motion vectors and 2D drawing coordinates do not align mathematically, resulting in a fatal inconsistency where the character scatters or the rig collapses.
- **Reason for the T-Pose Exception:** If a character is in a perfect T-Pose (arms horizontal, legs vertical), the initial coordinate systems of the 3D Viewer and the 2D rig happen to match mathematically (Zero-Offset). Therefore, skipping 1B may accidentally maintain consistency. However, this is an "exceptional situation" where the coordinate discrepancy just happened to be zero, and it will not work if the pose differs even slightly. For stable rigging, it is an engineering requirement to execute 1B even for T-Poses to establish logical binding.

### 3.8 Engineering Background of the Absolute "Frame 0" Rule

**Frame 0 (Setup Mode)** in Moho Pro is not just the starting point of the timeline, but a special area where the "Blueprint (REST Pose)" of all elements constituting the rig is stored.

- **The Immutable Reference Point:** All construction processes in RigWeaver (bone generation, pivot determination, binding establishment) refer to the state of Frame 0 as the "absolute correct answer."
- **Impact from Frame 1 Onwards:** Data recorded from Frame 1 onwards is merely the "Delta (change)" from Frame 0. If Frame 0 is contaminated or altered by 3D data injection, all subsequent motion data becomes deviations from a "corrupted reference point," causing the rig to exhibit permanently unpredictable behavior.
- **System Protection:** To avoid this fatal risk, RigWeaver enforces an automatic shift to Frame 1 for motion application from Step 3 onwards. Continuously protecting Frame 0 as a "Sanctuary" is the physical foundation for reproducing complex 3D performances with a 2D rig.

**[!CAUTION] Safety of Mid-Timeline Motion Placement & Baking, and Synchronization Constraints in Batch Generation (BatchGen)** In the latest version, a **"Multi-Action Bake"** feature has been implemented in Step 4B. Bake & Bind. The system now automatically scans all CSV paths and start frames registered on the timeline markers, baking them all in a single batch. Consequently, the previous engineering risk of "writing data mid-timeline causing rig collapse" has been completely resolved. Animators can now safely place and bake different action CSV files at any arbitrary position along the timeline.

However, for the batch generation (**BatchGen**) process, the constraint of **"Frame Index Offset Synchronization"** still remains. Because **BatchGen** references timeline marker frames to generate slicing commands for the raw source BVH file, it mathematically assumes that the Moho timeline frame indices and the raw BVH frame indices are **perfectly synchronized with a 1:1 ratio starting at Frame 0/1 (zero-offset)**. If you place markers on a segment retargeted starting at Frame 48 without synchronization, the system will instruct the engine to slice the wrong frame range from the raw BVH.

**[Safety Circuit: Timeline Sync Warning Integration]** : To prevent this data mismatch, the system triggers a **"Timeline Sync Warning"** dialogue immediately after selecting the BVH file during

**BatchGen** execution, prompting the user to confirm that the timeline starts synchronized with the source BVH.

In summary, **while the "Action Slicing (3B. BatchGen)" task must be performed on a synchronized timeline (or starting at Frame 0/1), the subsequent "Bake & Bind (4B. Bake)" process can be executed with actions placed at any arbitrary frame mid-timeline.** Understanding this workflow distinction is key to successfully staging and baking multiple actions.

### 3.9 Autonomous Avoidance and Stabilization Logic for Bone Length Inconsistency during Retargeting (Torso Collapse Prevention and Hybrid Scaling)

Even when there is a discrepancy between the skeletal proportions of the 3D motion data (CSV) and the 2D rig constructed from the 2D character illustration (PSD) (specifically torso length, etc.), RigWeaver is equipped with an "autonomous avoidance and stabilization logic" that successfully completes the motion injection without collapsing the rig.

- **Locking Mechanism of Intermediate Joint Translation during Retargeting:** During the execution of **Step 3. Retarget Motion**, the system injects translation values only into the reference points: "Hip," "Root," and any "Orphan" bones that do not have a parent bone. For all other intermediate bones such as arms, legs, and spine (torso), the system does not apply relative position changes from their parent bones. Instead, it locks their positions to their connected locations and only transfers and reflects "rotation around the Z-axis (Rotation)." Consequently, no matter how much the length of the torso or limbs in the original 3D data differs from the 2D character, it logically prevents the 2D character's body (proportions) from being forcibly stretched apart or joint positions from shifting and collapsing.
- **Dynamic Foreshortening and Scaling Based on the Initial Pose (Rest Pose):** During **Step 4B. Bake Motion 2D**, the system applies dynamic scaling (stretching) to limb segments during animation. This scaling factor is not directly calculated from raw 3D position coordinates. Instead, the system first measures and caches the "initial design distance (rest distance)" between each bone on the 2D rig at the setup frame (Frame 0). Then, for each frame, it dynamically calculates the "stretching ratio (scale)" by dividing the corresponding bone distance on the 3D guide by this initial design distance. Applying this ratio to the 2D rig's bone scale (**fAnimScale**) ensures that the initial proportions of the 2D character designed by the creator at Frame 0 are strictly maintained as the "standard (uniform scale = 1.0)," regardless of the original CSV bone length. This preserves the overall model structure while naturally applying only the dynamic foreshortening and perspective stretching inherent in the 3D performance to the 2D rig.

---

## Chapter 4: Operational Engineering: Multi-stage Verification Workflow

RigWeaver adopts a "multi-stage protocol" that allows for switching the depth of verification according to asset completion. This makes it possible to solidify direction without waiting for rigging completion, minimizing backtracking.

### 4.1 Phase 1: Skeleton-Only Previs

This phase is executed in the earliest stages of production or before the character design is solidified.

- **Mechanism:** Projects **\_3d.csv** movement directly onto Viewer bones via **1A. Load 3D Data**.



- **Purpose:** Advance confirmation of performance timing, camera work, and spatial placement.
- **Engineering Value:** Even without any character assets, it can be used as a "moving video storyboard" based on actual motion data, allowing for the elimination of fatal directional errors in upstream processes.

## 4.2 Phase 2: Layer-Driven Mockup

This phase is for verifying "visual thickness" once a rough character design is completed or using temporary assets.

- **Mechanism:** Pastes temporary layers (proxy images) that move identically to the skeleton and verifies the validity of overlap order based on the 3D Z-axis (depth).
- **Purpose:** Confirmation of silhouettes, character-to-character contact detection, and final confirmation of depth expression.
- **Engineering Value:** Detailed screen composition considerations using characters become possible without formal rigging (1A–4B), allowing for early discovery of "insufficient drawing range" or "alignment inconsistencies."

## 4.3 Phase 3: Full Production and Cleanup Sync

The completion phase where all assets are ready and final animation is exported.

- **Mechanism:** Completes the formal process from 1A to 5 and applies hybrid binding to final assets packed by SmartSwitch.
- **Synchronous Inking:** The primary benefit of this phase is vector cleanup performed within image groups linked to bones. Drawn vector lines inherit parent 3D transformation matrices with "zero delay and zero recalculation," instantly transferring the dynamism of 3D motion into drawing.
- **Engineering Value:** Technical rigging settings are assimilated into a "transparent infrastructure for expression," allowing creators to concentrate all resources on emotional direction through final "inking."

---

# Chapter 5: Directorial Engineering: Video Storyboards and Angle Analysis

In the Weaver protocol, the role of Moho is not just an animation output machine but a "Digital Director's Console" for visualizing direction and guiding drawing.

## 5.1 Generation of "Moving Drafts" via 3D Poses and Previs

Through RigWeaver's 1A. Load 3D Data and 3. Retarget Motion, performance data in 3D space is materialized as "moving skeletons" on a 2D canvas.

- **Practical Mechanism:** This skeleton maintains anatomically correct 3D motion trajectories. By overlapping rough PSDs mentioned in section 4.2 here, a "moving draft (video storyboard)" is completed for precisely verifying final character volume and motion timing.
- **Feedback for Drawing:** By using this moving skeleton as a "foundation," animators can concentrate on confident drawing based on directorial intent without worrying about complex foreshortening or joint movement.

## 5.2 Physics of High/Low-Angles via Controllers: Controlling "Lies" of Spatial Projection

RigWeaver separately extracts "vertical tilt (high/low-angle)" and "horizontal angle" from absolute rotations in 3D space and recalculates them as differences (Delta) between layer rotation and controller bone angles.

- **Mathematics of Spatial Projection:** MotionWeaver-calculated 3D->2D projection information is deployed on Moho by RigWeaver as "pseudo-perspective deformation" of a single image.
- **Fine-tuning by Sensitivity:** By operating controller bones, it is possible to directly fine-tune the "angle of tilt" for maximum impact as a 2D drawing in units of 0.1 degrees while maintaining 3D "correctness."

### 5.3 The Artisan's Realm: Reorganizing "Tilt Drawings" based on Guides (Recurrence to LayerWeaver)

The most effective operation of the Weaver series lies in the recursive flow of returning to LayerWeaver (Krita) based on these moving guides and **"redrawing tilt/high-angle parts optimized for specific poses."**

- **Guide Extraction via Preview Function:** Executing Moho's preview function (**File > Preview**) allows you to check the rendering result of the current frame in a separate window. Images can be saved in various formats or copied to the clipboard using this window's menu.
- **Recursiveness of Workflow:** Place (or paste) the extracted image as a layer in Krita and perform cleanup of parts specialized for high or low angles using the skeleton as a standard. Completed parts are re-imported into Moho via LayerWeaver and synchronize with 3D movement through auto-binding.
- **Engineering Result:** This is not mere "automation" but a process for providing drawers with "reliable auxiliary lines" called 3D guides to push drawing limits with 3D logic. 3D consistency and "emotional perspective lies" only drawable by human hands merge at a high level through preview images.

### 5.4 Viewpoint-Dependent Perspective Baking

The projection algorithm of RigWeaver is not merely a transcription of 3D space, but has a unique characteristic of baking the "workspace (orbit) angle at the time of Retarget execution" as the reference pose of the 2D rig.

- **Mechanism and Directorial Effects:**
  - By rotating the Moho workspace "slightly diagonally" before executing Step 3. Retarget Motion, the system interprets that projected view as the front (reference) on the 2D canvas.
  - When the orbit is returned to 0, the perspective (intentional distortion) from that diagonal viewpoint remains in the rig. This allows the character to internalize a 2D-animation-like "lie" of "always facing a specific angle" while keeping the Moho camera fixed.
- **Engineering Operational Limits and Characteristics:**
  - **Applicability:** This feature works most stably for "slight diagonal" or "gentle high/low angles."
  - **Depth Constraint:** It does not support extreme high angles (e.g., top-down view). This is because vertical 3D vectors converge to points on the screen, causing the angular projection onto the 2D plane to break down geometrically.
  - **Recommended Use:** It is most aligned with the design philosophy of this system to use it for giving a slight "sense of depth" to a character's standing pose or for "subtle perspective exaggeration" to enhance the impact of an action.
- **Perspective Intensity Adjustment and Inverse Calculation Logic:**
  - During rigging, **rig\_perspective\_base\_z** is referenced as the standard when converting the "Magnify" (scaling) of a layer to Camera Z.

- **Engineering Characteristic:** The smaller this value (e.g., 4.0), the closer (stronger) the perspective becomes for the same layer magnification, resulting in a more dynamic sense of distance. When using sources with intense depth changes like TDPT, setting this value to 4.0 minimizes the discrepancy between the visual appearance in Moho and the actual 3D depth, allowing the intended perspective expression to be internalized in the rig.

## 5.5 Hybrid Scaling: Separation of Foreshortening and Perspective Scaling

Starting with RigWeaver v1.0.4, a "Hybrid Scaling" logic has been introduced to optimize scaling for different body parts.

- **Limb Segments:** Arms, legs, spine, etc. These intermediate bones only receive "Foreshortening" calculated from the 3D projection distance. This prevents images from becoming unnaturally thick (uniform scaling) when approaching the camera, while still reflecting changes in length.
- **Terminal Segments:** Hands, feet, head, and parents of end bones. These receive "Perspective Scaling" based on CSV data via dedicated pin bones (`_Scale`).
- **Engineering Result:** This separation automates dynamic 2D-style exaggeration—such as "making hands look larger without thickening the arms"—while maintaining the geometric integrity of the 3D data.

---

## Chapter 6: Production Engineering: Automated Mass Production Flow

The utilization of automation in RigWeaver lies in operating Moho not as simple animation software but as a **"Digital Director's Console"** governing the direction of the entire project. This is mass production engineering for improving production speed in personal environments.

### 6.1 Timeline Markers: "Cutting" Direction and Instruction Definition

Markers on the timeline function as "logical switches" for linking directorial information to `BatchWeaver` (CLI), and also serve as **critical timing nodes to convey multiple action CSV paths and start frames for Bake & Bind (4B)**.

- **Marker Validity Conditions:** RigWeaver recognizes markers assigned with labels (names) for analysis.
  - **Presence of Label (Name):** Markers without names are ignored to ensure deterministic file naming and action CSV resolution.
  - **Output Type Determination & Pose-to-Pose Interpolation:**
    - **Video (CSV/Batch):** Markers with a valid width (Duration > 0) set by dragging.
    - **Still Image (PNG) & 1-Frame CSV:** Any labeled marker is stacked as a candidate for PNG export (`pendingPNGs`) regardless of width. **Specifically, still markers with a duration of 0 are now automatically generated as both a PNG image and a "1-frame only 3D CSV file" (with an automatically appended suffix `_f[frame]_s`).** This is a major breakthrough for animators: it enables a **"Pose-to-Pose (Pose-to-Pose Interpolation)"** workflow where you can slice and feed only key poses or stances from 3D animations at specific frame positions, allowing Moho's robust vector and bone auto-interpolation engine to generate smooth, custom 2D transitions between them. This eliminates the computing load of baking massive, frame-heavy raw motions while granting full artistic control over intermediate pose adjustments.
- **Automated Still Output Mechanism (`we_makepng.lua`):**

- Immediately after **ExecuteBatchGen** completes, if valid labeled markers are detected, the external helper script **we\_makepng.lua** is dynamically invoked.
- This script autonomously scans Moho's **fTimelineMarkers** channel (Document and Selected Layer) and executes batch rendering (**FileRender**) of all specified frames to the project directory. The exported PNG files are saved with a **\_p** suffix added to their filenames (**{label}\_p\_f{frame}.png**) to clearly categorize them as video/still pose references.
- **Consistency of Data Retrieval (Fixed to Controller):**
  - Markers can be placed on the global timeline or individual layers.
  - In all cases, animation data is fetched consistently from **3D\_Angle\_Controller**, eliminating inconsistencies due to multi-layered management.
- **Segmentation of Direction:** Marker "label names" become the final output filenames, and "marker width (length)" is defined as the scene duration (start and end frames). Creators can complete the "cutting" of directorial plans simply by placing markers in necessary ranges on the timeline.
- **Fully Automated Scan of Instruction Information:** When the **ExecuteBatchGen** function is executed, the system high-speed scans the entire timeline, extracts "start frame," "end frame," and "layer rotation (angle)" information for all marker sections, and converts them into a normalized CSV instruction sheet.

## 6.2 100-cut Automated Ejection via CLI: Physics of RUN.bat Generation

What is ejected from Moho is not just data. It is an "ignition switch" for moving a giant factory.

- **Automated Generation of RUN.bat (Execution Batch File):** **RUN.bat** integrating directorial information and a CSV instruction sheet are generated in the current folder. This batch file contains a group of batch command instructions for MotionWeaver.
- **Engineering Result of Batch Instructions:** Simply double-clicking the generated **RUN.bat** launches MotionWeaver in high speed in the background. Dozens of specified angles and scenes covering hundreds of frames are exported one after another without creator intervention.
- **Paradigm Shift in Mass Production:** This flow of solidifying direction in Moho and completely delegating export to the system makes "reconciling volume and quality," impossible in traditional 2D animation production, a reality through overnight automated processing.

---

## Chapter 7: Weaver\_SmartSwitch: Physics of Binding Inheritance

**Weaver\_SmartSwitch** is not just a layer packing tool but an intelligent conversion engine for "**promoting binding information to upper hierarchies and permanently propagating coordinate matrices**" built with RigWeaver.

### 7.1 "Promotion" of Binding DNA and Strengthening of Hierarchical Topology

Normally, putting individually bound layers into a simple group in Moho requires re-setting binding information or results in structural instability. SmartSwitch engineeringly solves this problem.

- **Automated Promotion of Binding Information:** At script execution, it scans binding information (DNA of which bone it is linked to) held by each **@tag** layer and automatically re-applies (promotes) that information to the generated parent group (Group/Switch).
- **Hierarchical Governance:** By firmly linking the parent hierarchy to bones, its interior is defined as a single "independent coordinate space." This allows internal layers to 100% transparently inherit the 3D/2D conversion matrix received by the parent without being conscious of individual binding settings.

## 7.2 Pleomorphic Performance Control via Switch Layers

Switch layers generated by SmartSwitch play a role beyond simple "display switching."

- **Pleomorphic Synchronization Logic:** Different angles such as "Front, Side, Back" and each frame of complex lip-sync are all governed under the same "parent binding information." This mathematically guarantees synchronization with bone movement at anatomically perfect positions the moment any frame is switched.
- **Liberation of Direction:** Animators can control characters' multi-faceted performances simply by switching, without any concern for "rigging collapse."

## 7.3 The Ultimate in Cleanup Sync: Why it moves "the moment you redraw"

The greatest engineering result of this system is the "zero-delay motion transfer" to subsequent vector cleanup.

- **Benefits of Matrix Inheritance:** Create new vector layers within groups governed by SmartSwitch and perform cleanup (inking). This new layer inherits the "wiring path to bones" held by the parent group the moment it is created.
- **Fusion of Emotional Lies and Logic:** Vector lines drawn by tracing 3D guides (preview images, etc.) directly receive the dynamism of 3D motion through the parent hierarchy. This completely merges "confident 3D movement" and "detailed lines drawn by 2D artisans" without any re-rigging process.

## 7.4 Bone-Driven Instantiation

Starting from SmartSwitch v3.2, a feature for dynamic generation of new parts starting from active bones has been implemented, in addition to existing layer packing.

- **Autonomous Judgment Logic:** When the script is executed with zero selected layers or only the Bone layer selected, the system automatically transitions to "Bone Selection Mode."
- **Forced Normalization of Topology:** Generated new containers (Group/Switch) are forcibly placed directly under the Bone Layer that owns the skeleton, rather than `moho.layer`. This covers any deficiencies in layer selection by the user and maintains a hierarchical structure where bone control is reliably applied.
- **Instant Injection of Binding Information:** Simultaneously with container generation, the selected Bone ID is instantly applied through `SetLayerParentBone`. Since the empty vector layer created inside 100% inherits the coordinate matrix of this parent container, it is possible to instantly add post-attached parts such as accessories to existing rigged models without geometric errors.

---

# Chapter 8: Technical Definition Cross-Reference

UI Button Name	Internal Function	Physical Role / Engineering Value
1A. Load 3D Data	<code>ExecuteImport3D</code>	Import of 3D motion data and visualization of spatial coordinates via Viewer bones (3D guides).
Set Thickness	<code>ExecuteSetThickness</code>	Application of depth (thickness) in "injection" calculation. Spatial realization via Z-axis deviation of Viewer bones.

UI Button Name	Internal Function	Physical Role / Engineering Value
1B. Bind Character	ExecuteBindImages	Physical linkage of PSD assets to the generated Viewer bone coordinate system.
2A. Build Hierarchy	ExecuteBuild3D	Deterministic construction of a formal 3D skeleton hierarchy based on inference rules in <code>templates.json</code> .
2B. Align Base Pose	ExecuteAlignMarkers	Synchronizes original illustration pose with bone rest pose, calibrating geometric alignment errors.
3. Retarget Motion	ExecuteRetarget	"Transfers" motion energy of 3D guides to formal bones. Injection position (start frame) and clearing of existing data can be controlled. <b>Risk of rig collapse if executed at frames other than 0.</b>
4A. Build 2D Rig	ExecuteCreateBones2D	"Motion injection" of 3D spatial rotation components to 2D canvas, dynamically generating controllers for animation control.
4B. Bake & Bind	ExecuteBakeMotion2D / BindCleanup	"Baking" of performance to 2D space and physical synchronization of body and skeleton via hybrid binding. Limbs receive "Foreshortening," while terminals receive "Perspective Scale" via <code>_Scale</code> bones.
(Note) <code>_Scale Bone</code>	-	Pin bones for perspective deformation of terminal parts. Spare bones for intermediate parts (arms, etc.) can be used for binding small items like watches or jewelry to maintain their shape.

| **5. Flatten Rig** | `ExecuteFlattenRig` | Destroys intermediate hierarchies for rigging work and normalizes the coordinate system to world standard (Matrix Baking). | | **Smart Switch** | `SmartSwitch:Run` | Promotion and physical propagation of binding DNA, and unitization (switching) for production. | | **Batch Gen** | `ExecuteBatchGen` | Fully automated scan of directorial information from labeled timeline markers, generating CLI instruction sheets (CSV/BAT) and **automatically generating still image (PNG) groups. Requires Frame 0 synchronization. Features a built-in sync warning dialogue.** |

## Chapter 9: Major Parameter Cross-Reference for templates.json (RigWeaver Only)

RigWeaver (Moho script) refers to the following items in `templates.json` during PSD loading and bone construction to control rigging behavior.

Parameter Name	Engineering Role	Example Setting
<code>bone_folder_main</code>	<b>Model Identifier:</b> Matches group names in Moho projects to determine whether to apply this profile.	<code>"RL_Bone_HumanV2"</code>



Parameter Name	Engineering Role	Example Setting
<b>virtual_bones</b>	<b>Topology Forging:</b> Generates non-existent bones and forcibly reorganizes existing bone parent-child relationships via <code>redirect_children</code> .	<pre>{"name": "Torso", "parent": "Hip", ...}</pre>
<b>flexi_binds</b>	<b>Flexible Structure Definition:</b> Places specified layers under the joint influence (Flexi-Bind) of listed bone groups.	<pre>{"layers": ["Hip"], "bones": ["Hip"]}</pre>
<b>flexi_force</b>	<b>Weight Gradient Setting:</b> Stipulates "strength (influence)" of bones targeted for flexi-bind in millimeters.	<b>0.1</b>
<b>bone_rot_offsets</b>	<b>Coordinate Translation Correction:</b> Forcibly calibrates discrepancy between 3D rotation axes and 2D original drawings via vector addition.	<pre>{"LHand": 90, "RHand": -90}</pre>
<b>face_bone_keywords</b>	<b>Exclusion Keywords:</b> Excludes layers containing specified strings from physical deformation and automatically binds them to the <b>Head</b> bone.	<pre>["hair", "face", "eye", ...]</pre>
<b>rig_perspective_base_z</b>	<b>Perspective Inverse Calculation Standard:</b> Reference constant for calculating Camera Z from layer magnification (Magnify) on Moho. Recommended around 4.0 for data with strong perspective like TDPT.	<b>4.0</b>

## Chapter 10: PinBone Design Philosophy and Manual IK (Inverse Kinematics) Adjustment Techniques

### 10.1 Engineering Significance of Placing PinBones on All Joints

Since RigWeaver v1.0.4, the system automatically generates "zero-length PinBones (point bones)" for all joints and intermediate body parts by default. This design offers two highly powerful engineering advantages:

- 1. Zero-Deformation (Rigid) Following in 3D Motion Baking:** Unlike standard wedge-shaped bones, using PinBones as parents for direct "Layer Binding" ensures that the bound images only rotate and translate uniformly. This completely prevents any unnatural rubber-like stretching or distortion of 2D illustrations, guaranteeing the cleanest and smoothest results when baking `_3D.csv` motion data.
- 2. Extreme Convenience for Attachment (Accessory) Placement:** Because independent PinBones are placed at each body segment (Chest, Shoulder, Hip, etc.) from the start, adding post-attached parts—such as placing a badge on the chest or a watch on the wrist—is incredibly simple. Users can just directly Layer Bind the accessory to that specific segment's PinBone, and it will follow perfectly without any geometric drift or deformation.

### 10.2 For Advanced Users: Techniques for Enabling Manual IK Following

Moho's IK (Inverse Kinematics) engine calculates joint bending based on a bone's "length (vector)". Therefore, when a zero-length PinBone is inserted in the middle of a limb, the continuity of the IK chain is physically broken. As a result, moving the end effector will not bend the parent bones automatically (behaving as FK, where only the selected bone moves).

Advanced users who wish to manually adjust poses using IK can easily enable it through the following steps:

1. **Delete Intermediate PinBones:** On Moho's canvas, select and delete unnecessary intermediate PinBones (e.g., Elbow or Knee joints) using the **Delete** key.
2. **Automatic Reconnection of the IK Chain:** Once the intermediate PinBones are removed, the parent-child relationships of the remaining wedge-shaped bones (which have actual lengths) will directly connect. This immediately enables smooth, natural joint bending (IK following) simply by pulling the end controller (Hand or Foot bone).

This hybrid approach—relying on the default all-PinBone structure for rock-solid stability, while allowing manual deletion of intermediate joints for custom IK adjustment—is a professional technique that maximizes the flexibility of the Weaver workflow.

---

## Chapter 11: Tool Integration: CSV Viewer Explorer Integration and "SendTo" Workflow Engineering

When operating **csv\_viewer** (a utility for 3D motion data preview and trajectory verification) in a Windows environment, a smart solution is provided to cleanly bypass OS-level file association conflicts while perfectly coexisting with the daily 2D animation production workflow.

### 11.1 Technical Background of File Association Conflicts in Windows

In Windows OS, the **.csv** extension is very strongly associated with text editors like VSCode or spreadsheet software like Microsoft Excel at the system level. Even if a user attempts to override the default program to **csv\_viewer.exe** using standard Explorer options (e.g., "Always use this app"), Windows updates or updates to competing editors often forcibly pull the association back to VSCode, breaking the user's setup.

To avoid this headache, this viewer features autonomous command-line argument parsing (**sys.argv[1]** for file absolute path validation) and officially advocates the following "SendTo" integration workflow.

### 11.2 Permanent Solution: Registering the Viewer to the "SendTo" Menu

This approach enables a perfect dual-use workflow: edit and inspect CSV files in VSCode by default (double-click), and **instantly send the file to the 3D viewer via the right-click menu only when you need to preview its 3D animation trajectory**. Setup requires only 3 simple steps:

1. **Open the SendTo Directory:** Press **Win + R** on your keyboard, type **shell:sendto** in the Run dialog, and press Enter. This opens the current user's dedicated "SendTo" folder.
2. **Place a Shortcut:** Copy your built **csv\_viewer.exe** (or a shortcut launching your **csv\_viewer.py**), right-click inside the "SendTo" folder, and select "Paste shortcut".
3. **Name the Shortcut:** Rename the newly created shortcut to something recognizable, such as **CSV 3D Viewer**.

### 11.3 Engineering Benefits and Practical Workflow

- **Seamless Coexistence:** Keep VSCode as the fast, default text editor for **.csv** files when double-clicking, while enabling an instant 3D preview of the motion via **Right-Click → Send to → CSV 3D Viewer**.
  - **Rock-Solid Stability:** Since it does not modify the Windows registry, it does not require administrator privileges, remains unaffected by OS updates, and guarantees a 100% reliable workflow.
- 

## Chapter 12: Conclusion: Weaver as an Auxiliary Device Supporting "The Final Step" of Production

RigWeaver is an auxiliary device for efficiently connecting 3D mathematical correctness to 2D emotional expression. It aims to liberate creators from the labyrinth of rigging and allow them to concentrate resources on the original creative processes of cleanup and direction to put "life" into characters. Logical operation based on this guide will make "a reliable scaffold for eliminating production frustration and reaching ideal of expression," the goal of the Weaver series, a reality.

---

*RigWeaver: Professional Technical Specifications v10.2 - Fully Integrated Reference*