Animation:Master Spline/Patch Model Notes

Working field notes on .mdl construction, Named Groups, Folders, extraction, and procedural helpers.

Date: 2026-04-29

This package captures the practical discoveries from recent Animation:Master spline/patch model experiments and preserves them with runnable Python helper scripts.

Animation:Master Spline/Patch Model Notes

Date: 2026-04-29

Purpose

This document codifies the working knowledge from our recent Animation:Master spline and patch model experiments. It is a field notebook rather than a formal .mdl specification.

The package focuses on four practical goals: describing what we think we know, preserving safe procedural workflows, documenting the role of Named Groups and Folders, and providing Python scripts that capture core inspection and planning concepts.

The previous shared ChatGPT conversation URL was not directly readable in this environment, so this is reconstructed from the remembered findings available in this session.

Working model of spline/patch data

A spline/patch model is best treated as a graph of related data. Control Points define editable locations. Splines connect CPs. Patches define surface areas through CP and spline relationships. Materials, colors, decals, images, and surface settings attach to or reference those geometry pieces. Named Groups and Folders provide human-readable organization over those pieces.

The important lesson is that visual geometry and organization are related but not identical. A Group can identify a part, but that part may depend on CPs, patches, splines, materials, colors, decals, and images elsewhere in the file.

Why known-good templates matter

Our more successful edits worked because we started with a model that Animation:Master already accepted, then made small predictable edits. This is safer than inventing a complete file from scratch.

Recommended workflow:

- Create a minimal valid object in Animation:Master.
- Save it and inspect the text structure.

- Duplicate or modify only validated block patterns.
- Give each generated part a clear Named Group.
- Assign visible diagnostic colors.
- Reopen in Animation:Master after each generation stage.
- Keep the previous known-good model as the rollback point.

For .mdl repairs involving progressively taller duplicated cube objects, the working fix was to start from the last known-good tall colored model and only shift explicit height coordinates so each object's bottom landed at Y=0, without changing the overall structure.

Coordinate and unit assumptions

Current working assumptions are: units are centimeters; Y is height; Z is depth; X is horizontal in many generated layouts. These assumptions must be verified with a small diagnostic model before applying large edits.

Named Groups

Named Groups were especially useful because they gave procedural code stable handles. We used group naming to label generated extrusions, assign random diagnostic colors, select patterned subsets such as every third extrusion, isolate a face from a grid, and preserve part identity while modifying geometry.

Group names became metadata. A generated object that is clearly named and colored is much easier to inspect and repair.

Folders

Folders appear to be higher-level organizational containers. They are useful for grouping construction parts, extrusion sequences, grid/face pieces, proxies, and temporary diagnostic elements. A Folder should not automatically be treated as a complete standalone mesh part. Like a Group, it may organize data that depends on external CPs, patches, splines, or materials.

Extraction of model parts

The central caution is that extracting a Named Group is not the same as extracting a complete valid model. A robust extraction needs dependency closure.

A dependency-closed extraction usually needs to:

- Identify the selected Group or Folder.
- Find all patches referenced by the part.
- Find the CPs and splines required by those patches.
- Preserve materials, colors, images, decals, and surface attributes.
- Preserve required hierarchy.
- Reindex references if the new target file requires compact numbering.
- Reopen and verify the result in Animation:Master.

The extraction script in this package therefore extracts text blocks conservatively. It is useful for inspection and fragment gathering, but it does not promise a complete standalone model part.

Patch normals and winding

Our working understanding is that patch normal direction is controlled by patch winding/order. If all normals face the wrong way, reversing the point order of the affected patches may flip the normals. This should be done only after identifying the exact patch statement syntax in a known-good file.

A safe rule: do not blindly reverse every numeric sequence that looks like a patch. First identify the exact patch record syntax, then write a targeted winding tool for that syntax.

Extrusion groups

One strong pattern was to group each extrusion step separately, such as Extrusion1, Extrusion2, and Extrusion3. This made it possible to select or transform patterned subsets. For example, one task moved the CPs of every third Extrude/Extrusion group backward in Z by 10 cm.

Assigning random colors to each generated group also gave immediate visual feedback. It showed whether the generator created the correct number of sections, whether the ordering was right, and whether a selection included too much or too little geometry.

Grid/image to patch workflows

For grid-based face models, the reliable approach is to start with a small grid such as 20x20, treat white or near-white cells as background, create colored cells for non-white areas, group cells by color, and create a separate Extracted_Face_NonWhite group. Only after the concept works should fidelity increase to a 50x50 grid.

The grid script in this package outputs JSON construction plans rather than final .mdl files. That keeps the data inspectable before a true generator is written against a known-good template.

Scripts in this package

am_mdl_structure_scan.py inventories brace-balanced blocks with guessed names, kinds, line ranges, parent IDs, and paths. It is the first pass before editing.

am_mdl_extract_named_blocks.py extracts candidate named blocks and can print dependency hints. It is block extraction, not dependency-closed part extraction.

am_grid_to_patch_plan.py turns a demo face, CSV, or image into a JSON grid-to-patch construction plan. It can also emit pseudo-MDL comments as a map for future generation.

am_group_transform_plan.py reads an inventory JSON and creates a safe transform plan for matching groups, such as every third Extrusion group. It intentionally does not rewrite the model.

Recommended next steps

- Collect known-good .mdl examples for a simple patch, grouped extrusion sequence, and grid/face model.
- Run the scripts against those files.
- Identify exact syntax for CPs, splines, patches, groups, folders, colors, and materials.
- Upgrade the dependency hints into a real dependency graph for that exact syntax.
- Build a dependency-closed extractor for one narrow case.
- Build a validator that compares source and extracted counts.
- Only then build a full grid-to-.mdl generator.

Open questions

- What exact syntax identifies a CP definition versus a CP reference?
- Are patch records consistent enough to reverse winding automatically?
- How are Folders represented when they contain Groups versus when they merely organize them?
- Where are surface color and material assignments stored?

- What references must be reindexed for a partial-model extraction?
- How much can be validated textually before Animation:Master opens the file?

Current best practices

- Keep a last-known-good file at every stage.
- Prefer additive generation over destructive rewriting.
- Name every generated part.
- Color every diagnostic group.
- Use Folders to organize generated constructs.
- Make tiny test files before scaling to large grids.
- Treat extraction as dependency closure, not simple text copying.
- Treat normal flipping as patch-winding surgery.
- Let scripts create inventories and plans before they modify .mdl files.
- Verify in Animation:Master after each important change.