

DEVELOPMENT · 2026-10

A hollow model needs a way out

Seesaw's October work on hollowing and drains shows why a convincing viewport still needs layer-level checks.



By Edwin Rodriguez

Published Oct 14, 2026 · 4 min read

October launch collection · Story context 2026-10-03



A hollowed 3D model can look completely reasonable from the outside. The interesting questions are inside: how the cavity is formed, where it connects to the surface, and what the sliced representation actually contains.

[Seesaw's early October work](#) included hollowing and drain preparation, followed by readback and negative-fixture checks. It's a good example of a general engineering problem: a visual edit has to survive conversion into the representation another system will consume.

These are software checks. Printer and material testing still need to happen, so keep the first physical trials small and focused on calibration.

| Follow the edit through the pipeline

The source model, the prepared geometry, the slice archive, and the encoded printer file are related representations. They can disagree if an edit is dropped or transformed incorrectly along the way.

A drain placed on the source model needs to remain associated with that model as the user rotates or moves it. A saved project needs enough information to reconstruct the intended edit. A reopened project should require a fresh validated slice before export readiness is restored.

That makes coordinates part of the product contract. Is the drain position stored in model space or world space? Which transforms apply when the model moves? An implementation needs a consistent answer, and the test should move the model enough to expose a mistaken assumption.

| Use a negative fixture with a clear purpose

A successful hollow example tells you the supported path can work. A deliberately trapped or otherwise invalid case tells you whether the checker refuses the result it is meant to catch.

Choose a small fixture where you can explain the expected failure. Keep the source unchanged when comparing a correction. Record which stage found the issue and whether export remained blocked.

The early October Seesaw work included a deliberately trapped-resin fixture. It gave the checker a problem it should catch, while leaving more complicated cavity shapes for further testing.

For another product, the same pattern might be a document with a deliberately missing reference or a data export with a broken relationship. The negative example should target the exact assumption that could otherwise slip through.

| Inspect layers and read back the final file

The viewport helps a person place geometry, but the printer consumes encoded layer information. Inspect the layers produced by the native tools and verify the final encoded representation.

A useful review sequence is:

1. Save the original model and the explicit preparation settings.
2. Apply the hollowing and drain changes through the actual interface.
3. Slice using the pinned backend combination.
4. Inspect the relevant cavity and opening across neighboring layers.
5. Run the supported issue checks and preserve their result.
6. Read back the encoded output and compare the expected properties.

Use a small model while developing the test so failures are quick to investigate. Once the simple path is understood, add more demanding geometry and supported transformations.

| Protect the project when preparation fails

A failed attempt should leave the editable project in a recognizable state. Keep the user's original geometry and settings, show what failed, and avoid advertising a stale file as ready.

Cancellation needs the same care. If the user edits a drain while a previous slice is running, the old result shouldn't overwrite the newer intent. Bind the result to the input snapshot and refuse adoption if that snapshot is no longer current.

Undo is another useful test. After undoing a preparation change, verify both the geometry and the output readiness state. A visually restored model with a still-approved old export is a subtle mismatch.

| Carry the lesson into other tools

Whenever a product turns an editable scene into a downstream artifact, inspect the downstream artifact. Keep transforms explicit, add a negative fixture, and make stale results lose their authority.

Try this with a small model you understand well. Move it, rotate it, reopen it, and inspect the layers after each change. If the drain stops lining up with the intended opening, you have a small, understandable case to debug.

| Sources & further reading

Checked 2026-10-14 · Story context 2026-10-03

1. [Seesaw source and documented limits](#) (2026-10-14)
2. [Seesaw backend validation](#) (2026-10-14)
3. [UVTools](#) (2026-10-14)

[Read the current article online →](#)

Support the workshop

Explore software and learning offers at [StackSocial](#). Affiliate link: ProgreTech may earn a commission from purchases. Check current price, license and renewal terms before buying.

Find our projects at progretech.com.

ProgreTech LLC · Build / Learn / Improve / Together

This file contains its text, illustrations and fonts. No account, scripts or network connection required to read. External reference links need internet access.