PLAYTEX AI / FIELD GUIDE 01

THE TEXTURE TROUBLESHOOTING FIELD GUIDE

WRITTEN BY THE PLAYTEX AI EDITORIAL TEAM

The Texture Troubleshooting Field Guide

Fast, evidence-based fixes for PBR maps, alpha, GLB assets, and texture budgets

PLAYTEX AI EDITORIAL TEAM

First edition / August 2026

PLAYTEX AI

https://www.playtex.ai

BEFORE YOU DIAGNOSE

What this guide can prove

THE CORE PROMISE

Every issue begins with a visible symptom, isolates one variable, and ends with a correction you can verify. The diagrams are deliberately simple. They show the diagnostic signal, not a decorative material render.

THREE KINDS OF EVIDENCE

MEASURABLE

Pixel values, image dimensions, channels, alpha, references, and file structure can be inspected directly.

CONVENTION

Normal orientation, packing, import flags, and extensions must be checked against the actual target.

JUDGMENT

Whether wood feels too plastic or repetition is too obvious requires a controlled comparison and a human decision.

Scope: real-time PBR texture workflows. Engine behavior changes, so verify project and importer versions. Photo-derived height, normal, and roughness are estimates unless captured with a measurement workflow.

FIND THE SYMPTOM

Contents

Use the flowchart first when more than one cause seems plausible.

FIND THE SYMPTOM

Contents / continued

Use the flowchart first when more than one cause seems plausible.

A REPEATABLE SIX-PART PATTERN

Fix the failure without changing five things

1

SYMPTOM

Describe only what you can see.

2

LIKELY CAUSE

Rank explanations before editing.

3

TWO-MINUTE TEST

Change one variable in a controlled scene.

4

CORRECTION

Fix the actual role, value, or binding.

5

PREVENTION

Turn the lesson into a preset or check.

6

BEFORE / AFTER

Verify the diagnostic signal, not decoration.

FAST RULE

Keep the camera, light, shader, and exposure fixed. Change one map or setting. Then decide.

EVIDENCE ORDER

Start with the cheapest decisive test

01

SLOT

Is the texture connected to the intended input?

02

ROLE

Is it color, normal, scalar data, or alpha?

03

CONVENTION

Does orientation or semantic match the target?

04

VALUES

Do channels and ranges match the material?

05

DELIVERY

Did compression, mips, packing, or packaging change it?

06

JUDGMENT

Only now tune artistry under controlled light.

WHY THIS ORDER WORKS

A wrong binding or color-space flag cannot be fixed by better painting. Remove deterministic faults before subjective tuning.

START HERE

What is the first visible failure?

Choose the strongest symptom. If two branches apply, test the cheaper one first.

RELIEF READS BACKWARD

Go to normal orientation and import checks.

FINISH OR MATERIAL ID FEELS WRONG

Go to roughness, plastic response, and metallic masks.

A LINE, GRID, OR FIXED LIGHT APPEARS

Go to seams, repetition, and baked illumination.

TRANSPARENT EDGES FAIL

Go to hidden RGB and alpha representation.

IT BREAKS AFTER IMPORT OR AT DISTANCE

Go to color space, mips, GLB bindings, and budgets.

BRANCH A

Surface response and tiling

DO BUMPS BECOME DENTS?

Flip green once. If only tangent Y changes, read Issues 01 and 02.

IS POLISH REVERSED?

Test roughness with 0.1 and 0.9 constants. Read Issue 03.

DO UNRELATED MATERIALS SHARE ONE SHINE?

Set known nonmetal to metallic 0 and sweep roughness. Read Issues 04 and 05.

DOES A CROSS OR GRID APPEAR IN 3 BY 3?

Offset by 50 percent, then inspect each map. Read Issue 06.

DOES A SHADOW STAY WHEN THE LIGHT MOVES?

View base color unlit and rotate the key. Read Issue 07.

BRANCH B

Transparency, import, and runtime

FRINGE CHANGES WITH BACKGROUND OR MIPS?

Inspect hidden RGB, edge dilation, and atlas padding. Read Issue 08.

SEMI-TRANSPARENT EDGES DARKEN OR GLOW?

Inspect a 50 percent alpha pixel and blend mode. Read Issue 09.

MAP VALUES SHIFT AFTER IMPORT?

Separate color textures from numeric data. Read Issue 10.

BLOCKS, SHIMMER, OR DISTANCE BLUR?

Compare uncompressed, force mips, then move the camera. Read Issue 11.

MODEL LOSES MAPS OR MEMORY?

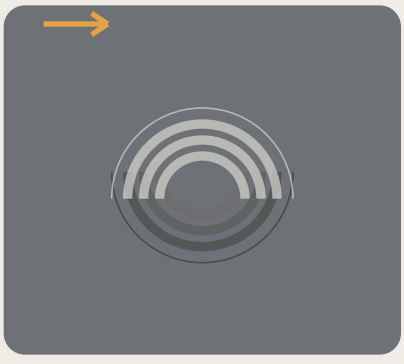
Validate links first, then inventory runtime cost. Read Issues 12 and 13.

ISSUE 01 / SYMPTOM

When a normal map looks inverted

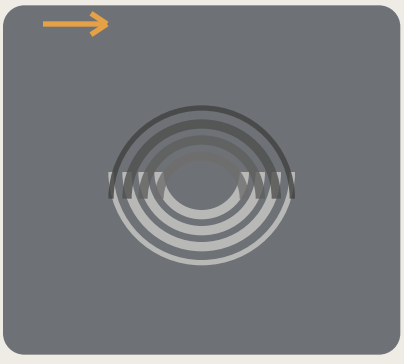
A raised edge reads like a groove, or a dent reads like a bump.

RELIEF READS BACKWARD



same ridge / same light

RELIEF READS CORRECTLY



same ridge / same light

SYMPTOM

Convex details appear concave under a familiar light direction. The error may be obvious on one axis, or the entire surface may look swollen, stamped inward, or strangely flat.

LIKELY CAUSE

Most often, the normal map convention does not match the target. But a wrong texture slot, color-space conversion, object-space map, mirrored tangent basis, or excessive strength can produce a similar read.

TWO ANCHORS

1. A flat tangent-space normal is near RGB 128, 128, 255.

2. If flipping green swaps bump and dent, the Y convention was the fault.

TWO-MINUTE TEST

Change one variable. Read the result.

RUN THIS IN ORDER

1. Put the material on a plain plane or sphere with one grazing key light.
2. Disable only the normal map. Confirm that the base mesh shades normally.
3. Restore the map and flip only its green channel.
4. If the relief direction swaps, fix the convention. If it does not, inspect the import type, tangent basis, and map type.

RANK LIKELY CAUSES

- Green-channel convention mismatch.
- Normal imported as color instead of normal/data.
- Object-space map placed in a tangent-space slot.
- Mirrored UV or tangent-basis mismatch.
- Height-to-normal sign or strength is wrong.

READ THE RESULT

- Green flip fixes it: convention mismatch.
- Disabling sRGB fixes it: data was gamma-decoded.
- Only mirrored islands fail: tangent or UV issue.
- Everything stays wrong: verify the texture slot and map type.

KEEP FIXED

Camera, mesh, shader, exposure, and light. If those move too, the test cannot isolate the texture fault.

CORRECTION + PREVENTION

Repair the cause, then make it hard to repeat

CORRECTION

- Export or convert to the target convention; do not flip multiple times.
- Import as a normal/data texture and use the engine's normal-map setting.
- Rebuild tangents when the mesh, UVs, or exporter changed.
- Reduce strength only after orientation and import settings are correct.

PREVENTION

- Suffix files with _n_ogl or _n_dx.
- Keep one known source and export target-specific copies.
- Use a standard ridge test material in every engine.
- Record texture type and convention in the asset manifest.

AVOID THE FALSE FIX

Do not rotate the light until the map appears right. That can hide the fault without fixing it.

PLAYTEX AI ROUTE

PBR Engine Converter

Convert orientation and export a named target profile.

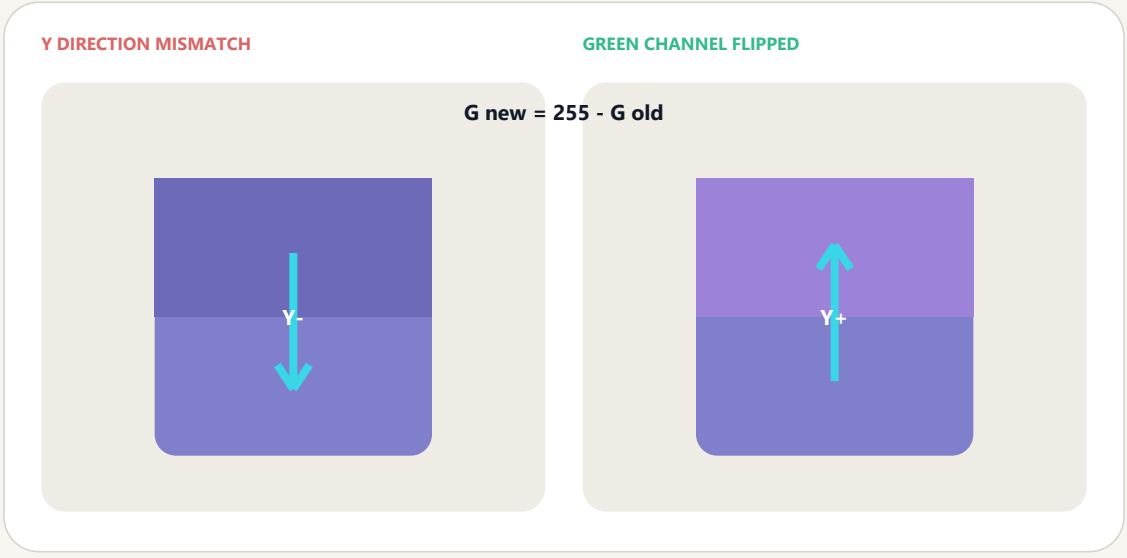
playtex.ai/tools/pbr-engine-converter

Technical sources: 1, 2, 3, 7. Full URLs begin on page 59.

ISSUE 02 / SYMPTOM

OpenGL Y+ versus DirectX Y- normals

The difference is one channel, but the names are conventions, not guarantees.



SYMPTOM

Slopes facing along tangent Y shade in the opposite direction while slopes along tangent X still look correct. The result often feels almost right, which makes this error easy to miss.

LIKELY CAUSE

Tangent-space Y is encoded with opposite signs. In 8-bit RGB, the conversion is G_new = 255 - G_old. Red and blue stay unchanged.

TWO ANCHORS

1. glTF defines tangent-space normal maps as +Y.
2. Unity documents Y+ normals; Unreal exposes a Flip Green Channel import option.

TWO-MINUTE TEST

Change one variable. Read the result.

RUN THIS IN ORDER

1. Choose a known asymmetric ridge, not random noise.
2. Light it from the top of the UV direction and note which slope brightens.
3. Flip green once and compare without changing the light or camera.
4. Keep the version whose relief agrees with the known geometry and the target documentation.

RANK LIKELY CAUSES

- An OpenGL-style map entered a Y-pipeline, or vice versa.
- A converter and engine both flipped green.
- A filename implied one convention but contained the other.
- A packed or recompressed normal was edited as ordinary color.

READ THE RESULT

- Only vertical-facing slopes swap: Y convention confirmed.
- Left and right also swap: red may be inverted or tangents differ.
- No visible change: map strength, slot, or normal decoding may be wrong.
- Different meshes disagree: compare tangent generation and mirrored UVs.

KEEP FIXED

Camera, mesh, shader, exposure, and light. If those move too, the test cannot isolate the texture fault.

CORRECTION + PREVENTION

Repair the cause, then make it hard to repeat

CORRECTION

- Flip only green exactly once, preferably during a named export step.
- Use +Y for glTF. Follow the importer and project convention for other targets.
- Re-normalize vectors after channel processing when the tool does not do it.
- Retest on the same ridge under the same light.

PREVENTION

- Treat OpenGL and DirectX as shorthand, then verify the actual target.
- Store the convention in filenames and metadata.
- Do not hand-edit purple normal maps in an sRGB paint workflow.
- Include one orientation test in automated preflight.

AVOID THE FALSE FIX

Do not infer the convention from purple pixels alone. Without known shape or metadata, the image can be ambiguous.

PLAYTEX AI ROUTE

PBR Engine Converter

Flip green once and preserve a traceable source preset.

playtex.ai/tools/pbr-engine-converter

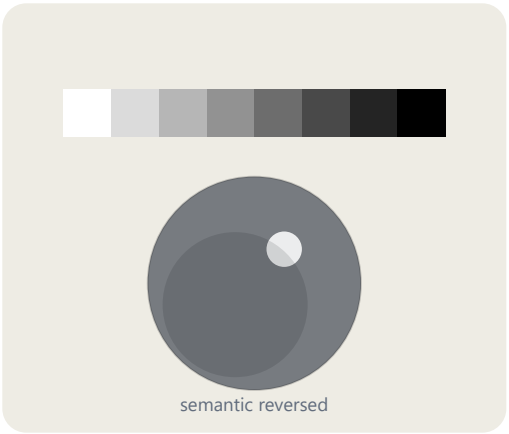
Technical sources: 1, 2, 7. Full URLs begin on page 59.

ISSUE 03 / SYMPTOM

Roughness treated as smoothness

The right map with the wrong meaning can reverse the whole material.

POLISH IS BACKWARD



ROUGHNESS IS CORRECT

SYMPTOM

Dusty stone reflects like polished glass, while a varnished surface looks chalky. Dark and light regions respond opposite to the intended finish.

LIKELY CAUSE

A smoothness or gloss map was read as roughness. In the common roughness convention, 0 is smooth and 1 is rough. Smoothness is the inverse: smoothness = 1 - roughness.

TWO ANCHORS

1. Dark roughness values produce tighter, clearer reflections.
2. Roughness is data and should not receive an sRGB color transform.

TWO-MINUTE TEST

Change one variable. Read the result.

RUN THIS IN ORDER

1. Replace the map with a constant value near 0.1 and observe the highlight.
2. Replace it with a constant near 0.9 without changing the light.
3. If 0.1 is broad and dull, the slot is not behaving as roughness.
4. Reconnect the texture, inspect its selected channel, then invert once if its semantic is smoothness.

RANK LIKELY CAUSES

- Smoothness map connected to a roughness input.
- Wrong packed channel selected.
- Data texture imported as sRGB.
- Map inverted in both export and engine setup.
- A constant or missing map replaced authored variation.

READ THE RESULT

- Constants behave, texture does not: wrong map or channel.
- Constants are reversed: shader input is smoothness/gloss.
- Midtones shift after import: color-space flag is likely wrong.
- No response: map is unbound, overridden, or masked out.

KEEP FIXED

Camera, mesh, shader, exposure, and light. If those move too, the test cannot isolate the texture fault.

CORRECTION + PREVENTION

Repair the cause, then make it hard to repeat

CORRECTION

- Invert once when converting smoothness to roughness.
- Select the documented packed channel; glTF uses green for roughness.
- Import the map as linear/non-color data.
- Judge the result under a reflection-rich neutral environment.

PREVENTION

- Use explicit suffixes: _rough or _smooth.
- Write channel packing beside every export preset.
- Keep a black-to-white roughness wedge in the test scene.
- Never mix color and data channels in one sRGB texture.

AVOID THE FALSE FIX

Do not compensate by changing metallic or specular. Fix the semantic first.

PLAYTEX AI ROUTE

PBR Map Generator

Preview roughness before export and keep aligned map sets.

playtex.ai/pbr-map-generator

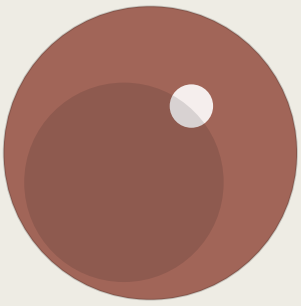
Technical sources: 1, 5, 6, 9. Full URLs begin on page 59.

ISSUE 04 / SYMPTOM

Materials that look like plastic

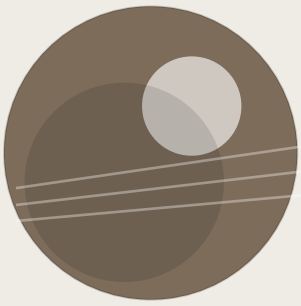
Plastic is often a roughness diagnosis wearing a lighting disguise.

UNIFORM PLASTIC READ



tight uniform highlight

MATERIAL-SPECIFIC RESPONSE



broader, structured response

SYMPTOM

Wood, stone, fabric, or painted metal all share the same tight white highlight. The base color may look saturated, the surface too clean, and the microstructure absent.

LIKELY CAUSE

Roughness is too low or too uniform. Wrong metallic values, baked light in base color, missing normal detail, and an uninformative preview environment often reinforce the plastic look.

TWO ANCHORS

1. Roughness shapes highlight width; metallic changes the reflection model.

2. A good diagnosis changes one channel at a time under fixed lighting.

TWO-MINUTE TEST

Change one variable. Read the result.

RUN THIS IN ORDER

1. Switch to a neutral studio environment and freeze the camera and exposure.
2. For a known nonmetal, set metallic to 0.
3. Sweep roughness through 0.2, 0.5, and 0.8 while watching highlight width.
4. Restore maps one by one: base color, roughness, normal, then AO.

RANK LIKELY CAUSES

- Roughness values are uniformly dark.
- Nonmetal areas contain metallic values.
- Albedo contains hard highlights or shadows.
- Normal detail is absent, noisy, or too strong.
- The preview light offers no useful reflections.

READ THE RESULT

- Roughness sweep fixes it: rebuild the roughness range.
- Metallic 0 fixes it: metal mask was contaminated.
- Unlit view reveals glare: albedo needs delighting.
- Only one environment fails: lighting, exposure, or reflection setup is misleading.

KEEP FIXED

Camera, mesh, shader, exposure, and light. If those move too, the test cannot isolate the texture fault.

CORRECTION + PREVENTION

Repair the cause, then make it hard to repeat

CORRECTION

- Author broad roughness structure before micro-noise.
- Set dielectric regions to metallic 0 and isolate true metal.
- Remove baked illumination from base color cautiously.
- Add restrained normal detail that matches real scale.

PREVENTION

- Review in at least two neutral light rigs.
- Use measured or trusted references for material families.
- Check each map alone before judging the combined shader.
- Avoid identical roughness ranges across unrelated materials.

AVOID THE FALSE FIX

Do not turn specular to zero as a general cure. It hides the symptom and creates a different physical error.

PLAYTEX AI ROUTE

Material Preflight

Check map roles and suspicious value ranges before engine import.

playtex.ai/tools/material-preflight

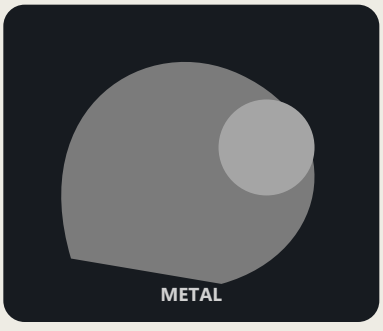
Technical sources: 5, 11, 12. Full URLs begin on page 59.

ISSUE 05 / SYMPTOM

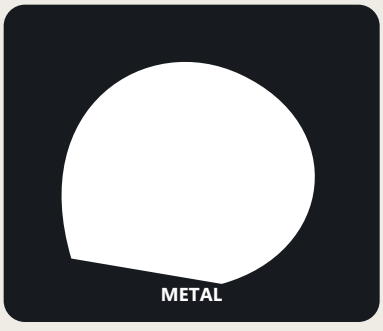
Metallic maps with incorrect gray values

For pure surfaces, metallic is usually a material identity, not a shading dial.

MUDDY GRAY MASK



CLEAR MATERIAL REGIONS



SYMPTOM

Metal looks weak or dusty, paint reflects with a metallic tint, and transitions feel chemically implausible. A mask that looks pleasing in grayscale can still be wrong for the material model.

LIKELY CAUSE

Pure dielectric and pure metal regions were painted with broad middle-gray values. Gray is valid for coverage blends, anti-aliased boundaries, layered mixtures, or materials that are intentionally neither pure endpoint.

TWO ANCHORS

1. For common metal/rough PBR, nonmetal is 0 and metal is 1.
2. Intentional gray should describe a mixture or partial coverage, not compensate for lighting.

TWO-MINUTE TEST

Change one variable. Read the result.

RUN THIS IN ORDER

1. Identify one unquestionably bare-metal region and one dielectric coating or dirt region.
2. Temporarily set them to 1 and 0 respectively.
3. Inspect reflections under neutral light, then view the mask histogram.
4. If endpoints improve the material, rebuild by material identity and reserve gray for real blends.

RANK LIKELY CAUSES

- The map was derived from image brightness.
- Paint, oxide, dirt, or rust was left metallic.
- A soft brush created wide gray transition zones.
- sRGB changed midrange data values.
- The wrong packed channel was connected.

READ THE RESULT

- Bare metal improves at 1: its mask was too low.
- Dirt improves at 0: contamination was metallic.
- Only edges need gray: keep a narrow anti-aliased transition.
- The whole result shifts: verify channel packing and color space.

KEEP FIXED

Camera, mesh, shader, exposure, and light. If those move too, the test cannot isolate the texture fault.

CORRECTION + PREVENTION

Repair the cause, then make it hard to repeat

CORRECTION

- Paint broad regions from a material-ID mask.
- Make paint, rust, dust, and most organic deposits nonmetal.
- Keep gray only where coverage or mixture is defensible.
- Import as linear data and verify the selected channel.

PREVENTION

- Build metallic from named material layers, not luminance.
- Review the mask alone in black, white, and histogram views.
- Document any intentional intermediate range.
- Test weathered boundaries at final texel density.

AVOID THE FALSE FIX

Do not use metallic gray to reduce strong reflections. Roughness controls reflection sharpness.

PLAYTEX AI ROUTE

Material Preflight

Flag suspicious metallic ranges and confirm channel assignments.

playtex.ai/tools/material-preflight

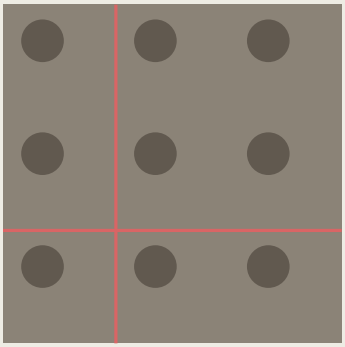
Technical sources: 1, 5, 6. Full URLs begin on page 59.

ISSUE 06 / SYMPTOM

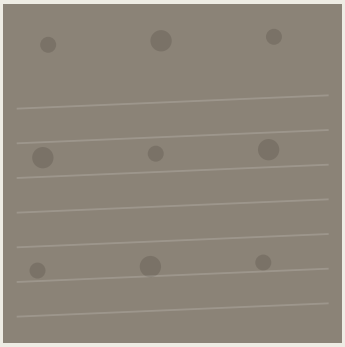
Visible seams and obvious repetition

A texture can be edge-perfect and still reveal its tile from across the room.

SEAM OR REPEAT EXPOSED



EDGES AND RHYTHM HOLD



SYMPTOM

Straight lines appear at tile borders, or a unique knot, stain, crack, or bright patch repeats in a grid. Sometimes base color tiles correctly while normal or roughness still breaks.

LIKELY CAUSE

Opposite edges do not match, low-frequency lighting crosses the source, a memorable feature dominates the tile, maps are misaligned, or the sampler clamps instead of repeats.

TWO ANCHORS

1. Edge continuity and repetition camouflage are separate tests.
2. Every PBR channel must share the same crop, transform, and wrap logic.

TWO-MINUTE TEST

Change one variable. Read the result.

RUN THIS IN ORDER

1. Tile the full material 3 by 3 at the final display scale.
2. Offset the source by 50 percent on both axes to move borders to the center.
3. Inspect albedo, normal, and roughness separately, then blur or squint to expose broad repetition.
4. Switch the sampler between repeat and clamp to confirm runtime behavior.

RANK LIKELY CAUSES

- Color or value discontinuity at opposite edges.
- Directional shadow or exposure gradient.
- One large landmark reveals the repeat frequency.
- Normal, roughness, or AO is shifted from albedo.
- Sampler wrap mode or atlas padding is wrong.

READ THE RESULT

- A cross appears after offset: edge mismatch.
- A motif forms a grid without edge lines: repetition problem.
- Only one channel breaks: map alignment or channel processing.
- Editor is clean, engine is not: sampler, mips, or atlas settings.

KEEP FIXED

Camera, mesh, shader, exposure, and light. If those move too, the test cannot isolate the texture fault.

CORRECTION + PREVENTION

Repair the cause, then make it hard to repeat

CORRECTION

- Heal the centered seam while preserving features that cross it.
- Remove broad lighting gradients before making the tile.
- Break up or redistribute dominant landmarks.
- Apply identical transforms and seam processing to every channel.

PREVENTION

- Review 1 by 1, 3 by 3, and distant tiled views.
- Keep high-contrast unique features away from the source stage.
- Generate all maps from the same aligned master.
- Test final wrap and mip behavior in the target engine.

AVOID THE FALSE FIX

Do not blur the whole texture to hide a seam. It removes useful detail and often leaves the low-frequency line.

PLAYTEX AI ROUTE

Image-to-Texture Generator

Repair edge continuity and inspect the repeated tile before map extraction.

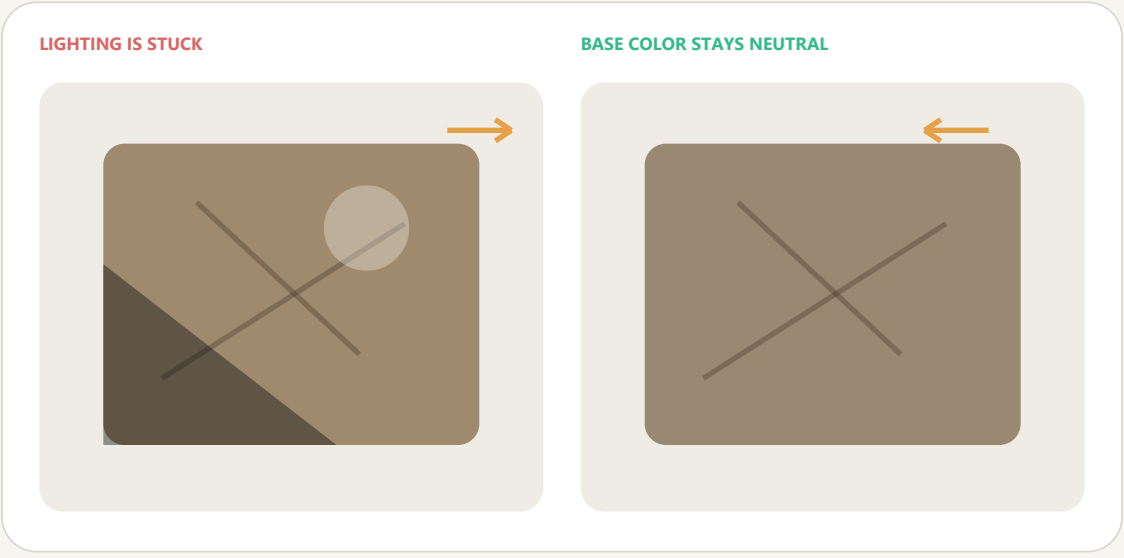
playtex.ai/image-to-texture-generator

Technical sources: 1, 10, 17. Full URLs begin on page 59.

ISSUE 07 / SYMPTOM

Baked shadows and highlights in albedo

Base color should not argue with the scene lighting.



SYMPTOM

A shadow remains visible when the light moves, highlights appear from two directions, cavities become too dark, or a surface looks flat because the photo's lighting conflicts with the shader.

LIKELY CAUSE

The source image contains illumination: cast shadows, ambient occlusion, directional gradients, specular glare, or white-balance shifts. Those cues were mistaken for intrinsic color.

TWO ANCHORS

1. Albedo/base color represents surface color, not a finished lit photograph.

2. A single photograph rarely contains enough information to recover physical ground truth exactly.

TWO-MINUTE TEST

Change one variable. Read the result.

RUN THIS IN ORDER

1. View base color in an unlit shader or as a flat image.
2. Rotate the scene key light by about 180 degrees.
3. Look for shadows and highlights that remain fixed to the texture.
4. Compare a heavily blurred copy; broad directional gradients are capture lighting, not fine material color.

RANK LIKELY CAUSES

- Directional or hard light in the capture.
- Specular highlight baked into a nonmetal source.
- AO or cavity darkening multiplied into base color.
- An AI or filter increased dramatic contrast.
- Automatic exposure or white balance varied across the frame.

READ THE RESULT

- Dark and bright cues stay fixed: baked illumination.
- Only the shader moves: base color may be acceptable.
- Blur reveals a large gradient: correct low frequency first.
- Specular color remains in metal: separate reflectance carefully; do not simply desaturate.

KEEP FIXED

Camera, mesh, shader, exposure, and light. If those move too, the test cannot isolate the texture fault.

CORRECTION + PREVENTION

Repair the cause, then make it hard to repeat

CORRECTION

- Use a delighting pass, then inspect the result under new lighting.
- Remove broad gradients and isolated glare with restrained masks.
- Recapture under diffuse or cross-polarized light when accuracy matters.
- Rebuild roughness and normal from evidence, not from leftover light cues alone.

PREVENTION

- Capture with flat, even, color-controlled illumination.
- Avoid hard shadows and clipped highlights.
- Keep the original photo beside the corrected estimate.
- Label single-image recovery as an estimate, not a measurement.

AVOID THE FALSE FIX

Do not remove every dark value. Real pigments and stains can be dark; remove lighting structure, not material identity.

PLAYTEX AI ROUTE

Material Delighter

Estimate a more neutral base color and compare before/after under relighting.

playtex.ai/tools/material-delighter

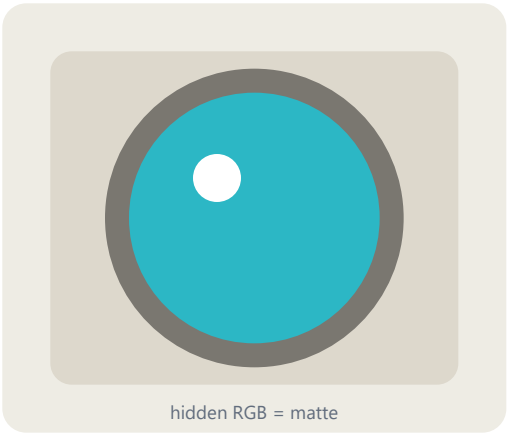
Technical sources: 11, 12. Full URLs begin on page 59.

ISSUE 08 / SYMPTOM

Black or white outlines around transparent PNGs

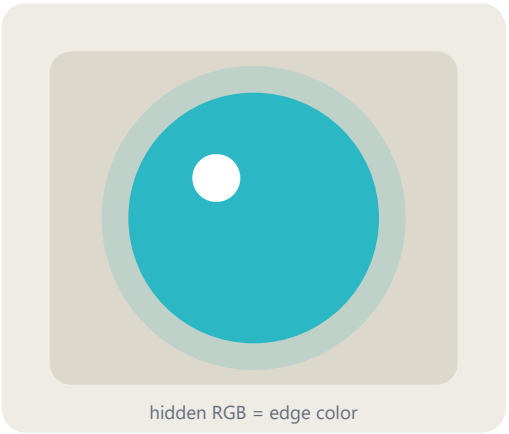
Invisible RGB becomes visible the moment filtering touches an edge.

MATTE COLOR BLEEDS IN



hidden RGB = matte

EDGE COLOR EXTENDS OUT



hidden RGB = edge color

SYMPTOM

Sprites, decals, leaves, hair cards, or UI elements show dark or pale fringes on certain backgrounds, at distance, or only after entering an atlas.

LIKELY CAUSE

Transparent pixels carry black, white, or unrelated RGB. Bilinear filtering and mip generation blend that hidden RGB into visible edge pixels. Atlas neighbors and insufficient padding create a similar fringe.

TWO ANCHORS

1. Alpha 0 does not erase RGB; it only makes the pixel transparent during compositing.

2. Edge-color dilation should change hidden RGB while preserving the alpha channel.

TWO-MINUTE TEST

Change one variable. Read the result.

RUN THIS IN ORDER

1. Display the asset over pure black, pure white, and a saturated color.
2. Inspect RGB with alpha temporarily hidden.
3. Compare nearest filtering to linear filtering and enable mips.
4. If the fringe appears with filtering or distance, inspect hidden RGB and atlas padding first.

RANK LIKELY CAUSES

- RGB under transparent pixels uses a matte color.
- Edge colors were not dilated before mip generation.
- Atlas padding is too small for lower mips.
- Straight data is rendered with premultiplied blending, or vice versa.
- Compression damages thin alpha coverage.

READ THE RESULT

- Black halo on light only: dark hidden RGB.
- White halo on dark only: light hidden RGB.
- Only atlas version fails: neighbor bleed or padding.
- Edge darkens at every scale: alpha mode may also be mismatched.

KEEP FIXED

Camera, mesh, shader, exposure, and light. If those move too, the test cannot isolate the texture fault.

CORRECTION + PREVENTION

Repair the cause, then make it hard to repeat

CORRECTION

- Dilate visible edge colors into transparent pixels without changing alpha.
- Regenerate mips after dilation.
- Increase atlas padding or edge extrusion for the required mip depth.
- Use a compression format that preserves the needed alpha quality.

PREVENTION

- Export transparent art with bleed beyond the cutout.
- Start around 4 to 8 pixels, then scale padding for resolution, atlas, and mips.
- Test on both dark and light backgrounds at final size.
- Keep alpha-mode and atlas settings in the import preset.

AVOID THE FALSE FIX

Do not paint the visible edge darker to cancel one background. The fringe will reappear on another background.

PLAYTEX AI ROUTE

Alpha Bleed Fixer

Extend edge RGB while preserving the original alpha silhouette.

playtex.ai/tools/alpha-bleed-fixer

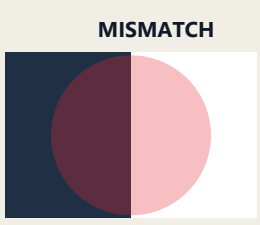
Technical sources: 13, 16, 17. Full URLs begin on page 59.

ISSUE 09 / SYMPTOM

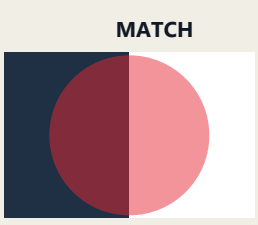
Straight versus premultiplied alpha

The pixels and the blend equation must agree.

DATA AND BLEND DISAGREE



ONE MATCHED ALPHA MODE



SYMPTOM

Semi-transparent edges turn too dark, too bright, or too faint. Particles look additive, smoke shows boxes, and an image may appear correct in one viewer but wrong in the engine.

LIKELY CAUSE

Straight-alpha RGB stores the unattenuated color. Premultiplied RGB already contains RGB times alpha. Using one data representation with the other's blend equation multiplies alpha twice or not at all.

TWO ANCHORS

1. Straight source-over: out = srcRGB * A + dstRGB * (1 - A).
2. Premultiplied source-over: out = srcRGB + dstRGB * (1 - A).

TWO-MINUTE TEST

Change one variable. Read the result.

RUN THIS IN ORDER

1. Pick a pixel with alpha near 0.5 and inspect its RGB numerically.
2. If a nominal red edge stores roughly red 0.5 at alpha 0.5, it is likely premultiplied; if red remains near 1.0, it is likely straight.
3. Render the same asset with the two matching blend modes over black and white.
4. Keep the pair that preserves hue and expected coverage on both backgrounds.

RANK LIKELY CAUSES

- Straight pixels sent to premultiplied blending.
- Premultiplied pixels sent to straight blending.
- The asset was converted twice.
- Filtering occurred in a representation the pipeline did not expect.
- The importer silently changed alpha handling.

READ THE RESULT

- Edge too dark: alpha may be multiplied twice.
- Edge too bright: premultiplied data may use straight blending.
- Opaque center is fine, fringe fails: inspect semi-transparent pixels.
- Different viewers disagree: pipeline metadata or import behavior differs.

KEEP FIXED

Camera, mesh, shader, exposure, and light. If those move too, the test cannot isolate the texture fault.

CORRECTION + PREVENTION

Repair the cause, then make it hard to repeat

CORRECTION

- Choose one alpha representation for the pipeline and match its blend state.
- Convert exactly once, with controlled color-space handling.
- For glTF base color, supply straight alpha as specified.
- Retest after atlas building, compression, and mip generation.

PREVENTION

- Write straight or premultiplied in the asset manifest.
- Keep a 50-percent-alpha color chip in the test suite.
- Avoid repeated save/load conversions through unknown tools.
- Test every renderer that consumes the asset.

AVOID THE FALSE FIX

Do not diagnose alpha mode from opaque pixels. The difference exists in RGB where alpha is below 1.

PLAYTEX AI ROUTE

Alpha Mode Inspector

Inspect semi-transparent RGB and compare compositing assumptions.

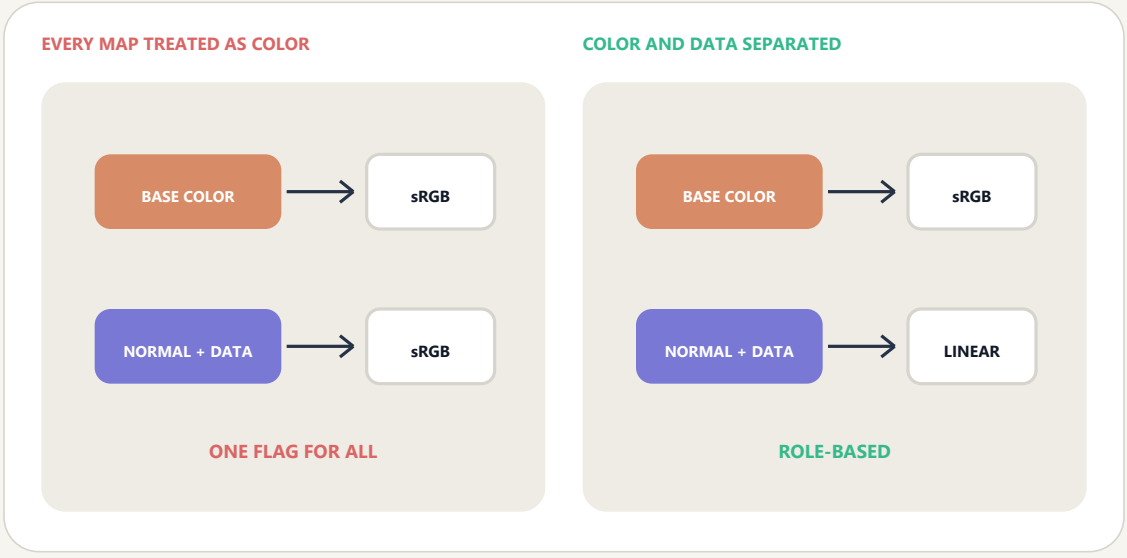
playtex.ai/tools/alpha-mode-inspector

Technical sources: 1, 13, 14, 15. Full URLs begin on page 59.

ISSUE 10 / SYMPTOM

Incorrect color-space handling

Color textures need interpretation. Data textures need preservation.



SYMPTOM

Base color is too dark or washed out, roughness midtones respond incorrectly, packed masks shift, and normals become weak, lumpy, or biased even though the source pixels look familiar.

LIKELY CAUSE

An sRGB transfer was skipped for color or applied to numeric data. Shading happens in linear space, but base-color images are commonly authored/encoded as sRGB. Normal, roughness, metallic, AO, and masks represent data values.

TWO ANCHORS

1. Base color commonly uses sRGB decoding before lighting.

2. Normal and scalar material maps should be sampled as linear/non-color data.

TWO-MINUTE TEST

Change one variable. Read the result.

RUN THIS IN ORDER

1. List every texture by role: visual color or numeric data.
2. Inspect importer color-space flags and shader declarations.
3. Replace suspect scalar maps with constants such as 0.25, 0.5, and 0.75.
4. Compare sampled values or rendered response before and after disabling sRGB on data textures.

RANK LIKELY CAUSES

- Base color marked linear when it is sRGB encoded.
- Data map marked sRGB.
- Gamma was baked into a packed texture before export.
- A mixed texture combines color and data channels.
- An engine and shader both perform conversion.

READ THE RESULT

- Base color corrects when sRGB is enabled: color was being treated as linear.
- Roughness/normal corrects when sRGB is disabled: data was being transformed.
- Packed channels disagree: inspect export packing and per-texture flag.
- Double correction persists: trace shader code and renderer output transform.

KEEP FIXED

Camera, mesh, shader, exposure, and light. If those move too, the test cannot isolate the texture fault.

CORRECTION + PREVENTION

Repair the cause, then make it hard to repeat

CORRECTION

- Mark base color and other authored color textures with the intended color space.
- Mark normal, roughness, metallic, AO, height, and masks as linear/non-color.
- Keep packed material data in a linear texture.
- Apply display encoding only at the output stage, not inside the maps.

PREVENTION

- Assign import rules from unambiguous suffixes.
- Separate color and data in export schemas.
- Include a numeric mid-gray test asset.
- Document renderer and project color-management settings.

AVOID THE FALSE FIX

Do not judge data maps by how attractive they look in an image viewer. Their numeric values are the content.

PLAYTEX AI ROUTE

Material Preflight

Audit role, channel, and color-space expectations before export.

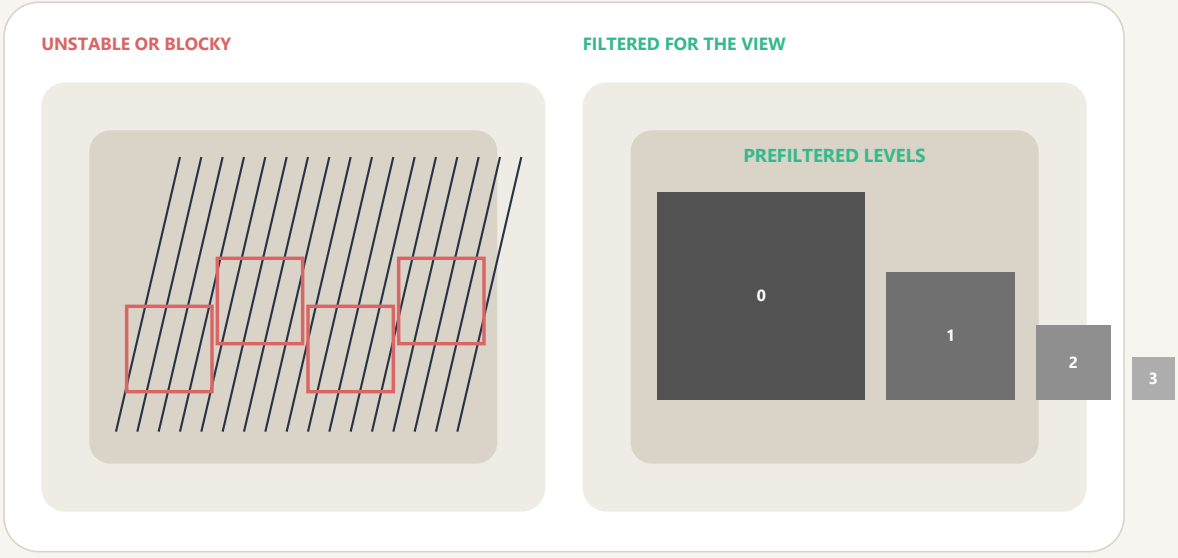
playtex.ai/tools/material-preflight

Technical sources: 1, 3, 6, 7, 9. Full URLs begin on page 59.

ISSUE 11 / SYMPTOM

Compression, mipmaps, shimmering, and blur

Judge the texture while moving, at the distance where it will ship.



SYMPTOM

Fine detail sparkles in motion, diagonal lines crawl, alpha edges break apart, normals show blocks, or the surface is unacceptably soft close up. The source image may still look perfect at 100 percent.

LIKELY CAUSE

Lossy block compression, missing or poor mip levels, unsuitable filtering, negative LOD bias, oversized downscaling, or an unsuitable format for normals and alpha can each trade stability for artifacts.

TWO ANCHORS

1. Mipmaps reduce minification aliasing by prefiltering smaller levels.

2. BC formats encode 4 by 4 blocks; artifacts depend on content and chosen format.

TWO-MINUTE TEST

Change one variable. Read the result.

RUN THIS IN ORDER

1. Compare uncompressed and shipping-format versions in the target renderer.
2. Force individual mip levels and inspect where the defect first appears.
3. Move the camera slowly through the intended distance range.
4. Inspect 4 by 4 block boundaries, alpha edges, and normal gradients separately.

RANK LIKELY CAUSES

- No mips, bad custom mips, or excessive LOD bias.
- Compression format does not suit normal, mask, or alpha content.
- Thin features collapse below one pixel.
- Insufficient alpha bleed or atlas padding.
- Texture resolution or anisotropy does not match viewing conditions.

READ THE RESULT

- Only compressed version fails: format or quality issue.
- Shimmer begins when minified: mip or filtering issue.
- One mip contains a halo: regenerate after edge bleed.
- Close-up is soft in both versions: source resolution, import limit, or LOD.

KEEP FIXED

Camera, mesh, shader, exposure, and light. If those move too, the test cannot isolate the texture fault.

CORRECTION + PREVENTION

Repair the cause, then make it hard to repeat

CORRECTION

- Generate clean mips after final edge and channel processing.
- Choose a target-supported format suited to the data, such as two-channel normal compression where appropriate.
- Preserve atlas padding through every mip.
- Tune anisotropy and LOD only after correct resolution and mips.

PREVENTION

- Preview the exact platform format, not just the source PNG.
- Run a camera-motion test at shipping resolution.
- Protect thin alpha features with coverage-aware workflows when required.
- Choose resolution from texel density, not a habit such as always 4K.

AVOID THE FALSE FIX

Do not disable mipmaps to make a still image sharper. The usual cost is instability and aliasing in motion.

PLAYTEX AI ROUTE

Texture Budget & VRAM Analyzer

Compare dimensions, mip overhead, compression options, and likely optimization savings.

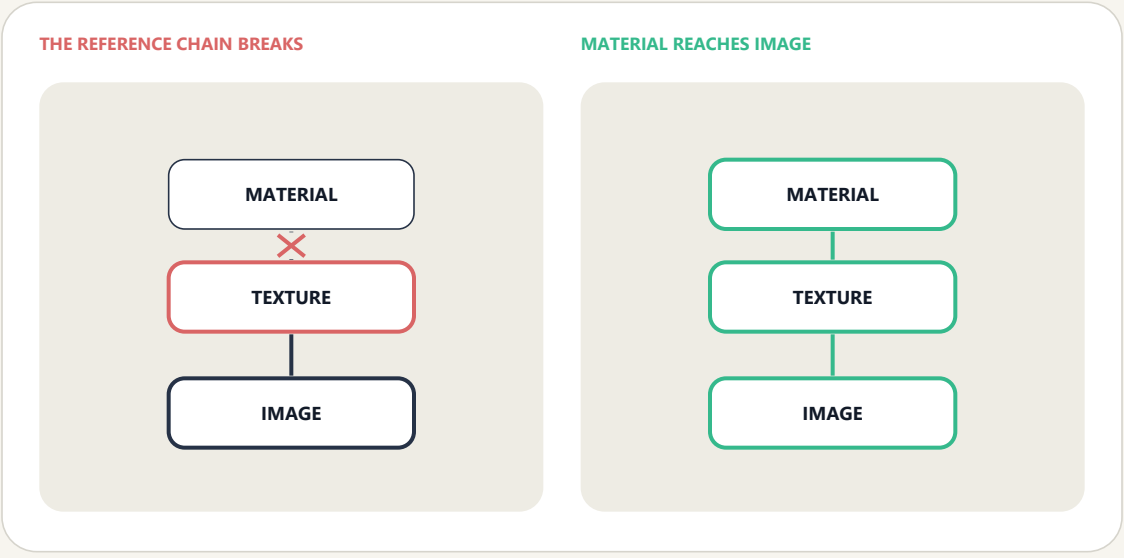
playtex.ai/tools/texture-memory-calculator

Technical sources: 4, 7, 8, 10, 16, 18, 19, 21. Full URLs begin on page 59.

ISSUE 12 / SYMPTOM

Missing or incorrectly bound GLB textures

A texture file can exist and still never reach the material input.



SYMPTOM

The model renders gray, white, pink, or partially textured. It works beside the source folder but fails after upload, or one map is present while roughness, metallic, normal, or alpha behaves incorrectly.

LIKELY CAUSE

A material does not point to the intended texture/image, an external URI is missing or case-mismatched, an extension is unsupported, MIME metadata is wrong, or channel packing does not match glTF's schema.

TWO ANCHORS

1. The chain is material input to texture index to image source to encoded bytes.

2. A .glb container can still reference external resources; do not assume the extension proves self-containment.

TWO-MINUTE TEST

Change one variable. Read the result.

RUN THIS IN ORDER

1. Run the asset through the Khronos glTF Validator and save the report.
2. Open it in the official Sample Viewer or another independent conforming viewer.
3. Inspect each material texture index, image source, URI or bufferView, and exact filename case.
4. Move the asset into a clean empty folder; anything it still needs must be included deliberately.

RANK LIKELY CAUSES

- External files were not shipped or paths changed case.
- Material, texture, or image indices are broken.
- Image bufferView or MIME type is invalid.
- Required extension is unsupported or undeclared.
- Metallic-roughness channels are authored for another engine.

READ THE RESULT

- Validator reports unresolved URI: package the missing file or embed it.
- Viewer agrees with your engine: asset data is likely wrong.
- Viewer works but engine fails: importer or extension support differs.
- Only material response is wrong: check channels, color space, and normal convention.

KEEP FIXED

Camera, mesh, shader, exposure, and light. If those move too, the test cannot isolate the texture fault.

CORRECTION + PREVENTION

Repair the cause, then make it hard to repeat

CORRECTION

- Repair material to texture to image references and exact path casing.
- Embed images as valid bufferViews or ship every external resource together.
- Use correct MIME types and declare required extensions.
- For glTF metallic-roughness, place roughness in green and metallic in blue.

PREVENTION

- Validate every release asset, not only the source scene.
- Test from a clean package and a web-hosted URL.
- Keep filenames portable: simple characters and consistent case.
- Preserve a manifest of included images, roles, and hashes.

AVOID THE FALSE FIX

Do not fix a broken package by copying the entire project directory. That hides undeclared dependencies.

PLAYTEX AI ROUTE

GLB Texture Extractor

List and extract embedded images so bindings can be audited visibly.

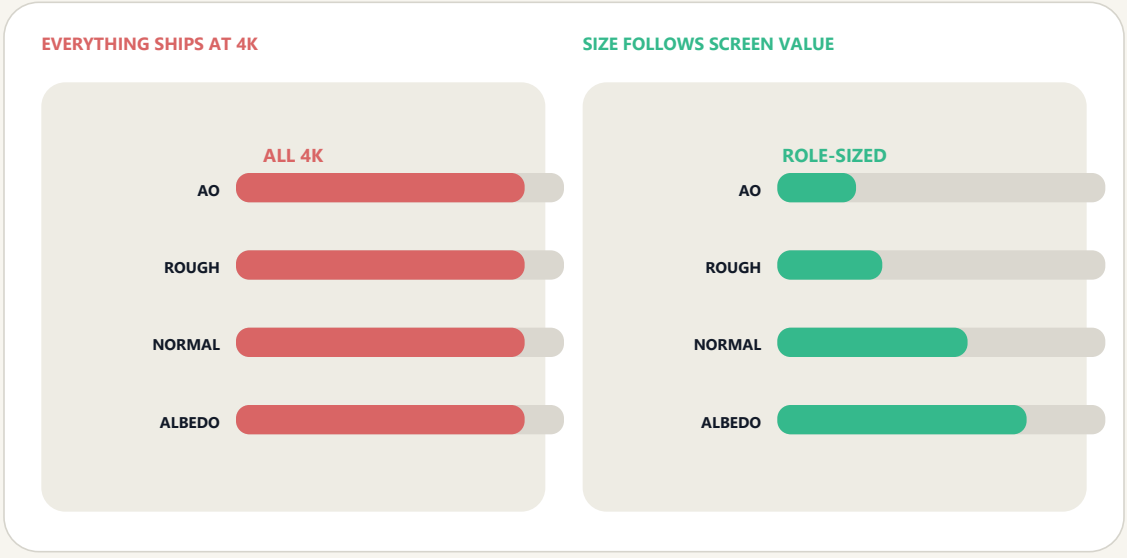
playtex.ai/tools/glb-texture-extractor

Technical sources: 1, 22, 23, 24. Full URLs begin on page 59.

ISSUE 13 / SYMPTOM

Texture memory and oversized assets

Download size, decoded size, and resident GPU memory are different numbers.



SYMPTOM

The scene stutters, uploads slowly, exceeds device budgets, evicts textures, or crashes under load. Individual files may look small on disk while their decoded or GPU-ready versions are much larger.

LIKELY CAUSE

Too many high-resolution maps, unsuitable runtime formats, full mip chains, duplicate assets, unused alpha, or weak streaming policy consume the budget. Container compression does not equal GPU residency.

TWO ANCHORS

1. RGBA8 base bytes = width x height x 4; a complete mip chain is about 4/3 of base for power-of-two textures.

2. 4096-square RGBA8 is 64 MiB at base and about 85.33 MiB with full mips, before runtime-specific overhead.

TWO-MINUTE TEST

Change one variable. Read the result.

RUN THIS IN ORDER

1. Inventory every texture's dimensions, mip count, format, and role.
2. Estimate base and full-mip memory, then sort largest first.
3. Resize the top offender by half in each dimension and compare at the closest real camera distance.
4. Measure the target runtime; treat spreadsheet estimates as planning numbers, not guaranteed residency.

RANK LIKELY CAUSES

- Resolution exceeds visible texel density.
- Uncompressed or poorly matched runtime formats.
- Unused alpha forces a larger format.
- Redundant maps, variants, or duplicated uploads.
- Streaming, mip residency, or reuse is not configured.

READ THE RESULT

- Half-size looks identical: original was oversized for use.
- Disk is small but residency is large: runtime decode/transcode dominates.
- Unused alpha appears widely: reclaim format cost where supported.
- Spikes occur during movement: streaming and mip policy need review.

KEEP FIXED

Camera, mesh, shader, exposure, and light. If those move too, the test cannot isolate the texture fault.

CORRECTION + PREVENTION

Repair the cause, then make it hard to repeat

CORRECTION

- Set resolution from object size, UV density, and closest viewing distance.
- Use platform-supported GPU compression suited to each map.
- Remove unused alpha, maps, duplicates, and unreachable variants.
- Channel-pack compatible linear data and configure streaming deliberately.

PREVENTION

- Create per-platform memory budgets before asset production.
- Track estimated and measured residency separately.
- Gate imports that exceed dimensions or role-specific limits.
- Review top memory contributors in every milestone build.

AVOID THE FALSE FIX

Do not use the PNG or KTX2 file size as a VRAM figure. The runtime format and residency policy determine memory use.

PLAYTEX AI ROUTE

Texture Budget & VRAM Analyzer

Estimate map-set cost, identify top offenders, and compare target formats.

playtex.ai/tools/texture-memory-calculator

Technical sources: 4, 18, 19, 20, 21. Full URLs begin on page 59.

FINAL PREFLIGHT

A clean handoff in four passes

Run this after the material looks good. A good render does not prove a good package.

01

IDENTITY

Know what every map means.

02

IMPORT

Preserve values and conventions.

03

RUNTIME

Test mips, compression, and memory.

04

PACKAGE

Validate the actual delivered files.

SIGN-OFF RULE

Do not sign off from a still image. Move the camera, rotate the light, switch backgrounds, and open the delivered package somewhere clean.

PREFLIGHT 1 OF 3

Map identity

Print it, mark it, or turn each line into an import rule.

FILE ROLES

- [ ] Every file has an explicit role: base color, normal, roughness/smoothness, metallic, AO, height, emission, opacity, or mask.
- [ ] Names distinguish roughness from smoothness and +Y from -Y normals.
- [ ] No map is connected merely because its filename looked familiar.

PIXEL EVIDENCE

- [ ] Flat tangent-space normals are plausible and no channel is accidentally blank.
- [ ] Metallic regions match material identity; broad gray values are intentional.
- [ ] Base color has no fixed shadows, glare, or exposure gradient.

SET ALIGNMENT

- [ ] All channels share dimensions, crop, UV transform, and intended wrap behavior.
- [ ] A 3 by 3 test reveals no edge cross or dominant repeated landmark.
- [ ] Alpha RGB, padding, and atlas extrusion survive the required mip chain.

Initials: _____ *Target build:* _____ *Date:* _____

PREFLIGHT 2 OF 3

Import and shader

Print it, mark it, or turn each line into an import rule.

COLOR SPACE

- [ ] Base color and other authored color textures use the intended color encoding.
- [ ] Normal, roughness, metallic, AO, height, and masks sample as linear/non-color data.
- [ ] Packed textures contain compatible data channels and no mixed sRGB color channel.

CONVENTIONS

- [ ] Normal +Y/-Y matches the target, verified with a known asymmetric ridge.
- [ ] Roughness/smoothness semantic and packed channel match the shader.
- [ ] Alpha representation and blend state are an explicit matched pair.

BINDINGS

- [ ] Every material input reaches the intended texture and channel.
- [ ] Fallback constants are sensible when an optional map is absent.
- [ ] No debug override, material instance, or import preset silently replaces authored values.

Initials: _____ *Target build:* _____ *Date:* _____

PREFLIGHT 3 OF 3

Runtime and package

Print it, mark it, or turn each line into an import rule.

DISTANCE

- [ ] Shipping compression has been reviewed, not only the source PNG.
- [ ] Mips are present, edge-safe, and stable during a slow camera move.
- [ ] Closest-view sharpness and distant shimmer are both acceptable at target resolution.

BUDGET

- [ ] Dimensions follow visible texel density rather than a blanket 4K rule.
- [ ] Estimated memory includes mips and the intended runtime format.
- [ ] Measured target-device residency, streaming, and upload spikes stay within budget.

DELIVERY

- [ ] GLB/glTF passes the Khronos Validator without unresolved resources.
- [ ] Every external URI ships with exact filename case, or images are validly embedded.
- [ ] The final package opens from a clean directory and a production-like hosted URL.

Initials: _____ *Target build:* _____ *Date:* _____

QUICK REFERENCE 01

Normal and color-space facts

These are selected facts, not a substitute for the target project's import preset.

TARGET	NORMAL Y	COLOR	DATA / PACKING
glTF 2.0	+Y	Base color sRGB	Normal/scalars linear; rough G, metal B
Unity	Y+ documented	Albedo as color	Normal/data linear; smoothness shader-dependent
Unreal	Verify; Flip Green exists	Base Color as color	Masks linear; packing is project-dependent
Three.js	Verify asset convention	Use SRGBColorSpace	Use NoColorSpace for non-color data

IMPORTANT

OpenGL and DirectX are useful shorthand for normal orientation, but the actual importer, shader, and asset profile are the authority.

Sources: 1, 2, 3, 6, 7, 9.

QUICK REFERENCE 02

Memory estimates for a 4096-square map

Planning numbers only. Runtime alignment, streaming, and platform details can change residency.

FORMAT	BASE LEVEL	FULL MIPS	BLOCK
RGBA8	64.00 MiB	about 85.33 MiB	4 bytes/pixel
BC1	8.00 MiB	about 10.67 MiB	8 bytes / 4 by 4
BC5 or BC7	16.00 MiB	about 21.33 MiB	16 bytes / 4 by 4

FORMULAS

RGBA8 base bytes = width x height x 4
BC1 per level = ceil(width/4) x ceil(height/4) x 8
BC5/BC7 per level = ceil(width/4) x ceil(height/4) x 16

A complete power-of-two mip chain approaches 4/3 of the base level. Measure the target build before calling any estimate exact.

Sources: 4, 18, 19, 20, 21.

QUICK REFERENCE 03

Straight and premultiplied alpha

A single 50-percent-alpha color chip can reveal the mismatch.

STRAIGHT

out = srcRGB x A + dstRGB x (1 - A)

RGB stores the unattenuated source color.

PREMULTIPLIED

out = srcRGB + dstRGB x (1 - A)

RGB already contains source color x alpha.

PIXEL CLUE

At alpha 0.5, a nominal red edge stores red near 1.0 when straight and near 0.5 when premultiplied. Metadata and pipeline behavior still matter, so confirm by rendering over dark and light backgrounds.

GLTF NOTE

glTF base-color alpha is straight, not premultiplied. Match every other pipeline to its documented storage and blend state.

Sources: 1, 13, 14, 15.

FROM DIAGNOSIS TO ACTION

PLAYTEX AI tools

Use a tool after the two-minute test identifies the class of fault.

PLAYTEX AI

PBR Map Generator

Generate and preview aligned base color, normal, roughness, metallic, height, AO, and emission maps.

playtex.ai/pbr-map-generator

PLAYTEX AI

PBR Engine Converter

Convert normal orientation and map packing for named target workflows.

playtex.ai/tools/pbr-engine-converter

PLAYTEX AI

Material Delighter

Estimate a more lighting-neutral base color from a photographed source.

playtex.ai/tools/material-delighter

PLAYTEX AI

Alpha Bleed Fixer

Extend visible edge colors through transparent RGB without changing alpha.

playtex.ai/tools/alpha-bleed-fixer

PLAYTEX AI

Alpha Mode Inspector

Inspect straight-versus-premultiplied evidence at semi-transparent pixels.

playtex.ai/tools/alpha-mode-inspector

FROM DIAGNOSIS TO ACTION

PLAYTEX AI tools / continued

Use a tool after the two-minute test identifies the class of fault.

PLAYTEX AI

GLB Texture Extractor

List and extract images embedded in GLB assets for binding review.

playtex.ai/tools/glb-texture-extractor

PLAYTEX AI

Material Preflight

Audit roles, channels, dimensions, color-space expectations, and suspicious map values.

playtex.ai/tools/material-preflight

PLAYTEX AI

Texture Budget & VRAM Analyzer

Estimate download size, decoded GPU memory, mip overhead, and optimization savings.

playtex.ai/tools/texture-memory-calculator

PLAYTEX AI

PBR Texture-Set Rotator

Rotate every material map together while transforming tangent-space normal vectors correctly.

playtex.ai/tools/pbr-texture-set-rotator

PLAYTEX AI

Image-to-Texture Generator

Turn a photo or scan into a clean tileable source before extracting material maps.

playtex.ai/image-to-texture-generator

SOURCES 1 OF 4

Primary specifications and documentation

Accessed for this edition in August 2026. Product pages can change after publication.

01 **Khronos Group / glTF 2.0 Specification**

Normal +Y, texture/image links, color spaces, alpha, and metallic-roughness packing.

https://registry.khronos.org/glTF/specs/2.0/glTF-2.0.html

02 **Unity Technologies / Normal map material parameter**

Unity Y+ normal convention and normal-map import context.

https://docs.unity3d.com/6000.1/Documentation/Manual/StandardShaderMaterialParameterNormalMap.html

03 **Unity Technologies / TextureImporter.sRGBTexture**

sRGB decoding for color textures and linear handling for normal/data textures.

https://docs.unity3d.com/ja/current/ScriptReference/TextureImporter-sRGBTexture.html

04 **Unity Technologies / Introduction to mipmaps**

Minification stability and the approximate 33 percent memory overhead of full mips.

https://docs.unity3d.com/cn/2020.3/Manual/texture-mipmaps-introduction.html

05 **Epic Games / Physically Based Materials**

Roughness response and metallic endpoint guidance.

https://dev.epicgames.com/documentation/en-us/unreal-engine/physically-based-materials-in-unreal-engine

06 **Epic Games / Using Texture Masks**

Packed material masks and disabling sRGB for data textures.

https://dev.epicgames.com/documentation/en-us/unreal-engine/using-texture-masks-in-unreal-engine

SOURCES 2 OF 4

Primary specifications and documentation

Accessed for this edition in August 2026. Product pages can change after publication.

07 **Epic Games / Texture Properties**

Normal-map sRGB behavior, mip filtering, and Flip Green Channel.

https://dev.epicgames.com/documentation/en-us/unreal-engine/texture-properties?application_version=4.27

08 **Epic Games / Texture Asset Editor**

Current texture import, mip, and compression controls.

https://dev.epicgames.com/documentation/unreal-engine/texture-asset-editor-in-unreal-engine?lang=en-US

09 **Three.js / Color Management**

sRGB color textures and NoColorSpace data textures.

https://threejs.org/manual/en/color-management.html

10 **Three.js / Textures**

Sampling, wrapping, filtering, and mip behavior.

https://threejs.org/manual/en/textures.html

11 **Roblox / SurfaceAppearance and PBR textures**

Lighting-neutral albedo and core map semantics.

https://create.roblox.com/docs/art/modeling/surface-appearance

12 **Adobe / Delight filter in Substance 3D Sampler**

Removing baked lighting from base-color estimates.

https://experienceleague.adobe.com/en/docs/substance-3d-sampler/using/filters/tools/delight-ai-powered

SOURCES 3 OF 4

Primary specifications and documentation

Accessed for this edition in August 2026. Product pages can change after publication.

13 **Microsoft / Premultiplied alpha in Win2D**

Straight and premultiplied source-over equations.

https://learn.microsoft.com/en-us/windows/apps/develop/win2d/premultiplied-alpha

14 **W3C / Compositing and Blending Level 1**

Source-over compositing model.

https://www.w3.org/TR/compositing-1/

15 **Microsoft / Supported pixel formats and alpha modes**

Straight, premultiplied, opaque, and ignore alpha definitions.

https://learn.microsoft.com/en-us/windows/win32/direct2d/supported-pixel-formats-and-alpha-modes

16 **Unity Technologies / Sprite atlas paddingPower**

Atlas padding and prevention of lower-mip bleed.

https://docs.unity3d.com/cn/6000.0/ScriptReference/Sprites.AtlasSettings-paddingPower.html

17 **Unity Technologies / Sprite texture import settings**

Alpha Is Transparency, dilation, coverage, and mip settings.

https://docs.unity3d.com/ja/2022.1/Manual/texture-type-sprite.html

18 **Microsoft / Texture block compression in Direct3D 11**

BC block sizes and format families.

https://learn.microsoft.com/en-us/windows/win32/direct3d11/texture-block-compression-in-direct3d-11

SOURCES 4 OF 4

Primary specifications and documentation

Accessed for this edition in August 2026. Product pages can change after publication.

19 **Microsoft / Block compression**

Lossy 4 by 4 block behavior and artifact tradeoffs.

https://learn.microsoft.com/uk-ua/windows/uwp/graphics-concepts/block-compression

20 **Khronos Group / KTX 2.0 Specification**

Container structure and Basis Universal transcoding context.

https://registry.khronos.org/KTX/specs/2.0/ktxspec.v2.html

21 **Epic Games / Texture format support and settings**

Texture formats, filtering, and platform settings.

https://dev.epicgames.com/documentation/en-us/unreal-engine/texture-format-support-and-settings-in-unreal-engine

22 **Khronos Group / glTF Validator source repository**

Validation scope for glTF and GLB structure and resources.

https://github.com/KhronosGroup/glTF-Validator

23 **Khronos Group / glTF Validator live**

Browser validator used in the two-minute GLB test.

https://github.khronos.org/glTF-Validator/

24 **Khronos Group / glTF Sample Viewer**

Independent reference viewer for asset comparison.

https://github.khronos.org/glTF-Sample-Viewer-Release/index.html

ABOUT THIS EDITION

How accuracy and simplicity were handled

PRIMARY SOURCES FIRST

Technical claims were checked against official specifications and vendor documentation. Engine-specific behavior is labeled as a convention to verify, not a universal law.

ONE IMAGE, ONE LESSON

The cover uses one generated material object. Every interior diagnostic figure was programmatically drawn to isolate a single comparison. There are no decorative dashboards or dense UI captures.

INFERENCE HAS LIMITS

A photo can suggest normal, height, roughness, and metallic structure, but it does not directly measure all of them. Single-image map recovery is described as an estimate wherever that distinction matters.

VERSIONED, NOT TIMELESS

This first edition reflects documentation available in August 2026. Recheck importer defaults, extension support, compression targets, and platform budgets when the target changes.

WRITTEN BY

PLAYTEX AI Editorial Team

https://www.playtex.ai

STOP GUESSING.
ISOLATE THE SIGNAL.

A practical field guide for the moment a material looks wrong and the cause is not obvious. Start with the symptom, run a two-minute test, make one correction, and leave behind a safer workflow.

- Normal, roughness, metallic, and albedo diagnostics
- Alpha, color space, compression, mips, and GLB checks
- A final preflight built for real-time delivery

PLAYTEX AI

playtex.ai