

GPUSight

Illustrated user guide

Scene diagnostics. Material and shader research. Frame timing. Saved comparisons.



A real GPUSight geometry workload map in Unity Scene View.

From a visible hotspot to a testable change

Find the scene content that deserves attention, inspect the geometry and materials behind it, and check your changes against repeatable measurements.

For Unity developers, technical artists and graphics teams. Every numbered step includes a screenshot of the real interface.

Guide edition 1.0 - September 2026

Based on the current GPUSight implementation. Package metadata: 1.0.0. Screenshots show Unity 6 with URP 17 on macOS/Metal; availability and appearance can differ with your Editor, pipeline and graphics API. Scene assets in the illustrations are demonstration content, not a statement of package contents.

NAVIGATION

Find your workflow

Read Getting started first, then follow the chapter that matches your question. Chapter names below are clickable. The PDF bookmarks also link to every page.

Getting started 3

Know what a number means (3); Open your analysis workspace (4); Choose a target profile (5); Define your own budgets (6); Refresh, freeze and resume (7).

Scene diagnostics 8

Run GPU pressure analysis (8); Interpret the overview cards (9); Read a Scene View map (10); Choose the right map (11).

Geometry and resources 12

Check LOD coverage and reduction (12); Inspect texel density and mip demand (13); Compare texture scale visually (14); Find large texture resources (15); Locate shader-heavy surfaces (16).

Object investigation 17

Break down scene geometry (17); Search and filter contributors (18); Inspect the selected object (19); Follow a material slot (20).

Material and shader research 21

Read a material's source overview (21); Understand source complexity (22); Run a controlled GPU benchmark (23); Keep a baseline and export evidence (24); Test what the material scales with (25); Compare two material configurations (26); Read the A/B timing result (27); Inspect material configuration (28); Analyze a specific shader variant (29); Follow source evidence to functions (30); Compare shader variants (31); Review retained material evidence (32).

Frame timing 33

Start a representative Play session (33); Read frame budgets and percentiles (34); Investigate spikes and budget misses (35); Preserve a timing moment (36).

Snapshots and sharing 37

Save the accepted analysis state (37); Choose a before and after (38); Read compatibility before deltas (39); Export and import snapshot JSON (40).

Reference 41

Compatibility and optional evidence (41); Troubleshooting (42); Quick reference (43).

GETTING STARTED

Know what a number means

GPUSight helps answer three related questions: **where to investigate**, **what an object or material contains**, and **whether a change helped**. The source and scope of a result matter as much as its value.

Evidence	What it tells you
Estimated	A model or proxy, such as geometry workload, source complexity, expected LOD or texel density. Useful for prioritizing work.
Measured	A timing sample from an available backend, with a specific device and test context. A preview benchmark measures that preview.
Observed	Configuration or resource facts exposed by the current environment, such as texture bindings, passes or native allocations.
Unavailable	The tool cannot provide a trustworthy value in this context. It does not mean zero work or zero memory.

A useful first session

Scene investigation: choose a profile, obtain fresh data, run GPU pressure analysis, then follow a card or map into Objects and Deep Introspection.

Material investigation: open Research Material, inspect source and configuration, run a controlled benchmark, and compare one alternative under the same conditions.

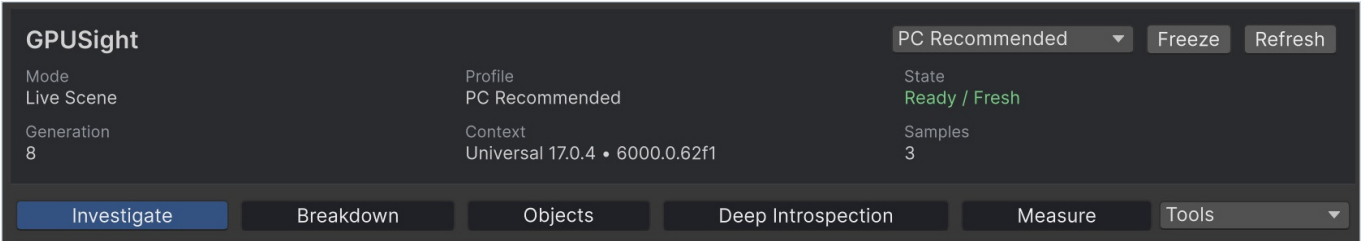
Validation: repeat a representative Play Mode scenario, inspect frame percentiles and spikes, then save and compare snapshots with matching context.

Keep the scopes separate. A renderer's estimated geometry share, a material's preview GPU time and a frame's GPU time answer different questions. They cannot be substituted for one another.

Illustrations are examples from real sessions. A number in a screenshot is not a performance target, a promised improvement or a prediction for your device.

Open your analysis workspace

After importing GPUSight and letting Unity finish compilation, open **Window > Analysis > GPUSight**. Keep Scene View beside the GPUSight Analysis window and open a representative scene in Edit Mode.



The GPUSight header and primary navigation. Tools contains Research Material, Profiles and Snapshots.

Location	Use it for
Investigate	GPU pressure overview and links to relevant objects or maps.
Breakdown	Geometry grouped by scene characteristics.
Objects	Search, filters, grouping and ranked contributors.
Deep Introspection	Details of the contributor selected inside GPUSight.
Measure	Play Mode timing, percentiles, frame history and spikes.
Tools	Material/shader research, target profiles and snapshots.

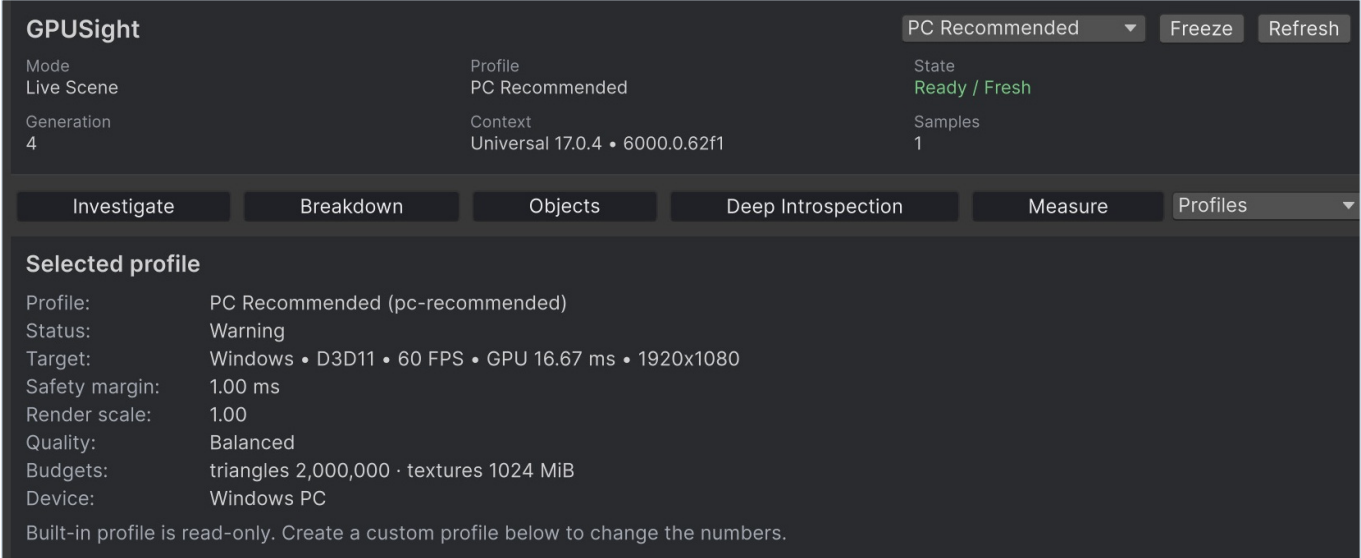
Before the first capture

Use a scene and viewpoint that represent gameplay. A distant overview and a close-up can produce very different visibility and texel results. Save the scene normally when you need a stable scene identity for future comparisons.

The Asset Store DLL delivery uses **Assets/GPUSight**. A source UPM delivery uses **Packages/com.gpusight**. Keep one delivery of GPUSight in a project to avoid duplicate assemblies. The current release workflow covers Unity 2022.3 with URP 14 and Unity 6.0 with URP 17; the compatibility reference at the end gives the exact verified combinations and feature limits.

Choose a target profile

Use the profile selector at the top of GPUSight. Start with the closest built-in target, then open **Tools > Profiles** to inspect its budgets and target conditions. The profile selection triggers a context check.



PC Recommended selected. The Profiles panel shows its target, safety margin and scene budgets.

Built-in target	FPS	Triangles	Textures
PC Minimum	30	750,000	512 MiB
PC Recommended	60	2,000,000	1,024 MiB
PC High End	120	4,000,000	2,048 MiB
Android Low / Mid / High	30 / 60 / 90	120k / 300k / 900k	128 / 256 / 512 MiB
iPhone Low / Target / High	30 / 60 / 120	150k / 400k / 800k	192 / 384 / 512 MiB

These values are starting budgets. They are not measured limits of all devices in a category. Pick budgets based on your project's visual goals and measurements on its target hardware.

A profile describes your target. It does not switch the Unity Build Target, change the active graphics API, modify quality settings or emulate a phone. A Windows/D3D11 profile can therefore report a warning in a Metal Editor, as in this example.

Define your own budgets

In **Tools > Profiles**, fill out **New profile** and press **Create profile**. Use a distinct Profile ID and a recognizable Display name. Select the created profile in the header and review its status and target summary.

New profile

Profile ID

custom-profile

Display name

Custom profile

Platform

Windows

Graphics API

D3D11

Target FPS

60.00

GPU budget (ms)

16.67

Safety margin (ms)

1.00

Render scale

1.00

Quality profile

Balanced

Device reference

Custom device

Visible triangles budget

Draw calls budget

Texture budget (MiB)

Use resolution

☒

Resolution width

1920

Resolution height

1080

Create profile

The New profile form. Built-in profiles remain read-only; a custom profile stores your own analysis targets.

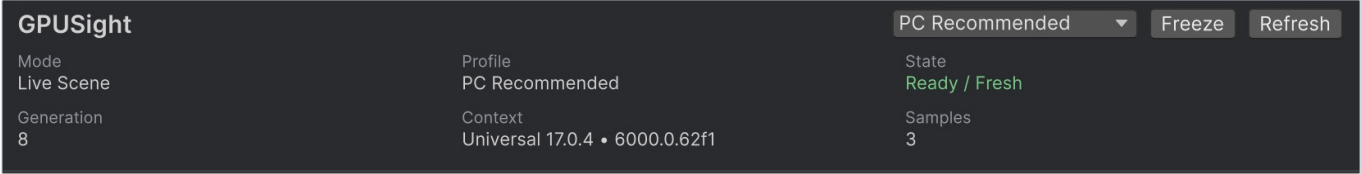
Setting	How to choose it
Platform / Graphics API	The environment you intend to ship and validate on.
Target FPS	The desired frame rate. The frame budget is 1,000 / FPS milliseconds.
GPU budget / Safety margin	Your GPU time allowance and reserve. Leave room for variation and other work.
Visible triangles / Draw calls	Optional scene budgets. An omitted budget is unspecified, not a zero allowance.
Texture budget	The allocation budget, in MiB, for the resource scope shown by the tool.
Resolution / Render scale	Target conditions recorded with the profile. They do not resize the current Game View.
Quality / Device reference	Labels that identify the intended quality tier and device in your workflow.

At 60 FPS the whole-frame interval is about **16.67 ms**; at 30 FPS it is **33.33 ms**. A custom GPU budget can be lower than that interval. CPU and GPU work can overlap, so adding their displayed times is not generally a frame-time calculation.

Profiles are stored locally for this project. The current panel supports creating custom profiles; it does not present a full edit/delete workflow. If the panel offers **Complete setup** or **Acknowledge recovery**, read the accompanying status before accepting it.

Refresh, freeze and resume

Press **Refresh** after changing the scene or analysis context, then check **State** and **Context**. Use **Freeze** to hold the accepted GPUSight data while inspecting it. Press **Resume** to accept live updates again.



Ready / Fresh identifies accepted data for the displayed context. Freeze and Refresh are in the same header.

State or label	What to do with it
Ready / Fresh	Read the result together with its mode, profile, source and scope.
Updating / Pending	Let the active operation finish before using the new result.
Stale	Treat the previous value as historical until it is refreshed.
Mismatch	Reconcile the profile or context before treating results as comparable.
Frozen	You are inspecting an accepted state held by GPUSight.
Partial	Read the omissions or limits; the result covers only part of the requested scope.
Unavailable	Read the reason. The requested signal is missing or unsupported.

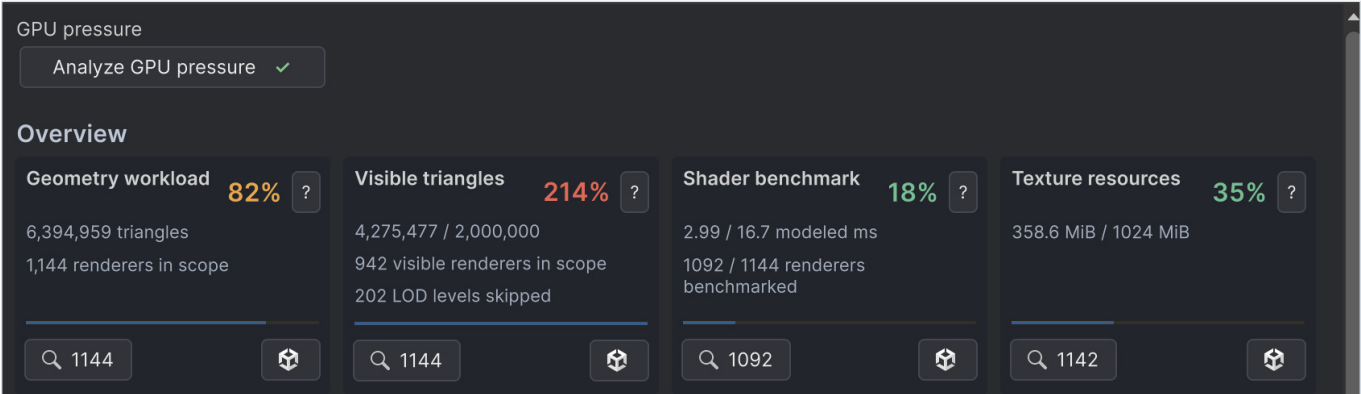
What Freeze does

Freeze holds GPUSight's accepted analysis state. It does not pause Unity's simulation, write a snapshot file or turn an estimate into a measurement. Use **Save snapshot** for durable storage, and the Measure history controls for a particular timing moment.

Generation and Samples describe the accepted data stream. More samples do not automatically make a result valid for a different scene, device or graphics API. When changing conditions, obtain a fresh result and preserve the new context with it.

Run GPU pressure analysis

Open **Investigate** and press **Analyze GPU pressure**. Allow the analysis stages to complete. Read each card's amount, budget and coverage before deciding which area needs attention.



The overview combines geometry, visible triangles, controlled shader tests and texture resources. Each card retains its own scope.

The operation coordinates scene diagnostics, resource information, material benchmarks, shader source analysis and texel analysis as available. It can create additional CPU and GPU work and take longer in large scenes. Treat its active measurement period separately from normal gameplay timing.

The **magnifying-glass/count** action opens the relevant object set. The **cube/map** action switches to the related Scene View visualization. A **?** button explains a card's metric.

Choose a useful lead

In this illustration, Visible triangles is above its configured budget. That is a reason to inspect visibility and LODs. The Shader benchmark card is below its budget, but its value is a model based on controlled tests; it does not prove that the scene is GPU-bound by geometry.

GPU pressure is not GPU utilization. The overall pressure signal uses the strongest scored domain. The cards are not independent pieces of a pie and their percentages should not be added.

SCENE DIAGNOSTICS

Interpret the overview cards

Card	Meaning and practical boundary
Geometry workload	A relative geometry proxy based on the accepted renderer scope. The triangle total and contributor count describe that scope; the score is not GPU milliseconds.
Visible triangles	Frustum-visible topology with estimated inactive LOD levels excluded, compared with the profile's triangle budget. It is not the exact total submitted across every pass.
Shader benchmark	A modeled contribution built from controlled material tests and screen share, compared with the GPU budget. It is not measured frame GPU time.
Texture resources	Captured allocations of resources associated with the visible material scope, compared with the texture budget. It does not represent all process or GPU memory.
Shader complexity	P50, P90 and maximum source-complexity indices for the covered visible materials. These are source-analysis values, not times or compiled instruction counts.
Captured frame GPU time	Appears only when suitable verified capture data is available. Its capture context determines how it can be interpreted.

Read coverage before comparing

Counts such as **1,092 / 1,144 renderers benchmarked** are part of the result. Missing contributors can change a distribution or hide relevant work. A partial analysis should not be presented as a complete measurement of the scene.

Shared textures are deduplicated for the resource total. Adding all per-object or per-material memory labels can count the same texture repeatedly.

Different camera positions can change visible triangles, coverage, texture scope and texel demand. To check a change, repeat the same view and record the same target profile and context.

A useful report states its scope. For example: "Visible triangle estimate exceeded this profile's budget in the courtyard view." Avoid replacing that statement with "the GPU is 214% utilized."

Read a Scene View map

In Edit Mode, use the **Map** dropdown in the **GPUSight Diagnostics** overlay. Begin with **Geometry workload**. Keep the legend visible, inspect highlighted objects and relate them to the Objects list.



Geometry workload with its legend and ranked markers. Warm colors identify larger estimated contributors within this map's scale.

Map colors are meaningful only with their legend. A red geometry hotspot, an LOD coverage color and a texel outlier describe different things. A map with a **Run** button needs that explicit operation before its measured or captured information is available.

Scene View maps use a diagnostic rendering path and do not change the saved materials. They run in Edit Mode; they are disabled during Play Mode. Return to Measure for gameplay timing.

The map is a diagnostic view. Replacement rendering does not reproduce every shader's alpha clipping, vertex displacement, property overrides or custom rendering state. Use it to locate a question, then inspect the original object and shader.

SCENE DIAGNOSTICS

Choose the right map

Map family	What to look for
Geometry workload	Relative geometry contribution and candidates for mesh or LOD investigation.
Coverage	How much of the diagnostic view is covered. This is a view-dependent visibility signal.
Overdraw	Layers of front-facing geometry per covered diagnostic pixel. Useful for locating overlap.
Transparent / Opaque overdraw	Overlap within each classified surface group. Each group has its own covered-pixel denominator.
Transparency	Surface classification. Unknown classification must not be read as definitely opaque.
Configuration maps	Available passes describes exposed passes; Shadow casting describes renderer settings; Shader features groups shader/material characteristics. These are configuration facts, not measured executed work.
Shader benchmark	Controlled preview timings attached to materials. Run the tests before reading the timing map.
Texture resources	Captured resource sizes associated with visible material slots.
Texture sampling / Fragment math	Source-analysis signals for texture and arithmetic work.
Loops & branches / Vertex processing	Source-analysis signals for control flow and vertex work.
Pixel work / Shader complexity	Combined source-derived estimates, useful for relative investigation.

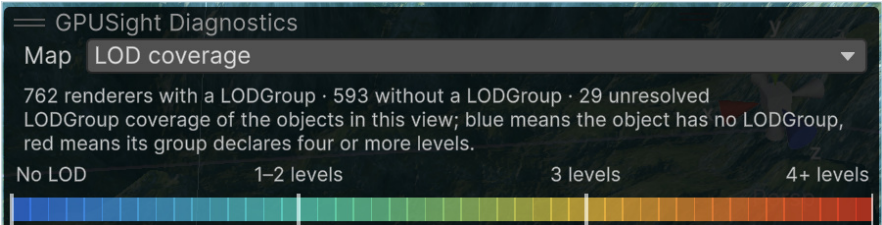
Overdraw is a proxy

The diagnostic overdraw path counts geometry layers without depth rejection. It is not a count of actual fragment-shader executions after all early-depth, alpha-clip and pipeline behavior. Opaque and transparent averages should not be added together.

Updates are bounded: approximately 5 Hz, a diagnostic target around 320 pixels on its longer axis, and limits of 4,096 renderers and 16,384 submesh draws. Omitted content can make coverage partial. Fine details and very large scenes therefore need additional inspection.

Check LOD coverage and reduction

Choose **LOD coverage** in the Scene View overlay. Look for large, detailed objects without an LODGroup. Use **Breakdown > LOD membership** to connect the visual result with contributor counts and triangle totals.



The LOD coverage legend. Blue means no LODGroup; warmer colors indicate more declared levels.

Geometry Breakdown				
Breakdown	LOD membership			
LOD membership	Objects	Triangles	Materials	Geometry share
▶ No LOD group	624	3,951,563	696	53.6%
▶ LOD 0	464	1,654,129	477	29.1%
▶ LOD 1	112	1,047,756	112	12.2%
▶ LOD 2	101	201,664	101	5%

LOD membership connects the visual finding with objects, triangle counts and geometry share.

LOD map	How to interpret it
LOD coverage	Whether an object belongs to an LODGroup and how many levels the group declares. Warm is not automatically bad.
LOD level	An expected level based on view, bounds, thresholds and LOD bias. It is not an observation of the exact rendered crossfade state.
LOD reduction	Triangle reduction to the next level. Weak reduction below about 10% deserves inspection; negative reduction means the next level has more geometry.

A culled LOD and an unknown LOD are different states. Do not treat unresolved information as proof that the object is already cheap.

Turn the finding into a change

For a high-detail object visible at long distances, inspect its LODGroup, transitions and next-level mesh in Unity. After an intentional asset edit, repeat the same camera view and compare both visual quality and Play Mode timing. Adding an LODGroup alone does not demonstrate a frame-time improvement.

Inspect texel density and mip demand

Select a relevant contributor in **Objects**, open **Deep Introspection**, and inspect the Geometry card's texel fields. The **Texel density**, **Mip demand** and **Texel outliers** maps help locate unusual areas in the current view.

Geometry	
Vertices	56,229
Submeshes	3
LOD group	No
Triangles	64,721
Geometry share:	0.6%
Texel density P50:	25.77 texels per pixel
Texel density P90:	317.92 texels per pixel
Ideal mip P50:	4.69
Texel source:	_MainTex 4096×4096 · _MainTex 4096×4096 · _MainTex 2048×2048
Texel coverage:	10,784 / 64,721 triangles · sampled
Streaming mips:	_MainTex: Streaming mips: — (Texture streaming is disabled in the current quality level.)

The selected object's Geometry card reports texel density, ideal mip, texture sources, sampled triangle coverage and streaming information.

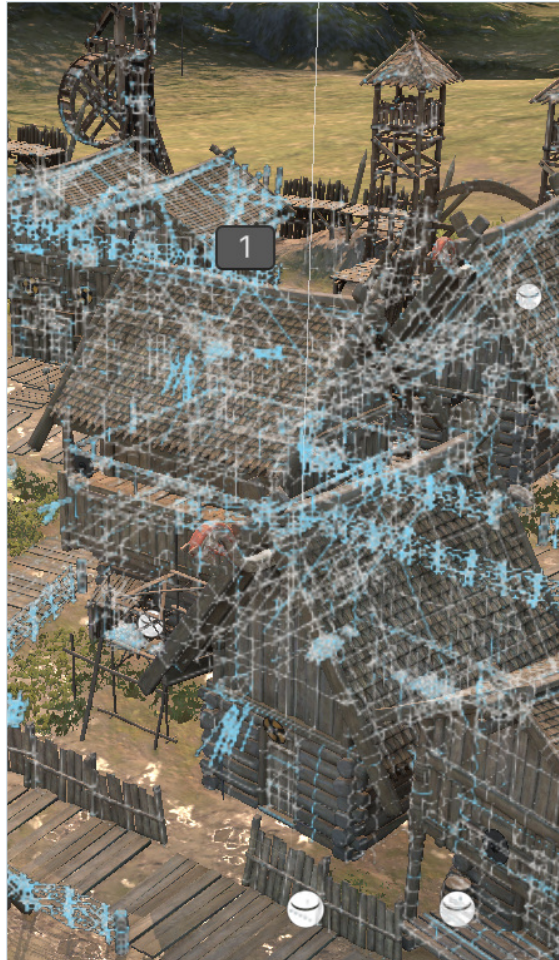
Texel density is a **linear texels-per-pixel** estimate derived from texture/UV area and projected screen area. An ideal mip of about 2 corresponds to about 4 texels per pixel in each dimension. Negative ideal mip values indicate magnification.

P50 and P90 summarize sampled geometry. Texel outliers highlight values more than about 1.5 mip levels from the view median. This is a relative diagnostic, not a universal art-quality rule.

Check the evidence beneath the value. Numeric analysis depends on readable mesh/UV data and supported primary texture bindings. Sampling limits make the percentiles approximate. Enabling Read/Write on every mesh can increase memory use; inspect the specific asset and trade-off first.

Compare texture scale visually

Choose **Texel grid** in the map selector and inspect neighboring surfaces from the intended gameplay distance. The grid helps reveal mismatched texture scale and heavily compressed screen-space detail.



A real Texel grid overlay on roof surfaces. Grid spacing reveals differences in texture scale across the view.

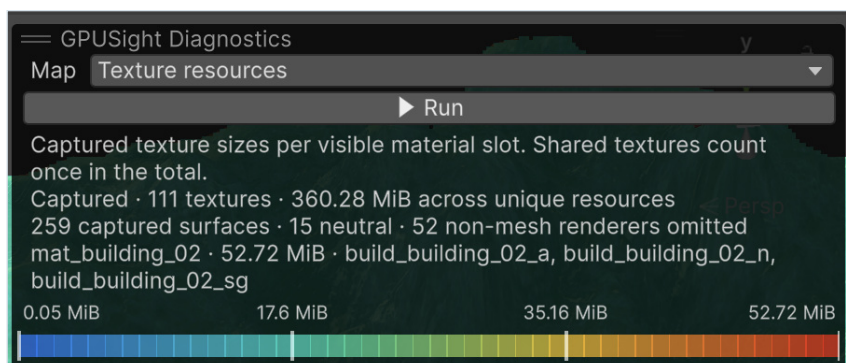
The visual grid uses texture coordinates: regular lines mark 256-textel intervals and major lines mark 1,024-textel intervals. It can remain visible even when CPU texel statistics are unavailable, because the visual and numeric paths have different data requirements.

The selected primary texture and its tiling affect the result. It does not reproduce every texture layer or every possible custom UV operation in a shader.

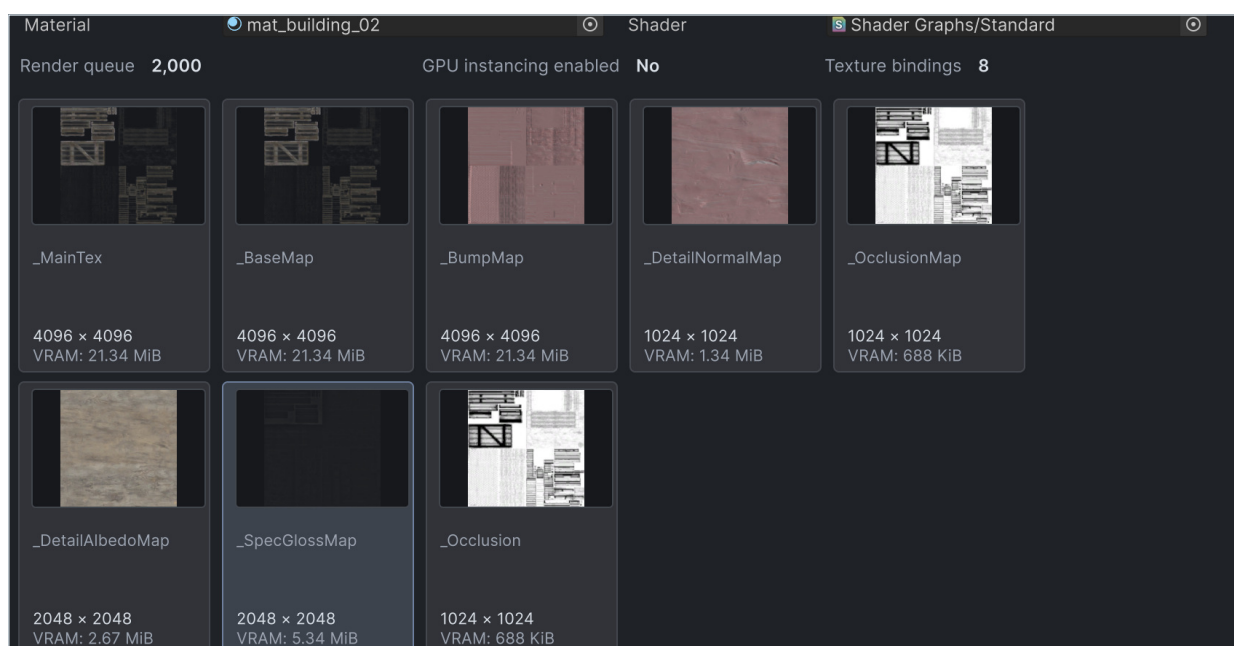
Use this view together with Texel density and Mip demand. A dense grid can suggest a texture-size or tiling review; the final decision still depends on close-up quality, mip behavior, memory and timing in your project.

Find large texture resources

Choose **Texture resources** and press **Run** when offered. Read the capture status and unique-resource total in the overlay. Follow a highlighted object into **Deep Introspection** and expand a material slot to inspect its texture bindings.



The resource overlay reports captured textures, unique allocation size and omitted surfaces.



A material's texture bindings show dimensions and captured allocation sizes. One texture can appear in more than one property.

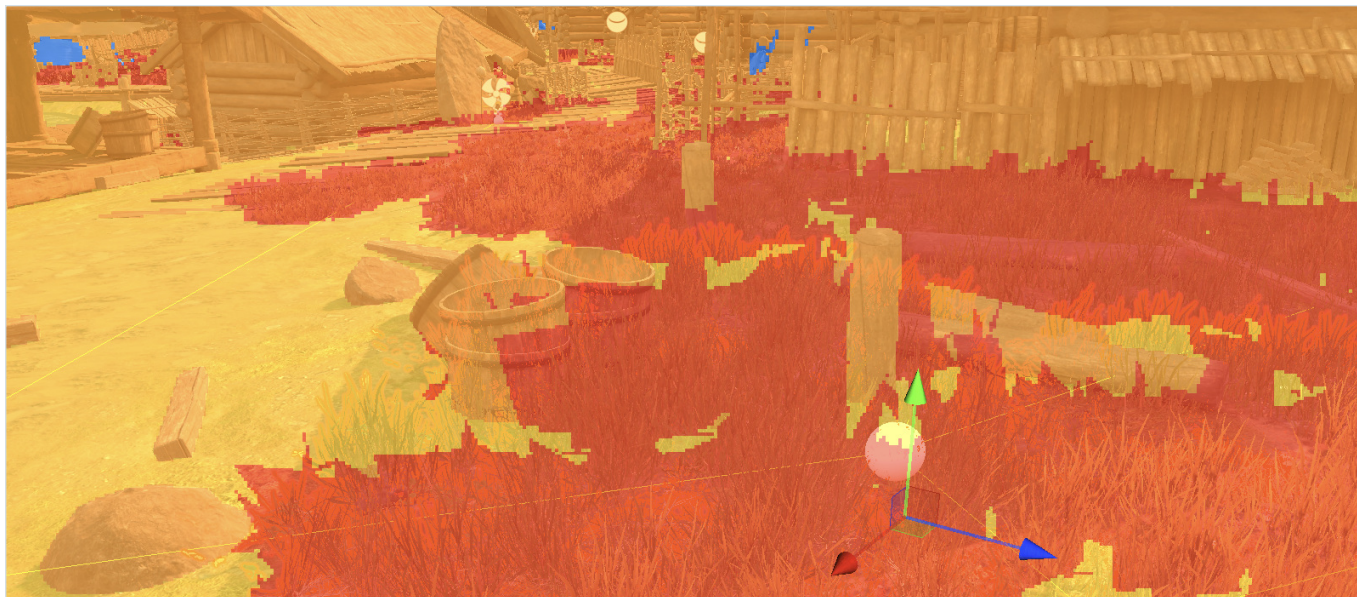
The native allocation inventory currently depends on Metal. An unsupported backend can leave allocation values unavailable even when ordinary material properties are readable.

Do not sum every binding's VRAM label: shared textures count once in the unique-resource total. Captured allocation size is different from the asset's compressed file size and from total process memory.

Useful candidates include oversized textures on small surfaces, unnecessarily expensive formats and duplicate resources. Review platform import settings and visible quality before changing them, then recapture the same scope.

Locate shader-heavy surfaces

Choose **Shader complexity** or a source-focused map such as **Texture sampling** or **Fragment math**. Run the source analysis if requested and allow its progress to finish. Follow a highlighted material into Research Material.



A real source-complexity map highlights differing material workloads. The color is an analysis signal, not elapsed GPU time.

Source maps analyze the relevant material configuration and supported passes. Their aggregate represents the strongest analyzed pass rather than the sum of all draws executed by the scene. Missing or unresolved source can make a result partial.

Shader benchmark is a separate map. It uses controlled preview timing from the available backend. A material's test time remains the time for that preview, even when its color is drawn over an object in the scene.

Combine the signals

A surface with a large source-complexity index and substantial screen coverage is a useful candidate for investigation. Open the material, identify the source of complexity, and compare a deliberate alternative under the same benchmark conditions. Finish with a representative frame-time check.

A source index and a benchmark in milliseconds cannot be placed on a common scale. Use each to answer its own question.

Break down scene geometry

Open **Breakdown** and choose a grouping in its dropdown. Start with **Surface** to compare opaque, alpha-clipped and other classifications, or **LOD membership** when investigating geometric detail at distance.

Investigate	Breakdown	Objects	Deep Introspection	Measure	Tools
Geometry Breakdown					
Breakdown Surface					
Surface		Objects	Triangles	Materials	Geometry share
▶ Opaque		377	2,402,301	400	98.7%
▶ Alpha clip		9	2,043	9	1.1%
▶ Other		2	744	2	0.2%

The Geometry Breakdown table groups contributors and shows object counts, triangles, materials and geometry share.

Expand a group to examine its contributors. Use the totals to decide whether a broad category or a small number of individual objects deserves attention.

Keep the denominator in mind

Geometry share is an estimated contribution relative to the accepted scene scope. It is not a measured portion of frame GPU time. Filtering the list does not make the surviving rows add up to 100% of a new scene.

Some groupings are many-to-many. A renderer can participate in several material, pass or camera groups; adding those group totals can count the renderer more than once. Use a renderer grouping when you need a direct list of individual contributors.

If a grouping includes Unknown or unavailable data, investigate the reason before drawing a conclusion about that category. The absence of a classification does not establish an inexpensive rendering path.

Search and filter contributors

Open **Objects**. Use **Search**, choose **Group by** and **Sort by**, then open **Filters** for a specific category or property. Select a matching value and inspect the resulting rows.

Objects

Search

Group by

Renderer

Sort by

Geometry workload

Filters

Filter

Category

Value

Choose value

Object	Triangles	Materials	Geometry share	Test GPU, ms
build_bighouse_01 build_bighouse_01 · Opaque · Buildings / pf_build_bighouse_01	64,721	3	1.7%	0.029–2.059
build_blacksmith_01_walls_02 build_blacksmith_01_walls_02 · Opaque · Buildings / pf_build_blacksm...	62,694	2	1.6%	0.029–2.059
pf_build_barracks_single_01 build_barracks_single_01 · Opaque · Buildings / Buildings	51,769	2	1.3%	0.029–2.059
build_blacksmith_01_walls_01 build_blacksmith_01_walls_01 · Opaque · Buildings / pf_build_blacksm...	49,906	1	1.3%	2.059
build_big_storage_01_pillars_01 build_big_storage_01_pillars_01 · Opaque · Buildings / pf_build_big_s...	45,544	2	1.2%	0.057–2.059
build_big_storage_01_straw_ro... build_big_storage_01_straw_roof_002 · Opaque · Buildings / pf_build...	32,516	4	1.0%	0.029–2.255
build_small_house_tall_roof_01 build_small_house_tall_roof_01 · Opaque · Buildings / pf_build_small...	33,035	2	0.9%	2.059–2.100
build_small_house_tall_roof_01	33,035	2	0.9%	0.029–2.059

Objects with Filters expanded. Geometry share and Test GPU, ms retain different evidence scopes.

Group by Renderer, Category, Material, Shader, Pass, Camera, Scene, Layer, Tag or Prefab depending on your question. Available sorts use trustworthy per-object values; a view-level map is not automatically a valid object sort.

Active filter chips show the current restrictions. Remove a chip to remove that restriction, use **Clear** when available, and clear Search separately if its text still narrows the result.

Rows with the same display name can be different assets or objects. Use the secondary path/context and Deep Introspection to identify the one you intend to change.

Test GPU, ms is a controlled material-test signal. A range can reflect different associated materials. It does not assign a measured slice of the original frame to that renderer.

Inspect the selected object

Select an individual contributor inside GPUSight and open **Deep Introspection**. Confirm its hierarchy and mesh identity, then review the Geometry, Rendering, Shader, Surface and Shadows cards.

Deep Introspection

build_bighouse_01

Hierarchy: Content/Buildings/Buildings/pf_build_bighouse_01/build_bighouse_01

Mesh: build_bighouse_01

Select in Scene

Open Inspector

Geometry

Vertices56,229

Submeshes3

LOD groupNo

Triangles64,721

Geometry share: 0.6%

Texel density P50: 25.77 texels per pixel

Texel density P90: 317.92 texels per pixel

Ideal mip P50: 4.69

Texel source: _MainTex 4096×4096 · _MainTex 4096×4096 · _MainTex 2048×2048

Texel coverage: 10,784 / 64,721 triangles · sampled

Streaming mips: _MainTex: Streaming mips: — (Texture streaming is disabled in the current quality level.)

Rendering

Motion vectorsObject

Reflection probesBlend Probes

Light probesBlend Probes

Baked lightmapYes

In static batchNo

Property overridesNo

Shader

Enabled shader passes27

Available shader passes30

Active shader keywords9

Assigned textures22

Surface

Transparent material slots0

Shadows

Shadow castingOn

Receives shadowsYes

Scene context

Layer: 0: Default · Tag: Untagged

Prefab: pf_build_bighouse_01

Cameras0

Deep Introspection for a building. Geometry, renderer settings and shader facts are shown together with scene context.

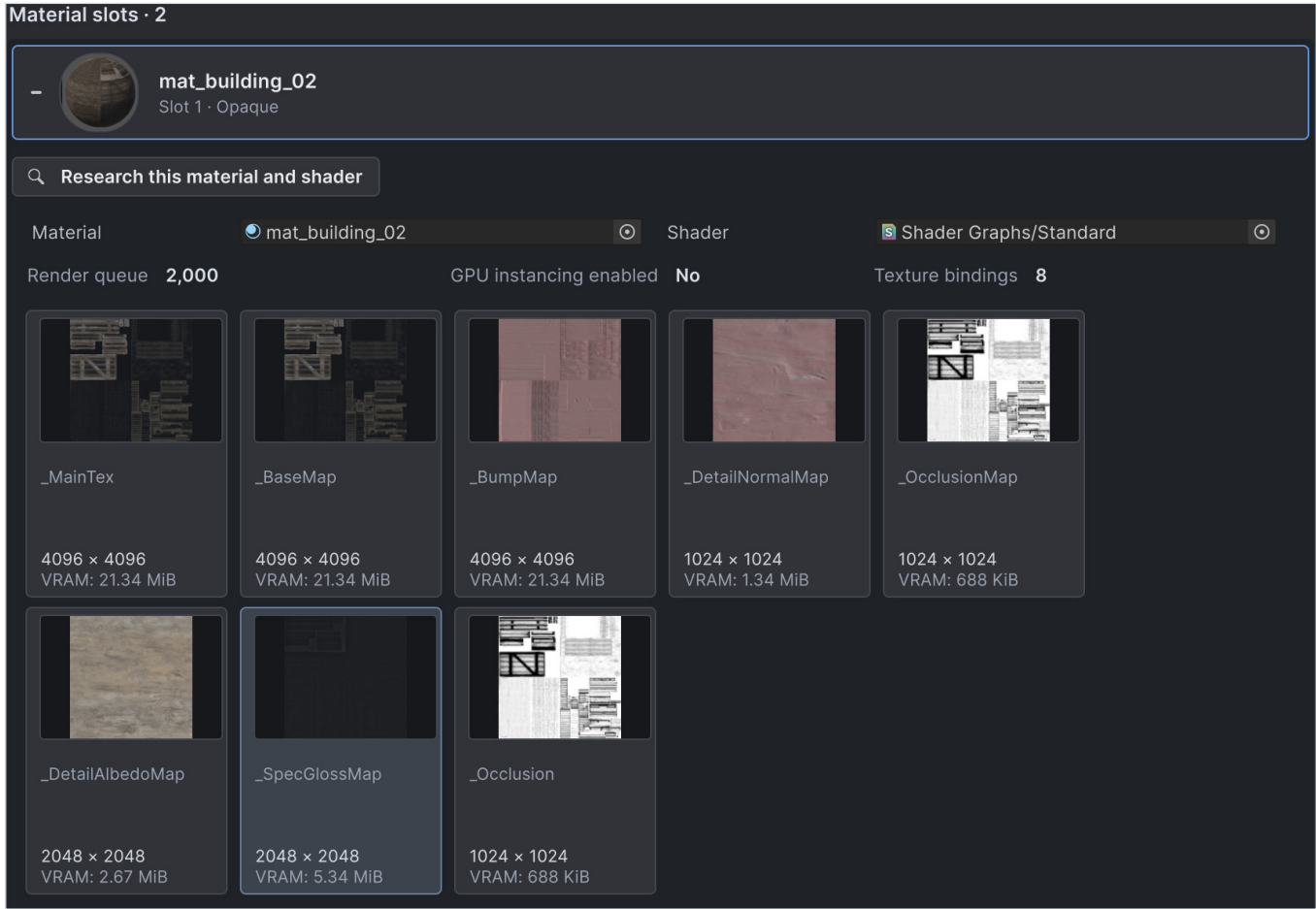
Select in Scene connects the contributor to Unity's scene selection. **Open Inspector** opens its editable Unity Inspector. GPUSight's facts help you choose what to investigate; any asset or renderer edits still need to be made deliberately in Unity.

Triangles, vertices and submeshes describe geometry. Pass counts and keywords describe exposed shader configuration; enabled/available passes are not a list of draws proven to have run in the frame.

If the panel says **Select an object**, return to an individual contributor row in Objects. A grouped row or an unrelated Project selection may not provide the renderer context this panel needs.

Follow a material slot

Scroll to **Material slots** in Deep Introspection and expand the relevant slot. Inspect its material, shader and textures, then choose **Research this material and shader**.



An expanded material slot exposes the Research action, asset references, render queue, instancing flag and texture bindings.

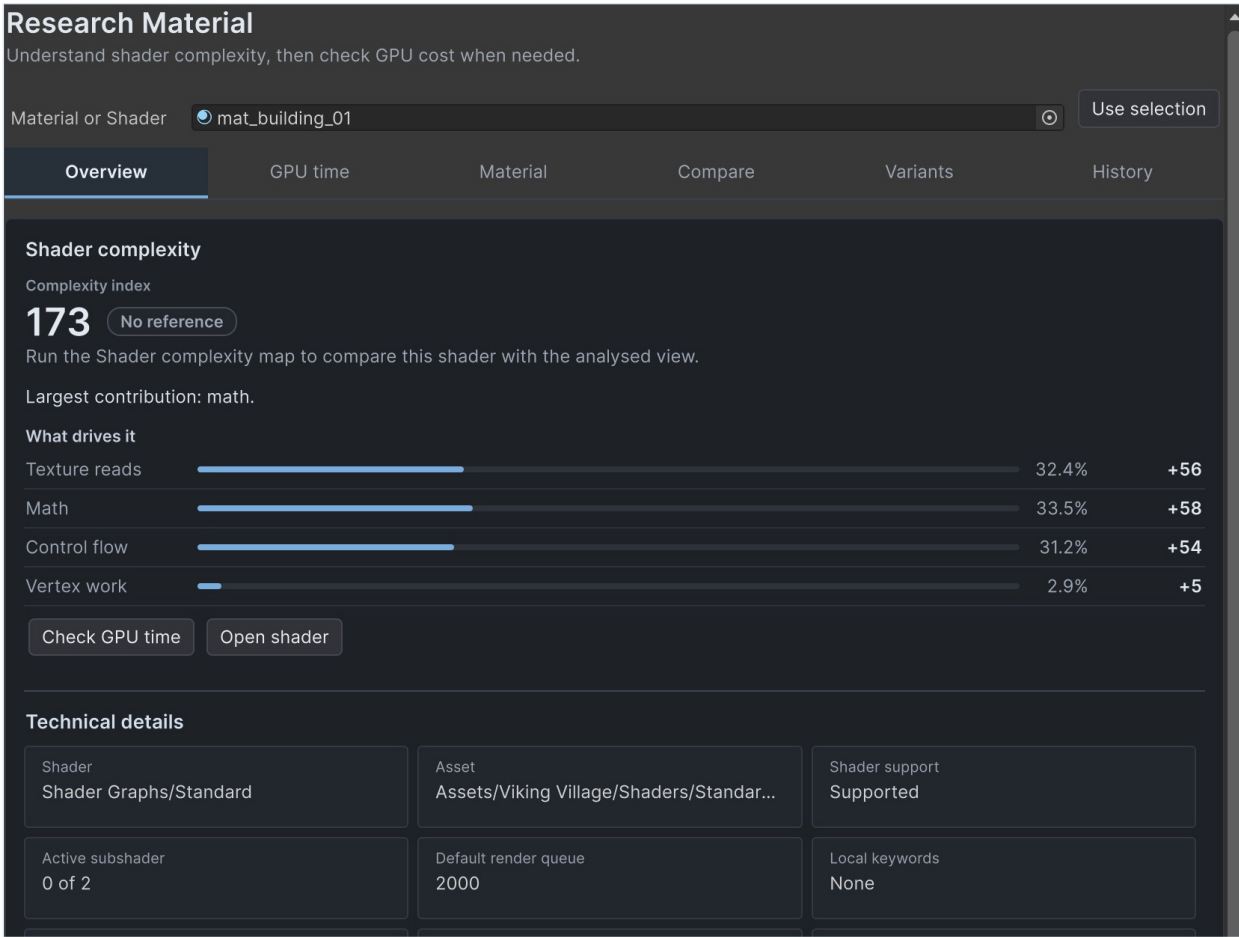
A renderer can contain several material slots. Check the slot's surface classification and binding before deciding which material to investigate. A shared material can affect many objects, so use an intentional duplicate or controlled alternative when experimenting.

Texture thumbnails and dimensions help identify what the material actually references. The same image may be bound under multiple property names, and native allocation values can be unavailable on unsupported backends.

The Research action carries the selected material into a focused workflow. You can also open **Tools > Research Material** and select a material directly without first choosing a scene object.

Read a material's source overview

Open **Tools > Research Material** and assign a Material or Shader in the picker. Alternatively, select the asset in the Project window and press **Use selection**. Read **Overview** after source analysis completes.



Overview separates the source-complexity index, work categories and reference status for the selected material.

Overview expands supported includes and analyzes the applicable source configuration. A selected Material supplies its valid keywords and material context. A Shader asset is useful for source inspection but does not provide a configured material for a GPU preview test.

No reference means there is no suitable scene distribution for a percentile comparison. It does not mean the shader has no complexity. Run the relevant scene shader analysis to obtain an applicable reference; percentile reporting needs enough comparable material samples.

Check GPU time navigates to the test panel. Selecting an asset or opening that tab does not start a benchmark; the measurement begins when you press the run button.

MATERIAL AND SHADER RESEARCH

Understand source complexity

Signal	Interpretation
Texture sampling	Weighted source evidence for texture accesses, including recognized local repetition. It is not measured texture bandwidth.
Fragment math	Source operations and recognized intrinsics contributing to fragment work. Compiler optimization and hardware execution can differ.
Loops and branches	Recognized control-flow structure. Runtime iteration counts and branch behavior may remain unknown.
Vertex processing	Source-derived work associated with the vertex stage. It does not measure the scene's vertex GPU time.
Index	A weighted summary used for comparison within a compatible analysis context. It is not an instruction count or time unit.

The current source index weights texture sampling by 8, combines math with control-flow weight, and includes vertex work at a lower weight. Local loop expansion is bounded. This makes expensive-looking source structure easier to find without claiming to reproduce the compiler or GPU.

Compare like with like

The selected API, build target, pass, stage and keywords can change the preprocessed source. The overview of a material and a single selected variant/stage are different scopes and can produce different indices.

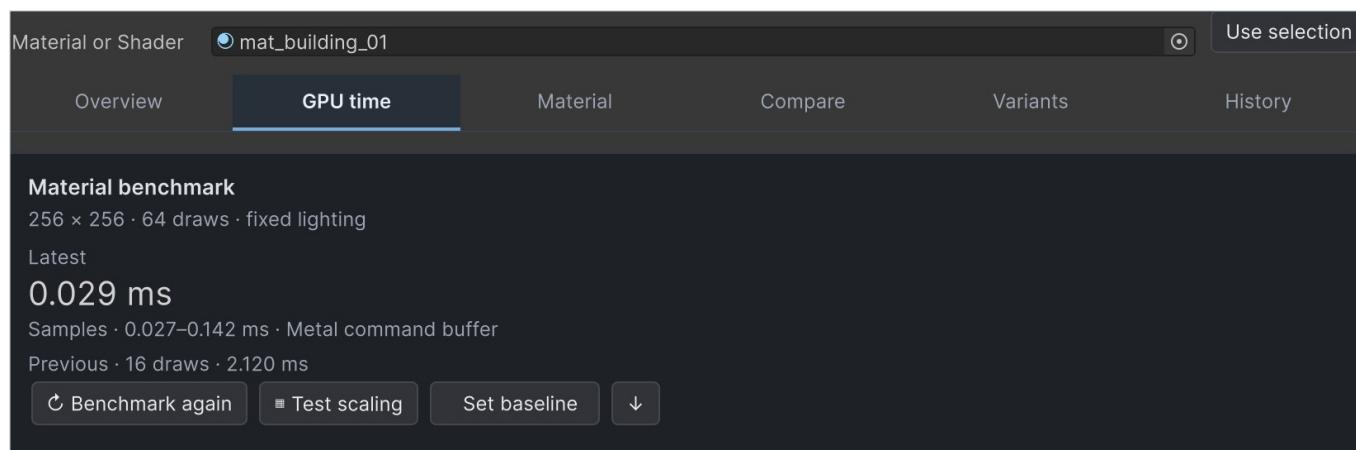
P50 is the median of an eligible reference set; P90 describes its upper end. A maximum describes an extreme within that set. These values change with the materials included and require a compatible analysis version and configuration.

Partial source coverage, unresolved includes or unsupported constructs reduce what you can conclude. Read the status and technical details. A low index on a partial result does not establish that the complete shader is cheap.

Use source analysis to form a hypothesis: for example, "this fragment variant performs additional texture reads." Then verify the practical cost with a controlled GPU test and the real scene.

Run a controlled GPU benchmark

With a Material selected, open **GPU time** and press **Benchmark** or **Benchmark again**. Wait for the result; use **Cancel** if you need to stop. Read the sample range and backend together with the median.



The GPU time panel shows a retained example result and its test controls. Your current device and run determine your own values.

The standard preview uses **256 x 256 pixels, 64 draws and fixed lighting**. It requires active URP, a supported material and an available timing backend. Inputs such as a live RenderTexture can prevent a repeatable test and are rejected where unsupported.

The reported GPU work belongs to the controlled preview. Camera setup, clears and pipeline work are part of its context. The result is not the material's isolated cost on every scene object and is not the original scene's frame time.

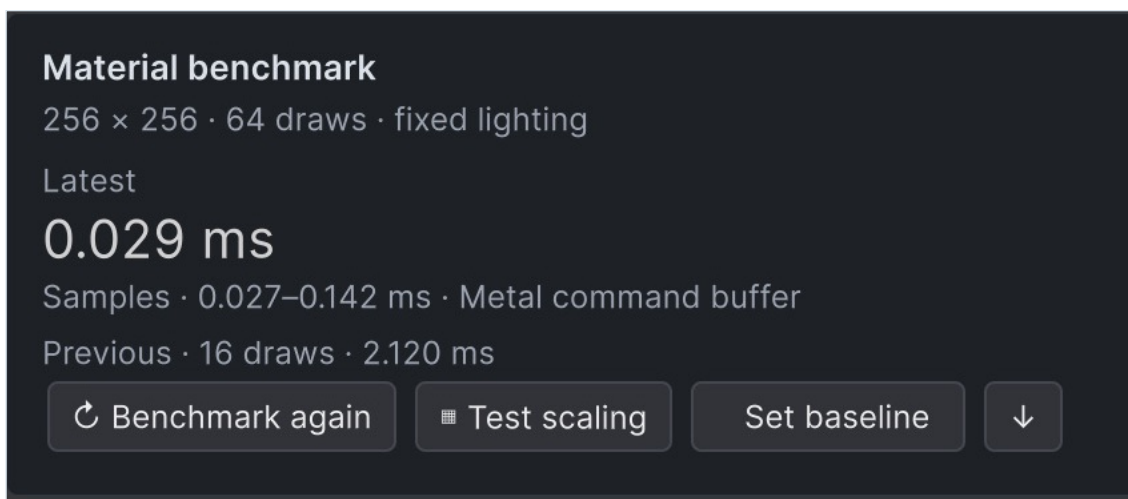
Make repeated runs useful

Keep the device, backend, resolution, draw count and material configuration comparable. Let asset import and unrelated Editor activity settle. A wide sample range is a reason to repeat and investigate variability, not to select the fastest sample as the result.

A target profile for Android or iPhone does not make this test run on that device. The benchmark's actual backend/device is authoritative.

Keep a baseline and export evidence

After a completed test, press **Set baseline** to preserve that result as the material's local reference. After an intentional change, run the benchmark again. Use the **down-arrow export** button to save the benchmark report as JSON.



The result card includes Benchmark again, Test scaling, Set baseline and the down-arrow JSON export control.

The baseline keeps its result and test conditions locally in the project's **Library**. **Replace baseline** replaces that reference when one already exists; the remove-baseline control clears it for the material.

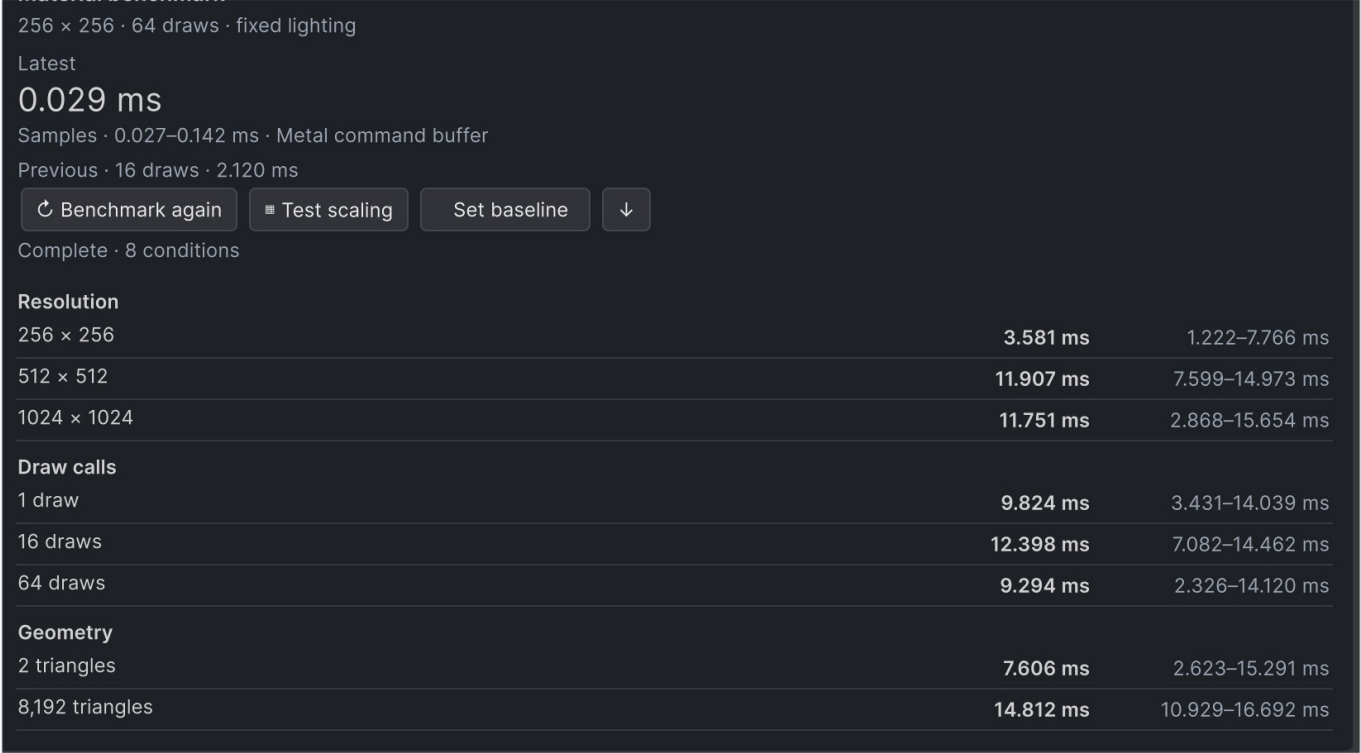
When current and baseline conditions are compatible, the displayed difference is **current median minus baseline median**. A negative time difference means the current preview run was faster under those conditions. When conditions differ, the panel reports **Different test conditions** instead of a trustworthy improvement.

The JSON report preserves conditions, samples and technical error information. Use it when sharing a result or investigating why a test could not run. Keep exports outside disposable Library data when you need a durable record.

This baseline is separate from **Snapshots**, which preserve a broader accepted analysis state. Use the material baseline for repeated tests of one material; use Snapshots for saved scene/context comparisons.

Test what the material scales with

On **GPU time**, press **Test scaling**. Wait for **Complete - 8 conditions** and inspect the Resolution, Draw calls and Geometry groups. A cancelled or invalidated run does not publish a complete set.



A completed scaling run. Each row includes a median and sample range; the values here show considerable run variability.

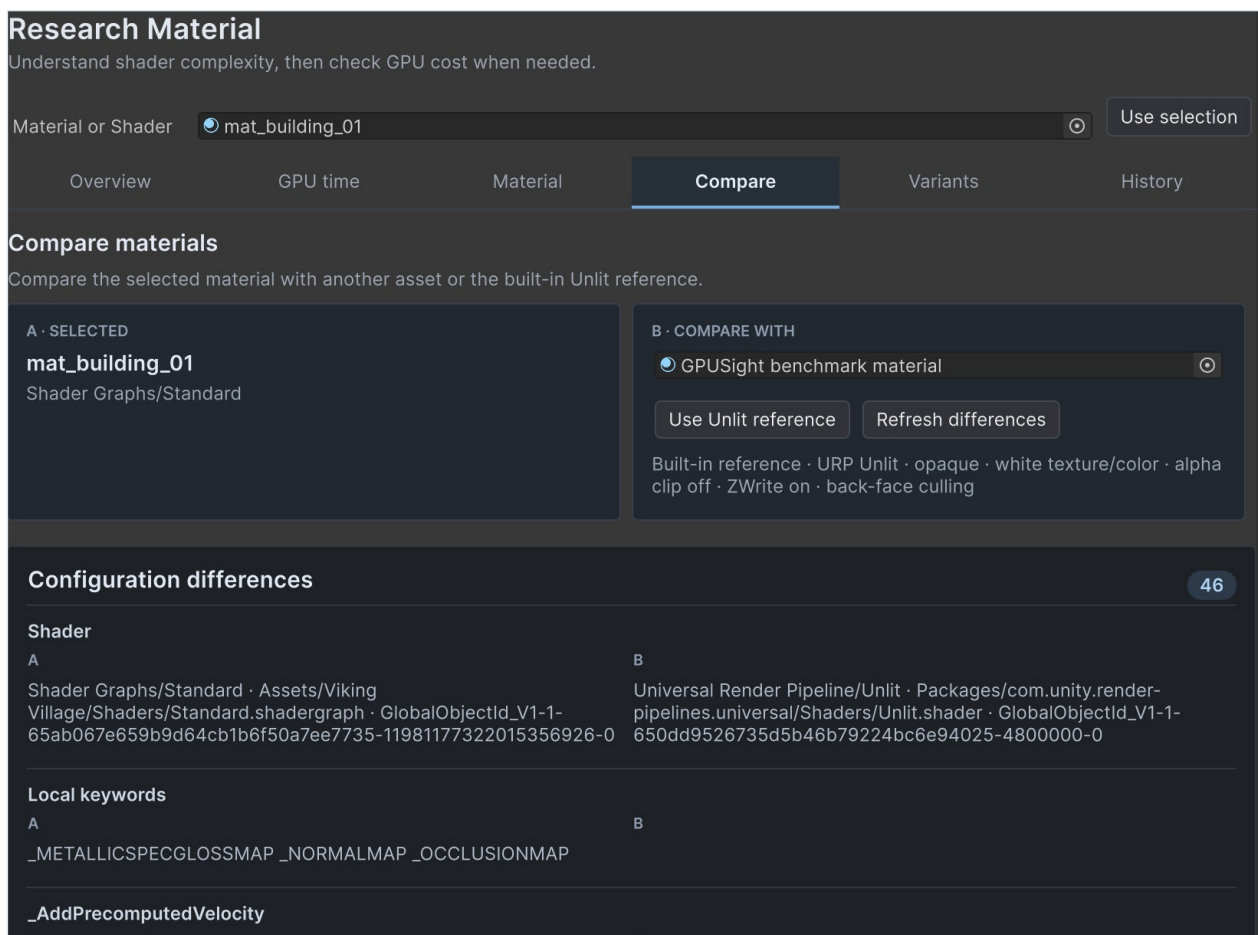
Group	Conditions tested
Resolution	256, 512 and 1,024 pixels per side at 64 draws.
Draw calls	1, 16 and 64 draws at 256 x 256.
Geometry	2 and 8,192 triangles at 256 x 256 with 64 draws.

These are eight grouped scenarios, not every possible combination. A rising resolution curve can suggest pixel-related work; a rising geometry curve can suggest vertex-related work. Fixed preview overhead, scheduling and measurement variability can affect either curve.

The illustrated run is noisy and not consistently monotonic. It demonstrates how to read uncertainty, not a clean scaling claim. Repeat stable conditions before deciding what the curve means, and verify the resulting change in the real scene.

Compare two material configurations

Open **Compare**. The Research target is **A**. Assign a candidate material as **B**, or keep the provided temporary URP Unlit reference. Review the configuration differences before pressing **Compare GPU time**.



Compare separates A and B and lists configuration differences before the controlled timing run.

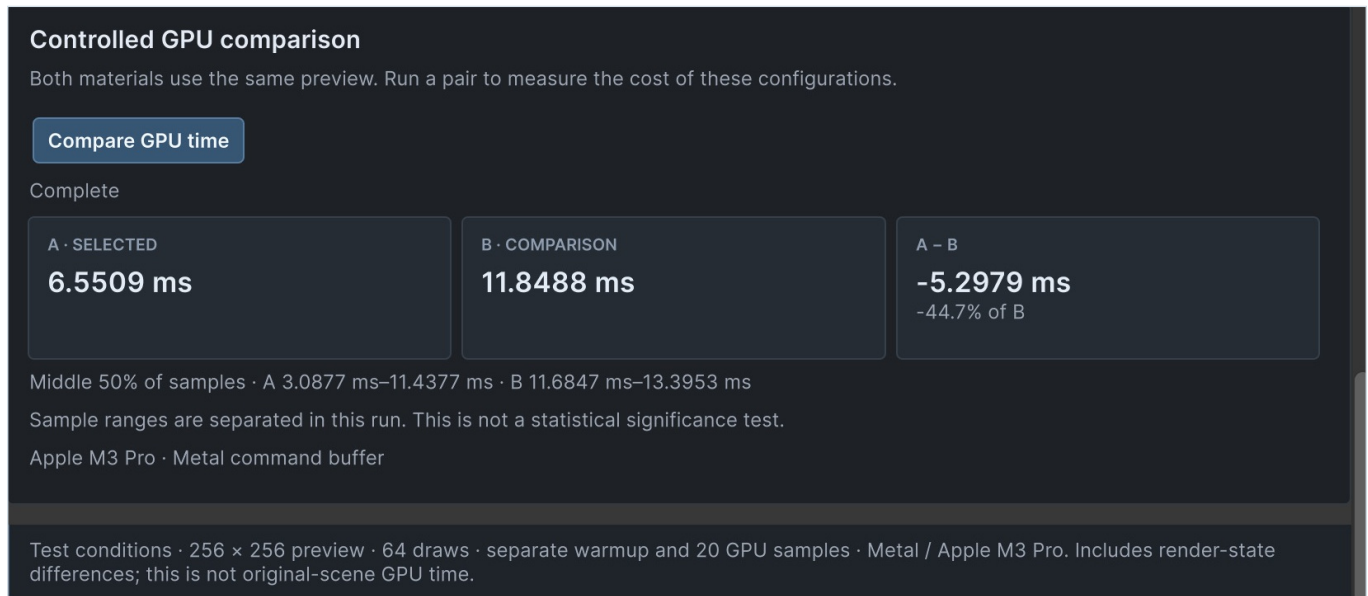
The temporary reference uses URP Unlit with white texture/color, opaque rendering, alpha clipping off, depth writing on, back-face culling and no instancing. It is created for the comparison and does not overwrite scene materials or assets.

Differences can include shader, keywords, property values, texture bindings, tiling/offset, render queue and enabled passes. A long difference list explains how inputs differ; it is not a performance score.

Use a deliberate B material when testing a specific optimization. An Unlit reference can provide context, but it is not visually equivalent to every Lit material and is not guaranteed to benchmark faster in every run.

Read the A/B timing result

Press **Compare GPU time**, wait for the complete A/B pair, then read both medians, **A - B** and the middle-50% ranges. Keep both materials unchanged while the comparison runs.



A completed material comparison with its device, backend and sample ranges. This is an example run, not an optimization claim.

The comparison runs A and then B under the same preview/backend conditions, with separate warm-up and **20 GPU samples per material**. Only a complete pair is published. Cancellation, changed inputs or an error must not be interpreted as a finished comparison.

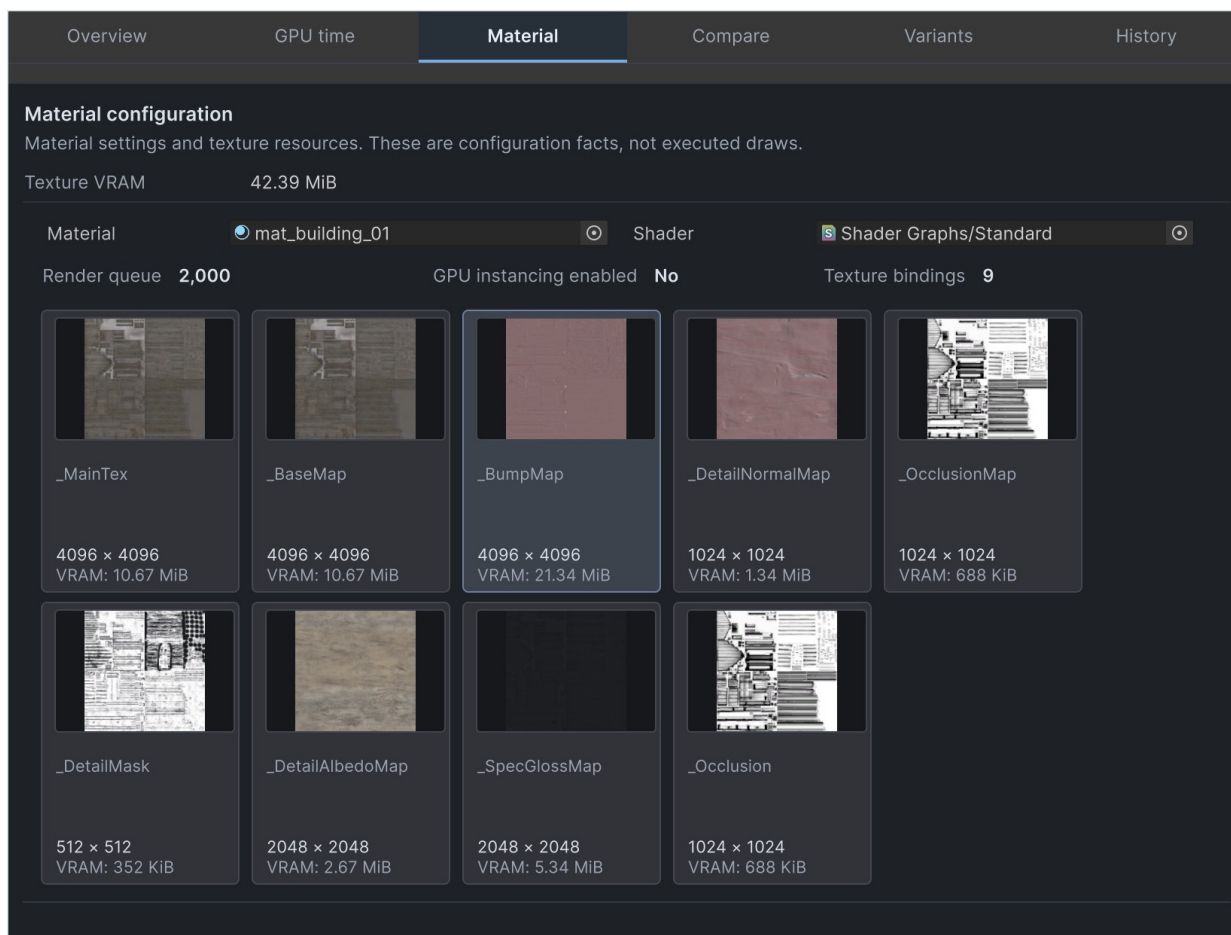
The sign here is **A minus B**: a negative delta means A measured lower than B. This differs from the current-minus-baseline convention on GPU time and the B-minus-A convention in Snapshots.

The middle 50% describes the central spread of samples. It is not a statistical significance test. Broad or overlapping ranges, or a surprising ordering, are reasons to repeat and inspect the conditions.

Finish by testing the candidate in its actual scene. A controlled preview cannot account for all scene geometry, coverage, overdraw, lighting, batching and pass scheduling.

Inspect material configuration

Open the **Material** tab to review the selected asset's shader, render queue, instancing flag and texture bindings. Use the asset references to confirm that you are examining the intended material and shader.



The Material tab shows texture bindings and resource sizes for the selected material. Repeated bindings can reference shared allocations.

This page describes configuration facts. The instancing flag alone does not prove that all scene draws were instanced, and a render queue does not measure draw cost.

Texture dimensions and allocation labels help explain a material's resource footprint. The capture's unique total and its individual binding rows can differ because one resource can be referenced multiple times.

A Shader-only selection does not expose the same Material, GPU time and Compare workflows because those require an actual configured Material. Choose a material asset when you need these panels.

Analyze a specific shader variant

Open **Variants** in **Inspect one variant** mode. Choose the pass, Fragment or Vertex stage, graphics API, target and keywords. Press **Analyze selected variant** to obtain source evidence for that exact request.

The screenshot shows the 'Research Material' interface with the 'Variants' tab selected. The interface is dark-themed and contains several sections for configuring a shader analysis.

- Material or Shader:** A dropdown menu showing 'mat_building_01' with a 'Use selection' button.
- Navigation Tabs:** Overview, GPU time, Material, Compare, **Variants** (active), and History.
- Inspect shader variants:** A section with the instruction 'Read the source work behind a pass. This analysis does not run a GPU benchmark.' It contains two buttons: 'Inspect one variant' (active) and 'Compare A/B variants'.
- Pass and stage:** A section with two dropdowns: 'PASS' (set to '0 · Universal Forward') and 'STAGE' (with buttons for 'Fragment' and 'Vertex').
- Compile context:** A section with 'GRAPHICS API' (set to 'Metal') and 'BUILD TARGET' (set to 'Standalone OSX'). Below these is a 'KEYWORDS' field containing 'FOG_EXP _ADDITIONAL_LIGHTS _LIGHT_COOKIES _MAIN_LIGHT_SHADOWS' and a 'Use material keywords' button. A note states: 'Keywords include current global overrides. Per-camera keywords and the variant used by an actual draw are not captured. Edits apply only to this source analysis; GPU benchmarks use the materials unchanged.'
- Analyze selected variant:** A button that triggers the analysis.
- Requested configuration:** A line of text showing the current settings: 'Requested configuration · Fragment · Metal · StandaloneOSX. Source heuristics, not compiler instructions or GPU time.'
- Source metrics:** A table with two columns: 'Source metric' and 'Selected variant'. The first row shows 'Weighted texture reads' with a value of '7'.

Variants exposes the selected pass, stage, source target and keyword configuration.

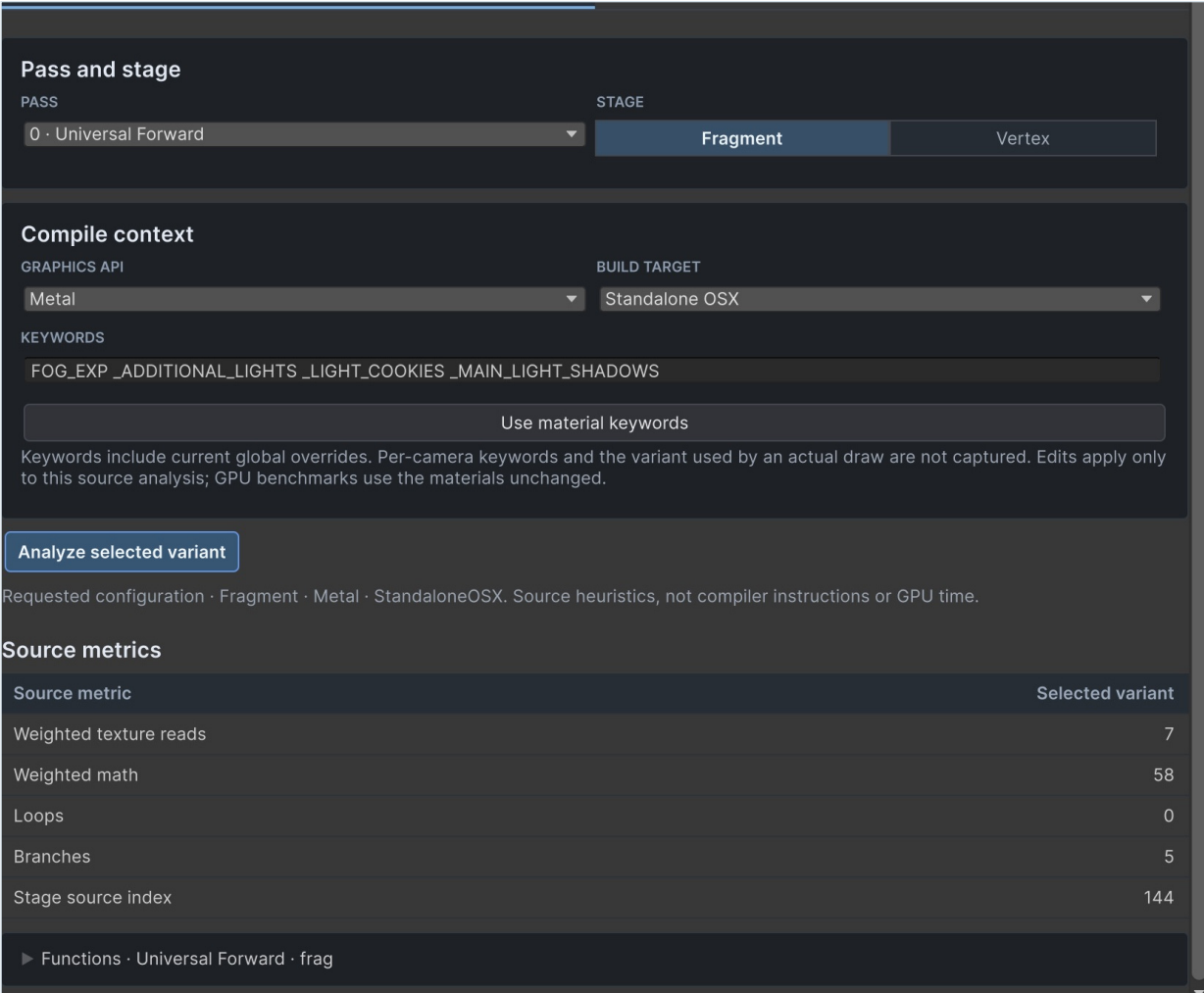
Use the selected material's current keywords as a starting point. Change them deliberately when comparing a feature switch; the reset action restores the material-derived request rather than editing the material asset.

The API and target fields control source preprocessing. Choosing Vulkan, D3D or Metal here does not switch Unity's active graphics API and does not run a test on hardware for that API.

If preprocessing or entry-point resolution is partial or unavailable, read the stated reason. An unsupported request should not be silently treated as a complete analysis of a different variant.

Follow source evidence to functions

After variant analysis finishes, scroll to the result and review its stage index, weighted texture/math/control signals and status. Expand **Functions** and use the available source-opening action to investigate a specific function or line.



The variant result exposes stage-level source metrics and a Functions section for deeper inspection.

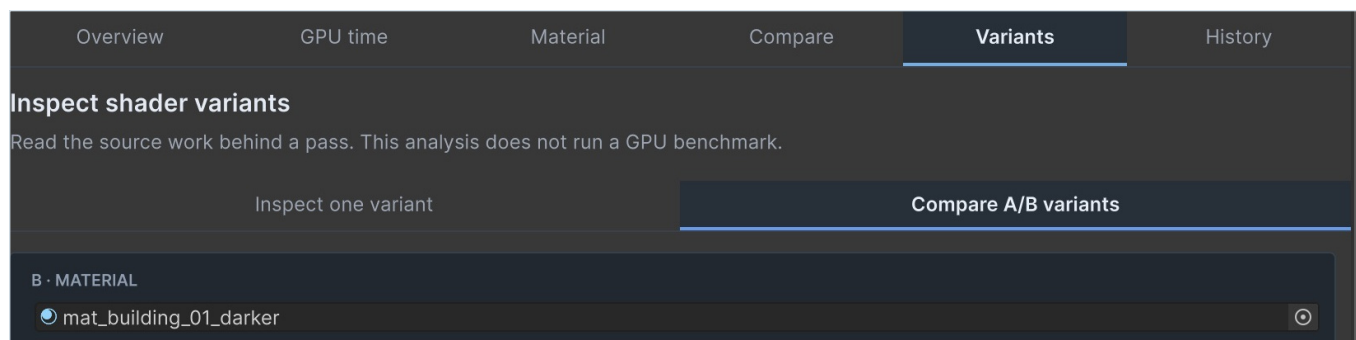
The function breakdown narrows a source-level question: for example, which function contains the additional texture work or loop. It is not a sampled GPU call stack or a per-function GPU duration.

Source locations refer to the files available to the analysis, including resolved includes. Some generated or incomplete sources can limit navigation. Retain the exact pass, stage, target and keyword context when reporting a finding.

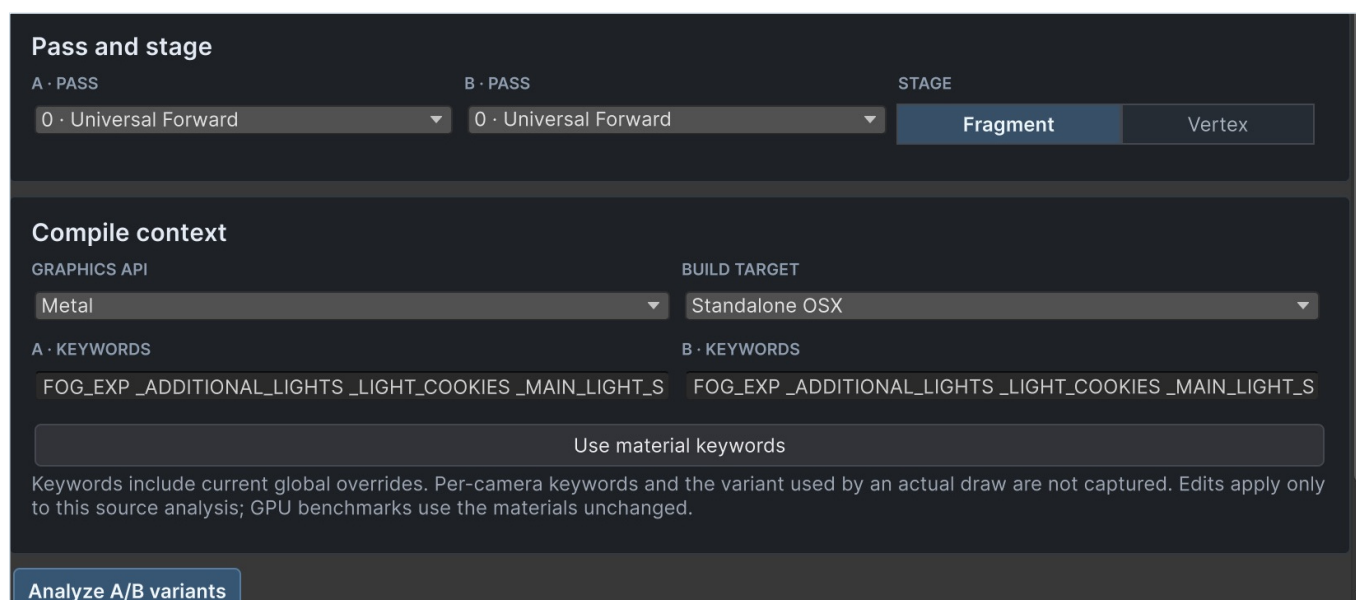
This is the point to form a small, testable shader change. Validate appearance, compare the intended variants, then run the material and scene timing workflows under comparable conditions.

Compare shader variants

Switch Variants to **Compare A/B variants**. Assign the B material, select the passes and keyword sets for A and B, and press **Analyze A/B variants**. Use the same stage and source target when asking about one feature difference.



Compare A/B variants and the B material picker. A remains the material selected in Research.



Passes, stage, target and keywords for the pair, followed by Analyze A/B variants.

Source deltas are **A minus B**. A lower source index can support a hypothesis about reduced source work, but does not prove a faster frame. Optimizations, compiler behavior and GPU architecture affect the actual result.

Use **Compare** for controlled material GPU timing after inspecting source differences. Then repeat your gameplay scenario in **Measure**. The three views answer source, preview and frame questions respectively.

Review retained material evidence

Open **History** in Research Material. Inspect the retained benchmark configurations, sample counts, recorded conditions, device and backend. Read Current captured frame separately when that evidence is available.

Overview	GPU time	Material	Compare	Variants	History
Current captured frame					
GPU attribution		Unavailable · no current-scene deep capture			
Retained benchmark results					
2 configurations · 40 samples					
mat_building_01					0.029 ms
0.027–0.142 ms · 20 samples · 256 ² · 64 draws					
9/27/2026 8:58 PM · Apple M3 Pro · Metal command buffer					
mat_building_01					2.120 ms
0.614–15.152 ms · 20 samples · 256 ² · 16 draws					
9/8/2026 7:32 PM · Apple M3 Pro ·					

History keeps the context of retained benchmark results. The current-scene deep-capture field is explicitly unavailable in this session.

Results from different dates or draw counts are not interchangeable. The example contains both a 64-draw and a 16-draw configuration; a direct speedup percentage between them would mix conditions.

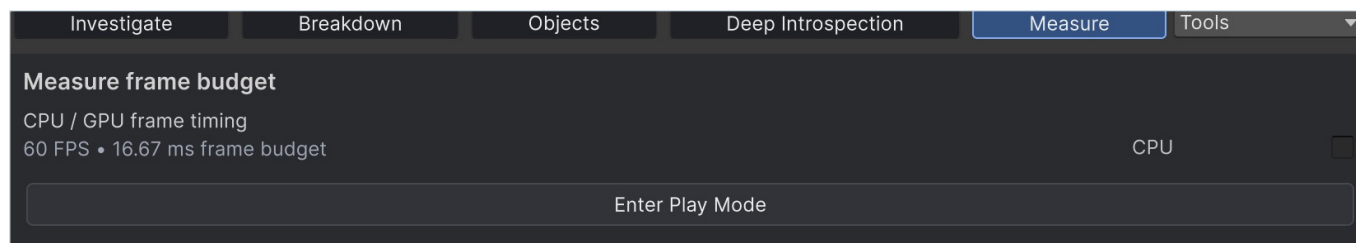
Current captured frame requires suitable current-scene deep-capture evidence. New RenderDoc capture is temporarily disabled in this release. Retained preview benchmark results do not fill in missing per-object frame GPU attribution.

History is useful for recovering the context of a result and deciding what needs to be remeasured. Export the material report or a broader snapshot when you need evidence to survive Library cleanup or to accompany a review.

An old measurement can remain readable without being a fresh measurement of the current material and scene. Its timestamp and conditions are part of its meaning.

Start a representative Play session

Open **Measure** and press **Enter Play Mode**, or use Unity's Play button. Run a representative gameplay sequence and wait for timing samples. The panel identifies the live context and available signals.



Measure in Edit Mode offers Enter Play Mode. The frame-budget target comes from the selected profile.

Enable **CPU** if you also want CPU timing visible. A missing GPU timing backend remains unavailable; the tool must not substitute a geometry estimate for a missing measured frame time.

Keep the scenario repeatable

Use the same route, camera, effects, quality settings and resolution before and after a change. Allow loading and warm-up to settle if they are not part of the scenario you intend to evaluate.

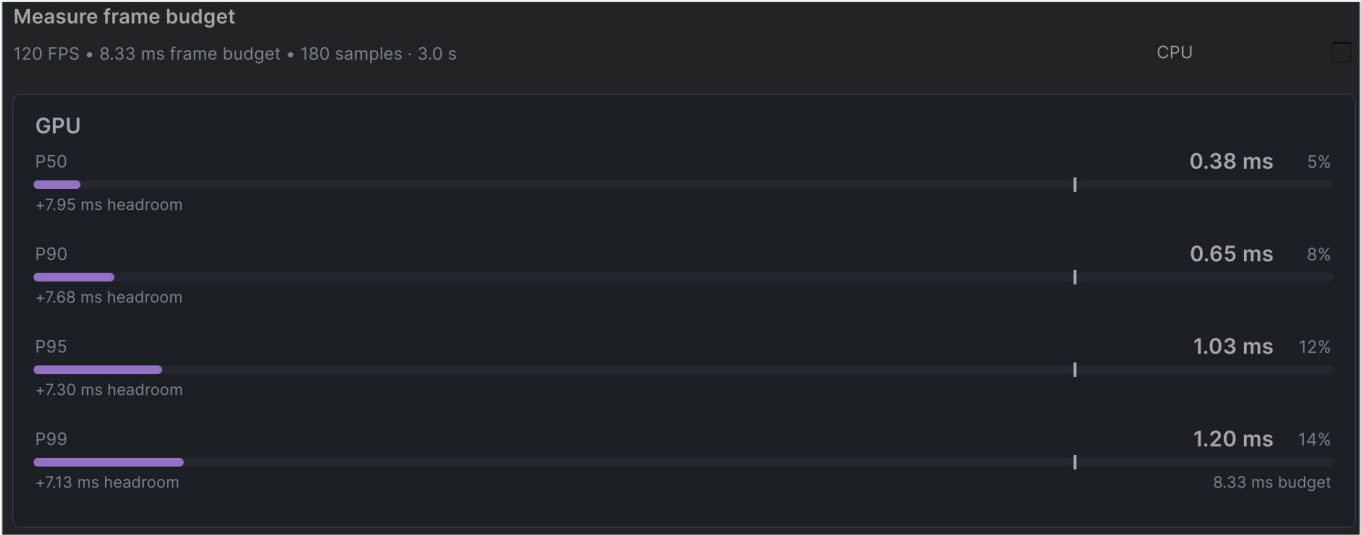
The measurements here describe the current Editor Play Mode context. They are useful for local investigation, but an Editor result is not a validation result for every Player build or target device. Use your normal target-device profiling workflow for final acceptance.

Background import, other Editor windows, explicit diagnostic tests and recording activity can alter the measured session. Record relevant conditions instead of assuming all runs are identical.

If the header still says Requires Play, Unity has not entered an accepted timing session. If samples arrive without GPU values, inspect backend/platform availability rather than interpreting the missing values as 0 ms.

Read frame budgets and percentiles

After samples arrive, inspect the GPU percentiles and enable CPU when relevant. Compare **P50**, **P90**, **P95** and **P99** with the displayed budget and headroom.



A real Play Mode timing capture. The profile targets 120 FPS with an 8.33 ms budget; the surrounding Game View does not change that target.

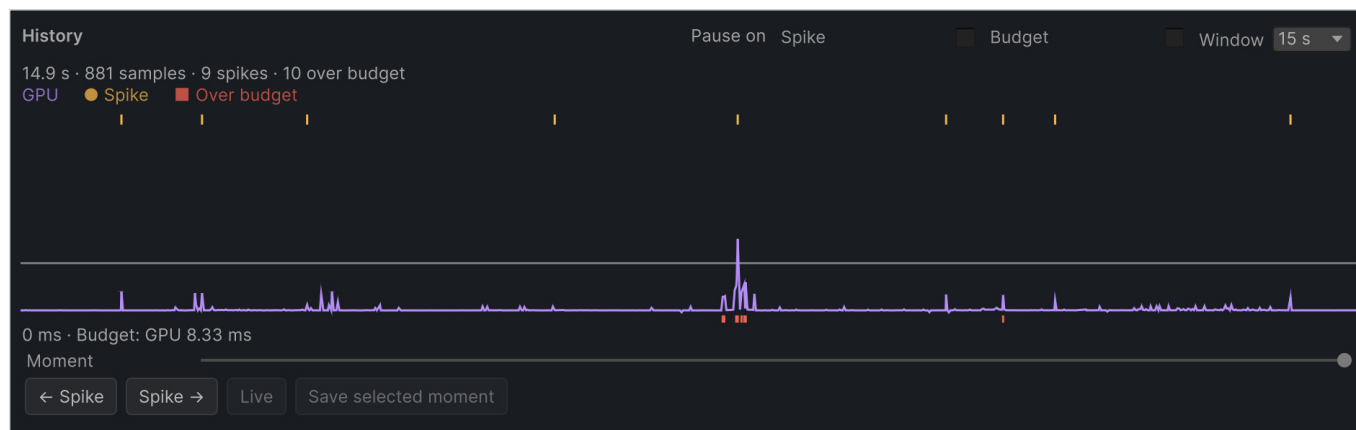
Value	What it describes
P50	The median frame sample: half the eligible samples are at or below it.
P90 / P95	The slower portion of the distribution becomes more visible.
P99	The upper tail. It needs enough representative samples to be useful.
Headroom	Remaining time relative to the relevant displayed budget.

A good median can coexist with noticeable hitches. Read the higher percentiles and history before concluding that a scenario is stable. Percentiles describe the accepted sample window; they are not an average FPS measurement.

CPU uses the target frame interval; GPU uses the configured GPU budget. CPU and GPU can overlap. Their displayed timings should not automatically be added to estimate the duration of a frame.

Investigate spikes and budget misses

In **History**, choose a **Window** of 10, 15 or 30 seconds. Use the previous/next **Spike** buttons or the **Moment** slider to inspect a retained point. Press **Live** to return to the advancing view.



The history timeline distinguishes spikes from budget misses and provides Moment, Spike and Live navigation.

Spike marks an unusual increase relative to recent local timing behavior. Detection uses recent samples and merges nearby events; it is not simply another name for over budget.

Over budget means the sample exceeded an absolute configured timing allowance. A consistently slow workload can be over budget without producing a new spike at every moment. A brief relative spike can still be below budget.

Enable **Pause on Spike** or **Pause on Budget** only when you want Unity to pause the local Play Mode simulation at the matching event. These controls have a different purpose from GPUSight's Freeze button.

Reproduce the event and inspect its surrounding gameplay and rendering activity. A timeline event identifies when a problem happened, not automatically the object or shader that caused it.

Preserve a timing moment

Select an eligible historical moment with the slider or Spike navigation, then press **Save selected moment**. Open **Tools > Snapshots** to find the saved state and inspect its context.



Moment and Save selected moment at the bottom of History. In this live-position example, saving is disabled until a suitable historical moment is selected.

The action saves the already accepted historical state. It does not rewind the GPU or create a new deep capture of a frame that has already passed. Evidence absent at that moment cannot be recovered by saving it later.

Read GPU history in its own scope



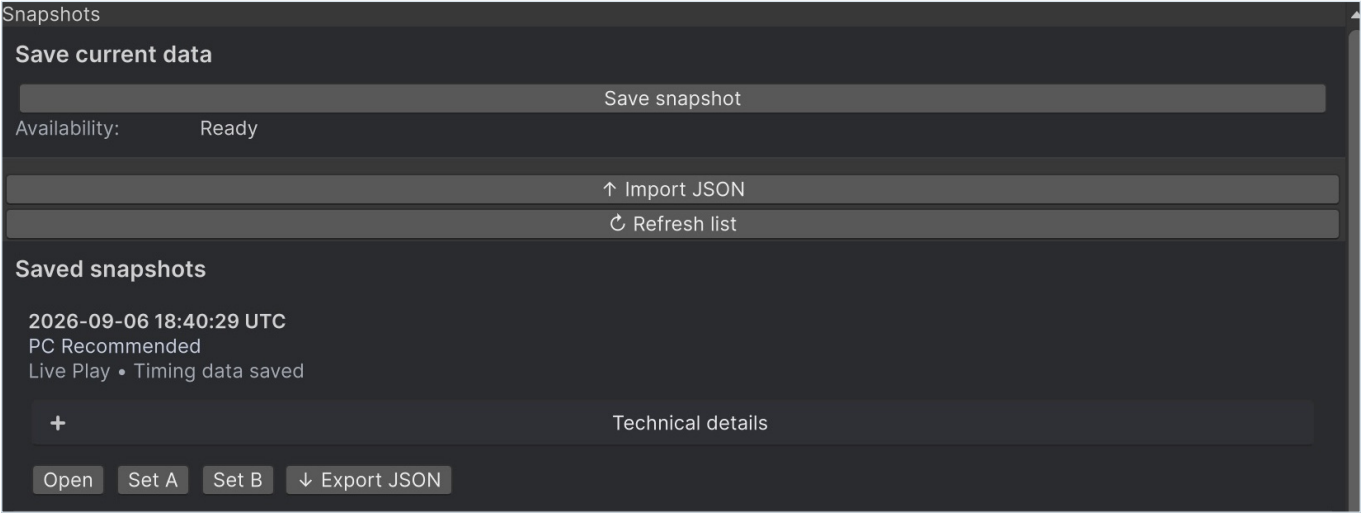
GPU history labels its native allocation scope as the Editor process in this Metal session.

Native allocations and observed GPU buffers describe information available from the current backend. They are not a complete measure of GPU utilization, and an Editor-process allocation total includes more than one selected material or renderer.

Use the saved timing moment as an investigation reference. A before/after comparison still requires a second run with compatible scene, target and measurement conditions.

Save the accepted analysis state

Open **Tools > Snapshots**. Check that **Availability** is Ready, then press **Save snapshot**. The new row identifies its time, profile, mode and saved evidence.



Save snapshot stores the current accepted data. Existing rows provide Open, Set A, Set B and Export JSON.

A snapshot can preserve context, metrics, timing already available, contributors and scene evidence, plus optional benchmark, resource or deep-capture data present at capture time.

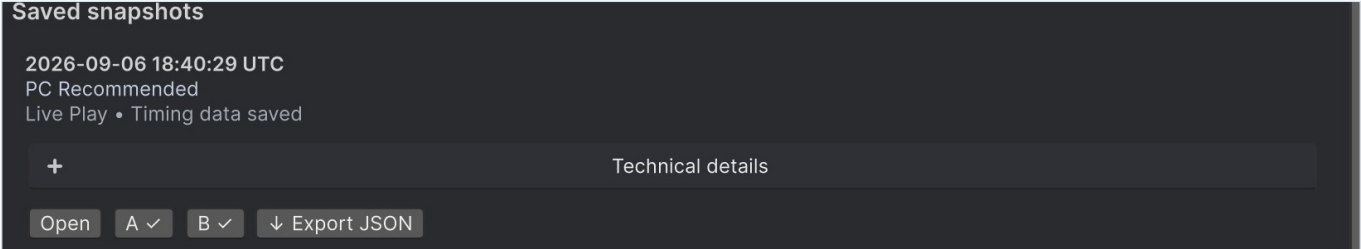
Save snapshot does not force Refresh or start a new GPU measurement. Obtain the evidence you need first, check its freshness and scope, then save it. Saving an unavailable signal keeps its absence honest.

Snapshots are stored locally under **Library/GPUSight/Snapshots**. They are not a scene backup and do not restore changed assets, GameObjects or material settings. Export important snapshots when they need to survive Library deletion or be shared.

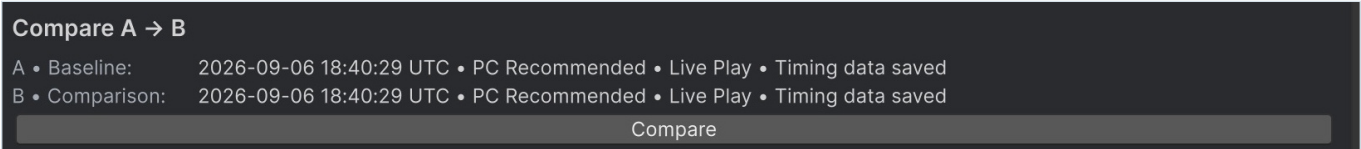
Use **Open** to inspect a saved snapshot read-only. Its recorded context remains historical even if the live scene has since changed.

Choose a before and after

In the snapshot list, press **Set A** on the baseline and **Set B** on the later comparison. Scroll to **Compare A > B**, confirm the two selected records and press **Compare**.



Selected A and B are marked on their rows. This illustration deliberately uses one record for both, so it demonstrates selection rather than an improvement.



The comparison area repeats A and B context before the Compare action. In a real before/after workflow, choose two distinct compatible records.

A good pair uses the same scene/view, profile, runtime context and measurement source, with a deliberate change between the two captures. If a camera position, graphics API or quality setting also changed, that difference needs to be part of the interpretation.

The selection does not edit either snapshot or the scene. Snapshot comparison evaluates the evidence stored in the selected files; it does not recalculate the old scene using today's settings.

Snapshot deltas use B minus A. For a lower-is-better metric such as compatible time, a negative delta is an improvement. Always check the metric's unit and comparability status before interpreting the sign.

Read compatibility before deltas

After comparison, read the verdict and reasons, then expand **Changed**, **Unchanged**, **Unavailable** or **Mixed**. Use **Technical details** to understand the stored context and evidence.

Warning — deltas shown with context differences

Reason: A supported legacy snapshot schema version is in use.

Reason: A supported legacy snapshot schema version is in use.

Warnings: 3

-

Changed (0)

No changed values.

+

Unchanged (8)

+

Unavailable (1)

+

Mixed (0)

+

Technical details

A real comparison with a Warning verdict. This example uses the same legacy snapshot twice, so it correctly shows no changed values.

Result	How to use it
Compatible	The comparison passed its context checks. Individual metrics still need compatible evidence.
Warning	Read the differences and limitations before interpreting eligible deltas.
Blocked	The contexts cannot support the requested comparison. Capture a suitable pair.
Unavailable	One or both values are absent or unusable. No numeric improvement is established.
Mixed	Unit, source, scope or evidence quality differs, so a numeric delta is withheld.

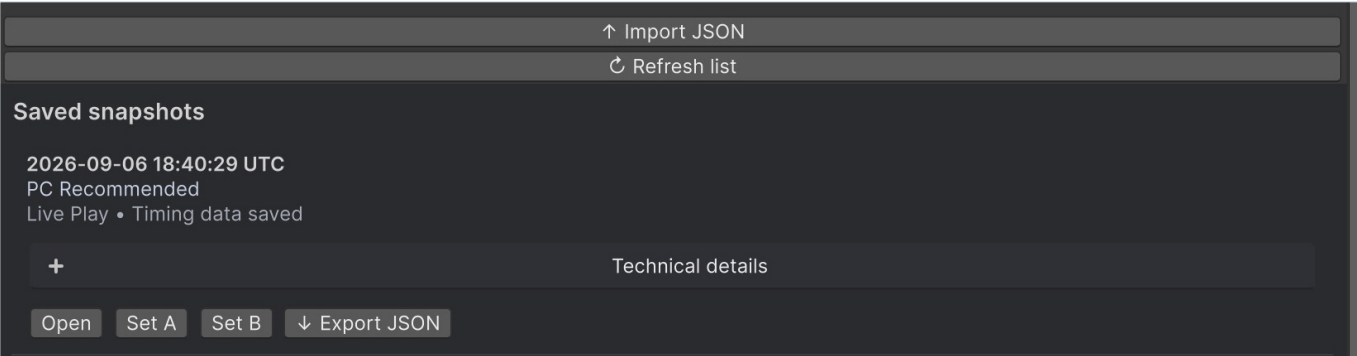
The warning text in this illustration says that a supported legacy snapshot schema is being used. Keeping old data readable does not manufacture missing measurements.

Snapshots from different package versions can be blocked by the compatibility policy. Updating GPUSight does not rewrite older snapshot files.

A visible numerical change should be traced back to its source and scope. Snapshot comparison can preserve a geometry estimate alongside a measured timing result; their separate evidence meanings remain intact.

Export and import snapshot JSON

Press **Export JSON** on the required snapshot row and choose a file location. On the receiving project/session, use **Import JSON** and select that file. Use **Refresh list** to reread the local catalog when needed.



The snapshot transfer controls: Import JSON, Refresh list and per-row Export JSON.

Exported snapshots keep the captured evidence and context. They do not package the Unity project, scene assets, textures or a reproducible running build. A recipient may be able to inspect values without having the original assets available for navigation.

Import/export is bounded to **64 MiB per snapshot JSON**. Reimporting the identical payload with the same ID reuses the existing record. A conflicting payload with the same ID is rejected rather than replacing the stored snapshot.

Share enough context to reproduce the question

Alongside the export, describe the scene or scenario, Unity and pipeline versions, graphics API/device, selected profile and the intentional change. Include the material benchmark JSON when a preview result is central to the finding.

Export important evidence outside Library. Library is local working data, and clearing it can remove saved snapshots and benchmark baselines.

REFERENCE

Compatibility and optional evidence

The current DLL release workflow has been checked with **Unity 2022.3.62f3 + URP 14.0.12** and **Unity 6000.0.62f1 + URP 17.0.4**. These exact checks do not certify every other Editor/pipeline/API combination or every optional backend.

Feature	Availability boundary
Scene workload and Scene View maps	Editor investigation; individual diagnostic paths have renderer and source-data limits. Shared Scene View support includes Built-in, URP and HDRP paths, but optional GPU tools have separate requirements.
Material preview benchmark	Active URP, a supported material and an available GPU timing backend. The test runs on the actual host device/API.
Native resource allocations	Current inventory depends on Metal. Missing native values can remain unavailable elsewhere.
Play frame timing	Depends on the platform/API and timing source. CPU and GPU availability can differ.
Optional Game/Base diagnostic capture	A separate URP integration for bounded object-ID/visible-coverage evidence. Ordinary Scene View maps do not require installing its renderer feature.
Deep-capture history	Previously saved deep-capture snapshots remain readable. New Windows/RenderDoc capture and external replay analysis are temporarily disabled in this release.
Player Session	Hidden in this release workflow. Remote device attachment is not a main user workflow described by this guide.

Reading older capture evidence

Some saved records can contain captured passes, resources or GPU attribution from an earlier verified capture. Their original context remains part of the result. Reading such a record does not enable a new capture or establish current-scene attribution.

Use the available scene, source, controlled preview and Play timing workflows for new investigations. Deep Introspection's geometry and material facts remain useful without RenderDoc. Where fresh GPU attribution is unavailable, preserve that status rather than substituting a preview benchmark.

REFERENCE

Troubleshooting

Symptom	Likely next check
GPUSight menu is missing	Let import/compilation finish and inspect Unity Console errors. Check that the intended delivery is present and not duplicated.
Deep Introspection says Select an object	Select an individual contributor inside Objects. Confirm that the accepted scene data contains that renderer.
A map disappears in Play Mode	Scene View diagnostic maps are an Edit Mode workflow. Use Measure for Play timing.
Freeze never updates	Resume live data, request Refresh and inspect the accepted context.
Research shows No reference	The selected source result has no eligible scene distribution. Run the relevant shader map analysis and check its coverage.
GPU benchmark cannot run	Read the reason. Check active URP, shader/material support, live texture inputs, backend availability and another active test.
GPU time or VRAM is unavailable	Check the feature's actual platform/API/backend requirements. Target profiles do not supply missing measurements.
Texel statistics are missing	Inspect mesh readability, UVs, primary texture support and sample limits. A visual grid can still be available.
A profile reports Warning	Compare target platform/API with the running Editor. Review the budget and context rather than assuming a device is emulated.
Snapshot comparison is Mixed/Blocked	Inspect context, unit, source, scope, freshness and evidence quality. Repeat the capture under comparable conditions.
Timing changes sharply between runs	Check background work, warm-up, conditions and sample spread. Repeat the same scenario before attributing a cause.

A useful issue report

Include a screenshot showing the header and relevant status, the exact reproduction action, Unity/pipeline/API/device, and the relevant exported JSON when available. State whether the result is an estimate, a controlled preview, a Play frame sample or a saved capture. This makes it possible to investigate the same question.

REFERENCE

Quick reference

Term	Meaning in this guide
Contributor	An object or grouping included in the accepted scene analysis.
Geometry share	Estimated geometry contribution relative to the accepted scope.
Coverage	The part of a diagnostic view covered by the captured geometry.
Overdraw	A diagnostic geometry-layer signal with its own rendering assumptions.
Texel	One texture element. Texels per pixel relates texture scale to the current view.
Ideal mip	A screen-space estimate of a suitable mip level, not a guaranteed observed sampling level.
Source index	A weighted source-analysis summary, not a GPU-time unit.
Backend	The actual mechanism that provides a measurement or resource observation.
Baseline	A retained reference result for a later compatible comparison.
Freshness	Whether evidence still belongs to the accepted current context.
MiB	1,048,576 bytes. Distinct from decimal MB.

Delta signs at a glance

Panel	Delta direction
GPU time baseline	Current minus baseline.
Material Compare	A minus B.
Variant Compare	A minus B for source metrics.
Snapshots	B minus A for compatible stored metrics.

A complete optimization claim needs three parts: the deliberate change, compatible before/after evidence, and acceptable visual behavior in the intended scenario. GPUSight helps keep those parts connected while preserving what each signal can actually tell you.