UI Blur — Manual

Real-time UI Blur for Unity UGUI, built natively for the Universal Render Pipeline (URP).

No more fake glass effects — blur the UI and 3D scene behind your panels in real time, using a high-performance Kawase blur pipeline.

---

1. Requirements

- Unity 2023.2 or before.
- URP (Universal Render Pipeline) — **no Built-In or HDRP support**

---

2. Setup

For step-by-step installation instructions with annotated screenshots, see the separate **Setup Steps** document included with this package.

In short: add the **UI Blur Feature** to your URP Renderer asset, assign the **Kawase Blur** shader to it, then add the **UI_UIBlurPanel** material and **UIBlurPanel** script to any UI Image you want blurred.

---

3. Canvas Compatibility

> ⚠ **IMPORTANT — Background and Blur Panel must be on SEPARATE canvases**
> The UI or image you want blurred **cannot live on the same canvas** as the blur panel itself. Put whatever you want blurred (background images, other UI) on its own **Screen Space - Camera** or **World Space** canvas, and keep the blur panel on a **separate, Screen Space - Overlay** canvas.
> If the background and the blur panel are on the **same** canvas, the panel will capture and blur itself, producing a smeared, compounding blur instead of a clean frosted-glass effect. This is the single most common setup mistake — always use two separate canvases.

- **Screen Space - Overlay**: fully supported out of the box for the blur panel, no extra setup.
- **Screen Space - Camera** and **World Space**: use these for the *background* content you want blurred (3D scene, other UI) — on its own canvas, separate from the blur panel.
- The canvas containing the blur panel itself must always remain **Screen Space - Overlay**.

See the included demo scene for a working reference setup with the background and blur panel correctly split across two canvases.

---

4. Settings Reference

Renderer Feature (on the URP Renderer asset)

Setting	Description
Kawase Shader	The blur shader used for the capture pass. Required.
Downsample (1–8)	Resolution divisor for the capture. Higher = cheaper but softer/blockier. 2–4 is typical for UI backdrops.
Blur Iterations (1–8)	Number of down/up Kawase passes. Higher = smoother, wider blur radius, more cost.
Blur Offset (0–4)	Per-pass sample offset. Higher = wider blur per iteration.
Render Pass Event	Defaults to After Rendering Transparents. Change only if you have a specific pipeline ordering need.

UIBlurPanel (per-panel component)

Setting	Description
Continuous Refresh	Off by default. Enable only if the background behind the panel animates on its own (e.g. a live 3D scene) — forces a fresh capture every frame instead of only on move/resize. Leave off for static UI to get the zero-cost-when-idle behavior.
Downsample	Mirrors the Renderer Feature's Downsample so you can tune it from the panel without opening the Renderer asset. Shared across all panels using the same Renderer Feature — changing it here changes it everywhere.
Blur Iterations	Same as above, mirrors Blur Iterations.

> **Note:** Downsample and Blur Iterations are currently shared/global settings — every panel using the same Renderer Feature instance uses the same blur radius. Independent blur *strength* per panel isn't supported yet; if you need visibly different blur amounts on screen simultaneously, add a second UI Blur Feature to the Renderer asset.

---

5. How It Works (brief)

1. The Renderer Feature captures the Main Camera's rendered frame after transparents.
2. The capture is downsampled and run through a multi-pass Kawase blur (down-then-up chain).
3. The result is published as a **single shared texture**, sampled by every visible blur panel — one capture serves all panels, not one per panel.
4. Each `UIBlurPanel` computes its own on-screen rect and samples the matching region of that shared texture.
5. The pass only re-runs when a panel first appears, moves, resizes, or Continuous Refresh is enabled — a static panel over a static scene costs one capture, then nothing further.

---

6. Performance

Performance depends on resolution and quality settings (Downsample / Blur Iterations).

Typical Editor GPU cost: ~2–3 ms at default quality settings.

Zero GPU cost while idle when Continuous Refresh is disabled and the scene is static — the pass doesn't run at all until something changes.

Tips:

- Leave Continuous Refresh off unless the background genuinely animates on its own; the dirty-refresh system already handles panel movement/resizing automatically.
- Downsample 3–4 with 2–3 Blur Iterations gives a fully "frosted" look at low cost — there's rarely a visual reason to go lower.
- Cost scales with how often panels move/resize and how many *distinct* Renderer Feature instances are active, not with panel count — many static panels sharing one blur setting cost the same as one.

---

7. Limitations

- ⚠ **URP only.** Built-In Render Pipeline and HDRP are not supported.
- ⚠ **XR/VR has not been tested.**
- ⚠ **Screen Space - Camera and World Space** backgrounds are supported, but the canvas containing the blur panel itself must remain **Screen Space - Overlay** (see Section 3).
- All panels sharing one Renderer Feature instance share the same blur radius (Downsample / Blur Iterations) — see Section 4.

8. Troubleshooting

Panel shows flat grey instead of a blur. Usually means the panel is sampling outside the valid region of the shared blur texture — double check the panel's canvas render mode and that a camera is correctly assigned where required.

Panel shows a smeared, compounding blur that gets worse over time. The panel's canvas is not in Screen Space - Overlay mode, so it's being captured by the same pass that's blurring it. See Section 3 — the panel's own canvas must stay Overlay even if the background behind it uses Camera or World space.

Changing one panel's blur strength changes all panels. Expected — Downsample and Blur Iterations are shared per Renderer Feature instance (Section 4). Add a second UI Blur Feature if you need independent blur amounts.

Support

For questions, please reach out via [karsh02526@gmail.com] — include your Unity version and a description of your canvas setup with screenshots when reporting an issue.