Liquid Glass Shader - User Guide

Liquid Glass Effect for Unity UI

1. Introduction

Liquid Glass Shader brings recent *“liquid glass”* design language directly into Unity UI.

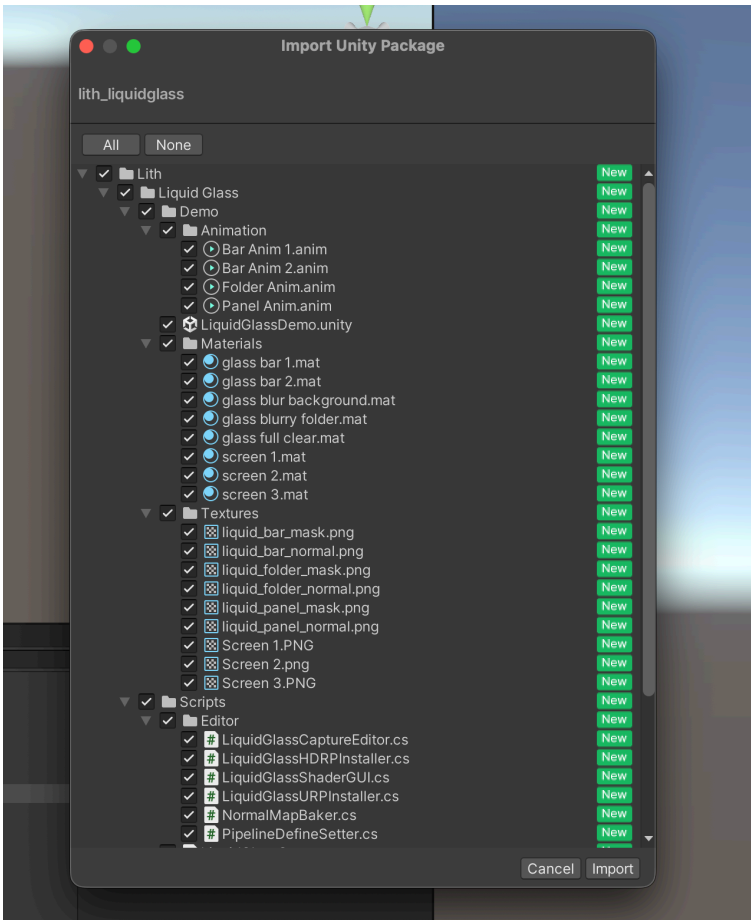
It delivers blurred, refractive, and glossy glass panels optimized for mobile, achieving **60 FPS even on older devices (iPhone 6s, low-end Androids).**

Key features:

- Single **dual-pass Kawase blur** (high-performance).
- Full support for Built-in, URP, HDRP.
- Flexible **shader inspector** with optional modules (Distortion, Gloss, Saturation and Normal Map Baker).
- Works seamlessly with Unity Canvas UI.

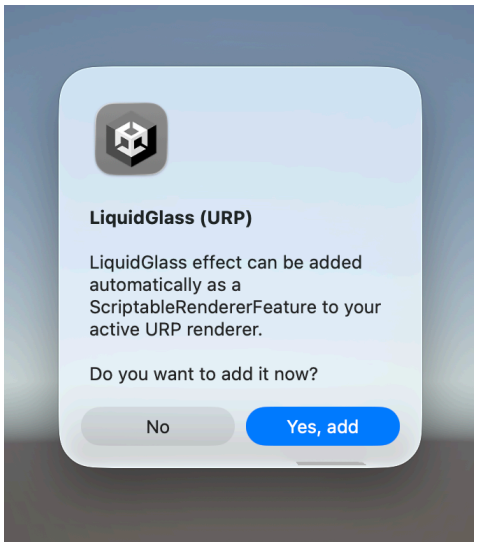
2. Installation

1. Import the package into your Unity project.



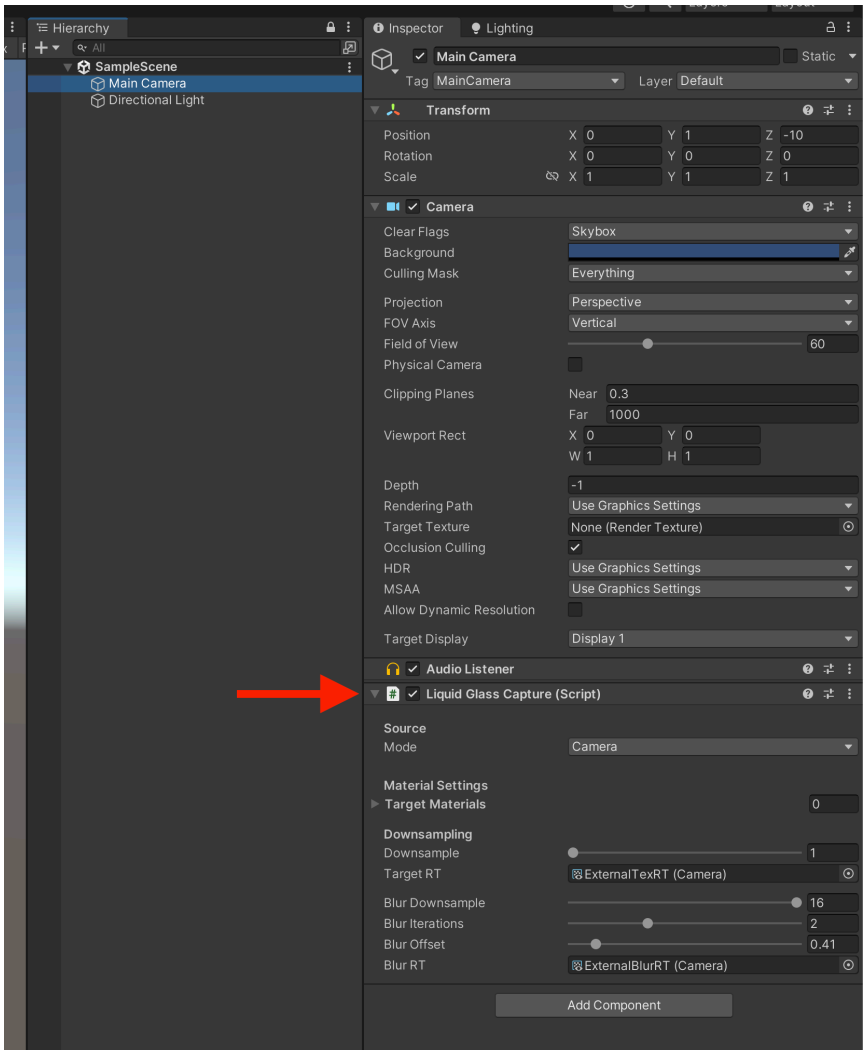
2. On first install:

- **URP / HDRP** → A popup will offer to auto-add the required Render Feature / Custom Pass.



- If you decline, you can add it manually (see Section 5).

3. Built-in RP → Attach LiquidGlassCapture script to your Camera.



3. Creating Liquid Glass UI

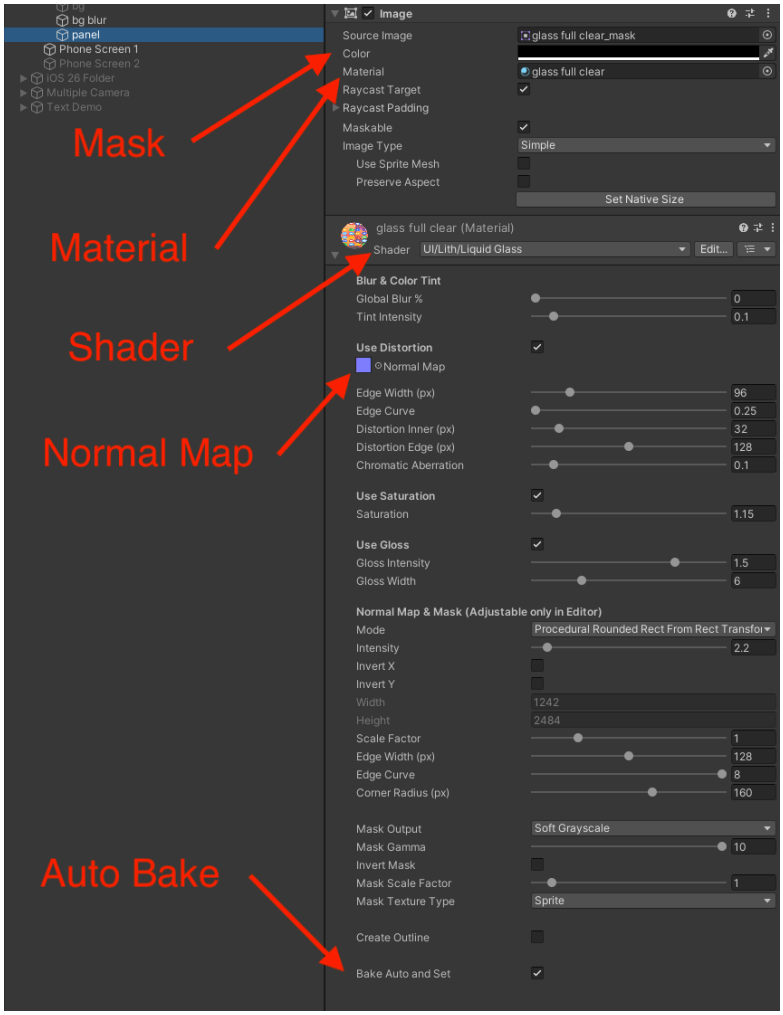
To replicate Liquid Glass effect, **every UI element requires its own Normal Map (and often a Mask).**

Workflow Example — A Rounded Bar (512×128)

1. Create a new **Material** using Liquid Glass shader.
2. In your Unity Canvas:
   - Add an **Image** component for the bar.
   - Apply your new Material to the Image.
3. Adjust shader parameters (edge width, distortion, gloss, normal map & mask etc.) until the effect matches your desired look.

4. Generate Normal Map & Mask textures.

   - When parameters change, the shader automatically rebakes the Normal Map and Mask.
   - Both textures are written directly over their previous versions, no manual export needed.
   - Auto Bake can be disabled if you prefer manual control.
     In that case, disable “Bake Auto and Set” and choose a save location.
   - The generated Normal Map is automatically assigned to the current material.
   - The generated Mask Texture is automatically applied to the attached UI Image or Raw Image component.



⚠️ Notes:

- If the UI element has **no rounded edges**, Mask is optional.
- **Gloss effect requires Soft Mask** (anti-aliased edges). With Hard Mask, gloss appears very weak.
- Each differently sized UI element should have its own Normal Map + Material (or use MaterialPropertyBlock if you want to reuse).
- Multiple blurred UI elements do **not** impact performance. Blur is computed once per frame.

To apply the Liquid Glass effect on your UI:

Make sure the UI elements you want to capture (the ones behind the glass) are placed on a Canvas set to Screen Space - Camera or World Space.

Assign that Canvas to a Camera which has the LiquidGlassCapture component.

Then, place your Liquid Glass UI elements (the glass panels themselves) on a Canvas set to Screen Space – Overlay.

Assign the corresponding material (with your normal map and mask) to those overlay elements.

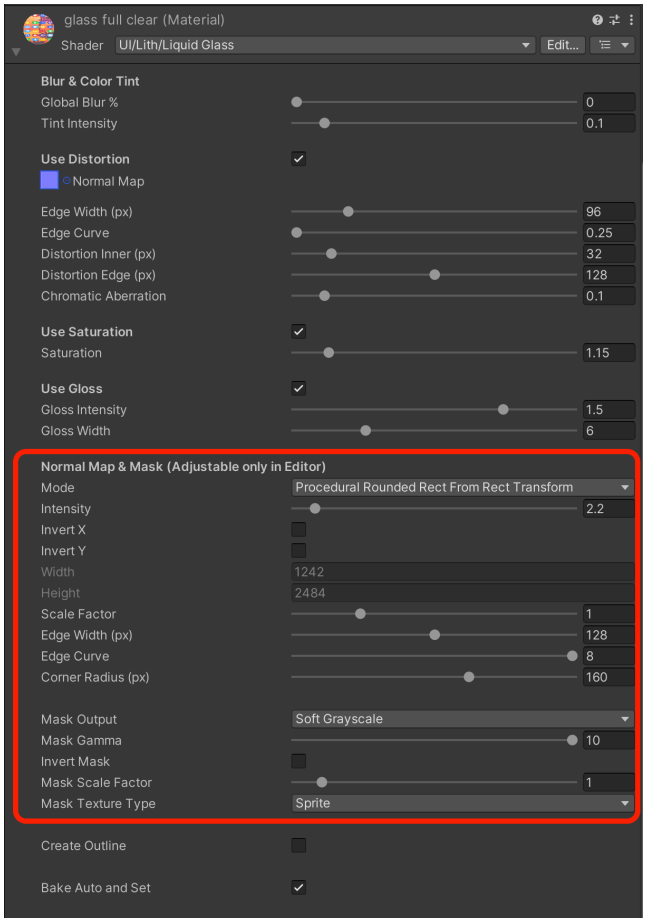
⚠️ This setup is required for the blur and refraction to appear correctly.
If both the captured content and the glass element are on the same Screen Space - Overlay canvas, the shader will not have access to the rendered background, and the effect will appear flat or transparent.

4. Normal Map Baker

The **Normal Map Baker** is essential. Liquid glass design relies on normals for curvature and depth. Without them, the effect looks flat.

Modes

- **FromSprite** → Generate normals directly from a sprite.
  - Parameters: blurPx (soften), invert X/Y.
- **Procedural RoundedRect (Custom)** → Generate beveled rectangles with adjustable corners.
  - Parameters: edgeWidthPx, edgeCurve, cornerRadiusPx.
  - Toggle individual edges (top, bottom, left, right).
- **Procedural RoundedRect (From RectTransform)** → Automatically reads the size of the selected UI element's RectTransform.
  - You can adjust Scale to increase or decrease texture resolution proportionally.



Mask Output

- **None** → no mask.
- **Soft Grayscale** → recommended for glossy glass.
- **Hard Binary** → sharp edges, less realistic gloss.
- Options: maskGamma, invertMask, hardThreshold, outlineThicknessPx.

Outputs:

- Normal Map → import as **Normal map** (non-sRGB, no mipmaps).
- Mask Texture → import as **Sprite** (UI Image).

5. Pipeline Setup

Built-in Render Pipeline

- Add **LiquidGlassCapture** component to your Camera.
- Configure downsample, blur iterations, and blur offset in Inspector.
- **Source Mode:**
  - *Camera* → capture the current camera output.
  - *RenderTexture* → assign your own RT as input (optional, advanced use).

URP

- On import, popup will auto-add **LiquidGlassRenderFeature**.
- If disabled:
  - Edit your URP Renderer asset → Add Render Feature → Liquid Glass.
  - Adjust parameters: injectionPoint, downsample, blurDownsample, blurIterations.

HDRP

- On import, popup will auto-add **LiquidGlassCustomPass**.
- If disabled:
  1. Create an empty GameObject.
  2. Add **Custom Pass Volume** component.
  3. Enable *Is Global*.
  4. Click **Add Custom Pass** → choose *Liquid Glass Pass*.

5. Configure parameters (downsample, blurDownsample, blurIterations, blurOffset).

⚠️ **Important:** HDRP requires a Custom Pass Volume in **every scene** where you want Liquid Glass to work. If you switch to a new scene, add it again.

6. Shader Parameters

In the Material Inspector, parameters appear dynamically depending on which modules are enabled.

Blur & Tint

- **Global Blur %** — overall blur strength.
- **Tint Intensity** — optional color tint overlay.

Distortion (requires Normal Map)

- **Edge Width / Curve**
- **Distortion Inner / Edge (px)**
- **Chromatic Aberration** (3-sample by default).

Gloss (requires Soft Mask)

- **Gloss Intensity**
- **Gloss Width**

Saturation

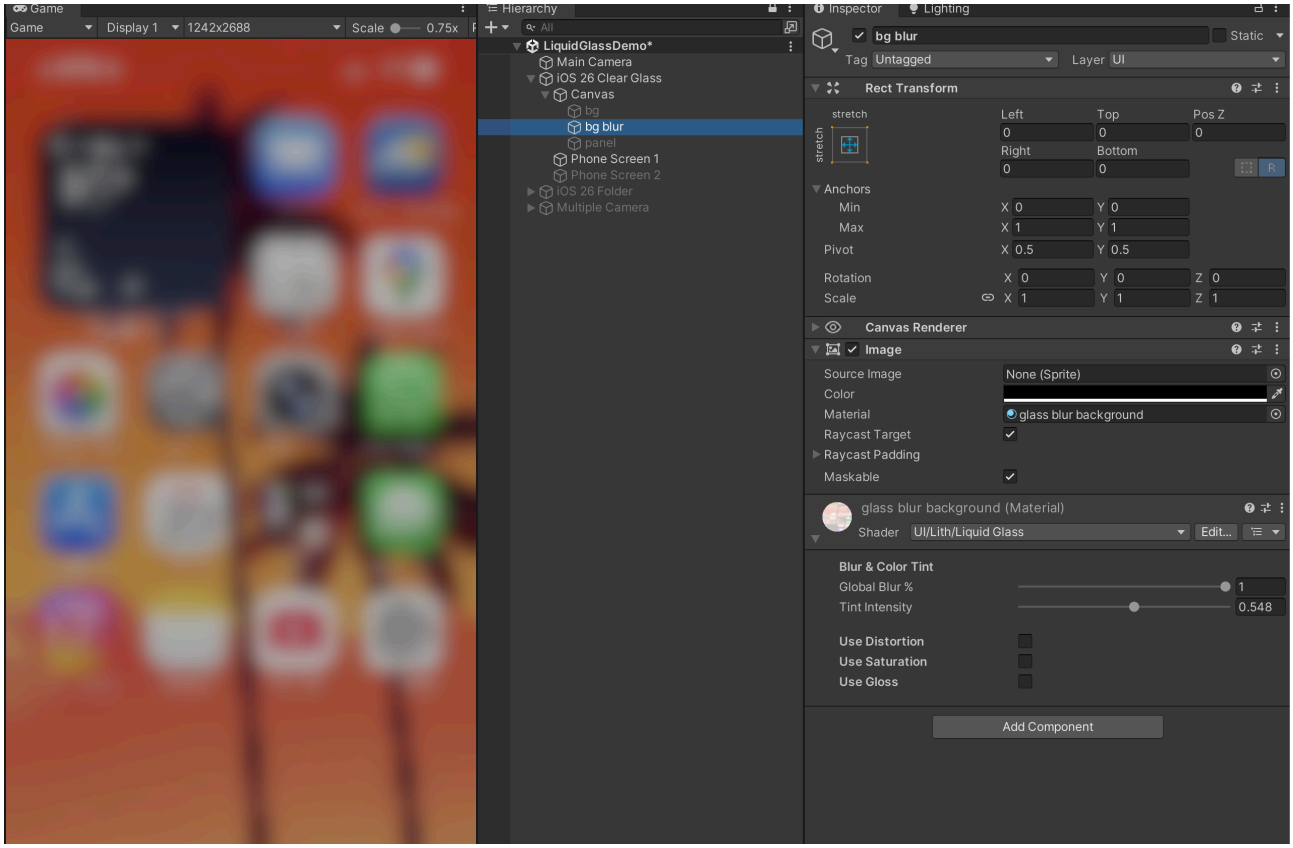
- **Saturation** multiplier for vivid glass effect.

7. Performance Considerations

- Single **dual-pass Kawase blur** only.
- Parameters affecting performance:
  - blurDownsample (keep high for faster results).
  - blurIterations (1–3 recommended for mobile).
  - downsample (texture res before blur).
- Real-world test: **1242×2688 reference resolution**, runs **60 FPS on iPhone 6s**.

Multiple UI Elements

- **Blur is computed once per frame for the entire screen.**
  - You can safely use many Liquid Glass UI objects without performance loss from blur.



⚠️ Distortion, Gloss, Saturation → still evaluated per-object, so enabling them heavily on many elements may affect performance.

8. Multiple Materials & Capture Sources

- **Different UI sizes = different Normal Maps.**
  A 512×128 bar and a 256×256 button should each have their own Normal Map + Material (or reuse the same shader via **MaterialPropertyBlock** if you know what you're doing).

Global vs Local application (critical)

- **Global mode (no assignments):** If targetMaterials is **empty**, the capture writes to **global shader textures**. **All** Liquid Glass materials in the scene will read this capture automatically.
- **Local mode (explicit assignments):** If targetMaterials contains one or more materials, the capture updates **only those materials**. Other materials are not affected.

Capture usage per camera

- When the source is a **Camera**, add **one LiquidGlassCapture per camera** you want to use as a source.
- Each LiquidGlassCapture updates **only** the materials listed in its own Inspector (Local mode), or **all** materials (Global mode) if the list is empty.

Multi-camera examples

- **Camera A → Local mode:** Renders a special UI/overlay layer and updates just the menu materials you assign.
- **Camera B → Global mode:** Renders the main scene and feeds all remaining Liquid Glass materials by writing to global shader textures.

⚠️ Warning: Multiple captures will increase GPU load. Use sparingly and only when separate views are necessary.

⚠️ Note: In the URP Render Graph pipeline, multiple materials and multi-camera support are currently not available. These features will be re-enabled in a future update.

9. Known Limitations

- UI-only (no 3D mesh support yet).
- Normal Maps required — flat without them.
- Sliced Sprites not recommended — scaling breaks normal map alignment.
- Layering not supported — overlapping Liquid Glass UI elements don’t stack.

10. Future Roadmap

- Potential optimization presets for ultra-low-end devices.

11. Credits

Developed by **Lith Games**

12. Support & Contact

For questions, bug reports, or feature requests please contact us:
info@lithgames.com

We’ll do our best to respond as quickly as possible.