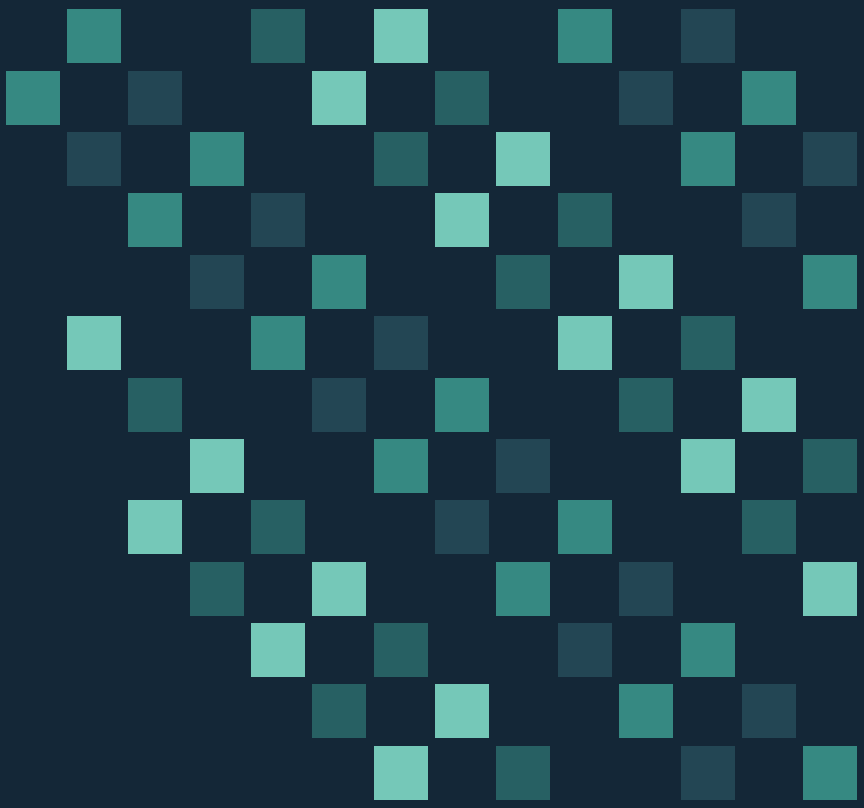
UNITY URP / USER GUIDE



PixelStudio

From first setup to finished sprites

A practical guide to pixelation, material styles, palettes, outlines, particles and 2D export.

Reference environment: Unity 6.3 / URP 17.3

English edition | Source snapshot: 10 November 2026

Setup instructions, controls, practical recipes and troubleshooting

Contents

Choose a chapter. Page numbers and bookmarks are clickable.

Start here 03
Quick start 04
Connect the camera 05
Tune the pixel grid 06
Build a lit material 07
Convert existing materials 08
Create and share Styles 09
Choose color behavior 10
Author palettes and ramps 11
Use Palette Builder 12
Control dithering 13
Anchor the pattern 14
Shape artistic highlights 15
Choose transparency behavior 16
Add per-object outlines 17
Style Particle Systems 18
Explore Style Lab 19
Start with Shader Graph 20
Wire a custom surface 21
Graph transparency & VFX 22
Export a static sprite 23
Export animated sprites 24
Batch capture & presets 25
Use settings at runtime 26
Troubleshoot rendering 27
Troubleshoot tools & graphs 28
Menus, files & references 29

Start here

Understand the layers of the PixelStudio workflow.

PixelStudio combines camera pixelation, material styling and editor tools for creating sprites. Each layer has a different job; configuring one does not automatically enable the others.

Layer	Responsibility
Camera	PixelStudioMainCamera defines the logical grid. A URP Full Screen Pass uses PixelStudioScreenPixelation to pixelate the rendered image.
Material + Style	PixelStudioLit owns textures, surface settings and lighting response. A PixelStudioStyle owns color processing, dithering and artistic highlights.
Object	Outline Target selects renderers for outlines. Dither Anchor Override supplies local pattern settings without creating another Style.
Editor tools	Style Lab compares looks. Palette Builder creates color libraries. Sprite Generator captures models into 2D assets.

Unity compatibility

Declared supported Editor versions: 6000.5.0f1, 6000.5.7f1, 6000.6.3f1 and 6000.6.21f1. Use the Universal Render Pipeline (URP).

Original reference environment: Unity 6000.3.21f1, URP 17.3.0 and Shader Graph 17.3. The versions above extend the declared support list; this documentation update does not represent fresh testing of each Editor version, render path or target platform.

Before importing

1. Import one complete **PixelStudio** folder under **Assets**. Preserve its subfolders and .meta files. Do not import a second nested copy or an older release alongside it.

2. Keep your own materials, Styles, palettes and generated sprites in a separate content folder. Let Unity finish importing before opening the tools.

About this guide

Based on the source snapshot of 10 November 2026, including the corrected Dither Anchor Override and Style Lab camera lookup. Instructions were reviewed against code and Unity documentation; no new Play Mode or platform validation was performed for this revision. Demonstration models, textures and scenes are not supplied by this guide.

Quick start

Create one styled object before applying the workflow to a full scene.

1. Create or open a URP scene containing a camera, a lit mesh and a directional light. A cube is sufficient.

2. In Project, choose **Create** > **Material**. Name it **Stone_PixelStudio**. Set its Shader to **PixelStudio/PixelStudioLit** and assign it to the mesh.

3. Choose **Create** > **PixelStudio** > **Style**. Name it **Stone_Style**. Set **Processing Mode** = **Base Color**, **Quantization Method** = **Hybrid** and **Perceptual Tone Levels** = **6**.

4. Set **Method** = **Bayer4x4**, **Anchor Mode** = **Object Space**, **Cells Per Unit** = **12** and **Strength** = **0.35**. These are suggested starting values, not a required preset.

5. Select the material and assign Stone_Style to its **Style** field. Set the Base Map or Base Color. Rotate the scene light to inspect the shading.

6. Follow the camera setup on the next page. Start with **Logical Width** = **320** and **Average Four Samples** disabled.

7. For a contour, add the outline Renderer Feature and an **Outline Target** to the mesh as described on page 17. Save the scene and assets.

Expected result

Changing the Style changes the mesh color treatment. Changing the camera grid changes the size of the final image pixels. Changing Outline Target affects the selected object contour.

Keep experiments independent
Duplicate a Style before trying a different look. Editing a shared Style affects all materials linked to it. Processing Mode Original bypasses its color processing, but does not disable camera pixelation; optional stylized specular can still be enabled.

Connect the camera

The camera component and Full Screen Pass must reference the same material.

1. Find the URP Pipeline Asset used by your active quality level. Check **Edit** > **Project Settings** > **Graphics** and any override under **Quality**. Select the **Renderer Data** used by your camera; a camera-specific Renderer override takes precedence over the pipeline default.

2. Create a material using **PixelStudio/PixelStudioScreenPixelation**. Name it **PixelStudio_Screen**. This fullscreen material is not assigned to scene meshes.

3. In Renderer Data, add a **Full Screen Pass Renderer Feature**. Assign PixelStudio_Screen to **Pass Material**.

4. For a crisp final grid, start with **Injection Point** = **After Rendering Post Processing**, **Fetch Color Buffer** = **On**, **Requirements** = **None**, **Pass Index** = **0** and **Bind Depth-Stencil** = **Off**. The shader reads the fetched camera color.

5. Select the actual rendering camera in the Hierarchy, usually Main Camera. Choose **Add Component** > **PixelStudio** > **Main Camera Pixel Grid**.

6. Assign the same PixelStudio_Screen material to **Full Screen Pass Material**. Enable **Full Screen Pixelation** and select the grid mode. Save the scene and Renderer Data.

If nothing changes
The camera script alone does not install a render pass. Check the active Renderer, the assigned material in both places and the feature enable checkbox. Avoid duplicate fullscreen passes.

Post-processing order

After Rendering Post Processing pixelates effects already present in the image. Before Rendering Post Processing allows later bloom or other effects to soften the result. Overlay UI rendered afterward is outside this pass. See the official Full Screen Pass reference on page 29.

Tune the pixel grid

Use logical resolution for an artistic target; use pixel size for screen-sized blocks.

Control	Use
Logical Width	Sets a horizontal grid from 16 to 2048. Height follows camera aspect ratio. A 320-wide grid at 16:9 is approximately 320 x 180.
Pixel Size	Requests blocks from 1 to 64 screen pixels. Effective logical width is calculated from camera resolution and bounded by the logical-grid limits.
Full Screen Pixelation	Enables image resampling. The component can still supply a logical grid to styles and outlines when this is disabled.
Preview Scene View	Allows Scene View and relevant editor preview cameras to use the controller grid.
Average Four Samples	Blends four samples per cell for a softer result. Disable for the sharpest output and strict palette work.
Sampling Softness	Controls the amount of averaging when Average Four Samples is enabled.

Practical workflow

1. Set the Game view to your intended display aspect ratio. Choose Logical Width and start at 320 or 480.

2. Inspect thin shapes, foliage and moving objects. Increase logical width if important details disappear.

3. Try softer sampling after choosing the grid. Compare at actual display size instead of relying on a scaled editor thumbnail.

Movement and multiple cameras
Orthographic and perspective cameras can use this effect. No camera snapping or render stabilization is included. Zoom, rotation and movement can change sampled pixels. Configure each gameplay camera deliberately. If several controllers preview Scene View, enable that option only on the one intended for the preview.

Build a lit material

Textures and surface response belong to the material, not the shared Style.

1. Create a **PixelStudio/PixelStudioLit** material, assign it to a renderer and choose a Style.

2. Open the material foldouts and assign maps according to their actual channel layout. Check the result under a white light before adding artistic color changes.

Control	Use
ALBEDO - BASE COLOR	Base Map and tint supply source color. Surface Mode, opacity, cutout, shadow and culling options are also here.
METALLIC + SMOOTHNESS	Metallic Map uses red for metallic and alpha for smoothness. Smoothness Source can use Metallic Map Alpha or Albedo Alpha.
NORMAL MAP	Import the texture as a Normal map in Unity, then assign it and adjust Normal Strength.
EMISSION MAP	Assign the emission texture and HDR color. Emission does not create a Unity Light; bloom is a separate URP effect.
DETAIL MAP	Adds albedo detail. Neutral gray is linear 0.5. Turn sRGB off for an authored gray data texture so it stays neutral in Linear color space.
ARTISTIC LIGHTING / VERTEX COLORS [BETA]	Vertex Color Influence controls vertex coloring. Ambient Tint Influence and Ambient Tint add artistic tint control. This section is explicitly beta.

Texture preparation

Use color import settings for color maps, Normal map import for normals, and linear data settings for masks. Opacity and highlight masks use red. A cubemap cannot be assigned to a 2D Base Map slot.

No Style assigned
The shader renders its PixelStudio PBR appearance without artistic quantization. It is not an exact replacement for every URP/Lit feature. Multi-selection uses the dedicated batch-Style Inspector.

Convert existing materials

Transfer recognizable properties without rebuilding custom shader logic.

One material

Duplicate the material first if you need the original. Switch its Shader to **PixelStudio/PixelStudioLit**. The Inspector transfers recognized maps, colors, tiling and offsets. Assign a Style afterward and review the surface settings.

A batch of materials

1. Select editable **.mat** assets in Project. Open **Tools** > **PixelStudio** > **Material Converter**.

2. Click **Refresh from Project selection** if needed. Inspect the list and warnings. Optionally assign **Style to assign (optional)**.

3. Choose **Convert ... material(s) to PixelStudio** and confirm. This edits the selected materials in place and does not create backups.

4. Review Base Map, Normal, Metallic/Smoothness, Emission, Detail and specular mask assignments. Check opacity, smoothness source and culling.

What is not transferred
Custom wind, SSS logic, refraction and Shader Graph networks are not reconstructed. ORM, metallic-roughness and specular-workflow maps are not blindly reinterpreted as metallic-red/smoothness-alpha. Texture files remain in the project, but unsupported references can be disconnected.

Repair incompatible texture dimensions

1. Select the affected materials in Project. Run **Tools** > **PixelStudio** > **Repair Selected Material Textures**.

2. The tool clears incompatible assignments without deleting texture assets or changing the shader. Assign valid 2D maps. Keep skybox materials on an appropriate Skybox shader.

Create and share Styles

Separate reusable art direction from each material texture set.

1. Choose **Create** > **PixelStudio** > **Style**. Name the asset for its purpose, such as Forest_Cool.

2. Assign it to the **Style** field of a native PixelStudioLit or PixelStudioParticles material. Shader Graph uses the bake workflow on page 20.

3. Choose Processing Mode, edit its controls and inspect representative materials under different lights.

Control	Use
Original	Bypasses palette, quantization, color adjustments and color dithering. Camera pixelation remains separate. Optional stylized specular is available.
Base Color	Reduces tones from the material lit color without requiring a Palette asset.
Palette	Maps output to swatches in a PixelStudioPalette. Requires a Palette Asset.
Ramps	Uses ordered dark-to-light colors from the Palette Asset ramp groups.

Apply a Style to several materials

Select one Style and compatible materials together. Run **Assets** > **PixelStudio** > **Apply Selected Style to Materials**. Alternatively, select several compatible materials and use the batch-Style Inspector. This changes the Style link while retaining textures and PBR settings.

Changes are shared
Editor Style edits propagate to linked materials. Duplicate the Style for an independent variant. If a Style was deleted, old values may remain baked in a material: reassign or explicitly clear its Style link instead of assuming its appearance reset.

Choose color behavior

Pick the processing mode before tuning tonal detail and color matching.

Control	Use
RGB Classic	Quantizes RGB channels separately. RGB Levels per Channel does not mean that the entire image has only that many colors.
Perceptual Lightness	Reduces perceptual tone levels while retaining hue more naturally.
Hybrid	Adds Chroma Levels and Chroma Reduction to perceptual tones.
Tone Distribution	Linear, Shadow Focus, Midtone Focus or Highlight Focus concentrates tonal steps in different parts of the range.
Color Matching	RGB compares channels; OKLab uses perceptual matching. Compare both with your palette.
Palette Output	Strict Before Post Processing constrains shader output to swatches and disables its hue shift and fog. Artistic permits the artistic treatment.
Color Adjustments	Saturation, Contrast and Hue Degrees alter the color treatment. Original mode bypasses these adjustments.

A starting recipe

Try Base Color / Hybrid, 6 Perceptual Tone Levels, 4 Chroma Levels and 0.35 Chroma Reduction. Use modest dithering and adjust the lighting before increasing contrast. These are suggestions, not shipped presets.

Strict palette is local to the shader
Transparency blending, antialiasing, fullscreen averaging and later post-processing can introduce new displayed colors. For the closest swatch match, use opaque materials, disable Average Four Samples and compare without color grading or bloom.

Author palettes and ramps

A Palette asset stores independent swatches and ordered ramp groups.

1. Choose **Create** > **PixelStudio** > **Palette**. Expand **Palette Colors** or **Color Ramps** in its Inspector.

2. Enter up to 16 palette swatches. For ramps, create up to four groups with 2-8 shades each, ordered from darkest to brightest.

3. Select the Style, choose Palette or Ramps and assign the Palette Asset. A missing library falls back to lit color rather than an arbitrary palette.

Use a specific ramp

With multiple groups, choose **Ramp Selection** = **Fixed** and set **Ramp Index (0-based)**: 0 means the first group. Materials using that Style share the selected group. Duplicate the Style to select another ramp independently.

Choose a ramp automatically

Choose **Ramp Selection** = **Automatic** and **Automatic Ramp Matching**. Compare differently colored surfaces. The shader selects from your ramp groups; it does not create additional ramps.

Control	Use
Tone Black Point	Lower bound for remapping lighting into the ramp.
Tone White Point	Upper remapping bound. Keep it above the black point.
Perceptual Ramp Lightness (OKLab)	Uses perceptual lightness for ramp tone evaluation.
Tone Distribution	Redistributes transitions among shadows, midtones and highlights.

Respect the limits
Only the first 16 swatches and first four ramp groups are used. Additional Inspector entries do not expand shader capacity. Correct dark-to-light ordering warnings before judging the appearance.

Use Palette Builder

Extract an image palette or import a color file into an existing Palette asset.

Extract from Image

1. Open **Tools** > **PixelStudio** > **Palette Builder**, or click **Open Palette Builder (Extract / Import)** in the Palette Inspector.

2. Assign an existing **Destination Palette**. Select **Extract from Image** and assign **Source Image**.

3. Set **Colors** between 2 and 16 and choose **Extract Palette Colors**. Inspect and refine the resulting swatches.

Import Existing Palette

1. Select **Import Existing Palette** and choose **Open HEX / GPL / JASC-PAL / PNG File**.

2. Choose **Import Into**: Palette or a ramp destination. Select the colors you want to keep.

3. Use 1-16 colors for palettes or 2-8 for ramps. Source order is preserved, so arrange ramp shades from dark to light.

4. Choose **Import ... Selected Colors**, then check the Style references the intended Palette asset.

Format	Notes
HEX / GPL	Reads supported hexadecimal lists or GIMP palette color entries.
PAL	Accepts JASC-PAL text. Adobe and RIFF PAL formats are not supported.
PNG	Reads distinct visible colors, listing up to the first 256 unique colors. Use a swatch image for precise selection.

Extraction versus import
Image extraction reduces a picture to representative colors. PNG palette import reads existing discrete colors. Neither creates a finished material or a library of presets.

Control dithering

Dithering distributes color transitions; it does not supply transparency.

Method	Behavior
Bayer 2x2 / 4x4 / 8x8 / 16x16	Ordered patterns with increasingly large tiles. Compare them at your final grid.
Clustered 4x4 / 8x8	Grouped patterns for a more pronounced printed or stippled appearance.
Horizontal Lines	Directional stripe thresholds.
Hash Noise / Blue Noise 64	Noise-based thresholds. Blue Noise uses the supplied shared texture.
Custom Pattern / None	Your threshold texture, or no color dithering.

Adjust the transition

Strength changes the contribution. **Dither Coverage Width** changes the breadth of the tonal transition. Choose **Dither Zone** for midtones, shadows plus midtones or the full range. **Zone Feather** softens zonal boundaries where available.

Protect Palette Colors and **Protect Discrete Tones** help settled colors stay solid. With **Only Dither on Lighting Transitions**, adjust **Minimum Lighting Variation** and **Transition Softness** to reduce patterning in uniformly lit areas.

Import a custom threshold tile

1. Prepare a grayscale tile. Use **Point** filtering, **Repeat** wrap, no mipmaps and linear data sampling (sRGB off).

2. Choose Custom Pattern, assign **Custom Threshold Texture** and set **Pattern Tile Size (pixels)** to its authored size within the 2-64 range.

> **Not alpha coverage**
> Dither Coverage Width is about color transitions. The discontinued dithered-transparency system is not part of this workflow. Use Surface Mode and opacity for transparency.

Anchor the pattern

Choose coordinates that fit the expected object motion.

Anchor Mode	Behavior
Screen Space	Uses the camera logical grid. The pattern can slide over surfaces as the camera or object moves. Cells Per Unit does not control this mode.
World Space	Anchored in the scene. A moving mesh passes through the world pattern.
Object Space	Follows the object local frame. Useful for rigid props; not a universal solution for deformation.
UV	Follows mesh UVs. Density means cells per UV unit; UV scale and seams matter.

Set a shared anchor

Select the Style and choose **Anchor Mode** under **DITHERING ANCHOR**. For spatial or UV anchors, adjust **Cells Per Unit**; start at 12. Move the camera and object to assess the pattern in motion.

Override an object through Add Component

1. Select the **GameObject in the Hierarchy**. In its **Inspector**, click **Add Component** and search for **Dither Anchor Override**. Select it to add the script.

The menu category is **Add Component** > **PixelStudio** > **Dithering** > **Dither Anchor Override**. This is inside Add Component, not a separate top-level Tools menu.

2. Choose **Anchor Mode** and **Cells Per Unit**. Enable **Include Children** when the intended renderers are on child objects; otherwise only renderers on this GameObject are affected.

3. Use a Style with visible dithering. Compare **Object Space** at 4 and 24 cells per unit. A Style with dithering disabled will not show an anchor change.

4. Disable the component to inherit the Style again. If an enabled ancestor override includes this object, that ancestor takes over instead. Prefer one override per intended hierarchy; the nearest eligible override wins when nested.

Using a Shader Graph material

The supplied template already exposes the two required properties. For a custom Graph, connect them to **PixelStudioSurface** as described on **page 20**. Adding the component alone cannot modify a Graph with unconnected inputs or different property References.

Stability limits

This component changes dithering coordinates, not camera pixel size, geometry or textures. It does not eliminate temporal aliasing or pixel creep. Fullscreen resampling, zoom and rotation remain independent; the package does not snap or stabilize the camera.

Shape artistic highlights

Enable highlights in the Style; tune their surface response per material.

1. Enable **Enable Stylized Specular** on the Style. Start with Sharp, modest Intensity and a neutral Highlight Color.

2. Adjust **Highlight Size** and **Edge Softness** while moving a light or camera. Soft gives gentler edges; Dithered uses the Style threshold pattern.

3. Choose **Palette Handling**. Palette Constrained passes highlights through color processing. Artistic Highlight adds its HDR color afterward. Strict palette output still takes priority in Palette mode.

4. On the material, open **STYLIZED SPECULAR - MATERIAL RESPONSE**. Adjust Response Multiplier, Surface Highlight Tint and Highlight Mask.

Control	Use
Highlight Color / Intensity	Shared highlight color and strength.
Metallic Influence	How strongly surface metallic affects the artistic highlight.
Smoothness Influence	How strongly smoothness contributes.
Response Multiplier	Per-material strength for surfaces sharing a Style.
Highlight Mask (Red)	White permits highlights; black suppresses them. Import with sRGB off.

Keep PBR and artistic response distinct

Metallic and smoothness still affect normal lighting. The stylized highlight is an additional artistic control. Original mode can use it even when other Style processing is bypassed.

Dithered highlights
Select an actual dithering method for a patterned highlight. Inspect at the final camera grid: a small highlight may become only one or two visible pixels.

Choose transparency behavior

Use a valid alpha or mask; black RGB is not automatically transparent.

Control	Use
Opaque	Solid surfaces without transparency blending.
Cutout	Binary visibility controlled by Alpha Cutoff. Use for leaves, fences and sharp holes.
Alpha	Soft transparency using texture alpha or an explicit opacity mask.
Premultiplied	The shader computes premultiplied output once. Provide straight RGB source textures.
Additive	Adds light-like color. Additive Black Fade Range suppresses blank source pixels before palette mapping.
Glass	Premultiplied transparency with view-dependent edge opacity.

Set up an alpha material

1. Open **ALBEDO - BASE COLOR** and choose Surface Mode.

2. For RGBA textures choose **Alpha Source = BaseMapAlpha**. For a separate mask choose MaskOnly and assign Opacity Mask; its red channel shares Base Map UVs.

3. Set Base Color alpha and Opacity. These multiply the alpha contribution; a zero value can hide the surface.

4. For Cutout, adjust Alpha Cutoff. For blended surfaces, review Sorting Offset only when needed. It does not solve every intersecting-transparency problem.

Glass recipe and shadows
Start with Glass, Opacity 0.12, Edge Opacity 0.65, Fresnel Power 4, Metallic 0 and Smoothness 0.85. Mode changes do not reset these values. Enable Receive Shadows and renderer transparent-shadow reception when needed. Cast Shadows offers Off, Solid or AlphaClip, with Shadow Alpha Cutoff for clipping. There is no refraction or colored/partial shadow transmission.

Add per-object outlines

Configure one Renderer Feature, then choose which objects receive contours.

1. On the camera Renderer Data, add **PixelStudioOutlineRendererFeature** once. Keep Render Graph enabled; this implementation uses that path.

2. If camera-color access fails, use **Intermediate Texture = Always**. Check Mask Shader and Composite Shader: the provided assets are PixelStudioOutlineMask and PixelStudioOutlineComposite.

3. Add **PixelStudio > Outline Target** to the mesh or root. Enable Outline Enabled and select Outline Color and Logical Pixel Width (1-4).

4. Enable Include Children for multi-part models. Check Game view and, if enabled on the feature, Scene View.

Control	Use
Automatic group	Automatically assigns the target outline group.
Manual group / Group Id	Use the same group for parts treated as one outline group. Different groups preserve object separation.
Detail Outline	Adds internal lines from depth and normal differences.
Detail Depth / Normal Threshold	Controls the discontinuity needed to create a detail line.
Detail Outline Color	Internal line color, separate from the outer contour.

Supported surfaces and cost
Opaque and alpha-clipped surfaces are supported. Blended transparent materials, glass and ParticleSystemRenderer are skipped. Graphs require the supported integration contract. Each selected submesh adds rendering work. Up to 254 groups are available per camera; Graph style/group combinations are also bounded.

Save Renderer Data with its shader references for builds. An editor preview alone does not establish compatibility with every platform or shader-stripping configuration.

Style Particle Systems

PixelStudioParticles is unlit, with explicit blend and alpha controls.

1. Create a **PixelStudio/PixelStudioParticles** material. Assign a Style and Base Map. Choose Alpha, Additive or Premultiplied particle blending.

2. Assign it in the Particle System Renderer module. Configure **Custom Vertex Streams**: Position, Normal, Color, UV, UV2 and AnimBlend, in that order. Disable GPU mesh instancing.

3. Check alpha in Start Color and Color over Lifetime. The shader uses vertex color and alpha per particle.

4. For animated sheets, configure Texture Sheet Animation and its blending data. Enable material flipbook blending and set Flipbook Grid to the atlas columns and rows.

5. Use the particle UV anchor to make the pattern follow a billboard. Adjust Style density and inspect animated frames for seams. Choose particle fog only when desired.

Effect	Starting point
Smoke and translucent debris	Alpha or Premultiplied with meaningful alpha or a red-channel mask.
Sparks and light streaks	Additive, with Additive Black Fade Range tuned to keep blank areas empty.
Hard-edged shapes	Optional alpha clip with a suitable Alpha Cutoff.

Black rectangles
RGB artwork may have fully opaque alpha. Alpha blending then draws the whole quad. Supply a real mask or choose Additive for suitable light-like artwork. Opaque geometry outlines are not applied to particle billboards.

The native particle shader does not reproduce PixelStudioLit PBR lighting or shadows. VFX Graph requires the separate manual workflow on page 22.

Explore Style Lab

Compare looks using an isolated preview and a three-light setup.

1. Open **Tools** > **PixelStudio** > **Style Lab**. Choose Original x Style and assign a Style, or Style x Style and assign Left Style and Right Style.

2. Assign Optional sample prefab or choose Sample geometry. Drag to rotate, scroll to zoom and double-click to reset. Use Refresh comparison after changing source content.

3. In PIXELATION PREVIEW, assign a Reference Camera containing PixelStudioMainCamera. Use Read Camera to copy its grid to the preview.

4. Adjust preview Grid mode, Logical width or Pixel size. Toggle pixelation and Four-sample averaging to compare readability.

5. Expand THREE-POINT LIGHTING (KEY / FILL / RIM). Set intensities and angles. URP Additional Lights must be enabled for Fill and Rim to contribute.

Control	Effect
Preview camera/grid	Local to Style Lab. Does not change the scene camera or Sprite Generator settings.
Key / Fill / Rim	Local preview lights, not scene light edits.
EDIT LEFT STYLE	Edits the actual Style asset. Linked materials can update.
Original comparison	Shows the source-side appearance; it does not restore a material previously converted in-place.

Explore safely
Duplicate a production Style first. Preview isolation protects the scene setup; it does not make the shared Style Inspector a disposable preview.

Start with Shader Graph

Use the supplied template, or expose the override inputs manually.

Prepare the Graph and Style

1. Duplicate **PixelStudio/ShaderGraph/PixelStudioShaderGraphTemplate.shadergraph** into your content folder. The supplied original remains unchanged. PixelStudioSurface is already connected to Base Color, Alpha and Alpha Clip Threshold.

2. Create a material from the duplicate. Configure its maps and surface inputs. The template starts opaque with alpha clipping; disable clipping if it is unnecessary.

3. Select a saved **PixelStudioStyle** in Project, then run **Assets > PixelStudio > Bake Style for Shader Graph and VFX**. Expand the Style asset and assign its **Graph Data** texture to the material Style Data input. Assign Blue Noise and Custom Pattern when used.

Connect per-object overrides in a custom Graph

In the Blackboard, add two **Float** properties (Vector 1 in some versions). Select each property and set its **Reference** exactly as below. Keep **Exposed** enabled and use the normal per-material declaration, not Global or a shader keyword. The supplied template already contains these properties.

Property / Reference	Default	PixelStudioSurface input
Anchor Override _SP_GraphAnchorOverride	-1	Anchor Override
Density Override _SP_GraphDensityOverride	0	Density Override

Drag both Blackboard properties onto the canvas. Connect each property output to its matching input on **PixelStudioSurface**, replacing any constants connected there. Save the Graph and assign its material to the object renderer.

Keep the **material values** at **Anchor Override = -1** and **Density Override = 0** to inherit the Style. Existing materials may retain older values even after you change the Graph defaults.

Now add **Dither Anchor Override** through the object Inspector (page 14). It writes per-renderer values using a MaterialPropertyBlock. The shared material and its Style are not edited. Disabling it restores Style inheritance, unless an eligible ancestor override is active.

Manual material values: Anchor -1 = Style, 0 = Screen Space, 1 = World Space, 2 = Object Space, 3 = UV. Density 0 = Style; a positive value overrides spatial/UV density. Screen Space uses the camera grid.

If the override has no visible effect

Check that the updated script is imported, the correct material is assigned, the property References match exactly and both properties reach the node. Enable visible dithering in the baked Style. Another script writing a property block can overwrite these values. If the baked look is stale, select the Style and bake it again.

A disabled bake command usually means the selected asset is not a saved PixelStudioStyle. Graph Data is a persistent dependency. The template appears under PixelStudio/Templates; replacing a custom Graph with PixelStudioLit would discard its custom shader logic.

Wire a custom surface

Calculate the surface before PixelStudioSurface; connect its outputs directly.

Existing data	Connection
Albedo / color result	Connect to the corresponding surface color input of PixelStudioSurface.
Normal / Metallic / Smoothness / Occlusion	Connect each channel using the space and range expected by the template.
Emission / authored SSS	Pass the authored contribution into the corresponding input. The node does not create the original SSS effect.
Alpha / clipping threshold	Pass through the node to the matching Fragment blocks.
PixelStudioSurface Color	Connect directly to Fragment Base Color in the supplied Unlit target. Do not light it a second time.
Wind / vertex deformation	Keep the effect in Vertex, preserving its Position, Normal and Tangent changes.

Target configuration

The manual equivalent uses **Universal / Unlit**, **Keep Lighting Variants** enabled, **Default SSAO** and **Default Decal Blending** disabled, and **Allow Material Override** disabled. Outlines require opaque depth-writing surfaces with Depth Test LEqual. Starting from the template avoids reconstructing this setup.

Preserve Texture2D references **_SP_GraphStyleData**, **_SP_BlueNoiseTex** and **_SP_CustomDitherTex**. The hidden **_SP_GraphOutlineSupport** marker is only valid for Graphs that preserve the complete integration contract.

> **Keep the final Color connection direct**
> Do not add Multiply, Lerp, Clamp or other operations after Color in outline-enabled Graphs: this can corrupt outline data. Put those calculations before the node. Keep renderer bounds large enough for vertex displacement.

Add Outline Target and the Renderer Feature normally. Internal detail lines depend on the camera normal buffer; Unlit geometric normals do not automatically include all normal-map detail.

Graph transparency & VFX

Advanced manual integration, not a ready-made VFX output.

Transparent Shader Graph

1. Duplicate the Graph. Choose **Surface** = **Transparent** and start with **Blend** = **Alpha**. Disable Alpha Clipping, Depth Write and Cast Shadows for this starting setup.

2. Build alpha from texture alpha, tint alpha, opacity and a mask. PixelStudioSurface passes Alpha through; it does not choose the Graph blend mode.

3. Optionally add a Float Custom Function in File mode using **Shaders/PixelStudioAlpha.hlsl**, function **PixelStudioOpacity**. Inputs: TextureAlpha, TintAlpha, VertexAlpha, Mask, Opacity (floats). Output: Alpha.

4. For additive art, **PixelStudioAdditiveSupport** takes SourceRGB before palette mapping and FadeRange. Multiply Support into opacity so empty source pixels stay empty.

VFX Graph boundary

The package provides baked Style data and a color-processing HLSL adapter, not a complete VFX Graph asset. Use a URP-compatible Shader Graph accepted by your VFX Output and integrate processing in its fragment path. VFX remains responsible for per-particle alpha, blend, depth and output geometry.

The lower-level Custom Function is **PixelStudioColor**, File mode, Float precision, from **Shaders/PixelStudioShaderGraph.hlsl**. Inputs are StyleData, BlueNoise and CustomPattern textures; Color, Albedo, Emission, PositionWS and NormalWS vectors; ScreenUV and AnchorUV; AnchorOverride and DensityOverride. Output is Out. Feed an appropriate already-lit or unlit color without a second lighting pass afterward.

Integration limits
Transparent Graphs and VFX particles do not gain opaque outlines. Avoid double premultiplication. There is no validated one-click conversion for arbitrary VFX Graphs, HDRP Graphs, 2D targets or third-party shader networks.

Export a static sprite

Capture a prefab or scene object with separate framing and studio lights.

1. Open **Tools** > **PixelStudio** > **Sprite Generator**. Assign **Source GameObject / Prefab** and select **Static Sprite**.

2. Set Rotation XYZ or use Front, 3/4 view, Side or Top. Adjust Camera zoom and Screen offset. Choose Orthographic or perspective.

3. Use Recalculate auto-fit after replacing the model. Drag to rotate and scroll to zoom in the preview.

4. Enable Pixelated icon if desired. Choose PNG output size: 64, 128, 256 or 512. Choose Logical pixel size: 32, 64, 128 or 256.

5. Enable Export at logical resolution for a smaller pixel-art image, or export an enlarged Point-sampled result. Prefer an integer enlargement ratio.

6. Configure the icon Outline, Key light and Fill light. Enable Transparent PNG or choose a background color.

7. Choose **EXPORT PNG + IMPORT AS SPRITE**. Save inside Assets and inspect the Sprite at its intended display size.

Preserve source appearance

The generator uses an isolated visual clone and reuses source meshes, materials and Styles. Assign the intended materials first; it does not automatically convert every custom shader. A prepared visual model is preferable to relying on gameplay scripts.

> **Preview and layout**
> Enlarge the window or use compact Settings and Preview tabs. The settings panel scrolls. If outlines are missing, check the Renderer Feature and use Refresh source + preview.

Inspect exported alpha and framing. An isolated camera does not make every shader or effect that depends on scene objects capturable.

Export animated sprites

Sample an AnimationClip into a consistently framed set of 2D images.

1. Assign the source and select **Animated Sprite**. Assign **Animation Clip** or use Find when the model/controller contains one.

2. Use a compatible rig. Humanoid clips need a compatible Avatar; check the sampler ready/status message.

3. Set Animation FPS from 1 to 60. Use Frames (0 = auto) to derive frame count from duration, or choose a manual count up to 256.

4. Choose Sheet columns from 1 to 16 and Pixels Per Unit. Enable Lock animator root if the sprite should stay in place.

5. Use Play, Pause and the scrubber. Check all poses for cropping; the tool uses shared animation-bounds framing.

6. Select the individual PNG, 2D AnimationClip and slicing options. Choose **EXPORT SPRITE SHEET + 2D ANIMATION** and an Assets folder.

Output	Purpose
Sprite sheet PNG	Frames in a grid. The exporter limits sheet dimensions to 8192 pixels per side.
Individual PNGs	Standalone Sprite frames. The exporter creates required frame assets when generating a 2D clip.
2D AnimationClip	Animates SpriteRenderer.m_Sprite. Use a SpriteRenderer and your own Animator setup; this is not a 3D rig clip.
Automatic slicing	Requires Unity 2D Sprite. Individual frame Sprites remain usable without that package.

What is sampled
This captures AnimationClip poses, not a running game. Scripts, IK and physics are not simulated. If frames are identical, check the clip, rig, Avatar and preview before trying again.

Batch capture & presets

Reuse capture settings across many objects and viewing angles.

Save a capture preset

1. Configure a representative object. Choose New beside Sprite Preset to create an asset.

2. Use Save after editing settings. Assign the preset and choose Load to reuse the stored setup.

3. Inspect framing for unusually small or large models; a preset does not remove that check.

Export a batch

1. Choose Batch Sprite and expand BATCH EXPORT / MULTI-ANGLE.

2. Select prefabs or objects and click Add selected prefabs / objects. Review the queue and remove unintended entries.

3. Select Angles: Front, ThreeQuarter, Side and/or Top. Choose a Folder inside Assets.

4. Enable Create contact sheet for an overview of static captures. Choose EXPORT ALL AS SPRITES.

5. For animations, enable Batch capture AnimationClip, assign a compatible clip and set animated options. Choose EXPORT ALL ANIMATION SHEETS. Review errors for incompatible rigs.

Organize the outputs

Use a dedicated destination such as **Assets/GameArt/GeneratedSprites**. Review transparency, alignment, contour thickness and scale. Contact sheets help review; the individual Sprite assets are the usable output.

> **Keep referenced frames**
> Presets are reusable authoring data. Generated AnimationClips reference their Sprite frames. Do not delete individual frame files just because a combined sheet also exists.

Use settings at runtime

Editor asset linking is separate from runtime parameter changes.

Styles and palettes are authoring assets. Native shader values and Graph Data are prepared in the editor. Editor menus, automatic link discovery and preview tools are not a runtime management system.

Switch a prepared Style

Keep a direct PixelStudioStyle reference. Call **style.ApplyTo(material)** on a compatible material to change looks. This updates properties; it does not install render features or reconstruct a Graph. Graph materials require the selected Style to already contain baked Graph Data.

Change blending correctly

Use **PixelStudioSurfaceSettings.SetSurface(material, mode)** or **SetParticleBlend(material, mode)**. These update blend, queue, depth and pass state. Setting only a numeric surface property can leave an inconsistent render configuration.

Concern	Guidance
Shared material	Changes affect every renderer that uses it. Use deliberately for group-wide switches.
Owned instance	Retain a material instance for independent object appearance and dispose of it with its owner.
Outline hierarchy	After replacing child renderers at runtime, call Outline Target RefreshRendererCache() when needed.
Anchor hierarchy	After changing targets, use Dither Anchor Override RefreshTargets() and Apply().

Bake before building
Graph baking uses UnityEditor APIs. Prepare Styles and retain their data references before a build. Changing an arbitrary ScriptableObject field in a player does not run editor synchronization.

Before shipping

Save shader references in Renderer Data and retain referenced generated assets. Verify your target platform, transparency and render path. This guide does not establish mobile, WebGL, XR or Deferred compatibility or performance guarantees.

Troubleshoot rendering

Change the smallest responsible part before modifying the entire scene.

Symptom	Action
Pixelation has no effect	Check the camera component, Full Screen Pixelation, same fullscreen material in both places, active Renderer Data and Fetch Color Buffer.
Style has no effect	Assign a Style. For Graphs, check baked data and final node wiring. Original mode intentionally bypasses most processing.
Palette looks wrong	Check mode, library, count and ramp order. Disable averaging and post-processing while evaluating strict swatches.
Patterns slide in motion	Choose an anchor suited to the surface. Screen-space patterns move relative to objects; stabilization is not included.
Glass receives no shadows	Enable Receive Shadows on the material and transparent shadow reception in the URP Renderer. Check the Light and pipeline shadow settings.
Glass casts a solid shadow	Solid is explicit. Choose Off or AlphaClip if appropriate. Colored transmission is not implemented.
Black particle rectangle	Check actual texture alpha and the mask. Use Additive for suitable artwork and tune Additive Black Fade Range.
Pink material	Read the first Console shader/compiler error. Confirm URP, package versions and complete imports before changing artistic settings.

Missing menus
Allow compilation to finish and fix compiler errors first. Integration and Setup and Check are discontinued workflows. The current entry points are listed on page 29.

Troubleshoot tools & graphs

Separate outline, preview and export requirements.

Symptom	Action
Outline missing everywhere	Check its Renderer Feature, Render Graph, shader references, camera layer mask and visible enabled Outline Target. Check Intermediate Texture for color access.
Outline missing on one surface	Blended materials and particles are excluded. Graphs require the template contract and a direct final Color connection.
Graph appears lit twice	PixelStudioSurface computes lighting for the supplied Unlit target. Do not send its result through another Lit stage.
Bake command disabled	Select a saved PixelStudioStyle, not a material, Graph or baked texture.
Style Lab lighting differs	Its Key/Fill/Rim are local preview lights. Enable URP Additional Lights for Fill/Rim.
Sprite is cropped	Recalculate auto-fit, reduce zoom and adjust Screen offset. Inspect all animated poses with shared framing.
Animation export is static	Check the AnimationClip, compatible Avatar and sampler status. Gameplay scripts, IK and physics are not evaluated.
Sheet cannot be sliced	Install Unity 2D Sprite in Package Manager for automatic slicing. Individual frames remain available.
CUBE assigned to 2D slot	Run Repair Selected Material Textures on affected materials. Reassign valid textures; keep skyboxes on Skybox shaders.

For an unresolved issue, capture the relevant Inspector, first Console error, Unity/URP versions and a minimal scene. Avoid changing several unrelated settings at once.

Menus, files & references

Return to the right tool and preserve the dependencies your content needs.

Entry point	Tools
Create > PixelStudio	Style; Palette.
Tools > PixelStudio	Style Lab; Palette Builder; Material Converter; Sprite Generator; Repair Selected Material Textures.
Assets > PixelStudio	Apply Selected Style to Materials; Bake Style for Shader Graph and VFX.
Add Component > PixelStudio	Main Camera Pixel Grid; Outline Target; Dithering > Dither Anchor Override.
Shader > PixelStudio	PixelStudioLit; PixelStudioParticles; PixelStudioScreenPixelation. Graph template under Templates.

Keep project dependencies

Keep Runtime, Editor, Shaders, ShaderGraph and Resources together with supplied .meta files. Retain your Styles, Palettes, Graph Data, materials and referenced Sprite/AnimationClip outputs. Keep Blue Noise. Old release archives and duplicate package folders are not imports you need.

Official Unity references

URP 6.3: Full Screen Pass Renderer Feature
Camera-color input, injection points and pass settings.

Shader Graph 17.3: Custom Function Node
File-based functions and precision.

Shader Graph 17.3: Sub Graph
Reusable node groups.

Documentation scope
Based on the inspected source snapshot. Labels may be spaced differently by Unity. Suggested artistic values are starting points, not bundled presets. Demonstration scenes, characters, models, textures and animations from third parties are not included with PixelStudio.