INTRODUCTION

LED Studio is a Unity package for building believable LED walls, ribbon boards, scoreboards, perimeter signage, and multi-panel venue displays. Instead of showing a flat texture on a mesh, it composes content into a logical grid of LED emitters and then renders that grid with pitch, diode shape, cabinet seams, brightness response, bloom, close-up diode detail, and optional hardware defects.

This guide starts with LED display basics, then walks through board creation, content sources, layouts, playlists, grouped panels, runtime control, render pipeline setup, and troubleshooting. It is written for users who may be new to LED systems but want production-quality results inside Unity.

CONTENTS

LED DISPLAY BASICS

A real LED display is made from many small light sources arranged in a grid. Each light source is usually called an LED, diode, emitter, or pixel. LED Studio uses the word emitter because the package is simulating the visible light point, not just storing an image pixel.

EMITTERS, PIXELS, AND LOGICAL RESOLUTION

The logical resolution is the number of emitters the board contains, for example 192 by 108. This resolution is not the same thing as the final screen resolution of your game view. LED Studio first composes all text, images, video, UI, scores, and timers into this emitter grid. The LED shader then presents the grid on a mesh so it looks like a physical LED surface.

Beginner term - If a board has a logical resolution of 512 by 16, it behaves like a long ribbon display with 512 emitters across and 16 emitters high.

PITCH AND VIEWING DISTANCE

Pitch is the physical distance between adjacent emitters, measured in millimeters. A P2.5 display has a 2.5 mm pitch, while a P10 display has a 10 mm pitch. Smaller pitch means more detail at close range. Larger pitch looks more visibly pixelated and is usually viewed from farther away.

Pitch	Typical feel	Good use
P2.5 / P3.9	Fine detail, readable closer to camera	Stage backdrops, trade show walls, hero screens
P5 / P6	Balanced detail and visible LED character	Scoreboards, venue screens, medium-distance signage
P10 / P16 / P20	Chunky, obvious diode pattern	Large outdoor boards, stylized retro displays, distant ribbons

CABINETS, SEAMS, AND BEZELS

Large LED walls are normally assembled from rectangular cabinet modules. The small gaps between cabinets create seams, and the dark structure around each module creates a bezel effect. LED Studio can draw these seams from the cabinet grid settings. For clean results, choose cabinet columns and rows that divide evenly into the logical resolution.

BRIGHTNESS, BLOOM, AND CLOSE-UP DETAIL

Real LED displays are bright light sources. In Unity, that usually means using HDR brightness and bloom in the camera or render pipeline. LED Studio also simulates emitter fill ratio, falloff,

brightness response, and optional close-up diode lens detail. At far distances, the board can blend toward a smoother image to reduce distracting aliasing and moire artifacts.

INSTALLATION AND QUICK START

REQUIREMENTS

- Unity 2022.2 or newer.
- BIRP, URP and HDRP are supported.
- The package declares dependencies for Timeline, TextMesh Pro, Input System, uGUI, UI Toolkit, and Unity Video because examples, integrations, and sources can use them.

INSTALL THE PACKAGE

1. Import LED Studio from the Unity Asset Store.
2. Let Unity resolve the package dependencies declared in package.json.
3. Open Tools > LED Studio > Welcome... or Tools > LED Studio > About... to confirm the package is installed.
4. Open Tools > LED Studio > Board Wizard to create a first board.
5. Open *Assets/LEDStudio/Examples/Scenes/Overview.unity* to inspect the example scene in this development project, or the equivalent imported example path in a packaged project.

CREATE YOUR FIRST BOARD

6. Open Tools > LED Studio > Board Wizard.
7. Enter a board name.
8. Choose a board style such as Scoreboard, Ribbon Board, JumboTron, Corner Wrap, Ribbon Group, Video Wall, or Custom.
9. Enter sample text, choose a color, and select the built-in bitmap font or a custom Unity font.
10. Leave Add Realistic Wear enabled when you want subtle dim LEDs, flicker, and color drift on the generated board.
11. Click Create Board and press Play.

What the wizard creates - A board GameObject normally includes a mesh, a renderer with the LED Studio/LED Board shader, a LedBoardController, and at least one data source.

CORE WORKFLOW

THE BOARD CONTROLLER

LedBoardController is the main runtime component. It owns the logical board texture, resolves inline settings or profile assets, discovers data sources, composes content, updates defects, binds material properties, participates in sync groups, and exposes the *ILedBoardRuntime* API for game code.

INLINE SETTINGS VERSUS PROFILES

For one-off boards, inline settings on *LedBoardController* are enough. For repeated displays, create reusable assets from the Create > LED Studio menu. When a profile is assigned, the matching inline controller settings are ignored.

Asset	Purpose
Board Profile	Logical resolution, pitch, topology, curve parameters, cabinet grid, emitter settings, and default reduction.
Source Asset	Reusable text, image, timer, score, render texture, or UI Toolkit content.
Layout Asset	Layered composition zones, priorities, fit modes, opacity, safe area, and fallback sources.
Playlist Asset	Timed state sequence with loop, sync group, transitions, and durations.
Defect Profile	Procedural or mask-driven hardware wear and failure simulation.
Quality Profile	Update budget, target update rate, reduction override, bloom, close-up mode, and defect quality.

TOPOLOGY

A board can be flat, curved, or wrapped around a corner. The wizard can generate flat panels, ribbons, video walls, and corner wraps. For custom boards, set topology and curve parameters directly in the wizard or in the board profile. The topology affects the mesh shape; logical composition still happens in a rectangular emitter grid.

CONTENT SOURCES

A source is any object that can provide content for the LED grid. Scene source components are discovered under the board. Layout assets can also reference source assets as fallbacks. Every source should have a stable *SourceId* when it needs to be addressed by layouts or runtime scripts.

Source	Use it for	Notes
LedTextDataSource	Short messages, sponsor text, labels, ticker text	Supports built-in bitmap font or custom Unity font.
LedImageDataSource	Static logos, cards, test patterns	Texture can be replaced at runtime by SourceId.
LedVideoDataSource	Video playback	Requires a scene VideoPlayer.
LedRenderTextureDataSource	Camera feeds, generated content, external systems	Useful for mini cameras or custom render pipelines.
LedCanvasDataSource	uGUI Canvas content	Renders a Canvas through a Camera.
LedUIToolkitDataSource	UI Toolkit interfaces	Uses UIDocument and PanelSettings.
LedTimerDataSource	Countdowns or clocks	Can be started, stopped, and assigned time.
LedScoreDataSource	Sports scoreboards	Receives structured home, away, period, and clock values.

FIT MODES AND REDUCTION

Sources are often higher resolution than the LED board. Fit mode decides how the source rectangle maps into the target zone: Stretch fills the zone, Fit preserves aspect ratio without cropping, Fill preserves aspect ratio while cropping, and None uses the source directly. Reduction mode decides how the high-resolution source is sampled into LED emitters. Text and score content usually benefits from Luma Priority, images can use Edge Preserving Sharpen, and video or render textures often use Box sampling.

UPDATE POLICIES

- *EveryFrame* updates when content must animate continuously.
- *OnChange* updates when content changes and saves work for static signs.
- *FixedRate* updates at a chosen Hz value for slower media or UI.
- *Manual* updates only when game code calls *MarkSourceDirty*.

LAYOUTS AND PLAYLISTS

COMPONENTS-ONLY MODE

If a board has no *LedLayoutAsset*, LED Studio composes the discovered data source components directly. Component priority, opacity, fit mode, and zone determine the result. This is the fastest way to create a simple sign or a single-source board.

LAYOUT MODE

When a *LedLayoutAsset* is assigned, the layout controls composition. Each layer has a name, *SourceId*, optional fallback Source Asset, normalized zone, fit mode, optional reduction override, priority, and opacity. Lower priority draws first. Higher priority draws over the top.

Source binding - A layout layer first looks for a scene component with a matching *SourceId*. If no component exists, the layer can use its assigned *LedSourceAsset*. Live scene-only sources such as Video, Canvas, and some integrations require scene components.

PLAYLISTS

LedPlaylistAsset schedules named states over time. Each entry has a state ID, duration, transition type, and transition duration. Playlists can loop and can carry a sync group ID so separate boards stay phase-locked.

Transition	Behavior
Cut	Switch immediately to the next state.
CrossFade	Blend between previous and next board output.
WipeLeft / WipeRight	Reveal the next state horizontally.
WipeUp / WipeDown	Reveal the next state vertically.

MULTI-PANEL BOARDS AND SYNC

BOARD GROUPS

LedBoardGroup lets multiple physical boards behave like one large display. It composes a master texture, adds optional gap pixels for bezels, slices that texture per child board, and lets each child keep its own material, emitter presentation, and defects. Layout modes include Horizontal, Vertical, and Grid. If a grid has more child boards than cells, extra boards are ignored and a warning is logged.

SYNC GROUPS

Boards with the same sync group ID read from a shared clock. *LedSyncManager* and *LedSyncDriver* coordinate clocks in play mode and editor mode. Use sync groups when separate boards need matching playlist timing, synchronized animation, or coordinated replay states.

LED STUDIO WINDOW

Open Tools > LED Studio > LED Studio Window to inspect scene boards. The window includes board discovery, sync group inspection, and playlist playback controls. In play mode, it can force-refresh boards, nudge or reset sync group time, and jump between playlist entries.

DEFECT SIMULATION

Real LED systems rarely stay perfect. Individual emitters can dim, die, flicker, drift in color, stick to a color, or fail in rows, columns, or cabinet regions. LED Studio simulates these issues through *LedDefectProfile* assets, inline defect entries, startup defect profiles, imported masks, and the runtime fault API.

Defect mode	What it simulates
Dead	Emitter is off.
Dim	Emitter is darker than expected.
Flicker	Emitter changes over time using temporal defect settings.
Stuck Color	Emitter is stuck on a chosen color.
Color Drift	Emitter color shifts subtly from neighboring LEDs.
Line Failure	Generated rows or columns fail together.
Cabinet Outage	A cabinet region loses output.

DEFECT TARGETS

Targets decide where defects appear. Common targets include scattered emitters, cabinet edges, corner cabinets, adjacent cabinets, generated rows, generated columns, border regions, and imported manual masks. Use scattered defects for general aging and cabinet targets for obvious hardware module failures.

RUNTIME FAULT CONTROL

LedBoardFaultController is a convenience component for applying a startup defect profile and exposing simple severity and enable controls. For more custom behavior, use *ILedBoardRuntime* directly.

RUNTIME API

Game code should talk to a board through *ILedBoardRuntime*. *LedBoardController* implements this interface, so scripts can drive content without depending on internal composition details.

```
using LEDStudio;
using UnityEngine;

public sealed class ScoreboardController : MonoBehaviour
{
    [SerializeField] LedBoardController board;
    [SerializeField] LedPlaylistAsset gamePlaylist;
    [SerializeField] Texture sponsorTexture;

    void Start()
    {
        ILedBoardRuntime runtime = board;
        runtime.SetPlaylist(gamePlaylist);
        runtime.SetSourceText("headline", "MATCH START");
        runtime.SetSourceTexture("sponsor", sponsorTexture);
        runtime.SetScore("HOME", "AWAY", "Q1", "12:00");
    }
}
```

Member	Purpose
SetPlaylist / ResetPlaylistPlayback	Assign or restart timed playlist playback.
SetState	Jump to a named state.
SetSourceText / SetSourceTexture	Update a source by SourceId.
SetScore	Push structured scoreboard values.
MarkSourceDirty	Refresh a manual-update source.
ApplyDefectProfile / ClearDefectProfile	Add or remove runtime defect profiles.
SetDefectSeverity / SetDefectEnabled	Adjust defect intensity or toggle defects.
SetSyncClock	Reset or scrub a board or sync group clock.
OutputTexture	Read the current logical board output texture.
CurrentStateId / PlaylistIndex / PlaylistTime / LifecycleState	Inspect board state.

TIMELINE INTEGRATION

When Timeline is available, LED Studio includes a *LedBoardTrack* that binds to *LedBoardController*. Clips can switch a board to a state ID and blend defect severity using the clip weight. This is useful for concerts, venue previsualization, cutscenes, scripted reveals, and synchronized media cues.

12. Create or open a Timeline asset.
13. Add an LED board track and bind it to the target *LedBoardController*.
14. Create clips with state IDs that match your board layouts or playlist states.
15. Use clip overlap and blending to fade defect severity or coordinate state changes.

PERFORMANCE GUIDANCE

- Choose the lowest logical resolution that still sells the display at the target camera distance.
- Use *LedQualityProfile* for shared performance settings across many boards.
- Set Target Update Rate for slow-changing signs instead of updating every frame.
- Use *OnChange* or Manual update policies for static text, logos, and UI.
- Disable bloom contribution if the scene does not use HDR bloom.
- Disable close-up mode for boards that will never be viewed near the camera.
- Keep cabinet grids aligned with logical resolution to avoid unnecessary seam artifacts.
- Use board groups for seamless multi-panel content instead of manually duplicating sources.
- Limit temporal defects on low-end targets if many boards are active at once.

TROUBLESHOOTING

BOARD IS BLANK

- Confirm the GameObject has a renderer and a material using LED Studio/LED Board.
- Confirm the controller target renderer and material slot are correct.
- Confirm at least one data source exists or the assigned layout has a valid source.
- If using layout binding, confirm the layer SourceId exactly matches the source component SourceId.
- Enter Play Mode for playlist-driven states and time-dependent content.

MATERIAL IS PINK

- Confirm the target render pipeline is installed and active.
- Reimport the package if the LED Studio/LED Board shader is missing.
- For examples, confirm non-board geometry uses LED Studio example shaders.

LAYOUT LAYER DOES NOT APPEAR

- Check layer opacity and priority.
- Check normalized zone values.
- Check fit mode and reduction settings.
- Confirm the layer source exists.
- For scene components, confirm SourceId matches exactly.
- Remember that Video, Canvas, Chart, and some live sources require scene components.

CABINET SEAMS LOOK WRONG

- Check that cabinet columns and rows evenly divide the logical resolution.
- Reduce seam width if seams are too heavy.
- Adjust bezel darkening for the target camera distance and lighting.

BOARDS ARE OUT OF SYNC

- Confirm each board or playlist uses the same sync group ID.
- In Play Mode, open Tools > LED Studio > LED Studio Window > Sync Groups and verify the group is registered.
- Use *SetSyncClock* when game code needs to reset or scrub time.

CHART GURU INTEGRATION

LedSourceType.Chart is reserved for Chart Guru integration. When Chart Guru is installed, it can contribute *LedChartDataSource* components. Scene components take precedence over layout factory sources, so Chart layers should bind to a component by SourceId. Without Chart Guru installed, Chart entries remain blank by design. Chart Guru 3D also includes optional LED Studio integration samples for streaming 3D chart output to LED boards.

SUPPORT

Web: https://www.wetzold.com/tools

Discord: https://discord.gg/uzeHzEMM4B