ButtonAnimator Easy

Compatible Unity Versions: Unity 2020.3 LTS+
Namespace: `ButtonAnimatorEasy`

A powerful, lightweight, Preset-first button animation tool for Unity that makes UI animations as simple as "Drag, Drop, Done."

ButtonAnimator Easy: Elegant, Precise, and High-Performance uGUI Animations

"Drag, Drop, Done." UI animation shouldn't be complicated.

`ButtonAnimator Easy` is a powerful, lightweight animation system designed specifically for Unity uGUI. By moving away from complex Animator Controllers and State Machines, we provide an intuitive **Preset-first system** that lets you breathe life into your UI in seconds.

Why Choose ButtonAnimator Easy?

The Ultimate VR Companion: Uses **Base Scale Caching** to keep a stable scale reference for UI interaction, significantly reducing Raycast instability issues in World Space Canvases commonly caused by scale animations in VR.

LayoutGroup Compatible: Automatically generates a `__BAE_Anim` Wrapper to handle position animations flawlessly without breaking Horizontal, Vertical, or Grid Layout systems.

Performance Optimized: Designed for VR high-frame rates (90/120 FPS), the `Update` loop performs only a minimal boolean check, ensuring near-zero CPU overhead.

Advanced Color Control: Automatically detects and animates multiple components simultaneously, including the Button Background, Icon, and TextMeshPro labels.

Flexible Override Logic: The **Local Override** system allows you to maintain global visual consistency via Presets while fine-tuning specific parameters for individual buttons.

Built-in Spatial Audio: Trigger 3D spatial sound effects at the button's exact world position without the need to manually manage `AudioSource` components.

Core Features

Animation System:

- **No coding required** - Everything can be done in the Unity Inspector
- **Preset-first design** - Use high-quality ScriptableObject presets or create your own
- **Local Override system** - Override specific animation parameters without breaking preset updates
- **Three animation states** - Hover, Click, and Disabled animations
- **DOTween-powered** - Fast and efficient animations using DOTween

Audio System:

- **Built-in audio support** - Play sound effects on Hover and Click events
- **Preset audio settings** - Configure audio clips and volumes in presets
- **Per-button audio override** - Override audio settings for individual buttons
- **No AudioSource required** - Uses AudioSource.PlayClipAtPoint for simple setup
- **VR spatial audio positioning** - Sounds play at the button's world position, providing accurate 3D spatial audio localization for VR applications

Preset System:

- **Many presets included** - Organized by physical properties: Scale, Position, Rotation, Color & Alpha, Compound & Special, Punch, and Shake
- **Easy preset switching** - Change presets without losing local overrides
- **Re-Apply functionality** - Reset to preset values with one click

Editor Integration:

- **Custom Inspector** - Beautiful, intuitive UI with status indicators

- **Preview system** - Test Hover, Click, and Disabled animations without entering Play Mode
- **Override toggle** - Easy switching between Preset and Local Override modes
- **DOTween dependency checker** - Automatic detection and setup guidance

Runtime Features:

- **Automatic interactable handling** - Disabled animations trigger automatically when Button.interactable = false
- **LayoutGroup compatible** - Base scale caching prevents layout issues
- **CanvasGroup support** - Automatically adds CanvasGroup when alpha animations are needed, prioritizes CanvasGroup.alpha over Graphic.color.a
- **Button transition management** - Automatically sets Button.transition to None to prevent conflicts
- **Multi-touch safe** - Independent Tween IDs per state prevent conflicts
- **TimeScale independent** - Animations work correctly in pause menus
- **Debug mode** - Optional detailed debug logging for troubleshooting

Design Philosophy

ButtonAnimator Easy is built on three core principles that guide every design decision:

Preset-first - Design animations once, reuse everywhere. Instead of customizing each button individually, create presets that can be shared across multiple buttons, ensuring visual consistency and rapid iteration.

Inspector-driven - No Animator Controllers, no state machines, no code. Everything is configured directly in the Unity Inspector, making UI animations accessible to designers and developers alike without the complexity of traditional animation systems.

Safe-by-default - Animations never break your UI layout or interfere with game time. Base scale caching prevents LayoutGroup issues, and TimeScale-independent animations ensure buttons work correctly even in pause menus.

When to Use / When Not to Use

Good For

- **uGUI Button** - Designed specifically for Unity's legacy UI system (uGUI)
- **Mobile, PC, and VR UI** - Works across all platforms that support uGUI
- **World Space Canvas** - Fully compatible with World Space Canvas for VR applications. Base Scale Caching ensures precise Raycast detection even during scale animations, solving a common issue where other uGUI plugins cause Raycast to break when button scale changes. All animations (Scale, Position, Rotation, Color, Alpha) function correctly regardless of Canvas Render Mode.
- **Projects requiring quick consistent animation styles** - Preset system enables rapid UI animation setup with consistent visual language

Not Suitable For

- **UI Toolkit (UIElements)** - ButtonAnimator Easy is built for GameObject-based uGUI, not the newer UI Toolkit system
- **UI deeply bound to Animator Controller** - If your UI animations are already tightly integrated with Unity's Animator system, this tool may conflict or be redundant

Quick Start

Installation

1. Import the ButtonAnimatorEasy package into your project
2. Make sure you have DOTween installed (the free version works fine)
3. Add the `ButtonAnimatorEasy` component to a GameObject with a Unity Button component

Basic Setup

1. Select a GameObject with a Button component
2. Add Component → UI → ButtonAnimator Easy

3. Assign a Preset from the Presets folder (or create your own)
4. Test in Play Mode or use the Preview buttons in the Inspector

Pro Tip: To ensure proper operation in LayoutGroup environments, it is recommended to reference UI components directly through the Inspector using `[SerializeField]` fields, rather than using `Transform.Find()`. This prevents issues when ButtonAnimator Easy automatically creates animation wrappers at runtime. See the Technical Details section for more information.

System Requirements

- Unity 2020.3 or newer
- DOTween (Free or Pro version)
- Unity UI (uGUI) package

Preset Categories

Presets are organized by physical properties for easy navigation:

Scale (Size):

- Jelly, Pulse Loop, Mobile Touch

Position (Move):

- Slide Right, Z Push VR, Levitate

Rotation (Spin):

- Tilt 3D, Spin Loading, Flip Toggle

Color & Alpha:

- Error Flash, Fade Simple, Warning Pulse

Compound & Special:

- Premium Glow, Morph Elastic, Digital Glitch

Punch:

- Strong, Medium, Light

Shake:

- Horizontal, Vertical, Multi-Direction

Key Concepts

Preset vs Local Override

- **Preset**: A ScriptableObject containing animation settings (shared across multiple buttons)
- **Local Override**: Per-button animation settings that override the Preset values
- **Override Toggle**: Enable/disable local overrides for each animation block (Hover, Click, Disabled)

Important: If you want to modify the included preset files, it is strongly recommended to duplicate them first before making changes. This prevents your customizations from being overwritten when updating to a new version of ButtonAnimator Easy, as the default preset templates may be replaced during package updates.

Animation States

- **Hover**: Triggered on OnPointerEnter (supports Loop)
- **Click**: Triggered on OnPointerDown/Up (no Loop support)
- **Disabled**: Triggered automatically when Button.interactable = false (supports Loop)

Absolute vs Relative Value Mode

The animation system supports two value modes for Position, Rotation, and Scale animations:

Absolute: The value represents the final target value that the animation will reach and stay at. The animation will animate from the current state to this absolute value.

- Example: If Position is set to (100, 0, 0) in Absolute mode, the button will animate to position (100, 0, 0) regardless of its current position.

Relative: The value is added to the current object's value. The animation will animate from the current state to (current value + relative value).

- Example: If the button is currently at position (50, 0, 0) and Position is set to (10, 0, 0) in Relative mode, the button will animate to position (60, 0, 0).

Note: Color and Alpha animations always use absolute values regardless of the mode setting. The mode only affects Position, Rotation, and Scale animations.

Color Mode

The animation system supports two color modes for controlling button color animations:

Normal: Uses a single `Color` value that applies only to the button's Image component (the first Graphic component found on the button GameObject).

- Only the button background Image component's color is animated
- Other visual elements (icons, text) are not affected by color animations in Normal mode
- This is the default mode

Advanced: Uses separate color values for different button components: `ButtonColor`, `IconColor`, and `TextColor`.

- **ButtonColor**: Controls the button background Graphic component (usually an Image on the button GameObject)
- **IconColor**: Controls the icon Graphic component (usually an Image in a child GameObject)
- **TextColor**: Controls the text label (TextMeshProUGUI component)

Component Detection:

In Advanced mode, ButtonAnimator Easy automatically detects the target components:

- **ButtonGraphic**: Auto-detects the Image component on the button GameObject (or manually assignable in the Inspector)
- **IconGraphic**: Auto-detects the first Image component in child objects (or manually assignable in the Inspector)
- **TextLabel**: Auto-detects TextMeshProUGUI on the button or in its children (or manually assignable in the Inspector)

Note: If a button has multiple components of the same type (e.g., two Image components for background and border), the auto-detection will select the first one found. You can manually assign these components in the Inspector for precise control, or leave them empty to use auto-detection.

Important: Color and Alpha animations always use absolute values regardless of the Color Mode setting. Color Mode only determines whether to use a single color or separate colors for different components.

Transform Animation Types

The animation system supports three transform animation types:

- **Normal**: Standard smooth animation from current state to target value
- **Punch**: Elastic bounce effect with configurable vibrato, randomness, and elasticity
- **Shake**: Random shake effect with configurable vibrato, randomness, and fade-out option

Animation Direction

For Normal type animations, you can choose the animation direction:

- **To**: Animates from the current state to the target value (default)
- **From**: Animates from the target value to the current state (reversed)

Note: Direction only applies to Normal type animations. Punch and Shake animations always use their default behavior.

Easing Options

The animation system supports two easing types:

- **Ease**: Use predefined DOTween easing functions (e.g., OutQuad, InOutCubic, etc.)
- **AnimationCurve**: Use a custom AnimationCurve for complete control over the easing curve

Priority System

When resolving animation values:

1. Local Override (if enabled)
2. Preset value
3. Fallback default

Audio System

- **Preset Audio**: Audio clips and volumes can be configured in Button Presets
- **Audio Override**: Each button can override preset audio settings using the "Use Preset Audio" toggle
- **Hover Sound**: Plays when the pointer enters the button (OnPointerEnter)
- **Click Sound**: Plays when the button is pressed (OnPointerDown)
- **No AudioSource Required**: Uses Unity's AudioSource.PlayClipAtPoint, so no AudioSource component is needed on the button GameObject
- **VR Spatial Audio**: Sounds are played at the button's world position using `AudioSource.PlayClipAtPoint()`, providing accurate 3D spatial audio localization. This is especially important for VR applications where users rely on spatial audio cues to locate UI elements in 3D space. The sound position automatically matches the button's world coordinates, ensuring proper distance attenuation and directional audio feedback

Frequently Asked Questions

General Questions

Q: Do I need to write any code?

A: No! Everything can be done in the Unity Inspector. ButtonAnimator Easy is designed to be completely code-free.

Q: Can I use ButtonAnimator Easy with DOTween Free?

A: Yes! ButtonAnimator Easy works with both the free and Pro versions of DOTween.

Q: Does it work in Edit Mode?

A: Yes! The Preview system allows you to test animations without entering Play Mode. Click the "Preview Hover", "Preview Click", or "Preview Disabled" buttons in the Inspector.

Q: What Unity version do I need?

A: Unity 2020.3 LTS or newer is required.

Preset System

Q: What's the difference between Preset and Local Override?

A:

- **Preset**: A ScriptableObject that can be shared across multiple buttons. When you update a preset, all buttons using it (without overrides) get the new values.
- **Local Override**: Per-button settings that override the preset. These are stored on the component itself and won't be affected by preset updates.

Q: Can I update a Preset without breaking my customizations?

A: Yes! That's the whole point of the Local Override system. Buttons with overrides keep their custom values, while buttons without overrides automatically use the updated preset values.

Q: How do I reset all overrides back to the preset?

A: Click the "Re-Apply Preset" button in the Inspector. This clears all local overrides and resets the button to use pure preset values.

Q: Can I create my own presets?

A: Yes! Right-click in the Project window → Create → ButtonAnimator Easy → Button Preset. Configure the animation blocks and audio settings, then save.

Q: Can I modify the included preset files?

A: While you can modify the included preset files directly, it is **strongly recommended** to duplicate them first before making any changes. When you update ButtonAnimator Easy to a new version, the default preset templates may be overwritten, which would cause you to lose your customizations. To avoid this:

1. Select the preset file you want to modify in the Project window
2. Press `Ctrl+D` (Win) or `Cmd+D` (Mac), or use `Edit > Duplicate` from the menu bar
3. Rename the duplicate to something like "MyCustom $PresetName$"
4. Modify the duplicate instead of the original
5. Assign the duplicate preset to your buttons

This way, your custom presets will remain safe during package updates.

Animation Questions

Q: Which animation states support Loop?

A: Hover and Disabled states support Loop. Click animations do not support Loop (by design, as click animations should complete).

Q: Why doesn't my Disabled animation play?

A: The Disabled animation automatically triggers when `Button.interactable = false`. Make sure:

1. The Button component exists
2. The interactable property is set to false
3. The Disabled block in your preset has animation settings configured

Q: Can I use multiple Graphic components?

A: ButtonAnimator Easy only applies Color/Alpha animations to the first Graphic component found (Image, Text, TextMeshProUGUI, etc.). This is by design to keep things simple and predictable.

Q: The auto-detection in Advanced mode selected the wrong component. How do I fix it?

A: If auto-detection picks the wrong component (e.g., when a button has multiple Image components like background and border), simply drag the correct component directly into the corresponding field in the Inspector (ButtonGraphic, IconGraphic, or TextLabel). Manual assignment always takes precedence over auto-detection.

Q: Does it work with CanvasGroup?

A: Yes! ButtonAnimator Easy automatically adds a CanvasGroup component when alpha animations are needed. If a CanvasGroup already exists, it will prioritize `CanvasGroup.alpha` over `Graphic.color.a` for alpha animations.

Q: What happens if I don't have a Button component?

A: You'll see a warning in the Inspector. The component will still work, but Pointer events may not fire correctly. It's recommended to always use ButtonAnimator Easy with a Unity Button component.

Q: Will ButtonAnimator Easy conflict with Button's built-in transitions?

A: No! ButtonAnimator Easy automatically sets `Button.transition` to `None` to prevent conflicts. This ensures that only ButtonAnimator Easy controls the button's visual feedback.

Editor Questions

Q: The Preview buttons don't work. What's wrong?

A: Make sure:

1. DOTween is installed
2. The component is properly initialized
3. A Preset is assigned
4. Try clicking "Reset Overrides" first, then preview again
5. For Disabled preview, ensure the Disabled block in your preset has animation settings configured

Q: I see "DOTween not installed" error. What do I do?

A:

1. Click "Install DOTween" to open the Asset Store
2. Import DOTween (Free version is fine)
3. Click "Recheck" in the Inspector
4. The error should disappear

Q: Can I use ButtonAnimator Easy without a Preset?

A: Technically yes, but it's not recommended. The system is designed to work with presets. If no preset is assigned, default values will be used.

Troubleshooting

Q: Animations don't play at all

A: Check:

1. DOTween is installed
2. Button component exists
3. Preset is assigned
4. Animation blocks have non-default values
5. Button is interactable (for Hover/Click animations)

Q: Disabled animation doesn't trigger

A: Make sure:

1. Button component exists
2. `Button.interactable` is set to `false`
3. Disabled block in preset has animation settings
4. The component is initialized (check Awake/Start)

Q: Override values don't work

A: Make sure:

1. The "Override" toggle is enabled for that animation block
2. The override values are different from the preset values
3. The component has been initialized

Q: Preview doesn't restore properly

A: Click "Reset Overrides" or manually call `RestorePreview()` in code. The preview system should automatically restore when animations complete, but if it doesn't, use the reset button. Note that infinite loop animations (Loops = -1) will not auto-restore until you manually call `RestorePreview()`.

Q: I get an error when running the Demo scene: "InvalidOperationException: You are trying to read Input using the UnityEngine.Input class, but you have switched active Input handling to Input System package in Player Settings."

A: This error occurs when your project is configured to use the **New Input System**, but the Demo scene uses the **Legacy Input System** (StandaloneInputModule). To fix this, you have two options:

Option 1: Switch the EventSystem to New Input System (Recommended if your project uses New Input System)

1. Open the Demo scene (`Assets/ButtonAnimatorEasy/Samples/Demo.unity`)
2. Select the `EventSystem` GameObject in the Hierarchy
3. In the Inspector, find the Standalone Input Module component and click the **"Replace with InputSystemUIInputModule"** button.

Option 2: Change Player Settings to support both Input Systems

1. Go to Edit → Project Settings → Player
2. Open the "Other Settings" section
3. Under "Active Input Handling", select "Both" instead of "Input System Package (New)"
4. This allows both Input Systems to work, but Option 1 is recommended for cleaner project setup

Integration

Q: Does it work with UI Toolkit (UIElements)?

A: No, ButtonAnimator Easy is designed for Unity's legacy UI system (uGUI). It works with GameObject-based UI elements.

Q: Does it work with World Space Canvas for VR?

A: Yes! ButtonAnimator Easy is fully compatible with World Space Canvas and is specifically designed to handle VR requirements. Unlike many uGUI plugins that suffer from Raycast precision issues when button scale changes (a critical problem in VR where precise interaction is essential), ButtonAnimator Easy uses a Base Scale Caching system that ensures Raycast detection remains accurate even during scale animations.

Key VR Benefits:

- **Precise Raycast Detection**: Base Scale Caching maintains the original scale reference, ensuring Unity's GraphicRaycaster calculates hit boxes correctly throughout animations. This prevents the common VR issue where Raycast misses or incorrectly detects buttons during scale animations.
- **Performance Optimized**: The `Update()` method only performs a simple boolean comparison (`Button.interactable` state check), making it safe for VR's high frame rate requirements (72/90/120 FPS). The overhead is negligible even with many buttons in the scene.
- **All Animation Types Supported**: Scale, Position, Rotation, Color, and Alpha animations all work correctly in World Space Canvas without Raycast or visual issues.

All animations function properly regardless of Canvas Render Mode (Screen Space or World Space), making ButtonAnimator Easy an ideal solution for VR UI development.

Q: Can I control animations from code?

A: While ButtonAnimator Easy is designed to be code-free, you can access the component and call methods like `PlayHover()` , `PlayClick()` , `PlayDisabled()` , and `RestorePreview()` if needed. However, the normal workflow doesn't require any code.

Quick Code Example:

```
GetComponent<ButtonAnimatorEasy>().PlayClick();
GetComponent<ButtonAnimatorEasy>().PlayHover();
```

This allows you to programmatically trigger button animations when needed, while still

benefiting from the preset system and all the automatic features of ButtonAnimator Easy.

Audio Questions

Q: How do I add sound effects to my buttons?

A:

1. Assign audio clips to the Preset's Audio Block (Hover Sound and/or Click Sound)
2. Set the volume levels (0-1 range)
3. The sounds will play automatically when buttons using that preset are hovered or clicked

Q: Can I use different sounds for different buttons?

A: Yes! Disable "Use Preset Audio" on a button and configure the Audio Override section with different audio clips and volumes for that specific button.

Q: Do I need an AudioSource component on my button?

A: No! ButtonAnimator Easy uses `AudioSource.PlayClipAtPoint()`, which doesn't require an AudioSource component. The sounds will play at the button's position, providing spatial audio feedback that matches the button's location in the scene.

Q: How does spatial audio work for VR applications?

A: ButtonAnimator Easy plays sounds at the button's exact world position using `AudioSource.PlayClipAtPoint()`. This provides true 3D spatial audio localization, which is essential for VR applications:

- **Accurate Positioning**: Sounds originate from the button's world coordinates, allowing users to locate UI elements by sound direction
- **Distance Attenuation**: Unity's audio system automatically applies distance-based volume attenuation based on the listener's position
- **World Space Canvas Support**: Works seamlessly with World Space Canvas setups, where buttons exist in 3D space and spatial audio is crucial for user orientation

- **No Setup Required**: No need to configure AudioSource components or manage audio positioning manually - it's handled automatically

This spatial audio system enhances VR user experience by providing audio feedback that matches the visual position of UI elements, making interaction more intuitive and immersive.

Q: Why aren't my sounds playing?

A: Check:

1. Audio clips are assigned in the Preset's Audio Block (or Audio Override if not using preset audio)
2. Volume is greater than 0
3. "Use Preset Audio" is enabled if you're using preset audio, or disabled if using overrides
4. The button is interactable (disabled buttons don't play hover/click sounds)

Technical Details & Performance

This section covers technical implementation details and performance considerations for developers who need deeper understanding of how ButtonAnimator Easy works under the hood.

Implementation Details

Interactable State Monitoring

ButtonAnimator Easy monitors `Button.interactable` in the `Update()` method. When it changes from `true` to `false`, the Disabled animation triggers. When it changes from `false` to `true`, the Disabled animation stops and the button restores to base state.

Important: When `interactable = false`, Unity's EventSystem does NOT send Pointer events. That's why the Disabled animation must be triggered via state monitoring, not Pointer events.

Performance for VR: The `Update()` method performs only a simple boolean comparison (`Button.interactable` state check). This minimal overhead makes it safe for VR applications that require high frame rates (72/90/120 FPS). Even with many buttons in the scene, the performance impact is negligible, as each button component performs only one boolean comparison per frame.

Note for Performance-Critical Scenarios: Monitoring only happens when the component is active and enabled. Inactive or disabled components do not execute `Update()`, so if you have 100 buttons in a scene but only 10 are active, only those 10 will perform the state check each frame. This Unity behavior ensures optimal performance even in extreme scenarios with many buttons.

TimeScale Independence

By default, animations use `SetUpdate(true)`, which makes them ignore TimeScale. This ensures animations work correctly in pause menus. You can modify this behavior in the code if needed.

Multi-Touch Protection

Each animation state (Hover, Click, Disabled) uses a unique Tween ID. This prevents animations from killing each other when multiple touch events occur.

LayoutGroup Compatibility & VR Raycast Precision

ButtonAnimator Easy implements Base Scale Caching, which stores the original button scale on `Awake()`. This design serves two critical purposes:

LayoutGroup Protection: By caching the base scale, LayoutGroup components (HorizontalLayoutGroup, VerticalLayoutGroup, GridLayoutGroup) are not affected by scale animations, preventing layout calculation errors.

VR Raycast Precision: This is particularly important for VR applications using World Space Canvas. Unity's GraphicRaycaster uses the button's RectTransform scale to calculate hit detection areas. When scale animations change the button's visual size, the Base Scale Caching system ensures:

- The original scale reference is always preserved, allowing correct hit box calculations
- Animation target values are calculated relative to the cached base scale, maintaining consistency
- Raycast detection remains accurate throughout animations, preventing the common VR issue where buttons become difficult or impossible to interact with during scale animations

This Base Scale Caching approach is a key differentiator that makes ButtonAnimator Easy reliable for VR development, where precise interaction detection is essential.

Animation Wrapper Name (`AnimWrapperName = "__BAE_Anim"`)

When a button is placed under a LayoutGroup (e.g., HorizontalLayoutGroup, VerticalLayoutGroup, GridLayoutGroup), ButtonAnimator Easy automatically creates an Animation Wrapper GameObject to safely handle position animations without breaking the layout system.

Wrapper Name: The wrapper GameObject is named `"__BAE_Anim"` (where BAE stands for ButtonAnimator Easy). This naming convention uses a double underscore prefix (`__`) to clearly identify it as a system-generated object in the Unity Hierarchy.

When Created:

- Only created at runtime when the button is under a LayoutGroup parent
- Automatically reused if it already exists
- Not created in Edit Mode to avoid scene modifications

Purpose:

- Prevents LayoutGroup conflicts when animating button position
- Allows position animations to work correctly within layout systems
- Maintains visual hierarchy by moving visual components (Image, Text, TextMeshProUGUI) into the wrapper

Technical Details:

- The wrapper is a child GameObject of the button
- It has a RectTransform with stretch anchors (anchorMin = (0,0), anchorMax = (1,1))
- Visual components are automatically moved from the button to the wrapper
- The wrapper's transform properties are immutable: localPosition = (0,0,0), localScale = (1,1,1), localRotation = identity

Note: If you see a GameObject named `__BAE_Anim` in your button's hierarchy, this is normal and expected behavior when using ButtonAnimator Easy with LayoutGroups. Do not manually delete or modify this wrapper, as it is managed automatically by the system.

Important - Script Reference Paths: If your code uses `transform.Find("Image")` or similar methods to locate child objects by name, be aware that when the Animation Wrapper is created at runtime, the hierarchy structure changes. Visual components (Image, Text, TextMeshProUGUI, etc.) are moved from the button GameObject to the `__BAE_Anim` wrapper, so the direct path breaks.

Recommended Solutions:

Use `**SerializedField**` **(Best Practice):** Assign component references directly in the Inspector. This is the most reliable method and works regardless of hierarchy changes:

Use `**GetComponentInChildren<T>()**` **:** This method searches the entire child hierarchy, including the wrapper, and will find components regardless of where they are located:

Avoid `**transform.Find()**` **:** Using `transform.Find("Image")` or `transform.Find("__BAE_Anim/Image")` is fragile and will break if the hierarchy changes. These methods are not recommended for accessing visual components when using ButtonAnimator Easy.

Integration with Other Animation Systems

ButtonAnimator Easy can be used alongside other animation systems, but be careful. If other systems are also animating the same Transform/Graphic properties, they may conflict. ButtonAnimator Easy uses unique Tween IDs to prevent conflicts within itself.

Debug Mode

Enable the "Show Debug Info" option in the Inspector. This will output detailed debug messages to the Console, including animation state changes, value calculations, and tween lifecycle events.

Performance

Performance Overhead

Minimal. ButtonAnimator Easy uses DOTween directly, which is highly optimized. The only overhead is:

- One Update() call per frame to monitor interactable state (performs only a boolean comparison)
- Component caching on Awake (one-time cost)

VR Performance: The `Update()` method is optimized for high frame rate applications. It performs a single boolean comparison (`Button.interactable` state check) per frame, making it safe for VR applications targeting 72/90/120 FPS. This minimal CPU cost ensures smooth performance even with many buttons in the scene.

Scalability

ButtonAnimator Easy is designed to be lightweight. Each button has its own component instance, but they all share presets efficiently. You can use this on many buttons without significant performance impact. The system is particularly well-suited for VR applications where performance is critical, as the overhead scales linearly with the number of buttons and remains minimal even in complex UI scenes.

```
    private Image _imgButton;

    private void Awake()
    {
        _imgButton = GetComponentInChildren<Image>();
    }
```

```
    [SerializeField] private Image ImgButton;
    [SerializeField] private Text TxtLabel;
```