UIFX
Offline Documentation

www.chocolatedinosaur.com

22 December 2023

Table of contents:

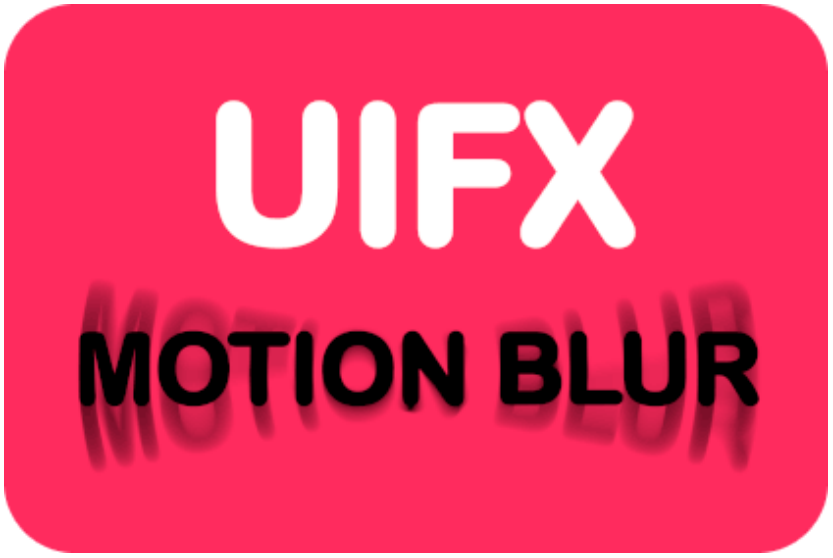
Unity Assets

At `Chocolate Dinosaur` we have released some of our internal Unity plugins to make them available to other developers. Our team has over 12 years of experience developing plugins for Unity, and we aim for high quality and customer satisfaction.

UIFX

`UIFX` is a collection of powerful yet easy-to-use UI assets for Unity.

Currently, there are THREE assets included in the bundle (also available individually), with more coming soon:


UIFX
MOTION BLUR


UIFX-BlurFilter


UIFX
TRAIL EFFECT

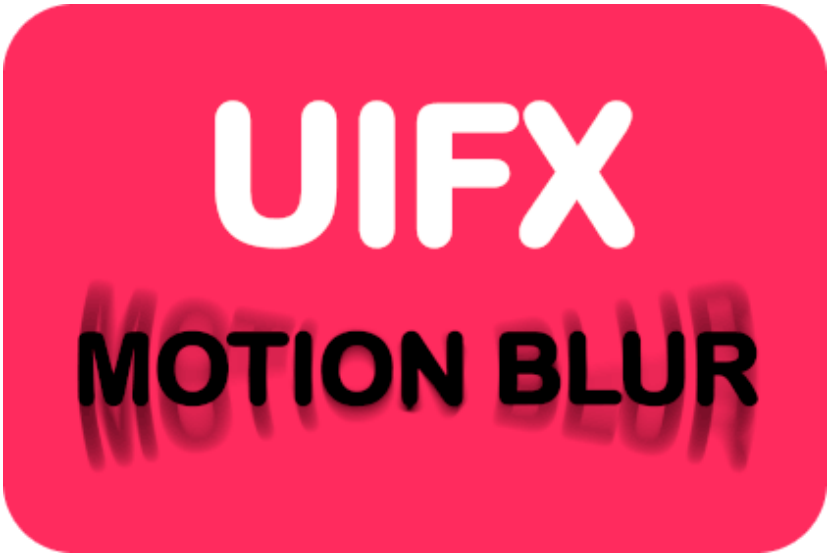
About

`UIFX-Bundle` is a collection containing ALL of the UIFX assets.

We offer the collection at a **discounted rate**.

When you own this bundle, you will automatically be able to download the included assets from the Unity Asset Store for free.

Currently, there are THREE assets included in the bundle, with more coming soon:



UIFX-BlurFilter

Download

`UIFX-Bundle` is available for download from the Unity Asset Store.

Rating / Reviews ★★★★☆

Once you've tried this asset, we would really appreciate your rating / review on the Unity Asset Store.

Release Notes

22 December 2023 - Version 1.1.3

- The new asset `UIFX - Blur Filter` version 1.0.0 has been added to the bundle

16 December 2023 - Version 1.1.2

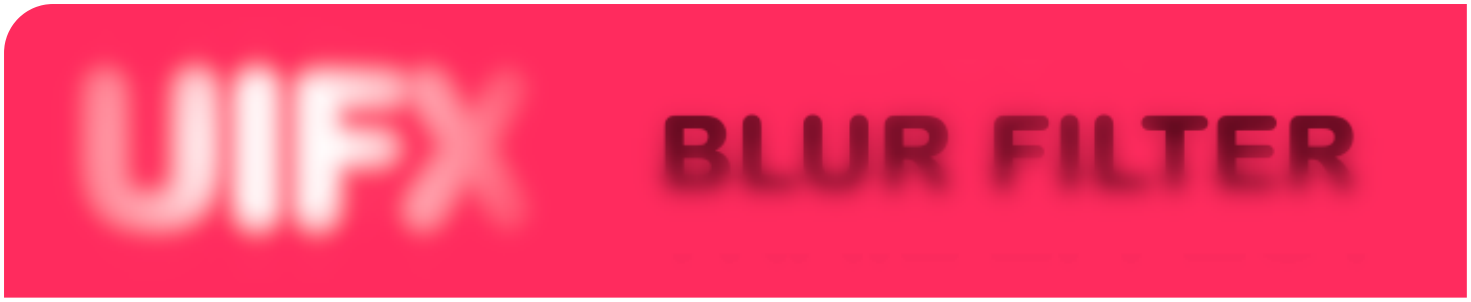
- `UIFX - Motion Blur` updated to version 1.2.0

6 December 2023 - Version 1.1.1

- `UIFX - Motion Blur` updated to version 1.1.1
- `UIFX - Trail Effect` updated to version 1.1.1

24 November 2023 - Version 1.0.0

- First release



About UIFX - Blur Filter

Overview

What?

`UIFX-BlurFilter` is a new visual effect for Unity's UI (uGUI) components that allows visual blurring of UI components.

Why?

Unity's UI system currently doesn't have an option to blur UI components.

Blurring can be used to take UI transitions to the next level, or to indicate deactivated UI elements.

How?

Just add the `UIFX-BlurFilter` component to your UI component. That's it!

Features

- Easy to use
- Highly optimised
- Cross-platform
- Built-in/URP/HDRP
- Source code included
- Well documented
- Well supported
- API documentation
- Gamma / Linear supported

- Effect works in edit mode (doesn't require entering play mode)

Compatibility

Unity Version	2023.x	2022.x	2021.x	2020.x	2019.x	2018.x	2017.x
Supported	✓	✓	✓	✓	✓	✗	✗

Platform	Windows/UWP	macOS	Linux	Android	iOS / tvOS	WebGL	PS4 / PS5	XBox One
Supported	✓	✓	✓	✓	✓	✓	✓	✓

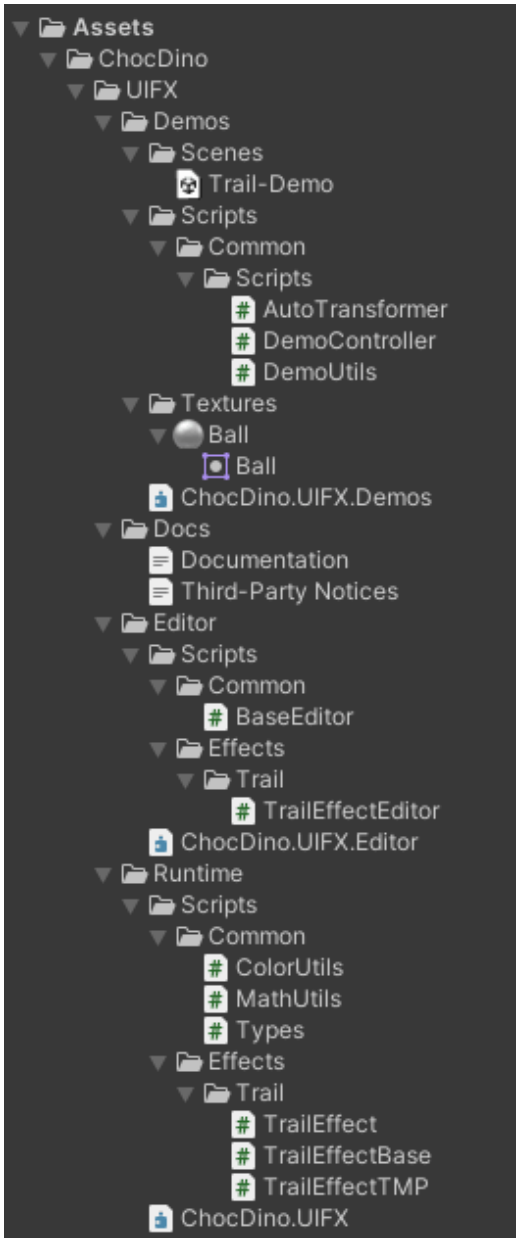
Render Pipeline	Built-in	URP	HDRP
Supported	✓	✓	✓

Dependencies

This plugin requires the `Unity.UI` package that is included with Unity by default.

The Asset Package

The asset package consists of the following files:



A demo scene is included. All scripts are grouped using Assembly Definition files.

Supported Features

UI Feature	Supported	Notes
Components:		
Text	✓	
Image	✓	
RawImage	✓	

UI Feature	Supported	Notes
Mask	✓	
Rect Mask 2D	✓	
CanvasGroup	✓	
TextMeshPro	✗	
Custom	✓	
Canvas Render Modes:		
Screen-Space Overlay	✓	
Screen-Space Camera	✓	
World Space	✓	
Effects:		
Shadow	✓	
Outline	✓	

UI Feature	Supported	Notes
Render Pipelines:		
Built-in	✓	
URP	✓	
HDRP	✓	
Colorspaces:		
Gamma	✓	

UI Feature	Supported	Notes
Linear	✓	

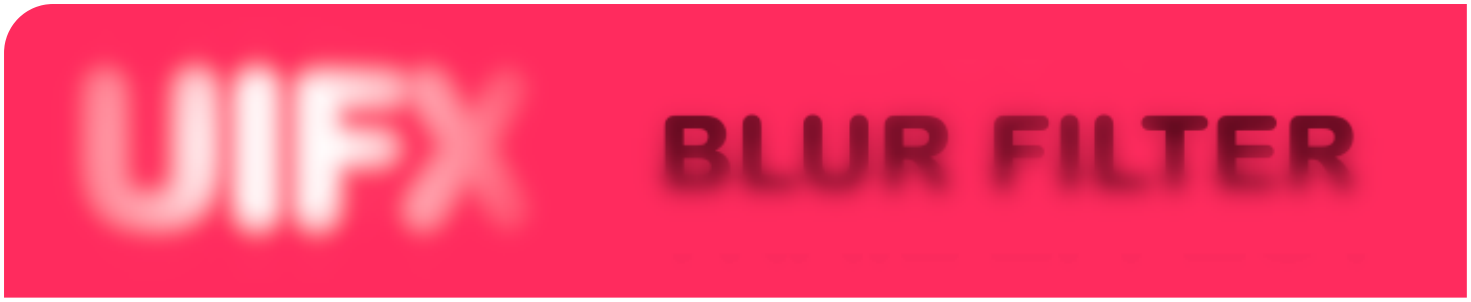
Advanced

Performance

`UIFX-BlurFilter` has had several optimisation passes, so we can say with confidence that it is highly performant. There are still a number of optimisations we have planned for version 1.1.0 coming soon!

Garbage Collection

This component has been optimised to reduce garbage generation.

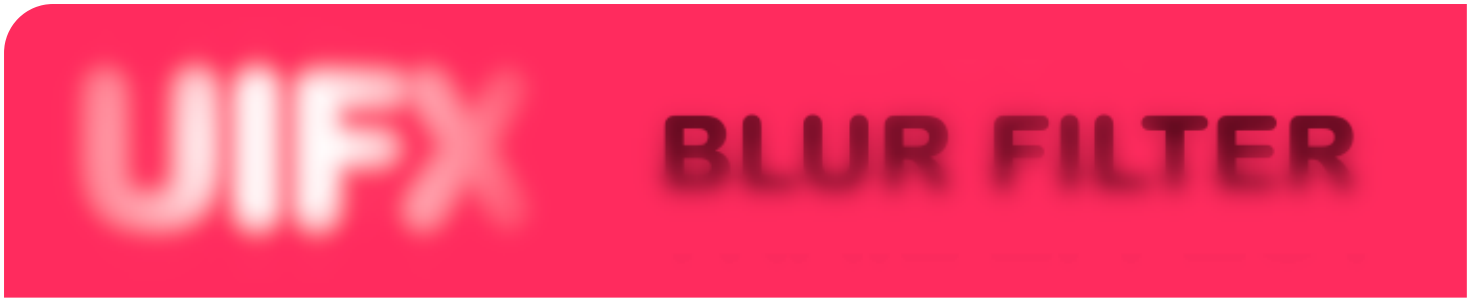


Demos

Video Demos

These videos show some examples of the `UIFX-BlurFilter` component:

Demo 1 - Blur Filter applied to a Text component.



Download

`UIFX-Blur-Filter` is available for download from the Unity Asset Store.

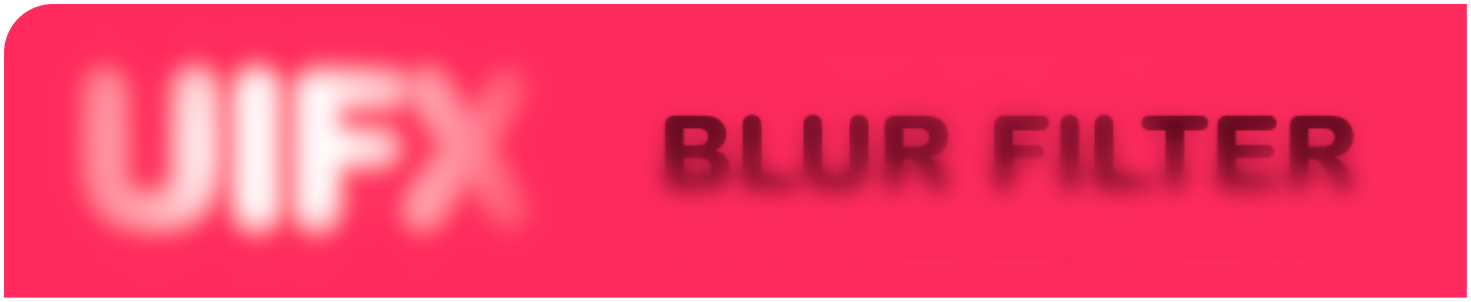
Rating / Reviews ★★★★☆

Once you've tried this asset, we would really appreciate your rating / review on the Unity Asset Store.

Release Notes

22 December 2023 - Version 1.0.0

- First release



BlurFilter Component

Overview

This component applies a blur filter to the UI object it is applied to.

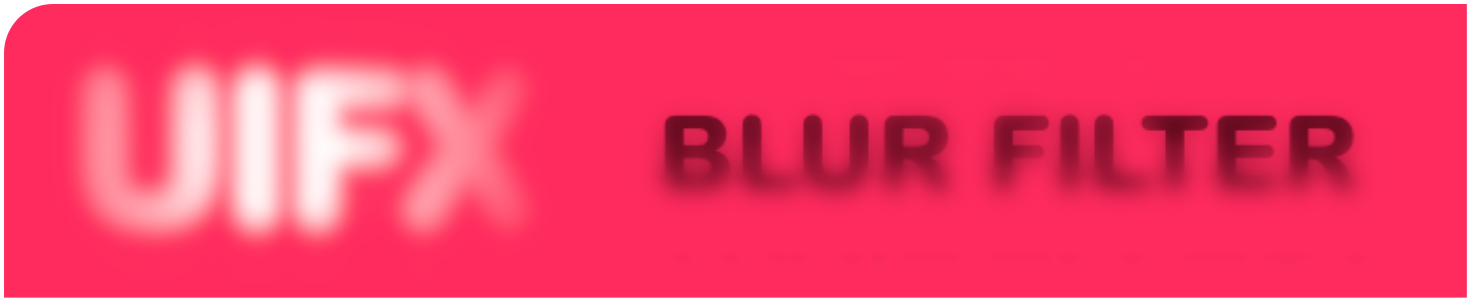
Properties

Property	Type	Details
Blur		
Axes	Enum	The 2D axes to blur. Options are: `Both` - Both horizontal and vertical blurring. (DEFAULT) `Horizontal` - Only horizontal blurring. `Vertical` - Only vertical blurring.
Blur	Float	Represents the maximum amount of blur (at `Strength` == 1.0). The value represents a fraction of the screen size that the `Graphic` occupies. So a value of 0.1 would create a blur with a kernel width 10% of the image width/height. Default value is 0.01, range is [0..0.1].
Fade		
Alpha Curve	Curve	Optional curve to control transparency. The curve ranges from time [0..1] and value [0..1] and is useful for fading down the visual as the `Strength` increases.
Apply		

Property	Type	Details
Strength	Float	Strength of the effect. Default value is 1.0, range is [0..1]

Usage

Add this component to any `GameObject` that contains a UI `Graphic` component (eg `Text`, `Image`, `RawImage`, etc). The object will now render with an adjustable blur effect applied.



Scripting

Code Snippets

The namespace

```
using ChocDino.UIFX;
```

Add the BlurFilter component to your GameObject

```
// Add the component to your GameObject and set default properties
var blur = AddComponent<BlurFilter>();

blur.MaxBlur = 0.05f;
blur.Strength = 0.5f;
blur.BlurAxes2D = BlurAxes2D.Default;
blur.AlphaCurve = new AnimationCurve(new Keyframe(0f, 1f, -1f, -1f), new Keyframe(1f, 0f, -1f, -1f));  // AlphaCurve is optional and can be null
```

Complete Example

TrailExample.cs

```
using UnityEngine;
using ChocDino.UIFX;

/// Demonstrates the scripting API for the BlurFilter component

/// Press keys 1 to 5 to test some API functionality.

/// NOTE: The GameObject is required to have a UI component of type Graphic
[RequireComponent(typeof(Graphic))]
```

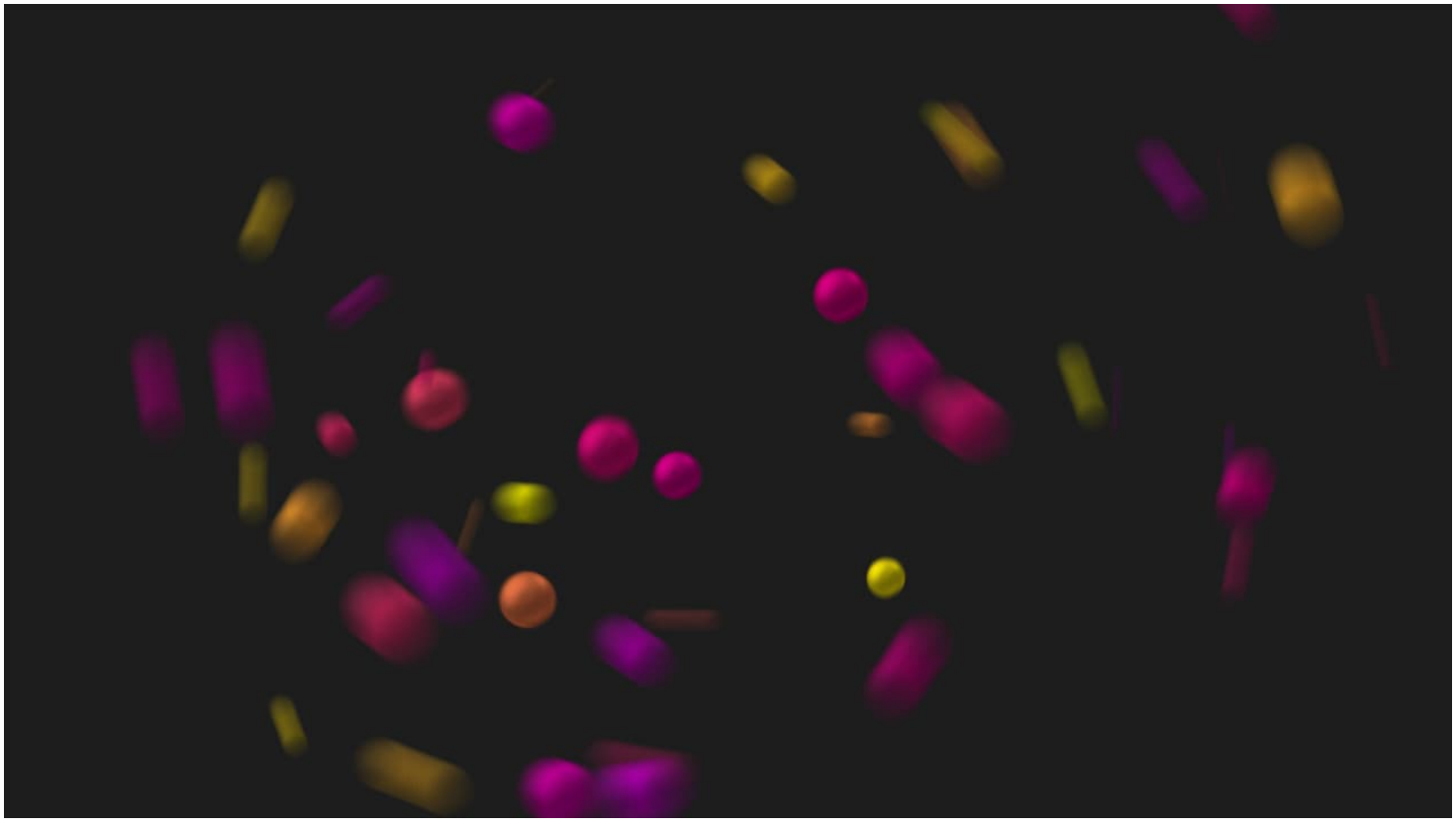
```
public class BlurExample : MonoBehaviour
{
    private BlurFilter _blur;
    private float _blurTarget = 0f;

    void Start()
    {
        // Get or create the component
        _blur = GetComponent<BlurEffect>();
        if (_blur == null)
        {
            _blur = AddComponent<BlurFilter>();
        }
    }

    void Update()
    {
        // Keys 1..5 demonstrate some API functionality
        if (Input.GetKeyDown(KeyCode.Alpha1))
        {
            _blur.BlurAxes2D = BlurAxes2D.Default;
        }
        else if (Input.GetKeyDown(KeyCode.Alpha2))
        {
            _blur.BlurAxes2D = BlurAxes2D.Horizontal;
        }
        else if (Input.GetKeyDown(KeyCode.Alpha3))
        {
            _blur.BlurAxes2D = BlurAxes2D.Vertical;
        }
        else if (Input.GetKeyDown(KeyCode.Alpha4))
        {
            _blurTarget = 0f;
        }
        else if (Input.GetKeyDown(KeyCode.Alpha5))
        {
            _blurTarget = 1f;
        }

        // Animate blur towards target amount
        if (_blur.Strength != _blurTarget)
        {
            _blur.Strength = MAthf.MoveTowards(_blur.Strength, _blurTarget,
Time.deltaTime);
        }
    }
}
```

About

Overview

What?

`UIFX-MotionBlur` is a new visual effect for adding motion blur to Unity's UI (uGUI) components, without using the Post Processing Stack package.



Why?

Adding motion blur can **greatly improve the visual quality and professional appearance** of fast moving UI components due to perceived smoother animation. This is especially true on large displays (eg video walls, digital signage) and text.

Unity's UI system currently doesn't have any option to render with motion blur.

At `CHOCOLATE DINOSAUR` we strive to create very smooth and professional motion, so we built this motion blur system to improve the quality of our UI animations and thought we'd make it available to others.

How?

Just add one of the `UIFX-MotionBlur` components to your UI component. That's it!

The component will track the full motion of the object (translation, rotation, scale) either at a transform level, or at a per-vertex level. A new mesh is then generated based on the motion and this is rendered in place of the original UI geometry. The effect is only applied when the object is in motion.

Features

- Easy to use
- Highly optimised
- Cross-platform
- Built-in/URP/HDRP
- Source code included
- Well documented
- Well supported
- API documentation
- Supports TextMeshPro (TMP)
- Doesn't require Post Processing Stack package

Compatibility

Unity Version	2023.x	2022.x	2021.x	2020.x	2019.x	2018.x	2017.x
Supported	✓	✓	✓	✓	✓	✗	✗

Platform	Windows/UWP	macOS	Linux	Android	iOS / tvOS	WebGL	PS4 / PS5	XBox One
Supported	✓	✓	✓	✓	✓	✓	✓	✓

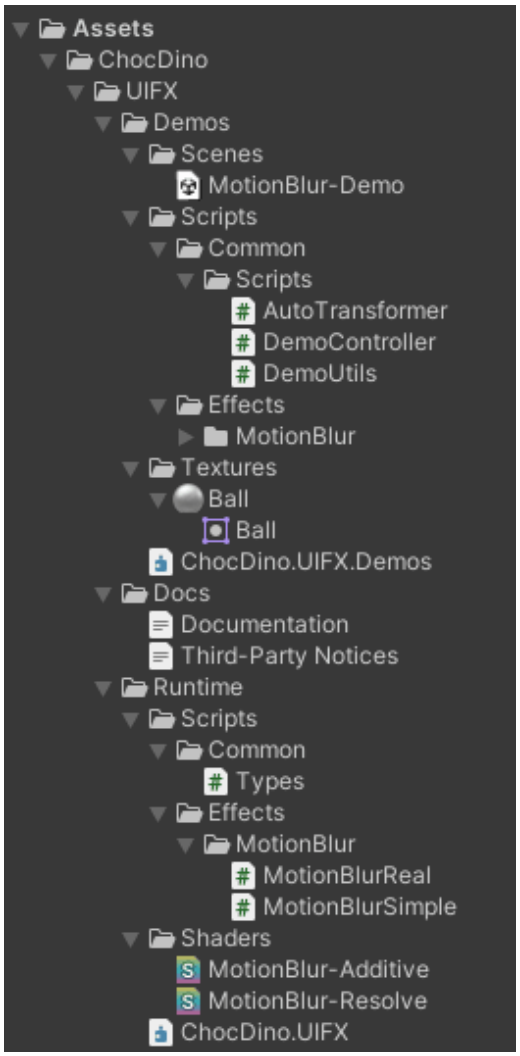
Render Pipeline	Built-in	URP	HDRP
Supported	✓	✓	✓

Dependencies

This plugin requires the `Unity.UI` package that is included with Unity by default.

The Asset Package

The asset package consists of the following files:



A demo scene is included. All scripts are grouped using Assembly Definition files.

Components

There are two components with different quality levels - `MotionBlurSimple` and `MotionBlurReal`.

While the `MotionBlurSimple` component is more performant, the `MotionBlurReal` component has much higher visual quality.

Additionally we include an extra components for `TextMeshPro` support: `MotionBlurSimpleTMP`.

General Comparison

Feature	`MotionBlurSimple`	`MotionBlurReal`
Visual Quality	Medium	High
CPU Load	Low	Low
GPU Load	Low	Medium

Supported Features

UI Feature	`MotionBlurSimple`	`MotionBlurReal`	Notes
Components:			
Text	✓	✓	
Image	✓	✓	
RawImage	✓	✓	
Mask	✓	✓	
Rect Mask 2D	✓	✓	
CanvasGroup	✓	✓	
TextMeshPro	✓	✗	See Notes for `MotionBlurReal`
Custom	✓	✓	See Notes for `MotionBlurReal`
Canvas Render Modes:			

UI Feature	`MotionBlurSimple`	`MotionBlurReal`	Notes
Screen-Space Overlay	✓	✗	Coming soon*
Screen-Space Camera	✓	✓	
World Space	✓	✗	Coming soon*
World Space Camera Types:			
Orthographic	✓	✓	
Perspective	✓	✗	
Render Pipelines:			
Built-in	✓	✓	
URP	✓	✓	
HDRP	✓	✓	
Effects:			
Shadow	✓	✓	`MotionBlurSimple` may render with flickering if Shadow effect is applied first
Outline	✓	✓	`MotionBlurSimple` may render with flickering if Outline effect is applied first
Colorspaces:			
Gamma	✓	✓	
Linear	✓	✓	

* We're working on these highly requested features.

Advanced Notes

Frame-rate Independent Notes

There is an property `FrameRateIndependent` which enables frame-rate independent motion blur. This option is enabled by default, and it causes scaling of the motion blur length based on the frame rate. This results in a consistent motion blur length regardless of the frame-rate. This is often desired if motion-blur is meant to be used artistically so that it looks the same at different frame-rates. With this option disabled, the motion-blur will behave more naturally - ie, when the frame-rate is low the motion blur trail will become longer, and when the frame-rate is higher it will become shorter.

UI Effects

Effects such as `Shadow` and `Outline` can be used with either motion blur effects, however for best results they should be applied after the motion blur effect.

Since `MotionBlurSimple` renders the motion blur quads directly to the UI, any effect (eg `Shadow` or `Outline`) that is applied will also be rendered multiple times. This can make using these effects more expensive than `MotionBlurReal` which only renders a single quad.

Performance

The `UIFX-MotionBlur` asset has had several optimisation passes, so we can say with confidence that it is highly performant for what it is doing. That said, there are some considerations that should be made. See the documented performance notes for each component.

Garbage Collection

Both components have been optimised to reduce garbage generation.

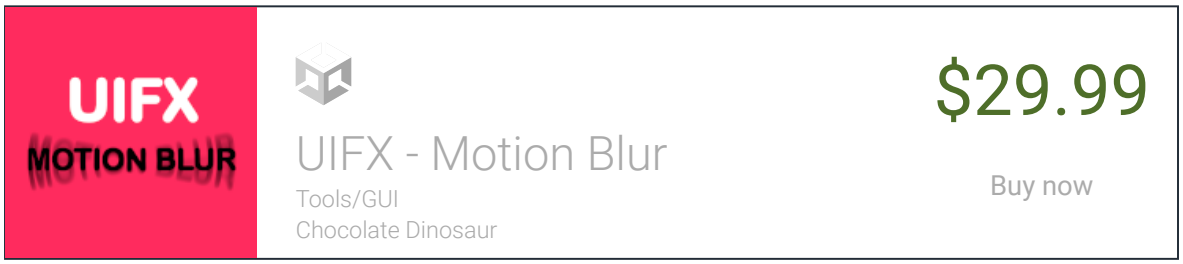
TextMeshPro (TMP) / Custom UI Components

See the notes for `MotionBlurSimple`.

See the notes for `MotionBlurReal`.

Download

`UIFX-MotionBlur` is available for download from the Unity Asset Store.



Rating / Reviews ★★★★☆

Once you've tried this asset, we would really appreciate your rating / review on the Unity Asset Store.

Release Notes

16 December 2023 - Version 1.2.0

New

- New property `FrameRateIndepedent` to allow the motion-blur trail length to look consistent across different frame rates. This can help to maintain a certain artistic look.
- New property `Strength` in `MotionBlurSimple` to control the strength of the effect

Fixed

- `MotionBlurReal` had incorrect interpolation values
- Motion blur effects are now disabled when `Strength` becomes zero

6 December 2023 - Version 1.1.1

Improved

- Added `Help/About` button for easy access to documentation and community links

25 November 2023 - Version 1.1.0

Improved

- Camera movement now contributes to motion blur (requires World Space canvas rendering mode for camera transform)
- Added optimisation to clip the texture processing to the screen edges for `MotionBlurReal` component
- `MotionBlurReal` component optimised to use a single texture instead of two
- `TextMeshPro` support added with new `MotionBlurSimpleTMP` component
- Editor component help buttons now directs to the website documention

Fixed

- Fixed a null exception when adding the component at runtime
- Fixed the 1 frame of latency issue for `MotionBlurReal` component

16 November 2023 - Version 1.0.0

- First release

MotionBlurReal

Component

MotionBlurSimple

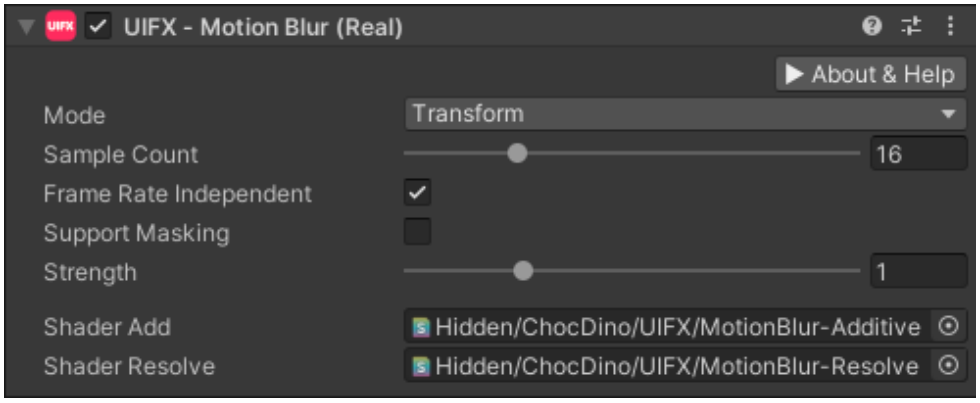
Component

MotionBlurReal Component

Overview

This component adds a realistic motion blur effect to UI objects that are in motion.

Properties



Property	Type	Details
Update Mode	Enum	Which vertex modifiers are used to calculate the motion blur. Which vertex modifiers are used to calculate the motion blur. Options are: `Transforms` - Only use transform modifications (translation, rotation, scale), this is the simplest. (DEFAULT) `Vertices` - Only use vertex modifications. `TransformsAndVertices` - Use both of the above, this is the most expensive.
Sample Count	Int	The number of motion blur steps to calculate. The higher the number the more expensive the effect. Default is 16.
Frame Rate Independent	Bool	Allows frame-rate independent blur length. This is unrealistic but may be more artistically pleasing as the visual appearance of the motion blur remains consistent

Property	Type	Details
		across frame rates. Default is enabled.
Strength	Float	Controls how visible the motion blur effect is by scaling the motion blur length. Setting to zero disables the effect. Above 1.0 will exagerate the length. Default is 1.0.
Support Masking	Bool	This can be enabled if masking support is required, however it requires more memory.
Shader Add	Shader	The additive shader to use while rendering. The default is a modified UI shader, however if you want to use TextMeshPro or a custom UI shader, then see notes below.

Usage

Add this component to any `GameObject` that contains a UI `Graphic` component (eg `Text`, `Image`, `RawImage`, etc). The object will now render with motion blur when it moves.

Quality Notes

- This component produces a much more accurate motion blur than `MotionBlurSimple`, however it is also more expensive.
- `MotionBlurReal` handles transparency much better than `MotionBlurSimple`.

Performance Notes

- The `MotionBlurReal` component can become very slow when the motion traveled in a single frame is very large on screen. This is components performance is relative to the amount of screen space area the Graphic occupies from one frame to the next. That said we have tried to optimise this as much as possible, so even if the motion is larger than the screen area, only the screen area will be computed.

How it Works

Within the `MotionBlurReal` component the following takes place each frame:

1. Store the mesh and transforms for a UI component for the previous and current frames.
2. Generate a new mesh containing multiple copies of the stored meshes interpolated from previous to current mesh.
3. Rendered this mesh additively to a RenderTexture.
4. On the next frame a quad is rendered to the canvas in place of the UI component geometry. This quad uses a shader to resolve the previously rendered motion blur mesh.
5. If no motion is detected then the effect is disabled.

TextMeshPro (TMP) / Custom UI Components

`TextMeshPro` support is coming soon. For now use the `MotionBlurSimple` component for `TextMeshPro` suppport.

Setting up a Custom UI Component

Follow the belows steps for your custom UI component but using the `MotionBlurReal` component. Any custom UI component will work with `MotionBlurReal` as long as it renders using the standard `Graphic` mesh.

We need to create a copy of the shaders that are used by the UI component we want to add motion blur to. We then modify the shaders to make them render using the additive blend mode. We then assign the shader to our motion blur component.

1. Create a new folder for the shaders.

2. To that folder, copy the shaders that you want to use (eg TMP_SDF.shader) and copy any .cginc files it depends on.

3. Open the shader you want to use, and rename the top line to a unique name. For example, from:

   ```
   Shader "TextMeshPro/Distance Field"
   ```

   to

   ```
   Shader "TextMeshPro/Distance Field/Add"
   ```

   to signify that it's an additive version.

4. Replace the line that reads:

```
Blend One OneMinusSrcAlpha
```

with

```
Blend One One, One One
```

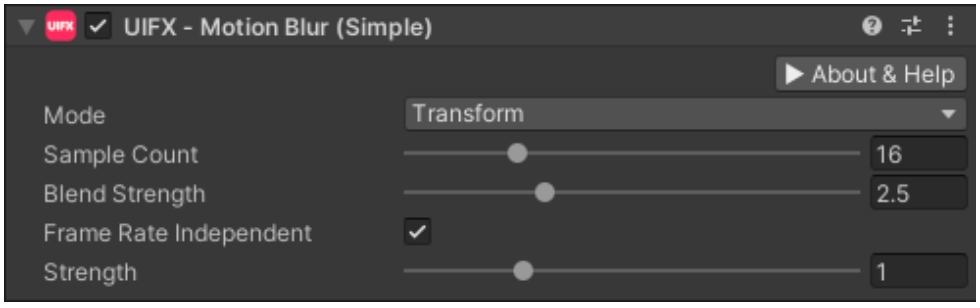
5. Save the file and assign that shader to the `Shader Add` property on the `MotionBlurReal` component.

MotionBlurSimple Component

Overview

This component adds an approximate motion blur effect to UI objects that are in motion.

Properties



Property	Type	Details
Update Mode	Enum	Which vertex modifiers are used to calculate the motion blur. Which vertex modifiers are used to calculate the motion blur. Options are: `Transforms` - Only use transform modifications (translation, rotation, scale), this is the simplest. (DEFAULT) `Vertices` - Only use vertex modifications. `TransformsAndVertices` - Use both of the above, this is the most expensive.
Sample Count	Int	The number of motion blur steps to calculate. The higher the number the more expensive the effect. Default is 16.
Blend Strength	Float	Controls how transparent the motion blur is. Default is 2.5.

Property	Type	Details
Frame Rate Independent	Bool	Allows frame-rate independent blur length. This is unrealistic but may be more artistically pleasing as the visual appearance of the motion blur remains consistent across frame rates. Default is enabled.
Strength	Float	Controls how visible the motion blur effect is by scaling the motion blur length. Setting to zero disables the effect. Above 1.0 will exagerate the length. Default is 1.0.

Usage

Add this component to any `GameObject` that contains a UI `Graphic` component (eg `Text`, `Image`, `RawImage`, etc). The object will now render with motion blur when it moves.

Quality Notes

- This component has faster rendering than the `MotionBlurReal` however it is less accurate. For most cases this approximate is good enough.
- `MotionBlurSimple` is a great approximation for solid objects, but for transparency objects it is less accurate and the `MotionBlurReal` component should be considered. When using this component with transparent objects, the background color can affect this approximation, and so the `BlendStrength` property needs to be set based on the brightness of the object being rendered and the color of the background. The more transparent the object is, the lower the `BlendStrength` value needs to be.

Effects Notes

- When using the built-in `Shadow` component, it will cause problems due to dark transparent layer under opaque layer, which causes flickering. Therefore when using `Shadow` component it's better to put it after this component.

How it Works

Within the `MotionBlurSimple` component the following takes place each frame:

1. Store the mesh and transforms for a UI component for the previous and current frames.

2. Generate a new mesh containing multiple copies of the stored meshes interpolated from previous to current mesh.
3. Replace the UI component mesh with the new motion blur mesh with a reduced per-vertex alpha (`BlendStrength` property).
4. If no motion is detected then the effect is disabled.

TextMeshPro (TMP) / Custom UI Components

The component `MotionBlurSimpleTMP` supports `TextMeshPro` and is easy to use.

Any custom UI component will work with `MotionBlurSimple` as long as it renders using the standard `Graphic` mesh.

Scripting

Code Snippets

The namespace

```
using ChocDino.UIFX;
```

Add the MotionBlurSimple component to your GameObject

```
// Add the component to your GameObject and set default properties
var mb = AddComponent<MotionBlurSimple>();
mb.UpdateMode = MotionBlurSimple.Mode.Transform;
mb.SampleCount = 16;
mb.BlendStrength = 2.5f;
mb.FrameRateIndependent = true;
mb.Strength = 1f;
```

Resetting Motion

```
// Call ResetMotion() whenever you want to reset the motion blur - for example after resetting the position/transform of the object.
mb.ResetMotion();
```

Globals

```
// Global static fields useful for debugging purposes:
// (These affect all instances of MotionBlurSimple at once)

// Disable the effect - useful for low-powered systems.
MotionBlurSimple.GlobalDisabled = true;

```

```
// Add an obvious color tint to the effects - useful to make sure the effect is running.
MotionBlurSimple.GlobalDebugTint = true;

// Stop updating the motion, just display the same frame as last time. This looks like time has frozen - useful for visualising fast moving motion blur.
MotionBlurSimple.GlobalFreeze = true;
```

Complete Example

MotionBlurSimpleExample.cs

```
using UnityEngine;
using ChocDino.UIFX;

/// Demonstrates the scripting API for the MotionBlurSimple component
/// Adds the motion blur component to the gameobject and runs a simple
/// translation and rotation animation on the Transform component.

/// Press keys 1 to 4 to test API functionality.

/// NOTE: The GameObject is required to have a UI component of type Graphic
[RequireComponent(typeof(Graphic))]
public class MotionBlurSimpleExample : MonoBehaviour
{
    [SerializeField] float _speed = 10f;

    private MotionBlurSimple _mb;
    private float _time;
    private float _rotation;

    void Start()
    {
        // Get or create the component
        _mb = GetComponent<MotionBlurSimple>();
        if (_mb == null)
        {
            _mb = AddComponent<MotionBlurSimple>();
        }

        // Set the properties to default values
        _mb.UpdateMode = MotionBlurSimple.Mode.Transform;
        _mb.SampleCount = 16;
        _mb.BlendStrength = 2.5f;
        _mb.Strength = 1f;
```

```
    }

    void Update()
    {
        AnimateTransform();

        // Keys 1..4 demonstrate API functionality
        if (Input.GetKeyDown(KeyCode.Alpha1))
        {
            // Toggle disabling
            MotionBlurSimple.GlobalDisabled = !MotionBlurSimple.GlobalDisabled;
        }
        else if (Input.GetKeyDown(KeyCode.Alpha2))
        {
            // Toggle debug color tinting
            MotionBlurSimple.GlobalDebugTint = !MotionBlurSimple.GlobalDebugTint;
        }
        else if (Input.GetKeyDown(KeyCode.Alpha3))
        {
            // Toggle time freezing
            MotionBlurSimple.GlobalFreeze = !MotionBlurSimple.GlobalFreeze;
        }
        else if (Input.GetKeyDown(KeyCode.Alpha4))
        {
            // Restart
            _time = 0f;
            _rotation= 0f;
            _mb.ResetMotion();
        }
    }

    void AnimateTransform()
    {
        _time += Time.deltaTime * _speed;

        // Animate position
        float x = Mathf.Sin(_time) * Screen.width;
        this.transform.localPosition = new Vector3(x, 0f, 0f);

        // Animate rotation
        float maxSpinSpeed = 5f;
        float spinSpeed = Mathf.Sin(_time) * maxSpinSpeed;
        _rotation += spinSpeed;
        this.transform.localRotation = Quaternion.Euler(0f, 0f, _rotation);
    }
}
```

About UIFX - Trail

Overview

What?

`UIFX-Trail` is a new visual effect for Unity's UI (uGUI) components that adds a trail that follows the motion of the component.

Why?

Unity's UI system currently doesn't have an option to render with a trail.

Trails can be beautiful and add an element of fun, so we created this component to be easy to use, yet have powerful options.

How?

Just add the `UIFX-Trail` component to your UI component. That's it!

The component will track the full motion of the object (translation, rotation and scale) either at a transform level, or at a per-vertex level. A new mesh is then generated based on the motion, and this is rendered in place of the original UI geometry. The effect is only applied when the object is in motion.

Features

- Easy to use
- Highly optimised
- Cross-platform
- Built-in/URP/HDRP
- Supports TextMeshPro
- Source code included
- Well documented
- Well supported
- API documentation

Compatibility

Unity Version	2023.x	2022.x	2021.x	2020.x	2019.x	2018.x	2017.x
Supported	✓	✓	✓	✓	✓	✗	✗

Platform	Windows/UWP	macOS	Linux	Android	iOS / tvOS	WebGL	PS4 / PS5	XBox One
Supported	✓	✓	✓	✓	✓	✓	✓	✓

Render Pipeline	Built-in	URP	HDRP
Supported	✓	✓	✓

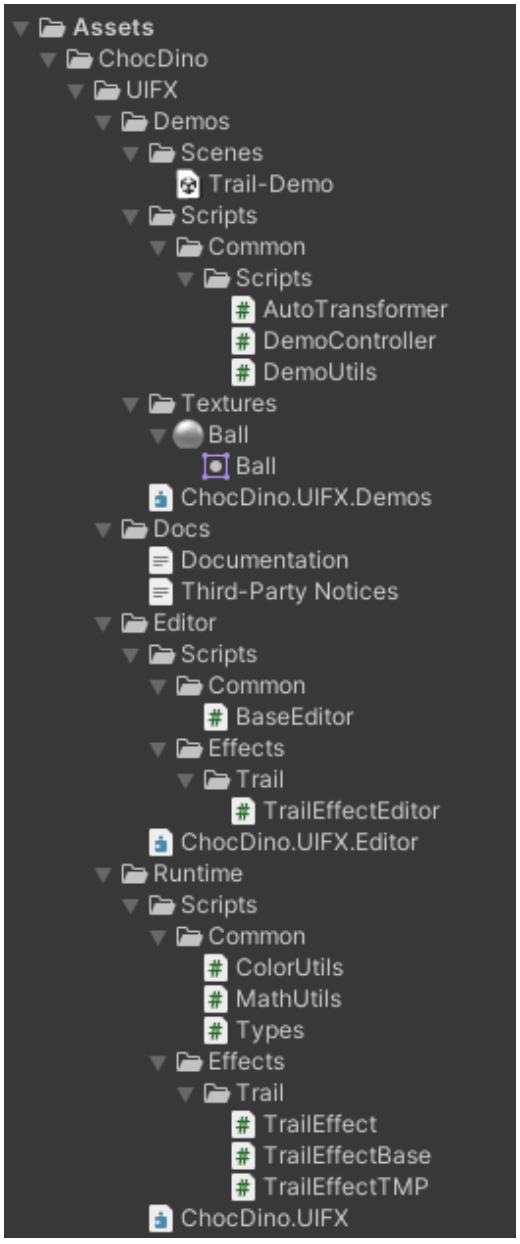
Dependencies

This plugin requires the `Unity.UI` package that is included with Unity by default.

If you want to use it with `TextMeshPro` then it requires version `2.2.0-preview.3` or newer. In the package manager you may need to manually switch to one of the newer preview versions by using package name: `com.unity.textmeshpro` and version: `3.2.0-pre.6`. For Unity 2022 or higher, you can use version: `4.0.0-pre.2`.

The Asset Package

The asset package consists of the following files:



A demo scene is included. All scripts are grouped using Assembly Definition files.

Components

This Unity asset includes two components:

- `TrailEffect` - Adds a visual trail to UI components
- `TrialEffectTMP` - Same as `TrailEffect` but specifically for `TextMeshPro`

Supported Features

UI Feature	Supported	Notes
Components:		
Text	✓	
Image	✓	
RawImage	✓	
Mask	✓	
Rect Mask 2D	✓	
CanvasGroup	✓	
TextMeshPro	✓	
Custom	✓	
Canvas Render Modes:		
Screen-Space Overlay	✓	
Screen-Space Camera	✓	
World Space	✓	
Render Pipelines:		
Built-in	✓	
URP	✓	
HDRP	✓	
Effects:		
Shadow	✓	`UIFX-Trail` renders with flickering if Shadow effect is applied first
Outline	✓	`UIFX-Trail` renders with flickering if Outline effect is applied first

UI Feature	Supported	Notes
Colorspaces:		
Gamma	✓	
Linear	✓	

Advanced

Performance

`UIFX-Trail` has had several optimisation passes, so we can say with confidence that it is highly performant. That said, it does rebuild a new mesh each frame, so some consideration is still necessary.

Garbage Collection

This component has been optimised to reduce garbage generation.

Download

`UIFX-Trail` is available for download from the Unity Asset Store.

Rating / Reviews ★★★★☆

Once you've tried this asset, we would really appreciate your rating / review on the Unity Asset Store.

Release Notes

6 December 2023 - Version 1.1.1

Improved

- Added `Help/About` button for easy access to documentation and community links
- Damping now has separate values for front and back of the trail which gives more control over the trail appearance
- Demo scene improved

Changed

- `Damping` property renamed to `DampingFront`
- Added new `DampingBack` property

22 November 2023 - Version 1.1.0

Improved

- Added support for running in edit mode
- Added better handling of dynamic text/geometry
- Matrix lerp now preserves scale and is more optimised
- Big improvements to TextMeshPro support
- Demo scene improved

Changed

- Texture coordinates are no longer interpolated, which fixes artifacts with dynamic text
- Scripts using TeshMeshPro moved into separate Assembly Definition

Fixed

- Animation no longer updated when there are no vertices
- Changing `VertexModifierSource` while running no long causes an exception

16 November 2023 - Version 1.0.0

- First release

TrailEffect

Component

TrailEffectTMP

Component

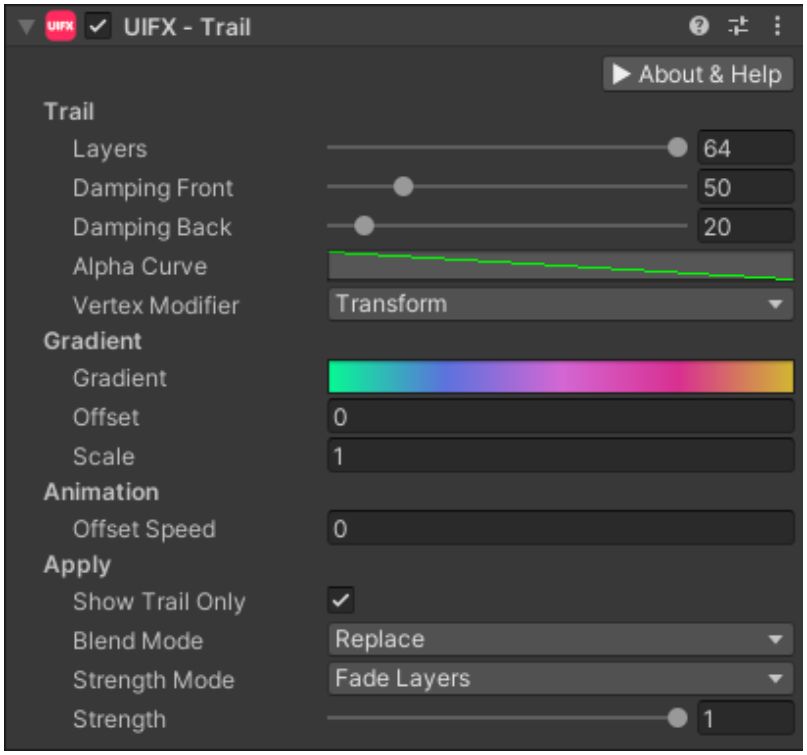
UIFX TRAIL EFFECT

TrailEffect Component

Overview

This component adds a visual trail effect to the UI object it is applied to.

Properties



Property	Type	Details
Trail		
Layers	Int	The number of layers the trail has. More layers is more expensive. Default value is 16, range is [0..64].

Property	Type	Details
Damping Front	Float	The rate at which the front of the trail catches up with the movement. Higher value results in a less laggy trail. Default value is 50, range is [0..250].
Damping Back	Float	The rate at which the back of the trail catches up with the movement. Higher value results in a less laggy trail. Default value is 50, range is [0..250].
Alpha Curve	Curve	Optional curve to control transparency. Transparency can also be controlled by the gradient property, but having this secondary control is useful when the gradient is animated but you still want to apply a static transparency falloff.
Vertex Modifier	Enum	Which vertex modifiers are used to calculate the trail motion. Options are: `Transforms` - Only use transform modifications (translation, rotation, scale), this is the simplest. (DEFAULT) `Vertices` - Only use vertex modifications. `TransformsAndVertices` - Use both of the above, this is the most expensive.
Gradient		
Gradient	Gradient	The gradient colors used by the trail
Offset	Float	The offset applied to the gradient. The gradient will wrap using mirrored repeating.
Scale	Float	The scaling applied to the gradient. The gradient will wrap using mirrored repeating.
Animation		
Offset Speed	Float	The animation speed for the offset property of the gradient. Allows easy simple scrolling animation without scripting. Set to zero for no animation.
Apply		

Property	Type	Details
Show Trail Only	Bool	Only show the trail, hide the original UI Graphic
Blend Mode	Enum	Which color blending mode to use to mix the original vertex colors with the gradient colors. Options are: `Source` - Only use the original color, this ignores any trail gradient/alpha settings. `Replace` - Ignore the original color and replace with the trail gradient/alpha settings. `Replace_Multiply` - Same as `Replace` for RGB, but multiply the original alpha with the trail gradient alpha. `Multiply` - Multiply the original color with the trail gradient/alpha settings. (DEFAULT) `Add_Multiply` - Add the original color RGB to the gradient gradient, but multiply the alpha value.
Trail Strength Mode	Enum	The mode to use for fading out the trail when strength < 1.0. Options are: `Damping` - Reduce damping so that when strength == 0.0 there is no lag in the trail. `Layers` - Remove each layer, starting from the back so that when strength == 0 there are no layers visible. `FadeLayers` - Same as `Layers` but with fading instead of a hard cut. (DEFAULT) `Fade` - Fade the entire trail down at the same time.
Strength	Float	Strength of the effect. Default value is 1.0, range is [0..1]

Usage

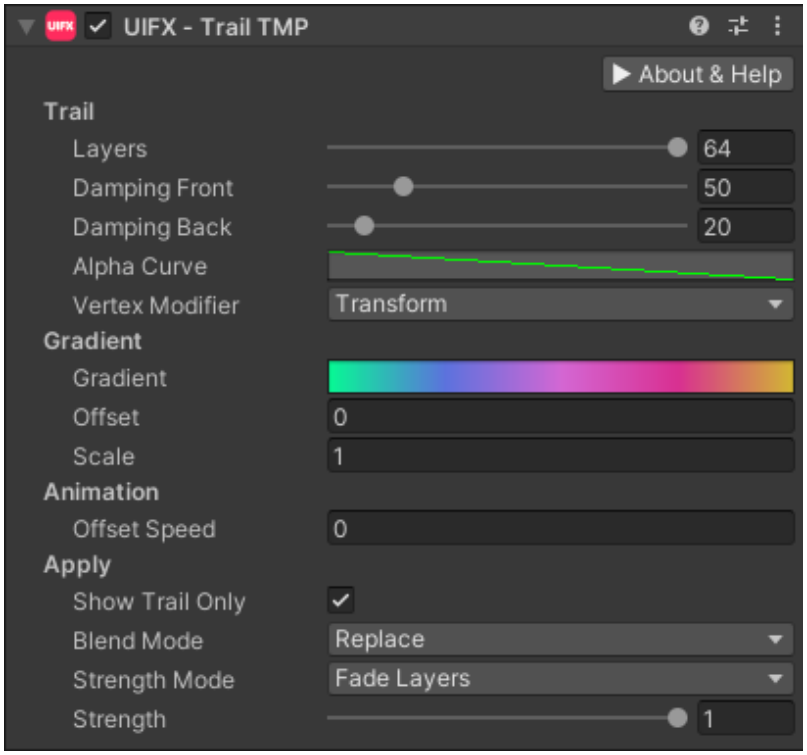
Add this component to any `GameObject` that contains a UI `Graphic` component (eg `Text`, `Image`, `RawImage`, etc). The object will now render with a trail that follows it as it moves.

TrailEffectTMP Component

Overview

This component adds a visual trail effect to the `TextMeshPro` text component it is applied to.

Properties



Property	Type	Details
Trail		
Layers	Int	The number of layers the trail has. More layers is more expensive. Default value is 16, range is [0..64].

Property	Type	Details
Damping Front	Float	The rate at which the front of the trail catches up with the movement. Higher value results in a less laggy trail. Default value is 50, range is [0..250].
Damping Back	Float	The rate at which the back of the trail catches up with the movement. Higher value results in a less laggy trail. Default value is 50, range is [0..250].
Alpha Curve	Curve	Optional curve to control transparency. Transparency can also be controlled by the gradient property, but having this secondary control is useful when the gradient is animated but you still want to apply a static transparency falloff.
Vertex Modifier	Enum	Which vertex modifiers are used to calculate the trail motion. Options are: `Transforms` - Only use transform modifications (translation, rotation, scale), this is the simplest. (DEFAULT) `Vertices` - Only use vertex modifications. `TransformsAndVertices` - Use both of the above, this is the most expensive.
Gradient		
Gradient	Gradient	The gradient colors used by the trail
Offset	Float	The offset applied to the gradient. The gradient will wrap using mirrored repeating.
Scale	Float	The scaling applied to the gradient. The gradient will wrap using mirrored repeating.
Animation		
Offset Speed	Float	The animation speed for the offset property of the gradient. Allows easy simple scrolling animation without scripting. Set to zero for no animation.
Apply		

Property	Type	Details
Show Trail Only	Bool	Only show the trail, hide the original UI Graphic
Blend Mode	Enum	Which color blending mode to use to mix the original vertex colors with the gradient colors. Options are: `Source` - Only use the original color, this ignores any trail gradient/alpha settings. `Replace` - Ignore the original color and replace with the trail gradient/alpha settings. `Replace_Multiply` - Same as `Replace` for RGB, but multiply the original alpha with the trail gradient alpha. `Multiply` - Multiply the original color with the trail gradient/alpha settings. (DEFAULT) `Add_Multiply` - Add the original color RGB to the gradient gradient, but multiply the alpha value.
Trail Strength Mode	Enum	The mode to use for fading out the trail when strength < 1.0. Options are: `Damping` - Reduce damping so that when strength == 0.0 there is no lag in the trail. `Layers` - Remove each layer, starting from the back so that when strength == 0 there are no layers visible. `FadeLayers` - Same as `Layers` but with fading instead of a hard cut. (DEFAULT) `Fade` - Fade the entire trail down at the same time.
Strength	Float	Strength of the effect. Default value is 1.0, range is [0..1]

Usage

Add this component to any `GameObject` that contains a UI `TMP_Text` component. The text will now render with a trail that follows it as it moves.

Scripting

Code Snippets

The namespace

```
using ChocDino.UIFX;
```

Add the TrailEffect component to your GameObject

```
// Add the component to your GameObject and set default properties
// NOTE: Change this to TrailEffectTMP for TextMeshPro
var trail = AddComponent<TrailEffect>();

Gradient gradient = new Gradient();
{
    GradientColorKey[] keys = new GradientColorKey[2];
    keys[0].time = 0f;
    keys[0].color = Color.white * 0.8f;
    keys[1].time = 1f;
    keys[1].color = Color.white * 0.8f;
    GradientAlphaKey[] alpha = new GradientAlphaKey[2];
    alpha[0].time = 0f;
    alpha[0].alpha = 1f;
    alpha[1].time = 1f;
    alpha[1].alpha = 0f;
    gradient.SetKeys(keys, alpha);
}
trail.Gradient = gradient;

trail.LayerCount = 16;
trail.DampingFront = trail.DampingBack= 50f;
trail.AlphaCurve = new AnimationCurve(new Keyframe(0f, 1f, -1f, -1f), new Keyframe(1f, 0f, -1f, -1f));  // AlphaCurve is optional and can be null
trail.VertexModifierSource = VertexModifierSource.Transform;
```

```
trail.GradientOffset = 0f;
trail.GradientScale = 1f;
trail.GradientOffsetSpeed = 0f;
trail.ShowTrailOnly = false;
trail.blendMode = BlendMode.Multiply;
trail.StengthMode = TrailStrengthMode.FadeLayers;
trail.Strength = 1f;
```

Resetting Motion

```
// Call ResetMotion() whenever you want to reset the trail - for example after resetting the position/transform of the object.
trail.ResetMotion();
```

Complete Example

TrailExample.cs

```
using UnityEngine;
using ChocDino.UIFX;

/// Demonstrates the scripting API for the TrailEffect component

/// Press keys 1 to 5 to test some API functionality.

/// NOTE: The GameObject is required to have a UI component of type Graphic
[RequireComponent(typeof(Graphic))]
public class TrailExample : MonoBehaviour
{
    private TrailEffectBase _trail;

    void Start()
    {
        // Get or create the component, detecting whether to use TextMeshPro or not
        _trail = GetComponent<TrailEffect>();
        if (_trail == null)
        {
            #if TMP_PRESENT
            if (GetComponent<TMP_Text>() != null)
            {
                _trail = GetComponent<TrailEffectTMP>();
                if (_trail == null)
                {
```

```
                    _trail = AddComponent<TrailEffectTMP>();
                }
            }
            else
            #endif
            {
                _trail = AddComponent<TrailEffect>();
            }
        }
    }

    void Update()
    {
        // Keys 1..5 demonstrate some API functionality
        if (Input.GetKeyDown(KeyCode.Alpha1))
        {
            _trail.LayerCount = 16;
        }
        else if (Input.GetKeyDown(KeyCode.Alpha2))
        {
            _trail.LayerCount = 64;
        }
        else if (Input.GetKeyDown(KeyCode.Alpha3))
        {
            _trail.GradientOffsetspeed = 0f;
        }
        else if (Input.GetKeyDown(KeyCode.Alpha4))
        {
            _trail.GradientOffsetspeed = 0.2f;
        }
        else if (Input.GetKeyDown(KeyCode.Alpha5))
        {
            _trail.ResetMotion();
        }
    }
}
```

Support

We believe that software is only as good as its support and maintenance.

If you're having a problem with one of our products or have a question, please get in touch via one of the methods below:

Customer Support

- Issues: Github
- Community: Discord
- Email: support@chocdino.com