



Vibration Shader

v1.0
Documentation

Contents

1	Introduction	2
2	Documentation	3
2.1	Universal Render Pipeline (URP)	3
2.1.1	Universal Renderer	3
2.1.2	2D Renderer	8
3	Support	9

Section 1

Introduction

The Vibration shader allows you to make sprite elements shake, wobble, and tremble by displacing their vertices over time. The vibration's strength, speed, and the degree to which neighbouring vertices fall out of sync are fully customizable.

WARNING

This shader supports only Universal Render Pipeline (URP) on Unity 6.0+. Built-in Render Pipeline (BIR) and High-Definition Render Pipeline (HDRP) are not supported.

A controller script is included for dynamically changing the shader variables, creating dynamic effects, such as a vibration that builds up from rest to a violent shake, or a vibrating element that changes its color as it trembles.

You can stack these effects on top of each other by adding several or all effects to the same element, creating unique vibration effects.

Feel free to check out the demo scenes included in the asset for inspiration.

Section 2

Documentation

2.1 Universal Render Pipeline (URP)

2.1.1 Universal Renderer

The Universal Renderer is the default URP Renderer when creating a 3D project. Please use the **/Universal Renderer** folder exclusively for this Renderer.

Setup

The following paragraph presents the steps necessary to make the Vibration shader work in your project.

Visit <https://docs.project-gamedev.com/vibration-shader/setup.html> for image-aided instructions.

The Vibration shader is, in principle, the same as the regular lit shader you would use in your Unity project, only reconstructed and expanded. The setup is thus the same as with any other lit shader:

- Create a new material for your object(s).
- Select the material; then, select **Project G.A.M.E.D.E.V./Essential 2D Shaders/Vibration Shader** as the shader source for this material.
- Assign the new material to your object(s).

That's it! The shader is now active. Feel free to play around with the material properties to find a vibration effect that you are happy with.

Keep in mind that certain post-processing settings (most notably **Bloom** and **Tonemapping**) will have a profound effect on the emissive parts of your vibrating sprite.

i NOTE

The vibration is produced by displacing the mesh vertices of your sprite. Sprites with very few vertices therefore vibrate more coarsely than sprites with a denser mesh. If the effect looks blocky, consider increasing the mesh detail of the sprite in its import settings.

In case you wish to animate the vibration effect, attach `VibrationController.cs` to an object in your scene. Consult with 2.1.1 for more information on the controller script.

Shader Options

Albedo The main (albedo/color) texture used by the shader.

Albedo Color An additional color multiplied with the one already present in the albedo texture. Use this to further customize the look of your sprite.

Alpha Cutoff This is a threshold value – all parts of the texture whose alpha value is less than the alpha cutoff will not be rendered.

Ambient Occlusion (Universal Renderer only) The regular ambient occlusion from the standard URP Lit shader.

Metallic Map (Universal Renderer only) The standard metallic map option in the default URP Lit shader. The metallic map is a black-and-white texture that determines the surface type of the object. It is used to make certain parts of the sprite reflect light like a metal would (metallic), and others like sand would (smooth).

Metallic (Universal Renderer only) This value determines how “metallic” the surface should be, i.e., whether it should reflect light like a metal or a wet surface, or should absorb it like sand or another rough, non-metallic surface.

Normal Map The standard normal map option in the default URP Lit shader. The normal map is used to simulate a 3D-surface by containing information on how the light should be reflected by the surface.

Normal Strength This value is multiplied with the normal map and is used to make the effect stronger or weaker. A negative value will mirror the 3D-effect (i.e., protruding surfaces will now be concave, and vice versa).

Emission Map The standard emission map option in the default URP Lit shader. A black-and-white map that determines which parts of the sprite will emit light. The white pixels of the texture are areas where the emission will have an effect, and the black pixels are the areas with no emission.

Emission Color The emission color as an HDR color. Keep in mind that **different post-processing effects (most notably bloom)** will have a **profound** effect on the look of your emission.

Emission Extra Intensity Since the default HDR color has a limited intensity (-10 to 10), this shader offers an additional intensity option that allows you to make your emission even stronger.

Vibration Amount The strength of the vibration, i.e., how far the vertices of the sprite are displaced from their resting position. Set to 0 to bring the sprite completely to rest. Larger values produce a more violent shake.

Vibration Speed How quickly the vibration oscillates. Set to 0 to freeze the displacement in a single, static deformation. Larger values make the sprite tremble faster.

Vibration Volatility Controls how far apart in phase neighbouring vertices vibrate. A value of 0 makes the whole sprite pulse in sync, so the sprite keeps its shape and moves as a single body. Increasing the value spreads neighbouring vertices further apart across the wave cycle, which distorts and warps the sprite instead of merely moving it.

i NOTE

The three vibration properties interact strongly with each other and with the vertex density of your sprite, so there is no single "correct" set of values. The demo scene ships with several materials covering everything from a subtle tremble to a heavy warp – use them as starting points.

API

Visit <https://docs.project-gamedev.com/vibration-shader/api.html> for the Vibration Shader API documentation. This API allows you to modify all shader values in-code, from your own scripts, if necessary.

Controller Script

A controller script – `VibrationController.cs` – is included for indefinitely lerping the Vibration effect on a sprite element. If you wish to set up repeating effects, attach this script to any object in your scene, and set it up with the elements and effects you wish to enable.

i NOTE

The controller script accepts any `GameObject` elements and searches them for a `SpriteRenderer` component.

The controller script is split into the following sections, visible in the Editor:

General The General section contains a single setting – **Individual Elements**. Enable this setting to have all effects applied to the same elements. If you disable this option, you will need to specify (different or the same) elements for each effect you enable!

Vibration Amount This is the main vibration effect. Enabling this section will let you customize how the strength of the vibration changes over time – for example, letting a sprite build up from complete rest to a violent shake and back.

Further Effects Each shader property can be lerped independently by enabling its section. The available effects are **Vibration Speed Controller**, **Vibration Volatility Controller**, **Color Controller**, **Emission Color Controller**, and **Emission Strength Controller**. For example, you can make a sprite tremble faster and faster by enabling the Vibration Speed Controller effect, or make it glow brighter as it shakes by setting up the Emission Strength Controller effect. Feel free to experiment with different effects and values for each effect.

Every effect shares the same set of settings: a **start value** and an **end value** for the shader property, the **smoothness** (i.e., the amount of steps) of the effect, the **duration** in seconds needed to go from the start value to the end value, and the **pause duration** in seconds between consecutive effects.

WARNING

You can have multiple instances of the `VibrationController.cs` script in your scene, and different effects can target the same elements. However, ensure **different instances of the script do not target the same elements!**

For example, it is fine to have `VibrationController 1` target elements A and `VibrationController 2` target element B, but do not have both `VibrationController 1` and `VibrationController 2` target element A!

NOTE

When enabling several effects at once, it is fine to target the same elements **from within the same `VibrationController`**.

For example, it is fine to have the Vibration Amount effect target elements A and B, and the Vibration Speed Controller effect target element A as well. Just make sure both effects are **inside the same `VibrationController` script!**

2.1.2 2D Renderer

The 2D Renderer is the default URP Renderer when creating a 2D project. Please use the **/2D Renderer** folder exclusively for this Renderer.

Setup

The setup is the same as described in subsection 2.1.1. The only difference is that you should select **Shader Graphs/Vertex Displacement Shader - 2D Renderer** as the shader source for your materials.

Shader options

The shader options are a subset of 2.1.1. Keep in mind several options are not available on the 2D Renderer as they are not supported there, namely **Ambient Occlusion**, **Metallic Map**, and **Metallic**.

NOTE

Since the 2D Renderer combines emission and albedo color in a single channel, keep in mind that your materials will look differently on the 2 different Renderers.

API

Similarly to the shader options, the API for the 2D Renderer version of this shader is a subset of 2.1.1, with **Ambient Occlusion**, **Metallic Map**, and **Metallic** missing.

Controller Script

Please consult with 2.1.1.

Section 3

Support

In case you experience any problems with the shader, can't manage to set it up properly, or would simply like to propose an improvement, don't hesitate to contact us at contact@project-gamedev.com, or fill out the form at <https://project-gamedev.com/contact.html>.

If you are satisfied with this product, please consider leaving a review on the Unity Asset Store page (<https://assetstore.unity.com/packages/slug/396936>).

Best of luck with your game development journey!