master 2 branches 1 tag Go to file Code

peeweek Updated Project to 2019.4 bc4a598 on Sep 21, 2020 3 commits

ShurikenMaster	Updated Project to 2019.4	17 months ago
.gitignore	Updated Project to 2019.4	17 months ago
LICENSE.md	Initial Commit	4 years ago
README.md	Initial Commit	4 years ago

README.md

"ShurikenMaster" Experimental Shaders

These shaders are experimental shaders intended to be used with Shuriken Particle Systems. They have been tested on PS4 and PC-DX11 platforms so far, using unity 5.6

They are not designed directly for being used on mobile platforms so use them at your own risk in this field :)

###Why diverge from standard particle shader?

Standard particle shader uses a Diffuse/Metallic/Roughness/Normal BRDF to define surfaces and thus it is not suited for many particle effects (Smoke, Steam, Fire, Electricity, etc..) This set of shaders is an attempt of additions in order to provide a good coverage set for particles.

I highly suggest you also download the standard particle shader from official unity forums.

Can I use these in production?

As these shaders are still in experimental state, they have not proven themselves in battle. If you still intend to use them in production I strongly advise you run a lot of tests before deciding to use them.

I strongly advise not to use them in the following cases:

- Your project is in gamma space (DO NOT USE THESE SHADERS)
- You target low/mid end mobile and want lit particles
- Your team is not skilled enough to perform optimizations suiting your needs.

What if I need a more standard BRDF?

For a particle shader that relies on Diffuse/Specular/Roughness/Normal, please see the experimental forums section.

Are these free and open?

They are released under Simplified BSD 2-Clause License

All the sample graphics assets released as example follow the Creative Commons Zero License

Contents

This project includes the following shaders:

- **ShurikenMaster** : All-Purpose particle shader for anything not specific (steam, fire, smoke).
- **ShurikenFire** : Unlit, Fire-Dedicated shader based on temperature maps and gradient mapping.
- **ShurikenLiquid** : Reflective and Refractive shader mainly used for liquids
- **ShurikenCloud** : Variation of the ShurikenMaster Lit shader intended to be used for clouds.

Disclaimer & Usage

- These shaders are meant to be used with shuriken in unity versions 5.6 and newer. Due to vertex stream format change after 5.5, compatibility cannot be assured anymore.
- These shaders are an experimental attempt to diverge from the Unity Standard Surface BRDF as it is a surface BRDF and is not suited for smoke, steam and fire.
- The lighting functions used in these shaders have been **truncated** and **approximated** to ensure a performance/quality balance. Still, many features are expensive and are not suited for rendering on mobile platforms.
- The shaders have been designed with Linear Rendering in mind and do not support Gamma rendering.

Shader Configuration

Shader Complexity

The Shader Complexity gauge expresses the relative cost of your effect, based on features enabled on your material. This gauge is totally arbitrary and is meant to orient your decisions towards enabling features, or not.

Please remember at all times that the actual cost depends mostly on which target renders your effects, the amount of surface drawn on screen and some rendering features such as HDR. These external costs are not evaluated in the shader complexity gauge.

Blending

The **shaders** use different blend modes that can differ from the unity standard names, here is a recap of all the available modes.

Blend Mode	Description
Cutout	Non-Blended binary opacity (use of an alpha threshold)
Alpha Blend	Alpha blended standard translucency : use of the alpha as a blend mask to display color.
Additive	Additive blending brightening every layer.
Premultiplied Alpha	Pre-Multiplied Translucency : use of the alpha channel as an occluder regardless of the RGB values (RGB is additive, alpha premultiplies to the background). This mode enables displaying in both Additive and AlphaBlend translucency
Dithered	Dithered binary opacity using a screen-space bayer dithering pattern.
Modulate	Multiplicative blending for lighting effects

Lighting

The lighting used for particles is made of approximations that are *quite far from unity standard lighting* and correspond more to an artist approach than the unity standard model that suits more a surface model.

Lighting Mode	Description
Unlit	No lighting is applied
Dynamic Per-Vertex	Lighting is computed per-vertex depending on the vertex normal then multiplied to the particle color.
Dynamic Per-Pixel	Lighting is computed per-pixel depending on the vertex normal (or pixel normal map, if provided)
Light Probe Proxy Volume	Lighting Values are gathered from the lightprobe proxy volume attached to the particle system.

- All Lit modes include Ambient SH computation, based on lighting settings of the scene.
- Per-Vertex and Per-Pixel lighting computes the lighting for one directional light and the 4 most influent point lights (use Light component's Render Mode to Important to flag ensure lights are influent
- Point Lights and Directional light attenuation are used for particle lighting, but particles does **not** receive these lights' shadows.

Lighting Directionality

Directionality controls the amount of light being purely directional (as It would be for a solid particle, for example a rock) compared to the amount of light being purely non-directional as a thin steam would scatter light.

- In the case of being purely directional the light attenuation is computed depending on the light radius attenuation and the normal vector of the particle (vertex normal or normal map).
- In the case of being purely non-directional, the light attenuation is computed depending solely on the light radius attenuation
- The values In-between are a simple lerp between these two results. Adjusting the value can make your particle render cheap fake scattering for your lights.

Main Options

These options are the common ones when it comes to configure a material.

- **Color Map**: The main map used for the particle, will be mainly used for color and alpha source for the particles
- **Brightness (HDR)** : A color Scale multiplier applied to the color of the particles in order to dim the luminosity, but mostly to brighten particles and make them bloom if post-processes are present.
- **Color Map Type** : A popup that lets you select the texture setup you want to use for this material

Option	Description
RGBA From Color Map	This default preset reads the Color Map and uses RGB for color and Alpha for transparency
Alpha from Color Map	This preset uses only the Alpha channel of the Color Map and assumes White RGB. This preset is useful if either you do not want to use RGB Values or your ColorMap is Single Channel (A8)
RGB From Color Map and Alternate Alpha	This presets reads only RGB Values from Color Map and Reads an alternate Alpha Map alpha channel for transparency. This case is extremely useful in the case where you want to compress HDR Color using BC6H. In this case the alpha channel has to be inside a separate texture.

Other Options

In the Other options rollout you will find toggleable features for your shader.

- **Soft Particles** enables Soft Particle Fading for your material, you can also set the fade distance which corresponds roughly to your particle thickness
- **Camera Fade** enables fading whenever your particle comes close to the camera. The particle will fade between the near distance (zero opacity) and the far distance (normal opacity). Whenever the particle is below the near distance, it becomes clipped to save performance.
- **Flipbook Blending** activates alternate flipbook reading mode in order to interpolate between your flipbook texture sheet frames, smoothening the animation. **Optical Flow Motion Vectors** blending is also available yet not tested extensively at the moment.

Vertex Streams

Whenever you configure your material, It will require sometimes that shuriken provide extra vertex streams (for instance Tangents if you use normal map or animation blend factor and secondary UV if using flipbook blending). To adjust these streams in all scene components, click the Apply to Systems button.

Log

The log sums up all relevant information that needs to be displayed to the user.

Variant-Specific Features

Shuriken Fire

In Shuriken Fire, the **Color map** is used as a **temperature map**: as such, the **Source** popup lets you choose how to interpret the color:

- **Linear Luma** : Interprets directly the linear values of the color as temperature.
- **SRGB Luma** : Converts color brightness into values used as temperature.
- **Alpha** : Uses the Alpha channel as Temperature.

The **Scale** slider lets you adjust the temperature scale as a simple multiplier.

The **Fire Gradient Map** texture slot is expecting a Horizontal Gradient containing fire colors. The texture requires address mode to Clamp.

The **Alpha Scale** is used to adjust the final opacity of the particle.

Shuriken Liquid

Shuriken Liquid adds **reflective** and **refractive** lighting to your particles along with a **murkiness factor** in order to create from crystal clear to trouble liquids.

- **Reflection** is achieved using a 4-Point Light and 1 Directional Light setup close to the Shuriken Master setup. The reflections are computed using a basic and approximate gloss function.
- **Refraction** is achieved through a single *GrabPass* before the first draw call of Shuriken Liquid Shader. The Refraction factor enables more refractivity of your liquids. Please note that the refraction is approximate and in Screen-space, and does not take into refraction all draw calls past the first refracted object.
- **Murkiness** enables making your particle trouble and uses a 4-Point light non-directional setup to light the particle.

Shuriken Cloud

Lighting Mode has been replaced by a different popup:

- **Fake Directional** enables a new Option where the user can define a fake global sun with the following:
  - Color : the tint of the sun color
  - Brightness : The intensity multiplier of the sun
  - Direction : The world-space vector that describes the sun direction.
- **Dynamic Per-Pixel** uses a 4-pointlight + 1 DirectionalLight setup to compute the attenuation.
- **Raymarch 2D** : HIGHLY EXPERIMENTAL - Assumes the particle sprite is a cube and the alpha represents a thickness. Raymarches in 2D space to calculate volumetric occlusion. Highly Experimental and probably more expensive than you can ever afford.