Sunshine!

Index

Getting Started

Getting up-and-running is usually painless, just follow these steps:

- Drag the Sunshine Prefab, located in *Sunshine/Prefabs/*, into your hierarchy.
- Add the *SunshineCamera* component to your primary camera(s).
- Select the Sunshine Prefab instance, and choose your primary Directional Light.
- If you wish to use Custom Shadow Mapping, enable the *Custom Shadow Mapping* setting.
  - If you're using the Deferred Renderer, simply restart the Unity Editor to ensure that the new deferred shader is loaded.
  - If you're using the Forward Renderer, press the *Install Forward Shaders* button.
- Run your project!

You should now see the *Volumetric Scattering* effect lighting your scene, and custom soft shadows if you are using *Custom Shadow Mapping*.

Settings

All of Sunshine's settings are accessible from the Sunshine Prefab you've dragged into your Hierarchy.

- **General Settings**
  - *Sun Light* - Sets the primary Directional Light in your scene.
  - *Occluders Per-Cascade* - Allows specifying separate occluder masks for each cascade
  - *Occluders* - Sets a LayerMask specifying which layers should occlude light.
  - *Light Resolution* - Sets the resolution of Sunshine's Lightmap.
    - *Custom Resolution* - Manually sets the Light Resolution.
  - *Light Distance* - The maximum distance that Light and Shadows will be drawn.
  - *Cascade Count* - The number of Light Cascades.
  - *Cascade Spacing* - A ratio of the total distance to split each Light Cascade at.
  - *Cascade Fade* - Amount to fade between cascades. For performance reasons, fading between Cascades is randomly jittered.
  - *Debug View* - Allows you to display Sunshine's Status, visualize the Cascade Splits, or visualize Volumetric Scattering.
  - *Advanced Settings*
    - *Occlusion Culling* - Toggles Occlusion Culling for Lightmap Cameras.
    - *Padding Z* - Distance to move the Lightmap Cameras back to ensure that far away objects aren't clipped. Try increasing if you run into trouble with tall buildings/mountains/etc.
    - *Refresh Interval* - Controls how often Sunshine updates the Lightmap.
      - *Every Frame* - Refreshes the Lightmap every frame, as you would expect.
      - *After X Frames* - Refreshes the Lightmap every X frames
        - *Frame Interval* - The update interval. Setting this to *2* will update every other frame.
        - *Bounds Padding* - The amount to increase the Lightmap bounds, to allow for slight movement between updates.
      - *After X Movement* - Updates the Lightmap after the camera has moved X distance. This can be useful for mobile games, where you want a replacement for static shadows.
        - *Movement Interval* - Amount the camera has to move before the Lightmap is updated.
      - *Manual* - Only updates the Lightmap when SunshineCamera.RequestRefresh() is called.
    - *Custom Bounds* - Specify the position and radius the Lightmap should cover. Useful for "baking" Lightmaps at runtime, or Character-Only shadows in Mobile games!
      - *Custom Radius* - Specify a *Radius* that fully encloses your *Player*. You'll want to be sure the player never reaches the extents, or you see nasty artifacts.
      - **Notes:**
        - You will need to use a *Custom Lightmap Resolution*.
- **Volumetric Scattering** - A Volumetric Lighting effect achieved by ray-marching through Lightmaps.
  - *Color* - Color of the dust/particulates in the air.
  - *Intensity* - Opacity of the Scatter effect.
  - *Exaggeration* - Amount to ramp up the Scatter effect. *0.0* requires the full *Shadow Distance* to achieve full saturation, whereas *1.0* can achieve full saturation anywhere.
  - *Sky Intensity* - Amount to apply Scatter to pixels in the sky.
  - *Resolution* - Resolution of the Scatter effect, relative to the Screen Resolution.
  - *Sample Quality* - Amount of samples to take per pixel, when ray-marching the Lightmap(s).
  - *Blur Samples* - Performs a depth-aware blur of samples This removes the appearance of stippling.
  - *Blur Depth Tollerance* - Controls how much depth discrepency is allowed to blur neighboring texels together.
  - *Animate Dither* - Allow Sunshine to animate the dither pattern. At slower speeds this can give a "dust" like appearance. At higher speeds, it can look like "noise".
  - *Animate Speed* - The speed to animate the dither pattern.
  - *Use Overcast* - Allow the scatter effect to take Overcast Shadows into account.

        - *Inherit From Shadows* - Should the Overcast settings be inherited from the Shadow Settings, or customized. Customizing Overcast Shadows for Scatter can be useful for creating a "dancing light" effect.
- **Custom Shadow Mapping** - Replaces the Built-in shadows with Sunshine's.
    - *Forward Renderer* - Tools to assist in swapping out shaders for those using the Forward Renderer.
        - *Install Forward Shaders* - Scans through all Materials in your project, and attempts to point *.shader* to a Sunshine-supported replacement.
            - This is **Required** for Sunshine's Forward Shadows.
            - This is **Not Required** for Volumetric Scattering.
        - *Uninstall Forward Shaders* - Scans through all Materials in your project, and attempts to point *.shader* to an appropriate built-in shader.
        - *Manual Installation* - Bypasses the *Install/Uninstall* process, allowing you to manually swap out Shaders in your Materials.
        - *Force Uninstall* - Forces an Uninstall, even though Sunshine isn't installed. This is only here for piece-of-mind.
    - *Overcast Texture* - A Light Cookie that can be used to simulate the effect of clouds on sunlight.
        - *Scale* - The physical size the Overcast Texture should cover.
        - *Plane Height* - The height of a virtual Overcast Plane. This setting comes in handy if your Directional Light rotates (ie. day/night cycles).
        - *Movement* - The speed to scroll the Overcast Texture, to create a wind-like effect.
    - *Fade Ratio* - Amount of the total Shadow Distance to fade out. This helps ease the transition between realtime and baked shadows.
    - *Terrain LOD Tweak* - Attempts to aleviate self-shadowing artifacts on Terrain by increasing LOD bias.
    - *Shader Set* - Tells Sunshine if you're using the "Mobile" or "Desktop" Shaders. Mobile Shaders require SM2.0, while Desktop Shaders require SM3.0. If you select *Auto*, Sunshine decides based on the target plaform. It is recommended that you manually configure this setting.

- **Visualize Lightmap** - Displays a preview of the current Lightmap.

Deferred Shadows

Sunshine's Custom Shadows integrate with the Deferred Renderer out of the box! There is no need to install or edit Shaders, and everything should *Just Work*™. To get started, make sure you **restart the Unity Editor** to force Unity to load the new deferred shader.

Caveats

- *Receivers* - You cannot control which objects receive shadows.
- *Filters* - The maximum supported *Shadow Filter* is *PCF 3x3*.
- *Multiple Directional Lights* - If you require additional Directional Lights, you must give them a *Light Cookie* to disable Sunshine's Shadows from being mis-applied to them. Unfortunately, this appears to be the only way to differentiate lights in Unity.
- *Forward Shaders* - *Avoid* using Sunshine Forward Shaders in the Deferred Renderer! It can look strange, and degrade performance. If they're installed, simply *Uninstall* them. If you need Forward Rendered shadows on specific objects (ie. transparent objects), then use the *Manual Installation* mode.

Forward Shadows

Sunshine's Custom Shadows *require* a custom set of Shaders. Built-in replacements are provided, but you will need to edit any custom shaders you are using if you want them to receive shadows.

To use Forward Shadows:

- Select the Sunshine Prefab in the Hierarchy
- Press the *Install Forward Shaders* if you haven't already.

Shadow Receivers

Controlling which objects receive shadows can be important for performance, especially on Mobile. There are a couple ways to accomplish this.

- *Manual Installation* (Recommended) - By using this setting (before or after installation), you can edit the Shaders your Materials use without Sunshine automatically "correcting" them on import. Because only Sunshine Shaders can receieve shadows, you can prevent a Material from receieving shadows simply by using a regular shader.
- *SunshineRender Component* (**Unity 4.1+**) - If you attach this component to an object using a Renderer, Sunshine will be able to respect the *Receive Shadows* option at runtime.

Trees

Modified Nature Shaders are included, but because Tree Creator Trees **Generate** their final Materials, you will have to right click on your Tree assets and select *Reimport*. Sunshine will automatically swap out the Shaders used by these Materials as they are imported.

Terrain

If you're using the standard Diffuse Terrain shader, everything should just work!

To use a custom shader, your Terrain objects must have a Material assigned to them (Unity 4.0+). Simply select the Terrain in the Hierarchy, press the *Settings* icon, then select a Material. Basic Terrain Materials have been included for your convenience named *Sunshine Terrain Diffuse* and *Sunshine Terrain Bumped Specular*

Customizing Shaders

If you are using custom Shaders, they will require editing to work with Sunshine's Forward Shadows.

Here is the basic process:

- Duplicate the Shader. This allows your original, non-Sunshine version to become the *Fallback*.
- Place the duplicate Shader in *Sunshine/Shaders/Custom/*
- Edit the duplicate, using the built-in Diffuse Shader as a guide:

```
Shader "Sunshine/Diffuse" { // Prefix the name with "Sunshine/"
Properties {
  _Color ("Main Color", Color) = (1,1,1,1)
  _MainTex ("Base (RGB)", 2D) = "white" {}
}
SubShader {
  Tags { "RenderType"="Opaque" }
  LOD 200
CGPROGRAM

// Include Sunshine functionality:
#include "Assets/Sunshine/Shaders/Sunshine.cginc"

// Multi-compile for different filters:
#pragma multi_compile SUNSHINE_DISABLED SUNSHINE_FILTER_PCF_4x4
SUNSHINE_FILTER_PCF_3x3 SUNSHINE_FILTER_PCF_2x2 SUNSHINE_FILTER_HARD

// Require SM 3.0 on Desktop platforms:
#pragma target 3.0

// Use the sunshine_surf_vert modifier, and exclude prepass to cut waste.
#pragma surface surf Lambert vertex:sunshine_surf_vert exclude_path:prepass

sampler2D _MainTex;
fixed4 _Color;

struct Input {
  float2 uv_MainTex;
  // Required Sunshine Params:
  SUNSHINE_INPUT_PARAMS;
};

// Generates sunshine_surf_vert modifier:
SUNSHINE_SURFACE_VERT(Input)

void surf (Input IN, inout SurfaceOutput o) {
  fixed4 c = tex2D(_MainTex, IN.uv_MainTex) * _Color;
  o.Albedo = c.rgb;
  o.Alpha = c.a;
}
ENDCG
}

// Fallback to original shader:
Fallback "Diffuse"
}
```

- For more examples (such as non-surface Shaders), browse through the *Sunshine/Shaders/Examples/* folder.
- To automatically point existing materials to the new shader, you simply need to re-import those materials.

Troubleshooting Shaders

- If you run out of Shader instructions, remove higher-kernel filters like *SUNSHINE_FILTER_PCF_4x4* from the *multi_compile* line.
- If you are customizing a Mobile Shader, use the following code insead:

```
#define SUNSHINE_MOBILE
#include "Assets/Sunshine/Shaders/Sunshine.cginc"
#pragma multi_compile SUNSHINE_FILTER_PCF_2x2 SUNSHINE_FILTER_HARD
```

  Remove the *#pragma target 3.0* line if you have added it.
- If you are already using a Vertex Modifier, sunshine can piggyback on it

```
SUNSHINE_SURFACE_VERT_PIGGYBACK(my_custom_vertex_modifier, Input) //"Input" should be the name of your Input struct
```

  Be sure to include *vertex:sunshine_surf_vert* in your *#pragma surface* line.

Mobile Overcast Shadows

By default, Mobile Shaders have Overcast Shadows disabled. It is possible to enable Overcast Shadows by defining *SUNSHINE_OVERCAST_ON* before importing *Sunshine.cginc*. However, this will likely exceed the shader instruction limit on anything more complicated than *Mobile Diffuse*. If Overcast shadows are critical to your project, you can try compiling for Hard Shadows (*SUNSHINE_FILTER_HARD*) only to save instructions.

Mobile Shadow Performance

Sunshine brings shadows to unsupported platforms like Tegra 3 devices (Ouya, Nexus 7, etc)... However, this means we must pay **very close** attention to performance issues like **fill-rate** on these devices! Here are some tips that should help you get the most out of Sunshine on Mobile:

Update Interval

Under *Advanced Settings* there is a useful option called *Update Interval* that allows you to control how often the Lightmap is refreshed.

- If you need dynamic objects/characters to receive shadows from static geometry, consider updating *After X Movement*... This way, your Lightmap will only refresh when the camera has moved a set distance!
- If you need fully dynamic shadows, consider updating *After X Frames* and setting the value to *2*. This will refresh your Lightmap every other frame.

Additional Tips

- Use *Hard Shadows* if possible. PCF filtering is expensive on Mobile.
- Use as few *Shadow Casters* as possible by using the *Occluders* setting. For most games, the safe bet will be to **only cast Character Shadows**. Other objects should use baked lightmaps.
- Use as few *Shadow Receivers* as possible. You can do this by switching *Forward Shadows* into *Manual Installation* and only using Sunshine Shaders on Materials you wish to *Receive* shadows.
- Never allow large *"Floor" objects* to occlude light! Aside from being pointless, this will **obliterate FPS on devices like Ouya**.
- Avoid Light Scatter and Image Effects in general.

Oculus Rift and VR Support

Sunshine provides a wonderful optimization for VR Games, by allowing you to share Lightmaps between 2 Eye cameras!
To use this feature, specify a *Stereoscopic Master Camera* on one of your two SunshineCamera instances. For example, you could set the Right Eye camera to be the master of the Left Eye camera. Sunshine will take care of the rest.

Troubleshooting

- **iOS:** If Light Scatter is not showing up, try one of the following solutions.
  - Try to **Disable Blurring** on the Light Scatter, and set them to *Full Resolution*. Enabling *Animate Noise* can improve the appearance.
  - Try setting the *Target Resolution* to *Native* instead of *Auto*.

Class Reference

- *Sunshine* - The main class. There should be only **one instance** in your Scene. Simply drag the Sunshine Prefab into your Scene and you're good-to-go.
- *SunshineCamera* - This class must be attached to any camera that needs to render Sunshine *Shadows*, or *Light Scatter*
  - *Stereoscopic Master Camera* - This setting allows you to share Lightmaps between eyes in VR/Stereoscopic games. Simply set one Eye Camera to be the Master of the other.
  - *RequestRefresh()* - Requests the Lightmap be updated for this camera. This is only useful in conjunction with *Update Interval*.
- *SunshinePostprocess* - This class is **automatically attached** to cameras using Light Scatter. By changing the *Script Execution Order*, you can control when Light Scatter is applied relative to other effects.
- *SunshineRenderer* (**Unity 4.1+**) - This is a **completely optional** class for use in the *Forward Renderer*. You can attach it to objects with a Renderer, and it will respect the "Receive Shadows" option at runtime.

Uninstalling

Uninstalling Sunshine is very simple:

- Select the Sunshine Prefab in your Hierarchy, or the Project Pane.
- Press "Uninstall"
- If you are using Tree Creator Trees, you will need to Re-Import these so that the Shader references are recreated.
- Delete Sunshine from your project.

Technical Details

Sunshine employs some unique techniques of my own design that you may find interesting... It is mathmatically magnificent!

Shadow Cascades

Sunshine can calculate the shadow coordinates of 4 different cascades per-pixel without any matrix transformations! What!?! It does this by pre-transforming the *Camera Position*, and *Frustum Rays* into *Light Space* on the CPU. A 2D *Scale* and *Offset* term is also calculated for each cascade. In the Pixel Shader, all that is required to switch cascade coordinates is a single MAD instruction! This is only possible because Directional Light matricies can be both *Parallel* and *Orthogonal*. Why has nobody thought of this before?

Light Scatter

Sunshine uses ray marching to scan through one or more shadow cascades and estimate the contribution of light scattering on the scene. Normally this would be prohibitively expensive, but by pre-calculating the rays per vertex, and jittering the sample offsets (not unlike SSAO), we can achieve a plausable light scatter effect in realtime! w00t!

Support

If you need help for whatever reason, just let me know in the Official Support Thread.
I can also be reached on Twitter. For more information, visit UnitySunshine.com.