ShaderBox v1

Advanced light shader overview

Features

- Blending of real-time shadows with near and far Beast light maps.
- Realistic shading of objects in a shadow.
- One shader for two pipelines: with light map and without light map.
- Shader for imitating precalculated Global Illumination and color bleeding.
- Specular, Self-Illumination and Transparent properties of shaders are understood by Beast when baking lightmaps. Light that is baked into a lightmap can change its color after passing through a transparent object.
- Toolset for calculating light probes for characters.
- A set of Blinn-Phong Light, Specular and Reflection mask, Reflection CubeMap, Normal map, Detail map shaders.

Future

- Shaders for imitating glass, cloth, self- illumination.
- Advanced shaders for SM3.0 (fresnel, reflection, realtime reflection, refraction).
- Toolset improvements for editing grids and probes.
- Interpolating light values from several probes.
- Improvement of probe grid generation algorithm.
- Adding Spherical Harmonic algorithm for calculating light probes.
- Hierarchical structure of Light Probes.
- Integration with native Light Probes when Unity 3.5 is released.
- Shaders for vegetation (trees, grass).
- Shaders for decals.
- Post-effect Screen glare, Screen dirt.(a-la Battlefield3).
- Flares for fake light sources.

This pack contains a set of shaders for imitation of complex lighting for surfaces of static and dynamic objects. With the help of these shaders, you will be able to imitate such surfaces as wood, plastic, metal.

The main difference between these shaders and Standard shaders in Unity Engine is that they perform realistic blending of **Dual Beast** light maps and real-time shadows. As a result, even at close distances we always see soft light maps from static objects and at the same time get real-time shadows from dynamic objects with adjustable sharpness which blend correctly with light map.

Shaders in the pack can be divided into the following types:

The 1st shader type is for opaque and transparent static objects which use **Beast** light map and dynamic objects without light map (basic use for this shader type will be objects in the game world).

The 2nd shader type is only for dynamic objects (characters) that use additional info about lighting from indirect light probes.

Ambient light probe overview

The pack also contains the system for calculating ambient lighting for dynamic objects. The system consists of a script for Unity editor which calculates probes for ambient lighting and scripts for applying light probes to objects in Play mode.

Shaders

Shader model

All shaders work with shader model 2.0 and are targeted for use with forward render pipeline on Mac & PC platform.

Lod, FallBack

The pack consists of 4 base shaders. Every shader has a simplified shader version for fallback. Every base shader has 2 LOD shaders (**Base LOD 400** High, **FallBack LOD 300** Middle, **LOD 100** Low).

Shader parameters

- ## ShaderBox/Opaque Specular/

This shader exists for imitating hard opaque surfaces with different properties. The shader has the following parameters and slots for textures.

Light Map Brightness – changes brightness of far light map.

Specular Color for Light Map – highlight color, used only when calculating lightmaps in Beast. Black color doesn't give any highlights when calculating lightmaps.

Shininess for Light Map – highlight size and intensity, used only when calculating lightmaps. Far left slider position – maximum highlight spread. Far right slider position – minimum highlight spread.

Diffuse Color – sets diffuse color; alpha component sets alpha brightness.

Specular – sets intensity of specular highlight from light sources in the scene.

Gloss – sets the value of specular highlight.

Reflection Power – sets the total reflection power of the material.

Reflection Brightness – sets the total reflection power of the material.

UV Detail Tile – scale of Detail map.

Detail Factor – opacity of Detail map.

UV Detail Normal Tile – scale of Detail Normal map.

Diffuse (RGB) Specular (A) – diffuse color texture; the alpha channel contains a mask for controlling intensity of highlight and cube map reflection.

Normal map – normal map packed in Unity, without B channel.

Reflection Cubemap – environment cube map texture for reflection.

Detail map – additional tile texture for diffuse color.

Detail Normal map – additional tile texture for normal map.

Rim Light map –rim light color.

Rim Light –visibility angle of rim light is controlled by a special texture. From right to left, the angle is from 90 to 0 degrees. White color of texture pixel sets the maximum intensity of rim light.

- ## ShaderBox/Soft Specular Transparent/

All parameters in this shader are similar to ShaderBox/Opaque Specular/ shader, but there's also transparency which is set by alpha channel in Diffuse map texture and alpha component in **Diffuse Color** parameter. Texture mask for specular highlight and reflection is controlled by a separate **Specular & Reflection** parameter and defined by red channel in texture. Objects with this shader don't receive realtime shadows and cast shadows not taking into account transparent parts of the model.

Transparency Color for Light Map – color of the light after passing through a transparent surface, used only when calculating lightmaps in Beast. Dark pixel in the texture doesn't let any light through pass, white pixel lets all light pass through.

- ## ShaderBox/Soft Specular Transparent/Vertex

All parameters in this shader are similar to ShaderBox/Soft Specular Transparent/ shader, but there's also transparency which is set by alpha channel in vertex color and alpha component in **Diffuse Color** parameter. Objects with this shader don't receive realtime shadows and cast shadows not taking into account transparent parts of the model.

- ## ShaderBox/Cutout Specular Transparent/

All parameters in this shader are similar to ShaderBox/Soft Specular Transparent/ shader. Objects with this shader do cast and receive realtime shadows.

Alpha cutoff – transparency threshold.

- ## ShaderBox/Opaque Specular/Self-Illumin/

All parameters in this shader are similar to ShaderBox/Opaque Specular/ shader, but there're also:

Emission for Light Map – object's emission power which is used only when calculating Beast lightmaps.

Illumination (A) – alpha channel of the texture is used as illumination mask. A white pixel means maximum illumination. Black means no illumination at all.

- ## ShaderBox/Soft Specular Transparent/Self-Illumin/

All parameters in this shader are similar to ShaderBox/Soft Specular Transparent/ shader. Objects with this shader don't receive realtime shadows and cast shadows not taking into account transparent parts of the model.

- ## ShaderBox/Probe/

This shader exists to create surfaces with different qualities –such as skin, cloth, metal. The shader has the following parameters and slots for textures.

Diff Brightness – sets the brightness for diffuse texture.

Specular – sets the intensity of specular highlight from light sources in the scene.

Gloss – sets the value of specular highlight.

Rim Factor – imitation of lighting created by a light source positioned behind the object (wrap lighting); the parameter controls the wrap angle.

Rim Power – sets the total intensity of rim light.

Reflection Power – sets the total reflection power of the material.

Reflection Brightness – sets the brightness of cube map reflection.

Diffuse map – diffuse color texture without alpha channel.

Specular (R) Gloss (G) Reflection (B) – here you specify the texture which has specular intensity mask in red channel, highlight size mask in green channel and reflection mask in blue channel. White pixel in the mask increases the intensity, black pixel removes the effect completely.

Normal map – normal map packed in Unity, without B channel.

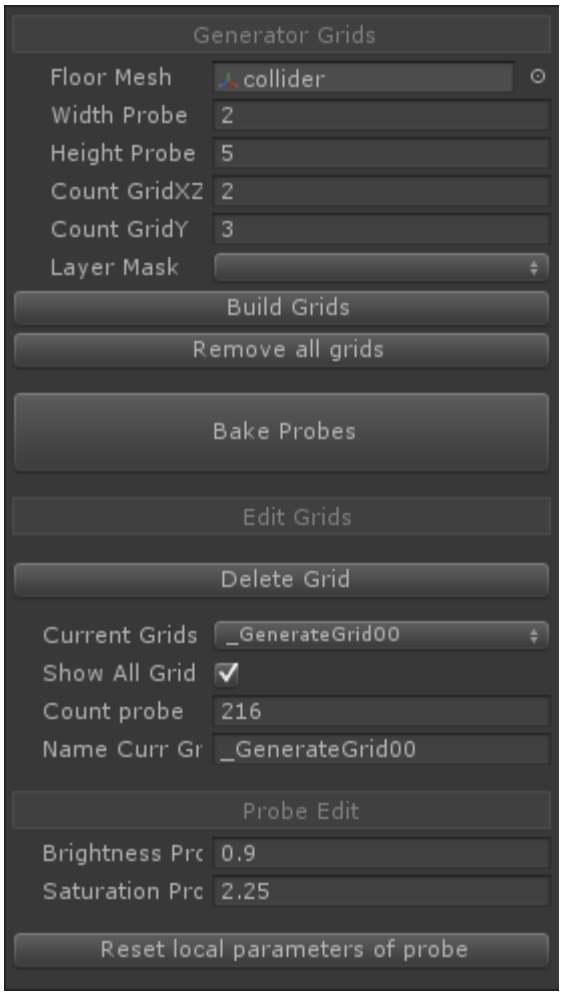
RimColor (RGB) – rim light color.

Reflection Cube map – environment cube map texture for reflection. Alpha channel is used for controlling brightness and contrast of the cube map.

Probe Light

With standard shading, dynamic objects are lit only by direct light sources in the scene and there's absolutely no indirect lighting affecting such model. Light probe system allows to partially solve this problem and improve the visual look of model in terms of quality by using probes of indirect lighting calculated in Unity editor. This method doesn't require any performance intensive calculations in real time. As a result, we get a pretty realistic secondary lighting for dynamic objects practically for free in terms of performance.

To enable **Probe Environment** editor, you need to create a GO with name **ManagerProbsEnv** and put **Manager Probe Environment** script on it. Alternatively you can select **Editor Probe Environment** in **Window** menu. In order to use lighting data from probes, you need to add **Character Probe Environment** script on a dynamic object.



Probe editor component has two sections: **Edit Grids** used to edit the existing grid and **Generator Grids** used to automatically create probe grids based on **Floor Mesh**.

Description of parameters:

Floor Mesh – the mesh based on which probe grid is created. This mesh should follow static horizontal surfaces on which dynamic objects can move. This mesh should be created as a separate object in any 3d package. In order to make it work, you need to add it to the scene, place it into a special "collider" layer, place it into the necessary position and drag'n'drop the mesh into Floor Mesh parameter from Hierarchy window. The mesh is vital for finding probe positions. After setup is complete, the mesh can be deleted or hidden.

Step Cell – horizontal grid step, a distance between probes in unity's coordinates.

Height Cell – vertical grid step, a distance between probes in unity's coordinates.

Count GridXZ – number of splits of **Floor Mesh** along XZ axis.

Count GridY – number of splits of **Floor Mesh** along Z axis**.**

Layer Mask – sets the layer to which **Floor Mesh** is assigned.

Build Grids – probe grid creation based on **Floor Mesh.**

Remove all grids – delete all probes and grids.

Bake Probes – calculate and save lighting for all probes.

Delete Grid – delete the selected probe grid.

Current Grids – select one grid for displaying and editing. If **Show All Grid** is ON, then all grids are displayed.

Show All Grid – show/hide all grids except for the selected one.

Name Current Grid – change the name of the selected grid.

Probe Count – info field setting the total quantity of probes, if **Show All Grid** is ON, or the quantity of probes in the selected grid, if **Show All Grid** is OFF.

Brightness Probe – Probe light's brightness (Brightness Probe **=** 1 Probe light is unchanged).

Saturation Probe – Probe color's saturation (Probe Saturation **=** 1- Probe color is unchanged).

Reset local parameters of probe – button for resetting all local settings of a Probe.

Note:

In order to squeeze even more performance from generated Probes they should be divided into cells - 50 – 100 Probes per cell. This can be done via ***Count GridXZ*** *and* ***Count GridY*** *parameters.*

Layer Mask is used for sampling height during the process of calculating Probe positions. For example if you have a container in your room and you want Probes to be placed on top of this container – you should specify the container in the LayerMask.

Lighting info is saved to **ManagerProbsEnv** object.
Alpha channel component is used for adjusting lighting brightness, the brighter the pixel, the more light the object has. When baking, all dynamic objects need to be hidden, so they don't affect the resulting lighting.

Character lighting is affected by the closest probe. When object moves, the probes which affect it are changing in due course of time, specified in **Character Probe Environment** script (**TimeLerp** parameter). **Shader Character** parameter specifies which shader to send the data to. You need to specify the shader which is going to be used on the character.

Note:

ManagerProbsEnv *can be saved to a prefab and transferred to other scenes or loaded/streamed on demand.*

Setting up selected Light Probe

In order to display and edit light probes in scene view window of Unity Editor, it is necessary to select **ManagerProbsEnv** object in Hierarchy window. Spheres with lighting data are displayed in scene view. Spheres are created when **ManagerProbsEnv** is selected. They are not included into the build. A probe can be selected in scene view window by clicking Right Mouse Button.

Interface:

Copy Probe – copy the selected probe.

Delete Probe – delete the selected probe.

Radius Sphere – gizmo sphere radius used for visual debugging of probe baking result and doesn't affect anything.

Brightness Probe – Probe light's brightness (Brightness Probe **=** 1 Probe light is unchanged).

Saturation Probe – Probe color's saturation (Probe Saturation **=** 1 Probe color is unchanged).

Note:

Brightness and Saturation can be changed individually for each Probe. These values take into account global settings in ***ManagerProbsEnv****. For example if you have global brightness 0.5, and you want to have it doubled for some particular Probe – set the individual setting to 2. If you want to reset Probe settings – press* ***Reset local parameters of probe.***

Tips and Tricks while using ShaderBox

- *Don't forget that shadows affect the color and brightness of Probes. If you turn the shadows off or set shadow distance to 0, then shadows won't affect Probes when baking Beast lightmaps. Sometimes this can be useful if you want to get bright and saturated Probes.*

- *Don't forget that you are able to affect the amount of shadowing of dynamic objects and also affect shadows that are cast from these objects for even more flexible setup for dynamic objects. By changing ambient color you can setup dynamic objects so they look no different from static lightmapped objects.*

- *Don't forget that ambient color affects how bright the baked lightmaps are.*

- *As shaders are made for pixel shader model 2.0 there's no ability to use tiling parameter for diffuse and normal textures. Please change UV coordinates in your favorite 3d package in order to be able to use tiling.*

- *As detail normalmap and detail textures use 1st UV channel, detail size can be uneven and disproportioned on different parts of a model. Try to use an even scale of UV patches or hide such parts of the model.*

Unfortunately, we don't have the time answer all possible questions about usage of ShaderBox. We work hard on updates and our future projects. If you have any questions, please have a look at how things are done in the test scene called "boxroom".

If you find a bug or something that lacks implementation of our package – please send an email to point.shaderbox@gmail.com. We'll try our best to fix the bug or add/improve the feature in the following ShaderBox update.

Updates will follow.
Thank you for using our product!