Unity 5 Lighting: Best Practices for Efficiency

by Anish Dhesikan

Co-Founder | Lead Programmer | Lead Designer at Wise Monk Games, LLC.

[This article is meant to be viewed on a website]

OVERVIEW

This article is critical for developers who have intermediate to advanced experience with Unity and are seeking advice on **how to improve the efficiency, or performance, of the lighting in their games**.

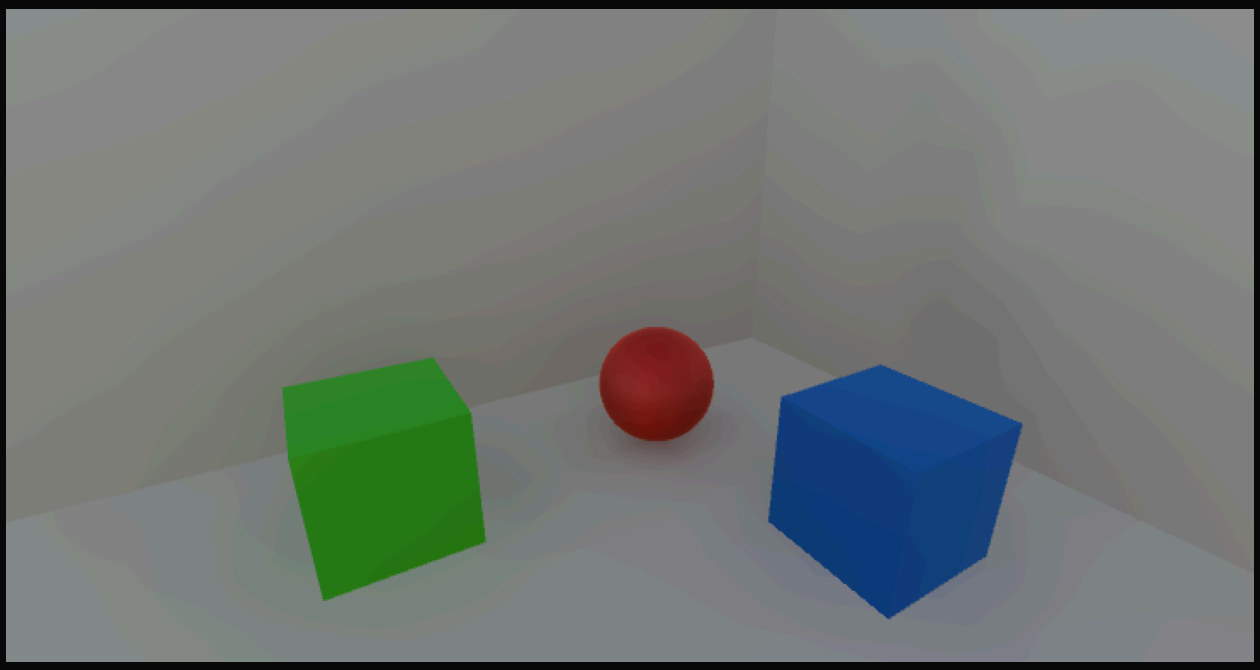
Lighting, specifically lightmapping, has gone through a complete overhaul in Unity 5, with an entirely new system and workflow. In the past, you could place a light in the scene and bake the lighting as textures into static objects with the Beast Lightmapping tool. In Unity 5, Beast has been replaced with Enlighten, which allows for global illumination calculations that feature bouncing light and other subtle lighting computations that result in a more realistic effect. However, there are several new factors that must be considered in lighting scenes in order to maintain performance.

QUICK TIP | Why is lightmapping important?

Without lightmapping, all lights need to be rendered in real time. This results in several computations being done every frame, which results in your games being slow. Lightmapping bakes some of this lighting information into textures or files that make it so lighting doesn't need to be calculated every frame. This severely improves the performance of your game, especially when targeting mobile platforms or less powerful hardware. Let's see how to use Unity's new lighting system to create a scene with efficient lighting.

SETUP

For this example, we will have a scene that is set up with some primitive Unity objects in a small room. There is currently no lighting in this scene, but everything is lit with some ambient lighting.



A small scene of primitive objects with only ambient lighting

Bring up the Lighting window by navigating to Window -> Lighting. You should then see the Lighting panel in your workspace.

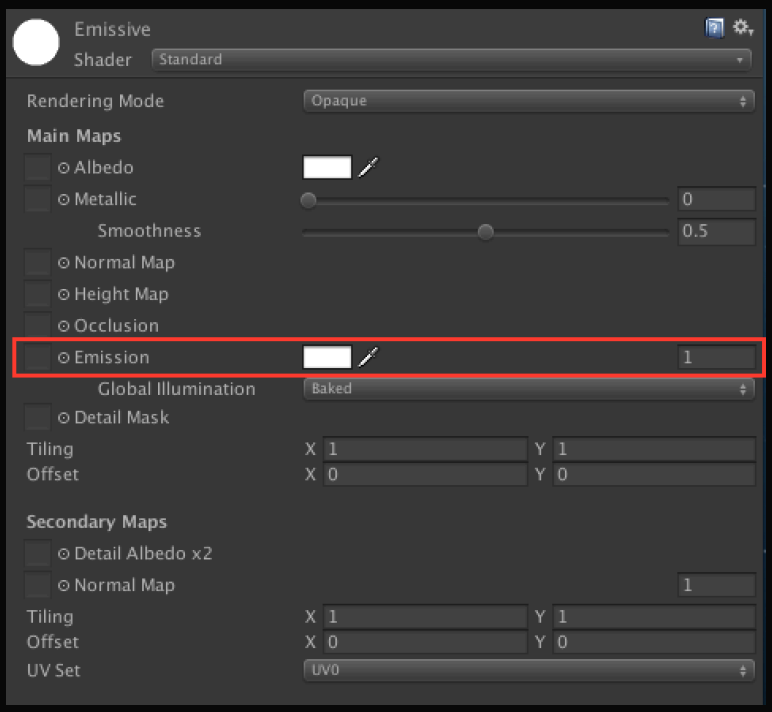
QUICK TIP | Precomputed vs. Baked GI

What is the difference between Precomputed Realtime GI and Baked GI? Precomputed Realtime GI is not as efficient because it will update the illumination if lights move or change during gameplay. Baked GI is more efficient, storing lighting and shadows as textures. For greatest performance, use only Baked GI and mark all objects that will not move as "static" in the scene. However, if any lights will be moving or changing, using Precomputed Realtime GI is sufficient.

LIGHTING THE SCENE

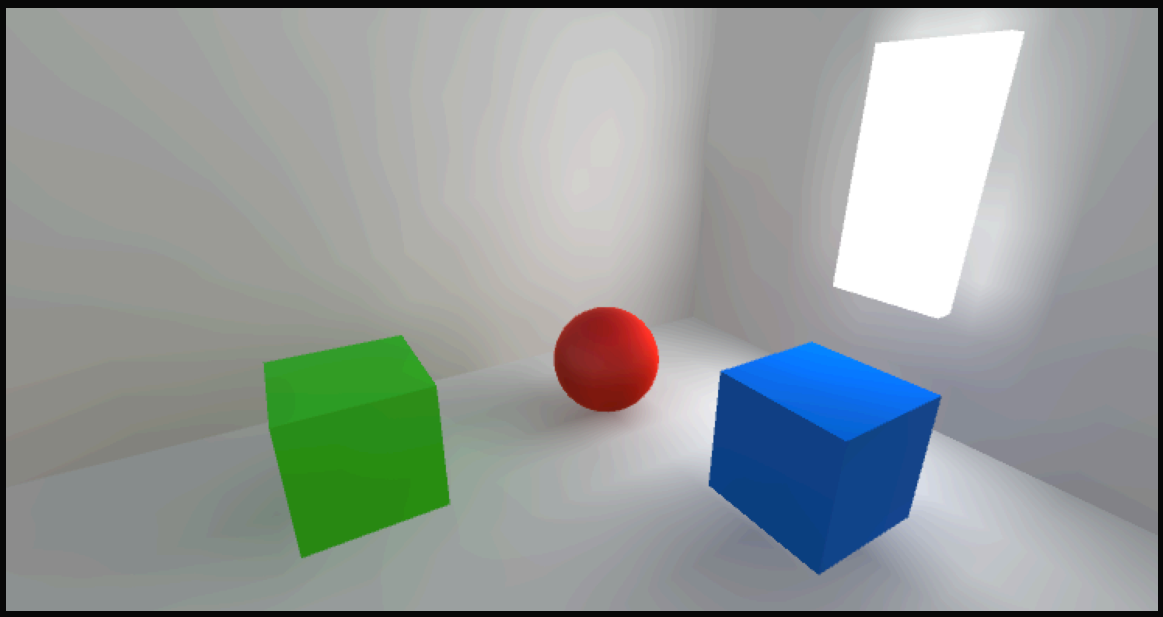
In Unity, objects are lit in 3 ways: lights in the scene, reflection / light probes, and **emissive surfaces**. Lights in the scene are most realistic, but are least efficient if not baked. Reflection / light probes are good for approximating the lighting of objects with minimal performance cost, but will not provide lighting inherently. Emissive surfaces are the best way to simulate lighting with Unity's new system, providing lighting and shadows for static objects with almost no performance cost.

How do you use emissive surfaces? Unity 5 comes with a new standard shader that is used for applying materials to nearly every realistic object. With this standard shader, there is a property known as "Emission."



Unity 5 standard shader with Emission property set to white with a value of 1

To enable emission, set the color and value of the emission property. The higher the emission value, the more light the object will emit. After creating the material, apply it to an object in the scene to turn that object into a light. Mark all objects in the scene that won't move as static, and then proceed to build the lighting.



The scene with an emissive cube on the wall acting as a light after building lighting

As you can see, the object on the wall is casting light onto all the objects in the scene, and shadows are drawn as well. A combination of emissive surfaces and ambient lighting is often enough to get scenes sufficiently lit with minimal performance cost.

Lighting in Unity is certainly one of the largest contributors to slow games. By using fewer lights and more emissive surfaces, you can greatly improve the efficiency of any game you create.

FURTHER READING

For more information on how to improve your game's efficiency, please read the following articles:

- Light Troubleshooting and Performance
- Optimizing Graphics Performance
- General Performance Tips