FastLight2D

A plugin by Mind's Eye Games

Table of Contents

Welcome to FastLight2D!

This is a highly efficient, minimal lighting system that leverages **shaders** to render your lights on the **GPU**.

It theoretically can render all of your lights+shadows in a single draw call (though common setups probably won't achieve that) and it mainly leverages core Unity systems to do so (Shader Graph, Render Textures, Global Shader Values), making it super lightweight.

It has some trade-offs:

- Only point lights are supported!
- It uses an internal shadow map, so it does not have perfect resolution (if you have really large lights, players will probably see some artifacting at shadow edges).
- It requires you to dedicate a layer to ShadowMap mesh objects, and create extra objects that will explicitly cast shadows.
- It only results in rendering lights that have their alpha channel adjusted to give the impression of shadows - more nuanced integration of lighting FX (for example, rendering to a light texture and then using that to modify how other sprites are rendered) is highly game-specific and thus beyond the scope of this asset.

If that's all fine with you, then good luck!

I have prepared a few demo scenes to help explain how to use this asset - if you have questions, please reach out to me on Twitter!

FastLight2D + FastLight2DManager

The primary component of this plugin is **FastLight2DManager** - you MUST have one active in the scene for any of this to work.

Every frame, FastLight2DManager looks at the list of all active FastLight2Ds, and assigns them their place in the **shadow map** for the frame. It then packages the essential data for every active light into Colors and writes them to a Texture so that shaders can retrieve this information on-demand.*

If there are more lights active than the shadow map supports, the light list is sorted and the overflowing lowest-priority lights have their shadows automatically disabled.

FastLight2Ds are assigned a "Y Value" (aka the UV coordinate of a row of pixels in the shadow map) - each pixel in that row corresponds to an angle pointing away from the light, and the color of the pixel indicates how far the light reaches.

THIS IS IMPORTANT! The width of the shadow map directly corresponds with how granular your shadows are. This is configurable - play around with it to see what best suits your game!

Shadow Meshes

The other component of this plugin is the **Shadow Mesh** shaders - they actually compute the shadows!

The shadow mesh(es) **MUST** be on their own dedicated layer, and the camera owned by the FastLight2DManager must render that layer (and nothing else). That camera's depth must also place it before any other cameras that render FastLight2Ds (as their shadows need to be updated for the rendered frame).

The process of rendering shadows is (warning, a bit simplified):

1. The FastLight2DManager sets the camera's target area to cover all active FastLight2Ds
2. The camera clears the shadow map to white (clearing all shadows)
3. All shadow meshes that the camera can see (and thus overlap with the lights) are rendered:
    a. The shadow mesh shader expects the mesh has been defined in a series of quads, each corresponding with a line segment or other part of a shadow caster.
    b. The shadow mesh vertex shader transforms each quad to cover the active area of the shadow map, while sending relevant data to the fragment shader
    c. Each evaluation of the fragment shader corresponds with a pixel in the shadow map (aka a light + specific raycast angle). The fragment shader computes a color that tells the shadow map how far that raycast can extend vs that line segment.
4. The shader merges all the separate results via **BlendOp Min**.

You're free to compute whatever shadow meshes you like, but this plugin only includes one script to do so: Collider2DShadowMesh2D. As the name suggests, its purpose is to take input Collider2Ds and generate a matching shadow mesh.

NOTE: Since the shader is doing all the heavy lifting, this enables shadows to be defined via various math functions! This gives things like nice round shadows, but there is quite a lot of room for customization.