


# Bake Materials

:

There is an experimental feature available that allows you to bake materials into a single material, potentially reducing the number of draw calls even further. Currently, this feature is only compatible with materials based on Universal Render Pipeline (URP) `Lit` or `Simple Lit` shaders. Additionally, the materials should not have any textures applied to them.

If you enable this baking feature, don't worry if you have materials that don't meet these conditions. The combiner will only bake materials that meet the specified conditions and leave other materials unchanged. This ensures that the combining process proceeds smoothly while still accommodating materials that do not meet the conditions for baking.

 Please note that this feature is experimental, and its compatibility and functionality may be subject to change in future updates.

## Dependencies

The current implementation of Universal Render Pipeline (URP) requires shaders that were built using the package `com.unity.shadergraph` in order to render baked materials correctly. If you want to omit this dependency, you have two options:

1. Implement your own implementation: You can create your own custom shaders or rendering solution to handle the rendering of baked materials without relying on `com.unity.shadergraph`.
2. Contact us: If you would like us to prioritize working on a specific implementation that doesn't rely on `com.unity.shadergraph`, please reach out to us. We are open to considering alternative solutions based on your specific needs.

Feel free to choose the option that best suits your requirements and constraints.

## Custom baker

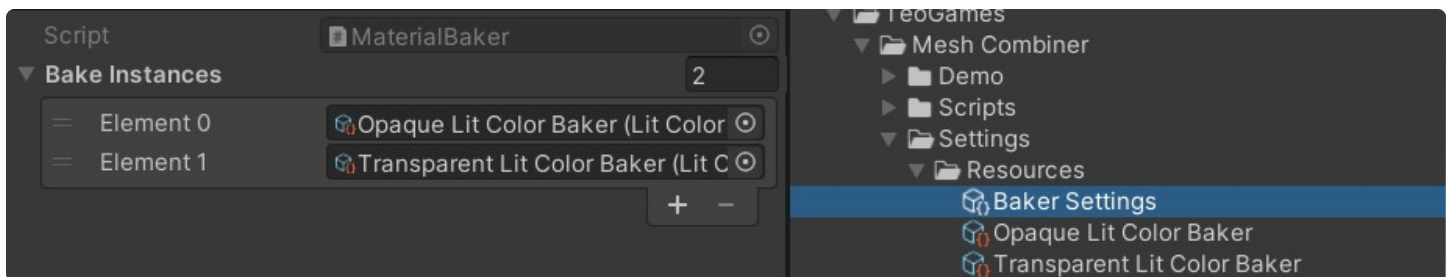
You have the ability to create your own material baker by following these steps:

## Step 1: Create a Custom ScriptableObject

Create a new ScriptableObject that inherits from the `AbstractShaderBaker` class. This will serve as your custom material baker implementation.

## Step 2: Register Your Baker

In the Baker Settings, add your custom material baker to the list of bakers. This ensures that your baker will be registered and used for specific shaders:



## Step 3: Implement Required Functions

### `void Inject(MaterialBaker instance)`

This function is called during startup to register your baker for a specific shader. Within this function, you can perform any necessary setup or configuration for your baker.

### `void Initialize(Material material)`

This function is called during the first bake and acts as a constructor. Here, you can define any necessary variables or perform initialization that will be used throughout the baking process.

### `(bool isBaked, Material bakedMaterial, MeshParser meshParser)`

### `Bake(Material material)`

This is the main function that is called for each material that needs to be baked. Within this function, you will perform the baking process, including baking textures and applying any necessary transformations to fit the mesh in the new textures. The function should return a tuple with three values:

- `isBaked`: A boolean value indicating if the material has been successfully baked. Typically

- `bakedMaterial`: The baked material.
- `meshParser`: A parser that will apply UV transforms to fit the mesh in the new textures.

It is recommended to review the example provided in the `LitColorBaker` class, as it can serve as a helpful reference for understanding the implementation details.

By creating your own custom material baker, you can extend the functionality of the combiner and customize the baking process to suit your specific needs and shaders.