
Driven Decals

A mesh-based PBR decal system for Unity. For use with the Universal Render Pipeline's forward renderer.

There are broadly two approaches to rendering projected decals in real-time graphics: * Generate a projected mesh for each decal instance in the scene. * Dynamically project the decal in view-space using a fragment shader.

Each approach has its strengths and many projects benefit from using both for different situations. For example *Half-Life: Alyx* appears to use projected mesh decals for some static scenery details, and view-space projected decals for dynamic effects like bullet holes. I've written a little interactive article about how decals can be rendered if you would like to know more.

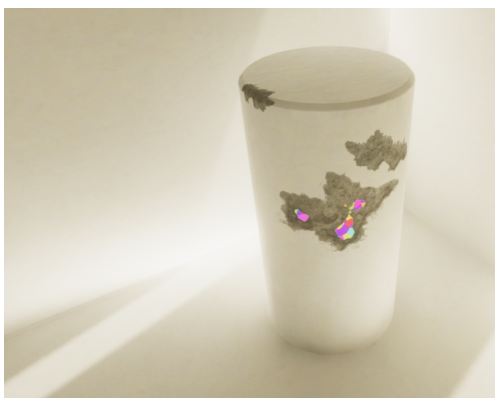
This decal system only deals with creating projected meshes, so you shouldn't expect it to be a complete solution to every decal use case.

60 second introduction video.

Status - 2021-04-06

Hello! This package isn't kidding about having a "preview" version number. You are of course welcome to use it as you like, but please be aware there are known missing features and almost certainly bugs.

I am still working on this and intend to bring it up to release quality. At the moment I'm busy working full-time on projects away from Unity, so progress is not fast. Pull requests are always welcome if you're feeling inspired!



Key Features

- Creates meshes that behave like any other mesh in your scene. Making them easier to work with and use with other features.
- Easy to customise using Unity's Shader Graph.
- Low rendering cost and full compatibility with URP's forward renderer makes it ideal for use in XR.
- Custom inspectors that provide immediate in-editor feedback.
- Support for multi-object editing and undo.

Key Limitations

- Decal mesh generation is relatively slow. This system is not recommended as a way to dynamically place bullet holes or other effects during gameplay.
- Once the decal mesh is generated it remains as it is and will not adapt to the other meshes changing. This system will not work nicely with skinned mesh renderers or other meshes that get distorted at runtime.

Changelog

Human-friendly changelog

Getting Started

Requirements

- Unity 2019.3.0f6 or later, using the universal render pipeline (URP) version 7.2.1 or later.

In theory it should work in the LWRP and Unity versions as far back as 2018. But it looks like shader graph really doesn't care about cross-version compatibility. If you need this to work in those earlier versions I may be able to put together something compatible with some extra work.

Installation

1. Within your project open the Package Manager from *Window → Package Manager*
2. Click the + icon at the top-left of the window and select "Add package from git URL..."
3. Paste in <https://github.com/Anatta336/driven-decals.git> and click "Add"

-
4. Wait a minute. It looks a lot like nothing is happening, but Unity is busy thinking about packages.
 5. When installed you'll see "Driven Decals" listed in the window.

Or you can manually modify your project's `manifest.json` file in the Packages folder to include:

```
1 {  
2   "dependencies": {  
3     "com.samdriver.driven-decals": "https://github.com/Anatta336/driven-  
4       -decals.git",  
5     ...  
6   },  
7 }
```

Documentation

Extensive documentation including step-by-step instructions to get you started.

Authors

Sam Driver - Website, Twitter, PayPal (any support is very gratefully received)

Special Thanks

Kenny5 - Made everything work nicely with the Unity Package Manager.

Licence

The source code of this project and associated documentation is licensed under the MIT licence.

The included example assets are licensed under the Attribution 4.0 International (CC BY 4.0) licence.

These licences mean you already have permission to use this in whatever project you like, including commercial releases. They place no obligation on you to release your source code. If you release something that makes use of this decal system a small acknowledgement in the credits would be appreciated, but is not required.