Pooling

Pooling allows us to recycle temporary decals like bullet-holes or blood, saving the cost of constantly instantiating new decals as well as allowing us to control how many decals we have in active circulation for any given system. Pooling is inbuilt into the included printers and can easily be built into your own decal scripts.

To set up pools for you game start by opening the settings menu (Window>Decals>Settings). From here we can create new pools, rename your pools and specify a limit per quality setting. It's recommended you use one pool per subsystem in your game (ie. one for blood, another for bullet holes etc). The "Blue Goo" and "Orange Goo" pools are used for the Goo demo scene and can be removed at your leisure.

The pool limit allows us to control how many decals the pool will allow before reusing old ones, limiting the cost of the system. If unused decals are available, these will always be used before instantiating new decals. If the pool limit is reached, the pool will grab the oldest active decal and use this instead. This means active and visible decals can be reallocated and will appear to pop out of view. To avoid this fade your old decals out over time or when the player isn't looking using the Fade and Cull components respectively. This ensures there are always inactive decals available for the pool to use.

Scripting

Scripting with the pooling system is easy. To request new decals we first get a reference to our pool, then request decals as required, specifying an example (Usually a prefab) for the pooling system to return a copy of. You can also optionally request it copy behaviors attached to the example decal. This is useful for retaining Fade and Cull components. This would look something like :

```
ProjectionPool myPool = ProjectionPool.GetPool("myPool");
ProjectionRenderer myNewDecal = myPool.Request(myDecalPrefab);
```

where "myPool" would be the name of the pool we created in the settings menu. When using pooling, it's important to return decals to the pool instead of destroying them like you would other GameObjects. This can be done by calling the ProjectionRenderer's destroy method instead of the default GameObject.Destroy. ie :

```
myNewDecal.Destroy();
```

You'll rarely need to script with the system, but the option is available to you if you decide to build your own custom printers/decal spawners. It's also important to note, this pooling system is entirely optional and built for the included printers. There's no reason you can't build your own pooling system or simply instantiate and destroy decals at your leisure.